Thèse de Doctorat

de l'Université Sorbonne Paris Cité
Préparée à l'Université Paris Diderot

École Doctorale Sciences Mathématiques de Paris Centre (ED386)

Institut de Recherche en Informatique Fondamentale (IRIF)

Spécialité : Informatique

Usability: low tech, high security

Par :

Nikola K. Blanchard

Dirigée par :

Nicolas Schabanel, CNRS, LIP et IXXI, ENS de Lyon
Ted Selker, University of Maryland, Baltimore County

Version corrigée après soutenance, indexée le 19 Juillet 2019

Soutenue publiquement le 21 juin 2019 à Paris devant un jury constitué de:

Nicolas SCHABANEL	DR	École normale supérieure de Lyon, CNRS	*Co-directeur de thèse*
Ted SELKER	PU	University of Maryland, Baltimore County	*Co-directeur de thèse*
David NACCACHE	PU	École normale supérieure de Paris	*Rapporteur*
Michelle MAZUREK	MdC	University of Maryland, College Park	*Rapporteuse*
Peter Y.A. RYAN	PU	Université du Luxembourg	*Rapporteur*
Adrian KOSOWSKI	CR	Université Paris Diderot, INRIA	*Examinateur*
Marine MINIER	PU	Université de Lorraine	*Présidente du jury*

Utilisabilité : haute sécurité en basse technologie

Résumé

Cette thèse est consacrée au domaine de l'utilisabilité de la sécurité, en particulier dans le contexte de l'authentification en ligne et du vote vérifiable.

Le rôle toujours plus important de nos identifiants en ligne, qu'ils soient utilisés pour accéder aux réseaux sociaux, aux services bancaires ou aux systèmes de vote, a débouché sur des solutions faisant plus de mal que de bien. Le problème n'est pas juste technique mais a une forte composante psycho-sociale, qui se révèle dans l'usage des mots de passe — objet central d'étude de cette thèse. Les utilisateurs font quotidiennement face à des compromis, souvent inconscients, entre sécuriser leurs données et dépenser les ressources mentales limitées et déjà trop sollicitées. Des travaux récents ont montré que l'absence de règles communes, les contraintes ad-hoc si fréquentes et les recommandations contradictoires compliquent ce choix, mais ces recherches sont généralement ignorées, victimes d'une probable incompréhension entre chercheurs, développeurs et utilisateurs.

Cette thèse vise à résoudre ces problèmes avec des solutions inspirées par la cryptographie, la psychologie, ainsi que sept études utilisateurs, afin d'obtenir des outils simplifiés non seulement pour l'utilisateur final mais aussi pour le développeur.

La première partie des contributions se concentre sur le fournisseur de service, avec deux outils permettant d'améliorer l'expérience utilisateur sans effort de sa part. Nous commençons par une étude sur la facilité de transcription de différents types de codes, afin d'obtenir un design réduisant les erreurs tout en augmentant la vitesse de frappe. Nous montrons aussi comment accepter les fautes de frappe dans les mots de passe peut améliorer la sécurité, en offrant un protocole compatible avec les méthodes de hachage standard.

La deuxième partie offre des outils directement aux utilisateurs, avec un gestionnaire de mot de passe mental qui ne nécessite que la mémorisation d'une phrase et d'un code PIN, avec des garanties sur la sécurité des mots de passe si certains sont compromis. Nous proposons aussi une méthode de création de phrase de passe à la fois plus facile et sécurisée, et terminons en montrant empiriquement des failles dans le principal modèle de calcul mental utilisé aujourd'hui dans le domaine.

Enfin, nous nous consacrons aux nouveaux protocoles de vote, en commençant par les difficultés à les faire accepter en pratique. Nous répondons à une demande pour des systèmes non-électroniques en proposant plusieurs implémentations de vote vérifiable en papier, une panoplie de primitives et un protocole de vote pour les très petites élections.

Mots-clefs: Utilisabilité de la sécurité, authentification, protocoles de vote, mots de passe, vote vérifiable, psychométrie

Usability: low tech, high security

Abstract

This dissertation deals with the field of usable security, particularly in the contexts of online authentication and verifiable voting systems.

The ever-expanding role of online accounts in our lives, from social networks to banking or online voting, has led to some initially counterproductive solutions. As recent research has shown, the problem is not just technical but has a very real psychosocial component. Password-based authentication, the subject of most of this thesis, is intrinsically linked to the unconscious mechanisms people use when interacting with security systems. Everyday, users face trade-offs between protecting their security and spending valuable mental resources, with a choice made harder by conflicting recommendations, a lack of standards, and the ad-hoc constraints still frequently encountered. Moreover, as recent results from usable security are often ignored, the problem might stem from a fundamental disconnect between the users, the developers and the researchers.

We try to address those problems with solutions that are not only simplified for the user's sake but also for the developer's. To this end, we use tools from cryptography and psychology, and report on seven usability experiments.

The first part of the contributions uses a service provider's point of view, with two tools to improve the end-user's experience without requiring their cooperation. We start by analysing how easily codes of different structures can be transcribed, with a proposal that reduces error rates while increasing speed. We then look at how servers can accept typos in passwords without changing the general hashing protocol, and how this could improve security.

The second part focuses on end-users, starting by a proposed mental password manager that only depends on remembering only a single passphrase and PIN, with guarantees on the mutual security of generated passwords if some get stolen. We also provide a better way to create such passphrases. As mental computing models are central to expanding this field, we finish by empirically showing why the main model used today is not adapted to the purpose.

In the third part, we focus on voting protocols, and investigate why changing the ones used in practice is an uphill battle. We try to answer a demand for simple paper-based systems by providing low-tech versions of the first paper-based verifiable voting scheme. To conclude, we propose a set of low-tech primitives combined in a protocol that allows usable verifiable voting with no electronic means in small elections.

Keywords: Usability of security, authentication, voting protocols, passwords, verifiable voting, psychometrics

Foreword

To the families we found

This thesis is not the product of four years of hard labour. A copious amount of effort was involved, but if that had been all, the result would not bear any resemblance to this work. Instead, it is the product of a most serendipitous random walk, from a quiet pub in Kosovo to a Basque palace, from a smoky cinema in Sarajevo where we debated Australian voting and Macronist policy to the hemicycle of the Council of Europe, from a loud discotheque filled with exhausted activists to a busy room of young cybersecurity scholars in Baltimore. It started with a non-trivial theoretical problem — dynamic clustering — before moving to a quasi-impossible one — the point location problem — although I naturally did not know that to be the case when it would have been most relevant. It began with an advisor who decided that letting their student spend six months studying classical Greek texts was not unwise during a PhD supposed to be in discrete mathematics. It was focused by another advisor with interests wider than my own.

Behind every decent text, there should be a story, if only because we are wired to understand subjects better through a narrative process. Each chapter will strive to tell a tale, building a bridge between the initial idea, the results, and their interpretation. That said, the existence of this collection has its own story, which is strangely dependent on many random events — other events would have led to a different story, for sure, but some stories had a much higher probability. The obvious story would have followed the initial tracks: after three years of studying mathematics and theoretical computer science at ENS, I had started an internship which led into a PhD on dynamic clustering. That could have lasted for the four years of funding I had, leading to a probably mediocre result (considering the way it was going after two years). Instead, the following happened, starting slow and becoming increasingly chaotic.

After a year working with my advisor Nicolas Schabanel — himself a student of my previous advisor at ENS, Claire Mathieu, to whom I owe quite a lot — we had some decent results and were looking further into the initial topic when a friend told me about random juries (as defended by a peculiar French public figure, Étienne Chouard). After thinking about the concept, I sat down to work on my own ideas for a few days, enjoying

the intellectual stimulation before looking online (I was only familiar with one previous work on the subject: the Three-Ballot idea invented by Ronald Rivest, which I was quite fond of). I ended up with a system that shared many objectives with what I then realised already existed: Random Sample Voting (or RSV). As could be expected, my idea was inferior in every metric I could think of, except one: the idea of decoy ballots, meant to prevent vote buying. This idea happened to be directly transferable but was not in the white paper that one could find online, so I tentatively approached a random member of the group on the 9th of November, 2015, and received no response. Twelve days later, I decided to email the original creator, David Chaum. Although I was quite intimidated, it was a rational calculus: I was no-one, and any inanity I proffered would be quickly forgotten, but an eventual speck of gold in it could lead to good things.

I received a one-line message the next morning, and I learned during the immediately ensuing video chat that the idea had been already thought of, but not put online as yet. This ordeal was followed a while later by a second even more intimidating call with both David Chaum and Deborah Hurley, who, amused by my interest, invited me to join the project — no doubt affected by the unending appeal of free work from graduate students.

Little happened in the next nine months, and this side project deserved its categorisation as such until I got a call from David, who asked me to organise the first public demonstration of RSV to non-cryptographers. Although I was the most junior and least experienced member of a team of more than thirty, I had one great advantage: living less than a thousand kilometres from where the conference was organised, at a palace in the middle of the Basque Country. This demo was not a success, although it was by no means a failure, as it still worked but raised many issues, mostly regarding usability. Besides that, one great thing came out of it for me: a collaboration leading to a close friendship with Géza Tessényi. To be fair, what amazed me initially in this strange character was not the wits he deployed during our panel together — for which I have acquired great respect since — but the ability of a sixty year old Hungarian legal scholar to dance like Travolta for more than three hours in the discotheque where the last night of the conference was organised.

Meanwhile, I had gone to teach in Kosovo with my friend and colleague Siargey Kachanovich, and developed a lasting appreciation for this country. Two events from the trip had lasting consequences: first, the electric scooter I used to go around broke down. Someone in front of the building where I taught graciously offered to help, and we opened the beast, getting our hands dirty together. I learned only a while later — and from someone else — that the random passerby was himself a solid mathematician, named Qëndrim Gashi, who had just been selected as Kosovo's ambassador to France. Staying in contact with him and following the events there shaped some of my opinions as to the best places to focus on when implementing large-scale changes on voting and political systems. The second event was crossing paths with the geographer Jacques Lévy in a restaurant and following this up with a long discussion in a pub, as we were among the forty or so French people in the capital and had no plans for the evening. We realised that we shared similar interests and many common questions as to how people interact but were attacking them from very different perspectives. After his early half-hearted attempt to poach me into pursuing a PhD in his field, this was to develop into another friendship where I learned a lot, making me join Chôros, his "rhizome" or network of scholars, and leading to discussions without which this work would not have been possible.

Getting back to RSV, I lobbied for improved usability with limited efficacy — and,

to be honest, effort — for close to a year, before an opportunity appeared, coming from Géza (this is the second dependent event in the chain). An exciting event, the World Forum for Democracy, was planned for the end of 2017, involving a few thousand people ranging from performance artists to ex-heads of states. Géza, having previously worked with the organisers at the Council of Europe, alerted me to the call for participation. I ended up trying to find a compromise between the interests of the RSV project, the organisers, and the other voting group they had thought to bring in, for the purpose of having the conference attendees vote on the official recommendations. The organisers had, alas, forgotten that the old saying on watches — whoever has one knows the time; whoever has two never does — also applied to public votes: organising multiple public consultations at the same time comes with great risk for little gain.

This gave urgency to the usability matter, at which point a member of the RSV Project, Alan Sherman, thought to bring in his old college buddy, Ted Selker. As I had been the one complaining, I was tasked with informing him of the issues we faced and working with him. Although it could have ended there in August 2017, we both got interested in one of the problems previously raised: making codes that are easily transcribed. As I was going to a convention with a few thousand people just a week after our first discussions — where I could find many willing guinea pigs — we decided to set up a quick experiment. This was built on enthusiasm, sleep deprivation, and a pile of horrid JavaScript, forming a fitting basis for all the research presented in this thesis.

Less than a month later, we had empirical results from not only from the convention pilot study but also from a follow-up study, and a first paper was written — although it wouldn't get published before quite some time. We had also raised more questions than we could have imagined. We resumed our work soon after the original voting experiment we had met over, expanding from vote codes to passwords, always with the idea of getting back to voting usability, something we wouldn't do for nearly one more year. In the meantime, and less than a year before I wrote this paragraph, we realised that he was de facto my advisor and scrambled to make this situation official.

Until September 2018, I had no guiding narrative behind my work. Thanks to Nicolas Schabanel, I had used my three previous years to work on quite a variety of subjects, sometimes productively, sometimes in vain. I hope that most advisors in his field would let a student spend a while working on computational geometry or graph algorithms for mapping Pluto, as those are reasonably close projects. I do, however, wonder how many would let their student spend six months writing a book about the role of randomness in political institutions or researching the intersection of queer and crip theory. Despite his concerns, he did, and for that, I will be eternally grateful. The one drawback is that I had a collection of works which was impossible to make sense of. The trick I used was eliminating half of it, and finding what kind of results would be needed to bring the rest together, leaving me with a general plan, but quite a few things to do before the end result would make sense. Six months of non-stop thinking and writing ensued, to get the missing links between the different chapters that follow.

Through my own readings, observations, and discussions with colleagues from the more human sciences, I had come to believe that the psychological and socio-technical aspects of voting and security are not just important but indeed primordial if we want our work to have an impact on the external world. As such, there is one overarching goal in this thesis: simplicity. Unless the idea is simple, and can easily be understood and implemented, it has few chances to have a positive impact, which is the objective. Of course, despite simplicity

being the goal, it is not always achieved, and that should only be counted as a failure on my part.

To conclude, I would like to once again thank Nicolas Schabanel for following me down the rabbit hole I went into and helping me anytime I needed him. I am grateful to Ted Selker, one of the most creative minds I've had the opportunity to interact with, not only as a researcher but as a fellow human, for taking me under his wing and fighting administrative hell to teach his vision of the world to someone halfway accross the globe.

I am also lucky for the many opportunities I've had to exchange ideas and discuss research with the following people: Michel Habib, Laurent Viennot, Adrian Kosowski, Pierre Charbit, Eugène Asarin, Juliusz Chroboczek, Étienne Mallet, and Laifa Ahmadi from my lab IRIF; David Chaum, Alan Sherman, Deborah Hurley, Hannu Nurmi, and Chris Vatcher from the RSV Project; Jacques Lévy and Patrick Poncet from Chôros; Ingolf Pernice and Kai Gärtner from HIIG; Herrade Igersheim from BETA, Fabien Mathieu and Marc-Olivier Buob from LINCS; Oded Lachish from Birkbeck, Nitzan Gur-Furman from Google and Oz Gur-Furman from Cybereason, Irène Gannaz from INSA Lyon, Yehuda Porath from OECD, Arthur Milchior from ULBruxelles, Robin Sultana from IPAG, Xavier Coquand from Bsecure, Alice Jacquot from Paris XI, Sunimal Mendis from CEIPI, and finally the five student interns I've supervised: Élodie Decerle, Amira Lakhdar, Ines Dardouri, Pierre Midavaine, and François Gaudré. Although I've never had the opportunity to meet her, this work would also not have been possible without Alexandra Elbakian. It would also not have been possible without the 550 anonymous volunteers who finished one of the five user studies we managed to organise without any funding, as well as the 154 who participated in our voting experiments.

I should also mention the various scholars I've met and worked with during my travels: Géza Tessényi, Josiah Ober, Enis Golaszewski, Matthew Grayson, Eli Sennesh, Noson Yanovsky, and Mark Mirmelstein, as well as the colleagues with whom I shared adventures while furthering our dissemination of science goals: Christian Duhamel, Anne-Claire Wang, Liubov Tupikina, and Martin Andler.

I have to thank two people who have been mentors and guides for close to half my life: Andrei Rodin and Michael Wright. Finally, the two members of my family, Dominique and Aurélien, as well as the ten friends and partners who have spent endless nights and days with me over the past few years, variously helping with data processing and interpretation, exchanging and attacking ideas, reading through many versions of my manuscripts, but also bringing me back up when I felt down: Clément Malaingre, Myriam Berlioz, Marc Daviot, Édouard Thomas, Awara Désiré, Olivier Pivot, Siargey Kachanovich, Charlotte Lemaistre, Florentin Waligorski and most of all Leila Gabasova.

Introduction

Six years ago, I attended a cryptography conference where a well-respected researcher announced that the big problems of the field were solved, and all that was left was to correctly implement the solutions already known, although he did not downplay the difficulty of the task. Half a dozen years later, and while I was writing the last words of this thesis, a tool called Evercrypt was released [Har19]. It achieves the goal announced previously by proving the security properties of the implementation through formal verification. It is a big step towards making users secure in their online experience, but sadly the initial framing of the problem might have been wrong: secure implementations of security tools are necessary, but getting people — both users and developers — to actually use those tools is even more critical.

And so, despite the technological advances, we still have weekly reports of stolen credentials [Sha18], not only from start-ups and smaller corporations but also from the biggest companies. Some reach catastrophic proportions, as can attest the discovery in mid-March 2019 that Facebook had stored between 200 and 600 million passwords in clear-text instead of hashed[a] [Kre19], going as far back as 2012. This is not a freak occurrence, as Twitter and GitHub both revealed similar failures to encrypt their confidential information in the previous ten months [Kre18, Whi18].

This failure to follow basic rules and standards — even ones that are decades old — can be seen everywhere in online security, with web services not hashing credentials or using the wrong kind of hashing function[b]. For example, it has been known for at least twelve years that the most common hashing algorithms such as md5, SHA-1 and SHA-256 are vulnerable to offline bruteforce [Pta07], but the Secure Web Application Technologies Checklist still recommended using SHA-256 in 2016 [Rie16] (although they've rectified it since). Other frequent mistakes include using homemade untested cryptographic primitives or biometric methods that are known to be vulnerable [MR18]. The source of these mistakes is no

[a] Facebook revealed that the stored passwords were only accessible to employees — and were accessed by about 2000 of them — but there is, in any case, no good reason why they would store them. Less than a month later, it was revealed that the social network also asked some of its users to provide their login details for their main email addresses, breaching all forms of privacy concerns [Kha19].

[b] In this thesis, all hashing functions considered will be cryptographic hashing functions. To be precise, they will be functions $f : x \to y$ that deterministically map an input to a fixed-length output, in such a way that it is computationally easy to check that $f(x) = y$, but computationally hard to get x when one knows $f(x)$. Moreover, given x, finding x' such that $f(x') = f(x)$ should also be computationally hard.

mystery: the poor communication between the "experts" and those who end up developing the products [AFM16]. Few are blameless, as the communication failures hit every level. Theoretical frameworks tend to require ever more mathematical background and time to understand, making them less accessible to non-theoreticians. Security systems tend to be built on top of older systems, interacting in unforeseen ways and making everything increasingly complex, making them potentially vulnerable [MDAB10, FGNT15] and nearly impossible to understand by anyone but their creators. All this is compounded by a relative lack of continuous training for developers when compared to the speed at which technologies become vulnerable — especially with the workload necessary to keep up to date security-wise. The complexity of rigorous solutions spurs the use of code snippets and other ready-made solutions taken away from their original context, which increases risks [ABF+17]. When it comes to authentication, solutions built ten years ago, such as the secure hashing algorithms mentioned, are still not widely used, as the majority of developers working today finished their training before they were invented. Some herald the death of passwords, and it is true that alternatives such as biometrics are increasingly used. However, the death of passwords has been announced repeatedly since at least 1997 [Poo97, Pot05, CJE+06], with biometrics being considered an alternative since at least 1995 [Kim95].

Naturally, this applies to many fields, and authentication is only one example among many, notable mostly for the scale of the problem and the catastrophic risks associated. Voting is a second field suffering from similar problems. Experts have been reporting problems for decades, regarding both security and usability. For example, the idea that voting software should be open source, to be easily auditable — among other reasons — has been extremely popular among researchers for more than 15 years [Sim04], with many experimental systems deciding to be open source. However, the vast majority of voting machine software is still proprietary, with only the state of New Hampshire having certified open-source voting software in the USA[c] [Tra18]. Moreover, decision makers are seldom really knowledgeable about technical issues, leading to inconsistencies and vulnerabilities. Many theoreticians — but not all [RR06] — on the other hand, focus on improving the technologies without listening to the real problems experienced by voters, and without deploying adequate effort to go from theoretical systems to ones that are used in practice.

Not all end-users suffer from this. Some — rare — people can remember more than 60 different passwords easily, and a few experts have well-configured password managers with fail-safes and redundancies. However, knowing about a problem and its solutions can make us blind to the options faced by others with different expertise. If we consider the three main kinds of actors in the field of security — theoreticians/academics, web service developers, and end users — there is a general problem of limited communication on three fronts. First, the developers seldom listen to their users directly, perhaps because they "know better". In turn, theoreticians don't necessarily listen to developers, and end-users don't have the required background to listen to academics. Knowledge does circulate from end-users to theoreticians, who then transfer it to developers, although with a significant delay. This systemic problem turns out to have very close similarities with a model studied in disability studies.

[c]Besides inertia, market forces and oligopolies play a large role by influencing the decision not to go for open-source systems, as it would limit the revenue of the few major companies developing voting machines.

An intuition from disability studies

Let us take a short detour and consider a similar triangle, where the theoreticians are medical experts and researchers, developers are general practitioners, and the end-users are the rest of the population. We can then observe a surprisingly strong parallel between the issues we observe and the ones that have been explored in the past few decades in the context of disability studies. In this context, an opposition was investigated between two conflicting *models* of disability, the *social model*[d] and the *individual* or *medical model* [Oli13]. In the *medical model*, corresponding to the traditional view, an *impairment* such as having trouble walking or hearing is seen as a disability by itself and the source of ill-being, and the *impairment* has to be fixed to get the person who suffers from it back to the *norm*. Opposed to this conception of impairment as disability is the *social model*, where the impairment is a given, and the source of the *disability* and associated suffering is instead the societal reaction to it, through impositions of unnecessary constraints and discriminative norms. Following our supposed equivalence with the online security field, the disabled person is not just a minority, but a member of the vast majority of end-users who do not have the expertise to understand every detail but still seek to live a normal life — that is, access web services without putting their data at risk.

The first similarity comes from the disconnect between experts, practitioners and users. In the *medical model*, experts assume they know better than the people they treat or decide policies for, often without listening to their voices. Their recommendations are then followed by normal practitioners, often with a large temporal delay (as they follow the recommendations of the experts who initially trained them, or sometimes just their own intuitions [Nea09, Lip00]). In the context of security and authentication, password policies have until recently gotten more and more complex, informed by often obsolete research. These policies are a major source of user issues [Cen14], and have been for a while, although experts have been looking at it from the user's perspective since before 2010 [KSK+11, SKK+10, SKD+16]. Despite the unequivocal findings, many service providers still have detrimental policies in place. In our parallel, the "disability" might be more adequately termed "lack of adaptability": some people do not feel the weight of the constraints or have efficient workarounds — such as well-configured password managers. But for those who do not have the expertise and know-how, the constraints can seem arbitrary, and the users might "choose" not to participate as the costs are too high, just like disabled people faced with the costs of existing in a public space that ignores their constraints.

This brings us to the second similarity: unexamined norms and constraining default assumptions. In the medical model, the first answer to a building that has steps at the entrance would be to push for the development of step-climbing wheelchairs. Going towards the social model, the possibilities start with having a removable ramp, maintaining the status quo and creating additional costs for the disabled clientele, and go up to having a permanent ramp. Finally, the initial assumption that buildings require steps at the entrance — by now mostly an aesthetic architectural choice — can be questioned and rejected, leading to accessibility by default. When it comes to online security, the default approach is still to have large constraints, often idiosyncratic ones, as with the previous example of complex password constraints. A more usable approach is to use adaptive poli-

[d] The *social model* was initially theorised in the 1980s and has since evolved, generating a fair share of critiques [GP04] and spurring the development of alternative models, such as the capability model where the cost to the individual is not only counted as lost abilities, but also lost potentialities [Bur04].

cies or to use password strength meters, imposing soft constraints instead of strict ones. However, that still means either imposing an additional cost to each user — in the form of new passwords to remember — or increasing their risk if they choose to forgo security and reuse passwords. The real question is whether there is a real need for users to have an account, as those are mostly used to track them for advertising purpose. In a world where the number of accounts per user increases by more than 10% per year [Mer15], increasingly frequent data leaks can cause very bad publicity [Sha18]. The habit of collecting comprehensive data by default just in case the company finds a way to use it might not be worth the risk anymore. This is partially linked to the previous similarity: work done by "experts" who are more knowledgeable is justified even if it puts the end-user at risk, and the lack of transparency is there to "protect" them. This reasoning is applied whether it concerns the thousands of Facebook employees who had access to hundreds of millions of clear-text passwords for diverse purposes, or doctors studying medical errors without disclosing them to their patients [PS09].

The third similarity lies in the onus on the individual, inspired by an ethos of individual responsibility. Either one follows the best practices — using different passwords for each account, not storing them anywhere, never sharing them and changing them frequently — or one is "obviously" asking for trouble. That said, even following best practices is not sufficient to protect oneself when service providers routinely admit they suffer from large-scale intrusions. However, rare are the users who follow them, making the blame-shifting campaign much easier[e], just as disabled people who do not spend their whole time looking for a cure are accused of not making the required efforts. Making it the user's responsibility to ensure their own access is reliable and secure is problematic when the service providers do not keep up with recent research themselves. It is even worse when we encourage an insular technocratic culture of ever-increasing complexity, instead of aiming to simplify the systems used, not only for the end-users but also for the people ultimately responsible for putting them into practice. Naturally, to conclude this list of similarities, the effects mentioned affect minorities even more harshly (especially linguistic ones with reduced access to resources).

What is the take-away from this parallel? That we might learn some lessons from the work already done: listening to the end-users in a continuous dialogue, rethinking the necessity of the constraints we might impose. Always keeping in mind who is excluded by our policies until the ecosystem is welcoming, and in the meantime, providing alternative solutions that are as adaptable to the individual needs as possible. Instead of seeing how we could get users to perform security tasks, we tried to create security from the bottom up, by looking at what people could easily do. As such, the solutions shown in this thesis were made as simple as possible, and some empirical results agree with our intuition so much that they can seem evident. However, it is still our duty to rigorously test what seems natural, even more so with the current replication crisis [EE15]. As will be shown in both our literature review and results from the first few chapters, cryptographic security and usability testing go hand in hand. Focusing on security proofs without testing usability in the real world has led to complex research with no outcome. On the other hand, natural solutions that are optimised for usability by being built on simple behaviours, such as typing biometrics, can have critical security flaws as the threat models are not appropriate. This thesis combines those approaches by checking which natural primitives humans can use, developing security systems built from those and then testing them in practice.

[e]One notable exception is detailed in Chapter 1, and could cost AT&T upwards of 200 million dollars.

Organisation of the thesis

To try to fulfill the goals announced, this thesis is split into three parts. The first focuses on *authentication* from the point of view of service providers, giving them tools to improve usability and security at no cost to the user. The second focuses on *adaptability*: creating strategies that the users can implement by themselves to improve how they create and use passwords and passphrases. The third part is more experimental and seeks to apply some of the ideas studied previously to *voting*, with both general reflections on voting technology and new designs focused on usability. Here is a detailed summary of each chapter.

Chapter 1 gives a general introduction and state of the art in authentication, following the common idiom "Something you know, something you have, something you are, something you do", going first through the general state of passwords today, briefly going over physical token systems and then looking at biometric authentication systems and their common vulnerabilities.

Chapter 2 looks at the difficulties people have when transcribing strings shown to them on a screen, as no systematic study had previously been done on the subject. We ran a pilot user study and a follow-up experiment with a total of 300 participants to look at how code structure and character sets influence speed, error rate, and perceived complexity. We show that alphabetic and syllabic codes outperform alphanumeric ones, even when taking account the increased length necessary to have comparable levels of security/entropy. We analyse a sweet spot composed of six consonant-vowel-consonant trigrams, which can be typed 20% faster, with less than half as many errors as 10-character alphanumeric codes, while having 6 additional bits of entropy out of 66.5 — when generated uniformly at random — and being preferred by at least two-thirds of users. Finally, we show a simple way to add error detection and correction without a high usability cost, while being able to correct all of the 495 errors found in codes of this structure typed during the experiment.

Chapter 3 shows that typo correction for passwords can be achieved without diminishing security or ease of implementation. Building up on work by Chatterjee and his colleagues [CWP+17], who created the first typo-correction system for passwords that is compatible with usual hashing techniques while correcting 32% of what they consider legitimate typos. We introduce a different typo correction system that requires next to no work on the server's side and corrects 57.2% of a bigger set of typos, at a small cost to security. We also establish some lower bounds on the general problem of typo correction and give a theoretical algorithm inspired by homomorphic encryption that is close to optimal in terms of security and space complexity, at a high computational cost.

Chapter 4 focuses on the user's side by introducing a mental password manager called Cue-Pin-Select. It is designed to be able to quickly create a large set of passwords that are adaptable to a wide variety of idiosyncratic constraints. Each password is hard to guess and highly secure, being pseudorandomly generated, and a few stolen passwords are not enough to reverse-engineer the rest of the set. The algorithm can be executed in one's head without pen or paper, and only requires remembering a single passphrase and a PIN. We also report on the results of a 4-day usability experiment to test the algorithm on a dozen volunteers.

Chapter 5 proposes a simple way to improve the passphrases, especially the ones required by the previous chapter, as people tend to generate them in an insecure way,

often drawing from literature. Instead of random generation from a dictionary — which comes with a set of usability and memory problems, we look at the effects of having 99 participants choose their passphrase using words from a randomly generated set. From a security standpoint, people's choice is biased, but the bias is surprisingly limited, costing at most 3% of the entropy. As the people can choose while ignoring words they do not know, this allows the use of a much bigger dictionary than would be used to generate random passphrases, having a net positive effect on entropy. From the usability standpoint, we observe a strong reduction in error rates and number of forgotten words among the participants when compared to a control group with imposed passphrases (between a factor 2 and a factor 6 depending on the type). Finally, we look at the effects of the number of words shown, with a larger set of words increasing bias but also memorability.

Chapter 6 looks at the cost model of mental computing proposed by Blum and his colleagues in a series of papers where they also analysed and framed the problems of mental password managers [BV17, BV15]. We give a preliminary report on a usability experiment with 81 participants, showing evidence of multiple problems in the original model. For example, the expected time for a human to access an element in a list (such as letters-to-numbers) was shown to be extremely dependent on the position of the element within the list, with non-trivial patterns when multiple access happen in quick succession.

Chapter 7 gives an overview of voting technology today, with a focus on concepts such as end-to-end verifiable voting. We give a quick presentation of three such systems: Prêt à Voter, Three-Ballot, and Random Sample Voting. We then analyse the difficulties in implementing such technologies on a large scale today, and finish by going over two voting experiments we organised with Random Sample Voting in the past two years — at the Global Forum for Modern Direct Democracy and the World Forum for Democracy.

Chapter 8 goes back to Three-Ballot and its usability issues and tries to address them without requiring electronic devices or cryptography. We propose three designs based on paper folding and either hole-punching, masking tape or partially translucent paper. They are all end-to-end verifiable and achieve varying levels of usability and security. The last design is the simplest to use, with the voter having to punch a single hole in their stack of ballots, and has a rare advantage: even voters filming themselves in the voting booth cannot prove how they voted. Finally, the physical systems shown require little theory and could have easily been implemented two millenia ago.

Chapter 9 looks at a specific instance of voting, called boardroom voting. In such a situation, all voters can see each other, and the focus is on usability and speed. We look at a large set of primitives that could be used to achieve certain properties such as anonymity of the vote or verifiability, and propose one example of a voting system built on them to achieve those properties, among others.

Elementary definitions

Before we start by looking at authentication in detail, we must give a general intuition for the three main terms of the title, which will be used throughout this thesis.

Usability will be a multi-faceted goal, representing how easy it is to use a given technology. This manifests in a variety of ways, such as:

- *Efficiency*: how much time and energy must be spent to achieve one's goals.
- *Error-resistance*: how easy it is for a user to get the system to fail if they don't use it correctly, and how negative the consequences of said failure are.
- *Learnability*: how easily and quickly a neophyte can get to a sufficient level of competence to use the technology.
- *Accessibility*: whether the technology can be used by people with a wide variety of impairments, such as visual, auditory or cognitive ones.
- *Memorability*: how much the acquired skills in using the technology hold after a period of disuse.

Security will have a meaning partially dependent on context. It will concern the resistance of the protocols shown to attacks of various kinds. For example, attacks on authentication systems can try to access forbidden data, obtain more information from unlawfully accessed data, or prevent someone from using the service when they should be able to. Attacks on voting systems can aim to prevent an election, change its outcome, or obtain information on what individual voters chose. Depending on the context, asserting security guarantees will be equivalent to asserting resistance against known attacks.

Low tech is defined by opposition to high tech. Here, technology is to be understood in its widest sense, with mathematics and political science counting as technology. As opposed to works that heavily rely on established knowledge — creating high towers of dependent results — a large part of the results shown will be built on a limited set of previous works, although drawing from a wider base (an exception is Chapter 3, which necessitates a decent amount of cryptography). When multiple tools are available, the simplest one will hence generally be chosen, as the very simplicity can be a boon in terms of usability to both users and developers. In the context of verifiable voting, for example, one goal was to create a voting system that could have been explained and used in Ancient Athens. When it comes to cryptography and passwords, the methods shown in Chapter 2, Chapter 4, and Chapter 5 do not rely on any technique from the 20th century onwards for their usage — although security proofs and analyses use modern methods.

Contents

1 A Primer to Authentication

1.1 Preliminary definitions

The issue of authentication has been of interest since humans started to interact regularly with other humans they did not know. Proving that our interlocutor is who they say they are might, however, be more difficult when the only information we have is transmitted through an electronic channel, as opposed to being there in person or using the letters of credence offered by the diplomats of old [Que60].

In its purest form, an interlocutor is an agent that, to be trusted, must prove — to a reasonable degree of certainty — that they are who they say they are, through any means from a physical passport to passwords or biometric information like a fingerprint. That can also include proving we aren't who we say we aren't (through a test such as a reverse Turing test or a CAPTCHA). In its electronic form, establishing an interlocutor's identity went from the password systems implemented on one of the first time-sharing OS, the CTSS — or Compatible Time-Sharing System — in the early 1960s [MT79], to the ubiquity of authentication methods used today, from accessing a bank account to unlocking one's phone. And, starting when the password file on the CTSS was made publicly visible on all terminals, electronic authentication has been riddled with security and usability issues. These two criteria to optimise often seem opposed to each other, as improving security to prevent adversaries from being able to access the system typically requires increasing complexity and decreasing flexibility of the system. On the other hand, making sure that the legitimate user always has access with limited delay and low complexity generally comes at a substantial cost to security, especially when one wants to have a backup system in place to correct errors in case the user has their identity stolen.

Two very telling instances of the tensions between access and security are highlighted in examples from cryptocurrency. Before the Bitcoin ecosystem had any time to mature, ownership of the cryptocurrency was equivalent to knowledge of a private key — or set of keys — often stored in insecure places. Before theft was ever an issue, many users lost their Bitcoins (BTC) in various accidents, sometimes merely by discarding the wrong hard-drive [Run14]. Even professionals have vulnerabilities, as shown when a major Bitcoin exchange platform — Bitomat — lost its own wallet and the 17000BTC it held for its users because of an error in the single virtual machine that held both the wallet file and the backup [Dot11]. It is hard to know exactly the amount of "lost" Bitcoins but estimates generally go from 2 to 6 million Bitcoins [RS13, RR17]. In its original state, the system

tended to be too secure for users not to sometimes lose access. As a consequence of this, various fail-safe systems have risen, to prevent permanent loss of cryptocurrency wallets [GD14]. However, this made the new systems more vulnerable to hacking, as happened recently when Michael Terpin, a cryptocurrency investor, sued AT&T on the basis that social hackers had managed to obtain control of his phone account using a SIM swap fraud [OTB18]. Using the phone, they managed to bypass the 2-factor authentication and steal close to 24 million dollars in various cryptocurrencies. This is but one of the myriad examples of failures to find the appropriate compromise between usability and security.

This chapter seeks to give a primer on authentication, covering the current issues on both the security and the usability fronts, as well as the solutions used today, to better situate the contributions of the first two parts of this thesis. We will focus on *discrete* authentication, where the system has to make a decision at a given point in time, rather than *continuous* authentication, where the system tries to lock out an adversary if the usage pattern is anomalous [PCCB16]. After some initial considerations, we will follow the usual adage in the field: *Something you know, something you have, something you are and something you do.* This means that authentication can be based on any source corresponding to one of these four categories, and should ideally come from multiple sources simultaneously. The first, for example, covers passwords and any shared secret, while the second can be electronic devices, physical keys or access cards. The last two — with the second sometimes subsumed in the first — generally correspond to biometric authentication, such as fingerprint or iris checking, but also gait analysis. Before that, however, we expand on some essential concepts and how they will be treated in this work.

1.1.1 Attacks

A central aspect of any security system is the type of attacks it is meant to counter. In this thesis, we will always consider two primary forms of attacks. The first corresponds to an adversary trying to masquerade as a legitimate user, by trying to give correct credentials, as the hackers did in the Terpin case. The second corresponds to an attack on the party doing the authentication — which we'll denote as the *server* from now on. The server has to store data to be able to compare it to the one the legitimate user provides when authenticating. Depending on how it is stored, this data can be useful either as it is, or as information to launch an attack against another service. This is the second main kind of attack we'll consider: *trying to obtain useful information* from the server's database. There are of course many other kinds, sometimes reducible to these two, sometimes not, as when an adversary's main goal is to prevent the victim from accessing the service[a], but we won't focus on them here. Another important type of attack targets the users themselves directly, through social engineering, phishing, shoulder-surfing attacks (spying on a user by looking directly "over their shoulder", possibly with a camera), or even physical theft. These considerations will be briefly expanded upon, but a detailed study is also beyond our purview.

A second important distinction is between *generic* and *targeted* attacks, which is particularly true in the third part of this chapter. Generic attacks can correspond to the massive dictionary authentication attacks on various online retailers' website, whereby adversaries

[a]Sometimes, the denial of service doesn't even require an attack, as in the Terpin case, when the hackers just timed the attack well so that the fraud department that the victim called was on break for the day.

rely on the statistics of access control. With millions of accounts in play, some will generally be vulnerable. As research has shown, automated illegitimate login attempts on retailers' websites account for 80% to 90% of total login traffic [Sha18]. A careful user can generally avoid falling victim to the generic attacks, although sometimes the security isn't in their hands. For example, no matter how strong a password is, if it is reused and one of the databases is insufficiently protected, the user will be as easily victim to credential stuffing as if they had used "MyPassword1" [IWS04]. On the other hand, targeted attacks are much harder to prevent, and the optimisation has to include the cost to the attacker. Case in point in Terpin's lawsuit, where even the additional security he had asked for wasn't enough to prevent hackers using some of his personal information to get their hands on his 25 million dollars.

1.1.2 Entropy

One of the most commonly used — and abused — measures of security in authentication research is entropy, typically Shannon entropy [Sha49], which is formally defined over a probability distribution by

$$S = -\sum_i p(i) \log_2 p(i)$$

where i goes over all possible events, each having probability $p(i)$.

This is useful when the distribution is uniform, in which case $S = \log_2(N)$, where N is the number of possibilities. In such a case, the entropy, given in bits[b], gives a good measure for the effort needed to cover a significant fraction of all the possibilities, which is why it has traditionally been used for passwords. A password system with one more bit of entropy requires on average twice as much computation for an adversary to obtain access. However, this is only appropriate as long as they are chosen uniformly at random among a large set. This measure can indeed fail easily, as when one considers the distribution where the password consisting of the single letter `a` is given the probability $1 - \frac{1}{k}$, and where 2^{k^2} different passwords[c] are each given probability $\frac{1}{k} \times 2^{-k^2}$ for a given k. Computing the entropy for such a distribution is easy, as it is

$$S = k + \frac{\log_2(k)}{k} - (1 - \frac{1}{k}) \log_2(1 - \frac{1}{k})$$

As k increases, the real security of the passwords in most of the distribution goes down — because the bad password becomes increasingly probable — but the computed entropy goes up. Because of this effect, many other metrics have been used, one of the most popular being the *min-entropy*:

$$S' = \min_i(-p_i \log_2(p_i))$$

For our example, this would give $S' = (\frac{1}{k} - 1) \log_2(1 - \frac{1}{k}) \approx \frac{1}{k}$ bits of min-entropy. As a metric for security, it is much more stringent, but also potentially harder to use. As opposed to Shannon entropy which is mostly influenced by the lowest probability events, it is entirely determined by the highest probability event. Intermediate metrics, such as guessability and password quality, have also been introduced to better model the resistance of password systems [KKM+12, MCTK10, MKV+13, RK18, TAM+13].

[b] The correct unit is the shannon, corresponding to the information content of a uniform distribution between two possibilities, but *bit* is often used in the same sense.

[c] In this case, the passwords can correspond to all the different k^2-bit possible passwords.

Two interesting examples, respectively introduced by Boztaş and Pliam [Boz99, Pli00], are the β-success-rate, defined as

$$\lambda_\beta = \sum_{i=1}^{\beta} p_i$$

where the p_i are sorted by decreasing probability, corresponding to the success rate of an adversary with at most β tries, and the α-work-factor:

$$\mu_\alpha = \min \left\{ j \middle| \sum_{i=1}^{j} p_i \geq \alpha \right\}$$

corresponding to the number of guesses necessary to gain access to a proportion α of all accounts.

In an extensive review [Bon12], Bonneau compared many different metrics, and normalised the previous two metrics to get numbers directly comparable to our usual interpretations of entropy.

Over the course of this thesis, the entropy values given will be calculated using Shannon entropy unless otherwise specified. The rationale behind this choice is that there will always be a strong random component in the distributions considered, such that most of the p_i will be similar, with the entropy computed coming not from a few very low probability events but from the bulk of the distribution. Where appropriate, alternate entropy measures will also be given.

1.1.3 Kerckhoffs's law

In practice, the previous counter-example with the lopsided distribution might resist many adversaries who only get access to a few hard passwords and infer from this that all passwords follow this structure, discounting the single high-probability element[d]. This is where an important 19th-century assumption known as Kerckhoffs's law — known under many names and also attributed to Claude Shannon — comes into play. Shannon's formulation, "the enemy knows the system", is — as is standard — one of the central assumptions in all security proofs that will follow. As such, an adversary might not know a given user's password, but they could know the distribution of all user passwords. This is realistic as the distributions are increasingly well-studied thanks to the frequent credential leaks: more than 2.3 billion username/password pairs were leaked in 2017 [Sha18]. A few of the security solutions introduced in this thesis are relatively easy to think of, and very similar to solutions already used by some experts individually. This is by design, as the main goal is not to improve the security of the top 1% of experts who already have good homemade security systems, but to make good solutions accessible to the other people who are hopefully obsessed with other things than cryptography but also deserve security. We will work to prove results on them that follow reasonable security assumptions. The average user is famously bad at estimating the security of a system [WDCJ09, KDFK15, GF06, UBS+16] — as are most "experts" [TBA14, ÖTC16] — and home-made solutions often become vulnerable once generalised. For a simple example of this, one can think of "ji32k7au4a83", an apparently secure password that pops up in many databases with high frequency. Although

[d] To make this scenario more reasonable, instead of "a" being the password with probability $1 - 2^{-k}$, it could be any medium-strength password, which would only marginally improve the security.

it looks random, it makes complete sense when contextualised: the keystrokes correspond to "mypassword" in Han characters, when typed using the Taiwanese typing system — Zhuyin Fuhao — instead of qwerty [Lia19], and the "clever trick" is worryingly common, defeating its purpose. This line of reasoning will be extended in Chapter 4, but first, let's focus on the different authentication methods.

1.2 Something you know

Authentication based on what one knows is the default and most versatile method. This encompasses, among other things, passwords, which have been an enduring subject of research for academics, although there are still frequent misconceptions, especially on how ubiquitous they are. This concerns not only the total number of passwords available but also the frequency at which they are created and used. A study focused on the USA and the UK observed that more than 37% of respondents needed a password for a new service at least once per week, and more than 92% needed at least a few per year [Cen14]. This was despite 47% of the respondents answering that they have at most 5 active online profiles, strongly underestimating their own prolificacy.

The main advantage of passwords is that — in their best form – they are pure information, meaning that they have multiple important features:

- They are *adaptable*, with an arbitrary amount of security that depends only on the distribution (and hence on the length).
- They are *mutable*: if they get compromised, they can easily get changed — as long as it is possible for the legitimate user to prove their identity and legitimacy through secondary means.
- They are *shareable* and *transferable*: if needed, the legitimate user can give the password to a trusted party.

These features have consequences on both security and usability, and we will first focus on the former before looking at the latter.

1.2.1 Faking authentication

As stated earlier, we will focus on two main kinds of attacks on passwords, discounting the cases where the user is vulnerable to keyloggers, shoulder-surfing and social engineering.

The first matter is then making sure that no-one can guess the user's password. Traditionally, this has been interpreted as having a high entropy, which, as passwords are adaptable, can be extremely variable. This is the crux of the problem, as even today, some trivial passwords are still used very frequently, with 3% of users whose credentials were leaked still using "123456" in 2018 [Doe18, Van10][e]. In older but more rigorous studies, multiple databases of leaked passwords were shown to have around 6 bits of min-entropy [MM12].

Bad password quality problems have persisted for decades with initial reactions to the trivial passwords chosen by users mostly being to add increasingly complex constraints, typically of the following kinds:

[e] In an ironic note, the 25th most used password leaked in 2017 was "trustno1", replaced by "qwerty123" in 2018 [Doe17].

- Requiring passwords to pass a "strength" test.
- Requiring a mix of upper- and lower-case characters.
- Requiring a certain length (but often forbidding passwords of more than 16 characters).
- Requiring numerals, and sometimes symbols.
- Requiring users to change their passwords frequently, with little allowed overlap between passwords each time.

Recent research has gone against this trend through two main lines of argument. The first, following Florêncio, Herley, and Coskun [FHC07], questioned this practice by giving strong reasons why 20 bits of entropy would be enough. They stated that, combined with appropriate access rules such as a three strikes policy — blocking the account after three wrong passwords — this would guarantee high security. Thanks to those policies, one can make a calculus of the utility to a potential adversary: if the proportion of accounts that can be brute-forced is low enough — at least compared to the cost of the attack — the attack isn't worth doing. However, there are some problems with this method, such as the greatly increased attractiveness of *denial-of-service attacks*, when an adversary overwhelms the system's ability to process all requests [GSD16, vOS06], or spams using one person's login to prevent them from accessing the service. To address this, the original paper advised keeping the list of usernames secret, but this evidently cannot work against targeted attacks.

Rate-limiting — for example, by imposing a delay on the user's side or having to solve a CAPTCHA — can also be used [PS02, vOS06] to reduce the efficiency of online attacks, but it has its own set of issues. Linked to this reasoning is the analysis that most adversaries aren't trying to brute-force credentials — which is too costly and visible — but instead re-use credentials that were obtained through phishing (in which an adversary dupes a user into exposing their key), keyloggers, or attacks on another credential database [Pin14, FHvO14a]. With the rate of new account creation, password re-use is then a much bigger threat than low-entropy passwords [DBC$^+$14, FH07, WRBW16]. In a similar vein, other papers have raised doubts on the utility of some of the constraints from a security standpoint, covering many different types, from the expiration of passwords after a given time [ZMR10], to the requirements of extended character sets [UNB$^+$15].

Paralleling this line of questioning based on the fact that the security threats have changed, a second line investigated the amount of effort users could reliably provide [KSMD08, PJGS12, FHvO14b]. Indeed, since the first password constraints were introduced in 1979, the one reliable effect has been the bypassing of those constraints by users [Cra16, KSK$^+$11, Mar12, SKK$^+$10, WT99]. One typical example gave its title to a provoking paper investigating this behaviour by Ur et al.: "I added '!' at the end to make it secure" [UNB$^+$15], as users' perception of the security can be badly off-target. Unforeseen secondary effects have been observed, such as the tendency to try many different passwords when one is forgotten [RAS$^+$17], or to discard memorising the password altogether and reset it when needed, both of which come with risks [Lip16]. Thanks to this line of research, many different kinds of solutions have been introduced focused on either creating directly usable passwords [Lee14, SMK$^+$17, TAT07], or studying how to make users safer by teaching them better practices in more efficient ways, from password meters [UAA$^+$17, CMP12]

to better adaptive constraints [Yan01, PLV+02, FB08] and reflections on how to educate users [SL12, Aba14, YBAG04]. One easy result from these studies is that longer passwords are generally thought to be both more usable and more secure than complex passwords [SKD+16, SKD+14].

An interesting problem that is still not entirely solved is whether and how to use constraints, as hiding them limits the user's ability to avoid them consciously and maliciously [MMD14], but can lower usability by being extremely frustrating [She15]. With the advent of mobile computing, now source of a significant fraction of login attempts, changing usability issues have also been diagnosed [NOP+06, CFSW12], as text input on those devices comes at a higher cost [MKS+16]. With the issues mentioned, there has been a significant financial and time investment into developing alternatives to usual passwords.

1.2.2 Alternative methods

Passphrases

Besides passwords, many other knowledge-based systems exist which can fulfill the same role. The simplest and most natural one is an extension of the password, which is the *passphrase*, composed of multiple common words. This solution has the strong advantage of using our natural abilities to remember multiple units of meaning — words — instead of multiple random characters [Mil56, BH74], and was used throughout history, and electronically as early as 1982 [Por82]. Despite multiple indications that it has good properties [SKK+12, KSS07, KSS09], it does suffer from two main defects. The first is that, passphrases being much longer than passwords, the time cost of typing them might not be worth the added security for most users. The second is that passphrases have the exact same vulnerabilities as passwords when users are free to create them without constraints. Indeed, they have been shown to typically be made from insecure, linguistically easy-to-crack patterns [KRC06], like song lyrics or famous quotes. A significant number of passphrases created by the Amazon PayPhrase system were easily hackable, with 1.3% of accounts being vulnerable to a 20,000-word dictionary of terms used in popular culture [BS12]. In another study [YLC+16], one passphrase method led 2.55% of users to choose the same sentence (quoted from Shakespeare). Despite multiple protocols encouraging users to make personalised sentences, five out of six methods had many occurrences of different users ending up with the same passphrase. With the one method out of six where no two users chose the same sentence, the researchers still observed some quotes and famous sentences, but few enough not to have collisions on a database of only 777 passwords. These studies show that letting people choose a passphrase with no constraints leads to low-entropy passphrases, even when giving them instructions to make personalised passphrases. Helping users choose better passphrases will be the focus of Chapter 6.

Challenge systems

Unlike passphrases which are just a specific form of long passwords, challenge-based systems have very different properties. By asking the user to answer a series of questions or to complete a list of tasks, they can achieve high levels of security, with a higher promised level of usability. This idea has been around since at least the early 1990s, initially as a list of personal questions [ZH90].

A natural evolution of this kind of system is the visual — or graphic — password,

where the user is confronted with a sequence of images and has to react, for example, by identifying known pictures [JGK+03, BS00], or by clicking on certain zones in a sequence of pictures [CBvO07, KD14, WWB+05]. Some research has also looked at mixing different methods and mnemonics, such as storytelling plus visuals [PLB14] or sound-augmented visual passwords [SA11].

This can be problematic, however, for multiple reasons:

- Free-form answers can lead to high error rates and frustration as people might misremember the exact spelling they used [SBE09].
- To achieve high entropy, a potentially long sequence of challenges is needed. This requires more time from the user and increases the complexity [JA09].
- To avoid repetition of challenges between different services — and collisions in the image set — there needs to be a large set of potential challenges. User-provided challenges are also riddled with usability and security issues [JA10].
- They can be more vulnerable to shoulder-surfing attacks depending on the structure [LFZS09].

With the different alternatives to ensure security and usability on the user side covered, let's now look at the server side.

1.2.3 Attacking the server

When a user seeks to prove their identity to the server — we remind that we use *server* as a generic term for the person or service checking the identity — the server needs to have some data to verify the authenticity of the user. Ideally, this data should be encrypted, unlike what was done in the 1970s when one could store all the passwords unencrypted in a single text file [MT79]. The type of encryption matters and service providers should follow the evolving best practices, but the frequent credential leaks show that, even among large service providers supposed to have experts working on the subject, security is severely lacking.

The general server architecture for password authentication has stayed similar for a few decades, with the best practice consisting in hashing the password on the client side, storing this hash on the server, and comparing the hashes to make a decision when the user tries to login again[f]. The service provider should also include a *salt* — a pseudorandom string that is concatenated with the password before hashing — to prevent attacks that make use of a large table of pre-computed hashes, also known as a rainbow table [KKJ+13]. Finally, the salts should be different from user to user — which isn't always understood by service providers[g] — for example, by computing the hash as a function of the website and the username. The question is then whether service providers manage to implement up-to-date security and use the right hash functions. To get an idea of how slow the adoption of new security technologies is, let's give a quick timeline of milestones:

[f] There is no good reason for a general user to transmit a non-hashed password to the server in a web environment, but many still work under that assumption, with the server hashing cost being a major point of contention in a recent password hashing algorithm competition [HPM15].

[g] If a single salt is used for a whole database, it prevents attacks using generic rainbow tables, but allows the computation of a website-specific rainbow table, only marginally improving security.

- In 1991, Ronald Rivest improved MD4 to prevent certain recently discovered attacks, replacing it by the now widely used MD5 hash function [Riv92].

- MD5 is cracked in two steps in 2004 and 2005, making it possible to find collisions quickly [WY05]. This does not imply that reversing the hash function became easy, but it did introduce vulnerabilities. For example, if the hash function is used as is — without adding a salt — instead of finding the original password, it is enough to find a preimage — that is, an input whose hash is the same. Experts then recommended moving to SHA-1 or SHA-2 (Secure Hash Algorithms), respectively introduced in 1995 and 2001.

- Starting in 2005 and continuing until a full collision attack in 2017, costs got progressively lowered for collision attacks on SHA-1. This happened both through improvements on the theoretical side and through the development of better and cheaper hardware [SBK+17]. As such attacks became financially viable, experts recommended moving to SHA-2 or SHA-3. Both alternatives offer high security and co-exist as potential solutions with very different designs (as unlike the first three families of secure hash algorithms SHA-0, SHA-1 and SHA-2).

With vulnerabilities in both MD5 and SHA-1 known for more than a dozen years, combined with regular public outcry due to leaked data, one could expect that most service providers would update their policies, but this is not the case. In an extensive analysis [JPG+16], Jaeger et al. looked at 31 credential leaks going from 2008 to 2016, totalling close to 1 billion email/password pairs worldwide. Of the leaks considered, more than half stored entirely unencrypted credential pairs (including gmail.com in 2014 and twitter.com in 2016, although they couldn't make sure the data was authentic), and only one, ashleymadison.com, used a strong level of encryption — bcrypt — while still making some mistakes[h]. Table 1.1, partially extracted from [JPG+16], shows the main authenticated leaks they analysed. Although this table paints a frightening picture of current practices, the reality is even worse. The two main hashing algorithms currently considered to be secure are SHA-2 and SHA-3 — or rather, algorithms from these families. However, this has a caveat. Unlike online attacks where 10^{10} bogus login attempts are easily noticeable, leaks can be studied offline, on special-made machines. This is one of the main risks with database leaks, as even consumer machines can nowadays brute-force more than 1 billion hashes per second [GXQ+17, DK15, Spr11]. With that kind of capability, dictionary-augmented brute-forcing becomes a strong option, as most passwords deviate only poorly from real words. This is true no matter the kind of hashing algorithm used: if the space of all possible inputs remain small and the search can be parallelised efficiently, it is enough to hash all probable inputs to get the inverse of most hashes. To address this, two main algorithms were initially used, PBKDF2 and bcrypt, which can run recursively n times on their own output to artificially increase the computing time [WZ14]. This slows down the algorithm in an unavoidable way, negating the gains due to more powerful machines but is still vulnerable to parallel brute-forcing. Argon2 is a more recent solution, specifically made to be hard to parallelise by requiring an arbitrarily large amount of memory [BDK16], and is now one of a set of good alternatives [HPM15].

[h] The main mistake made was storing md5 hashes of case-insensitive versions of the passwords, from which it was possible to compute a preimage, leaving the option of computing the full password by hashing the 1000 or so remaining possibilities through bcrypt [Goo15].

website	encryption	number of accounts leaked	date of leak
myspace.com	SHA-1	358986419	2008
gawker.com	DES	487292	Dec. 2010
aipai.com	md5	4529928	Apr. 2011
csdn.net	clear	6425905	Oct. 2011
tianya.cn	clear	29642564	Nov. 2011
vk.com	clear	92144526	2012
linkedin.com	SHA-1	112275414	Feb. 2012
imesh.com	md5+salt	51308651	Sep. 2013
xsplit.com	SHA-1	2990112	Nov. 2013
51cto.com	md5+salt	3923449	Dec. 2013
xiaomi.com	md5+salt	8281358	May 2014
000webhost.com	clear	15035687	Mar. 2015
sprashivai.ru	clear	3472645	May 2015
ashleymadison.com	bcrypt	36140796	July 2015
17.media	md5	3824575	Sep. 2015
mpgh.net	md5+salt	3119180	Oct. 2015
r2games.com	md5+salt	11758232	Oct. 2015
nexusmods.com	md5+salt	5918540	Dec. 2015
mate1.com	clear	27402581	Feb. 2016
naughtyamerica.com	md5	989401	Apr. 2016
badoo.com	md5	122730419	June 2016

Table 1.1: List of leaks analysed by Jaeger et al. with number of credentials leaked, date and encryption method used in each case.

To summarise, secure passwords require not just good policies on the front-end but good management on the server side, with good hashing algorithms. That this is still far from being the case is attracting attention, with some work focused on education or design guidelines for developers [BA17, HA14]. For a more comprehensive review focused on password-based authentication, one should read Merve Yildirim's PhD thesis [Yil17].

1.3 Something you have

This second source of authentication was, for the longest time, the one most used in practice, as it can mean all sorts of devices, including physical keys to open one's door-lock. It is also the basis for many 2-factor authentication systems. This is not the focus of this thesis, but some basic results and solutions should still be mentioned. When it comes to online authentication, this method has many forms:

- Notebooks — or post-it notes — on which passwords are written.
- Physical seed cards with rows of characters on them, from which a user can derive their passwords, as in [Top10].
- Smart-card readers such as the ones used by many government agents, which lock down the attached device when an authentic card is not inserted [LS09].
- Hardware-based one-time-password generators, such as RSA SecurID [SE07].
- Smartphones, especially through the distribution of text messages [SCL12].
- Specialised hardware, like Pico, which automatically handle authentication and can shut themself down automatically when potentially compromised [Sta11, BHvOS12].

The main reason we won't study those methods at length is that they all suffer from the same vulnerability, by design: they make the user dependent on a physical object. This might be catastrophic (as when the document with all the passwords written on gets stolen) or just highly inconvenient (when the device is stolen but the thief has no chance of unlocking it). The strongly increased possibility for denial attacks makes this sub-optimal for critical uses, especially against targeted attacks. Besides denial attacks, the same problem persists: a system is only secure when it is correctly implemented, and the standard 2-factor authentication — using both a password and a code received by SMS — which has become commonplace is now very much at risk, whether it is because of attacks directly on phones [MBSS13], or poorly trained agents working for service providers [OTB18]. It is, naturally, still better than a lot of what is currently done, but it is preferable to limit the dependency on physical devices, which is the goal of Chapter 4.

One special mention must be made for password managers. They come in two flavours: online and local. The local password managers are the most secure (as long as they are correctly implemented, but that generally seems to be the case). They constitute a single point of failure in two ways. First, stealing the device creates a complete denial of service for the user if they don't have a frequently backed-up password database. Moreover, as access to the password manager is centralised through a single password, they suffer increased risk from potential keyloggers and shoulder-surfing attacks.

When it comes to online password managers, we can distinguish pure password services like LastPass or My1login [KSC11], from the services integrated into larger service providers such as Google or Facebook in their quest to centralise user accounts [BMK+18]. The latter creates potential privacy and social problems on top of the security issues [KEA+14], but the increased centralisation can improve usability. Both types are potentially vulnerable to database leaks, though — and have been in practice — with some using outdated hashing systems [LHAS14], while others have better security while still suffering from leaks that are not immediately costly [TL12]. Those effects are also compounded by the

use of multiple devices, with people being even more distrustful on mobile devices [AR16, HK18]. Besides the security issues, there is the important question of usability, which has been repeatedly studied, showing that user distrust of online and centralised password management introduced a high cost [CvOB06, KSC11]. This is interesting as it is at odds with the regular complaint of low password usability.

1.4 Something you are and something you do

The last two methods to authenticate someone are strongly linked, as "something you do" is in fact "something you do in a way no-one else does", hence linked to who you are. It then makes sense to study them jointly, both static biometrics — such as fingerprints or retinas — and dynamic biometrics — such as speech or gait — which can also include behavioural biometrics — such as signing or typing patterns.

1.4.1 Biometrics basics

Biometric authentication methods have been used for decades, from initially limited uses in specific high-security sectors, such as banking or the military[i] since the 1970s, to the biometric passport introduced for the first time, in Malaysia, in 1998 [IRI19]. They benefit from a high level of perceived security, and since their introduction in mainstream smartphones in 2011 (Motorola Atrix 4G, followed by the Apple iPhone 5S in 2013 and alternatives from Samsung and most other constructors in 2014), their prevalence has increased dramatically, with 42.5% of users unlocking their phones through fingerprints or face recognition in 2018 [GB18]. As such, they constitute a serious alternative to passwords and must be analysed here, but first, we must provide a few definitions. A very complete analysis of the state of the art can be found in a 2018 survey by Choudhury et al. [CTI+18], and an older survey by Yampolskiy and Govindaraju also provides interesting viewpoints [YG09]. Thorough analyses and additional references for the biometric authentication systems shown below can be found in those surveys.

Biometric systems can be compared on their performances along a large set of criteria, among which we can cite the following:

- *Security* (mostly dependent on error rates).
- *Renewability*: whether the features observed can be changed at will if stolen (easier for a signature than for DNA).
- *Durability*: whether the features are stable over time or not (true for DNA, not as much for ear shape).
- *Anonymity* and *pseudonymity*: whether one can mask their identity without raising attention (changing one's fingerprints is harder than speaking differently).
- *Usability*, dependent on the intuitiveness, speed and requirements for complex equipment (such as eye scanners).

This section will focus on the first two, as they are the source of many issues.

[i] The US Air Force started investing heavily in the development of a centralised biometric authentication system in 1974 but ultimately scrapped the project, as there were apparent security flaws. They are still using biometrics to this day, although on a case-by-case basis [dLB07].

1.4.2 Error rates

Unlike passwords, passphrases or access cards, biometric authentication does not correspond to checking whether a single provided value is correct. Instead, a master template of the pattern is recorded once (for example, a high-quality image of one's fingerprints), and stored securely. When authenticating, a new pattern is measured and compared to the stored original — or a transformed version of the original, typically through a pseudo-one-way function that partially preserves proximity. The decision is then made depending on how similar the two patterns are. Naturally, this comes with a potential for errors of two types: false acceptances (false positives) where a forbidden user gets wrongly authenticated, and false rejection (false negatives), where a legitimate user gets wrongly locked out. The stricter the decision process, the lower the *false acceptance rate* — or FAR — becomes, at the cost of increasing the *false rejection rate* — or FRR. A typical example of the error functions — in this case on non-human biometrics — can be seen in Figure 1.1. We can observe that the FAR, initially at 100% — corresponding to the algorithm accepting every entry — quickly diminishes to reach a plateau, and progressively diminishes afterwards. The point at which both curves intersect is called the *equal error rate*, or EER. Inconveniencing one legitimate user by rejecting them once seems less costly than allowing a security breach by giving access to an illegitimate user, but the EER is often used to compare biometric authentication method as it gives a first approximation of the discerning ability of the system.

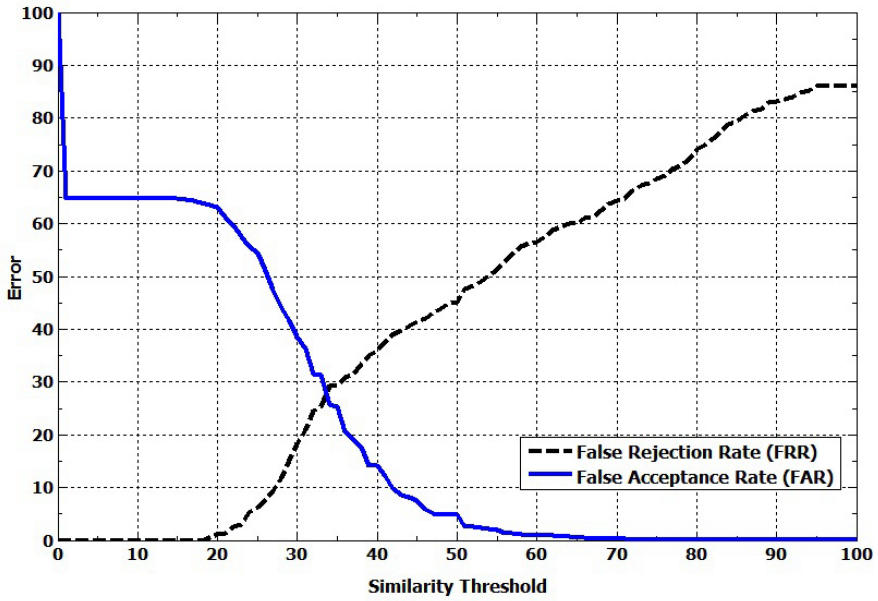

Figure 1.1: A curve displaying the false acceptance and false rejection rates, as a function of the computed similarity between the stored and measured patterns. The Equal Error Rate corresponds to the intersection point of the two curves. Sourced with permission from Awad et al.'s paper on using muzzle print images to identify cattle [AZM+13].

1.4.3 Listing biometric features

To give a quick idea of the number and the diversity of potential sources used for biometric authentication, here is a not-quite-exhaustive list of the features used for static biometrics:

- Fingers, mostly through fingerprints, focusing on the pattern of minutiae, or special points, corresponding to the endings or junctions of ridges and furrows [HAK13]. Alternatives include knuckle structure, focusing on the texture and the shape of the wrinkles [AIA14], or nail-bed structure, focusing on the arrangement of skin folds under the nail [GTP09].

- Hands, either through general geometry (size, length and shape of the fingers and gaps) [GTAF03], palmprints — similar to the fingerprint but extended to the minutiae of the whole palm [KZK09] — or even hand vein patterns captured by infrared sensors [YAS10].

- Face shape, either with normal pictures, using a diversity of features as minutiae [ZCPR03], or by facial thermography, seeking the pattern of heat generated by the vascular system through infrared captors [HV12]. Ear geometry has also been considered as it is quite distinctive and can easily be analysed in combination with other features [ARH+13].

- Eyes, mostly through the texture of the iris [BHF08], although the retina is also an option (when shining a bright light into the eye) [BBW08].

- DNA, generally by the analysis of selected loci of high variability, although more complete scans are becoming available [ZVJ09].

To these static features used for biometric authentication, we can add the following dynamic features:

- Signature, through analysis of handwriting, possibly the oldest and most used authentication pattern, still used in contract law all over the world [PS00].

- Dynamic face patterns, either through micro-movements [JJY08], general head movements in 3D [KFB05], or even the dynamics of smiling [AG15].

- Voice, which has been one of the oldest features used electronically (at least since 1974 [dLB07]), either by repeating a sentence on which one is trained or by reading an arbitrary text.

- Whole body dynamics, such as one's gait or one's way of walking, either captured by external cameras [BCD02] or by accelerometers on the body [GHS06] or even in one's phone [DNBB10].

- Eye dynamics, such as the saccadic movements [KKK12] — especially when seeing familiar images — gaze trajectory [DG11], or blinking patterns [WPPS05].

- Keystroke patterns, generally by looking at the duration between hitting two specific keys, which depends on the hand geometry and muscle memory [RLCM98].

Although this is already a rich list of features, we can mention more experimental ones such as heartbeat analysis, skin deformation on the finger when pressed on the sensor, micro-movement patterns when using a mouse or even perspiration and whole body salinity levels.

1.4.4 A common security architecture

It turns out that, despite the wide range of features considered, the authentication systems are almost always based on the following architecture:

1. A sensor — or an array of sensors — receives data, often in audio or video format.
2. An algorithm checks the quality of the data input and extracts specific features (for example, minutiae for fingerprints [CDJ05, LJY02], pitch and rate for voice recognition). This can include liveness detection: checking that the data is coming live and is not being reused.
3. The features are sent to a database, sometimes after being encrypted or hashed.
4. The database compares the new data to the recorded pattern and measures similarity before making a yes/no decision depending on its similarity threshold.
5. The decision is sent back to the application requiring the authentication.

This architecture is simple, but it still leaves many different avenues of attack to adversaries. A complete taxonomy of attacks can be found in a good review by Singh and Singh [SS13], but here are the main ways to bypass biometric authentication systems:

1. Targeting the sensor itself, either by changing its environmental conditions, presenting a physical copy of similar data or similar methods.
2. Simulating a sensor, where fake data or already stolen data gets injected to the feature extraction algorithm.
3. Attacks that go one step further and inject fake or stolen data after the feature extraction and liveness detection.
4. Injecting wrong patterns directly into the database, or intercepting and modifying the communication[j] between the database and the comparison system.
5. Changing the behaviour of the comparison algorithm, or intercepting and modifying the communication between the comparison system and the initial application.

To give practical examples, let's consider an authentication system based on fingerprints. An attack on 1) could be to lift the user's fingerprints on any item they generally use[k] before replicating the fingerprint and showing it to the sensor [CJ16] — possibly on a

[j] Here, the communication can be either through the network in case of a centralised database, or between the different processes in a device, especially when interfacing with trusted hardware.

[k] This also has an application against passwords, as fingerprint patterns from oily residue on phones have been used to reconstruct the touching pattern and infer the password [ZXL+12].

conductive and heated support to bypass the liveness detection systems[l]. This is generally called a *replay attack*, although some attacks of the second kind are also in that category. An attack on 2) would be to replace the sensor by either software or hardware methods and send a stolen scanned fingerprint directly to the feature extractor. An attack on 3) would be to send stolen features directly to the database, whereas an attack on 4) would be to add the adversary's fingerprints features to the database so that the adversary can unlock the system with their own fingers, and an attack on 5) would be to bypass the system directly and replace the message received by an acceptance message.

1.4.5 Unchanging versus changing biometrics

The previous reflection brings us to a fundamental dilemma in biometric authentication:

- If the biometric features considered are unchanging — such as iris pattern, facial bone structure or DNA — it becomes possible to make an arbitrarily precise model of the target, and to use it to obtain access [SWL15].
- If the biometric feature considered is renewable — such as signature or speech — the inherent variability requires the similarity threshold to be low. This makes it vulnerable to brute-force (or at least optimised brute-force).

The terms of *physical* versus *behavioural* biometrics are often used instead of *changing* and *unchanging* but this is not quite equivalent and can be misleading: for example, facial recognition is physical but quite changing — unlike facial bone structure. Let's go into further details, putting aside, for now, the issue of liveness detection, and consider the pattern collected as a point in the space of all possible patterns. As the capture and feature extraction is an approximative process, the authentication system should accept all points in a topological ball[m] centred on the original point. So far, the system is qui similar to passwords — for which the ball is of radius 0 — but the latter have one major advantage. When one gets stolen, the user can easily create a new one, anywhere else in the given space. On the other hand, with the unchanging biometrics, for any given feature used, all the measured points will be in a very limited region of the space, meaning that if an original point gets stolen, that kind of authentication is ruined forever for that user.

If the renewable biometric authentication is used, then it becomes possible to renew the pattern, for example, by getting a different sentence to pronounce for voice detection, a different text to type, or a different movement to make. However, this suffers from two distinct problems. In many cases, the new point still ends up being in a similar region of the space. Moreover, all the biometric methods with renewable patterns have to allow for a certain amount of imprecision: as long as the pattern is partially voluntary, it becomes very hard to repeat it perfectly. This leads to high levels of inaccuracy, making the system vulnerable to brute-force attacks.

This is very visible in practice, with the high levels of equal error rates in most biometric authentication systems. Those rates are generally between 1% and 6-7% for everything but iris-based authentication, which has an equal error rate of 0.01% but is not renewable and vulnerable to theft. Even multimodal biometrics — where one uses multiple sources

[l] An old but common way of spoofing sensors has been to use gummy fingers made of cheap gelatin but presenting properties similar to human fingers [MMYH02].

[m] In the abstract sense of the set of all points at a distance at most D from the center of the ball, for a constant D and a certain distance.

and biometric features for liveness detection and improved error rates [RAK16] — cannot really improve past an EER of 0.2% [CTI+18, GNRW16, Rob07]. Even assuming that the user data are quite well distributed — which is far from guaranteed — this corresponds to a min-entropy below 7 bits, on par with current password systems. Moreover, most of the EER mentioned in papers on biometrics are against non-optimised adversaries: for a given set of user patterns, they check which proportion would be accepted as similar enough to another user. For an adversary that can compute an optimal distribution of fake patterns to cover the space of user data-points more efficiently, having a success high enough to get more than 10% of users despite a three strikes policy[n] seems a strong possibility.

When the adversary has a sample of the user's behaviour, this becomes even more worrisome: in 2018 McCulley and Roussev managed to fool typing biometrics with between 55% and 85% success rates depending on the user by having access to a limited sample of their typing patterns [MR18]. Even samples as small as a few dozen characters were enough to greatly limit the search space.

The weaknesses considered can be mitigated by liveness detection, but this is becoming an arms race between attackers and people who build better biometric systems to detect stolen and synthetic data, often with multimodal biometrics. So far, every commonly used biometric authentication system has been found vulnerable within a year of its release, and the issue of this arms race is uncertain, so basing all our security in these systems is not a viable strategy [Mem17]. Another option is called *cancelable biometrics*, where the data is not collected as is but is first transformed to prevent replay attacks. This might work temporarily, but will probably lead to a secondary arms race with an unknown conclusion, as the transformations used aren't always impossible to inverse [LK11, SLMM09, LK10, MYL14]. Moreover, some users are already barred from ever using biometrics securely, as with the fingerprints and facial recognition data of more than 1 million users of the Biostar 2 biometrics access system that were found to be leaked in August 2019 [RL19].

A second troubling fact comes from how we perceive and use biometric authentication. As long as the systems were rare and only used in high-security environments, they were hard to compromise. With government IDs often requiring them and 42.5% of smartphone users now using them, asking someone for that kind of data doesn't raise suspicion anymore[o]. It would be interesting to see whether the movie industry — with its many depictions of biometrics as high-security systems [Gat04] — influenced not only the popular perception of how secure the systems are but also made them easier to accept. As these systems are now ubiquitous, it becomes ever easier to massively steal biometric data. To make matters worse, even if all systems start using cancelable biometrics, nothing prevents an adversary from creating a phishing tool to extract raw data from which the cancelable biometrics can then be computed.

1.4.6 Reflexive biometrics.

As we can see, there are serious issues with static, dynamic and behavioural biometrics so far. There are a few potential solutions imaginable, although they aren't ready for deployment yet. One of them, inspired by challenge systems, consists in focusing not on

[n] Meaning that the person trying to authenticate is blocked after three failed attempts.

[o] To prove this point further, a mobile application developed by the FBI to give workout goals for prospective agents attracted some criticism because of its privacy policy in late 2018 [McK18]. The app could access accelerometers, GPS data and other local services, which was defended as entirely standard for this kind of app, showing the public acceptance of such policies.

features intrinsic to the user but instead their reactions to a given stimulus. One early candidate for that kind of system is based on brain electro-encephalography, with the brain's activity being recorded by a set of sensors and the task changing each time. A precursor to this was proposed in 2007, reaching equal error rates between 7% and 30% depending on protocol [MM07]. This was done using cumbersome equipment (caps with 32 integrated electrodes) during lengthy sessions. Systems based on this have since much improved, with error rates for complex systems going below 1% [DMC16], and single-channel systems (where the electrode is worn above the ear [ABTV11]) managing to lower EER below 10% [CYMC16] — on small samples, however. This is still far from being applicable in practice but has many advantages, including being renewable and having extremely good liveness detection [MLRC16].

A second option comes from the rich field of eye biometrics. Besides iris and retina static features, dynamic biometrics have relied on gaze trajectory [GNR+13], pupil sizes [BKMF05], or eye movement patterns [ERLM15]. Some challenge-based systems have also been developed, based on eye trajectories when shown images of human faces [REF12], or moving points on a screen [SRRM16], reaching an EER of 6.3% for a 5-second authentication time.

Surprisingly, it seems that no-one so far has investigated the possibility of mixing the reflexive attention metrics shown with eyes in pupil size biometrics with visual passwords on a challenge basis. Such a system would be very simple to operate: the user might, for example, start by memorising a set of pictures, and when trying to log on, the system shows a list of pictures, randomly taken from either the memorised list or a large database. As has been known for a long time [Lof72], the pupil dilates or not depending on the familiarity of the image[p] [Ein17, GMM15]. Besides the renewability and low complexity, a central advantage of this system would be speed: the pupil reaction is not conscious and starts within hundreds of milliseconds of the stimulus occurring [NFRE13], and effects can be seen for images shown for 30ms [Ray78]. Multimodal biometrics combining this with the previous system could also be used, inspired by work by Cody, Noton and Stark [Cod15, NS71]. All this would, however, require high frame-rate cameras — and potentially electrodes — as well as the ability to detect recognition while compensating for the previous stimuli. A third method was proposed in 2012 by Bojinov et al. [BSR+12] and is based on training a user to accomplish a specific task (in their case, a typing game). By then challenging them to a game where some of the patterns were familiar, they could detect whether the user was trained on a given pattern or not, and authenticate them. It remains to be seen whether this is viable with a high enough security level or suffers from the same problem as general typing biometrics.

We conclude this review and analysis of authentication here, having discussed the basics of password use today, as well as potential replacements. As stated in the introduction, the death of passwords might not come as soon as most are hoping, and the next few chapters will build on what was shown here to improve our interactions with security systems, starting by single-use codes. As stated before, we will seek to create systems from the bottom up. Instead of trying to set in advance a security threshold and finding ways to make users follow the protocol, we will be looking at what they can do and try to build security from that. The first step is then to look at how users type.

[p] This effect is further increased by the use of triggering images, typically ones with a violent or erotic content [BL15] [KM11], although their use cannot be advised in this kind of system. Moreover, due to the many behaviours correlated with pupil size (such as arousal or cognitive load [Ein17]), the number of factors to compensate for would increase even more.

2

Consonant-Vowel-Consonants for Error-Free Code Entry

2.1 Definitions and contributions

Before we start thinking about improving the usability of passwords, we have to consider a simpler problem: how do people enter codes into devices? This depends, of course, on how they get the code — is it on the screen, on a piece of paper, or known by heart — and what they're supposed to do with it. For now, let's focus on single-use codes, which people generally see on paper and have to type a single time. This includes, for example, Wi-Fi passwords — which every user has to type at most a few times — confirmation codes for 2-factor authentication, which are increasingly frequent and long, as well as cryptographic codes used in some electronic voting systems, as in Chapter 7.

The codes are almost always automatically generated and suffer from the same defects as randomly generated passwords — although, as they are single-use, usability has been less of a focus. As such, they've had a tendency to use bigger character sets, with upper- and lower-case characters and special characters [WM11, BDN+12].

User-created codes versus single-use or automatically generated codes present very different challenges for usability, as readability becomes more of an issue, whereas memorability is of little importance[a]. Creating codes adapted to their use, whether it is memorability, ease of entry, or speed, can greatly reduce stress and breakage in everyone's work, and trade-offs are frequent [AAB+17]. Typical systems today might require people to use at least 8 characters, upper- and lower-case, numbers, and special characters, despite works showing the low benefits of this approach [GL14, Cra16].

The studies in this chapter stem from one problem observed in a code-based voting experiment — detailed in Chapter 7 — where frustration and confusion with character recognition caused at least 10% additional abstention [Bla18].

Automatically-generated codes depend on multiple frequent assumptions that haven't been extensively questioned:

- Does increased character set complexity [SKD+16] make better codes?

This chapter is based on research done with Leila Gabasova and Ted Selker.

[a] And in some cases, even detrimental, especially for voting applications.

- Does separation of a code into multiple fragments (here called *chunking* a code [McC14, HKB+15]) with spaces in between reduce errors?
- Are nonsense patterns of alphanumerics better than syllabic codes or even words when it comes to security and efficiency?
- Are there trade-offs between length, character sets and structural patterns that can improve people's ability to use codes?
- How do these trade-offs change when considering the ability to reduce transcription errors for one-time codes?
- Can techniques for making codes easier to enter work across cultures or even languages?

Some preliminary results were already known; such as the prevalence of i–l–1 confusion, which was studied in the medical field [Gri12], but we couldn't find any systematic study on the subject, hence the organisation of the pilot experiment and its follow-up. After defining the types of codes studied and announcing the main results, we introduce the experimental protocol for the two crowd-sourced tests of usability and transcribability. We then analyse the data collected, looking at error rates, speed, participant preference and memorability.

Using those results, we show one sweet spot for the usability/security trade-off, and propose a variant that includes error detection and correction. Finally, we show how more work could be done to further explore the design of cross-cultural, easy-to-transcribe, high-entropy codes.

2.1.1 Definitions

Our experiments included randomly-generated codes composed of sequences of the following type:

- *Numeric*: numbers from 0 to 9, such as 958905239.
- *Alphabetic*: lower-case Latin letters (excluding diacritics), such as ienkzeiwa.
- *Alphanumeric*: numbers, lower-case, and upper-case alphabetic characters, containing at least one of each, such as Ok9Kh51ml.
- *CVC*s: consonant-vowel-consonant alphabetic trigrams in lower-case. Vowels are a, i, e, o, u and y. Consonants go from b to z, excluding q due to demonstrated discrimination problems between y, q and g [Gri12].

2.1.2 Main contributions

This chapter has 4 main experimental observations and one small theoretical contribution.

- Transcribing codes takes concentration and is highly dependent on the code's structure. For codes of equal length, code structure can reduce transcription error rates from 16.9% to 1.9%.

- A majority of code transcription errors can be eliminated by using a set of unambiguous alphabetic characters (excluding visually ambiguous g/q/y and i/l), eliminating mixed case to prevent upper-case/lower-case confusions, and eliminating numbers.
- The relationship between code length and time needed to enter it strongly depends on the code's structure; spaces can help for long alphanumeric codes but can be confusing for others. Using a consonant-vowel-consonant (CVC) pattern in codes can reduce time to transcribe even with codes twice as long.
- People have a 75% chance of recognising a code they had seen 2 to 5 minutes earlier. However, they will correctly reject a novel code they haven't seen in 87% of cases.

2.2 Experiment Design

The following experiments were meant to demonstrate trade-offs between character sets, number of characters, and patterns of characters to create easy-to-enter secure codes. We developed a web-based interface in Javascript to sequentially present discrete code transcription problems. We first tested it in a pilot experiment and then improved and extended it for a larger secondary experiment.

Our goal was to have people type codes in the kinds of places they typically are when using online services. To understand how codes can be improved in the wild, the experiments were conducted wherever a person was (in real-life conditions, not a laboratory environment). Our analyses generally avoid raw averages and focus on trimmed averages and medians to eliminate anomalies (such as one participant taking close to 5 hours to answer a single question).

2.2.1 Pilot Experiment

A protocol was developed that would take no more than a few minutes and test transcription of code length, character sets, and spaces. Engagement was initially solicited personally by a docent from 33 random attendants of the science fiction conference Worldcon 75 in August 2017 for a pilot experiment. The initial design did not adequately distinguish capitalisation problems and issues around the way input is entered on smartphones. Unfortunately, it also didn't correctly disable auto-correct. Despite those setbacks, the data still showed that codes following syllabic patterns had many advantages and laid the groundwork for changes to put into the experiment.

As we thought that being able to pronounce the code as we read it might make it easier to remember and transcribe, the pilot experiment explored multiple types of syllabic and pronounceable codes such as CVC and simple alternating vowels and consonants. As CVC seemed the most promising trade-off, the main experiment focused on that design to get enough accurate data.

While several of the pilot experiment's results were statistically significant, the main experiment corroborates and extends these on a larger and more diverse sample. The pilot helped validate and improve the Javascript protocol and show where more data was needed. Results below detail only the main experiment (data will be available for both studies in a public online repository).

2.2.2 General Protocol

Participants were individuals that responded to an opportunity to volunteer online. They were told that they could quit the experiment at any time. Their data was only collected (through FormSpree) if they confirmed submission at the end. The total time taken generally varied from 3 to 10 minutes.

For security as well as privacy, all code executed was on the participants' device, was visible to them, and only recorded their final answers and timestamps. The codes to transcribe were created independently for each participant, uniformly at random among all possible codes of that structure.

The study was presented as a sequence of web forms with an introduction and three main sections, designed to measure transcription performance, preferences between different kinds of codes, and ability to remember the codes shown in the first section. Screenshots from the interface can be seen in Figure 2.1, whereas a complete flowchart for the organisation of the experiment is shown in Figure 2.2.

Sections

- Welcome and basic participant information.
- Transcription: Nine codes of different length to transcribe into a prompt,
- Choice: Nine pairs of codes varying from 9 to 22 characters in length and structure (alphabetic, alphanumeric, and Consonant-Vowel-Consonant (CVC)) where the participants were asked to choose and transcribe the easiest.
- Memory: Participants were presented with seven codes and asked if they had seen this code earlier. With 50% probability, the codes were from the transcription section, otherwise, they were randomly generated codes with the same structure. Participants were also asked to give an estimate of the number of codes they had transcribed.
- Accept: Participants were asked to upload their answers and given the choice of adding their email to be kept informed.

Introduction and Basic Information

The welcome page presented the experiment as an opportunity to help research, and informed participants that they could leave at any time. It took care not to prime them with research goals. It asked their age, country, main language used, self-rating of their ability to remember passwords and strings of numbers/letters, as well as whether they were using a mobile device or a numeric keypad in the experiment. Optionally, participants could submit their email to be kept updated with the complete experiment results once a paper was published. Those emails were stored securely and separately from the data that was analysed. Participants were not asked their gender or other personal characteristics as they were not pertinent to the research.

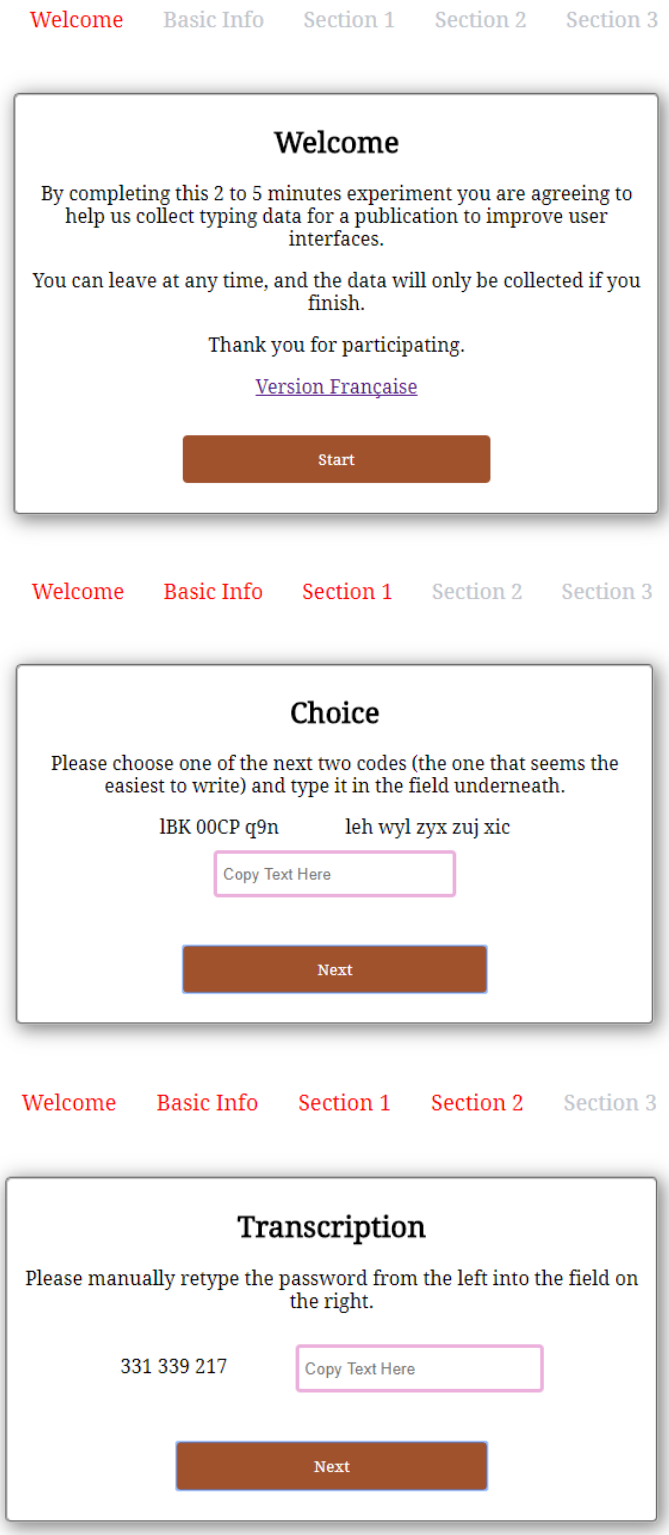

Figure 2.1: Screenshots from the experiment's interface, featuring the welcome page and one example each from the Transcription and Choice sections.

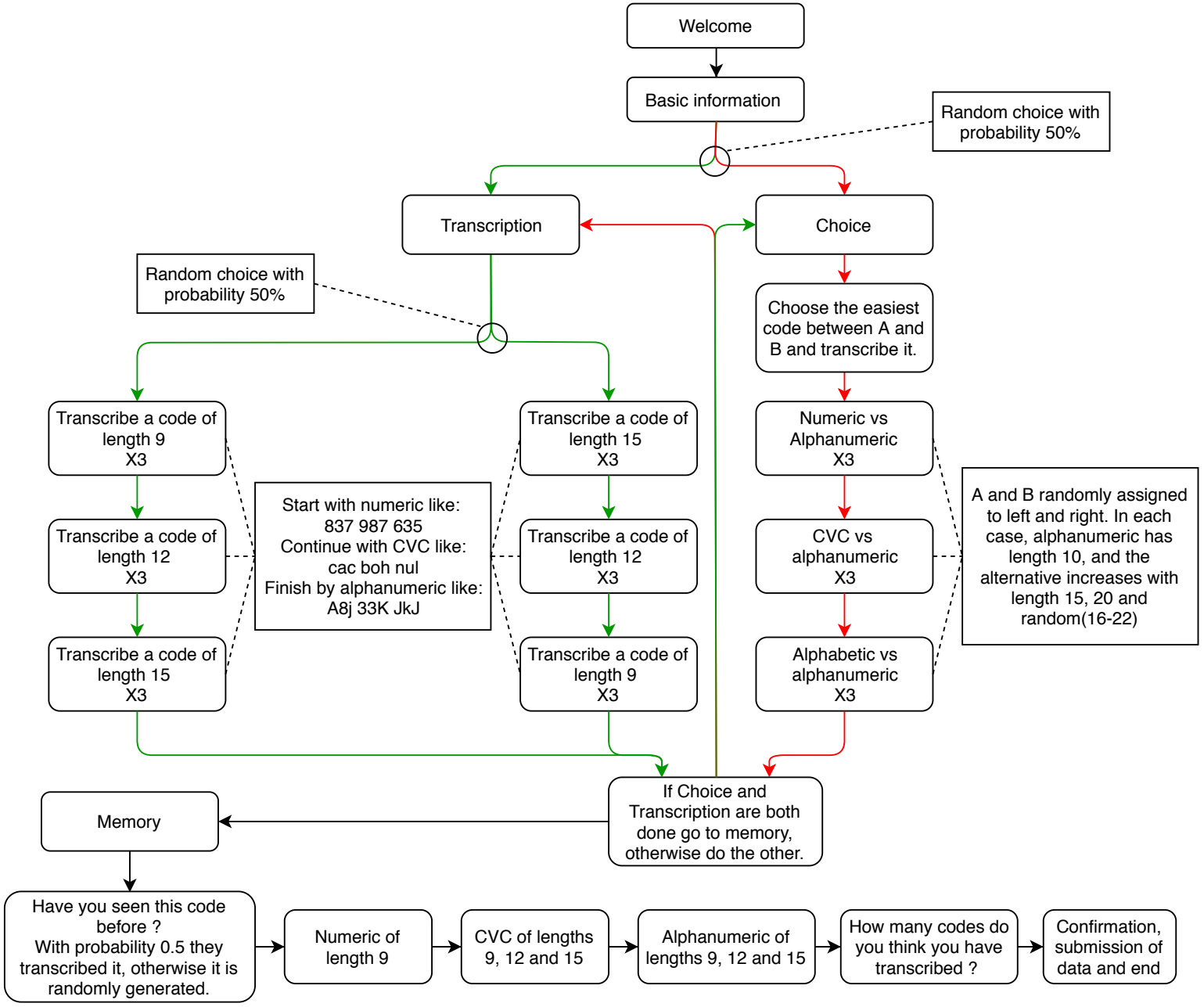

Figure 2.2: Flowchart representing the organisation of the experiment with the between-groups design.

Transcription

The codes were grouped by length (9, 12, and 15 characters), each group presenting three types of code trials in the following order: numeric, CVC, and alphanumeric. This gave a baseline error rate from which to replicate standard findings [Gri12, KB81] such as the prevalence of the g/q/y error. It also gave rates for other types of errors, allowing the comparison of different code structures. Participants were not informed of any error they might have committed and had a single try for each code. Copying and pasting were also deactivated.

Choice

Each trial included choosing to type in a 10-character alphanumeric control code or a second code. The codes were grouped by character sets used — 3 using numeric, 3 CVCs, and finally 3 alphabetic. Trials were given in order of increasing length for each type.

Memory

Every question included a code participants had seen earlier, or a randomly generated code of the same type and length, with probability 0.5 for each. The types were numeric of length 9, and CVCs and alphanumeric of length 9, 12, and 15.

Randomisation

A between-subjects approach was used to observe priming and learning. Half the participants did the Choice section first, and the other half started with the Transcription. Half the participants also received the Transcription questions in reversed order.

For the Choice section, the order in which the two codes were presented was also randomised to avoid preference for the one on the left or right. The 9 codes in the Choice section were presented in order of increasing length. The pilot experiment seemed to indicate a tipping point close to 18, so we bracketed by testing codes of length going from 15 to 22. As this phase was already the most time-consuming for the participant, having one code for each length would have made the experiment too long. Hence, we broke it up in A/B testing, giving participants two codes of length respectively 15 and 20, and one of random length between 16 and 22.

Times

Time taken was measured for each question, as well as the time spent reading the different sets of instructions. As the protocol was self-administered and self-paced, some people took breaks, ranging from a few minutes to five hours. Large delays on a single question were observed in around 15% of respondents. Taking breaks or getting distracted is part of life; long breaks alone did not disqualify all trials from the analysis. Data for each question was independently evaluated and the abnormally short or long responses were removed (top and bottom 10%). Medians were consistently $5-10\%$ under the trimmed averages and are not shown as they lead to the same conclusions.

Chunking

The Transcription section codes were split into "chunks" of 3, 4, or 5 characters followed by a space. In the Choice section, chunks of 3 were used. For lengths not divisible by 3, the last chunk had between 2 and 4 characters, and the 10-character alphanumeric codes avoided a 1-character last chunk by using a 4-character central chunk.

2.2.3 Demographics

The main experiment included 267 respondents, with some skipping a few questions[b]. Participants were solicited for the main experiment using three methods, creating three groups. The web links followed to get to the experiment identified which group a participant was in. The first group was international in scope, spread through Facebook and totalled 61 respondents.

The second was mostly French, using a translated form, and was composed in majority of software developers, as it was spread through a French computer engineering school's social network and Internet Relay Chat, with 91 respondents. Members of this group were highly tech-savvy compared to the other two groups (due to how they were recruited).

The third was overwhelmingly composed of people from the USA, with 115 respondents recruited through John Krantz's Psychological Research on the Net website [Kra19] which promotes and indexes experiments of this kind.

All three groups included a wide range of ages, with the youngest being 19 years old for the pilot and 13 for the main experiment[c]. The eldest were respectively 70 and 73 years old, with most participants between 18 and 32.

People from 24 different countries and speaking 14 languages participated, including a few who were used to scripts written from right to left. English was the most frequent language indicated (129 people), with French second (114 people), and 34 participants indicating other main languages.

The goal in this recruitment method was to avoid having anomalies coming from a bias stemming from a single recruitment process. The three groups, with their different demographics and methods of recruitment, did show some variations in their performances. However, all the effects analysed below were observed not only in the general data, but also within each group, increasing their ecological validity as they do not depend on recruitment peculiarities. The most salient difference was that group 2 took more time but made fewer errors than the other groups. This could come from a variety of things such as their supposedly higher technical expertise (being mostly computer engineering students) or different keyboard layouts. The effect is also observed when we cluster by language (although the overlap is big between those two clustering methods).

2.3 Empirical results

2.3.1 Error type and frequency

The first section acted as a control to get a transcribing performance baseline, and allowed different patterns in transcribing behaviour to be observed. Figure 2.3 shows the

[b] This accounts for less than 3% of questions and is generally caused by a double-click on the "next" button, as timestamps show the participants spending a few hundred milliseconds on a page.

[c] The three participants who were younger than 16 all came through the psychological study website.

different error types observed in both sections. The rates differ as the text to transcribe varies: in the Choice section, someone who struggles with capitalisation. Underneath are the definitions for the error types.

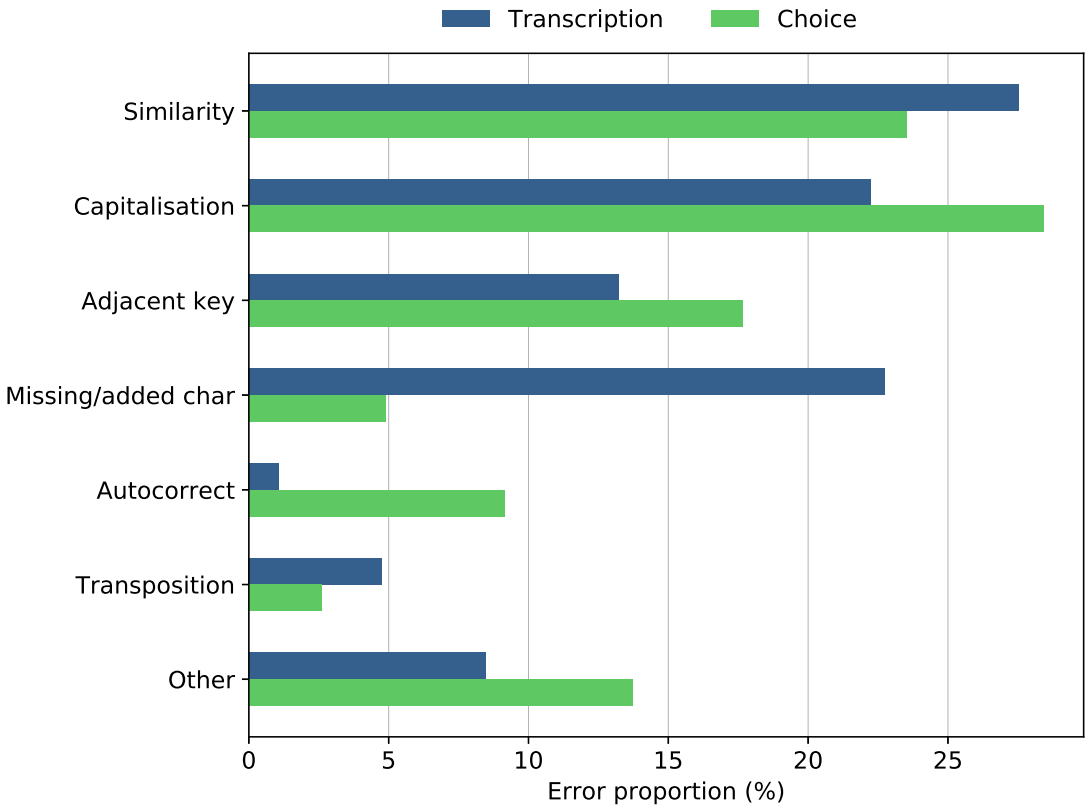

Figure 2.3: Repartition of the errors in each section as a proportion of the total errors. Differences between the sections are due to differing code length and structure, and the ability to potentially avoid certain structures that lead to specific errors.

- *Missing/added char*: a single character is either missing or was duplicated, which changes the length of the code.
- *Similarity*: confusion due to the similar shape of two characters, most commonly where one writes 0 instead of O, g instead of q or y (mostly present in the pilot), or confuses I with l and 1.
- *Transposition*: the order of two characters was reversed.
- *Adjacent key*: a key next to the target was hit, such as g instead of h. This mostly happens with horizontally adjacent keys.
- *Capitalisation*: an upper-case letter is written in lower-case, or vice-versa — this nearly only happens with alphanumeric codes.
- *Autocorrect*: despite our disabling of autocorrect via JavaScript, 2% of participants showed repeated revealing mistakes where whole words were changed, making up a large proportion of errors in the Choice section.

As we can see in Figure 2.3, similarity between characters and capitalisation make up around half of the total errors. Avoiding those characters — or using a single type and automatically correcting errors to it — and using constant case would already prevent half the errors. Using constant length codes and thoroughly disabling autocorrect would also allow to detect and or prevent between a quarter and half of the remaining errors left. We can now look at error rates as a function of the structure and length of the code to transcribe, shown in Figure 2.4. This concerns only the transcription section as it would be hard to get a good control for the Choice section: people more prone to making errors with certain structures (such as with numbers) would probably avoid choosing codes of that structure, leading to artificially lowered error rates. Although longer codes led to generally longer times, the relationship is not automatic: for example, the longest codes to transcribe were alphanumeric of length 12 and not 15.

That said, we can see that, for each length, alphanumeric codes were much more error-prone than both other structures. Compared to CVCs, the error rate for alphanumeric codes was higher by a factor between 2 and 4, depending on length ($p < 0.005$). There was a small although not statistically significant advantage for CVCs against numeric codes.

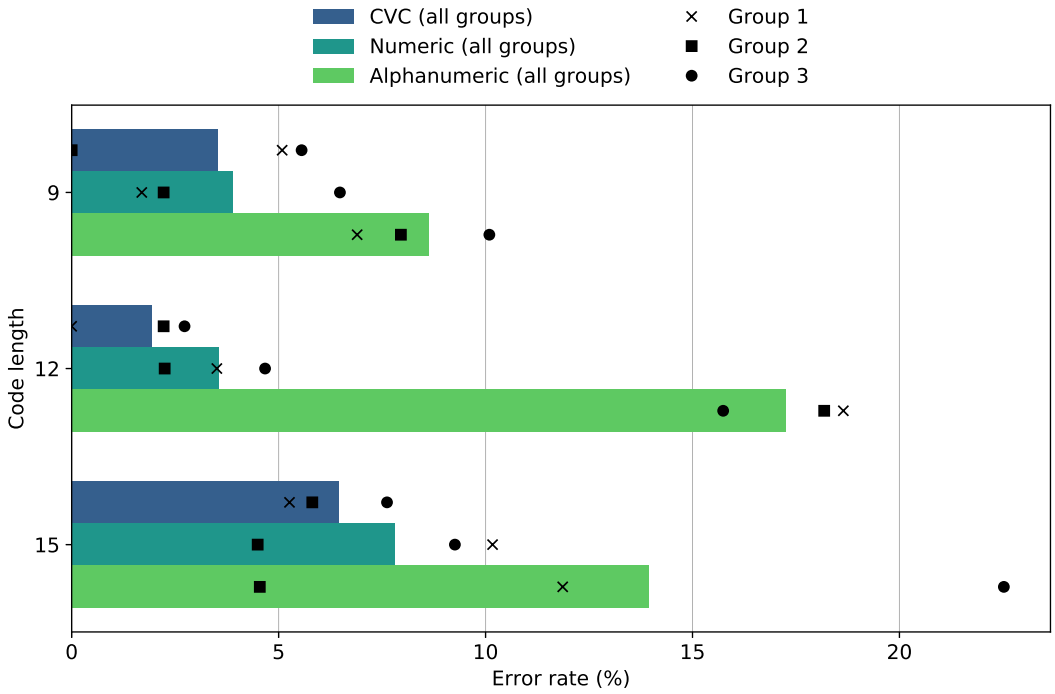

Figure 2.4: Error rate in the transcription phase, by code type and length. Bar length corresponds to global results for all participants, and black symbols represent the performance of distinct groups in each trial. To recall, group 1 is international with 61 participants, group 2 has 91 French engineering students, and group 3 was recruited online mostly among Americans, with 115 participants.

If chunking had an effect on the error rate, this effect was limited, as it barely reaches statistical significance, and only for alphanumeric codes with $p = 0.033$ — before Bonferonni correction — and a small effect size[d].

[d]Moreover, as we were looking for multiple effects and not just checking whether chunking affects error

2.3.2 Speed

Transcription section

Besides error rate, speed is the second main factor to study, especially as it impacts usability. Figure 2.5 shows the time taken for each code in the Transcription section. Once again, we can observe that alphanumeric codes are more cumbersome: they are always slower than numeric codes and CVCs, at least 42% more time than CVCs and up to 59% at length 15 ($p < 10^{-4}$)[e]. Numeric codes were also slower than CVCs, by 17% to 25%.

Unlike with errors, chunked codes had a significant effect and increased speed by an average of 8%, and up to 17% on CVCs ($p < 0.015$). This was despite the probable added slowdown, as 91% of participants who were shown chunked codes also typed in the spaces when transcribing.

There was also a learning effect, with participants typing on average 18% faster by the time they reached the end of the transcription section ($p < 10^{-4}$). This was observed no matter the order in which the codes were presented, and the between-group design of the experiment means that the effect of this on other metrics was negligible.

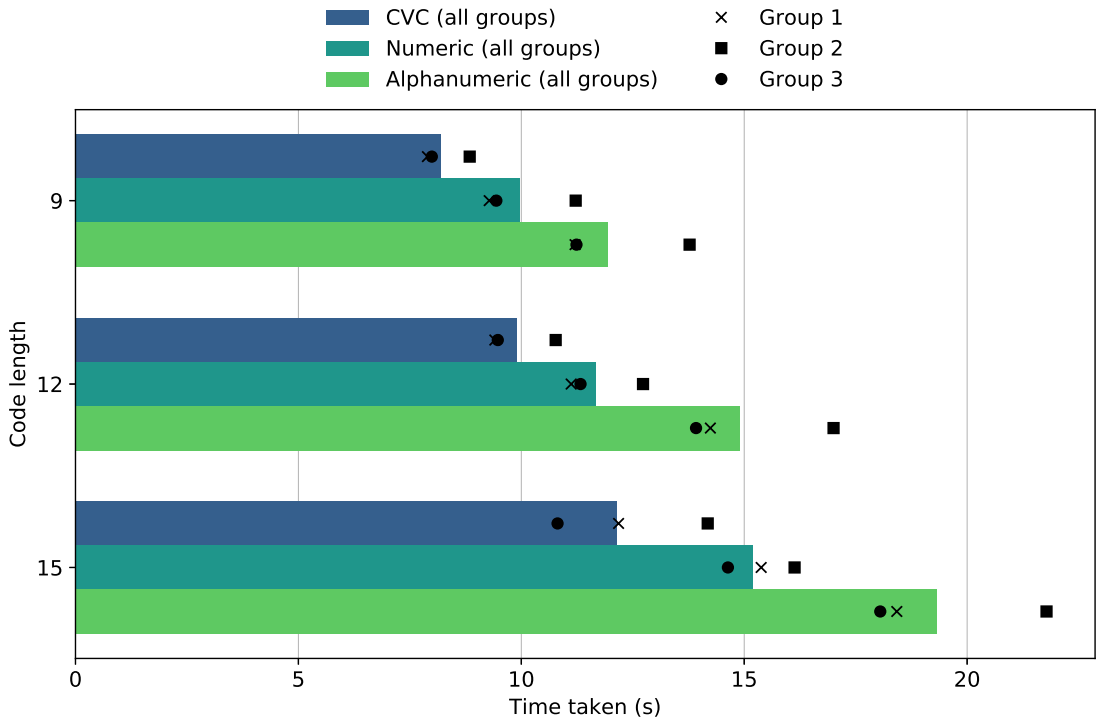

Figure 2.5: Time taken to transcribe, by code type and length. We can see that CVCs are consistently faster than numeric codes, which are in turn faster than alphanumeric codes. The relationship between CVC and alphanumeric is still true when comparing equivalent entropies and not length: CVCs of length 15 have nearly two additional bits of entropy than alphanumerics of length 9 and take the same time to be typed.

rates, this p-value should be taken with a grain of salt as it could be a simple artefact that doesn't withstand a Bonferonni correction on family-wise type 1 errors [Fra15], as opposed to most other effects shown.

[e]This whole section has very low p-values, each announced comparison between two groups being statistically significant.

Strangely, typing speed and error rate were not statistically correlated: faster typists made mistakes as frequently as the others. This is true whether we analyse them independently on each code structure, or if we look at the overall performance of each participant.

As could be expected from a diverse sample, there was a wide variability in typing speed: the top quintile hit an average of 1.34 keys per second, compared to 0.75 for the bottom quintile (within the normal bounds for non-professional typists [NF82]). The top 5% typed on average 3 times faster than the bottom 5%.

Choice section

Although it cannot be used to provide a baseline — due to both added time from choosing and a potential *specialisation effect* — we can still get some information from the speeds in the Choice section, as shown in Figure 2.6. The *specialisation effect*, in this case, comes as participants seemed to at least partially decide which code to transcribe based on their own ability to transcribe them, leading to improvements no matter what they chose — which could also be linked to improved familiarity and proficiency with the exercise.

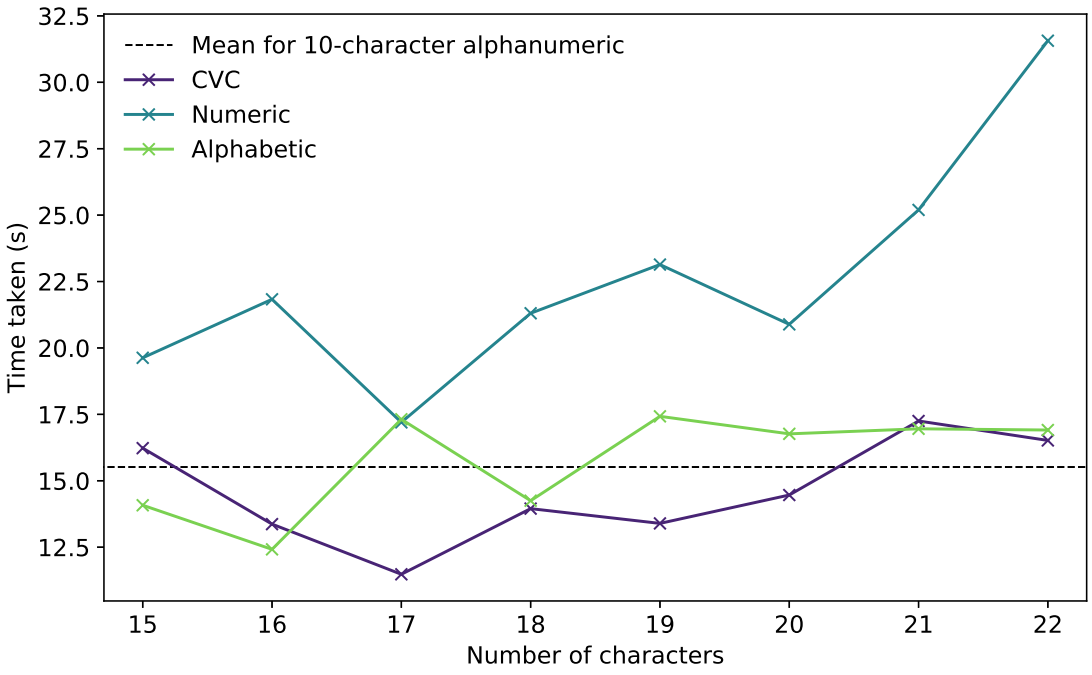

Figure 2.6: Time taken in the Choice section by participants who chose to transcribe the alternative to the alphanumeric code, depending on the code structure and the length of the code proposed. The horizontal line is the average time over all 10-character alphanumeric codes.

Figure 2.6 still features a baseline at 15.5s for the average over all 10-character alphanumeric codes, although this is a simplification. In fact, looking at the people who chose alphanumeric codes, we can see an improvement in speed, going from 20.8s in the first Choice question to 13.5s in the ninth, as we can see in Figure 2.7. Multiple factors could explain this, such as a learning effect or people focusing on what they are good at, increasing performance as they choose better options. Presenting them with long numeric codes did slow them by up to 7 seconds, even for the people who ignored those long codes.

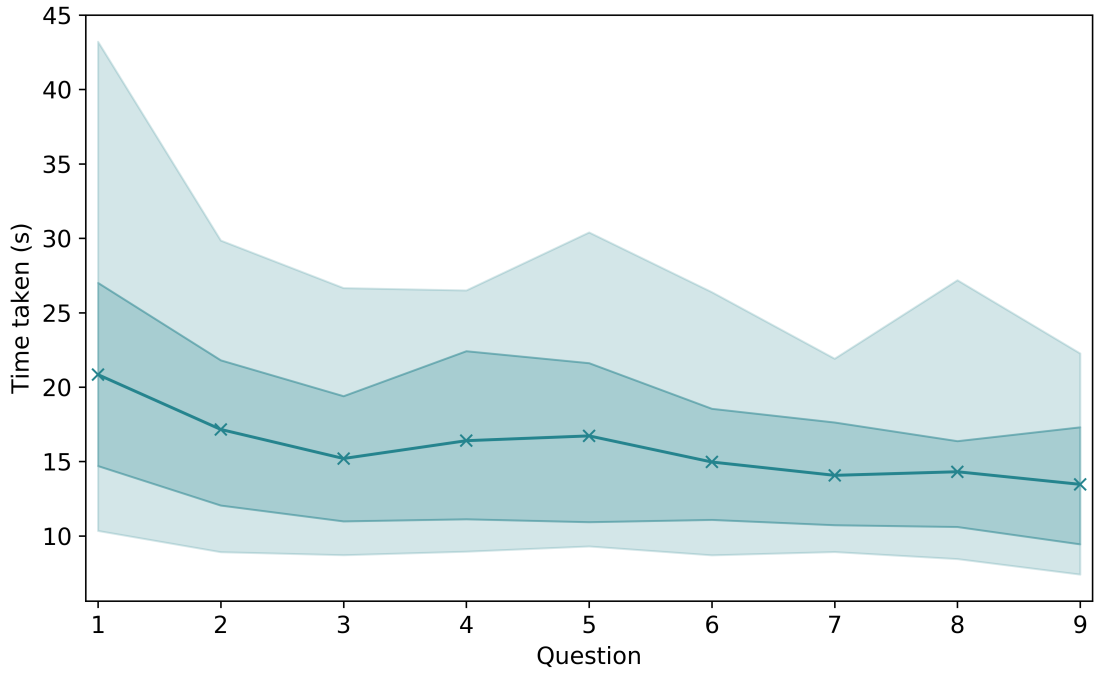

Figure 2.7: Time taken by participants who chose to transcribe alphanumeric codes, depending on which question they were on (each question having a different code structure, as shown in Figure 2.2). The central curve corresponds to the trimmed mean, whereas the shaded areas correspond to the 10th, 25th, 75th and 90th percentiles.

Despite this, we can see that CVCs tended to be faster for codes of lengths at most 20, and alphabetic codes faster for lengths at most 16. This is relevant as CVCs of length 20 are equivalent to alphanumeric codes of length 11.1 — in terms of entropy and number of possibilities, allowing for fractional length comparisons. Alphabetic codes of length 16 are equivalent to alphanumeric codes of length 12.6. Both options are then superior to alphanumeric codes in terms of speed per bit of entropy.

2.3.3 Preferences

General choices

To gauge perceived usability, we asked participants to transcribe the one they perceived as easier. This is different from performance, and people might prefer harder options, or be bad at finding the easiest one. For example, the people who chose to transcribe a 22-character numeric code took an average of 31.6s, and even the fastest, at 21.2s, was still 39% slower than the average among the people who took the other option.

Figure 2.8 shows the proportion of participants who chose the alternative to the alphanumeric code, as a function of the structure of the alternative and its length. As was expected, long codes are less often chosen. Participants strongly preferred CVCs of length at most 20 over 10-character alphanumeric codes, with rates varying between 72% and 48% in the worst case ($p < 10^{-4}$). Numeric and alphabetic codes were never really preferred,

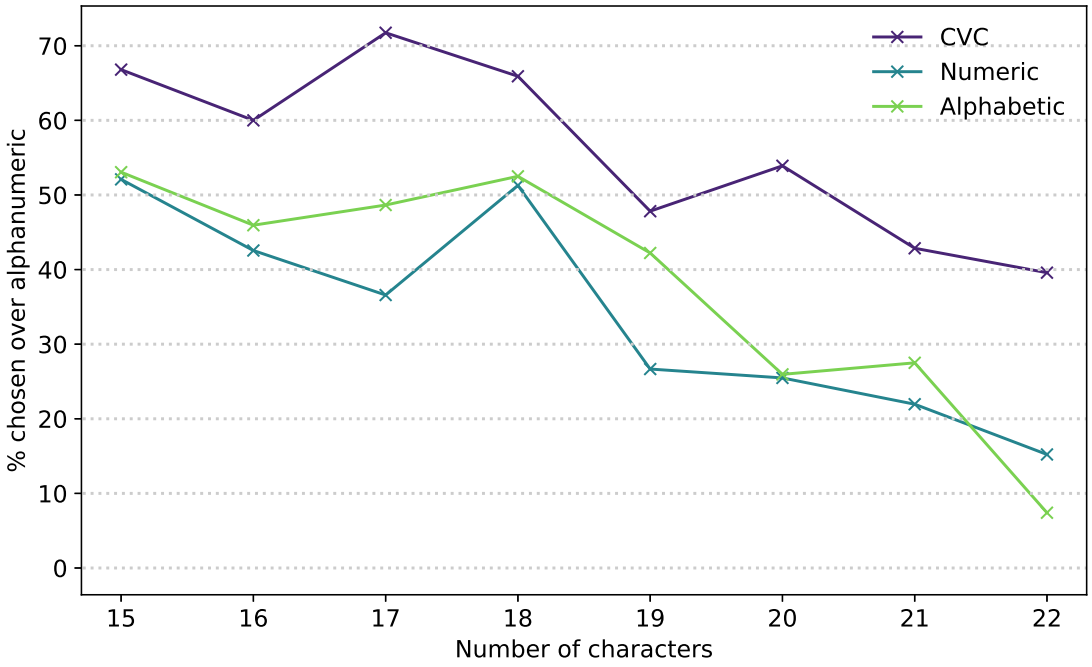

Figure 2.8: Percentage of participants preferring alternative codes to 10-character alphanumeric ones, by code type and length.

staying below 55% even for short codes, and having a steep decline from length 18 onward, ending with respectively 13.6% and 7.4% choosing numeric and alphabetic codes of length 22 over the alphanumeric ones.

Strategies

Many people appeared to follow a strategy to choose which code to transcribe. Across 267 participants, we identified 121 different patterns, of which here are the 5 most common ones (accounting for more than a third of participants):

- 31 people always chose the alphanumeric.
- 24 people chose the alphanumeric for all cases but one (either short or mid-length CVCs or numeric).
- 18 people only chose the alphanumeric against numeric codes.
- 12 people only chose the alphanumeric in one case.
- 11 people never chose the alphanumeric.

2.3.4 Memory

One last goal of this experiment was to check the memorability of the codes shown. Table 2.1 shows the recall rate for each of the different code structures.

Error type	NUM9	CVC9	CVC12	CVC15	ANUM9	ANUM12	ANUM15
Type 1	28.6	39.0	6.5	19.0	40.1	18.4	25.7
Type 2	15.7	18.3	10.9	9.4	17.7	6.4	5.8
Total	22.5	28.8	8.6	14.4	29.2	12.0	16.7

Table 2.1: Error rates (in percentage) for each code structure and length, in the order they were shown to the participants. Type 1 errors are false positives (participants stating they recognise a code they hadn't seen) and type 2 are false negatives.

There are a few main observations to make. First, type 1 errors were way more frequent, with 25% false positives overall, against 13% false negatives ($p < 10^{-4}$). This is probably strongly due to the fact that the exercise wasn't trying to mislead participants, and unfamiliar codes were randomly generated from scratch instead of being closely related to codes they had seen, and had at most a few characters in common. As such, longer codes participants hadn't seen were easier to discard. There was also a strange recognition peak at length 12 ($p < 10^{-4}$), for which we have no explanation.

When asking participants to estimate the number of codes they had written, the results did not converge toward the real value. They were relatively precise when we look at the trimmed average (18.7 for a true value of 18), but not with a simple average or a median (both at $20.0 - 20.1$). This is due to a large variance, a strong tendency to write 20 (more than a quarter of participants did so) and the 8% of people who overestimated by a factor between 2 and 6. When asked to estimate the number of questions, there was also a tendency to answer with multiples of 5, as done by 76% of participants.

Finally, participants' recognition ability was correlated with self-rating in the memory section, with two cohesive clusters appearing. The first was around an average 28% error rate for people who rated their memory 1 or 2 out of 5. The second was between 16% and 18% for those who rated it higher ($p < 10^{-4}$).

2.4 Creating better codes

2.4.1 A simple solution: CVC^6

The main finding from this experiment is that codes following the Consonant-Vowel-Consonant structure are faster to type and more accurate than random alphanumeric ones — as are probably other types of pronounceable codes. Not only that, but the magnitude of the effect is such that it renders longer codes a viable alternative, even in contexts where security is the objective. This leads us to propose using sets of 6 CVC trigrams — denoted as CVC^6 – to replace most medium-length alphanumeric codes, as they have many advantages, as stated below.

Security

From a security standpoint (for use as passwords), a randomly generated CVC^6 has high entropy, with 1.03×10^{20} total possibilities, or 66.5 bits of entropy[f]. This is superior to 8-10 character alphanumeric codes by at least a factor 100. Those have at most 48 and 59.5 bits of entropy.

Speed

Despite its length, CVC^6 is faster to type than other codes of similar or lower entropy. In the Choice section, CVC codes demonstrated average and median speeds higher by 10% to 80% compared to equivalent code structures. This is despite entropies being as low as 59.5 bits for alphanumeric and 50 bits for numeric (the complexity/speed trade-off meaning that a lower entropy generally implies a faster typing speed). Moreover, nearly two-thirds (66%) of the study's participants perceived CVC^6 as easier than equivalent alphanumeric codes, as opposed to a maximum of 54% for other types of codes.

Errors

11 of the 31 errors present in the transcription phase of CVC were preventable by checking the length. An additional 4 could be automatically fixed by checking whether the letter typed was a vowel or a consonant. Among the 106 errors found in alphanumeric codes, only 7 were preventable in such ways. The error rate, already more than a third lower in CVCs than in comparable codes, can then be improved even further. CVC^6 can get under 5% error by eliminating the following sources of error:

- Capitalisation errors, as the code isn't case-sensitive.
- Symbol confusions, which would almost entirely disappear, leaving only v/w (which is very rare).
- Thanks to the alternating consonant and vowel pattern, character deletion and transposition would be immediately detectable by the system and visible to the user. This would also apply to 10% of near misses.

Finally, this can be additionally improved by handling error correction, shown below with the improved CVC^{6++} approach.

2.4.2 Error correction with CVC^{6++}

Getting an error when typing in a code frustrates most users, and not being able to find its location even drives some to abandon whatever task was at hand [Bla18]. One improvement of considerable value would then be for the system to detect an error and point it out, possibly indicating what the error was.

This would eliminate mistakes CVC^6 is vulnerable to (mostly near misses and phonetically similar characters). It would have eliminated all of the 495 errors in this chapter's experiments. Only double or triple errors wouldn't be corrected, and the three double

[f] This naturally follows Kerckhoffs's law with the adversary knowing the format of the code used. Against a blind brute-force on same length codes, the number of possibilities jumps to 2.95×10^{25} possibilities or 86 bits

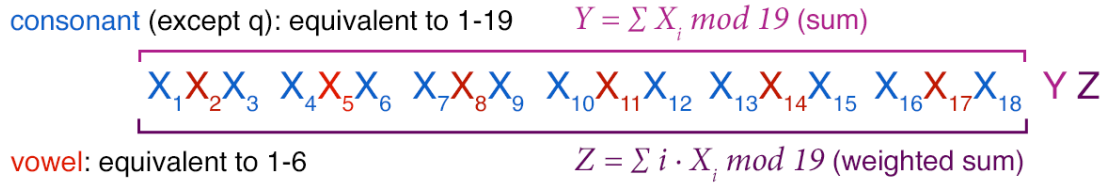

Figure 2.9: Error correction in CVC^{6++} . The last two characters, Y and Z, respectively correspond to the sum and weighted sum of the previous 18 characters.

errors observed in the transcribed CVC codes would have been detected correctly using length considerations. Error detection, localisation, and/or correction would reduce user confusion and input time. A natural extension to CVC^6 , CVC^{6++} achieves all of those.

Protocol

The extended error detection/correction protocol is shown in Figure 2.9 and works as follows:

- To add correction without compromising on entropy, one last chunk "YZ" of two consonants after the last trigram is added to the code.
- To detect, localise, and correct the error:
  - Values from 1 to 19 are assigned to each consonant: b = 1, c = 2, d = 3, etc. Since consonants and vowels are not used in the same position, numbers can be reused by vowels: a = 1, e = 2, i = 3, o = 4, u = 5.
  - Y is computed by summing all the values modulo 19.
  - Z is computed by summing all the values, multiplied by their position in the code, modulo 19.

Suppose that there is an error concerning a single character in position $i \leq 18$ (i.e. the error is not on Y or Z). If the value of the entered Y differs from the sum computed from the input, the error is detected. The system can then get the difference d between the character entered and the correct one. The difference between the computed Z and the Z' entered with an error, $d \times i \mod 19$ is also computed.

The combination of those two directly shows the unique possibility for a single-character error. This is where having a base 19 system is crucial, as only prime bases allow this (as the multiplication modulo 19 is bijective). As it happens, having a code whose body — without the error correction — has as its length the size of the alphabet minus one is one optimal point.

In the case where the single error concerns Y or Z, the value of the other matches the computation. This can only happen when either a single error is on Y or Z, or there is at least a double error. Assuming the latter isn't happening, the system then knows that the error concerns either Y or Z (and can safely ignore it).

Advantages and Limitations

The obvious advantage of CVC^{6++} is that it allows the system to automatically identify/correct the code and avoid wasting the time of a frustrated user. This automatic correction should not be used where correcting a double error into a different code would be strongly detrimental, such as voting, as it could — with very low probability — change the decision on a cast ballot. Instead, the system could indicate the location of the error to the user, to allow them to quickly check and correct it themselves.

The main limitation of CVC^{6++} is that it cannot work as naturally for CVCs longer than 6 trigrams, as multiple conflicting correction possibilities would appear. Probabilistic error correction could solve this, or an extension of the last two letters to a larger character-set. Its length (for the same level of security) would also make it less popular than CVC^{6} , although a majority of participants in the experiment would still prefer it to alphanumeric codes.

2.5 Discussion and future work

2.5.1 Use in voting

We've seen how code structure and length all affect speed, error rate and preference, and proposed one code structure as a result. Besides the properties already mentioned, we would like to point out two features that make it especially suitable in the context of electronic voting. The first is that CVC^{6++} can be most helpful in proxy voting, where one person can vote by giving a code to a trusted third party — systems like the one motivating this study, which are presented in Chapter 7. One problem is that this third party does not have access to the list of valid codes and would normally have no way to notice if something went wrong during the initial transmission of the code. By making sure that it is correct immediately, they can wait for the vote to happen confident that they have the correct information.

A second advantage is that the CVC syllabic pattern is present — with generally high frequencies, although less so than the CV pattern — in most Indo-European languages [AM10, Sch99, BMLZ17]. This is relevant as, instead of syllabic patterns, whole words could be used, as with passphrases [KSS07]. Although using whole words would boost speed and reduce errors even further, it would also give a real advantage to people more proficient in the language in which they are given. As knowingly imposing a higher difficulty to vote correctly on certain demographics would be problematic — and often illegal — passphrases couldn't easily replace the codes used for voting. On the other hand, CVC^{6++} would be a strong improvement without giving a measurable disadvantage to any large demographic group[g].

2.5.2 Future work

This study raises new questions on transcribing ability and code structure. Follow-up experiments could be motivated by the questions below:

- Fonts have been shown to strongly impact reading ability [BLM01]. What is the impact of font, spacing, and case on codes?

[g]It remains to be seen whether other linguistic groups — especially ones that do not use the Latin alphabet — would be penalised unduly.

- Is there a cost associated with not typing spaces? Is the speed increase for chunking hampered by having to enter an extra space character? Why doesn't it increase transcribing ability? How would chunked input zones affect it?
- Other surface features also have important effects on memory and language learning [Ste12]. How would the colour and texture coding of chunks affect transcribing ability?
- Different syllabic patterns, such as CCVC or CVCC, have higher entropy but are less frequent and even absent in certain languages [AM10, Sch99, BMLZ17]. Could they constitute viable alternatives to CVC and would they be less language-dependent? Even further, could chunks made of real words be used, and would they be worth the entropy loss for English speakers?
- Some letters (like q or x) being less frequent in many languages, would transcribing ability increase with an even smaller alphabet? Could this compensate for the entropy loss?
- The memory performance measured purposefully avoided tricky codes that were close to ones the subject had seen. What makes codes distinguishable? For goals of privacy, can easily transcribable but not memorable codes be formulated?
- The different error patterns shown are quite predictable, and could potentially be used for a CAPTCHA system where the error would be human. Could one game such a system?
- What is the impact of differences in typing ability among people who are used to a different alphabet (such as Ge'ez, Hiragana, or Cyrillic), non-alphabetic languages (Mandarin Chinese) or right-to-left writing systems?

With this first study on general transcription done, it is time to look at authentication in its most well-known form: passwords. We have observed that typos are quite frequent, and they are not always preventable. Following the ideas of the previous section, the first step will then be to see how to correct them, before looking at memory issues.

3

Secure and Efficient Password Typo Tolerance

3.1 Constraints and contributions

The previous chapter has introduced one way of making easier one-time codes, useful for some service providers although of secondary importance compared to the biggest problems: password-based authentication. Although an important amount of research has been accomplished recently to improve both the security and usability, web services still allow bad passwords and keep ignoring best practices on how to secure password databases [Gre17, JPG+16].

Naturally, as our online accounts become more and more important in our daily lives, being refused access is increasingly frustrating. A recent study showed that forgetting one's password is perceived about as frustrating as forgetting one's keys [Cen14]. Moreover, just as users sometimes forget their passwords, they often mistype them, making typos and getting accordingly frustrated. To prevent some of this frustration and improve usability, some services like Facebook have decided to accept incorrect passwords that correspond to certain specific typos, starting years ago. This generally concerns only two or three extremely frequent typos, such as forgetting that caps lock was activated, or a mobile device capitalising the first character of a password [Pro11, Lam12].

In an innovative paper in 2016 [CAA+16], Chatterjee et al. developed the first generic typo-tolerant password checker and performed a study to analyse typo patterns. They discovered that a vast majority of authentication failures comes from a few simple typos, and that it could turn 3% of the users away (on Dropbox). Although it came with next to no security degradation, this first typo-tolerant password checker could only correct about 20% of typos made by users and was computationally intensive. Together with some colleagues, they then developed a second system called TypTop [CWP+17], that is personalised to the user, more efficient both computationally and memory-wise, and corrects up to 32% of typos. This system works by keeping a cache of allowed password hashes corresponding to the frequent typos made by the user, and updates this cache at each successful authentication. Finally, Woodage and some of the original authors created a new distribution-sensitive scheme that adjusted the error rate and hashing time, improving the resistance to certain attacks and providing better time/security trade-offs [WCD+17]. Those systems can actually have a positive impact on security as they increase the usability of long passwords — which are more error-prone, especially when randomly generated as in Chapter 2 — and lowers the cost of using highly secure passwords.

One issue with the schemes proposed is that they are technically complex, which often creates difficulties in the implementation, a regular problem in practical security [SB12, HB16]. As such, we wondered whether similar performances could be attained with simpler designs, and how to create a system that increased the usability even further, while satisfying the following constraints:

- *Usability*: the system should correct as many legitimate typos as possible, and improve users' experiences, by making it easier to log into a service without delay.
- *Security*: the system should resist against known attacks on passwords, and offer performance similar to the usual method of comparing password hashes.
- *Efficiency*: the system should have a low impact on the server, requiring as little computation and storage as possible. It should also avoid having multiple rounds of communication to limit the delay.

Main contributions. Based on a completely different design, we introduce multiple simple typo-tolerant frameworks, building up to one complete system that satisfies the different constraints mentioned. It improves usability by correcting up to 57.7% of total typos, or up to 91.2% of acceptable typos. It is efficient, requiring limited client-side and next to no server-side computation, as well as low communication bandwidth and limited storage. It is simple, being easily implementable and compatible with other systems, as well as being retro-compatible with other frameworks. Finally, it is secure, limiting the risks of both credential spoofing and credential theft.

We also introduce a simple metric called the *keyboard distance*, and a protocol that can compute this distance (as well as the Hamming distance) between a queried string and a secret string, without it being possible to find the secret string in polynomial time (assuming the security of the discrete logarithm). This technique is related to fuzzy extractors [DORS08], which have been used to tolerate errors in biometric authentication systems without revealing information [BCT07], as well as the modular subset product problem very recently defined by Galbraith and Zobernig [GZ19].

Finally, we show some simple lower bounds: for a large set of edit distances, given an oracle to that distance between a secret string and a queried string, it is possible to get the secret string in a polynomial number of queries. This holds even in the case of probabilistic and approximate distances. This means that any general typo-correction system that can compute arbitrary distances is vulnerable in at most a polynomial number of queries. The previous protocol's resistance to this method comes from having queries of non-uniform — and potentially exponential — complexity.

3.2 Typo-tolerant frameworks

The goal of this section is to provide different frameworks that work in practice, being efficient, secure and easy to implement. Here, a framework is a set of three algorithms: one to create a password (*key-setting*), one for the user to compute and send their password to the server when asked their credentials (*key-sending*), and the last one for the server to check whether the credentials received should be accepted (*key-checking*). Moreover, frameworks should work with a variety of typo tolerance policies, such as only accepting capitalisation errors, or only certain forms of keyboard proximity errors (accepting an "r"

instead of "e", but not a "d"). For example, the simplest imaginable framework (which tolerates no typo) hashes the password with a salt and sends it — in both the key-setting and key-sending — and checks whether the value corresponds to the stored hash. The simplest efficient typo-checking framework would consist in storing the value for both the hash of the normal password and the hash of the string corresponding to typing the same password in caps lock. By checking whether the hash received is one of those two, around 15% of typos could be handled by storing and comparing a single additional hash [Pro11].

Given a set of allowed typos, there are two naive frameworks which accept exactly the passwords with an allowed typo — with an exception for improbable hash collisions. The first works by storing the hash for every possible allowed password and checking whether the one that was sent is in the list. Equivalently[a], in the second, the user could compute all possible hashes and then send them for the server to check whether the stored one is in the list. The problem with both methods is that they require either the transmission or the storage of $\Omega(k \times n)$ hashed passwords, where k is the number of allowed typos on each character. For example, allowing just the neighbouring keys and single capitalisation errors, passwords of length 12 already require the storage of more than 100 hashed passwords. Also allowing inserting certain characters (by pressing a key twice for example) requires more than 1000 stored hashes, making the systems less efficient and more vulnerable to random collisions.

After showing an analysis of the most frequent user errors, we'll introduce three frameworks, each building on the previous one, to get typo-tolerant password checkers that can handle substitutions, transpositions, and finally insertions. A final complete framework will then be shown which integrates all the features[b].

Throughout the chapter, n will be the length of the passwords considered. To make things simpler and more adaptable, the frameworks are built to work using two primitive functions which they call multiple times, HASH and PRNG — for deterministic Pseudo-Random Number Generator. The security analyses will focus on Argon2 as HASH and SHA-3 as a PRNG, although other cryptographic hash functions and PRNGs could be used if vulnerabilities were found in the ones mentioned[c].

3.2.1 Typology of errors

As motivation for the first error-tolerant password checker, Chatterjee et al. ran an experiment using Mechanical Turk to look at the types of errors committed by users typing other people's password. They published a summary analysis with their algorithm in [CAA+16] and made the data publicly accessible. In a follow-up paper [CWP+17], they also ran a second study where users were asked to repeatedly enter a password over time (at intervals of at least one hour)[d].

[a] There is one difference between the two methods: the server can't easily check whether the hashes in the query are all computed from the password, giving a potential speedup to an adversary who would send hashes of different dictionary passwords in each query.

[b] The three initial frameworks have security vulnerabilities that are corrected in the complete framework. They can still be used, although care must be taken to make the same modifications to address those vulnerabilities.

[c] The main constraint is that the PRNG should be secure on correlated and non-uniform inputs.

[d] It would be hard to get better ecological validity than their study without using real-world data — which would be problematic. That said, the modalities of this second study still aren't entirely fitted to study normal password mistakes, as the high frequency of input changes the probability of having an approximate recollection of the password.

Typo category	Typo %
Single substitution	31
Caps lock	14
Single insertion	12
Single deletion	12
Double substitution	10
Shift first char	4
Single transposition	4
Double insertion	3
Double deletion	3
Other	8

Table 3.1: Types of typos in second data-set extracted from [CWP+17]

Typo category	Wrong password %
Single substitution	29.7
QWERTYnumpad neighbour	14.0
Single shift	8.5
Single deletion	19.4
Caps lock	14.7
Single insertion	13.1
Space	2.0
Duplicated letter	3.8
Single transposition	3.9
Other	19.0

Table 3.2: Types of typos recomputed on the original data-set from [CAA+16], over all passwords at distance at most 6 from the correct version, plus complete capitalisation errors.

We ran a more detailed analysis of the first data-set, shown in Table 3.2, under Table 3.1 from [CWP+17] to compare the performance and error types in the two settings. Two main questions arise when looking at such data: which errors are legitimate typos, and which legitimate typos should be corrected. In both the original study and the follow-up, the authors chose to only look at strings whose *Damerau-Levenshtein distance*[e] was less than 2, as well as errors where the caps lock was inverted for the whole string. Considering the length of passwords in the database, we chose to look at Levenshtein distances up to 5, discounting transcription errors — such as confusing "1" and "l".

Of the errors shown, some would probably not happen in the real world: inserting spaces and transcription errors — both of which probably came from a reading error and not mistyping — were removed from the data-set. Although the numbers vary somewhat between the data-sets, one common finding is that handling caps lock as well as single substitution, transposition, insertion and shifting errors would handle between 65% and 73.9% of errors.

In this chapter, *acceptable typos* will include all typos at distance at most 2, except ones involving deletions or substitutions by a distant character. We chose to exclude both, as deletions would greatly increase the risk of targeted attacks as shown in the next section,

[e]The Damerau-Levenshtein distance is one kind of edit distance between two strings, allowing four kinds of operations: deletions, insertions, substitutions, and transpositions of two adjacent characters [Dam64].

and to only allow proximity substitutions. Such a substitution happens when the key pressed is one of the six closest keys to the original one (two above, two below, and one on each side). The sets of *allowed typos* accepted by a framework will depend entirely on the framework and will be subsets of the *acceptable typos*. The rest of this section will present frameworks that can correct the most frequent acceptable typos.

3.2.2 General intuition

This section presents four different frameworks, each building on the previous ones, correcting more errors at the cost of increasing storage and computation. The first framework addresses the most frequent typo: single adjacent substitution errors, where one character in the password is replaced by another neighbouring character. For example, an "e" could be replaced by an "r", a proximity error that should be accepted, whereas replacing "e" by "m" should lead the algorithm to reject. As with all subsequent algorithms, it relies on the agreement over a canonical *keyboard map*, assigning every key-press to an integer. For example, one could use the JavaScript key codes, whose main list goes from 8 till 255, but less than 100 of those numbers correspond to usual keys. The keyboard map is not directly dependent on the layout but is instead a map from key-presses (such as "a" or "SHIFT+a"). A more detailed analysis of layout problems is done in subsection 10. The second framework addresses transpositions, and the third adds insertions. Finally, the complete frameworks addresses more practical considerations but aims to be directly implementable. All the frameworks share the same structure, in multiple ways. First, they are all made from three different algorithms: one to create/set the initial key, one to send it from the user's side when authenticating, and one for the server to check whether the authentication attempt is legitimate. Second, they all rely on the following architecture:

- The password of length n is split into n partial passwords, each missing one character.
- Salted hashes are computed for each of the partial passwords.
- Pseudorandom permutations within the set of character codes are computed (generally $[0; 255]$, based on the hashes.
- Each excluded character and all the adjoining ones on the keyboard are encoded using the corresponding permutation.
- The user sends the authentication message, a list of n pairs of (hash, number list).
- If one hash is correct, and the stored number is in the corresponding list, the server authenticates the user.

With this structure in mind, we can start looking at the first framework.

3.2.3 Substitution tolerance

Intuition for the general scheme

The substitution-tolerant key-setting algorithm (Algorithm 3.2, and Figure 3.1) works by creating hashes of every substring of length $n-1$ — which we denote as the P_i, where $i \in [1; n]$ is the position of the missing — or extracted — character. This means that, for any substitution of a single character, one hash — not containing that character — will

be correct. It would be possible to stop the algorithm here and send the list of hashes, but this would allow specific kinds of targeted attacks examined in Section 3.3. Sending — or storing — the remaining character in plaintext would give a potential adversary too much information, and even the whole password if this is done for all characters independently. For this reason, the remaining character is also sent, although in an encrypted fashion, with the key depending on the other characters. Here, this encryption is just a pseudorandom permutation, entirely determined by the other characters and computed lazily through Brassard's virtually initialised array algorithm, which is explained in the design choices [BK88].

The key-sending algorithm (Algorithm 3.2) works in a similar fashion to Algorithm 3.2. However, instead of sending the hashes of every P_i and the image of the extracted character through the permutation, it sends the image of the extracted character, its shifted version (inverted case) as well as its neighbours on the keyboard. Finally, Algorithm 3.3 checks that the full hash (H_0) is correct. If it isn't, it checks that at least one partial hash is correct and that the corresponding extracted character is among the ones allowed.

Sample run

Let's run the algorithm once to make its functioning clearer, and let's suppose for the sake of the example that the login — from which the salts are computed — is *admin* and the password *PassWord*. As long as we are in low security, we can also use a truncated MD5 as both the hash function and the PRNG.

Let's simulate the key-setting phase:

- The salts are computed, giving $S_0 = \text{truncated\_MD5}(\text{"admin0"}) = 62f04a011$, and similarly $S_1 = e00cf25a$, $S_2 = c84258e9$.
- We then get that $P_0 = \text{assWord}$, $P_1 = \text{PssWord}$, and so forth.
- We also get, directly from that, that $H_0 = aa464a06$, $H_1 = cb746ed3$ and $H_2 = c772759e$, and so forth.
- The random bits are computed (this is simulated here as it is very dependent on the implementation).
- The K_i are computed, giving $K_1 = 25$, $K_2 = 112$ and so forth.
- The hashes and the K_i are then sent and stored on the server.

When the user tries to log back in, if they type their password correctly, then H_0 is sent, compared to the stored one, and the user gets accepted. Let's now suppose that they try to log in while typing "PAssWord".

- With the new computation, $H_0 = 1686ed9ac$ which does not correspond to the stored value. $H_1 = 4196a818$, also not corresponding, but $H_2 = c772759e$ and is the same as the stored one.
- As π_2 uses the same random bits as the one computed during the key-setting, the image of a is still 112. Computing L_2 gives another number as the first element of the list (corresponding to A instead of a), but gets 112 for the second element of the list, and numbers corresponding to "Q", "W", "S", "Z", and "|".
- The key-checking algorithm checks that $112 \in L_2$ and authenticates the user.

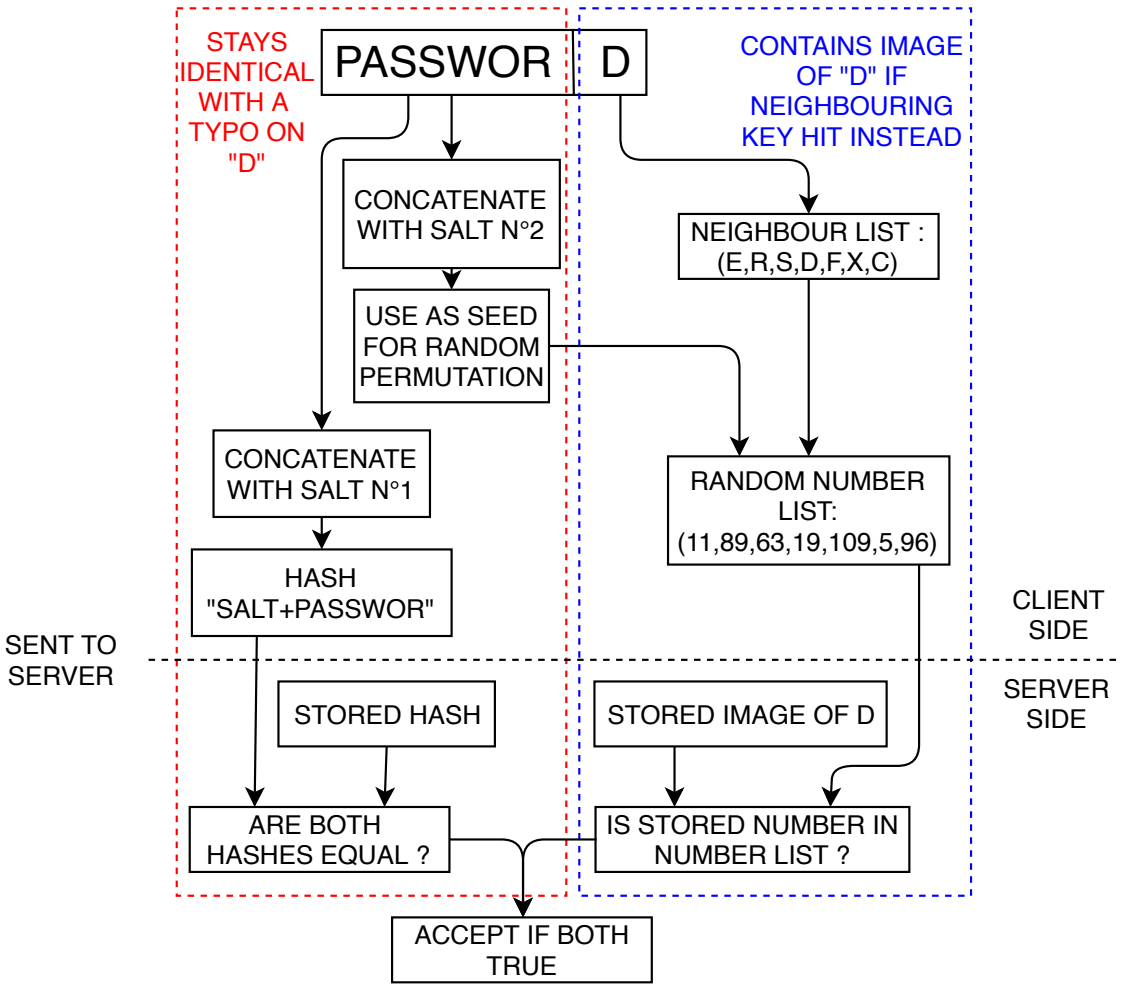

Figure 3.1: Diagram for the substitution tolerant algorithm

Data: Salts S_0, S_1, S_2, Password P of length n
Keyboard map M: Keys $\rightarrow [0; 255]$
Hash function HASH
Pseudorandom number generator PRNG
Result: Main hash and list of n (hash / integer) pairs

```
1  begin
2      H_0 ⟵HASH(Concatenate(S_0, P))
3      for i from 1 to n do
4          P_i ⟵ P \ P[i]   /* Removing the i-th letter from the string    */
5          H_i ⟵HASH(Concatenate(S_1, P_i))
6          Random_bits ⟵ PRNG(Concatenate(S_2, P_i))
7          π_i ⟵ Brassard(Random_bits)
8           /* The (pseudo) random bits are used lazily each time the
               permutation is called for a new number                    */
9          K_i ⟵ π_i(M(P[i]))
10     return (H_0, (H_i, K_i)_{1≤i≤n})
```

Algorithm 3.1: Key-setting substitution-tolerant algorithm

Data: Salts S_0, S_1, S_2, Password P of length n
Keyboard map M: Keys $\rightarrow [0; 255]$
Hash function HASH, Pseudorandom number generator PRNG
Result: Main hash and list of n (hash / integer list) pairs

```
1  begin
2      H_0 ⟵ HASH(Concatenate(S_0, P))
3      for i from 1 to n do
4          P_i ⟵ P \ P[i]
5          H_i ⟵HASH(Concatenate(S_1, P_i))
6          Random_bits⟵ PRNG(Concatenate(S_2, P_i))
7          π_i ⟵ Brassard(Random_bits)
8          L_i ⟵ [π_i(M(P[i]))]
9          L_i.append(π_i(M(SHIFT(P[i]))))
10         foreach a ∈ Neighbours(P[i]) do
11             L_i.append(π_i(M(a))
12         L_i.sort()
13     return (H_0, (H_i, L_i)_{1≤i≤n})
```

Algorithm 3.2: Key-sending substitution-tolerant algorithm

```
Data: Length n, Original list (H_i, K_i) of (hash/integer) pairs
      received list (H'_i, L_i) of (hash / integer list) pairs
Result: ACCEPT if and only if the password corresponds to the correct one or a
        version with an acceptable typo.
1  begin
2      if H_0 = H'_0 then
3          return ACCEPT
4      else
5          for i from 1 to n do
6              if H_i = H'_i then
7                  for j from 1 to |L_i| do
8                      if L_i[j] = K_i then
9                          return ACCEPT
10     return REJECT
```

Algorithm 3.3: Key-checking substitution-tolerant algorithm

Security

In this framework, the size of the set of neighbours varies which can lead to vulnerabilities if it isn't addressed, as "1" has fewer neighbours than "g". This is analysed, and a solution is proposed in the complete algorithm.

Design choices

Two approaches could be used to compare the extracted character. The first, shown here, sends the list of neighbours of the typed character. If the typed character is a neighbour of the correct one, then the image of the correct one through the permutation will be among the numbers sent. The other way is to store the list of neighbours during the key-setting and only send the image of the typed character. This has the advantage of slightly lowering the amount of data transferred but makes the system less adaptable to different keyboards. For example, we could consider a user that sets their key on a QWERTY keyboard layout and then uses an AZERTY layout. If instead of typing an "E" in their password they type a 'Z', the password would get rejected, as it is not in the list of neighbours on the initial keyboard. With the version shown in the algorithm, only the present set of neighbours counts[f].

Instead of a permutation, a function that goes from $[0, 255]$ to a greater set could also be used, as it could increase the security by reducing the probability that an adversary could guess the correct number. This is a trade-off between simplicity, efficiency, and security. The main advantage is that it would lower the probability of success of attacks with hashes of different dictionary words. This is not relevant as the advantage of this type of attacks over dictionary attacks is limited in scope by the already low probability of success ($\leq \frac{7}{255}$).

The algorithms call Brassard's algorithm to lazily get the permutation by computing the image of an element only when it is needed (instead of computing all images at the

[f]This does, however, require the system to know which keyboard layout the user is using, which is not always easy in practice.

initialisation, through the Fisher-Yates algorithm [Bla05] for example). In our case, we require 8 pseudorandom bits per element. We need the images of $k = |\text{Neighbours}(P[i])|$ random element chosen uniformly among all possible permutations in a deterministic way dependent on the seed. Fisher-Yates' algorithm would require about 713 random bits if implemented correctly[g], which could be attainable using a longer salt for the seed (hundreds of bits) and a PRNG with variable output. Using Brassard's algorithm [BK88], we require at most 8 bits per call, and at most 80 bits in the calls made by the key-sending algorithm. This allows us to use most PRNG with fixed output length. In all cases, the algorithm used to get the random bits should not be too efficient, as seen in Section 3.3.

The presence of the full hash H_0 is not strictly necessary, but it allows the server to check if everything is right in one comparison. An alternative would be to check $(H_1, L[1])$ and $(H_2, L[2])$, thus detecting the presence of an error — in which case at least one of the hashes would be incorrect — and to check the others lazily only if both tests lead to rejection.

3.2.4 Transposition tolerance

Problem and intuition

On top of single substitution errors, the second framework also corrects transposition errors, where the string "correction" becomes "correctoin".

In the first framework, a single letter was removed before hashing, but this method couldn't detect or correct transpositions. To that end, this framework works similarly but removes two adjacent letters each time and encodes the extracted characters through four different permutations. The first two are used to identify the neighbours of each character and prevent single proximity errors. The next two permutations are used to check whether the two characters are transposed. Instead of sending a hash and a list of neighbours for each character, the key-sending algorithm sends a hash and a set of lists of neighbours for each pair of adjacent characters.

Inner workings

In this framework there are two lists, LA_i and LB_i, which together represent the sets of neighbours of the first and second extracted characters for $0 \leq i \leq n-1$. The checking algorithm accepts if three conditions are true for one index: the hash is correct, one of the two extracted characters is in the set of allowed characters and, finally, the second extracted character is correct (which is true if it has the first index in LA_i or LB_i).

For any single character error, that means that two hashes will be correct (instead of one), and in each case, one character will be correct and the other will be a neighbour. For example, if the string "password" is the correct password, and the user sends "passwotd", HASH(passwd) and HASH(passwo) will be correct hashes. Let's assume that for a moment that the *accepted typos* only cover horizontal proximity errors (the only substitutions accepted are the keys to the right or left of the normal key).

The first hash will be then accompanied by $LA_5 = (f(o), f(i), f(p))$ and $LB_5 = (g(t), g(r), g(y))$, where f and g are permutations. This is compatible with the stored values

[g]The information lower bound is 373 bits, but low-efficiency implementations that require a new random integer at each call would require up to 6400 random bits.

$KA_5 = f(o)$ and $KB_5 = g(r)$. The checking algorithm also accepts if there is a transposition error, in which case a single hash is correct, and both $KC_i = LC_i$ and $KD_i = LD_i$. This is because, during the key-setting phase, $KC_i = f(P[i])$ and $KD_i = g(P[i+1])$, but during the key sending, the indices are exchanged. Thus, the only way to accept is if they are exchanged once again.

Data: Salts $S_0, S_1, ...S_5$, Password P of length n
Keyboard map M: Keys $\rightarrow [0; 255]$
Hash function HASH,
Pseudorandom number generator PRNG
Result: Main hash and list of $n-1$ (hash / integer list) pairs

```
begin
    H_0 ⟵HASH(Concatenate(S_0, P))
    for i from 1 to n − 1 do
        P_i ⟵ P \ {P[i] ⋃ P[i + 1]}
        H_i ⟵HASH(Concatenate((S_1, P_i)
        for j from 1 to 4 do
            Random_bits[j] ⟵ PRNG(Concatenate(S_2, P_i))
            π_{i,j} ⟵Brassard(Random_bits[j])
        KA_i ⟵ [π_{i,1}(M(P[i]))]
        KB_i ⟵ [π_{i,2}(M(P[i + 1]))]
        KC_i ⟵ [π_{i,3}(M(P[i]))]
        KD_i ⟵ [π_{i,4}(M(P[i + 1]))]
    return (H_0, (H_i, KA_i, KB_i, KC_i, KD_i)_{1≤i≤n−1})
```

Algorithm 3.4: Key-setting transposition-tolerant algorithm

Design choices

In this framework, the use of different permutations to encode the two excluded characters is essential, as otherwise the security would be inadvisably lowered. The first two permutations have to be different to prevent an adversary from finding out whether the two characters are neighbours, or have common neighbours. The other two have to be different to prevent adversaries from finding whether a character is present twice in a row.

Optimisation

If we don't allow double proximity errors, LB_i is nearly superfluous, as it's not used to check transpositions, and all single-character typos that are not on the last letter of the password could be corrected using only LA_i. However, computing it has only a marginal cost client-side, sending it has a reasonable cost (at most 19% of the total size), and it would not improve the server-side computation.

Data: Salts S_0, S_1, S_2, Password P of length n
Keyboard map M: Keys $\rightarrow [0; 255]$
Hash function HASH
Pseudorandom number generator PRNG
Result: Set of n (hash / integer list) pairs

```
1  begin
2      H_0 ⟵ HASH(Concatenate((S_0, P))
3      for i from 1 to n − 1 do
4          P_i ⟵ P \ {P[i] ⋃ P[i + 1]}
5          H_i ⟵ HASH(S_1 + P_i)
6          for j from 1 to 4 do
7              Random_bits ⟵ PRNG(Concatenate(S_{j+2}, P_i))
8              π_{i,j} ⟵ Brassard(Random_bits)
9          LA_i ⟵ [π_{i,1}(M(i))]
10         LA_i.append(π_{i,1}(M(SHIFT(P[i]))))
11         foreach a ∈ Neighbours(P[i]) do
12             LA_i.append(π_{i,1}(M(a))
13         LA_i.sort()
14         LB_i ⟵ [π_{i,2}(M(i + 1))]
15         LB_i.append(π_{i,2}(M(SHIFT(P[i + 1]))))
16         foreach a ∈ Neighbours(P[i + 1]) do
17             LB_i.append(π_{i,2}(M(a))
18         LB_i.sort()
19         LC_i ⟵ [π_{i,3}(M(P[i + 1]))]
20         LD_i ⟵ [π_{i,4}(M(P[i]))]
21     return (H_0, (H_i, LA_i, LB_i, LC_i, LD_i)_{1≤i≤n−1})
```

Algorithm 3.5: Key-sending transposition-tolerant algorithm

Data: Length n, H_0 Original list $(H_i, KA_i, KB_i, KC_i, KD_i)$
received hash H'_0 and list $(H'_i, LA_i, LB_i, LC_i, LD_i)$ of (hash / integer lists) pairs
Result: ACCEPT if the password corresponds to the correct one or a version with an acceptable typo.

```
1  begin
2      if H_0 = H'_0 then
3          return ACCEPT
4      else
5          for i from 1 to n − 1 do
6              if H_i = H'_i then
7                  for j from 1 to |LA_i| do
8                      if LA_i[j] = KA_i AND LB_i[1] = KB_i then
9                          return ACCEPT
10                 for j from 1 to |LB_i| do
11                     if LB_i[j] = KB_i AND LA_i[1] = KA_i then
12                         return ACCEPT
13                 if LC_i[1] = KC_i AND LD_i[1] = KD_i then
14                     return ACCEPT
15     return REJECT
```

Algorithm 3.6: Key-checking transposition-tolerant algorithm

3.2.5 Insertion tolerance

Problem and intuition

Insertions are the third most frequent typo, and can be handled by combining the previous two frameworks. The key-setting algorithm sends information corresponding to both previous frameworks, and the key-sending algorithm is the same as the one for the transposition tolerance (Algorithm 3.2). By combining both kinds of hashes, it can detect and evaluate insertions as well as proximity substitutions and transpositions.

Suppose that the password typed has one extra character. We can then check the hashes created by Algorithm 3.2, as two of them will correspond to hashes computed for a single removed character. It is then enough to check that one of two characters left is exactly the one that was not added, which will be true for at least one of the two hashes.

Security

The question of which insertions should be allowed is non-trivial. For example, duplicated letters or added spaces seem like good candidates, whereas letters far from the nearby keys might not be legitimate typos. Additionally, some insertions go with other typos, especially with shift errors. This happens when, instead of hitting the shift key followed by the targeted letter, the user hits a key next to the shift key, committing a double typo. All those can be corrected by this framework, depending on what is chosen to be the list of neighbours. Although this third framework corrects all the typos mentioned (except for the caps lock error), it doesn't address some of the comments mentioned earlier in this section, and reflections in the next. We must then show the complete framework.

Data: Salts $S_0, S_1, ...S_5$, Password P of length n
Keyboard map M: Keys $\rightarrow [0; 255]$
Hash function HASH
Pseudorandom number generator PRNG
Result: Main hash and lists of (hash / integer) and (hash / integer list) pairs

begin
  $H_0 \longleftarrow$HASH(Concatenate(S_0, P))
  for i from 1 to n **do**
    $PA_i \longleftarrow P \setminus P[i]$ `/* Removing the i-th letter from the string */`
    $HA_i \longleftarrow$HASH(Concatenate(S_1, PA_i))
    Random_bits $\longleftarrow$ PRNG(Concatenate(S_2, P_i))
    $\pi_i \longleftarrow$ Brassard(Random_bits)
    $K_i \longleftarrow \pi_i(M(P[i]))$
  for i from 1 to $n-1$ **do**
    $PB_i \longleftarrow P \setminus \{P[i] \bigcup P[i+1]\}$
    $HB_i \longleftarrow$HASH(Concatenate(S_1, PB_i))
    for j from 1 to 4 **do**
      Random_bits[j] $\longleftarrow$ PRNG(Concatenate(S_2, P_i))
      $\pi_{i,j} \longleftarrow$Brassard(Random_bits[j])
    $KA_i \longleftarrow [\pi_{i,1}(M(P[i]))]$
    $KB_i \longleftarrow [\pi_{i,2}(M(P[i+1]))]$
    $KC_i \longleftarrow [\pi_{i,3}(M(P[i]))]$
    $KD_i \longleftarrow [\pi_{i,4}(M(P[i+1]))]$
  return $(H_0, (HA_i, K_i)_{1 \leq i \leq n}, (HB_i, KA_i, KB_i, KC_i, KD_i)_{1 \leq i \leq n-1})$

Algorithm 3.7: Key-setting insertion-tolerant algorithm

```
Data: Length n, Original hash H_0, Original list (HA_i, K_i)
      Original list (HB_i, KA_i, KB_i, KC_i, KD_i)
      Received hash H'_0 and list (H'_i, LA_i, LB_i, LC_i, LD_i)
Result: ACCEPT if and only if the password has at most one acceptable typo.
1  begin
2      if H_0 = H'_0 then
3          return ACCEPT
4      else
5          for i from 1 to n − 1 do
6              if HB_i = H'_i then
7                  for j from 1 to |LA_i| do
8                      if (LA_i[j] = KA_i AND LB_i[1] = KB_i) OR
                          (LB_i[j] = KB_i AND LA_i[1] = KA_i) then
9                          return ACCEPT
10                 if LC_i[1] = KC_i AND LD_i[1] = KD_i then
11                     return ACCEPT
12             else
13                 if HA_i = H'_i AND LB_i[2] = KB_i then
14                     return ACCEPT
15         if HA_n = H'_n AND LB_n[2] = KB_n then
16             return ACCEPT
17     return REJECT
```

Algorithm 3.8: Key-checking insertion-tolerant algorithm

3.2.6 Complete framework

The previous framework can easily be made into a complete framework that is ready for implementation by tweaking a few things. The first is to add an additional hash to account for the inverted case. The second is to make small passwords indistinguishable in the database, padding them all to length[h] 16. The letter lists are also made of equal length by adding dummy character numbers to prevent an adversary from gaining information through the number of neighbours. Finally, we also replace the arbitrary hash function and PRNG by SHA3-256 and use Argon2 as a key stretcher. The parameters on Argon2 require fine-tuning depending on the assumed client hardware at the time of implementation, and the estimated abilities of adversaries, as they create a direct trade-off between usability (in login delay) and resistance to credential theft attacks.

3.2.7 Performance summary

The frameworks shown differ in terms of the proportion of typos handled, computational costs, communication requirements and storage space needed. Those can also vary with the parameters of each framework's implementation, such as the list of allowed typos. Table 3.3 sums up the performance on each front. The difference between the conservative and tolerant methods corresponds to the strategy on accepting random insertions, the first one accepting only duplicated letters or spaces, and the second accepting all insertions. To note, the 15.5% of corrected typos gained by handling caps lock could also be added to any of the other frameworks.

The final algorithm only handles 55.7% of passwords with typos. This is quite normal, as we decided to ignore any typo that includes a deletion, which accounts for at least 22.7% of mistyped passwords. If we consider all the typos we do not want to accept as they do not seem legitimate or pose a security risk — deletions, substitutions which are not with adjacent characters, or more than 2 typos — the forbidden typos represent 36.7% of all mistyped passwords. The complete framework can then handle up to 91.2% of acceptable typos.

Algorithm	Substitution	Transposition	Insertion	Complete
Computation in # of				
Permutations	n	$4n-4$	$4n-4$	$\max(4(n-1),60)$
Hashes	$n+1$	n	n	$\max(n+1,17)$
Numbers	$n \times k$	$(n-1) \times 4k$	$(n-1) \times 4k$	$\max(4(n-1)k,60k)$
Storage in # of				
Hashes	$n+1$	n	$2n$	$\max(2n+1,33)$
Numbers	n	$4n$	$5n$	$\max(5n,80)$
Typos handled				
Conservative	24.2 %	28.4 %	34.5 %	50.2 %
Tolerant	24.2 %	28.4 %	42.2 %	57.7 %

Table 3.3: Performance comparison of the different frameworks. The first part shows the number of permutations, hashes and numbers computed or sent by the client. The second part shows the amount of storage needed on the server for each client, and the third shows the proportion of total typos handled by each, depending on the strategy used.

[h]This "16" is an arbitrary parameter that is a good compromise to prevent revealing small passwords while not costing too much in storage and time.

Data: Username $NAME$, Password P of length n
Keyboard map M: Keys $\rightarrow [0;255]$
Result: Main hash and lists of (hash / integer) and (hash / integer list) pairs

begin
  $S[0] \longleftarrow$SHA3-256($NAME$)
  for i from 1 to 5 **do**
    $S[i] \longleftarrow$SHA3-256($S[i-1]$)
  $H_0 \longleftarrow$Argon2(Concatenate($S[0], P$))
  if $n < 10$ **then**
    return H_0 `/* Preventing general typo correction on very short passwords. */`
  else
    while *Length* $(P) \geq 16$ **do**
      P.append(S[0][0]) `/* Making the passwords have uniform minimum length of 16. */`
    for i from 1 to n **do**
      $PA_i \longleftarrow P \setminus P[i]$
      $HA_i \longleftarrow$Argon2(Concatenate($S[1], PA_i$))
      Random_bits $\longleftarrow$ SHA3-256(Concatenate($S[2], P_i$)
      $\pi_i \longleftarrow$ Brassard(Random_bits)
      $K_i \longleftarrow \pi_i(M(P[i]))$
    for i from 1 to $n-1$ **do**
      $PB_i \longleftarrow P \setminus \{P[i] \bigcup P[i+1]\}$
      $HB_i \longleftarrow$Argon2(Concatenate($S[1], PB_i$))
      for j from 1 to 4 **do**
        Random_bits[j] $\longleftarrow$ SHA3-256(Concatenate($S[j+1], P_i$))
        $\pi_{i,j} \longleftarrow$Brassard(Random_bits[j])
      $KA_i \longleftarrow [\pi_{i,1}(M(P[i]))]$
      $KB_i \longleftarrow [\pi_{i,2}(M(P[i+1]))]$
      $KC_i \longleftarrow [\pi_{i,3}(M(P[i]))]$
      $KD_i \longleftarrow [\pi_{i,4}(M(P[i+1]))]$
    return $(H_0, (HA_i, K_i)_{1 \leq i \leq n}, (HB_i, KA_i, KB_i, KC_i, KD_i)_{1 \leq i \leq n-1})$

Algorithm 3.9: Key-setting complete algorithm

Data: Username $NAME$, Password P of length n, Keyboard map M: Keys $\rightarrow [0; 255]$
Result: Two main hashes and list of (hash / integer list) pairs

1 **begin**
2 $S[0] \longleftarrow$SHA3-256($NAME$)
3 **for** i from 1 to 5 **do**
4 $S[i] \longleftarrow$SHA3-256($S[i-1]$)
5 $P' \longleftarrow$Invert_caps_lock(P)
6 $H_0 \longleftarrow$Argon2(Concatenate($S[0], P$))
7 $H'_0 \longleftarrow$Argon2(Concatenate($S[0], P'$))
8 **if** $n < 10$ **then**
9 **return** (H_0, H'_0) `/* Only sending caps lock typo correction on very short passwords. */`
10 **else**
11 **while** $|P| < 16$ **do**
12 P.append(S[0][0])
13 **for** i from 1 to $n-1$ **do**
14 **while** |Neighbours($P[i]$)| < MAX_NEIGHBOURS **do**
15 Neighbours($P[i]$) $\longleftarrow$ any k with $k > \max_l(M(l))$ `/* Making the neighbours lists have uniform length by adding dummy characters. This also concerns the padding characters from line 12. */`
16 **for** i from 1 to $n-1$ **do**
17 $P_i \longleftarrow P \setminus \{P[i] \bigcup P[i+1]\}$
18 $H_i \longleftarrow$Argon2(Concatenate($S[1], P_i$))
19 **for** j from 1 to 4 **do**
20 Random_bits[j] $\longleftarrow$SHA3-256(Concatenate($H_i + S[j+1]$))
21 $\pi_{i,j} \longleftarrow$Brassard(Random_bits[j])
22 $LA_i \longleftarrow [\pi_{i,1}(M(i)), \pi_{i,1}(M(SHIFT(P[i])))]$
23 **foreach** $j \in$ Neighbours($P[i]$) **do**
24 LA_i.append($\pi_{i,1}(M(j))$
25 LA_i.sort()
26 $LB_i \longleftarrow [\pi_{i,2}(M(i+1)), \pi_{i,2}(M(SHIFT(P[i+1])))]$
27 **foreach** $j \in$ Neighbours($P[i+1]$) **do**
28 LB_i.append($\pi_{i,2}(M(j))$
29 LB_i.sort()
30 $LC_i \longleftarrow [\pi_{i,3}(M(P[i+1]))]$
31 $LD_i \longleftarrow [\pi_{i,4}(M(P[i]))]$
32 **return** $(H_0, H'_0, (H_i, LA_i, LB_i, LC_i, LD_i)_{1 \leq i \leq n-1})$

Algorithm 3.10: Key-sending complete algorithm

```
Data: Length n, Original hash H, Original list (HA_i, K_i)
      Original list (HB_i, KA_i, KB_i, KC_i, KD_i)
      Received hashes H_0 and H'_0 and list (H'_i, LA_i, LB_i, LC_i, LD_i)
Result: ACCEPT if and only if the password has at most one acceptable typo.
1  begin
2      if H = H_0 OR H = H'_0 then
3          return ACCEPT
4      else
5          if n < 10 then
6              WAIT(RANDOM(0.1-1))/* in ms, against timing attacks     */
7              return REJECT
8          else
9              for i from 1 to n − 1 do
10                 if HB_i = H'_i then
11                     for j from 1 to |LA_i| do
12                         if (LA_i[j] = KA_i AND LB_i[1] = KB_i) OR
                               (LB_i[j] = KB_i AND LA_i[1] = KA_i) then
13                             return ACCEPT
14                     if LC_i[1] = KC_i AND LD_i[1] = KD_i then
15                         return ACCEPT
16                 else
17                     if HA_i = H'_i AND LB_i[2] = KB_i then
18                         return ACCEPT
19             if HA_n = H'_n AND LB_n[2] = KB_n then
20                 return ACCEPT
21         WAIT(RANDOM(0.1-1))
22         return REJECT
```

Algorithm 3.11: Key-checking complete algorithm

3.3 Security analysis

The frameworks developed seek to improve authentication systems, which have two goals: preventing people without correct credentials from logging in, and preventing people with — potentially illegitimate — access to the database from getting the credentials of other users. This second point is crucial, as credential stuffing attacks — where an adversary steals a list of login/password pairs on an unsecured website and tests them systematically on other websites — are increasingly frequent, with up to 91% of login attempts coming from credential stuffing, of which on average 0.50% are successful [Pon17].

3.3.1 Preventing access

As the frameworks considered seek to tolerate certain typos, they inevitably cause an increase in the probability of a successful login attempt by an adversary. This is where the choice of which typos are allowed is crucial. For example, allowing single deletions might seem like a good idea: it corresponds to many typos, and only reduces the entropy by a limited amount (around 5 bits on average). However, this would be extremely detrimental in one important case: partial password re-use. As users become aware of credential stuffing, some make small variations to prevent such automated attacks [WRBW16, Pin14]. Accepting deletions makes such attacks much more likely to succeed, which is why a substitution — being very similar to a deletion in terms of security — should only be accepted if the substituted letter is a neighbour of the original. As long as the adversary follows the protocol, the security loss entirely comes from the fact that more passwords are allowed. With a generally lax typo-tolerance system this means that the set of acceptable strings goes from 1 to around 100 for a 12-character password[i]. This makes bruteforce and dictionary attacks somewhat easier, but as countermeasures are shifting the online setting away from those and towards more refined attacks, this should not be a risk for users with passwords of reasonable strength. Typo correction also makes it easier to use safer, longer passwords — which come with a higher risk of typos.

The goal here is to prove that the security loss is exactly that of the added typos. In other words, those frameworks should not reduce the security much beyond accepting the allowed typos. This is done by proving the following lemma in which *smart bruteforce* means that the bruteforce follows the frequent password list by decreasing frequency. The proof is at the end of the following subsection.

Lemma 1. *Using only the username and knowledge of the framework, finding a correct authentication message for a password of length less than* 16 *takes in expectation at least* $\frac{1}{114}$ *times as many queries as a smart bruteforce attack against a system without typo correction.*

Remark 1. Although the bound of $\frac{1}{114}$ sounds bad, there are two reasons that explain and compensate for this. The first is that a query in this system corresponds to a set of queries in a standard system, so the number of queries naturally goes down (but the bound on the number of queries accepted by the server before triggering an alarm should go down accordingly). The second point is that for this bound to be reached, the bruteforce algorithm must be able to distribute queries in an optimal way to make full use of the complex query.

[i] This discounts insertions as the benefit from testing longer passwords is anecdotal.

Remark 2. The lemma here could also be applied to passwords of length strictly greater than 16, but the bounds get more complex and as these passwords are generally not vulnerable to the attacks considered, it was deemed unnecessary.

Intuition

There are two ways an adversary could obtain access if they have no prior information besides the username. The first is to take a set of passwords and send each through the key-sending algorithm, to gain access with either the password itself or a version with an allowed typo. The second is to fake the algorithm's outputs and send at least partially incorrect messages to the server, in an attempt to attack the hash directly.

Let's suppose that an adversary decides to send partially inauthentic login queries. Each query is composed of a main hash, and a set of (hash, number lists) pairs. All the hashes in the complete framework are salted, and the hash space — using for example SHA3-256 — is much greater than the usual password space. This means that sending a random string instead of a real hash can be made to have a lower probability of success (per time unit) than computing a real password hash. For example, assuming a very generous bound of 160-bit passwords (uniformly random password on 20 ASCII characters), it would still take at least 10^{26} login queries before having a reasonable chance of getting a correct hash, evidently costing more than computing one of the correct hashes[j]. Taking a more realistic bound on passwords would only decrease the success probability. As the main limit isn't the number of hashes computed but the number of queries, to maximise their chance of success an adversary would accurately compute all the hashes in the query.

Because sending random hashes is not efficient, an adversary could instead send the same hash in multiple positions, with different additional letters each time. This way, they could cover all possibilities for a single missing letter in only two or three login queries. The checking system couldn't easily prevent this, as common hashes would be possible (for example, the password "*encoded*" has two identical hashes at the end corresponding to removing either "de" or "ed"). Moreover, $n-1$ correct hashes could be computed and then checked in parallel through interweaving.

This effectively increases the efficiency of an adversary by testing multiple passwords per login query. Replacing the permutation of the additional letters by a hash to make their computation harder comes to mind, but gives no real advantage. Indeed, the main deterrent against such attacks nowadays isn't the time needed to compute the hashes and make all those queries, but the maximum number of queries accepted by the service provider (who can also use rate-limiting instead of a maximum number of queries). As the method proposed greatly increases the probability of a user logging in successfully when they make a typo, the maximum number of queries allowed can be reduced accordingly without lowering the usability. Additionally, one could make a counter for a given hash to prevent bruteforcing them: if the server receives a correct partial hash with a wrong additional character, they could temporarily reject all typoed submissions from the user. Essentially, this would be equivalent to typo correction on the first try, and normal password checking on all subsequent tries.

Proof. Any authentication message that doesn't follow the correct structure can be discarded. A message is deemed correct if at least one of the hashes is correct, and the corre-

[j]This assumes that the adversary knows the salt, which is reasonable as it could, for example, be computed from the login.

sponding numbers are also correct. A message must either contain a correct hash/number pair, or a correct number and a hash collision. As the hash space is much greater than the space of 16-character passwords, using random hashes to find collisions has a probability of success so low ($< 2^{-128}$) that it is irrelevant. As the checking algorithm prevents timing attacks, just finding the hash by itself is not reasonable. The adversary must then have at least one (hash, number list) pair correct. Every query they make has 18 possibilities of getting an acceptance: one for the first two hashes, and one for each of the 16 (hash, number list) pairs. Each query has 7 chances, hence an uper bound of at most 114 acceptance chances. □

3.3.2 Obtaining credentials from the database

The second attack can be performed by an adversary with access to the database and focuses on obtaining correct pairs of password and email/login credentials for use against other targets. The goal is then to prove the following lemma:

Lemma 2. *Let's consider an adversary with access to the usernames, corresponding (password hash, number) lists and transcripts of successful login interactions. Using generic attacks, they require at least $\frac{1}{16}$ as much computing power to get a password of length < 16 from a single user as if the database only stored simple hashes of the passwords without typo-correction.*

Remark 3. Once again, the bound of $\frac{1}{16}$ given is a worst case analysis, and real data shows that the real speedup is close to 1.5.

Securing structural information

The first step to prevent credential theft is to make sure that the database itself doesn't give structural information on the passwords through the way it stores them. For example, storing hash lists of varying lengths would reveal the length of the stored passwords, indicating to adversaries the ones that would be easiest to crack.

For the users with passwords of length < 10, exactly two hashes are stored, and the adversary gains at most a factor 2 in the bruteforcing (less in practice due to non-uniform distribution). Let's now consider users with passwords of lengths ≥ 10.

Deterministically adding extra characters to the end of the password to reach a common length prevents this kind of attack. However, we should avoid compromising users with already long passwords by imposing length upper limits. Adding characters only if the passwords are of length less than 16 seems a good compromise, with only a few passwords standing out from the database as being extra-hard. Despite the uniformity of the database, a successful attack could still happen if an adversary also has access to the messages received by the database. In messages received, the length of the allowed key list — the list of numbers — is also important as it can give a lot of information on the position of the keys on the keyboard. To avoid this, having a few numbers on the client-side reserved for non-existent keys and filling up the neighbour list with those prevents this information leak.

Cracking the hashes

We are left with the problem of computing passwords from a set of list of hashes and numbers, with each list having a single salt. The adversary has three avenues of attack.

The first is by bruteforce: enumerate all the possible passwords and check when they are correct by comparing with the recorded hashes. This is where key stretching[k] is central but must be used wisely, to make the computation of each hash expensive and prevent the adversary from bruteforcing billions of passwords per second [Spr11]. The second attack uses hashes directly and computes their preimages. The third method uses the recorded numbers to get information on specific letters of the password and simplify the rest of the work. We will first show that these two methods are less efficient.

With the second method, considering each list independently, finding the preimage of a single hash in it is enough for the attacker, as the number of possibilities left for the missing letter becomes trivial. We are then looking at multi-target preimage attacks with a promise on the structure of the targets (that their preimages are close together[l]). As stated in [BDPvA08], however, the resistance of even SHA3-256 against generic attacks is much stronger than the security requirements for passwords. This means that the main avenue of attack doesn't go by finding the preimage of the password hashes.

When it comes to the third avenue of attack, collisions are frequent, as opposed to hashes, as the image space of each permutation is small. If computing the permutations were more efficient than computing the hashes, it would be possible for the adversary to eliminate lots of potential passwords quickly. Two methods can be used to prevent this. The first is running the key stretching method on each random bit computation, which either increases the delay for the user or decreases overall security by reducing the time spent on each key stretching, which is done in the complete framework. The second method uses the same key stretcher for both the PRNG and the hashing. This works by first using the key stretcher on PB_i, hashing the output with different salts to get the random permutations and finally the hash itself. This could slightly affect preimage resistance but makes bruteforce attacks to find the permuted characters at most as efficient as the bruteforce attacks against the hash itself. This is because, if an adversary wants to eliminate possibilities for the k-th character, they must compute the permuted character for each password, and then eliminate all the impossible ones. If they don't run the procedure for the correct password they can't reliably eliminate passwords or characters, and if they do they automatically get the correct hashes (and the answer) at no additional cost.

Bruteforcing the passwords

The only way is then to use bruteforce from the password side, testing every password until the adversary finds one with the correct hash. The traditional way to prevent this is to use key stretching methods such as PBKDF2 [Kal00] — or rather Argon2 [BDK16], which also has security guarantees against generic attacks. This is where the frameworks shown have a security flaw, as we have at least 16 different hashes instead of one to create and send the password, but the adversary only has to find one. Making all of them go through key stretching methods either takes more time or lowers the number of iterations on each of them[m]. Two factors mitigate this flaw: first, even running a key stretching method for a

[k] Essentially, key stretching is the process of running the hash function on itself a fixed number of times to make computation slower — and bruteforce more expensive.

[l] It would be interesting to check whether that kind of promise problem makes preimage computation any easier, but in any case, they could also be made irrelevant by the use of different salts for each of the $(n-1)$ password hashes.

[m] Using a key stretcher on the central salts that are used afterwards by the rest of the algorithm centralises this proof of work but does not provide any extra security.

few milliseconds is enough to make bruteforce attacks very costly. Assuming we use Argon2 — which prevents efficient large parallelisation — for 2ms on each hash, cracking a 48-bit password would naively take an average of sixteen billion seconds, or 544 years, on the same machine. This does not use the fact that it is enough to guess one of the hashes containing a typo. Assuming a 5-bit loss of entropy — which requires a well-optimised bruteforcing algorithm — the expected time is still more than 17 years. We simulated the use of this method on the Rockyou leaked password data-set [Van10, KKM+12], bruteforcing until we get hashes for the 50% most frequent passwords of length > 10 from the list. The speedup varied a lot depending on which two characters were removed, as shown in Table 3.4, where the proportion of the original number of passwords needed to get 50% of the total frequency is shown on the third line.

Characters removed	none	0-1	1-2	2-3	3-4	4-5	5-6	6-7	7-8	8-9
Unique passwords ($\times 10^6$)	4.40	4.26	4.33	4.29	4.29	4.28	4.26	4.22	4.12	3.96
Proportion for 50%	33.1	29.9	31.4	30.6	30.7	30.4	30.0	29.0	26.7	23.0
Speedup	1	1.11	1.05	1.08	1.08	1.09	1.10	1.14	1.24	1.44

Table 3.4: Speedup gained for dictionary attacks by removing 2 characters from Rockyou passwords of length >10. The first line has the number of unique passwords (in millions), and the second indicates the proportion of passwords needed to get the 50% most frequent passwords if we remove the characters in the i-th position.

As we can see, even among a list notorious for containing many bad passwords with lots of redundancy, removing two characters only reduced the average number of hashes to compute by about 31% when setting the character position in advance — and dynamically removing the best 2 characters would improve this by at most a few percentage points. This could be further reduced by using more efficient hardware but is still prohibitive. The second mitigating factor is that a smart user interface would compute the key stretching before the user is done submitting the password, recomputing from scratch each time a new character is typed. This can guarantee at least an additional 10ms of key stretching per hash without the user noticing.

With these arguments, we can now prove Lemma 2. We only consider users with passwords at least 10-characters long, as otherwise the proof is immediate due to the trivial typo correction.

Proof. The hashes are all computed with different salts, so rainbow tables can't help, and cracking a single user's credentials doesn't help the attacker with the credentials of another user.

The data under each user is composed of the same number of hashes and corresponding numbers, except for the users with increased security, and the transcripts are also structurally identical, so finding the users with passwords of lower lengths is as easy as finding out that the last characters of those passwords are made of padding. Knowing that they are made of padding requires knowing that they are the image of a non-existent character, which is equivalent to finding that they are the image of a given character.

However, finding whether the number stored corresponds to a given character through bruteforce is not easier than finding the password itself, as a large set of passwords with different characters in its stead will yield the correct number. If the actual password wasn't in the set tested, the adversary can't guess the extracted character with probability much

bigger than uniform, whereas if the correct password was in the set the adversary already knows a correct hash.

As the preimages under the hashing functions considered are much harder to compute than bruteforcing from the password side, and as the additional numbers give no information unless the adversary knows the rest of the password, the only viable generic attack goes through bruteforcing from the password side.

An adversary can then consider one position, ignore the two letters concerned, and bruteforce all the others. In a best case scenario, this method could remove close to 14 bits of entropy, or improve by a factor 15000 the speed of the bruteforce. However, using NIST estimates [BDP04], at best 4 bits of entropy would be lost, corresponding to a factor 16 speedup — much higher than the 45% speedup observed on the data.

□

3.4 Alternative algorithm based on the discrete logarithm and lower bounds

Although the algorithms shown previously should be sufficient for practical applications, it is worth wondering whether better solutions could exist, at least from a theoretical standpoint. Taking inspiration from homomorphic encryption, can we safely compute some forms of edit distances between two strings on encrypted data?

3.4.1 Impossibility results and lower bounds

The following trivial lemma poses a first obstacle to the existence of such frameworks:

Lemma 3 (Folklore)**.** *Let A be an n-character string on an alphabet of size m. Knowing n and having access to an oracle that can compute the Hamming distance between A and B for any n-character string B, one can find A in $O(m \times n)$ queries to the oracle.*

Proof. By taking any B as an initial string and iteratively changing one character repeatedly until the oracle indicates a lower distance before moving to the next character, we can converge to the correct A in at most $m \times n$ queries. □

Remark 4. The bound of $O(m \times n)$ can be reduced to $O((m+n)\log m)$ by cutting the string into bits of size $< m$, finding which letters are present in each substring and getting their positions by dichotomy.

Lemma 3 is but a first obstacle, as this reasoning applies to other metrics, shown in the following lemma and its corollaries. Let $\mathcal{F}$ be a set of operations that take a single string, a set of indices in that string, and a set of characters in an alphabet of size m and return a string. Let $\mathcal{F}$ be such that for any operation that can transform A into B, there is a symmetric[n] operation transforming B into A. Let the $\mathcal{F}$-edit distance d between two strings be defined as the minimal number of operations from $\mathcal{F}$ required to transform one into the other.

Lemma 4. *Let A be an n-character string on an alphabet of size m. Knowing n and having access to an oracle that can compute the $\mathcal{F}$-edit distance between A and B for any*

[n] Through the definition, we have the properties of non-negativity, identity of indiscernibles and subadditivity, thus we only need to add symmetry to get a metric.

n-character string B, one can find A in $O(D \times (\max(m,n)^k))$ queries to the oracle, where k is the maximum arity of operations in $\mathcal{F}$, and D is the maximum $\mathcal{F}$-edit distance between two strings of length n.

Proof. As before, we start with an arbitrary string B and query its distance to the target. We then run each possible operation in $\mathcal{F}$ on all possible operands, querying the oracle with each result. As there are at most $(\max(m,n)^k)$ operand combinations, and at least one of them reduces $d(A,B)$, we need at most $O(D \times (\max(m,n)^k))$ queries to find A. □

Corollary 1. *Finding A without knowing n takes at most $O(m + n \log n)$ queries with an oracle giving the Levenshtein distance, $O(n(\max(m,n)))$ with the Damerau-Levenshtein distance [Dam64], and $O(n^4)$ for Kendall's tau metric [Ken38].*

Proof. The second and third assertions are obtained by direct application of the previous lemma, the first is a bit more complex.

The first step is to realise that we can find n in $O(n)$ queries (for any of the considered distance and including when only queries of at least a minimal size are accepted). Starting with the smallest query size, we increase the query at each step while keeping all the letters equal to any constant letter. The distance should decrease or stay constant until it finally goes up — corresponding to a deletion — in which case n is now known to be equal to the current length minus one.

Once we have n, we can find how many occurrences of each letter we have by querying words composed of a single repeated letter, which takes $O(m)$ queries. In $O(n)$ queries we can get the exact location of all occurrences of one given letter (by putting it successively in each position). Using this letter as a background, we can do a dichotomic search for the positions of each remaining letter, which takes $O(\log n)$ queries per letter, or $O(n \log n)$ total.

□

This lemma also holds when the answer given by the oracle is imprecise. For example, instead of computing the exact distance, it could indicate $f(d(A,B))$, for a function f like $f(x) = \lceil \frac{x}{4} \rceil$.

Corollary 2. *Let f be a non-decreasing function, and let*

$$D = \max_{B,|B|\leq|A|} d(A,B) \quad \textit{and} \quad p = \max_{i,j<D,f(i)=f(j)} (i-j)$$

If the oracle answers queries by returning $f(d(A,B))$, finding A takes at most $O(D \times (\max(m,n)^k)^p)$ queries.

Proof. Following the proof of Lemma 4, we can iteratively try all sequences of at most i $\mathcal{F}$-operations, for $1 \leq i \leq p$. This requires at most $(\max(m,n)^k)^p)$ each time, and has to be done at most D times before converging to A. □

Corollary 3. *For a given A and B, let the oracle have a probabilistic outcome which follows a distribution with values in $[0,T]$ and with expectation $d(A,B)$. There is a probabilistic algorithm that computes A with probability $\Omega(\frac{1}{2})$ in $O(D \times (\max(m,n)^k) \times k \times \ln(D \times \max(m,n)) \times 2T^2)$ queries.*

Proof. The proof follows that of Lemma 4, except that instead of querying for a single value from the oracle, we make $2kT^2 \times ln(4D \times \max(m,n))$ queries and take the integer closest to the mean. By Hoeffding's inequality [Hoe63], we have a probability at most $\frac{1}{2D\times(\max(m,n)^k)}$ of getting a wrong value for the distance. Using Boole's inequality — or union bound [Hun76] — we can bound by $\frac{1}{2}$ the probability of getting at least one distance wrong over the execution of the algorithm. Thus, with probability $\Omega(\frac{1}{2})$, all the distance estimates are accurate and the correct string is computed. □

The proofs of those corollaries can also be combined to prove that the lemma holds even against an oracle that answers a probabilistic approximation of a function of the distance. We can then see that any simple system that allows computation of an edit distance between a secret string and arbitrary queried strings can be used to find the original secret string in a polynomial number of queries. Any method that seeks to prevent the discovery of the secret string must then be able to overcome this, for example — as is done hereafter — by having variable costs for queries. We need a final elementary lemma to establish the optimality of the following algorithms.

Proposition 1. *Let f be a function from $\{0,1\}^*$ to $\{0,1\}^n$, such that finding x from $f(x)$ takes in expectation $\Omega(2^{\alpha\times n})$ operations, with $\alpha > 0$. Let there be a three-party system where the first party has a secret A, and the second party must check whether the third knows either A or A' with $d(A,A') \leq E$ for a given distance d and a constant E. Let's also assume that the second party receives a single message from the first, followed by a single message from the second before deciding. Then for any deterministic algorithm that guarantees that a good message gets accepted with probability 1, that a random message has probability $O(n \times 2^{-n})$ of getting accepted, and that finding A takes $\Omega(2^{\alpha\times n})$ operations, the second party must store at least $n - g(n)$ bits, and the third must send at least $n - g(n)$ bits, where $g(n) \in o(n)$.*

Remark 5. This proposition basically states that, in order to achieve the security corresponding to the comparison of two password hashes, it is necessary to store one hash on the server, and send the other.

Proof. Let us suppose that there exists $g(n) \geq C \times n$ for a constant C such that a protocol satisfying the assumptions exists, which requires sending only $n - g(n)$ bits in the first step. Then the server can only have at most $2^{n-g(n)} \in o(2^n)$ internal states. Assuming the adversary knows the protocol, as there is no other secret data, they can craft one message corresponding to each of these internal states. As there is at least one message that leads to acceptance, sending a random message from the crafted set leads to acceptance with probability $\omega(n \times 2^{-n})$. The proof follows the same idea when the message in the second step takes only $n - g(n)$ bits.

□

3.4.2 Discrete logarithm method

Before the algorithm, we must first introduce a distance between strings which, although simple, is not generally used. Let's consider a keyboard, with a standard QWERTY layout, as in Figure 3.2. The 48 main keys of the keyboard and the different characters they can create can easily be modelled by a 3-dimensional coordinate system. The first

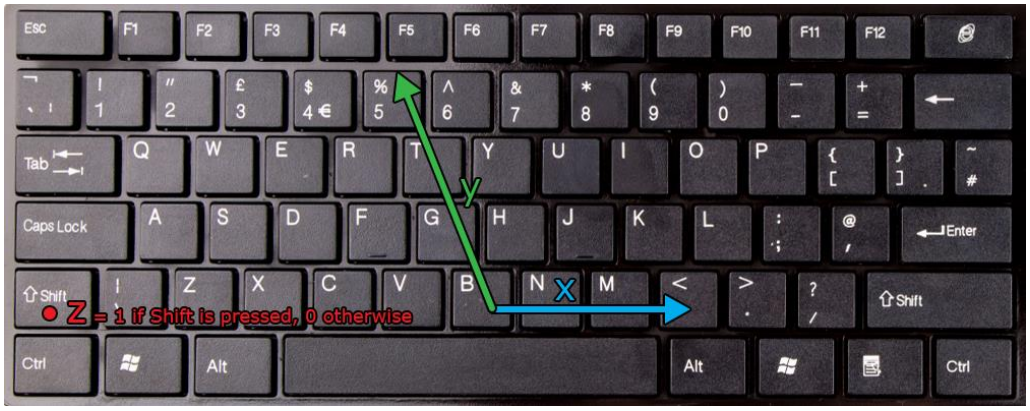

Figure 3.2: Keyboard coordinate system, starting at the bottom left. The string "Arc" has coordinates $((1,1,1),(4,2,0),(3,0,0))$.

dimension corresponds to the horizontal position of the key (or the row), the second dimension to the vertical (the diagonal column), and the third dimension to the *modifiers*, here only considering Shift although it could easily be extended. This forms a subset of a $14 \times 4 \times 2$ lattice[o] as shown in Figure 3.2.

Definition 1. Let s be a string of length n. The string coordinates of s are defined as the sequences $(x_i)_{1\leq i\leq n}$,$(y_i)_{1\leq i\leq n}$ and $(z_i)_{1\leq i\leq n}$, where (x_i, y_i, z_i) are the coordinates of the i-th letter in the previous coordinate system.

Definition 2. Let s and s' be strings of identical length n. Let the keyboard distance between s and s' be defined as the L^1-distance between their string coordinates, that is,

$$d(s,s') = \sum_{1\leq i\leq n} \left(\left|x_i - x'_i\right| + \left|z_i - z'_i\right| + \left|z_i - z'_i\right|\right)$$

By this definition, the distance between *homomorphic* and *homimorphic* is 1, but the distance between *homomorphic* and *Bomomorphic* is 3, the same as the distance between *homomorphic* and *homomor;jkc*. The expected distance between two random n-character strings is then $\frac{59707}{10296} \times n$, or about 58 for 10-character keymashes.

Definition 3. Let s be a string of length n, and let $(x_i)_{1\leq i\leq n}$,$(y_i)_{1\leq i\leq n}$ and $(z_i)_{1\leq i\leq n}$ be its string coordinates. Let p_i be the i-th prime number. We define the integral representation $X(s)$ of s as

$$X(s) = \prod_{1\leq i\leq n} p_i^{x_i} \times p_{i+n}^{y_i} \times p_{i+2n}^{z_i}$$

To follow the example in the figure, the integral representation of "Arc" $2\times3^4\times5^3\times7\times11^2\times17 = 291579750$. The integral representation of "ArC" is $2\times3^4\times5^3\times7\times11^2\times17\times23 = 291579750 \times 23$.

Remark 6. Alternatively, we could have used a more intuitive definition, with $X(s) = \prod_{1\leq i\leq n} p_{3i-2}^{x_i} \times p_{3i-1}^{y_i} \times p_{3i}^{z_i}$. This means that strings that include others as prefixes have integral representations that are multiples of the prefixes' integral representations. As we

[o]One could also add the space key, in which case the following proofs still work although with a slightly different structure. Similarly, adding the Alt key would only make it a 4-dimensional coordinate system.

only consider strings of constant length, this leads to higher values of $X(s)$ with no real advantage. On a standard keyboard, for a string s of length 10, $X(s) < 2^{966}$ with the second definition whereas $X(s) < 2^{768}$ with the first (and $X(s) < 2^{853}$ for length 12). In all cases, they are in expectation quite above 2^{250}, which is enough to prevent discrete logarithm attacks on small exponents [GM16].

With the coordinate system, the associated distance and an integral representation, we can now define the key-setting and the typo-checking algorithms, inspired by the Diffie-Helman key exchange. Intuitively, we take a random element in a group and put it to the X-th power, where X is dependent on the password. Because of the function's structure, it is easy to compare the elements corresponding to two closely related strings. The security lies in the assumed hardness of computing the discrete logarithm.

Here, the key-setting and key-sending algorithm are identical: the stored key is exactly the one that is sent the first time the client creates the password.

Remark 7. The PRNG in the algorithm could, be a derivative of Keccak but does not require a high level of security, and any algorithm to get an element from a set of pseudo-random bits — such as a PCG one [O'N14] — would be appropriate.

Data: Username string U, Salt string S, Password string P
Group G, Pseudorandom number generator f
Result: An element $g_0 \in G$ corresponding to the "hashed" password, that is sent to the server

1 **begin**
2 Compute the string coordinates $(x_i, y_i, z_i)_{1 \leq i \leq |P|}$ of P
3 $X \longleftarrow \prod_{1 \leq i \leq n} p_i^{x_i} \times p_{i+n}^{y_i} \times p_{i+2n}^{z_i}$
4 $Y \longleftarrow U + S$
5 $N \longleftarrow f(Y)$
6 Let g be a pseudorandom element g of G computed from N
7 $g_0 \longleftarrow g^X$
8 Transfer g_0 to the server

Algorithm 3.12: Key-setting/sending discrete logarithm algorithm

Remark 8. The reason why we compute two lists of elements is that computing errors where a_i is greater than expected is easy, as $g^{Xp_i} = (g^X)^{p_i}$. Computing errors the other way around is actually akin to computing a discrete logarithm in the group. As such, the distance computation in this algorithm always goes from the "smaller" to the "bigger" password, which can thankfully be mixed when the keyboard distance is greater than 1.

The comparison on lines 14-16 is not optimal, but it is simple and should not affect the running time as the main factor is the computation of the powers of the elements in G.

3.4.3 Security and performance

The security of this algorithm directly comes from the discrete logarithm assumption: computing P from g_0 corresponds exactly to solving the discrete logarithm with the promise that the solution is a $3n$-smooth number — for potentially high n in case of added padding. To implement it in practice, one would have to be careful to choose an appropriate group [ABD+15]. A cyclic group of order p with p a 2048-bit prime should be enough

Data: Group G, constant D, maximum length n
Stored element $g_0 \in G$, received element $g_1 \in G$
Result: Keyboard distance between the passwords if it's less than D.

begin
  for i from 1 to D **do**
    for j from 0 to i **do**
      $L_0 \longleftarrow [\,]$
      $L_1 \longleftarrow [\,]$
      foreach $1 \leq a_1 \leq a_2 \leq ... \leq a_j \leq 3n$ **do**
        $X_0 \longleftarrow \prod_{a_k} p_{a_k}$
        $g' \longleftarrow g_0^{X_0}$
        $L_0 \longleftarrow \text{Concatenate}(L_0, g')$
      foreach $1 \leq b_1 \leq b_2 \leq ... \leq b_{i-j} \leq 3n$ **do**
        $X_1 \longleftarrow \prod_{b_k} p_{b_k}$
        $g' \longleftarrow g_1^{X_1}$
        $L_1 \longleftarrow \text{Concatenate}(L_1, g')$
      foreach $g' \in L_0$ **do**
        if $g' \in L_1$ **then**
          return i
  return REJECT

Algorithm 3.13: Distance-checking discrete logarithm algorithm

for current computational capabilities, or a similar algorithm could be adapted for elliptic curves.

With this framework, the login queries are all of the same format — a single element of the group. In a way, as shown in Proposition 1, this could lead to a proof of optimality in terms of space and communication bits required, depending on the group used in practice. It also means that faking an id is is not easier than the hardest typo-tolerant framework that accepts the same typos. As the size of the group is much greater than the general password space, the discrete logarithm assumption also implies that bruteforcing the password is once again the best avenue of attack. As previously, this can be slightly optimised by not trying passwords that are close to each other.

Besides the fact that it only allows the correction of substitution errors, the main downside of this algorithm is the time needed to compute the distance. This is still acceptable on the client side, where the main hurdle is squaring an element at most 1600 times in a large group. Using efficient libraries, this can be done in less than 10ms. However, the server-side computation is where the cost becomes prohibitive. For strings of length 12, checking whether they are at distance 1 takes at most 72 exponentiation operations, or less than 500 squaring operations, doable in a few ms. At distance 2, computation already takes 35 times more operations, which is on the edge of noticeable from the client-side. Checking whether they are at distance 3 (probably the highest reasonable distance for typos) is, alas, prohibitive, taking at least a few seconds. Using the trinomial revision, the

number of expected exponentiations at distance $D \leq n$ is on average

$$\frac{1}{2}\sum_{i=0}^{D}\left(\binom{3n}{i}\binom{3n-i}{D-i}\right) = 2^{D-1} \times \sum_{i=0}^{D}\binom{3n}{D} \geq \frac{1}{2}\left(\frac{6n}{D}\right)^{D}$$

This illustrates why the method is not concerned by the lower bounds shown previously: although a linear number of queries might be enough to find the original string from the computed distances, most of those couldn't be computed because of the exponential cost.

Remark 9. There is, in fact, one potential risk that requires investigating with this method. The discrete logarithm assumption concerns normal elements of the group. However, the elements considered here are not random elements but X-th powers, with B-smooth X, for $101 \leq B \leq 181$. Although B-smooth numbers are essential in discrete logarithm problems [Pom94], this does not seem to be a situation where X being B-smooth is an issue.

3.5 Conclusion

The main contribution of this chapter is a set of frameworks that can be combined in a complete system with the following properties:

- It corrects 57.7% of all typos, or 91.2% of legitimate typos.
- It stores 32 hashes and 90 integers on the server. It is compatible with lazy evaluation — only checking the remaining hashes when the main one is incorrect — and does not require any extra computation on the server's side.
- It requires no additional waiting time for computation on the user side, as it can run between the moment the user presses the last key and the moment they submit the password.
- It creates little extra communication cost as the additional data can still fit in an average size packet (420 bytes for the numbers, 544 bytes for the hashes), well below the IPv6 MTU [DH14].
- Assuming optimised code that runs on specialised hardware 15× faster than an average client's browser's hashing ability, bruteforcing a single password from the database still takes more than a year[p].
- Faking a correct authentication message is at best 114 times more efficient than normal bruteforce, but this can be compensated or eliminated by having stricter constraints on the number and frequency of queries while still having a positive impact on usability.

When compared to TypTop, the best typo-correction system today[q], it has greater usability — correcting about twice as many typos — and lowered computing costs, at

[p] This assumes that the client interface runs fast hashing algorithms, for example, in a WebAssembly or PNaCl environment, which can have a 20× speedup over asm.js [Ant18, DMCS10, Ros16].

[q] This title of best is easily attributed as the only competitors — to our knowledge — are previous systems by the same authors.

the cost of an increased storage and lowered security guarantees. Multiple practical improvements could still be added to the system considered. For example, as the system can detect typos, it might be interesting to let the user know when they've made one (although this might lower usability). Looking in another direction, it would be possible to associate given (hash / number) pairs with frequencies and allow typos probabilistically, with the system being more forgiving when the typo is repeated. Combining both approaches, if a typo happens with great frequency, it would be possible for the system to ask if the user wants to make that their new password. It would also be possible to use some secret sharing system to combine the different hashes and simplify the computations, but this seems to require a challenge system with at least two rounds of communication.

Naturally, as with the results shown in the previous chapter, the schemes proposed depend on the service providers' will to implement them. Thankfully, we can easily address this. Switching from a system where passwords are simply hashed requires two things to be changed: the database must be transformed, and the client's code must also be made to compute the new kinds of hashes. The first part is relatively simple and can be done by adding an extra column that points to the new complete hashing information and is accessed only when the main hash is not correct. Each time a user correctly logs in, the database uses the occasion to add the relevant data (which is sure to be correct as the main hash matches). This allows the service provider to maintain compatibility with a legacy system and lazily upgrade the security of all users.

The client's code must also be transformed so that it transfers not just the main hash but all the necessary information. This can be done without requiring redeployment or updating clients when considering web services. Indeed, the service provider is also the one providing the Javascript code for the web page, and can update this centrally without directly implicating the users.

An important change between this system and what is commonly used is that hashes are computed on the client's side, but there are nowadays next to no reason to compute them on the server's side — unlike two decades ago when they could be necessary to assure compatibility with legacy systems. As such, it is then time to turn to solutions that can be implemented by the users themselves, without needing any input from service providers.

4

Cue-Pin-Select, a Secure Mental Password Manager

4.1 Previous work and contributions

As mentioned in Chapter 1, the number of passwords is ever-increasing, and having different passwords for each service generally requires password managers or memorable (but weak) passwords, introducing other vulnerabilities [DBC+14, Lip16]. This is often the result of an unconscious trade-off between security and usability, sometimes leading to cognitive dissonance [Las16]: although users know they are vulnerable, they do not take actions to remedy this. A wide variety of factors affect this choice, among which mainly stand effort, lack of information about alternatives and lack of perceived usefulness [AR16]. Inadequate mental models of security also play a role [FHvO14b, SL12, MMD14, KB84].

This isn't only the users' fault: well-meaning but counter-productive constraints (such as mixed-case, numbers and symbols) have been mostly detrimental [SKD+14, KSK+11]. They not only pushed users to have weak passwords — focusing their efforts on satisfying or bypassing the constraints instead of making good passwords — but also forced them to create passwords in an ad-hoc way, preventing them from following habits which improve memorisation. Those new passwords become not only weak but forgettable, and lead to frequent resets of the rarely used passwords. The traditional solutions for users have been to write down their passwords [GF06], use the same few passwords for everything [Cen14], or use password managers, constituting a single point of failure from which an adversary can completely steal an identity [LHAS14].

A different potential solution is to create a set of non-independent passwords, related by a common pattern. As humans are natural pattern seekers, many intuitive ways have been devised to avoid password re-use without incurring too high a mental cost. Those schemes which create new passwords automatically can be arbitrarily simple or complex, going from very small variations at the end of a word to word association schemes [Lee14, BV15]. They alas tend to have security barely above password re-use, as users may produce families of related passwords which an adversary can easily infer if they learn some examples (such as Mypassword1! Mypassword2...). Finding a good method to remember large sets of passwords at little cost to the user would then be a boon.

This chapter is based on research done with Ted Selker and Eli Sennesh.

Previous work. There have been a few recent attempts at secure methods, but they generally require a large amount of rote memorisation [BV15, HB01] or computation on a physical device[a] [SCL12, BBDV17]. Efforts have mostly been led by Manuel Blum and his co-authors N. Hopper, J. Blocki, A. Datta and S. Vempala, with six papers on the subject, including five in the past five years [BBD13, BBDV17, BV17, BV15, SVK18]. In them, they have framed, formalised and brought forward many important issues in mental passwords managers and password schemes, on both the security and usability fronts. They mention five criteria that schemes should satisfy:

- *Analysable*, meaning that the schema should be a well-defined and deterministic algorithm.
- *Publishable*, corresponding to Kerckhoffs's law.
- *Secure*, typically resisting both online and offline attacks from an adversary with superior computing power.
- *Self-rehearsing*, such that the process of using the scheme regularly enough is sufficient to remember it.
- *Humanly usable*, such that an average human can learn and use the system at no great cost.

They have also proposed a series of algorithms, generally based on challenge systems, where the authenticating system sends some information to the user, to which they must respond accordingly. The challenges are combined with computational primitives that the users are supposed to execute. For example, their DS1 protocol requires a user to memorise a random letter-to-digit map as the seed to their password creation, while WS1 asks users to memorise both letters and words about a topic [BV15]. They also describe LP1 that requires users to memorise a random letter permutation. These different kinds of memory tasks are all valid, but memorising random mappings is difficult, and may take time and a large amount of effort.

One of the most interesting systems they created, for which they prove strong security bounds, requires the user to remember a mapping between digits and a set of 14 images [BBDV17]. Once they have the mapping, the challenge system shows the images on the screen in a random order, and the user has to compute

$$x_{13} + x_{12} + x_{(x_{10}+x_{11} \mod 10)} \mod 10$$

where x_i is the number corresponding to the i-th image in the list shown to the user. Although this is complex enough, authenticating using this kind of system requires at least 3-5 different challenges. The original paper mentions that the main author managed — with some training and solid natural mathematical abilities — to reliably compute the function for a single challenge in 7.5 seconds. Most users would then probably take at least 40 seconds for the complete authentication procedure.

In spite of their excellent work and prescient points about what would be usable password creation and retrieval, people today still don't know which method to use. We are still left with no good example algorithm for easily making secure families of passwords,

[a] This can be problematic as it lowers usability, can create problems on untrusted machines — which is sometimes addressed in the papers cited — but can mostly make denial attacks much easier.

and the main usable methods are still vulnerable to typical attacks. Moreover, it seems that the criteria they created might not be enough.

Contributions. This chapter introduces three new criteria and proposes and characterises a specific usable scheme (though certainly not the only one possible) that allows the easy creation, memory, usage, and retrieval of a password, while remaining secure against a large variety of attacks. The criteria are as follows:

- *Agent-independence*, meaning that the user should not require any outside help, be it from their own computing devices, an untrusted calculator or phone, or even pen and paper.
- *Scalability*, an extension of their *publishability* criterion, with two constraints: the security of the scheme should not strongly diminish with either increased popularity or increased use by a single user with many different passwords.
- *Adaptability*, the final and most important criterion, permitting the user to always be able to use the system no matter the idiosyncratic constraints they face.

This last criterion is crucial as otherwise the user is faced with two possibilities: creating a new password in an ad-hoc way, which might be insecure and which they will probably forget, or change the algorithm used at some point. Moreover, remembering many algorithms, their associated secret information and where they were used seems even harder than remembering many passwords.

The system we propose, called *Cue-Pin-Select*, is a password generation scheme for 12+ characters that is provably strong and adaptable to the requirements of today's systems. Designed to profit from our natural linguistic abilities, it performs well on constraints of usable memory and learning, while fulfilling strong security constraints. The system relies on selecting a cue from or for a service onto which the user might log (such as part of its name), and applying a PIN to create an index into a common six word phrase. The intent behind its creation was to show a system that cannot be reasonably hacked even if the adversary knows some of the plain-texts, while also making it human computable without command-line tools or too much work. We can learn six words and a PIN much more easily than the random mappings proposed and they are directly usable without any specific knowledge or ability. A preliminary user study showed how all alpha-testers with only a few minutes of training were able to produce a new secure retrievable password in well under a minute and improved their times consistently in 19 trials over 4 days.

In general considerations on password schemes, we will suppose that each scheme is composed of three kinds of information:

- Some user-only information which can include the initial seed for random data, or any piece of information that should absolutely not be spread. If an adversary obtains it, this information can potentially allow them to recreate all the user's passwords.
- The passwords themselves, from which it should be difficult to find the user-only information.
- An environmental cue, a piece of information (in our case four characters), or the scheme used, from which a user can (re)create a password if they have the user-only

information. Those can be guessable and should offer little direct information on the rest by themselves.

The rest of the chapter follows the following structure. We start by explaining the proposed scheme and analyse its resistance to the main types of attacks. We then look at it from a usability standpoint and show the promising results of a small-scale user study that put participants in real-life conditions for multiple days. Finally, we introduce multiple variants and explain the design choices leading to Cue-Pin-Select

4.2 The Cue-Pin-Select Scheme

Cue-Pin-Select uses four different pieces of information. A user starts with one long high-entropy passphrase that is highly memorable despite its length, and a 4-digit PIN. The process uses an algorithm that is easy to remember and implement, and finally, for each service where a user needs a password, they need to choose a small four-letter string called the *cue*.

4.2.1 Passphrase

The main secret data has two components. The first one is a passphrase of at least 6 English words[b], and the second is a 4-digit PIN of the kind that people are accustomed to associating with bank cards. To generate the passphrase, the user is supplied with a randomly-generated sequence of 6 words, adding words (such as articles or connectors) if they so desire. The 4-digit number is also randomly generated. A simple method — detailed in the next chapter — was also created that can be used to help create a high-entropy memorable passphrase. The analysis uses random words from the same dictionary as this method, which is composed of the 87 691 most frequent correct English words, using Peter Norvig's list of most frequent ngrams [Nor09]. This high number is further explained in the next chapter. In the usability test, participants were encouraged to use an online random word generator.

4.2.2 Password generation algorithm

Each time the user needs a password for a new service, they need only apply the Cue-Pin-Select algorithm.

The algorithm makes a 12-character password in 12 steps. Let's suppose a user has created the passphrase *parallel major domain disastrous divergent waterways* and that their PIN is *6908*. Say they are making the password for their Amazon account. They start by coming up with a 'cue': a 4-character string corresponding to this service, say *amzn*. This cue will then be used to extract password parts from the passphrase.

Figure 4.1 shows this process, where each operation corresponds to a colour, creating the password `majrouterusd`, using Algorithm 4.1.

[b]The choice of English was for the entropy analyses and the user study, but the scheme is adaptable to any language that is written in an alphabetic system.

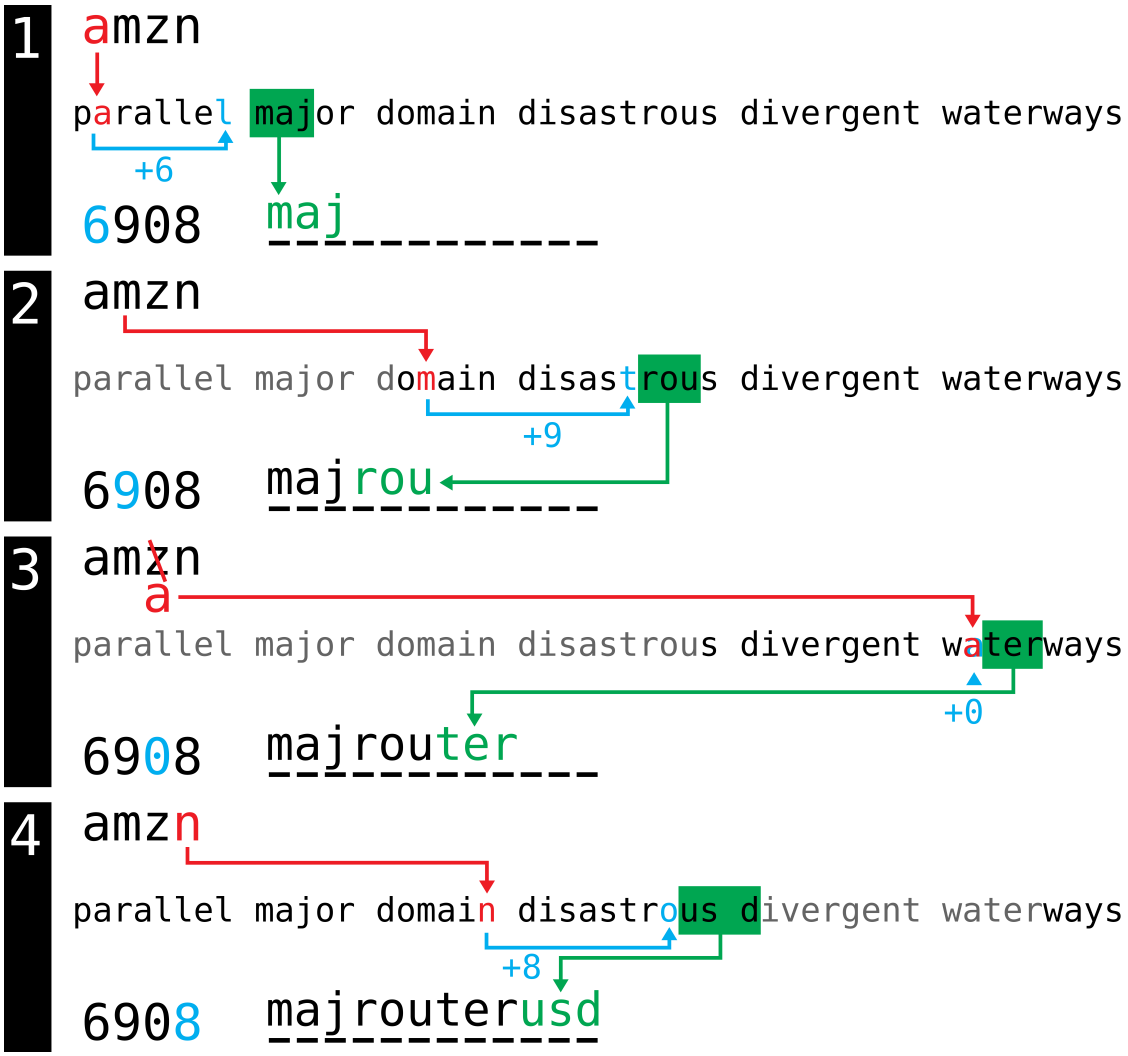

Figure 4.1: Running the four phases of Cue-Pin-Select using AMZN as a cue and *parallel major domain disastrous divergent waterways* as a passphrase.

Once they have chosen (or remembered) their cue, they proceed as follows:

1. They look for the first letter of their cue, `a`, in the passphrase. In our example, this would be the first `a` found in the word `parallel`. They then step though the letters indicated by the first number of their PIN, in this case 6. This would be the last `l` of `parallel`. They add the next three letters in their passphrase to their password, `maj`.

2. They look for the next letter, `m` from where they left off. This leads them to the `m` of `domain`. They skip 9 letters, getting to the `t` of `disastrous`, and add the next three letters, `rou` to their password.

3. They look for the next letter, `z`, but can't find it. As the letter in the cue isn't in the passphrase, they look for the next letter in the alphabet: `z` is then replaced by `a`, and they continue where they left off. The next `a` is the first one in `waterways`, and as the third number in their PIN is 0, they take the next three letters, `ter`.

4. For the last step, they have to look for an `n`, but reach the end of the sentence, they then continue from the start, get the `n` of `domain`, skip 8 letters and end up with `usd`.

5. They are then left with their password: `majrouterusd`.

Data: Passphrase $PHRASE$ of at least 6 random words
PIN of 4 random digits
service name $NAME$
Result: String S of 12 characters

```
1  begin
2      From NAME, create string CUE of four characters
       /* User-chosen, which should be easy to remember                    */
3      LEN ⟵ Length(PHRASE)
4      INDEX ⟵ 0
5      S ⟵ []
6      for i = 0 ; i < 4 ; i++ do
7          LETTER ⟵ CUE[i]
8          while LETTER ∉ PHRASE do
9              LETTER ⟵ letter following LETTER in the alphabet
10         INDEX ⟵ index of next occurrence of LETTER in PHRASE after
               INDEX
           /* Or wrap around to the first occurrence if the end of
              PHRASE is reached                                            */
11         INDEX ⟵ (INDEX + PIN[i] + 3)  mod LEN
12         S ⟵ Concatenate (S, PHRASE[INDEX − 2, INDEX − 1, INDEX])
           /* All the indices are modulo LEN                               */
13     Print S
```

Algorithm 4.1: Cue-Pin-Select

4.2.3 Finding forgotten passwords

As the procedure is deterministic — for a given passphrase — the only variability comes from the cue. In case they forget their original cue, the user should be able to find it within a few tries, from which they can derive the whole password. However, there is another simpler option: the cue and the PIN could hypothetically both be written down by the user, as the security analyses don't assume that they are secret. This way, only the passphrase stays secret, and it is the most frequently rehearsed bit.

The analyses in the coming section pertain to the model shown here. Variants to the algorithm can be introduced for making new passwords or responding to various password requirements. Some representative variants will be studied after the analysis below.

4.3 Security analysis

As a combinatorial analysis of all combinations of words from the dictionary with the algorithm would be intractable and highly dependent on specific properties of our dataset, analyses here rely on Monte-Carlo models. The entropies are computed exactly from the k-grams index, the list of k letter sequences present in sentences made from the dictionary.

4.3.1 Preliminary considerations

One of the main assumptions used in the following analyses is that the distribution of three-letter trigrams composing each password is very close to the distribution of a random trigram taken in a random passphrase. The PIN is an essential part of the randomisation mechanism. It is important because simply reading a sequence of characters in words when reading the cue letter `q` without the PIN step would give trigrams like `qu` where `q` is followed by `u` in 1248 out of 1266 cases, with only 2.7 bits of entropy. An `o`, however, would give 10.4 bits as it reveals little information on the following characters.

Distribution uniformity is nearly achieved as the number of characters stepped over each time through the passphrase is random enough that the probabilities of landing on any letter of a given word are quasi-uniform. The simulation shown in Figure 4.2 presents four curves[c] representing the probability distribution for the number of letters stepped over (one for each letter in the 4-character cue). Since it is much bigger in expectation than an average word length of 9, the probabilities of landing on the n-th letter of a word are then close enough to uniform along n to provide no real advantage to an adversary.

The length of the passphrase itself follows a Bell-like distribution (being a product of distributions that are themselves Bell-like). It has 99% probability of being between 33 and 65 characters long, centred around 48 and has a large variance. As a consequence, with high probability, the second trigram comes later than the first in the passphrase. However, thanks to the large variance in the probability function, the probabilities of the second trigram preceding or following the third trigram are not too far apart, and the same can be said for all the other pairs. Figure 4.3 shows the logscale distribution of passphrase lengths (exact computation based on our dictionary).

[c]The shape of those curves might seem to follow Zipf's law [HSGMS02], with the number of letters covered being inversely proportional to the letter's rank frequency — to which an offset has been added because of the random PIN. However, in such a case the maximum would be reached with fewer than 10 letters stepped over.

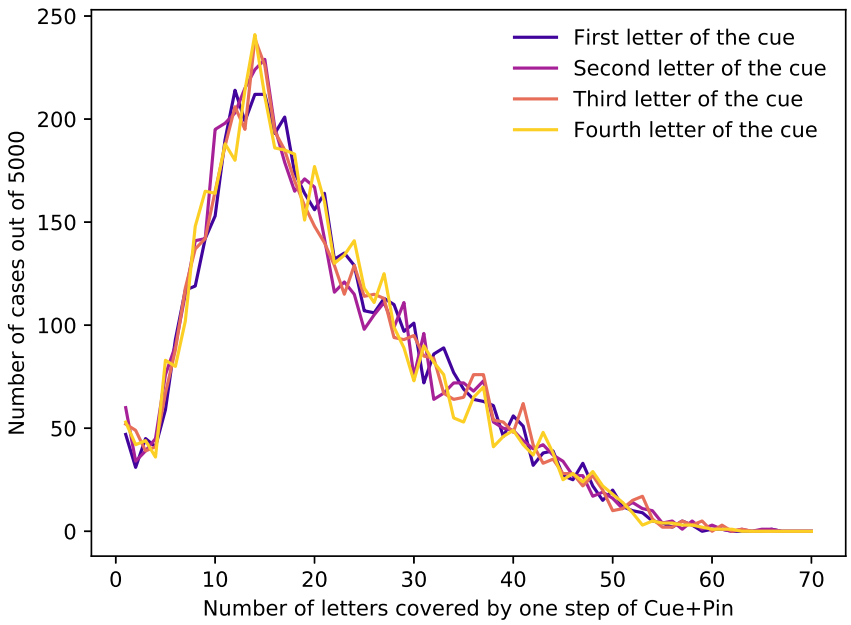

Figure 4.2: Distribution of the number of characters covered in one step of Cue+Pin, obtained by simulation on 5 000 (passphrase/cue) pairs, for passphrases of average length 48.

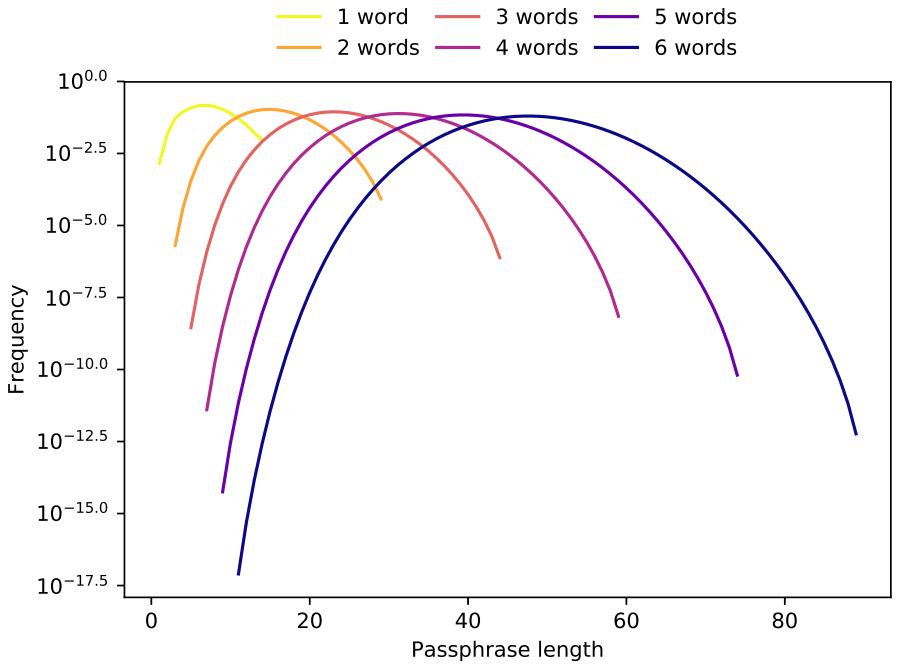

Figure 4.3: Distribution (in vertical logscale) of passphrase length for passphrases made up of k words. The recommended passphrase length of 6 words corresponds to a peak around length 48. Most of the passphrases generated this way are secure against multiple-plaintext attacks, although the few that are under 35 characters are more vulnerable than the others.

4.3.2 Scalability

Publishability

The scalability of a scheme corresponds to two different notions. First, it should be scalable in number of users. Any person who uses an obscure but adequately complex scheme will be protected by a lack of specialised attacks targeting it. This is not true for a scheme used by millions of people. As such, all the threat models should assume Kerckhoffs's principle that only the key (cue, PIN, and service code) are private, the algorithm being public. This corresponds to [BV15]'s notion of *publishable*. In our situation, there could actually be a small positive impact of large-scale implementation, in that people using it in a variety of languages reduces the possibility of statistical and dictionary attacks, marginally increasing the general level of security (as opposed to only American English users).

Creating many passwords

The second type of scalability corresponds to the number of passwords used by a single user; frequently using a scheme should not make it more vulnerable (besides the higher risk of multiple plain-text attacks). For a given attacker with specific computing resources, knowing some plain-text passwords, and other information, the probability of uncovering passwords should only marginally increase with the number of passwords created by the user through the scheme. A simple unscalable example would be a system that solely depends on a passphrase composed of four sections, where the user randomly selects two sections each time they need a password. If they use this scheme less than 3 times, assuming each section has sufficient entropy, passwords don't reveal each other. After 7 uses, however, some of the passwords will be repeated. On the other hand, while having a completely new password for each new service is infinitely scalable, as described above, it will require some way of remembering the passwords, which introduces vulnerability.

For Cue-Pin-Select, it is enough to show that all the passwords generated will be different from each other. It's clear that it should be the case in general, when the user has different cues (in particular, ones that don't have the same first three letters). However, a simulation where each passphrase generates 20 passwords demonstrated that the average distance between two passwords is close to what would be expected from two random strings (at most a few letters being shared). The edit distance between the two closest passwords generated was also calculated (corresponding to the risk of having one other password stolen when the worst password is stolen). This showed that even in this worst situation, in more than 99% of cases an adversary would have to change at least three letters (a quarter of the password, although they don't know which one, corresponding to 15 bits of security[d]), assuming they already possess one of the two closest passwords.

[d] Although 15 bits of security might seem low, it stills corresponds to more than 10 000 login attempts, assuming that the adversary is lucky and already knows the closest password and not just a random password.

4.3.3 Brute-force and dictionary attacks

Attacking the password

Current entropy recommendations against brute-force attacks vary from 29 bits to 128 bits of security, depending on the attack model [ESC05]. One common recommendation proposes 36 bits of security on any given password for web services; such a password would require 1 000 tries per second for one year to break. Assuming the attacker uses online servers to distribute the attack, in 2018 this would require more than $1000 per password, even with strong economies of scale [Ama18].

In our case, assuming the adversary knows the scheme used, a smarter attack would be to guess which trigram is used in each position. However, an analysis of the distribution of trigrams in the dictionary shows that each trigram adds around 13 bits of entropy. To get this number, we computed explicit probabilities for each potential trigram in English using the SOWPODS dictionary — not only within words but also trigrams crossing over words. We also assumed one additional hypothesis: that the start of the trigram in the word is uniformly distributed. This is not entirely accurate, as it depends on the letter chosen in the Cue phase, but the Pin phase adds enough uncertainty to make it quite uniform. Explicit computation on trigrams gave the amount of entropy per trigram — within our dictionary — depending on the type of trigram, which can be either within a word or across two words. Computing them showed that the entropy is highest for trigrams composed of the last letter of a word and the first two letters of the next, with 13.95 bits. The lowest was for trigrams composed of the last two letters of a word and the first of the next, with 11.42 bits, and trigrams within single words had 13.17 bits of entropy. As the latter are by far the most frequent type of trigrams, it is reasonable to assume that each password has around 52 bits of entropy. This is close to the optimal performance of uniform alphabetic passwords of length 12, which have 56.41 bits of entropy.

Attacking the passphrase

The passphrase is much more valuable than any of the passwords; however, it also has much higher security. Indeed, the six mandatory words are uniformly distributed among a dictionary of 87 691, leading to a raw entropy above 98 bits. Adding the PIN gives 111 bits of entropy, way more than any user could reasonably use, even against distributed attacks. Two factors reduce this value: the user chooses the order of the passphrase, which can reduce entropy by 3-7 bits depending on the model, and they can also redraw random words a few times if they don't like the first ones (removing one or two bits). This small cost is partially compensated by the fact that they can use auxiliary words. Overall, assuming the worst case and finicky users, the passphrase and PIN should still give at least 102 bits of entropy. Against dumb brute-force attacks, it would have more than 210 bits of entropy, confirming the problem with using raw entropy without specifying the adversarial model.

Resistance to plain-text attacks

Plain-text attacks are one of the main vulnerabilities found in most user behaviours today, generally stemming from password reuse. This is also where typical methods as described by LifeHacker fail [Lee14]. Assume that, with the drop in computing costs, the adversary tries not just the exact password they have access to but also simple variants of

it. The remaining entropy should stay high, even assuming that the adversary knows both the method and at least one plain-text password [GF06].

The scheme was designed to provide high security even in the event that one (or even a few) of the passwords are compromised, which can happen independently of the user's best practices. As said earlier, trying to guess one password from another in Cue-Pin-Select is a hard problem in the general case, as the edit distance is great (only marginally lower than the edit distance between the password and a random string). The easiest way of attack then seems to go through the derivation of the passphrase from a password obtained by the attacker.

Plain-text attacks. To analyse the security of the passphrase from plain-text attacks, suppose that the adversary knows not only the plain-text but also the length of the passphrase and the position of the plain-text inside the passphrase. This gives way more information to the adversary than is realistic, due to the variation in passphrase length discussed above. Even in such a case, we can show that it is hard to find the passphrase from a single password.

10^4 random (passphrase/cue) couples were computed to get in each case a passphrase where only certain characters were revealed. Dynamic programming was then used to compute the number of passphrases that used exactly 6 words, compatible with the revealed letters and had the right length. This gave the number of potential combinations in each case, which is shown in logscale (hence corresponding to the number of bits of entropy) in the top curve of Figure 4.4.

This shows that Cue-Pin-Select can guarantee a minimum of 40 bits of entropy in case of a plain-text attack, with an average of 54 bits (the standard deviation is 6 bits), and a maximum of 79 bits.

The curves for the remaining entropy, when the lucky adversary has access not only to the length and positions of revealed letters, but also to either two or three passwords, are shown on the bottom curve of Figure 4.4 (5×10^3 runs each).

In those two curves, the average number of bits of entropy left is respectively 32 and 20 bits. However, a large standard deviation (around 9 bits in both cases) and a variability in passphrase length means that in degenerate cases (which happened once in each group of 5 000 simulations of the adversary having several revealed passwords, the length and position of the revealed letters), a double plain-text lead to only 7 bits of entropy, and a triple plain-text completely revealed a passphrase. This is to be expected as this would reveal up to 36 characters, and close to 2% of passphrases are smaller than that. As some of the plain-texts can also give redundant information, the maximal entropies left were 64 and 56 bits.

The guaranteed high level of entropy against at least two plain-text attacks means that users should be secure if they maintain good security hygiene and change their passphrase when they think they have been compromised. Even adversaries with decent computational means and knowledge of the system used should have no real chance of cracking their passphrase.

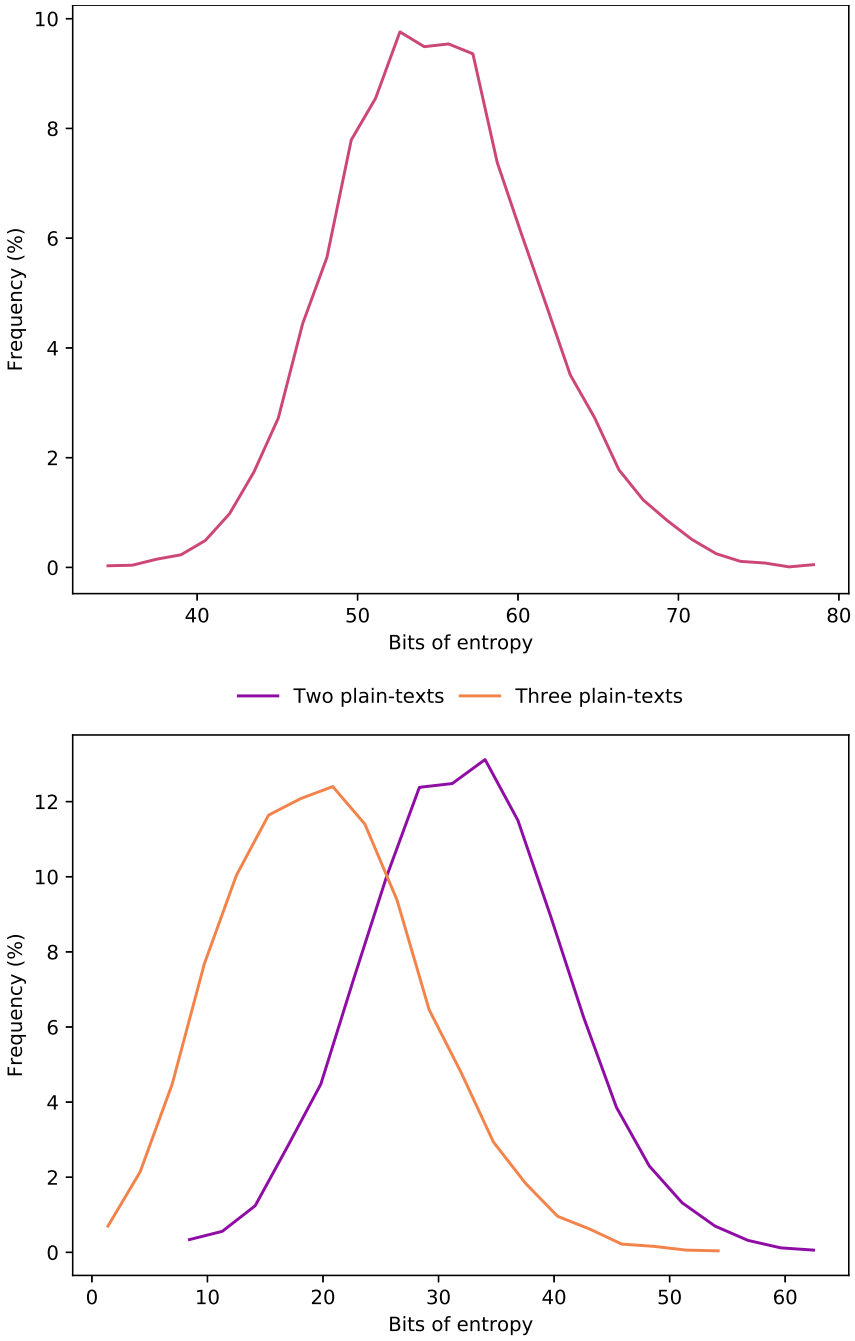

Figure 4.4: Bits of entropy left on a passphrase when a plain-text and the position of its letters are revealed. The top figure corresponds to a single plain-text, while the bottom one features the curves for 2 and 3 plain-texts. Obtained by simulating 10 000 random (passphrase/cue) couples – for the top figure — and 5 000 (passphrase/cue 1/cue 2/cue 3) tuples for the bottom figure. The bits of entropy come from the exact number of possibilities remaining using the passphrase length and the revealed passwords (computed from the cues) as constraints.

Resistance to side-channel and other attacks

Schemes have been proposed that are resistant to brute-force while requiring little effort on the user's part, but require some computation, or storing of information on a trusted device. This can be as simple as a persistent physical memory, corresponding to writing passwords in a notebook, using a password manager on a potentially vulnerable computer (e.g. to keyloggers), or, as in [BBD13, BBDV17], requiring semi-secure computation in the form of challenges from a computer. It can also include hybrid methods such as the password card [Top10, Top11], in which obtaining the card does not give complete access but reduces the entropy of the owner's passwords to about 10 bits each.

As Cue-Pin-Select can stay entirely in the user's mind, it should be entirely secure against side-channel attacks, as the only links between the passwords are entirely immaterial. Those passwords are the only information available no matter the adversary's means, so the security of the scheme corresponds exactly to the security shown above.

This, however, ignores two possibilities. The first is that someone could know the user well enough that they could guess the user's choices. While other schemes that rely on mental association are also potentially vulnerable to someone who knows the user extremely well (even more so if they also have access to computational power to get through the last bits of entropy), Cue-Pin-Select does not. This is why the words of the passphrase should be generated randomly, in a way that doesn't depend on the user's typical choices.

The second possibility is more down-to-earth: some users might write down their passphrase to help them create their first few passwords, or to create a new one. As long as they destroy this physical (or digital, if written in a text editor) evidence, they still have the same level of security, but it is a behaviour that should be discouraged, especially as it is possible to perform the task mentally, as shown in the usability test.

4.3.4 Remaining threats

One smart way of attacking this system relies on finding a big intersection between two passwords. This mainly happens when two cues are extremely similar — ideally sharing the first three letters — leading to very close passwords that can leave the adversary with only three characters left to guess. This could push some hackers to target the user data of services whose natural cues might be close to the ones of valuable services.

There is, however, a simple way to lower the risk: telling users to create their cue in a memorable way, while trying to avoid very similar cues: if they need cues for "GoDaddy" and "GoAir", they should choose `gdad` and `gair` for their cues, instead of `goay` and `goai`. This is reasonable, and a quick simulation shows that, even without changing a single letter, the median number of cues generated before a real collision is 79, quite above the number of accounts most users have at any point [Cen14].

The scheme is *analysable*, as it is deterministic. We have also shown that the scheme is *publishable* and *secure* against most known attacks. We must now look at its usability.

4.4 Usability

Regardless of how secure a password is, if it is too hard to make, it will be reused. If it is too hard to remember, it might be stored in a possibly insecure way, such as on paper or in a file. Usability constraints of retrievability, low effort, and adaptability are all critical to the success of a scheme.

4.4.1 Retrievability

One of the biggest sources of online frustration [Cen14] is forgetting a password and trying several plausible ones before abandoning and resetting it — when possible. This can be compounded by the fact that the next few proposed passwords might get discarded as they correspond to past passwords, pushing the user to create ever harder passwords, getting confused about which ones worked and so on, and forgetting them even more frequently. Moreover, frequent resets can pose security risks by themselves.

Passphrase

In the case of Cue-Pin-Select, the information needed to retrieve the passwords is easily memorable or retrievable, and generally both. The most important is the passphrase itself, which should be easy to remember by being quite short [Mil56] and meaningful to the user. Three factors play a role in its memorability. First, allowing users to *create* their own passphrase from six given words by manipulating the order and having the opportunity to add their own words makes it more personal, safer in its extra length and still easier to remember. Better methods to make such passphrases are addressed in the next chapter.

Second, more than 35% of users need a password for a new service at least once per week, and more than 90% need at least a few per year [Cen14]. Repeated use of the scheme will serve as rehearsals and cement the passphrase in their memory.

Third, if the user never uses their passphrase or the generated passwords, it has low utility for them, and no scheme would truly work in such a case. However, there is the possibility of a user never creating new passwords after an initial period and memorising the ones they have. In such a case, those passwords could serve as strong information that would help their memory.

The passphrase becomes more memorable after the user has started using it. It is also retrievable if they forget part of it after a period of disuse. Together, the last two correspond to the *self-rehearsing* constraint mentioned in [BV15].

PIN. As can be seen from the global use of 4-digit numbers for credit cards, phone passwords, and door codes, people are used to and capable of remembering this kind of PIN. Despite this, some users could forget their PIN. In such a case, it would be easy to find it back using only the passphrase and one of their passwords. Finally, if some users struggle with numbers and might risk forgetting both the PIN and their passwords, we provide a variant that does not require it (albeit at a small entropy cost).

Cue. As the cue is short and users are discouraged from having a complex cue, it should be memorable. More importantly, it is retrievable as there are only a few imaginable cues each user could create from a given service, so a few tries would be enough to get the cue back. Also, it is possible to write down the list of cues in a file, as the security analyses don't assume that they are secret. This is especially true for the variant where the user has to change their password frequently.

Password. The password is the least memorable piece of information, being composed of 12 pseudorandom alphabetic characters. However, any secure password of such length will also be hard to remember, unless it shares strong similarities with other passwords (thus making it vulnerable). Despite this, thanks to the fact that it is created from parts of words

and through associative memory and repetition, users should be able to remember their most frequently used passwords. Finally, the password is easily and entirely retrievable from the other pieces of information in a quick and deterministic fashion.

When a user forgets their password, which will happen, their first step is to rerun Cue-Pin-Select to obtain their original password. However, that might not work if they are starting from a wrong cue or have forgotten about special constraints. In such a case, the user just needs to look at the password constraints on the service they are using to figure whether they had added any special characters (which is a deterministic process). This means that if they know the constraints and their passphrase, the only unknown left is the cue, so at most a few tries would be needed.

4.4.2 Low effort

Initialising the password scheme, creating or retrieving a password, and entering it should all be tasks that are not difficult for users, in both time and effort.. If they are hard, the user will resent it or make mistakes as they have other goals for using services than simply securing them. There are also many cases where it is strongly advantageous to have no dependence on physical devices (such as when one is in public, forgets their computer, or tries to give their password on the phone). Hence, having a computer, or even a pen and paper, should only marginally help the user and it should be feasible to use the password scheme without those (see user study below). This corresponds to the constraints of *human usability* and *agent independence*.

The efforts needed for mental password managers can be split into three categories, as long as we're following the principle of *self-rehearsal*:

- Time and difficulty to learn how to use the scheme and initialise it (such as by creating a passphrase).
- Difficulty to remember and enter passwords once it is used in everyday life.
- Effort to get your password back when you lose it.

Generating the initial passphrase and PIN is easy, as one only needs to get six random words and a random PIN from any generator (some being findable online), and organise them in the order of their choice.

To create a new password, one starts by generating a cue which is immediate as it should be the first 4-letter string that comes into the user's mind. The rest consists of counting and moving 4 groups of 3 characters. It has only 3 steps between adding to the password being typed — 12 steps total. It requires no mental arithmetics, so it should be accessible to people with dyscalculia and should not even require a piece of paper once the user is familiar with the system.

Following the findings of Chapter 2, passwords generated by Cue-Pin-Select should be easy to enter, being composed entirely of alphabetical characters (and when absolutely necessary the required one or two special characters), but the user must first remember the password, and this is not a given. We can distinguish three main cases:

- Frequently used passwords (entered at least every few days): they tend to be remembered by the users even in case of relatively high complexity. As this also applies to passwords created using Cue-Pin-Select, frequently used passwords should not suffer from increased effort or loss of performance.

- Rarely used passwords (entered at most a few times per year): they tend to be forgotten by the users, leading to frustration, many tries, and password resets. For Cue-Pin-Select, this only leads to password regeneration, which requires recomputing it from a small remembered — or stored — cue and is still lower effort compared to trying a few times before resetting the account.

- Passwords used with medium or variable frequency: users might forget them, and a simpler password could be more memorable than one generated with Cue-Pin-Select. However, Cue-Pin-Select can be used to regenerate the password without changing it, giving the user one more rehearsal opportunity. On the other hand, with the usual password schemes, having to reset it and pick a completely different one wipes out the previous effort made to remember it, even if it happens less frequently, making it costlier in the long term.

4.4.3 Adaptability

It then seems that for most users and use-cases, systematic use of Cue-Pin-Select should be easier. That, however, requires the scheme to be always applicable, no matter the constraints imposed by the service provider. It must then have no dependence on the password character set, length, and reuse change policy that a service imposes. Password requirement can also be contradictory between different services (like short maximal length constraints or forbidden numbers). Any exception that prevents the user from using one scheme drastically diminishes the interest in using that scheme. Among protocols that are now known to create usability errors [KSK+11], some still ask users to regularly change their passwords, avoiding any that have some similarity to ones used previously in a large time frame. Some users also forget their password despite their best efforts or make a typo while defining it for the first time. Adaptable password scheme must then include the possibility of creating new passwords for a single service without introducing confusion as to which one is the current password, as users dislike changing habits and will keep one scheme (or one password) for multiple years at a time.

The passwords created by Cue-Pin-Select heretofore in this chapter have been in lowercase alphabetical characters. This provides enough security by itself but could be changed as needed to work with idiosyncratic password requirements. Even the most trivial extension that takes care of this for each of the following requirements does not reduce entropy:

- If the password requires capitalisation, the user should remember to capitalise one letter in their cue, and capitalise the corresponding three letters of their password.

- If it requires a number, the user can start with 0 and insert it at the center of their password, then increase it by one each time they renew this password.

- If it requires a special character, they can pick one in particular, like "!" that they will put at the end (or in the center) of every password that requires it.

- If it has a maximal length, they can just truncate the password without losing too much entropy (and a service that limits passwords to lengths smaller than 12 probably has bad security in any case).

- If it requires the user to change their password at set intervals of time (such as every month), without repetition for a certain time, they can change the first letter of

the cue (or the first two letters) by cycling slowly over the alphabet. AMZN would become BMZN and then CMZN and so forth, and the passwords would be strongly different each time (with high probability), as it changes the starting point.

These simple changes give ways to adhere to the security constraints required by service providers without reducing the entropy of the password or significantly reducing usability.

4.5 Testing Cue-Pin-Select

With strong arguments in favour of the usability of Cue-Pin-Select, a usability test was organised, with 11 subjects using it for short tasks each day for four days. Their personal data was neither stored nor shared. We encouraged them to take the benefit of learning this simplifying system for later use for their own passwords. The group consisted of five men and six women of diverse backgrounds, varying in ages from 18 to 65.

4.5.1 Protocol

Users were initially given a document explaining what they needed to know, including how to use Cue-Pin-Select. The document explained that they could leave at any time, that they should not use the passwords they would generate over the experiment as we would ask for that information, but that they were free to use the system with another passphrase after the experiment. They were progressively given three sets of tasks, each lasting a few minutes. They were then told to follow the self-administered tasks at the rate of one in the morning and one in the afternoon, and send us the results, as well as the time it took them to accomplish each task. The list of tasks is as follows:

- Day 1, task 1: Create their passphrase and PIN (task 0). Create two passwords with cues already provided. Create cues and then passwords for two services (*Hotmail* and *Yahoo*). After this task, they were given feedback to explain potential errors in making a password they might have done.
- Day 1, task 2: Create a password with a provided cue, and then a (cue/password) couple for *New York Times*. Told to try to remember their passphrase, as they would have to recall it from memory from the second day onward.
- Day 2, task 1: Recall the passphrase, then create a password with a cue provided and a (cue/password) couple for *Twitter*.
- Day 2, task 2: Create a (cue/password) couple for *Snapchat*, and recall the one they did for *New York Times*.
- Day 3, task 1: Create 2 (cue/password) couples for *Reddit* and *AT&T*.
- Day 3, task 2: From this step on, the participants were told to apply the algorithm entirely in their head (writing down only the letters of their passwords as they computed them). Create 2 (cue/password) couples for *The Guardian* and *HP*.
- Day 4, task 1: Create 2 (cue/password) couples for *Spotify* and *Gmail*.
- Day 4, task 2: Recall the couples they created for *Snapchat*, *AT&T* and *HP*.

After the experiment, users were presented with questions asking them if they had trouble remembering their passphrases (instead of asking them if they cheated), which tasks they had done with pen and paper and which in their head (as it appeared that some switched earlier than in the instructions), whether certain aspects made them lose some time, and whether they would consider using it in its current state.

4.5.2 Results

Only one participant had trouble remembering their passphrase on the second day (they were missing one word) and had to be given it back. Most of them developed mnemonic schemes to help them remember (or increase their speed), such as splitting it into two sentences or creating mental imagery. A few had trouble remembering their cues.

Some users had trouble following the initial instructions (or found them unclear) as they were not very well explained. The most common mistake was restarting from the start of the word at each new cue. Feedback was given after the first set of tasks and first password to teach them to use Cue-Pin-Select correctly. By the second set of tasks, all users could correctly execute the algorithm (with only three recorded mistakes in that task and the following).

As is shown in Figure 4.5 — with detailed data in Table 4.1 — there was a general speed-up over the course of the experiment. We can see a speed-up within each set of tasks, and a slowdown between sets, with a stronger slowdown on the afternoon of the third day, as all users had to compute passwords mentally and couldn't use pen or paper or their electronic device anymore[e]. As we can see from the end of the table, speeds were still increasing at the end of the fourth day, but that's when we had to stop the experiment. The real time taken by users who regularly use this method can then be inferred to be lower than the one attained at the end of the experiment, although we do not know by what margin.

Users saw the algorithm's value, despite the lower case only, no special characters demonstration. Four out of eight users who gave feedback said they would use this system, at least for their important passwords. Two were hesitant; one thought that Cue-Pin-Select wasn't adaptable enough (indeed, they were not shown how to use capitalisation or special characters). Finally, one said it didn't fit their personal security/usability needs, and they wouldn't use it.

4.5.3 Feedback

Multiple participants observed that there was a strong cost in time (and usability) when they had a letter that was absent from their passphrase and had to go through their passphrase multiple times. They said that if they ever used the scheme again, they would make sure to have more vowels in their passphrases, as well as a higher letter diversity (which is also good from a security standpoint). Two of them also mentioned they would prefer a PIN with lower numbers.

[e]Two users decided to do all tasks mentally from day 2 onward, and their data was not counted in the tables for days 2 and 3, but they show the same learning behaviour as the others. Instead of writing down the passwords as they created it, some users tried to create all of it before writing it down; this data is included in the tables.

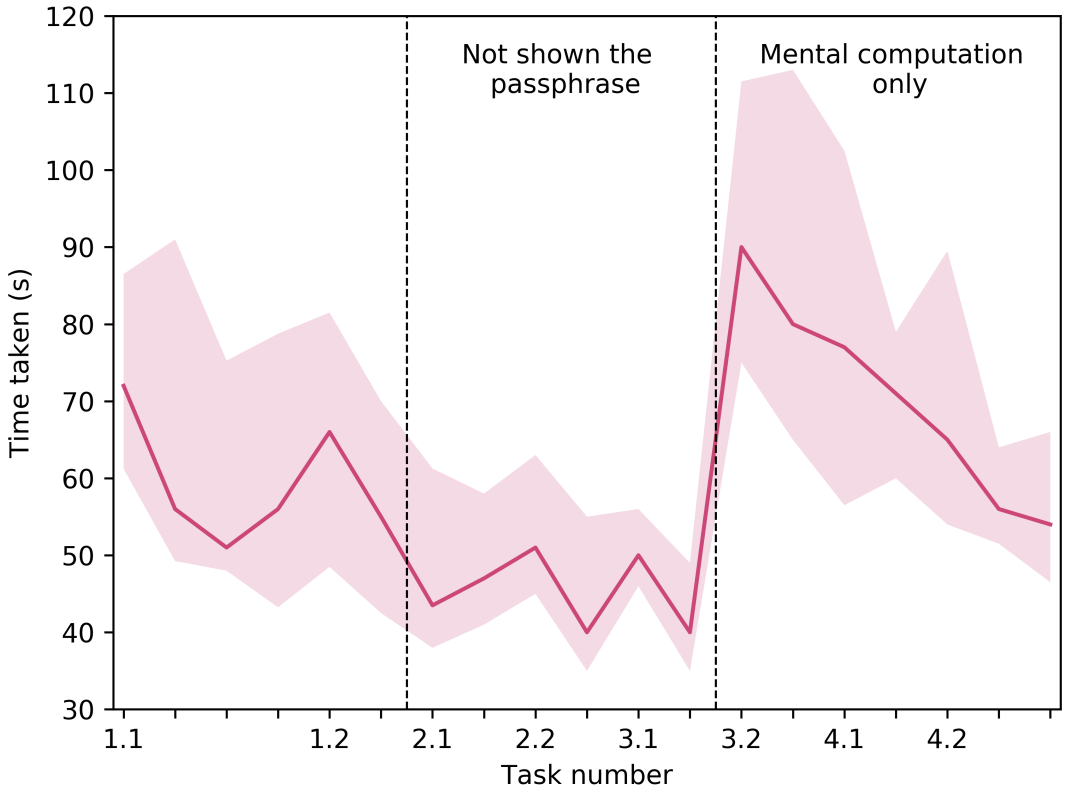

Figure 4.5: Time taken by participants to create a password over the course of the experiments. The solid curve corresponds to the median time and the shaded area represents the time taken by people between the 25th and 75th percentiles. The sudden increase at 3.2 corresponds to the switch to mental-only tasks.

Task	1.1a	1.1b	1.1c	1.1d	1.2a	1.2b	2.1a	2.1b	2.2a	2.2b	3.1a	3.1b
Mean	89	82	72	63	70	59	50	49	54	45	51	42
Median	72	56	51	56	66	55	44	47	51	40	50	40
Max	233	211	222	108	132	113	87	68	70	61	74	53
Min	47	35	35	32	32	33	30	32	42	31	38	30

Task	3.2a	3.2b	4.1a	4.1b	4.2a	4.2b	4.2c
Average	105	86	81	74	67	58	57
Median	90	80	77	71	65	56	54
Max	220	131	130	117	106	86	71
Min	65	47	46	47	24	33	31

Table 4.1: Time taken by participants to create each password, in seconds. The top table has the times from the first day to the afternoon of the morning of the third day, and the bottom table has the afternoon of the third day and the fourth day. As in Figure 4.5, the increase at 3.2 is caused by the added rule that forbade participants from using pen and paper, having to compute the passwords mentally instead.

4.6 Variants of the algorithm

As it is easy to add vulnerabilities, each variant must be systematically analysed. Here are some natural variants with serious flaws. The first variant of Cue-Pin-Select is shown in Algorithm 4.2. It simply incorporates the modifications mentioned above to make it compatible with most online constraints, not requiring any change in the security analysis. There are many other ways to create variants, and most are problematic, implying that Cue-Pin-Select might be a local optimum in the security-usability trade-off. We analyse a few hereafter.

4.6.1 Lower security variants

Fewer words or fewer characters. Reducing the passphrase from 6 to 5 might be tempting. Computer simulations using the same protocol as before show that with this approach, many single-plain-text attacks would leave the user with fewer than 28 bits of security, down to 16 bits in certain cases. Fewer characters in the password would be imaginable but would lower the individual resistance, and the number of possibilities would be limited as:

- Having 3 passes in the algorithm would lower the entropy too much.
- Extracting 1 or 2 characters would require more passes, and be less user friendly.

Getting rid of the PIN. Getting rid of the PIN might seem to simplify the algorithm. This is a dangerous idea; as described above, this PIN plays an important part in making the algorithm secure. As cues can be easily guessable by an adversary, they could accumulate way more information on the passphrase by guessing the cues (and then could perform targeted attacks to obtain the rest).

If the problem is not using the PIN but remembering it, a viable password generation scheme with a further reduction on memory would come from making the PIN from the passphrase. This alternative that strongly mitigates the risk is to have a PIN that corresponds to the lengths of the last four words (modulo 10). This maintains a high level of security while making the PIN trivial to retrieve.

Using parts of words

Instead of extracting trigrams from the passphrase, it would be simpler to extract larger character-groups from words one at a time. For example, one could take a random letter and the corresponding prefix or suffix to create those groups, which could be faster and easier for the user. Unfortunately, this would reduce entropy so much[f] that it would either require more words, more passes or make the passphrase itself insecure, depending on how many letters are extracted each time.

[f]The multiple potential methods differ somewhat, but the common loss in entropy is due to two factors. The first is that the set of prefixes and suffixes is in fact not that large when compared to trigrams that can include part of a suffix and part of a prefix. The second is the very variable length of prefixes ans suffixes, meaning that some would be composed of a single letter, drastically diminishing entropy, or instead be very long (up to 7 characters), diminishing usability while only marginally increasing entropy.

Data: Passphrase $PHRASE$ of at least 6 random words
PIN P of 4 random digits
service name $NAME$

`/* If the user struggles with numbers, the PIN can correspond to the length of the last 4 words */`

Result: string S of about 12 characters

1 **begin**
2 From $NAME$ create string CUE of four characters
`/* This user-chosen string should be easy to remember */`
3 **if** *Service requires mixed-case* **then**
4 CUE should be in mixed-case
5 **if** *User had a previous cue for this service* **then**
6 $CUE[0]$ and $CUE[2]$ become the next letters in the alphabet
7 $LEN \longleftarrow$ Length($PHRASE$)
8 $INDEX \longleftarrow 0$
9 $S \longleftarrow []$
10 **for** $i = 0$ *;* $i < 4$ *;* $i++$ **do**
11 $LETTER \longleftarrow CUE[i]$
12 **while** $LETTER \notin PHRASE$ **do**
13 $LETTER \longleftarrow$ letter following $LETTER$ in the alphabet
14 $INDEX \longleftarrow$ next occurrence of $LETTER$ in $PHRASE$ after $INDEX$
`/* Or the first occurrence if we reach the end of` $PHRASE$ `*/`
15 $INDEX \longleftarrow (INDEX + PIN[i] + 3) \mod LEN$
16 $TEMP \longleftarrow$ Concatenate $(PHRASE[INDEX - 2, INDEX - 1, INDEX])$
17 **if** $CUE[i]$ *is upper-case* **then**
18 Make $TEMP$ upper-case
19 $S \longleftarrow$ Concatenate $(S, TEMP)$
20 **if** *service requires a number* **then**
21 $S \longleftarrow$ Concatenate $(S, 0)$
22 **if** *service requires a special character* **then**
23 $S \longleftarrow$ Concatenate $(S, !)$
24 **if** *service requires a password of length* $LEN - y$ **then**
25 $S \longleftarrow$ Suffix $(S, LEN - y)$
`/* We avoid the security measures by adding characters that don't change entropy, and remove the first few letters if needed */`
26 Print S

Algorithm 4.2: Adaptable Cue-Pin-Select

4.6.2 Higher security variants

A few variants have either a higher degree of security or more easily provable and analysable security, coming at the cost of a reduced usability.

Using letter-values. The first version of this algorithm behaved differently in the Cue step. Instead of looking for the next letter identical to the one in the cue, the user was supposed to convert the cue into a number, and then advance by that many characters (plus the PIN digit). This means that the distribution of trigrams is even closer to a uniform one, and also means that any modification to the first letter of the cue entirely changes the rest of the password. However, the mental load associated with converting a letter to a number and moving by more than 15 characters on average slows down the scheme and makes it more complex to explain and perform. It is still shown in Algorithm 4.3 for numerically adept users.

Data: Passphrase $PHRASE$ of at least 8 random words
PIN P of 4 random digits
service name $NAME$
Result: string S of 12 characters
begin
  From $NAME$ create string CUE of four characters
  `/* This user-chosen string should be easy to remember */`
  $INDEX \longleftarrow 0$
  $L \longleftarrow$ Length(P)
  $S \longleftarrow []$
  for $i = 0$; $i < 4$; $i++$ **do**
    $LETTER \longleftarrow$Integer($CUE[i]$)
    `/*` $LETTER \longleftarrow$ `n, if` $CUE[i]$ `is the n-th letter in the alphabet */`
    $INDEX \longleftarrow INDEX + LETTER + PIN[i] + 3 \mod (L)$
    $S \longleftarrow$ Concatenate $(S, PHRASE[INDEX - 2, INDEX - 1, INDEX])$
  Print S

Algorithm 4.3: Higher Security Cue-Pin-Select

Using more words. Finally, we could imagine having the same scheme but extracting $k \geq 4$ trigrams from $m > 6$ words. This is obviously less usable, but it increases security by a huge factor, especially in the case of multiple plain-text attacks (when we increase m). For example, going to $m = 8$ guarantees high entropy against triple plain-text attacks. Experimentally, every added word increases entropy by 16 bits, and every added plain-text reduces entropy by less than 20 bits, although those two procedures also increase the entropy variance. This is only true as long as the algorithm still goes through the whole passphrase more than once on average, so adding words until the passphrase has more than 60 characters is a good compromise. Adding more words afterwards would generally be counterproductive, unless $k \geq 4$.

4.7 Discussion

This chapter has introduced the Cue-Pin-Select usable password creation scheme and analysed its security and usability. It can be learned and applied by a novice in less than two minutes and after making a few passwords, all testers were able to create 12-character retrievable passwords in under a minute in their head. Some participants were able to create or retrieve passwords in less than half a minute. The scheme is robust against many types of attacks, including against an adversary in possession of some of the generated passwords. All of the passwords in the scheme are easily retrievable assuming the participant remembers their passphrase. The passphrase itself is easily memorable, frequently rehearsed, and retrievable via associative memory if the participant remembers some of the passwords. It allows a participant to create a cue that works for them. Finally, it is compatible with text-based password constraints and can be used durably without frustration or risk from an evolving constraint environment. In summary, it satisfies [BV15]'s five criteria of *analysability*, *publishability*, *security*, *self-rehearsal* and *human usability*, as well as our criteria of *agent independence*, *adaptability*, and *scalability*.

The analysis performed gives worst-case lower bounds on the security of Cue-Pin-Select, by making strong assumptions on the amount of information available to the adversary. Instead of having clear-text passwords, analyses supposed that they knew the length of the passphrase and the location of each letter, while a significant chunk of the security of the scheme relies on their secrecy. Despite those assumptions, the system guarantees 40-bit security even against single plain-text attacks. Those bounds could be increased by a more thorough analysis of realistic attacks, to prove a higher level of security against multiple plain-texts attacks.

This exercise had several goals. It shows the existence of an easy-to-use password generation and retrieval system. It gains security from not using a computer, from the entropy of a memorable secret, and from its adaptability. It gains usability by being personalised, based on language, and by rehearsing the only things that have to be remembered. It gains robustness in the ease it gives for retrieving any passwords a user does not remember, and for giving the users simple rules to make up alternative passwords for any service.

With a greater speed and no reliance on mathematical primitives, the system is easier to use and more secure than earlier proposals [BV15]. Ideally, we hope to see this working demonstration leading to new families of more secure, usable password creation and retrieval systems. For example, the strong security constraints could be relaxed to get a more usable scheme with a slightly weaker but still strong security features. A thorough analysis of more realistic threat models against Cue-Pin-Select and derived schemes that rely less on private information held by the adversary would help in this endeavour.

We now turn our attention to one primitive that was used throughout this chapter without much thought: passphrases, and how to create secure and memorable ones.

5

Improving Passphrases with Guided Word Choice

5.1 Issue statement and contributions

The previous chapter made use of passphrases, which are built on the principle that a sequence of words might be at the same time more secure (entropy-wise) and easier to remember than usual passwords [SKD+14, TAT07]. As opposed to complex passwords which are hard to remember [KSK+11, YBAG04, PJGS12], passphrases benefit directly from our natural abilities to remember sequences of words [BH74, Mil56]. Although passphrases have been suggested as a possible solution to replace passwords as early as 1982 [Por82], dozens of years later they still suffer from problems similar to the ones passwords have.

Passphrases have been shown to typically be made from insecure, linguistically easy-to-crack patterns [KRC06], like song lyrics or famous quotes. The wealth of human literature often seems rich enough to get some security, but it has surprisingly low entropy. The number of possibilities available by using a simple algorithm to extract a passphrase or password from all published literature can easily be bounded by 10^{13}. If one selects from all this uniformly, the entropy is still only 46 bits, a bit less than a random 10-character alphabetical string[a]. For example, a significant number of passphrases created by the Amazon PayPhrase system were easily hackable, with 1.3 percent of accounts being vulnerable to a 20,000-word dictionary of terms used in popular culture [Bon12, BS12].

In another study [YLC+16], one passphrase method led 2.55% of users to choose the same sentence — *to be or not to be, that is the question.* Despite multiple protocols encouraging users to make personalised sentences, five out of six methods had many occurrences of different users ending up with the same passphrase. With the single method that had no two users choose the same password, the authors still observed some quotes and famous sentences, but few enough not to have collisions on a database of only 777 passphrases. These studies show that letting people choose a passphrase with no constraints leads to low-entropy passphrases, even when giving them instructions to make personalised passphrases.

This chapter is based on research done with Clément Malaingre and Ted Selker.

[a]This is assuming, generously, that there are 150 million books of 50 000 words each, with a choice 10 different algorithms to use.

Although, as mentioned in Chapter 1, the value of entropy as a measure of password strength has been debated [MCTK10, RK18, TAM+13], it remains central for passphrases. Indeed, the concerns for passwords are not directly transposable to passphrases, as a central use-case for those isn't as a replacement for passwords, as this would require too many passphrases and add a significant time cost. Instead, they can be used as tools that can serve as sources of entropy for other password generation methods such as the ones mentioned in the previous chapter [BBDV17]. The usefulness of passphrases then depends on having high entropy — and low guessability. To achieve this, they have to be not only longer but, more importantly, less predictable.

To avoid users choosing common passphrases, one could take inspiration from standard password practice and draw words uniformly from a dictionary. However, this has two drawbacks: first, the passphrases generated are not individualised and can suffer from low memorability. Second, to make sure that the user knows all the words generated, the dictionary from which the words are drawn must be of limited size.

Contributions. This chapter explores a method of guiding the user to choose their passphrase from an imposed set of random words. A usability experiment explores the factors affecting the word choice, the participants' ability to remember their passphrases, and the type of mistakes they make. Both for primary and secondary English speakers, this method leads to highly increased memorability of the passphrases created, with error rates divided by factors between 2 and 6 depending on the type of mistake. Since the user can choose from several words, the dictionary need not be made of only words that are sure to be known to a user, allowing the use of much bigger dictionaries. This means that the decrease in entropy due to linguistic bias, already smaller than 3% of total entropy, can be compensated by the larger dictionaries, increasing final entropy by 20-30% against a dictionary of 10 000 words.

5.2 Method

All the results come from an online usability experiment which explored the impact of creating a passphrase by choosing words from an array.

5.2.1 Protocol

The experiment was hosted on a privately hosted server and accessed remotely over the web. It ran on the Scala Play framework and a PostgreSQL database. Participants were shown the following pages, with instructions at the top of each:

1. A welcome page that gave participants an overview of the activity and informed them of their rights.

2. A question asking their age and another asking the primary language they used.

3. A dynamically generated array of either 20 or 100 words (using between-group design). Participants were told to select 6 words, in the order of their choice, and input them in the 6 text-boxes at the bottom of the page, as is shown in Figure 5.1. The words were presented in five columns of either 4 or 20 words that were not aligned with the input boxes to limit the potential bias of choosing one word per column.

4. A page that repeated the passphrase then prompted participants to repeat it to themselves until they could remember it.

5. A text-box was presented with instructions to type in the first two letters of each of their words.

6. A page was presented, showing an array of words that had previously been presented to another participant. They were then told to try guessing what words the other participant had chosen.

7. A page informing them which if any of their guessed words were correct, and telling them that they could try to guess more passphrases if they wanted, or could continue with the rest of the experiment.

8. A page asking them to repeat all six words from their passphrase in the same order, or as many as they could think of if they didn't remember all of them. If some were missed, they were then presented with their original array of words as a clue and asked to find all six of them.

9. A page thanking them for their participation and inviting them to encourage others to become participants.

A control group was also used, to create a baseline for remembering a 6-word passphrase. They followed a similar protocol with the sole difference happening in step 3. Instead of choosing their words from an array, a sequence of 6 randomly generated words was directly given, while informing them that it had been randomly created.

5.2.2 Word choice

Presented guide words were drawn uniformly (each word having the same probability) from a dictionary crafted for this purpose. This dictionary is based on the first third of Peter Norvig's 300,000 most frequent n-grams [Nor09]. As those 100,000 words still included words from other languages such as "unglaublichen" as well as some non-words like "unixcompile", we only kept the ones which were also in the SOWPODS (the list of admissible words in English Scrabble tournament and other word games). This created a list of the 87,691 most frequent English words.

This might seem a lot, but it includes many words derived from others, forming what are called word families. For example, family, families, familiar, familiarity, unfamiliar, and 23 other words share the same root. Estimates on the number of words known vary from 27 000 to 52 000 for native English speakers [BSMK16], going down to less than 10 000 for non-native speakers. However, those studies generally look at *lemmas* or roots of words, and do not include many derived forms (when including those, the studies show a vocabulary as high as 215 000 words for native English speaking undergraduate students [Har46]). As the participants chose only 6 words from the 20- or 100-word arrays, having a few unknown words in it shouldn't, in any case, change the process or outcome.

To get a variety of behaviours, we experimented with word arrays of different sizes. To give participants a real choice, the guide array to choose from was created to be several times the length of the passphrase created from it. It also needed to not be so long that people got overwhelmed, had to scroll through them, or took too much time making decisions. An array of size 100 was the biggest that could reliably fit on a computer screen,

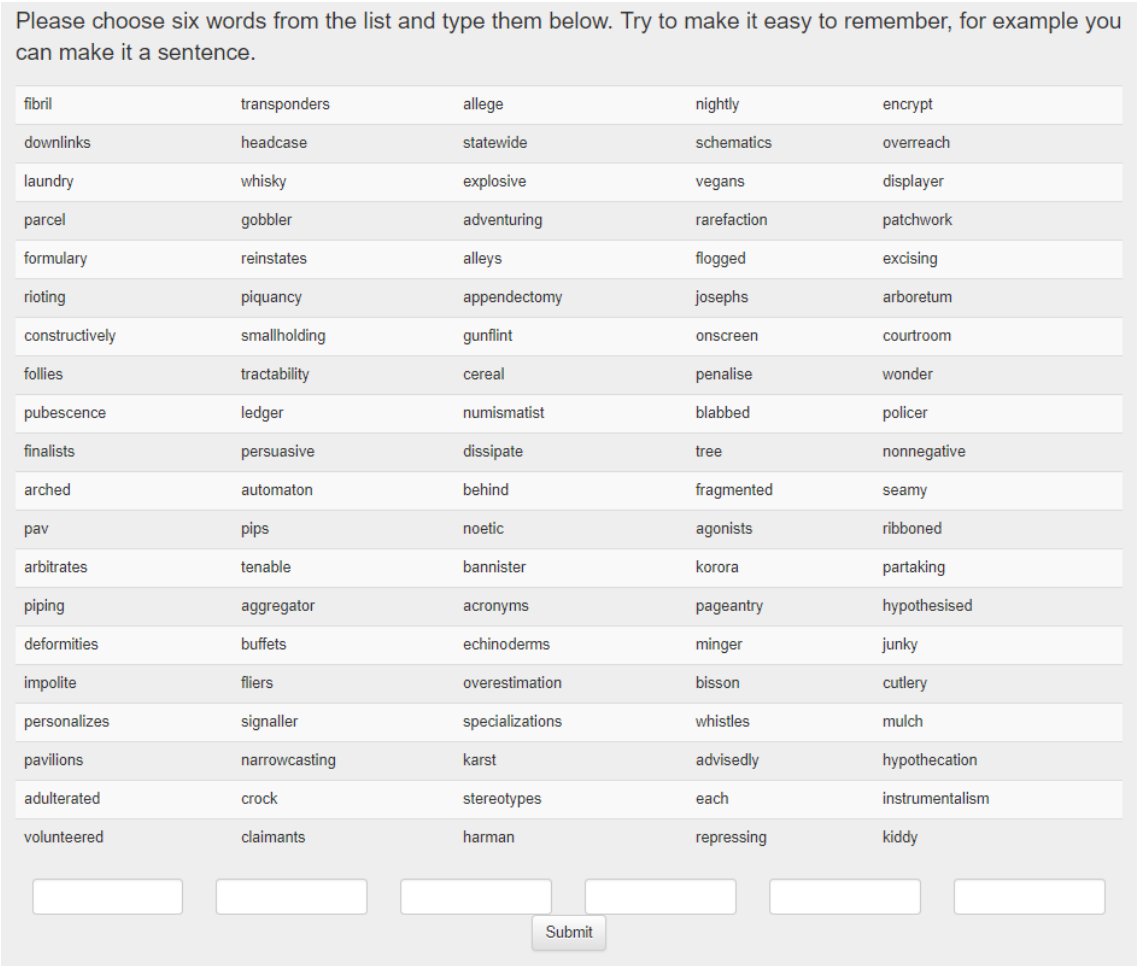

Figure 5.1: Screenshot of the word choosing interface for group "100". Participants had to select six words and retype them in the text boxes below. Selecting and copy/pasting words directly was disabled to help participants remember their words better.

and 20 was the smallest that guaranteed enough possibilities for the participant to choose from (38,760 in total, discounting order).

The experiment collected of the following data for analysis:

- Any information entered in the text-boxes.
- All the words and arrays shown to the participants.
- Time spent on each page.
- List of (keystroke/timestamp) couples.

Unlike the user experiments shown in other chapters, this one collected data as it was created, and the partial data of participants who left the experiment was analysed. To make sure that no one would try the experiment multiple times to improve their performance, IP addresses associated with the participants were temporarily kept. A single occurrence of a second try by a single IP was detected. The initial participant could also have shared their computer and recruited another participant, but we took the conservative approach and deleted the second entry from the database.

5.2.3 Design choices

Passphrases of length 6 were chosen as they provide the length and level of entropy required by Cue-Pin-Select in the previous chapter — although passphrases have many other uses. Moreover, this is compatible with known bounds on memory and information processing ability [Mil56].

The guide array was purposely designed not to line up with the spaces for words in the passphrase below (as shown in Figure 5.1) to avoid confusion in step 3 and separate the guide words from the user-chosen passphrase. Participants were also told to try to make their passphrase memorable, for example, by creating a phrase, rhyme, or sentence from their selected words. Step 5 was meant to help learn the passphrase and check whether it was memorised. Step 6 was then introduced as a distractor exercise introduced to interfere with their short term memory for passphrases. The idea was to eliminate short term memory of their initial passphrase by making them think of someone else's passphrase.

5.3 Demographic information

5.3.1 Participant selection

The principles of informed consent (including the right to quit and the right to privacy), not using people in protected classes, beneficence, justice and minimal deception were followed. All the participants were volunteers, and were informed of the length of experiment and that they could quit at any point. They were told that it was an opportunity to help them test their memory and for us to understand how people typed, and that their typing would be monitored and stored. They were not asked any identifying information and were informed not to re-use their input for security purposes as their data would be collected for analysis. For privacy, minimal demographic data was collected, corresponding to aspects that would be relevant to analysing results.

5.3.2 Recruitment of volunteers

Volunteers were recruited through John Krantz's Psychological Research on the Net website [Kra19]. Volunteers were also encouraged to invite others to participate, through snowball sampling [HG11], so some late volunteers were recruited through social networks.

5.3.3 Statistics

Groups. A total of 125 people finished the experiment. At the start of the experiment, they were randomly assigned to three groups: "20", "100" and a control group. Group "20" was shown an array of 20 words to choose from and was composed of 47 volunteers. Group "100" was shown an array of 100 words and had 52 volunteers. The control group, with 26 volunteers, had their words chosen for them and shown on the screen.

Age. The participants' ages showed a large variation, from 16 to 69, with a notable concentration around 24. The average was 31 years old, and the median 25.

Language. 51 participants wrote down English as their primary language. French (28) and Hebrew (14) were the next two most reported primary languages, followed by Arabic, Norwegian, Russian, and Romanian.

5.4 Results

The results focus on how participants chose words and how different variables affected their choices, on their ability to remember the words chosen, and on how they guessed other people's words.

5.4.1 Word selection

Based on problems recognised in other studies [Blu84, LBP51, LH14, YLC+16, YPR10], our original hypothesis was that word choice would be mostly influenced by three behaviours:

- *Semantic*: participants might choose words that are more frequently used (and with which they were more familiar).
- *Syntactic*: participants might choose words that are compatible with others they chose, to create a sentence with a common structure.
- *Positional*: participants might choose words that are either among the first they read (on the top left corner), or closest to the input fields.

A possible outcome that would make the proposed method ineffective — because of low entropy — is that people could all choose the most familiar or frequent words from the array of n words shown to them. Similarly, strong syntactic tendencies could lower entropy by reducing the number of probable passphrases. On the other hand, positional effects tend to increase the entropy as they don't depend on human biases but on random position, making the distribution more uniform.

Results below showed that the second effect is much weaker than could be expected, with some choosing bias caused by semantic and positional effects.

Semantic effects

To analyse the semantic effects, we used sorted the dictionary by decreasing order of frequency in the n-gram corpus. Frequent words are all within the first few thousand words in this dictionary, with rare words at the end. Figure 5.2 shows the distribution of the words chosen depending on their frequency rank in the dictionary, for each of the two main groups. To make the figure more legible, ranks were aggregated in 30 buckets of 2923 words. Although we can observe a bias in favour of more frequent and familiar words, 23% of words chosen still come from the least frequent half of the dictionary; group "20" used rare words 26% of the time, group "100" used them 20% of the time.

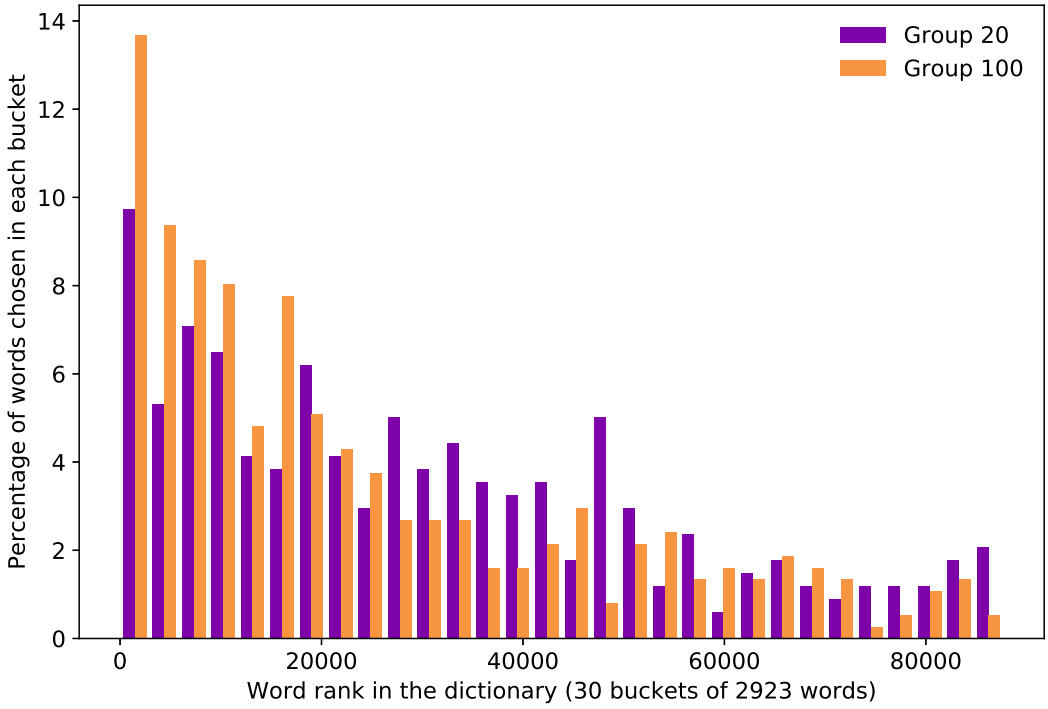

Figure 5.2: Distribution of the words chosen by each group as a function of how frequently they are used. The x-axis corresponds to the word's rank: its position in the dictionary when the latter is sorted by most to least frequent. To make the data more readable, the ranks were aggregated in buckets of 2923-2934 words each (splitting the set into 30 equal parts).

The significant fraction of words chosen from the second half of the dictionary is not just due to some participants getting only rare words, as the following figures show. Figures 5.3 and 5.4 show the distribution of words chosen depending on their frequency relative to the frequencies of the words shown to the participant. This frequency, $F(i)$, is equal to the number of times the i-th most frequent word in the array was chosen. Those histograms also include the distinction between primary English speakers and others, inside each group. When given enough choice — in group "100" — non-primary speakers have a bigger tendency to choose frequent words, with only 15% choosing rarer words. A single participant in group "20" chose the 6 most frequent words in their array, which is more than the expectation of uniform random choice of words[b].

[b] Choosing 6 words at random from an array of 20 gives any given outcome with probability 0.003%.

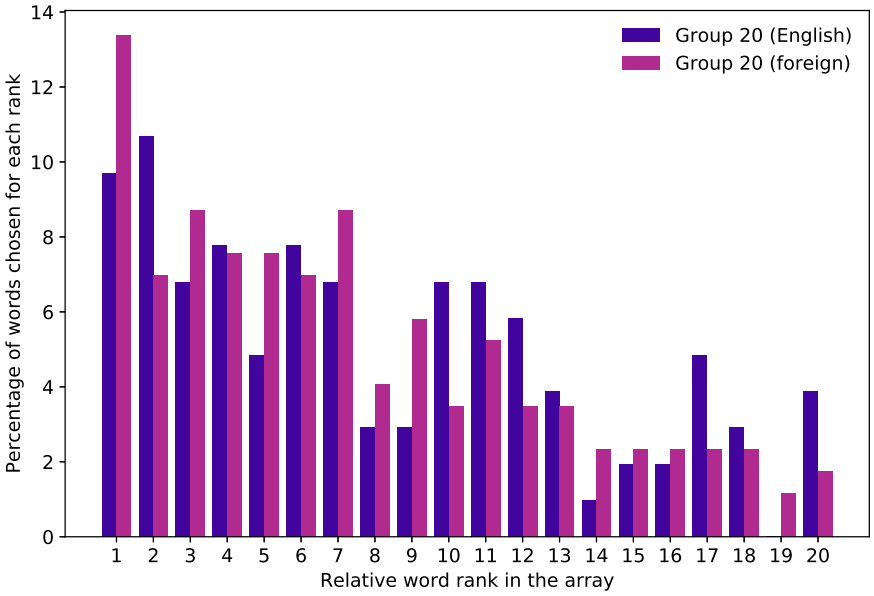

Figure 5.3: Distribution of the words chosen in group "20" as a function of how frequently they are used and whether the participant was a native English speaker. The x-axis corresponds to the word's rank among the other words shown, as sorted by most to least frequent.

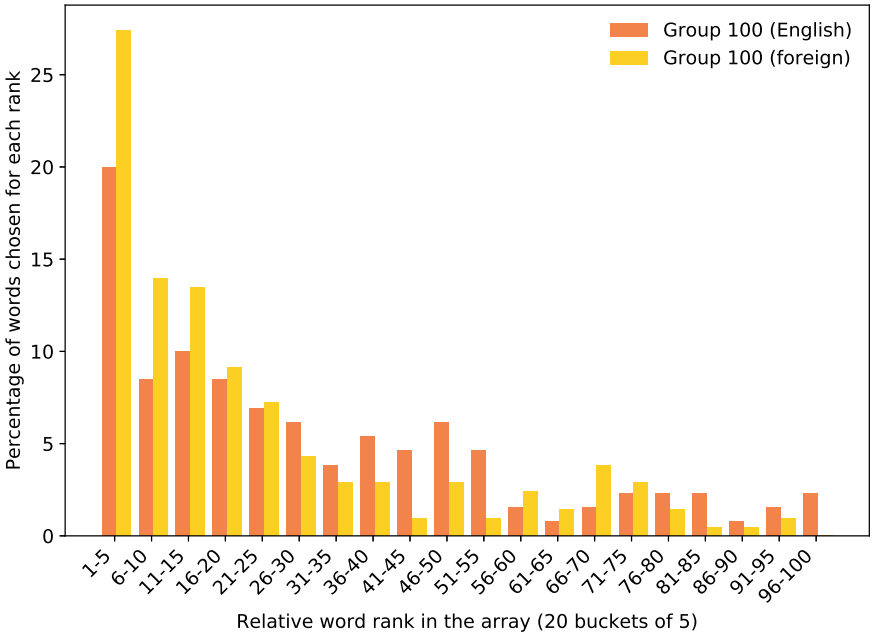

Figure 5.4: Distribution of the words chosen in group "100" as a function of how frequently they are used and whether the participant was a native English speaker. The x-axis corresponds to the word's rank among the other words shown, as sorted by most to least frequent.

Syntactic effects

Using common sentence structures might help memorisation. Some participants apparently tried to make use of this, one of them choosing "Freshman minions cinematically crumble lavender prints". However, most participants did not try to make a grammatical passphrase. Even when limiting the analysis to the four broadest grammatical categories (noun/verb/adjective/adverb), most sentence structures were unique, with 67 grammatical structures seen only once out of 99 passphrases. The grammatical structure in the example passphrase (noun-verb-noun-noun-verb-noun) is among those unique structures. 9 structures were seen twice and 3 were seen thrice. The only relatively common structure, which was present 8 times, corresponds to a sequence of 6 nouns.

Figure 5.5 shows how frequent each grammatical category was in each position in the passphrase. For example, adjectives are less present as a second word than as a first. There is some imprecision as words can fit in multiple categories (e.g. *scars* as a noun or a verb). Nouns seem over-represented, but this is consistent with their frequency in the dictionary ($\approx 60\%$). No correlations were found between successive word categories, which would be difficult in any case because of the multiple potential roles of each word and additional inquiry should be conducted on the subject.

In group "20", 12 people created passphrases that made some amount of semantic sense and followed English syntax, such as the example given. 13 passphrases could make some sense but had unusual or incorrect syntax, such as "furry grills minidesk newsdesk deletes internet". 22 appeared to be six randomly ordered words, such as "wastewater refundable sweatshops misspelling sellout ailment ". In group "100", only 6 people created passphrases that made some amount of sense and were syntactically correct. 15 made passphrases that could make some sense, and 30 had passphrases that seemed randomly ordered.

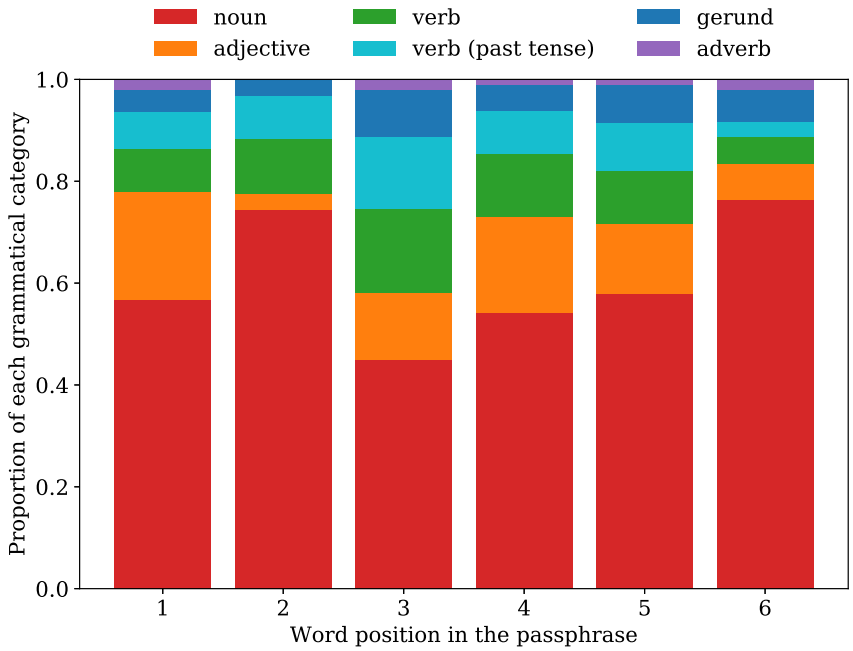

Figure 5.5: Stacked bar chart representing which grammatical role is played by words in each position in the passphrase. The proportions are over all passphrases were the grammatical role of words could be inferred.

Positional effects

The heat maps in Figures 5.6 and 5.7 show how the position of a word in the array shown affected the probability that it would get chosen. The numbers correspond to the percentage of participants who chose the word in that cell, with a deeper red indicating a higher percentage. Numbers don't add up to multiples of 6 as some of the words did not correspond to any word in the array presented to the participant.

From the main heatmaps — Figure 5.6 top and Figure 5.7 left — we can observe three different effects. First, there is a small but significant bias in favour of the lowest lines of the array, as well as top line (the choosing bias is around 11% above average for group "20", with $p < 0.02$, and 26% for group "100", with $p < 0.03$, both analyses being done with ANOVA). There is, however, no discernible horizontal bias. The middle and bottom heatmaps in Figure 5.6 also reveal that, although overall choice is well spread among all lines and columns, the first choice is strongly biased for the first column (with 57% of participants choosing a word from it), which is compensated by the other choices, such as the one for the last word where the top left corner is all but ignored.

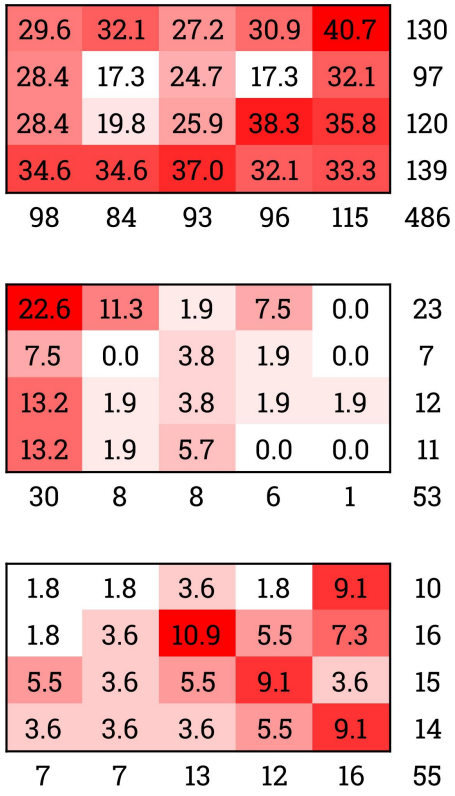

Figure 5.6: Heatmaps indicating the percentage of participants choosing the word in each cell for group "20" and the total number of words chosen in each line/column besides the heatmap. The top heatmap corresponds to the whole group "20", including participants who did not finish the experiment. The central heatmap corresponds to the percentage who chose each cell as their first word, whereas the bottom one corresponds to the percentage who chose each cell as their last word. Both of the latter are restricted to the participants who finished the experiment.

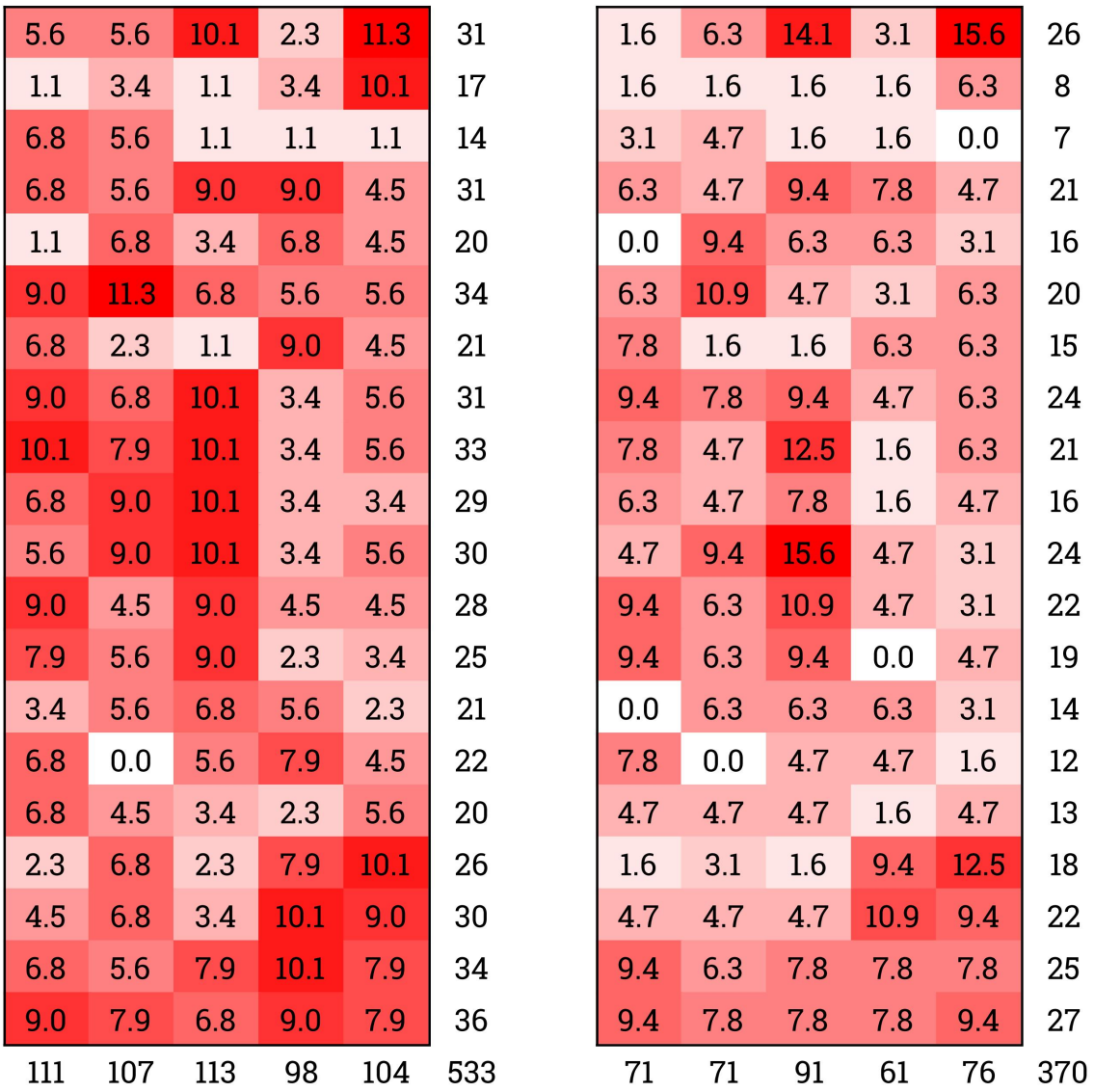

Figure 5.7: Heatmaps indicating the percentage of participants choosing the word in each cell for group "100", and the total number of words chosen in each line/column besides the heatmap. The left heatmap corresponds to the whole group, including participants who did not finish the experiment. The right one is restricted to the participants who finished the experiment and shows the same tendencies — a weak bias in favour of the top and bottom three lines.

5.4.2 Memorisation

After choosing a passphrase and performing a distractor task, participants were asked to recall their original passphrase; 46% of participants recalled all 6 words in their passphrase without any errors. An additional 20% remembered their words but made a typo.

Table 5.1 sums up the recall success rate and types of errors when recalling passphrases. The rates are calculated separately for group "20", group "100", and the control group. They are also split between the first and the second section, in which (except for the control group) participants were reminded of their original array of words. The following mistakes and errors showed up in participant recall of newly chosen passphrases:

- Correct shows the proportion of participants who wrote the passphrase perfectly[c].
- Typos are simple one-letter errors or exchanges between two adjacent letters.
- Variants are like typos in that they are related English words. Most of those were verbs where the participant added or removed an "s" (or less frequently an "ed" or "ing").
- Orders are errors where at least two words are exchanged in the passphrase.
- Misses are words that are entirely missing from the passphrase entered.
- Wrong words are ones that have no relation to any word in the original passphrase.

Section	Correct	Typo	Variant	Order	Miss	Wrong
Control	23% (6/26)	0.42 (11)	0.42 (11)	0.38 (10)	1.19 (31)	0.46 (12)
1:20	40% (19/47)	0.13 (6)	0.17 (8)	0.13 (6)	0.55 (26)	0.11 (5)
1:100	51% (26/51)	0.20 (10)	0.10 (5)	0.06 (3)	0.31 (16)	0.08 (4)
2:20	48% (14/29)	0.03 (1)	0.07 (2)	0.28 (8)	0	0.10 (3)
2:100	58% (15/26)	0.15 (4)	0.08 (2)	0.11 (3)	0.04 (1)	0.15 (4)

Table 5.1: Percentage of fully correct passphrases, followed by the rate at which each error type is committed by each group, as number of expected errors per passphrase. Total number of errors of each type observed is indicated in parenthesis.

Section	Correct	Typo	Variant	Order	Misses	Wrong
Control	40% (6/15)	0.33 (5)	0.27 (4)	0.07 (1)	0.47 (7)	0.13 (2)
1:20	46% (19/41)	0.10 (4)	0.20 (8)	0.05(2)	0.34 (14)	0.12 (5)
1:100	57% (26/45)	0.20 (9)	0.11 (5)	0.02 (1)	0.31 (14)	0.02 (1)

Table 5.2: Same categories as Table 5.1, with data restricted to the participants who had the training exercise correct. Total number of errors of each type observed is indicated in parenthesis.

[c]In the table, there is one more participant per group in the second section than there should be. This is due to one participant in group "100" double-clicking the submit button and getting directly to the second section, and one participant in group "20" writing nonsense in the first section but getting five correct words in the second.

Overall, 94% of the 51 people that were guided with "100" words remembered at least 5 of their words, and 69% remembered all their words but made a small mistake (like getting them in the wrong order). 81% of the 47 people who were shown 20 words remembered at least 5 of their words, and 64% made at most simple mistakes. The control group demonstrated lower performance, with 38% remembering 5 of their words, and 27% making only simple mistakes.

When comparing the number of people who correctly remembered their whole passphrase, group "100" is superior to the control group ($p < 0.02$). This effect is magnified when comparing not the participants but the words directly. The words in passphrases made from the 100 words array were better remembered than those made from the 20 words array ($p < 0.03$), with only 0.4 words forgotten or wrong per participant against 0.7. Similarly, the words in passphrases made from the 20 words array were themselves much better remembered than the ones given to participants in the control group ($p < 10^{-4}$), who had an average of 1.3 words forgotten.

The creation of sentence-like passphrases had no statistically significant impact on the overall success rate in either group, with a small decrease in misses and an increase in false words ($p > 0.05$).

Table 5.1 shows the error analysis restricted to those who typed the passphrase correctly in the first exercise (the one asking them to type the first two letters of each word). The same effects can be seen there — although with smaller magnitudes. This could be interpreted to slightly mitigate the effects mentioned previously, although other factors add some noise (such as typos or variants during the first exercise).

Remarks. The preceding error tables do not take into account four anomalous behaviours. Two participants (one in each group) made a typo in their original passphrase (phrases were only counted as correct when typed without the typo). One participant, when asked for the first two letters for each word in their passphrase, typed random letters on their keyboard, and one typed something that looked like the requested twelve-character string with lots of mistakes. Both of those were in group "20" and were not counted in the analysis. One participant in group "100" also double-clicked on the next button and was taken directly to the second try and shown their array of words. Finally, four participants in the control group showed no attempt to recall their passphrase, responding with random words (and in one case not even filling all text boxes), and were removed from the dataset. Including these would have only strengthened the results that guided choice helps.

Language. People that identified their primary language as English were balanced between the two groups (25 of 51 in group "100" and 26 of 47 in the other). Language did not show a statistically significant effect: 21 out of 51 people who indicated English as their primary language were correct on the first try, as were 23 out of the 48 who indicated another language. Primary English speakers had more misses (29 against 13) and an equal share of wrong words.

Time. No statistically significant advantage was shown for participants who spent more time designing their passphrase. People who recalled their words perfectly appeared to take 5-10% longer on average, but 10-15% less time for the median, showing no clear effect. Some of the participants had also disabled the JavaScript functions needed to record the time taken (these are disabled by default on most Apple iPhones).

5.4.3 Guessing

Most participants only tried to guess 1 other passphrase, but 21 participants tried to guess between 2 and 4 passphrases. Words that were in the original passphrase with a minor modification (such as a typo) were counted as correct in this exercise, like those that were only in the wrong place.

On average, participants succeeded at guessing 0.85 of 6 words chosen by another person from 100 word arrays, and 2.15 of 6 words from 20 word arrays. This is significantly higher than a random guess (which would on average get 0.36 and 1.80 words correct), but not better than (an educated guess) focusing on common words or positions. With access to the array, positional guessing would get respectively 0.63 and 2.21 words correct. Purely semantic guessing based on word frequencies would get respectively 1.40 and 2.93 words correct.

5.5 Statistical modelling

5.5.1 Strategies and entropy

The effect of participant choice on the entropy of passphrases was tested. Models simulated in Python were used to analyse the word choice of participants when presented with arrays of words, with three main strategies:

The *Smallest(n)* strategy corresponded to picking the six most frequent words presented in the array of n words.

The *Uniform(n)* strategy being equivalent to sampling random words uniformly from a dictionary of size n, entropy was computed exactly. Table 5.3 shows the entropy per word for a few strategies. The *87,691* here corresponds to our described dictionary of *87,691* words, and *300,000* to Norvig's more complete dictionary of *300,000* words.

Finally, the *Corpus(n)* strategy corresponded to picking each word from an array of n words according to a distribution where the probability to pick each word w is a function of $f(w)$, its frequency in the language. This corresponds to models for how the distribution of words is biased in many different corpora of texts. The model used here is Zipf's law [HSGMS02], stating that the probability of choosing a word $p(w)$, is inversely proportional to its rank in the frequency list:

$$p(w) \propto \frac{1}{rank(w)}$$

Informally, the 100^{th} most used word is chosen with a frequency about twice the frequency of the 200^{th} most used word.

10^9 simulations were run for both 20-word and 100-word arrays with each strategy to estimate the probabilities — and the entropy[d].

A much bigger sample would be needed to exactly compute the entropies corresponding to user behaviours. However, experimental entropy can be bounded by using distributions for known entropies. As such, the cumulative distribution functions for the experimental groups and models were computed. Although they do not make direct strategic sense, we included *Corpus*(17) and *Corpus*(13) in comparisons and in Figure 5.8, as they bound the observed curve for group "20".

[d] The error bounds due to the simulations are quite smaller than 0.01 bits.

Strategy	Entropy (bits)	Strategy	Entropy
Uniform(87,691)	16.42	*Smallest*(20)	12.55
Corpus(13)	16.25	*Uniform*(5,000)	12.29
Corpus(17)	16.15	*Uniform*(2 000)	10.97
Corpus(20)	16.10	*Smallest*(100)	10.69
Corpus(30)	15.92	*Corpus*(300 000)	8.94
Corpus(100)	15.32	*Corpus*(87 691)	8.20
Uniform(10 000)	13.29		

Table 5.3: Entropy (computed or simulated on 10^9 runs) of 13 potential choosing strategies. The number in parenthesis next to *Smallest* and *Corpus* represent the number of words shown (out of 87 691). *Corpus*(300 000) then represents the expected entropy if people could independently select 6 words using the general language bias — but no dependency between words. People naturally don't pick the 6 words independently in practice.

Experimental values for group "20" are slightly above *Corpus*(17) around the 50,000$^{\text{th}}$ word. An upper bound of *Corpus*(20) could be chosen, but there is a strong argument for using *Corpus*(17). This is more affected by the values of the high p_i, as the function changes less as it gets to the least common words (making it concave). As the p_is shown in Figure 5.8 are also sorted in decreasing order, it means a small bump in word choice in the first part of the curve is more than compensated by the lack of a bump in the second part. As such, it is reasonable to infer that the entropy corresponding to participants' behaviours in this group is between *Corpus*(13) and *Corpus*(17).

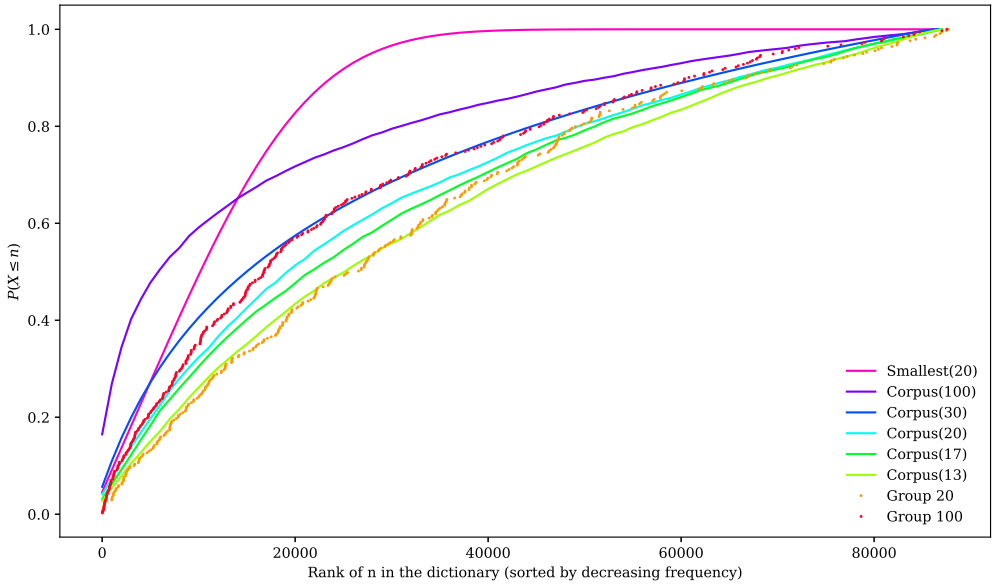

Figure 5.8: Cumulative distribution function showing the proportion of words chosen by participants among the n most frequent words in the dictionary, as a function of n. The x-axis represents the ranks words in the dictionary, sorted by decreasing frequency, and the y-axis which proportion of words chosen by participants were of lower rank than that, for both experimental groups and 6 different models.

A slightly tighter fit can be obtained by taking not the simplest Zipf's formula but the more general one, with, for $\beta > 1$:

$$p(w) \propto \frac{1}{(rank(w))^{\beta}}$$

In such a case, setting $\beta = 1.35$ makes *Corpus-Zipf*(13) a tighter fit than previous curves, giving an entropy of 16.19 bits per word[e]. However, the presence of noise in the data means that a search for a more accurate model would be premature.

This analysis shows that six-word passphrases generated with the method proposed and a 100-word array have at least 95 bits of entropy, and ones created with a 20-word array have nearly 97 bits of entropy. As the difference in entropy is very small when compared to the effect on memory, showing a 100-word array is generally the best method.

5.5.2 Semantic aspects

This section has focused on the frequency of words as a proxy for their familiarity. This ignores other possibilities such as emotional attachment to certain words, linked to the particularities of each user. For example, a dog owner might choose the word dog if it appeared in the array. We can show here that the magnitude of such an effect should be quite low. Let's suppose that each user has a set of 100 words that they will automatically choose whenever they appear. They have the opportunity of choosing such a word with probability at most $1 - (1 - \frac{100}{87685})^{100} \approx 11\%$.

An adversary with the word set could, for each position, try this set of words and fill the rest with a dictionary[f], lowering the number of possibilities to test. Assuming there is one such word — which happens with probability 0.11 — the remaining entropy goes from 95 bits to 89 bits.

Moreover, many emotionally-loaded words such as "dog", "love" or "president" are already among the most frequent words (the ones given are all in the top 1,000). As such, people might choose them with a high probability, no matter their individual preferences, so the marginal information — and the corresponding entropy loss — should be even lower than the bound already given.

5.6 Limitations

5.6.1 Ecological validity

As this was an online study done in the wild, it did not happen in a controlled laboratory environment, and anomalous participant behaviour could not be detected. The main problem with this design might come from participants writing down their passphrases somewhere, affecting the memorability results. Two arguments help us believe that this had little to no impact on the results. The first comes from the fact that writing down one's passphrase should take a certain amount of time, and there was no demonstrated correlation between the speed during the creation and the ability to remember one's passphrase

[e] One could also use a Zipf-Mandelbrot model [Old68], but the additional parameter would be hard to validate accurately without having a sample of at least 10,000 participants

[f] Getting a single word from the set is already a low-probability event, and getting more than one happens with probability at most $2 * 10^{-4}$.

correctly. The second comes from [YLC+16], where participants in the study were encouraged to create passphrases and use any technique they generally used to remember passwords and passphrases. Despite knowing in advance that they would have to remember the passphrase for a week, over 80% of participants reported not having written down their passphrase.

5.6.2 Short-term and long-term memory

As the participants were not asked any identifying information, it was not possible to ask them to return to the experiment, to estimate the effects of the method on long-term memorability of passphrases. The main reason we added the distractor task in the experimental protocol was to disrupt participants' short-term memory to look at long-term memory effects. We compared recall rates with the long-term recall rates for the variety of passphrase creation strategies shown in [YLC+16]. The after-distractor memorability observed here is closer to the long-term memorability they observed. This is consistent with the fact that no variation in recall rates between strategies was observed in their short-term experiment, unlike their long-term experiment and our data. Moreover, as the participants had no intrinsic or financial motivation to recall their passwords, their recall rates should be lower than in real world usage.

5.6.3 Free choice of words

This study did not include a fourth group who would be free to choose words in any way they wanted. This could have been interesting in the goal of comparing memorability, but two factors motivated the absence of this second sort of control group. The first is that we could not ensure that people would pick new sentences and not ones they were already trained on, which would skew the results. More importantly, we believe that the behaviours observed in [YLC+16] indicate that free word choice is too much of a security concern and not a viable option.

5.7 Discussion

The above results demonstrate that passphrases created by choosing words from an array of random words are more memorable than automatically generated ones. While past studies have shown that choosing familiar words for passphrases led to huge entropy reductions, our technique obviated this. The entropy cost due to the choice in our system is negligible, staying between 1% and 3% depending on array size. Since people can choose words they know, larger dictionaries can be used. Compared to random generation within a dictionary of 10 000 words, the final entropy by word using this method is superior by $20-30\%$. The long-term memory performance can only be intuited at, but the magnitude of the effect, when compared with alternatives on short-term memory with a distractor task, should be a good indicator. The reduction in entropy when compared with random passphrase generation is offset by the improved usability and memorability that come with choice.

Biases. Multiple surprising behaviours were observed, confirming certain hypotheses and refuting others. The tendency to choose familiar words was observed although this bias

was weaker than in the English language (as predicted by Zipf's law). The participants' choices were also influenced by the positions of the words in the arrays presented to them, but the positional biases mostly compensated each other over the whole passphrase. The main effect we detected was a small but significant bias in favour of the last lines, and a smaller one for the top line, with no significant horizontal effect.

Both types of bias interacted, and this could be why *Corpus*(13-17) might be better fits than *Corpus*(20) for group "20". This should not be seen as a weakness of the system but as another source of its security: as the position is truly random, a stronger positional bias reduces the number of potential words the participant can choose from, leading to a more uniform — and higher-entropy — passphrase. This is true as long as the adversary cannot get the initial word array, but in such a case the system is already insecure. Similar positional effects could partially explain why the distribution is closer to *Corpus*(30) than *Corpus*(100) in group "100".

Mnemonics One might expect participants to use mnemonics and create sentence-like passphrases that used common patterns, but the only syntactic pattern that appeared more than thrice was 8% of participants choosing three nouns in a row. As nouns form the bigger share of the dictionary used, even those passphrases are secure, reducing the entropy by at most 5 bits (out of more than 96 bits of entropy). Having more words to choose from did not increase the tendency to create syntactically correct sentences but reduced it instead. Word choice patterns held across the range of proficiencies in English. Those results are true not just for primary English speakers but also for people for whom it is a second language, with small differences in word choice and no significant difference in memorability.

Strategies. The task where participants were supposed to guess each other's words had a purpose beyond distracting and impairing their memory. We were hoping to find whether a simple strategy could explain the participants' choices, in which case some would get very good results. The absence of such successful participants shows that if a general strategy to explain participants' choices exists, it has eluded both us and them.

Using it in practice. This method should be used carefully: an adversary getting their hands over the list of words would be dangerous, even if the list has 100 words. The word array should only be generated when needed, locally on the user's machine instead of it being sent over any network. However, this needn't be a problem in practice as the secret array to select from can be produced on any machine, the dictionary itself and the code needed taking less than 300KB of memory. Although it could be of help if the user forgets their passphrase, storage of the array itself on a local machine once used would introduce real vulnerabilities.

Naturally, helping people despite their own will is a hard task. As such, this method should only be used at a user's initiative — being a tool to help them make better passphrases — to avoid the same pitfalls as counter-productive password constraints. The goal here was then to provide a viable alternative to make secure passphrases to the users who want them and who are warned of the dangers of manual passphrase creation — warning them being an altogether different issue.

5.8 Open questions

This work demonstrates that large improvements can be achieved in passphrase usability while increasing their entropy, but is only a start. Below are a few suggestions for usability/security questions left to test for the guided passphrase scenario:

- How well does the short-term memorability with a distractor task predict long-term memorability for passphrases?
- Can putting high-frequency words closer to the middle of the array compensate for the weak positional advantage?
- We did not compare other types of visual presentations, such as a single column or a word cloud. Could using those make word choice from the presented set even more uniform? Would making it more uniform even be a good idea?
- Would the high uniformity of word frequency be as or even more successful with even larger dictionaries, for example with the full SOWPODS and its 276663 words?
- Why is choosing from 100 words more memorable than choosing from 20? It isn't because people took longer, as they didn't. It might be as it gives users more personalised choices or the ability to select more familiar words. It might also be for some not yet explored reason. Does memorability continue to increase with arrays of more than 100 words? What is the nature of the trade-off, and must there be a compromise between entropy and memorability?
- How does the size of the array affect reading patterns and word choice? Does the left-wise bias in group "20" come from different reading patterns?
- Are there better ways to make highly-memorable high-entropy passphrases?

This empirical study gave us an improvement that can be used as a primitive for algorithms like Cue-Pin-Select, as well as general-purpose passphrases. It is now time to come back to the problem of mental password managers with a more general focus in mind: how do we compare and analyse mental algorithms efficiently?

6

Towards a Cost Model for Mental Computations

6.1 Previous work and contributions

Chapter 4 introduced algorithms for mental computation of passwords but did not go into the details of what they are based on. Mental computations, once a mainstay of many educative curricula, have progressively been set aside thanks to the universal presence of devices able to perform them more easily, safely and efficiently.

In a series of papers partially overlapping their work on passwords, Blocki, Blum, Datta and Vempala have introduced not only a series of mental algorithms meant to replace password managers, but also general models for mental computing [BBD13, BBDV17, BV15, BV17]. This is a formalisation of what was already happening in a limited fashion [Lee14], and has since then been extended further with a focus on usability as in Yang's work and Chapter 4 [YLC+16].

One important problem hinders the development of new mental algorithms: the high cost linked to testing one. This is due to the fact each new algorithm (and every variant) requires an independent usability study to ascertain its performances when compared to other solutions. To remedy this, the ideal would be having a cost model for mental algorithms, which, for any algorithm, would give the distribution of how time-intensive it is for a normal human population. There are, however, quite a few hurdles to overcome, as humans are highly non-trivial computing machines, and for any cost function C:

- We can't assume that the $C(A;B) = C(B;A) = C(A)+C(B)$ for the concatenations of two sub-algorithms A and B, as task switching might create new non-symmetric costs.
- We can't assume that $C(A;A) = 2 \times C(A)$ for an algorithm A, due to both habituation and mental fatigue.
- C can be highly dependent on the human considered. This is why, assuming there are clusters of humans with similar performance on similar problems, a distribution of time (instead of an expected time) might be obtainable and more informative, although it would require much more work.

This chapter is based on research done with Ted Selker and Florentin Waligorski.

Cognitive science approaches. Some of this has been studied extensively with approaches from both psychometry and neuroscience. For example, some studies — dating back to the 1960s — have focused on our memory abilities for a variety of stimuli (such as words, sentences, pictures) [NS67], not only in general but also in conjunction with mnemonics [Bel87], or in comparison with other primates — who can have better numerical memory than us [IM07]. Relevant to the previous chapter, another study also looked at the advantages enjoyed by high-frequency words when it comes to short-term memory [HRS+97], with one experiment looking at the differences between recalling words in the same order they were shown or in reverse order and showing that word frequency only affected forward recall.

Another focus of study has been the arithmetical capabilities of the average human. This went from establishing the source of our numerical abilities (which depend partially on linguistic abilities depending on the magnitude of the numbers considered) [Deh92] in the 1990s to figuring out a neurological basis for our representation of numbers [Ash92, But99, BZGJ01]. Correlations between arithmetic, algorithmic, and linguistic skill were also established [FFC+06].

Differences between demographic groups have been investigated, especially when it comes to arithmetic abilities. This has been done to compare performances with people with math anxiety [CSM+17], with short-term memory problems [BCW96], as well as across cultures [RZT+15] and age groups [But02].

In parallel with their increasing importance in computing, research into different modes of computation such as complex probabilstic models of mental reasoning [JL10, CTY06, Leb99] and, in a different direction, using tasks distributed over groups of human subjects behaving as a single entity to investigate iterative versus parallel processes in social computing [LCGM10]. On the other side, the use of cognitive science and neural models for computing purposes have been well studied since at least the 1980s [And83], with variable success [Bis09, Tay09].

Blum's model. Despite this extensive amount of work, very little was done in the search for a cost model for human computing that would accurately predict, quantitatively, how hard and time-intensive different mental algorithms are. So far, the main model used in security was introduced by Blum and Vempala in 2015 [BV15] — and slightly refined afterwards [BV17]. Here is the full cost model, as presented in the original paper:

- Working memory for letters and digits is a stack of two elements (letters or digits). The cost of accessing the top (most recent) entry is 1, and that of accessing the bottom entry is 2.
- Retrieving a sequence from recently retrieved long-term memory, i.e., retrieving a pointer to the beginning of the sequence, has Cost=1.
- Following a pointer into a recently-retrieved sequence in long-term memory, moving it 1 step to the right, resetting the pointer to start/start+1 or to end/end-1 has Cost=1.
- Operations of =, + and x (mod 2,3,4,5,9,10) on two single-digit numbers. Cost = number of digits created (not necessarily all written) during the operation. Example: 4+3 (mod 10) = 7 has cost 1; 4+9 (mod 10) = 3 has cost 2.

- Operations of =, + and x (mod 11) on two single-digit numbers has Cost = number of digits created (not necessarily written) before applying the mod operation + 1 for the mod operation on numbers > 11. Example: 4+3 (mod 11) has cost 1; 4+9 (mod 11) has cost 3.
- Operations of =, + and x (mod 2,3,4,5,9) on two single-digit numbers are treated similarly.
- Applying a map such as a character-to-digit map is the same as following a pointer and has Cost =1.

The authors mentioned that this model is meant to give a rough expectation of time taken and needs to be experimentally validated. They also insisted on the fact that the total cost for a mental algorithm should be kept low, giving a maximum cost of 12 as a recommendation — which is reasonable assuming that an average individual spends 2 to 3 seconds on a task of cost 1.

We were surprised by the assumption that getting a digit from a character-to-digit map had uniform cost, and decided to run this experiment to validate — or not — the original model. As a mind experiment, maybe these costs seem reasonable, but it seemed artificially simple, with no place for queries of variable complexity (such as getting the n-th letter in the alphabet, which intuitively seemed easier for $n = 1$ than for $n = 13$). We show here the preliminary results of a pilot study on 81 participants that sought to shine a light on the main mechanisms involved in creating a cost function for mental algorithms. Among other preliminary results, we show that, instead of having uniform cost, there is strong evidence that accessing a mental map might have very skewed performance, with a difference of one order of magnitude depending on the element accessed.

It has been established in a thought-provoking paper [Bor06] — that came out before the replication crisis — that many of the practices used until recently in psychology and psychometry suffer from mathematical defects. This concerned mostly incorrect usage of statistics and confusion between exploratory and confirmatory studies. To be clear and avoid making similar mistakes, this study is mostly exploratory, and although it establishes — or refutes — some of the assertions explicitly or implicitly made by the previous model, its main goal was to show the need for a more rigorous and empirically tested model.

6.2 Design of the experiment

The goal of this pilot study is to give preliminary answers to some of the following questions and offer insight as to how to design a larger-scale investigation to study the others. The main goals were the following:

- Get baseline time cost distributions for the elementary operations considered. This is a frequent subject in psychometrics and has already been well studied, although in a different context, and generally not in a holistic way.
- Check whether accessing the n-th element in an array is indeed done in constant time or not, as asserted by Blocki et al [BBD13]. Check the effects of repeated accesses.
- Check whether the cost commutes: is it easier to perform an addition after a multiplication or the other way around? Check also for alternating linguistic and arithmetic tasks.

- How independent are users' performances in elementary linguistic and arithmetic tasks? This is especially relevant for password algorithms as the existence of clusters with different abilities could motivate the need for a family of algorithms, each one tailored to a group of people.

Moreover, we were also interested in the effects of mental fatigue, priming, and memory. Ideally, the data from this pilot experiment could inform a second experiment with the eventual goal of creating a model that could give cost distributions for increasingly complex mental algorithms.

6.2.1 Demographics

As the country where the experiment was organised does not have institutional review boards, local regulations and best practices were followed instead. No protected demographics were targeted, all the participants were informed of their rights and could quit at any point they wanted, and their data was only collected if they confirmed at the end. Besides age and main language spoken, no personal or identifying information was collected.

Participants were recruited through a mix of snowball sampling on social networks [HG11], referral through a website that indexes psychological experiments [Kra19], and through the first author's professional website. As a high number of participants were expected to be French, two versions of the experiment were developed and participants could choose their preferred version on the first page. To analyse and compensate for bias in the recruitment, different links were used to identify the recruitment method of each user.

81 people finished the experiment, with data from two being discarded as they skipped most of the questions. 45 were native English speakers, 26 were French, and the rest spoke 7 different languages. Ages went from 16 to 63 with a median at 26.

6.2.2 Protocol

The participants interacted with the system through a web-page that recorded their final answers to each question, as well as the time taken to type a character for the first time — for each question — and the total time before submitting the answer. When looking at speeds for simple tasks where the answer is a number or a word, the measures and comparisons are all made on the basis of the first character typed — to compensate for words of different lengths among other things. To attenuate the delay due to reading the question, complex instructions were shown as an example before the real question was asked.

The experiment had 9 sections of varying length:

1. Participants were initially told their rights and asked their age, the main language they used, whether they were on a mobile device, and whether they had a number keypad.

2. They were told to learn a four-word sentence. In a between-groups approach, four different sentences of variable complexity were used in each language[a]. They were

[a]The English sentences were "Fortune favors the brave", "Colorless ideas sleep furiously", "They are extremely childish", and "Caravans passing along quietly".

immediately asked to type the sentence. After the end of sections 3, 4, and 5, they were asked to type it again and were shown their sentence if they had made a mistake.

3. They were given five different arithmetic exercises: first a set of three single-digit additions, then a set of three single-digit multiplication, and three double-digit additions. Then a set of two multiplication on lower double-digit numbers, and finally four remainders, each time a double-digit number divided by 3,5,9 and 11.

4. They had to write a series of three words starting by t, b and a (three different questions), like tortoise, blueberries and Austria. In a between-groups approach, half were asked for an animal, fruit and country, while the other half were asked to type in any word starting by the same letter.

5. They had to give the n-th letter in the alphabet, for four different letters in a row. They were then asked to say, for a letter, what position it had in the alphabet, again four times in a row. All the groups but one had numbers written with Arabic numerals while the last one had the numbers spelled out. After this, they were successively given three words and asked to write the letters in alphabetical order. Finally, they had to say what was the n-th month of the year (for two different months), followed by two reciprocal questions where they had to give the number corresponding to a month.

6. Participants were shown a sentence on the screen and had to find a given letter and write down the next two letters. This was done three times, each with a different sentence corresponding to the three other sentences not given in section 2. In a second time, they had to do the same exercise twic,e but with the sentence they had memorised and without help from the screen.

7. They had to type the words they had chosen in section 4.

8. Participants were given three different algorithms. The first had four arithmetic operations, the second text-based tasks (finding letters in a sentence), and the third a mix.

9. Finally, participants were asked whether they had used a pen and paper, and for which tasks.

Between-group design was used in multiple ways in this experiment. One was that sections 3, 4 and 5 were given in a random order. For most of the questions, participants were divided into 3 to 5 groups with different but structurally identical questions given to each group.

For sections 3 and 5, one group was given random numbers or letters, while the others were given predetermined questions. This was to ensure that enough data would be collected to establish statistically significant results on certain properties, while maintaining the possibility that, if enough people participated, there would be general data available on more than few specific points. For example, in section 3, a quarter of the participants started with the question "2+2=?", another quarter with "9+8=?", and the rest with random pairs of digits.

6.3 Preliminary results

One common problem in all the measurements that follow is that they have limited ecological validity, as the participants had to read the instructions and assimilate them before performing the trials, adding a measure of uncertainty — although instructions were minimal, as a more thorough explanation was given before introducing a new type of task. However, this only strengthens the announced results — as making the data fuzzy is not enough to make the effect disappear. As the experiment was not in a controlled environment, we couldn't prevent people from taking a break (and some participants did take to a 10 min break). On the large datasets, we simply trimmed the 5% highest and lowest values, while on small datasets we decided to conservatively discard only the most obvious anomalies, typically when the person taking the longest time is more than three times slower than the second slowest. The p-values generally correspond to Student's t-tests. No significant effects were observed between the different language groups.

6.3.1 Access time in a list

The first question that interested us was the veracity of the constant access time in a list assumption in the original model. Two different tests were focused on this. The first was getting the position of a letter in the alphabet and reciprocally, and the second concerned the months of the year.

Our expectation was that the first few elements of the list might indeed get accessed in constant time, with a sub-linear increase until close to the end of the list, where it would go back down as people chose to go backwards instead. This was partially shown to be true:

- On the first question, there was no significant difference between asking for the letter "A" or the letter "B", with both average and median times between 2.6s and 2.9s. However, "D" is quite slower than "A" (3.5s on average), with $p < 0.03$.

- On the second question, there were no statistically significant differences between "K", "M" and "N" (respectively 8.7, 7.1 and 8.3 seconds on average and 7.2, 7.2 and 8.2s in median).

- The strongest result came from the third question. All the users who were asked to give the 13th letter ("M") had previously had to give the 14th ("N") in the third question, and did so in 2.1s on average. The ones who were previously asked to give the 14th letter were now asked for the 13th, and did so in 3.9s on average, showing that getting the next element in the list is much easier than the previous one ($p < 0.0005$). The speed ratio is already close to a factor 2, but is probably at least a factor 3, assuming a 1s reaction delay (reasonable as only 1% of answers were faster than 1s).

- Still on the third question, the participants who had had the 11th letter ("K") and had to give the 18th ("R") did not seem to benefit from their previous computations and took an average of 14.9s. This is confirmed by the fact that getting the 22nd and 23rd letters in the last questions also took 14.3 and 14.7 seconds, with no visible effect from what participants had to answer just before.

- Although the sample size is too small to be sure, it seems that there is indeed a speed increase for the last 3 letters of the alphabet, with an average of 9.0s mostly due to 2 out of 8 participants who took more than 8.6s ($p < 0.03$).

Similar results can be shown for character-to-number maps. For example, giving the number corresponding to "D" took an average of 4.8s, as opposed to 3.7s for "A" on the first question, and 7.2s for "K" on the second question ($p < 0.001$ and $p < 10^{-4}$ respectively).

It seems that, at least when it comes to mapping numbers and characters, people poorly re-use already computed values and access them sequentially, with cost factor variation upwards of 5 depending on which value is accessed. However, it might have limited ecological validity in Blum's model as, although we all have extensive training with the alphabet, most of the training corresponds to accessing it sequentially. This is where the exercise on the months comes into play, as we are generally more used to accessing the values directly:

- When initially asked for either the first or the fifth month of the year, participants were much slower in the latter case (2.8s and 4.3s on average, with medians of 2.4s and 3.8s, $p < 0.005$).

- In the second question, after asking for the fifth month in the first question, there was no statistical difference between asking for the next or previous month. When asking for the tenth month, there was no statistical difference between the groups who had been asked for the first or the fifth previously. However, there was a large difference between the group that was asked for the tenth month (3.1s on average, median of 2.7s) and the fourth or sixth months (2.1s on average, median of 1.7s).

With a cost difference of at least a factor 3 and up to 10 depending on the map considered and the value taken as a delay, this is enough to show the limits of the original model: an algorithm with a promised cost of 2 being doable (for the same person) in either 4s or 30s depending on the values involved.

6.3.2 Arithmetic operations

With the data from the arithmetic section, we find multiple relevant effects:

- There was a large speed increase between the first and second question — both being single-digit additions — which does not continue for the third question or afterwards (going from 3.1s to 2-2.1s, with the medians going from 2.6s to 1.6s). This is not surprising, but it reduces the validity of data collected from the first question, forcing us to discard it for the following analysis.

- The time distributions for 2+2 and 4+5 were very similar, but there was a difference between 4+5 and 9+8, with the average time going from 1.9s to 2.6s ($p < 0.005$), with a small difference in median time (1.6s to 1.7s) but a much bigger spread in the second distribution.

- The same effect can be seen with multiplication: results with a single digit were computed faster (average of 1.7s) than ones with double digits (average of 4.7s, with $p < 10^{-5}$).

- There was a large minority (38 times out of 144) who did not seem affected by the result for the multiplication, computing double-digit results faster than the median for single-digit, maybe due to remembering multiplication tables.

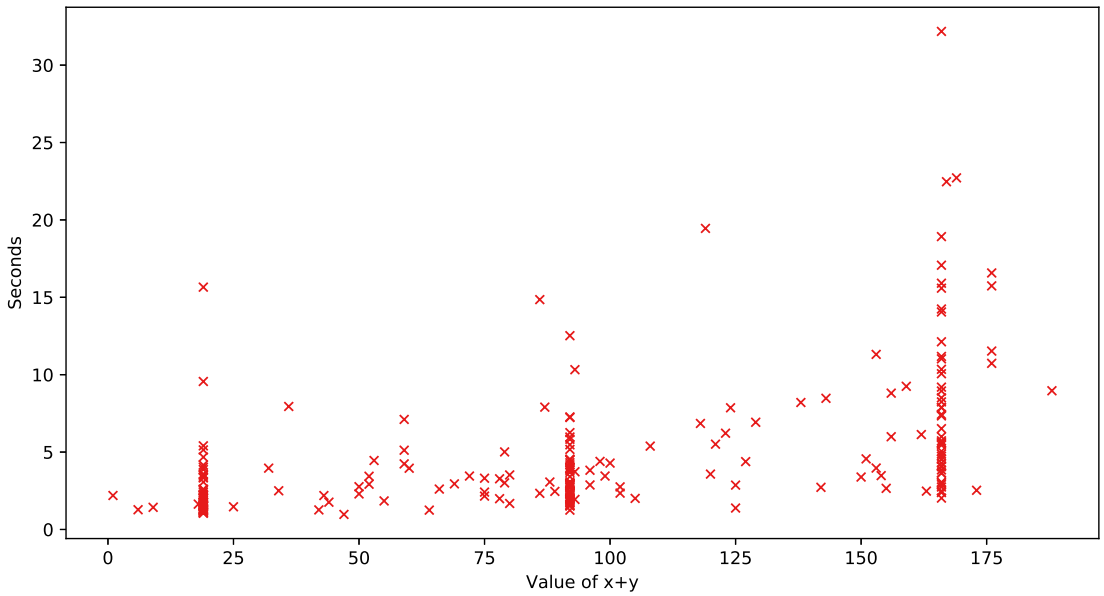

Figure 6.1: Time taken by participants to compute "x+y" as a function of the expected value. Two-thirds of the participants were given the tasks of computing "12+7", "79+87" and "87+5" in various orders, with the rest having random 2-digit additions. It seems that there is a global tendency to increase linearly, unlike the logarithmic growth announced in the original model.

The previous results could be seen as a partial confirmation of the model, but detailed analysis and extension of the addition to 2-digit numbers show that the cost in time is more probably quasi-linearly dependent on the result, with the spread increasing with the complexity, as shown in Figure 6.1. Interestingly, for numbers between 100 and 200 (and especially for the ones around 150), the time seemed to only depend on the result, with no difference on whether it was obtained through multiplication or addition.

One surprising effect was observed in section 8. The participants had to enter the result of the following algorithms:

1. "Start with 2. Multiply by 2. Add 4. Divide by 4."
2. "Start with 2. Add 2. Add 4. Divide by 4."
3. "Start with 2. Multiply by 2. Add 4. Subtract 6."
4. "Start with 20. Add 20. Add 40. Divide by 4."
5. "Start with 6. Add 6. Add 12. Divide by 4."

Although the second might seem easier than the first (addition instead of multiplication, and no change of operations), it was not the case. The first was faster on average (6.6s) than the second (8.2s, $p < 0.05$), in turn faster than the fifth (12.4s, $p < 0.05$). The third was not distinguishable from the second, and the third had a huge spread which made comparisons impossible. Thus, although there is no statistically significant difference between "2+2" and "2*2" and only a minor time difference (0.3s) in the 3rd section, the effect was much bigger when the operations were part of an algorithm.

6.4 Discussion and future work

To be fair, the model developed by the authors of the original series of papers assumed that people would have a training period, although one limited to a few dozen minutes. This means that two possibilities arise: either they thought that this training could compensate decades of semi-regular use, which is improbable, or the "time to access" cost only concerns new maps/functions that people haven't practised before. In such a case, it would stand to reason that most people would get better performance on the functions on which they are trained since childhood, and one needs to separate the costs depending on the kind of function. In both cases, the model requires changes.

This work is obviously quite preliminary as data collection proved difficult initially, and the analysis is on data collected until the 14th of February. There are some obvious things left to do, such as measuring the time cost of memory recall (sections 4 and 7), letter manipulation (sections 5 and 6), letter searching in a text (sections 6 and 8), and the additional cost linked to alternating between linguistic and arithmetic tasks (section 8). Also planned is the measurement of effects stemming from the recruitment method, age, language, or presentation of the question (for example, whether the numbers are spelled out). Finally, we also plan to try finding how much people are clustered by ability.

The end goal of this pilot experiment was to get good enough preliminary data (which is still being collected) to design a rigorous secondary study which could lead to the first empirically-based model to compute the cost of mental algorithms. This model should not only include time costs (although they are the easiest to measure), but also perceived cost, linked to the complexity and repetitiveness of the tasks involved. Moreover, just as pre-WW2 cryptosystems relied on a mix of manual labour and precomputed code-books [Rij18], it would be interesting to investigate the impacts of tool-aided mental computing.

As we've seen, a lack of user studies and poor communication between fields — in this case, security and psychometrics — can lead to a waste of good work and models being used while they are not adapted for most users. This brings us to the third main part of this thesis, focused on voting systems, where a similar disconnect can be observed between the people developing systems and the people deploying them or using them. We initially observed some issues we deployed a verifiable voting system and had it used by non-expert users in late 2016. Problems with ballot usability and code entry served as the initial motivation for Chapter 2, leading us to the research presented in the past few chapters, before we could get back to voting issues. The next chapter will introduce multiple voting systems, and focus on the socio-technical side of election technology, including the difficulties in deploying voting systems in practice. The last two chapters will then focus on different low technology — generally paper-based — voting protocols.

7

21st-Century Voting and its Issues

7.1 Main concepts and contributions

Voting, whether it is on a proposal in parliament or to elect politicians, has been a driver of innovation for more than a century, from Edison's invention of the first electrical voting system in 1868 [JH06] to the recent blockchain-based voting system proposals [Bou16a, MG17, WSH+18]. Correspondingly, voter resistance to the technological changes has followed, starting with the 40-year delay in implementing the secret ballot in the USA, after its successful introduction in Australia — from which stems the name "Australian ballot". Some countries, like France, have voting done in nearly the same fashion today as they did a century ago (1913 was the last major change in France, with the simultaneous adoption of the secret ballot and the voting booth). This resistance to change has come first from elected officials wanting to keep the ability to influence and coerce, sometimes under the guise of defending the "*manly pride that scorns concealment, and the sturdy will that refuses to bend to coercion*" [McK01]. Many costly or complex systems were created specifically for dealing with votes within a parliament, offering a higher level of secrecy against the higher usability of the frequently used system of voting by raising one's hand [JH06]. This proposed secrecy has been the source of arguments from both citizens and party leadership, generally aimed at keeping an elected official beholden to their promises [Gia15], as secrecy can both ruin transparency of a representative and create the possibility for coercion.

A central focus of research and debate on political reform at the turn of the century has been the use of audits and the technological tools to make them easier. Errors with counting and re-counting ballots are well-publicised, leading to a slew of systems that produce both a mechanised or electronic tally and an auditable paper ballot, from lever machines to optical scan methods [BHI02]. Some of the improvements proposed come in the form of small modifications to the voting process to make voting or auditing easier, such as secret-ballot receipts [Cha04], Scantegrity [CCC+08, CEC+08] — an end-to-end independent verification system that coexists with a normal ballot — or audio audit trails [SC05], which seeks to improve the usability of auditing. Others require changing the whole infrastructure

This chapter is partially based on research done with Ted Selker.

by using electronic-only systems [HR17, JL97, Hus08], sometimes not even requiring polling places but instead some forms of e-identification [SFD+14, Vin15]. Before we can look at them in details, we need to define a few often-used terms.

Central concepts for verifiability The main improvements in voting made over the past two decades have been around the concepts of *integrity*, *secrecy* and *verifiability*. There are many flavors of those, depending on what part of the voting system is considered. Using some of the terminology from [RBH+09], here are the three main components of integrity:

- *Cast as intended*, meaning that the vote cast corresponds to the desire of the voter. This can be harder than it seems because of usability issues: for example, in the California gubernatorial election, about 0.4% of the votes meant for the top candidates instead went to candidates immediately above or below on the list due to user error [Sle03].
- *Recorded as cast*, meaning that the vote data (either as a physical ballot or as electronic data) corresponds to what was cast. This can address issues such as hacked voting machines that change the data once the person has voted.
- *Counted as recorded*, meaning that the tally indeed corresponds to the total of what was recorded, without discounting or adding any votes, unlike the 2011 Russian elections where up to 14 million ballots were added to the tally [Har16].

Then, we have *verifiability*, which is central to the integrity guarantees. It can correspond to the following:

- *Individual verifiability*, where any voter can confirm that their vote has been recorded correctly.
- *Public verifiability*, where the public — and anyone in it — can verify that the tally is correctly computed from the recorded votes.

The integrity and verifiability constraints can be very easy to satisfy if we forgot *privacy*. For example, voting by raising one's hand satisfies all the properties above, as anyone can count all the votes. The main issue is that, as people can see the votes of everyone, coercion and vote-buying both become too easy. As such, we need some form of *secrecy*, such that no-one can guess what someone else voted for. There is a good way to formalise this in [RBH+09], stating that if there are two voters A_1 and A_2, and two votes v_1 and v_2, it should be impossible to distinguish between the case where A_1 votes v_1 and A_2 votes v_2, and the reciprocal case.

Most democracies have adopted secret ballot schemes for most votes[a], which gives a limited form of *secrecy*[b]. However, they come at the cost of some integrity, generally concerning *recorded as cast* or *counted as recorded*, as the voter has no idea what happens to their ballot once cast. *End-to-end verifiability* is the ideal property where every step of the process can be checked, either by the public (and auditors), by individual voters, or by a mix.

[a] The only common exception is that elected officials within parliament are often voting openly and with no secrecy, making them accountable to both electors and lobbyists.

[b] This secrecy is not perfect, as can attest the phenomenon of *ballot selfies*, where the voters film themselves in the voting booth while voting, which is held to be legal in certain states [Hor15].

In the past decade, work has generally focused on trying to make end-to-end verifiable voting systems that preserve secrecy. All the systems mentioned try to improve accuracy, integrity, and prevent coercion, miscounting, ballot box stuffing and related fraud — or at least claim to until tests reveal otherwise [SFD+14, Auf19] — generally through technologically complex means. While those were major problems up until the middle of the 20th century [Mil95], their scale is nowadays dwarfed by other considerations[c]. First, manipulation of voter registration lists[d] [Cİ13], accessibility of voting[e] [SGA+14] and turnout buying [LMQ14] can be orders of magnitude above the previously mentioned problems [Nic08, Lev07, BHI02]. Second, familiarity with the voting system is essential[f], and technological changes without adequate training generally come with a strong temporary increase in error rates [HPT+10, HS03, Sel04].

Still, despite the worsening situation in some major democracies and the diminishing trust in the very concept of voting[g], some countries are experimenting with new systems, and better alternatives should be made available to them. Let us then introduce three major candidates, after a summary of the contributions.

Contributions. This chapter makes two distinct contributions. The first is an analysis of some of the factors impeding the deployment of new voting technologies (in Section 7.3). The second is a report on two real-life tests of one voting technology (Random Sample Voting), both organised by the author. The report includes the feedback received from the first test and how this informed the modifications made for the second test, especially regarding ballot design and usability.

7.2 The diversity of verifiable voting systems

Most verifiable voting systems generally trace their origins to David Chaum's 2004 paper on secret-ballot receipts [Cha04], and on the technique he previously developed on anonymising mixnets [Cha81], where a message is encrypted and decrypted successively by a network of servers such that authenticity can be guaranteed without compromising anonymity. System design has improved since the original concept, although many designs

[c] This is mostly true in western democracies where the error rate is generally at least one order of magnitude lower than the margin of victory [EG14], but they are still extant in many countries such as Honduras [GOKdJN15], Albania [DR12], or Russia. The latter, which has recently been accused of hacking recent American elections [Ohl16], has also seen its share of domestic election manipulation through a mix of *coercion*, *turnout buying*, *vote buying*, and *ballot stuffing* [Har16, FRS18].

[d] The most striking recent example of this concerns the 21 million women mysteriously missing from Indian electoral rolls prior to the 2019 general election [Bis19]

[e] Accessibility can be a problem in many different ways, whether it concerns making voting systems usable by people with visual impairments or making polling places reachable not only to people whose physical disabilities make it impossible but also to people for whom the cost of going there physically can be too high — often the elderly and those who live away from polling stations [BRB12].

[f] It is possible for redundant electronic and paper-based systems to co-exist, as in Estonia, but this has an adverse effect on the adoption rate [VSV+16].

[g] A good example of this is the Brexit, where a repeatedly denied request for a second popular vote in the face of low government approval and inability to find a solution created a legitimacy crisis. The legitimacy contested here is not the technical aspect — although there is solid evidence of a technically advanced targeted disinformation campaign — but the fact that the vote itself, although open to all, does not represent the public opinion [Low16], contesting the primacy of the popular will. Compounding this effect, the opinions as to whether to organise a second vote are well split between the people who agree or disagree with the result [Bou16b], and most of it is driven by demographic effects [Kel18b].

Figure 7.1: Sample ballot for Prêt à Voter system. The voter receives the ballot and makes a mark on the right side before tearing off the left side, making it impossible to find out how they voted.

today are still based on similar ideas. We shall now quickly describe three different proposals, starting by Prêt à Voter, which has the advantages of being among the simplest and most intuitive to understand, greatly enhancing its usability, and being among the best studied. The second, Three-Ballot, is also simple but seems counter-intuitive to many, and as such suffers from low usability. However, it has the rare — but not unique — advantage of not relying on cryptographic techniques. The last one, called Random Sample Voting (RSV), is more recent and was also developed by David Chaum along with the RSV project — of which the author of this thesis is a member. It has the main advantage of adding a rare feature: incorporating sampling into the voting process.

7.2.1 Prêt à Voter

Prêt à Voter has more than a few different versions, due to its continued development since 2004-2005 [RBH+09, CHJ+14, CRS05, Rya11, DHvdG+13, KTR13]. It is based on a simple idea: every ballot can be split into two halves by cutting along a central line, with the left half having the candidate list in a pseudorandom order, and the right featuring boxes for making a selection as well as a unique string. The string — called the *onion* — encodes information as to the order of the candidates on the left half.

Part of the security of the system comes from the fact that the right half of the ballot, by itself, provides no information as to the voter's choice if nothing about the left half is known[h]. As the left half is destroyed, the only way to find the original vote is to decrypt the onion. This onion allows the tallying of votes without linking the result to the original ballots, as long as some of the servers involved in the decryption are honest — moreover, the process is auditable.

[h] It does provide one piece of information, namely the position on the paper of the voter's choice, which can be used in randomisation attacks as seen below.

There are multiple different ways to implement an onion, depending on the properties sought. One strength of Prêt à Voter is that it is compatible with many vote tallying systems. People can select one — or multiple — candidates, or instead rank them in the order of their choice, by putting either crosses or numbers on the right half of the ballot. Although having a random cyclic shift is enough to guarantee *integrity* and *privacy* in the simple selection case, it becomes vulnerable when using ranked choice. For simplicity and brevity, we will describe a high-level view of the cyclic case, using decryption mixnets — more details can be found in [RBH+09] and [Cha81]. The process goes as follows:

1. A set of k mix servers are initialised, and each creates two RSA key pairs, publishing the public keys and distributing the private keys among the others in a threshold fashion.

2. An election authority creates a list of ballots, and for each ballot a set of $2k$ random values $g_0 ... g_{2k-1}$.

3. The onion D_{2k} is iteratively encrypted using the i-th public key and random value, as $D_{i+1} = \{g_i, D_i\}_{PK_i}$. Each time, the ballot is shifted by $HASH(g_i) \mod \#(\text{candidates})$ cyclically.

4. The voter makes their choice on the ballot.

5. The first decryption server uses their first private key to decrypt the onion D_{2k}, obtaining g_{2k-1} and D_{2k-1}, they can then inverse the permutation as they know g_{2k-1}. They do this for a batch of ballots and sort them lexicographically — hence the *mix*-net — before sending them to the next server. The sorting makes it impossible[i] to correlate the input ballot with the output ballots.

6. Auditing can then be used as in the original paper to show that the mix-net is working as it should.

Prêt à Voter has many good properties, such as a large set of variants when it comes to encryption back-ends or compatibility with a wide range of vote tallying methods, and being quite familiar and user-friendly. It seems to resist most attacks — or has versions that do — but two of them still merit mention. The first is the *discarded receipt attack*: if voters get rid of their receipts, and those get acquired by adversaries, they can know which ballots to target, making other attacks easier (although still quite hard to pull off). The second attack is called a *randomisation attack*, and corresponds to asking voters to vote for the top candidate, no matter who it is. This gives — in expectation — an equal share of the vote to all parties, which can be quite useful when the adversary targets specific polling offices where they expect a bad performance. It can, however, be quite detected relatively easily if two properties are true: the tallying is done by district, and there are many candidates including some from parties that expect a very poor performance, where a boost would be most visible[j].

[i] Impossible with the usual assumptions, such as a correct implementation of the algorithm and polynomially bounded adversaries.

[j] A dummy candidate or a blank spot on the left side could also be added to detect that kind of attack more efficiently.

7.2.2 Three-Ballot

A completely different take on the subject comes from an early design by Ronald Rivest [RS07]. Unlike the previous design, it does not require any cryptography, relying instead on a probabilistic principle. The idea is extremely simple and can be shown in many different ways, but here is probably the most intuitive variant, for a single election with two candidates:

- Each voter is given three quasi-identical ballots with the possibility of voting for one candidate on each. The only difference between the ballots is that they all have a unique identification number.
- The voter votes for candidate A on one ballot, candidate B on another, and the candidate of their choice on the last one, with a machine checking that their choices respect this rule.
- The voter receives a receipt corresponding to a full copy of one of the ballots — either randomly or at their choice.
- The ballots cast are all published publicly, either physically or by scanning them and putting them online, where the tallying can be done directly.

Although this system adds many votes, it is strictly equivalent: instead of giving a single vote to the candidate of their choice, the voters give a one-vote advantage. The secrecy constraint is satisfied because the voter has no proof that they voted any single way: the person for whom they voted on the receipt is not linked to the person to whom they gave a one-vote advantage. The verifiability constraint is satisfied, although in a probabilistic fashion. As long as no-one knows which receipts are kept, removing or changing the content of a ballot comes with a $\frac{1}{3}$ probability of being caught if the corresponding voter checks their receipt. Naturally, not everyone checks their receipt, but as long as a constant fraction of the voters do, the probability of a modification not being discovered becomes exponentially small with the number of ballots affected.

The system can naturally be extended to handle many different candidates and multiple races. There are a few advantages to this method, the most important ones being that it is paper-based and requires no cryptography. However, it is also harder to understand, slow, and cumbersome, ending up with low usability [JJB06]. It is also mostly vulnerable to something called the *Italian attack*. When used for multiple races with many candidates, a ballot receipt can become unique, getting rid of the secrecy property, and the numbers needed aren't outlandish for American elections [HSS09]. Variants of Three-Ballot made to improve its usability will be shown in the next chapter.

7.2.3 Random Sample Voting (RSV)

The last system showcased is not only a voting system. Instead, it integrates a round of sampling before the vote, in which a fraction of the electorate gets selected uniformly at random. This fraction then votes using an electronic verifiable voting system[k] [Zha17]. In a slightly simplified version for binary choices, the vote happens as follows:

[k] Another possibility is sometimes compared, where everyone votes but only some of the ballots are tallied, at random. This alternative has next to no advantages, however. The main difference is that RSV — and other sampling systems — reduce the costs of organising elections, and allow the voters to focus on one issue only, with the possibility of organising parallel votes with different representative samples.

- The election authority prepares a database containing a list of n ballots, plus a certain number of decoy ballots — explained further down. It then computes 250 different randomisations of the database, encrypts them independently and publishes them. Each ballot contains a random number uniformly taken between 1 and the number of people on the voter roll.
- The complete voter roll is published, and random numbers are drawn publicly for each ballot.
- The random numbers are added to the previously set number on the ballot to obtain a random individual on the voter roll.
- The people to whom the ballots correspond receive a ballot sent by the election authority (who doesn't make their names public).
- The selected people can cast a vote electronically[1] by using one of the unique serial numbers on the ballot they receive — each code corresponding to YES or NO. They also check that no-one else has used the codes.
- At the end of the voting period, more random numbers are drawn, and the election authority decrypts certain pairs of columns of each database, chosen randomly, in a way that allows auditing to make sure that every step happened correctly.

Just like Prêt à Voter, RSV has many variants, as it has been in continuous development for more than 6 years, and has also inspired the creation of other systems, such as Alethea [BRS18]. The protocol shown above is simplified, as a complete explanation of the cryptographic reasoning would take more than a dozen pages. The full version would take a few dozen pages to describe entirely, and can be found in [Cha16]. A (complex although simplified) schematic of the complete system can be seen in Figure 7.2.

As a voting system, RSV is more complex than the previous two, making it harder to understand — having probabilistic and cryptographic components — and less usable. However, it has many advantages when it comes to social aspects. For example, it makes it easier to have a high turnout and more informed voters, according to most versions of the rational voter model, which states that the time expenditure for a voter increases with the relative power, which increases when the number of voter diminishes [Con01, Gey06, BDT03, RO68]. Moreover, it allows for multiple votes on different subjects without the detrimental effects of voter fatigue [RTB03].

[1]To address the issue of voting accessibility for people with no internet access, the scheme is also compatible with proxy voting. It is enough for the voter to ask someone to login using their code, only giving them the code for the option of their choice. By doing this with multiple proxies and asking them to check that the code is in the list, they can make sure that their vote was indeed cast.

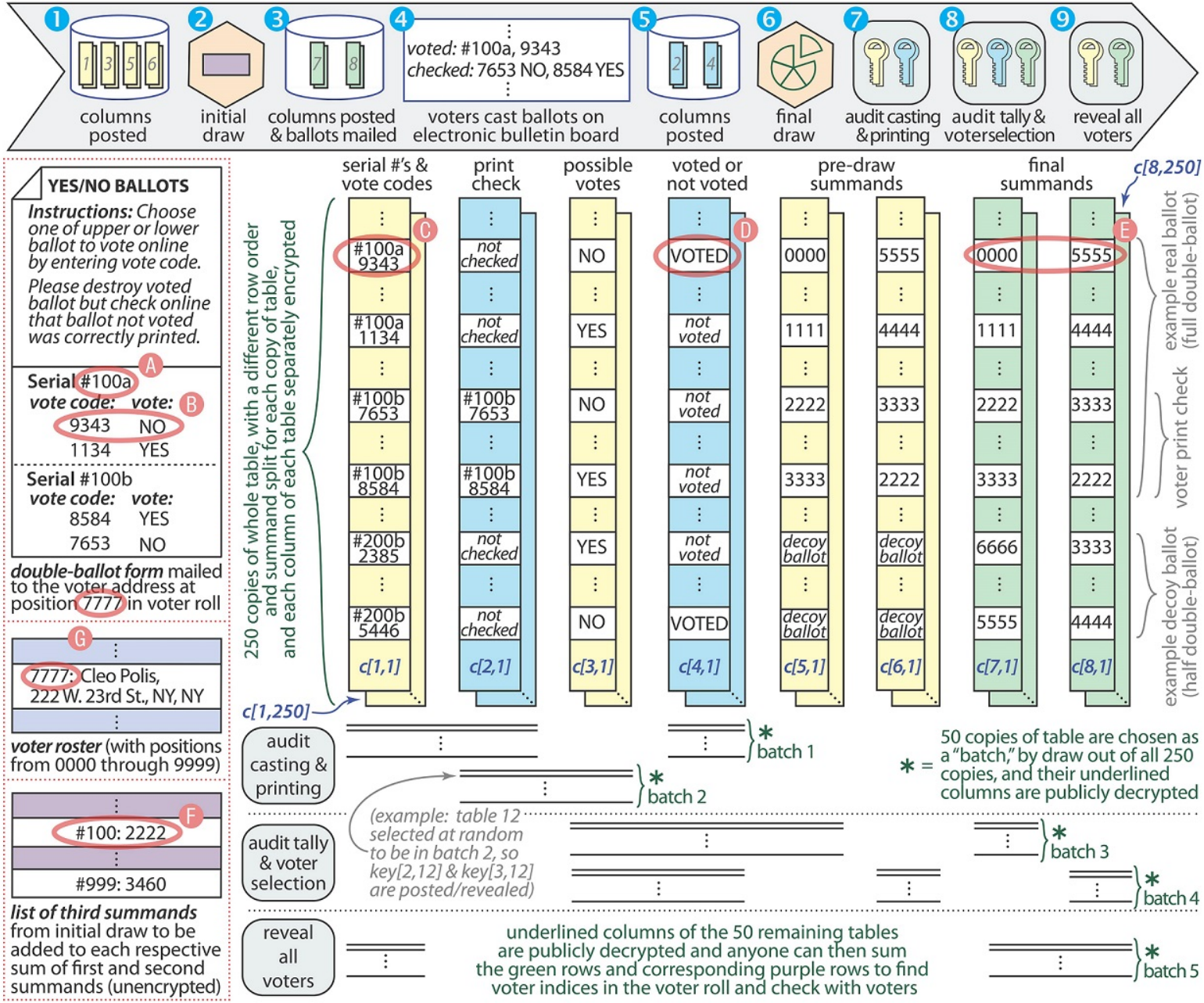

Figure 7.2: Schematics for the complete RSV protocol, as shown in the white paper [Cha16].

With all these new alternatives, one can legitimately ask why none of them have yet been used on a large scale. It turns out that there are many obstacles to overcome, especially when it comes to the perceived legitimacy of the proposed systems. In the rest of this chapter, **legitimacy** will be defined as the perception (generally by the public) that the decision-making system used is suitable in the given context, or, for a decision, that it was made using legitimate methods. **Trust** will be defined as the belief in the integrity of the system: belief that it is behaving as it should, with no risk of manipulation or modification of results (e.g. through voter suppression, fraud and hacking).

7.3 The difficulties inherent to implementing new voting systems

The first problem we face is that voting is far from only being a mostly technical issue: it's a social act, and this social aspect permeates everything around voting. Moreover, it is an act that we do rarely; this makes it more unusual and therefore easier to make mistakes on. It is an activity that carries great import, which brings with it stress, making it, again, easier to make mistakes. Most elections are local: they are run by local people, and the decisions on ballots generally mostly surround local decisions. In some places, the content of the ballot might change up until the election, and it is typical that ballot design is laid out by a self-taught election official. This need for planning and executing elections locally is part of the reason why decisions are often made on the spot or by individuals that might or might not know all the ramifications of them. Local officials are generally tasked with presiding over the following issues, all of which can impact elections and their legitimacy [AAA+01]:

- How people register, which is now often done online but has historically been an important source of lost votes in the USA [Bea14, Lev07].
- How people find out about the polling places, which is being obviated by mail-in voting and independent information campaigns, although fraud in mail-in voting is being increasingly scrutinised [Fay05].
- How people find out about the times and places they will vote: making this difficult in various ways has been a popular approach for voter suppression.
- How polling places are set up, especially regarding accessibility, and how people line up to enter, which is also one of the biggest ways votes have been lost.
- How voters are allowed to present themselves (with or without ID and at which station), which has been a source of confusion and has been used for voter suppression as well.
- The way the equipment is made available to voters: improving voting equipment in Florida in 2002 didn't immediately change the percentage of residual votes [m] — a metric used to measure the quality of voting technology — which was 2.9% in 2000 [THP+05]. Using the same equipment again in 2004 with a large training

[m] Residual votes are known as informal ballots in Australia, rejected ballots in Canada, and blank or null ballots in France and Spain. They consist of ballots that have been spoiled — for example, by writing on them — and are discarded during the counting.

and supervision element brought this number down to 0.4%, a quality never seen before [HPT+10, SI06].

- How the ballot is designed: the result of what was then the most famous House election in the USA was called into question because a design problem led more than 15,000 voters not to make a selection on that race in Sarasota in 2006 [Jef07].
- The way the ballots are counted, with centrally counted votes having typically 0.5% higher residual ballots than locally counted votes.
- The way the ballots are transported before the tally is made, which can be by helicopter, by car, on foot or electronically.
- The way the tally is made (spoken over the phone, by hand, with humans transferring numbers, with ballot modules, with ballots themselves) and the way the database is accrued.

Each of these steps must be done carefully; all steps need supervision, never letting a step be done by a single piece of software or person without being checked. With all these decisions to make and guidelines to follow, when a vendor says they can simplify the process, it is tempting to have them print or even lay out the ballots, deliver the ballots, choose the ballot machines, set up the counting machines, run the counting machines, supply the back-end counting equipment and even run the equipment.

New voting technologies also face a peculiar conundrum, which we could call a trust-legitimacy-participation loop, where each one of these aspects requires the other two. Here, turnout is used as a proxy for participation, as it is a commonly used measure of how invested the population is in the result of the vote [SJT04, Cze06].

7.3.1 The trust-legitimacy-participation loop

This cycle is based on three different phenomena, detailed below:

- High turnout depends on the presence of strong incentives.
- Perceived legitimacy depends at least on participation.
- The ability to give better incentives requires the legitimacy of the system to already be established.

Incentives and turnout

The first link in the cycle is well established: increasing the incentives (what pushes the individual to vote) increases the turnout. When we consider incentives that have to do directly with the election, such as decision-making power, this has multiple forms, all positively affecting turnout:

- Reducing the size of the electorate for similar decisions.
- Increasing the importance of the decisions.
- Increasing the closeness of the vote (making it harder to guess who will win).

These effects have been studied extensively both in theory [Nur99, Gey06], in the laboratory [LP07], and a posteriori on national and supranational statistics [BRB12, CG16]. For example, one can look to French elections for how voter turnout increases with perception of the value of the election. Turnout in recent years went from 40.6% in European legislative elections to 74.6% in the presidential elections, as the public perceived the first as having lower importance [PGY16]. To compensate for this and maintain high turnout in all elections, some countries, like Australia, have implemented compulsory voting — with fines for those who do stay away from the polling station without a good excuse, although said fines aren't always applied. In its first year, in 1924, compulsory voting increased turnout from 59.4% to 91.4%, and kept it between 90% and 95% since, in both state and federal elections [Fow13].

Turnout, trust and legitimacy

Turnout is a common standard for the legitimacy of a democratic system that can be directly linked to trust. This manifests in two ways: first, trust in the fairness of the electoral process and equitable access to voting is validated by high turnouts, establishing the legitimacy of the outcome. Second, trust in that fairness pushes people to vote, increasing turnout — the observed low turnouts in ex-soviet Eastern European countries have been tentatively explained by the limited legitimacy of post-USSR regimes [Cze06]. The correlation has been observed repeatedly all over the world [Cİ13, Jar14, GS04]. There is no simple causality at hand here, as multiple feedback effects come to bear: the first being that turnout is often seen as a proxy for the legitimacy of the vote (or the regime) [Bee91]. What complicates the matter is that trust in the voting process — which partially depends on the legitimacy of the regime — in turn, affects turnout. As stated in [OvdB04], "apart from the obvious requirement that the votes are tabulated correctly, it is vital that the votes are seen to be tabulated correctly". Indeed, people who don't believe their vote will matter will tend not to vote — and a perception that the politicians are corrupt has been shown to negatively affect turnout on a European scale [SS15]. The most appropriate model for this might be as an evolving multi-variate system, where the levels of both legitimacy and trust at time t have an impact on turnout at time $t+1$, and reciprocally.

Perversely, vote buying and coercion have been used to give the impression of trust and boost the perceived legitimacy of elections, from Russia to Honduras [FRS18, Cİ13, GOKdJN15]. The other way around, claims by Donald Trump that the election was rigged, during the American 2016 presidential race, could have been meant to lower turnout, following the commonly stated — although debated [NM96] — theory that increased turnout favours Democrats in the USA.

Because of this established link, any vote that suffers from low turnout could be challenged with the result perceived as illegitimate[n]. This has happened with the presidential election in the United States of America, the Brexit, and more recently in the French legislative elections [Hol16, Low16, Mic18]. More severely, in August 2018 in Pakistan, an election was declared void due to low turnout among women [Kha18], and in April 2018 a by-election was cancelled in Kashmir due to both fear of violence and very low turnout (7%) at the previous by-election [Baw18].

[n] Historically, this could have been reversed: after the implementation of secret ballots in the USA, which hindered vote buying, turnout decreased by 7% in gubernatorial elections [Hec95], presumably as people had fewer incentives, not being paid to vote.

This is a real risk for innovative voting technologies, as a single failure to provide high enough turnout in a real election could be used as a motive to void its outcome and drop the technology. Thankfully, two aspects could mitigate this and offer potential remedies. The first is that this effect is mostly present in countries where voting is not compulsory or where the compulsory nature is not enforced [GS04]. Countries with mandatory voting could then provide interesting testing grounds while mitigating the risk of harming the legitimacy. The second comes from work done in countries with low turnouts, such as the USA, with historical turnouts oscillating around 55% for presidential elections and around 40% for midterm elections. In those, other metrics for election quality have already been used. One common standard for voting technology quality in the USA has focused on residual votes: which percentage of the people who went to vote did not succeed at having a selection recorded for the top race on the ballot [AAA+01]. Care should be taken with this metric, however, as residual votes can also be interpreted as "protest" votes that were left intentionally blank to fulfil the voting obligation without giving assent. This is dependent on the political culture, and residual vote can be a consequence of political choice and not technological issues. For example, in the USA, residual vote rates around 1% have been considered normal, and their uptick to almost 2% in 2000 was a scandal. With better supervision and technology many states in the USA have since enjoyed residuals under 0.5% [ASI05]. In France, residuals might run between 0.9% and 3%, with frequent bumps due to *protest voting*, up to 12% in the 2017 presidential election [Sau12].

Incentives and legitimacy

The third link in the loop comes from the problem that implementing strong incentives is difficult when legitimacy is not established. This is related to the fact that changing technology on a large scale is hard to do unless there is a strong public desire to do so. Let's first show the relationship between those two before studying the second.

Incentives can come in a variety of ways, but the reason why people vote is generally not the belief that one's vote will be the deciding factor [LP07]. Instead, major incentives include civic duty, social pressure, passion for an election, legal obligation, or financial reward. The first two are some of the most frequently cited reasons to vote, with more than 45% of voters in certain countries citing that civic duty is the main reason they vote [UK 17]. The last one can also be dangerous as it has come with undemocratic fraud in certain settings [LMQ14, Nic08], but also because it can have detrimental effects as an incentive [McG78, BHSY00, MW09].

To attain high turnout, tests and demonstrations of voting technologies require not only strong incentives, but also incentives that follow the same structure as the system they seek to replace [BIL+14]. For example, saying that a voting technology is easy to use and showing as proof its huge turnout in an experiment where participants were paid to vote would probably not get accepted by the public (or experts). This is mostly true for experiments that change the nature of the voting rules, and less applicable to interface changes. Usability experiments have helped develop better ballots, better accessibility technology and even tested how to improve people's ability to audit their ballots. Still, acceptance can be fickle and without it one can't get voters. Making people vote on subjects of no importance, on the other hand — or artificial importance, through gaming, for example — is a poor indicator for the participation rate in actual live elections. The final test must then be to actually implement them in live meaningful elections.

To get strong ecologically valid incentives, voting technology might then need to be used in real-world situations and live elections where the result of the vote matters. This can typically fall into two categories: either in the public eye following a national deliberation on the subject, or independently through the initiative of local officials in low-profile "pilot" elections. Evidence for the difficulty of the first abounds, as, even when the legitimacy of the new system is well-established, changing it requires enormous pressure.

The secret ballot (also called the *Australian ballot*), which was adopted in 1858 in Australia, took about 40 years to be used in the USA. There is strong evidence that the main reason that eventually led to its adoption was the concerns over massive fraud and intimidation in the 1876 and 1884 elections [Jon03, Kin01]. Even when the legitimacy of the candidate system is already established, changing systems generally requires a scandal — more recently, the 2000 USA presidential election scandal was the impetus for creating and funding the Help America Vote Act [Kim03], which allocated large funds to modernise equipment and training. And when local officials decide to change the system's rules, this is fraught with risks, as is shown in section 4.

7.3.2 Comparing voting systems and the standard for legitimacy

Two effects reinforce the previous point on the difficulty of implementing large-scale changes in voting technologies. There is a nuance between them, but they work in the same fashion, by comparing the new system to an idealised pre-existing one. The first effect is quite simple, as for any voting system:

- Either it cannot change the outcome of an election, in which case it can only be legitimised by being more secure, cheaper or easier to administer.
- Or it can change the outcome, in which case it is not considered legitimate by a large part of the electorate.

It is known that various electoral systems — like first-past-the-post, Borda or instant-runoff[o] — elect different candidates under the same conditions [Nur99]. And recent experimentation has shown that this happens not only in theory but also in practice [BIL+14, BGI+13]. Recently, some activists have had a measure of public support in fighting for alternative voting systems, like majority judgement [BL16]. However, the ultimate decision generally resides in the hands of elected officials, and those have little interest in changing the system that elected them unless it improves their odds — for example, through gerrymandering [Per02, Nor04].

This doesn't mean that changing systems in radical ways is impossible, but it does create an additional obstacle to overcome. It is especially true for projects that include one form of direct or liquid democracy [BZ16] — which has seen a revival recently, partially due to the emergence of blockchain technology [Bou16a]. What matters here is not whether those projects have inherent value, but that there is sometimes large public support for it, which officials might want to use. This can lead to actual successful implementation on

[o]For a quick recall, first-past-the-post or plurality elects as the winner whoever has the most votes, when voters choose a single candidate. In Borda count, voters rank candidates, who receive points depending on their position on each ballot, with the candidate with most points winning. In instant-runoff, voters also rank candidates, but only their top choice counts initially. Until one candidate has more than 50% of all votes, the one with the least amount of votes is eliminated, and the votes it received are transferred to the next candidate on each ballot.

a large scale, as in Taiwan [Ms19]. However, it can also be a front, as when the city of San Sebastian organised the Global Forum on Modern Direct Democracy in 2016 [Dem16], which led to public demonstrations, as citizens criticised the hypocrisy of the city having rejected all local direct democracy initiatives[p].

Although elected officials might not idealise the system that got them elected, they generally have an incentive to keep it intact unless there is massive public pressure. But the public is the target of the second effect, as it is more vulnerable to an idealisation of the status quo. Studies have shown that the perception of a voting system is not directly linked to its properties. Instead, although security aspects have a relevance in the public estimate of the trustworthiness of a system, social considerations and networking effects also have a major impact [OvdB04]. This is even more relevant in unusual voting systems, for example, ones that use probabilities either as part of the validation [RS07, SCM08], or as part of the voting itself [Cha16]. In the first voting experiment detailed further down in subsection 7.4.1, participants expressed great concerns over the legitimacy of probabilistic systems, with the main argument being that it raised the possibility of an error or misrepresentation. This status quo bias came from the assumption that the system they were most used to — paper ballots that are hand-counted under supervision of multiple assessors — is error-free. This is naturally a wrong assumption, as a recent study has established that the error rate for such ballots in the same country (measured as discrepancies between the number of votes and the number of voters) was around 0.18% with 9.7% of polling stations reporting at least one error [EG14][q]. When discussing this with the participants, they expressed great surprise at this error-rate, initially thinking that errors happened at a much lower frequency.

7.3.3 A hostile environment for scientifically rigorous innovation

The trust-legitimacy-participation loop and the problem of standards for legitimacy mentioned previously would not necessarily be detrimental by themselves, as they only increase the inertia of voting systems by making it harder to change the status quo. Imposing a high level of evidence to make changes at the centre of our democracies only follows the precautionary principle.

The main issue with those effects is that they only apply when the changes made to the voting technologies and methods are implemented with well-designed evaluation, supervision and following a public decision. In many cases, the public is barely informed of changes being made by election officials, who can often make changes in an arbitrary fashion. Here we must be careful, because the changes are generally made in good faith by people who believe they notice longstanding problems in their voting systems but do not have funding or access to technology and expertise. Election officials are the people that see the system work — or not — first-hand. They are typically motivated, in a place where they can fix things, and have the desire to do so.

In one example of good initiative on the part of election officials, poll-watching in some split precincts in Chicago revealed that they had different instructions for different

[p] To push this a bit further, while discussing such technologies with UK politicians, one salient comment came up, with a politician claiming that they agreed with direct democracy, but that the elected officials should be able to override the people. Approving a system where you follow the will of the people only as long as the people agree with you didn't seem to raise ethical concerns for this politician.

[q] This study showed that despite their automated nature, usability issues raised the error rates on electronic voting machines to 0.86%, with 34.4% of polling stations concerned.

punch cards in the same jurisdiction. In most such places election officials had taken it upon themselves to put a sign up directing the voters to the booth with the right ballot instructions. However, in one location they hadn't done this, which meant that 50% of the voters in that precinct were being given the wrong punch card template and instructions, frustrating voters and risking the loss of half the votes — a danger made possible by separating the instructions from the ballot. We must, therefore, not depend on the ingenuity of poll-workers to make things work correctly; they have too many other things to concentrate on.

Election officials have a complex job to do running elections, and far too often they inherit antiquated procedures, materials and technology. They may go to vendors to make up ballots, clean data or even write software — or they may invent solutions in-house, sometimes taking it upon themselves to build new voting systems from scratch. As these systems are developed in an ad-hoc fashion to solve local problems and not generally subjected to rigorous analysis, however, they sometimes ignore best practices and decades of prior work[r]. One striking example comes from the work of Wendy Noren.

In Boone County, Missouri, Wendy Noren, working as a County Clerk, was frustrated with an inadequate and antiquated system she had inherited to run her elections. Before any state had a real solution, she "*decided to develop her own election handling software. She learned how to code and programmed the entire election system, attempting to make it also tamper-proof, and improved voter experience by making it faster*" [San18]. According to the County Assessor in 2018, "*Wendy Noren wrote our personal property software, which is state-of-the-art and still in use today, second to none, in the state of Missouri*".

Noren's software continued for some time to be ahead of that used in most states — and then it wasn't[s]. At some point, professional programmers at election companies wrote statewide database software, which was subject to certification and audits and tested in different contexts and by different organisations.

Although Noren's approach was innovative and critical at the time, it does not hold up against standards of public oversight. The state was recently given a D rating on election security by the Center for American Progress [RKSP18] and the county had problems in their election software in their last elections [KMI18]. Despite this, county officials were certain of the security of their system, their main stated reason being that the voting equipment was not connected to the internet; and although they didn't release how or even whether they tested their technology, they expressed strong reticence at the prospect of changing to the new statewide system [Kel18a]. Moreover, even when there is motivation to change technologies because of security risks, the companies standing to lose have sued decision-makers in the past [Pal05].

Changing the implementation of part of a voting system — whether it is the counting software or the registration database — might affect trust and turnout even if it doesn't affect the outcome of an election [CM07, KH16]. On the other hand, changing the voting system or the mode of computation of the winner can have major effects on both turnout and outcome [Nur99, BC90]. In particular, going from plurality rules on a local scale to proportional or semi-proportional representation has an immediate effect on the balance

[r]The same behaviour has been observed in cryptography and network security with people developing new flawed encryption systems instead of using ones that are already established as secure [Sch98]. Alas, they are often very hard to change even in the face of serious problems.

[s]Noren used her experience with registration databases in her work with the Election Assistance Commission, and her work contributed to requiring statewide registration databases to assess individuals voting in more than one place.

of forces in elected assemblies. Although decisions on the first kind of change might be legitimate without strong public supervision, the second kind is much more problematic. However, quoting Bowler, Brockington and Donovan in [BBD01]: "*There is no single reason why some places adopted cumulative voting rather than single-member districting as a remedy, but contributing factors include the preferences of individual attorneys handling the plaintiffs' cases, differences between defendants and plaintiffs over potential districting plans, and local minority-group leaders' willingness to use an experimental system*". Hence, the changes made to voting systems and technologies often depend on arbitrary decisions with little global consistency.

The difficulty of changing a system once it is implemented can also be shown with one of the leading experimenters in voting technology today, Estonia. The country enjoys low political inertia — having declared independence only decades ago — and a reduced population of only 1.3 million inhabitants. The country's citizens carry one smart ID card used by 98% of citizens, as well as most long-term residents. This card allows them to travel, log into their bank account, manage health insurance and prescriptions, and, more recently, vote.

After an initial deliberation phase in the early 2000s, the country decided to hold its first remote electronic vote in 2005, and used it in all subsequent elections with progressively increasing turnout [VSV+16, Vin15]. Despite agreement on the importance of having a publicly available source code, or at the very least an independent review, they only allowed the latter in 2013. During that first review, the testers found multiple fatal security flaws, and recommended an immediate shutdown of the system, in agreement with multiple critics inside Estonia's institutions [SFD+14]. Despite this warning, Estonia is still using its e-voting technology, improving and affirming its security [Tef17].

7.4 Testing Random Sample Voting

Random Sample Voting suffers from multiple trust issues as it intrinsically depends on randomness. To have better ideas on how to address these concerns, we decided to confront people with it and get feedback. Two demonstrations were organised in international conferences, and the author also developed a voting *simulator*. The simulator was meant to give users the possibility to try many different samples, both on arbitrary populations and on data from historical French elections. Its interface can be seen in Figure 7.3. Due to emergencies during the voting process, the simulator could not be the subject of a user study, but a beta version can still be found online at `www.koliaza.com/rsvp`.

The next two subsections show results from two experiments we ran with RSV, reinforcing the points mentioned previously.

7.4.1 A preliminary test in San Sebastian

The RSV Project ran a demonstration which doubled as a usability study at the Global Forum on Modern Direct Democracy, from November 16th till November 19th 2016 [Dem16]. This gave us the opportunity to find multiple failing points, and gave us a lesson on the importance of usability in those systems. The first problem arose with the absence of a list of participants (or even of an accurate expected number), which meant that we had to simulate the random drawing of the sample. We decided to do that by holding two concurrent votes on different questions: "Should voting in national elections

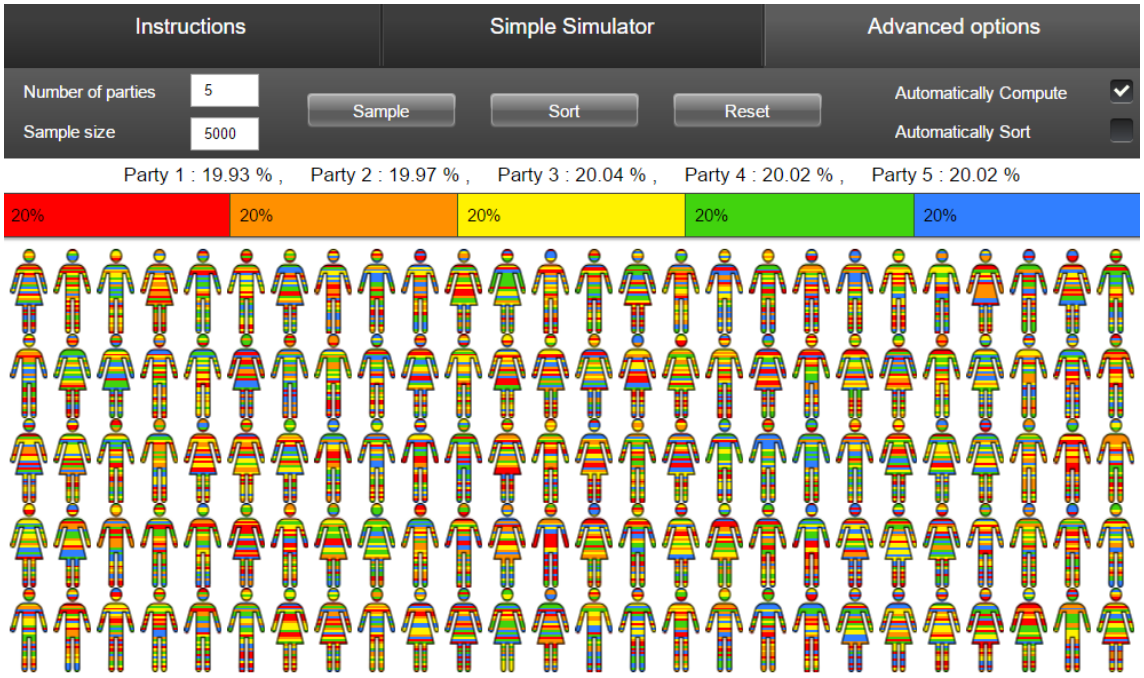

Figure 7.3: Voting Simulator for Random Sample Voting, which allows the user to select a number of parties, proportions of the electorate for each, and then run samples to see how representative the sample is if everyone responds.

be compulsory?" and "Should negative campaigning be prohibited?", with participants getting to vote on one of the two randomly. The process went as follows:

- 146 ballots of each type were printed and put inside identical envelopes and then shuffled, no one knowing in the end what each envelope contained, thus simulating the random selection, and making sure that everyone had a chance to participate.
- During the introductory talk about RSV on the first day, those envelopes were distributed to all the participants, ensuring that everyone got a ballot, but that no one knew who had what (130 ballots were distributed because of low turnout at the conference that day).
- The polls opened at 00:00 on the night of the first talk and stayed open for nearly two days until 21:00 on the penultimate day.
- People could vote on their phones or computers, and a public polling station was established for those who didn't have internet access. They were also encouraged to vote through a third person (generally an RSV representative), to demonstrate that feature. This contributed greatly to alleviate concerns over online voting, especially when it comes to unequal access to Internet.

Two complications arose in the experiment: just before the beginning of the conference errors were found in the voting parameters — thankfully we had just enough time to change those before the daily publication of the random bits — forcing us to redraw and reprint all ballots. Another problem was that on some devices the voting interface would hide the

last numbers of the input. We found a solution before the second day — which was as simple as holding one's phone horizontally — but many potential voters were confused.

Of the 130 ballots distributed, 34 were used, an abstention rate on par with previous tests of similar systems. Despite the low turnout, the results of both races were quite indicative (75% disagreed with the first proposal, and 64% disagreed with the second). If one assumes that voting intentions did not affect the probability of voting (which is a strong assumption), the probability that those opinions would be shared by the general public present at the conference would be respectively of 7.3% and 14.3%.

Moreover, we were able to analyse the different reasons behind this low turnout by getting feedback:

- The ballots were complex and confusing, being double-sided, making the perceived cost of voting too high. A sample ballot can be seen in Figure 7.4.
- The voting site's web address written on the ballots was long ,and people didn't want to write a long string into the browser.
- People were mistaking 'g' and 'q' in that link and couldn't access the voting interface. This motivated the work we later did, shown in Chapter 2.
- The secret IDs on the ballots suffered from the same problem, being 16-character strings.
- These problems, along with the interface bug, meant that many voters got frustrated and did not vote. Some were ready to try voting again after the fix was released (we recorded 6 ballots before the fix and 28 afterwards), but most had given up after the first try.

Moreover, we could observe — but not measure — additional psycho-social factors: the concept of negative campaigning was foreign to people from countries where such practice is forbidden, resulting in low turnout. More importantly, the timeline of the vote was badly chosen: people did not vote on the first night as it was not urgent, and they often forgot their ballots in their rooms on the second day, meaning that a few dozen couldn't get to their codes before the deadline. During the conference, someone remarked that they got a ballot for one question and were upset that they didn't get the other. They thought it would be a common feeling, and it is a real concern, although if RSV were implemented in practice, the frequent votes might prevent people from feeling they were chosen for one and not the other.

Multiple people in the crowd also voiced concerns about the popular reaction if the result of a vote differed from the popular expectation, which is unavoidable. We set up an anonymous feedback collection website accessible from the voting page, and found that people mostly trusted the accuracy and security of the system but were nearly all on the fence on its legitimacy for a mass election, meaning that we need to improve the popular appeal of the system.

7.4.2 A demonstration at the Council of Europe

The World Forum for Democracy is a three-day annual event organised by the Council of Europe, with a thematic focus that changes each year. On November 8th, 2017, as part of their "Is populism a problem" theme, the forum solicited multiple teams of researchers

Random-Sample Voting Ballot

QUESTION: Should voting in national elections be compulsory?

VOTING TIME: 12:00PM CET Thursday 17 November 2016 through 9:30PM CET Friday 18 November 2016

INSTRUCTIONS:

1 Choose either half of this sheet randomly (ballot number and password are the same for both halves).

2 Use a web browser to visit the webpage: `https://vbb.rsvoting.org/rsv/vbb/gfmdd2016-q1/`

Your ballot number is your **login ❶**: **`001`**

Your **password ❷** is: **`vhbe-buhb-mrda-fwpz`**

3 When prompted, enter the vote code that is printed adjacent your vote.

4 You should discard or destroy at least the half of this sheet that you used to vote; it is recommended, however, that you keep the other half of this sheet and write down on it in the space provided your vote code for later use in the audit.

Choice	**Vote-Code ❸**
Yes	4457-1444-2131
No	6975-7435-2625

Random-Sample Voting Ballot

QUESTION: Should voting in national elections be compulsory?

VOTING TIME: 12:00PM CET Thursday 17 November 2016 through 9:30PM CET Friday 18 November 2016

INSTRUCTIONS:

1 Choose either half of this sheet randomly (ballot number and password are the same for both halves).

2 Use a web browser to visit the webpage: `https://vbb.rsvoting.org/rsv/vbb/gfmdd2016-q1/`

Your ballot number is your **login ❶**: **`001`**

Your **password ❷** is: **`vhbe-buhb-mrda-fwpz`**

3 When prompted, enter the vote code that is printed adjacent your vote.

4 You should discard or destroy at least the half of this sheet that you used to vote; it is recommended, however, that you keep the other half of this sheet and write down on it in the space provided your vote code for later use in the audit.

Choice	**Vote-Code ❸**
Yes	4457-1444-2131
No	6975-7435-2625

Figure 7.4: Sample ballot used at the Global Forum on Modern Direct Democracy. Each ballot is double-sided, with a single side being selected at random to cast a vote, the other being used for potential auditing [Cha16].

— including the author — to set up voting experiments. Four separate democratic tests were planned:

- A global vote using *evaluative voting* to rate all but one of the proposed recommendations for the published conclusions of the forum.
- A second vote to approve or not each of the proposed recommendations using *Random Sample Voting* [Cha16]. Ballot-holders could vote only on one of the five proposed recommendations — they couldn't choose which, as it was selected randomly.
- A public deliberation on the last recommendation to decide its wording and vote on it using the same technology as the Council of Europe.
- A vote on the third and last day to decide which initiative was to receive the Democracy Innovation Award.

Participant eligibility Around 2000 people from more than 80 countries participated in the forum, with a third of academics, a third of representatives from NGOs and political parties, and the rest from a variety of fields [Wor17].

The first two voting methods required ballots with secret codes. These were distributed on the second day by a team of volunteers, making sure that no-one got multiple ballots by distributing it simultaneously in different places during a limited time frame and by putting stickers on the recipient's access card. The ballot promoted the voting experiment and gave an explanation, an online address and voting codes. Those codes were required to access the second voting system, but not the first (they were used afterwards to check that no-one tried to vote without a correct code). A total of 834 ballots were distributed by the volunteers, who also answered questions about the experiment. The voting period was from 10am on the second day till 10:30pm on the same day.

The third vote happened on the second afternoon in the main hemicycle auditorium of the Council of Europe, which has more than 500 seats equipped with voting technology and at least as many observer seats. It had a partial overlap with some of the conferences of the secondary tracks. It was open to every person present at the conference.

Finally, the fourth vote also took place in the hemicycle in the last main session, with everyone at a normal seat also having the possibility to vote.

Voting methods The first method involved evaluation voting, with participants being asked to go online and rate all 5 proposed recommendations of the forum from -2 to +2. The focus was not on security but on consistency with other voting methods.

The second method used Random Sample Voting, which — in its designed use (slightly changed for this experiment) — randomly selects a fraction of the voters to vote on an issue. Here, instead of selecting them randomly for one vote, they were randomly assigned to vote on one given issue (a YES or NO question), the focus being to test the security and legitimacy of the process.

We had learned from our previous mistake and improved the design of the ballot, making it clearer (according to the feedback we received), despite the constraint of making it trilingual to accommodate the diversity of languages spoken at the Forum. A sample ballot with the improved design for the election can be seen in Figure 7.5, to be compared with Figure 7.4. We also limited the voting time and teamed up with the team doing Evaluation Voting to advertise both systems.

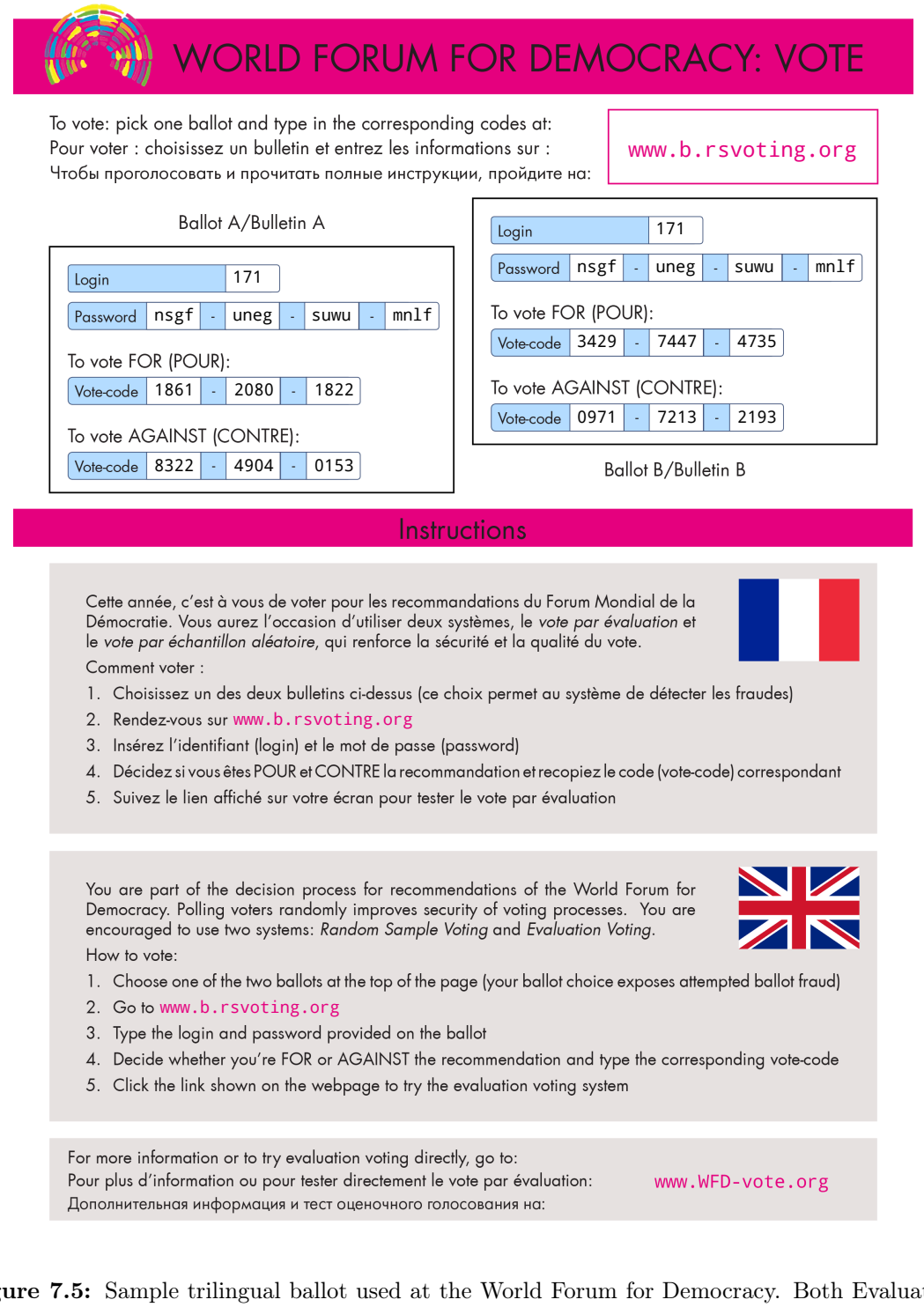

Figure 7.5: Sample trilingual ballot used at the World Forum for Democracy. Both Evaluation Voting and Random Sample Voting were advertised separately, with each voting system linking to the alternative system after the vote was cast.

The third method involved debating before voting using the push-button technology present at the hemicycle of the Council of Europe (previously used by the European Parliament). This technology is a simple electronic voting mechanism[t] where one puts their hand inside the dedicated slot in their desk with three buttons present (but shielded from the view of others) and presses the one they want (with the possibility to vote YES, NO, or NULL).

The last method used the same technology as the third, but instead of debating, the three candidates for the prize gave presentation speeches explaining their project just before the vote.

Awareness and incentives The last vote demonstration had the highest level of awareness and strongest incentives with many advantages:

- It happened in the main time slot when nothing else was happening, so the marginal cost of voting was low.
- It was advertised in the program of the conference and was supposed to be the conclusion of many of the secondary tracks.
- Participants had a decent understanding of the issues, having just seen presentations on them.
- It had a real immediate consequence (the attribution of the award).
- It included the lustre of voting from the seats and using the historic private voting technology that had been used by the European Parliament.
- It was a shared experience.

The third demonstration also had a high level of awareness, being part of the main program and using the same technology.

The first two experiments suffered from an organisational mishap in which an organiser dismissed the assembly just before the keynote speeches announcing the experiments, with around 80% of the participants leaving the main room before that could be corrected. Despite work by volunteers to compensate this, awareness remained low. The marginal cost of voting was higher than for the other two demonstrations experiments, but both experiments redirected voters to the other experiment once they had voted to raise mutual awareness. Finally, the incentives were supposed to be relatively strong (as they decided the recommendations of the forum). Alas, many participants felt that the recommendations were obvious, diminishing the importance of that factor.

Results As most recommendations being voted on were popular, they were all highly rated and selected with large enough margins to compensate for the low turnout — with no disagreement between the first two experiments. A recommendation was debated and voted on for approval in the third voting technology demonstration, and the innovation award in the fourth vote went by a large margin to Russian investigative newspaper *The Insider*.

The critical point in this comparative experiment was the turnout:

[t] Despite the simplicity of the technology, it is still error-prone, with at least 13 members of the European Parliament making a mistake in the recent vote on article 13 of the Copyright Directive [Kar19].

- Evaluative voting received a total of 67 votes, out of 834 distributed ballots.
- Random Sample Voting got 120 total votes for the same number of ballots.
- 25 people were presente at the deliberative experiment (including organisers), of whom 21 voted.
- No accurate number exists for the number of people present at the last vote, but between 60% and 75% of them voted. This number includes the many votes for the NULL option[u].

New voting technologies, even with people who should have cared (but had low incentives), didn't get adequate participation: even in a conference dedicated to democracy, among political scientists, politicians and activists, with a strong message that high turnout was essential, and despite all its advantages, the highest-turnout system didn't get more than 75% participation, and others were all but ignored.

7.5 Discussion

Despite, or maybe because of many recent advances, voting technology is far from being unified, and technologists don't even easily agree on which metrics are best-suited to evaluating it. Audit trails, generalised supervision, and end-to-end verifiable voting might seem like an achievable elements of a gold standard, but none of these constraints are yet entirely standard, let alone all of them at once. There are strong reasons which motivate the discrepancies, even inside countries like the USA, such as expressly delegated authority to organise elections[v]. Rogue technologists or election officials implementing their personal ideas have been a frequent avenue for innovative systems, or pilot studies, or promising new methods. Still, such experiments can and often do add problems of integrity, security, and accuracy. We must learn how to innovate without endangering the technical integrity and public legitimacy of outcomes.

Despite these obstacles, some new systems have improved legitimacy, an exemplary one being participatory budgeting[w], today used in hundreds of cities — such as Paris, Pune, Seoul, and New-York City. Multiple factors can help explain this relative success. First, unlike developments that seek to change the way we use voting, participatory budgeting was typically implemented in places where the constituency had little to no say in the decisions taken. This means that the legitimacy could only increase [CF16], even with low turnouts generally around 10% [SHA+13, GF15].

The initial low turnout can probably be attributed to a difficulty in getting to the polls, linked to the infrastructure in the South and Central American countries which were the first to use this technology. As such, the baseline for turnout comparisons is also low, with correspondingly low expectations. The relatively successive forays made by this idea could potentially be replicated in other settings. New voting systems could be demonstrated in

[u] The lack of accurate information is due to only some of the seats in the hemicycle having the electronic voting system, movement between the votes, and only partial video evidence of the process.

[v] As stated in Article I, Section 4, of the United States Constitution

[w] Participatory budgeting is the inclusion of direct democratic processes into the decision-making apparatus at the municipal level, with inhabitants typically voting for their favourite projects to be funded, with a small percentage of the municipality's budget appropriated for all potential projects. The modern iteration of the principle was first implemented in 1988 in Brazil.

specific settings where people have naturally stronger incentives but where the leadership has fewer constraints (for example, when it comes to votes inside large unions, or for shareholder votes).

Voting legitimacy is often challenged by the people who don't win, especially in contemporary times when anti-democratic discourse is visible and frequent. Advocates and developers of new voting technologies should be careful, both to avoid the pitfalls of low turnout if they want to be implemented, but also to avoid contributing to the delegitimisation of elections. This applies not only to the voting part but also to the auditing, where some promising technologies that avoid frequently found problems in paper audits — such as audio auditing — are discarded by officials and paper audit trail advocates [SRP06]. Testing of these has so far focused on only one side of the problem, generally security or usability, while public legitimacy has been set aside. However, it is critical, as it is required both for ecologically valid live tests and for actual implementation of the technology on a large scale. Moreover, even systems for voting that seem simple depend on the smooth functioning of many different parts. New approaches must then be tested holistically and rigorously with teams including not only mathematicians and engineers but psychologists and voting theorists, knowing that problems and constraints vary greatly with geography and jurisdiction.

Keeping this in mind, the next two chapters will look at proposals for systems that seek to achieve high security and privacy properties while focusing on the usability side. The systems presented in the next chapter haven't been tested empirically yet, but serve as a proof of concept that some of the primitives necessary for verifiable voting can actually be implemented in truly low-tech settings, without the need for trusted machines. The goal of the designs proposed won't be to be deployed in a large scale as a general solution, but instead to provide stepping stones that can introduce voters to the concepts while staying relatively intuitive and addressing the concerns about electronic systems.

8 Usable Paper-Based Three-Ballot

8.1 Issue statement and contributions

As stated by Randell and Ryan when verifiable voting systems were still at their very beginnings [RR06], and detailed in the previous chapter, voting is a socio-technical problem, with an insistence on the first half, and trust is a major component of it. With people being increasingly concerned with the threat of election hacking [Mai18] — and legitimately so [SFD+14] — a number of experts have warned about the lack of adequate technology [Orm17], and there is a strong pressure to return to paper-based systems, as it is supposedly much harder for an external adversary to massively manipulate them [Ju18]. Unlike the USA, some countries, such as France, are also still using paper ballots massively with little evolution in voting practice since the early 20th century [EG14]. Relying only on paper poses a problem for most of the newly developed end-to-end verifiable voting systems that guarantee the authenticity and anonymity of all ballots. The two main exceptions were introduced in the previous chapter: Prêt à Voter and Three-Ballot. The problem with the first is that, although it is mostly paper-based, it does rely on some cryptography for the decryption or re-encryption mixnets, requiring an election authority that can potentially be decentralised, but is still electronic. Even if experts could be convinced unanimously by its security properties, there is no guarantee that the public would follow, as it relies on complex proofs in an era where trust in experts is low.

Three-Ballot, on the other hand, does not involve any complex cryptography — although understanding its inner workings requires some basic knowledge of probability theory, which is slightly problematic[a]. It is also impervious to the randomisation attack that can target Prêt à Voter. However, it suffers from two issues. The first is the Italian attack mentioned previously: when voting for more than a few different races, unique voting patterns on ballots become possible, reintroducing the risk of coercion and vote-selling. This effect and its probability of happening in real races has been studied well in a variety of papers [App06, HSS09, Str06b]. Although it poses a real risk in places with many

This chapter is based on research done with Ted Selker.

[a] We are notoriously bad at problems of this kind, even in the face of overbearing evidence [HS10].

concurrent races[b], many countries — such as Spain, Greece, France or Malawi [RS06] — don't have many parallel elections.

The biggest weakness of the system has been its low usability, not only in the practical implementation [JJB06, Str06a] but also because of the very complexity of the scheme — requiring voters to accurately vote 3 times, once against their selection – which is known to make it harder for voters to use correctly [SL12]. Finally, the system relies on the assumption that the ballots are all correctly filled and checked, which is dependent on an optical scanning machine which scans and validates the ballots without storing them, introducing a vulnerability coming from the use of potentially insecure hardware. The proposed solutions so far all rely on electronic remedies either through trusted hardware [UB14] or online services [SCM08].

Contributions The above problems motivate the three potential solutions proposed below. They are all physical implementations of Three-Ballot that seek to optimise usability without requiring to be checked by electronic devices. The first solution relies on translucent paper, allowing a voting official to check that the ballot is correctly filled without knowing who the voter voted for. The second is similar but simpler for the voter, with the higher usability coming at the expense of increased manufacturing complexity and cost. The third solution is based on folding and hole-punching and has multiple desirable properties, including resistance even to attacks where voters film themselves in the ballot booth, a practice sometimes authorised under the name of "ballot selfies" [Hor15].

As with Rivest's original scheme, it is possible to use optical scanning machines to check the ballots. However, the fact that a voting official can check the ballots without gaining information means that one doesn't have to rely on those machines. The ideal system might be to have people randomly assigned to one or the other, with discrepancies indicating probable fraud. One surprising feature of this last system is that it could conceivably have been used in ancient Greece, if one replaces the bar-code by a random unique symbol, as it could have reasonably been explained without requiring advanced theoretical concepts.

8.2 Constraints

To limit the confusion of voters, the execution of any potential protocol should be familiar, hence close to the following:

- The voter comes into the polling station, receives information and proves that they are a registered voter (e.g. by showing the relevant ID);
- They are given instructions as to how to vote[c];
- They obtain some physical objects if necessary (e.g. ballots, pens, envelopes, magnifiers);

[b] Linked to the problems with many parallel races, having many different candidates on a single ballot increases confusion and proximity errors, with smaller candidates adjacent to high-ranked ones getting an additional 0.4% of the latter's vote [Sle03].

[c] As has been suggested [HPT+10], in the first few public uses of any new voting system, all users should receive detailed instructions and a test experience to show how they can use the voting system and ask for support before they mark their actual ballot.

- They move into a privacy booth where they can manipulate the ballot;
- If needed, a machine or a voting official checks that their ballot (or envelope) is correct;
- They cast their ballot, either by inserting it into a ballot box or by any other method.

Moreover, the protocols should satisfy the following constraints, in decreasing order of importance:

1. It should not allow multiple voting: there should be no way for a voter to give a multiple-vote advantage to a single candidate. This should hold even if some but not all other agents (such as voting officials) are corrupt.
2. No third party should be able to find out a particular voter's vote, and there should be no way for a voter to prove that they voted a particular way, to prevent corruption and coercion.
3. As a consequence of the previous constraint, if a receipt is given, the vote indicated on it should be either chosen by the voter or close to uniformly distributed among all possibilities.
4. If some of the ballots are modified after being cast, voters should have a constant probability of being able to find out and prove that there was a modification.
5. A voter should not be able to prove there was a modification when there wasn't, even if their initial ballot was not correctly filled.
6. Finally, the whole system should not depend on any single electronic or human agent that could change the meaning of any ballot or count unnoticed [d].

The above constraints have to be supplemented by some additional concerns which are crucial to any voting system, not just the ones considered here. The voters should be comfortable with the ballot, with its use, and be reasonably confident whether they have used it correctly. They should also know how to spoil their ballot and get a replacement one if they make a mistake. Finally, they should have confidence in the fact that they voted correctly and that their vote is private and secure.

This forms a part of the main goal, which is then to optimise usability and simplicity while satisfying the constraints. With this said, we can present the first protocol.

8.3 Translucent ballot

8.3.1 Protocol

This first protocol uses a ballot on which voters can write. The design, as indicated in Figure 8.1, has three similar single ballots side-by-side, with one receipt under the left ballot. Each ballot has four different parts:

[d] We can reasonably assume that some voting officials should be honest, which introduces redundancy for counting, and each of the steps should be corroborated by a group such as one representative from each party and one election official.

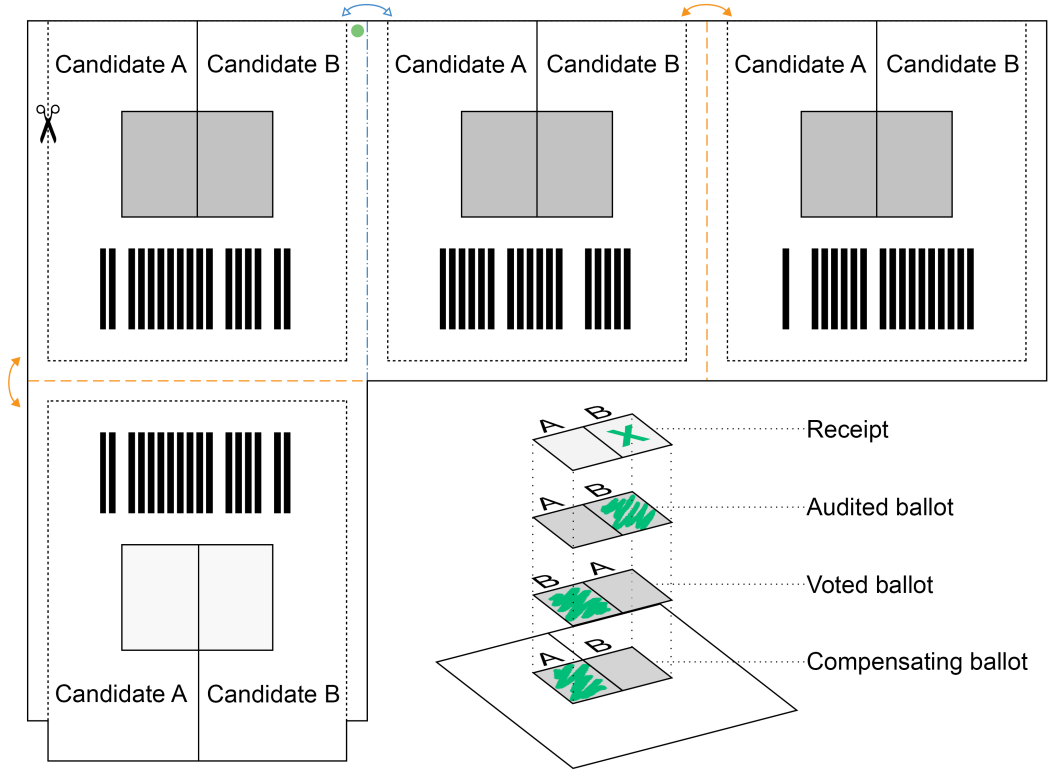

Figure 8.1: The translucent ballot and on the bottom right a view of the superposition of the translucent rectangles when folded. The three ballots and the receipt are separated by dotted lines which correspond to the folds. Orange folds — with the filled double arrow — correspond to valley folds, meaning that the two faces visible to us will be face to face. The blue fold line — with the empty double arrow — is a mountain fold, such that the two visible faces will be facing outwards. Once folded, all the cutting lines are aligned with the receipt sticking out, allowing the voter to keep a receipt that allows them to know their ballot was included. The ballots are simultaneously cut and dropped in the ballot box. The only difference between the three ballots lies in the green dot which is cut off in this process.

- A central translucent rectangle split into two cells, one of which the voter has to cover by marking over it;
- A legend over each cell, indicating which candidate it corresponds to;
- A single unique but not memorable ballot segment identification method — here a bar-code — under the translucent rectangle;
- A single green dot in the top right corner of the left ballot.

The receipt has a fully transparent rectangle in the same position, but otherwise, the elements are the same as in the left ballot with the vertical order reversed, with the bottom of the receipt being slightly narrower and longer. When folded over, rectangles should be aligned with each other, and the green dot should be visible, with the bottom of the receipt protruding, to be removed after the voter casts their ballots.

The instructions for the voter are as follows:

- Choose whether you want to audit[e] your ballot for A or B, colour the corresponding cell on the left ballot, and make an X on the corresponding cell on the receipt. Colour the cell corresponding to the other option on the right ballot.
- Choose whether you want to vote for A or for B, and colour the corresponding cell on the central ballot.
- Fold the three ballots horizontally, leaving the central ballot between the two others.
- Fold the receipt vertically on the same side as the ballot it's attached to.
- You should end up with a single stack of ballots, with no visible bar-code and a green dot visible in one corner.

The instructions can be indicated directly on the ballot in the space left (if there is enough space, which depends on ballot size), both textually and diagrammatically to avoid language issues. Alternatively, it could also be printed on the remaining space if rectangular sheets are used, but that creates security risks if one isn't careful[f].

The ballot must have the following properties:

- On both ends of the stack, there is a single cell that is entirely coloured. This cell is different on each end. Other than the cell, ballots on each end aren't marked.
- On one side, an X is superimposed on the coloured cell, and a green dot is visible in the corner.

Once this is done, the ballots are separated from each other with a paper guillotine, along the dotted lines. The ballots are all cast into a ballot box[g] and the voter keeps their receipt. The ballots are then all mixed and revealed to the public (which can be scaled by scanning them and putting them online, this electronic part being independent of the vote).

8.3.2 Constraint satisfaction

We can now check the six constraints:

1) To check the first property, the officials make sure that there is at least one ballot that is for A, and one for B. Those two ballots are easiest to check are they are the ones on the bottom of the stack and the one below the receipt on top. The last (central) ballot doesn't matter, as it is either valid (a vote for one candidate), blank or entirely coloured, and the last two options make no difference. Thus, the voter can't give a 2-vote advantage to a candidate.

2) Because the rectangle is translucent and there is at least one fully coloured cell in the stack, if the correct materials are chosen, there should be no way to discern whether it

[e] In this context, auditing a ballot for A means following the ballot that gives a vote to A and checking that it corresponds to the receipt.

[f] For example, having a full rectangle and not an L-shape makes the folding more complicated, and introduces the problem of how to handle having translucent cells inside the instructions. As those cells could be coloured or not, the complexity of the ballot and the number of variables to check to prevent double-voting increases.

[g] To prevent problems between those two steps, the guillotine can be integrated with the ballot box.

is the second or the third layer that is coloured. Thus, finding whether the central ballot is for A or B should not be doable.

4) The receipt is a copy of the ballot that the voter chose to audit, with the same bar-code. As long as ballots with receipts aren't distinguishable from other ballots, if one is modified, there is a $\frac{1}{3}$ probability that the voter kept the corresponding receipt. In such a case, as they have a copy of the ballot with the same bar-code, they can prove that the ballot was modified.

3) and 5) The voter chooses whether they keep a receipt for A or B. However, because the green dot has to be visible, the X mark and the coloured cell right underneath have to correspond to the receipt and the left ballot.

Constraint number 6) is satisfied as there is no need for any device that could monitor or alter the vote, except potentially for the publication — which is partially independent of the vote — where it can be done in parallel to publicly accessible ballots.

8.3.3 Design choices

Multiple design choices are relevant in this ballot, while some are of no importance. The first important one is the bar-code, which can be considered poorly usable as it is not human-readable. However, this is a feature in this context, as the bar-code is there to ensure three properties. The first is that every ballot should be unique (easily done with a bar-code). The second is that it should be easy to check that the one on the receipt and on the corresponding ballot are identical, which is done here by aligning them. Finally, it should be very hard for the voter to keep receipts for all three ballots. If the unique identifiers were human-readable and easy to remember or copy, it would be much easier to coerce the voter into keeping receipts for all three, for example, by writing them down discreetly[h]. Other kinds of unique identifiers, such as random visual patterns, could be used, as long as they verify those properties.

The green dot, on the other hand, can be changed, as long as there is one feature that ensures that the receipt and the left ballot are on the same side while not being present on the ballots that are cast in the end to prevent identifying which ballot has a receipt.

Unlike some versions of Three-Ballot, the voter does not choose which ballot to keep a receipt for, but instead has an imposed ballot with a receipt on which they vote however they want (there is a small difference analysed at the end of the chapter).

8.4 Taped ballot

8.4.1 Protocol

This is a variant of the previous ballot design which uses masking tape and string. Instead of colouring multiple translucent cells independently, which can lead to making some mistakes and spoiling one's ballot by colouring in the wrong pattern, the voter instead has to tear off two sets of masking tape, the pieces in each set being linked by some string as can be seen on Figure 8.2. Like with the other systems in the chapter, the strings also operate as a memory aid and a guide to understanding the system and performing the procedure reliably.

[h]Some humans can read bar-codes, but it is quite harder to coerce and train someone into reading one without error and then remembering the result than into simply writing down a number.

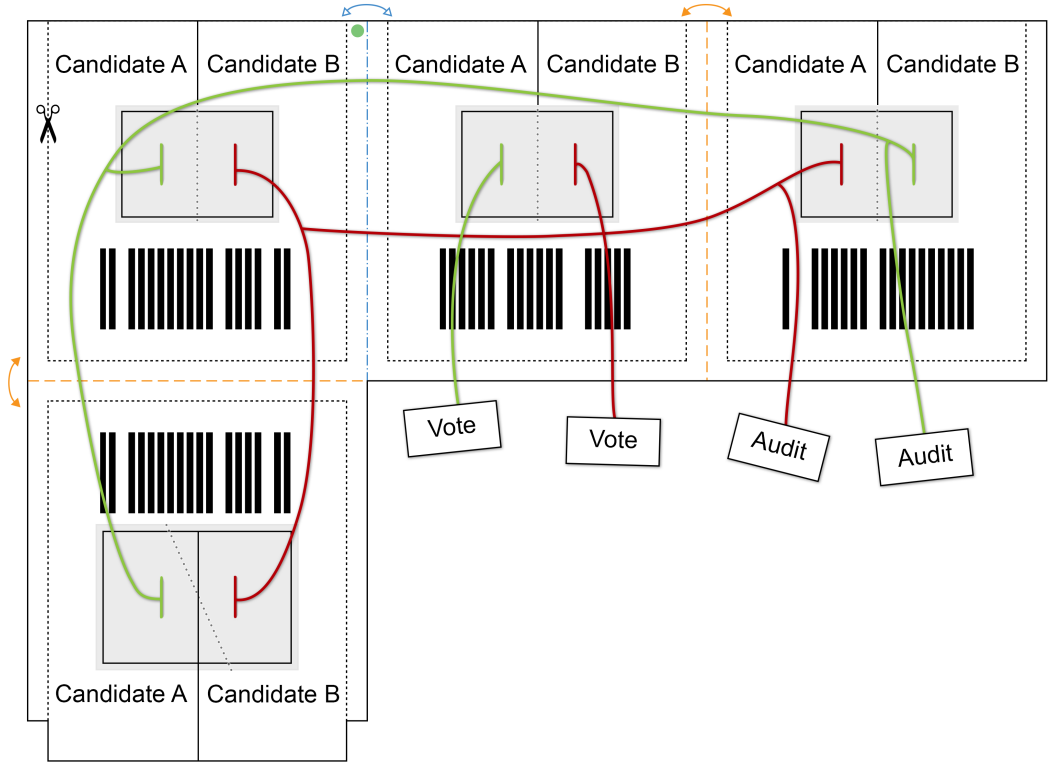

Figure 8.2: The taped ballot. Four strings are visible (in different colours here for ease of understanding), attached to different pieces of tape covering holes in the ballots. The voter picks one of the two audit strings and removes all corresponding tapes (by pulling the string), and does the same with one of the two voting strings before folding the ballot as in the previous protocol. As the holes in the receipt are bigger, it makes it easy to check that the receipt corresponds to the left ballot.

In this design, the translucent rectangles are replaced by rectangular holes in the ballot, covered by masking tape. The receipt has a slightly larger hole, with two strips of diagonal masking tape that shows both sides of the underlying rectangle when removed.

The instructions are simpler, as the voter has to make only two actions: choose and tear off the tape of their choice on the central ballot (corresponding to their vote), and choose and tear off the one they want to audit as well as the ones it is attached to.

8.4.2 Constraint satisfaction

When it comes to constraint 1), the official just has to make sure that, beneath the hole of the receipt, the left ballot only has the corresponding piece of tape removed, which is visible thanks to the fact that the tape covering the hole is not aligned with the tape underneath, being diagonal.

Constraint 2) is satisfied because the official can check that, on both sides of the ballot, a single piece of tape has been removed.

As this design is very similar to the previous one, it fulfils constraints 3), 4), 5) and 6) for the same reasons, but it also has different properties, analysed further down.

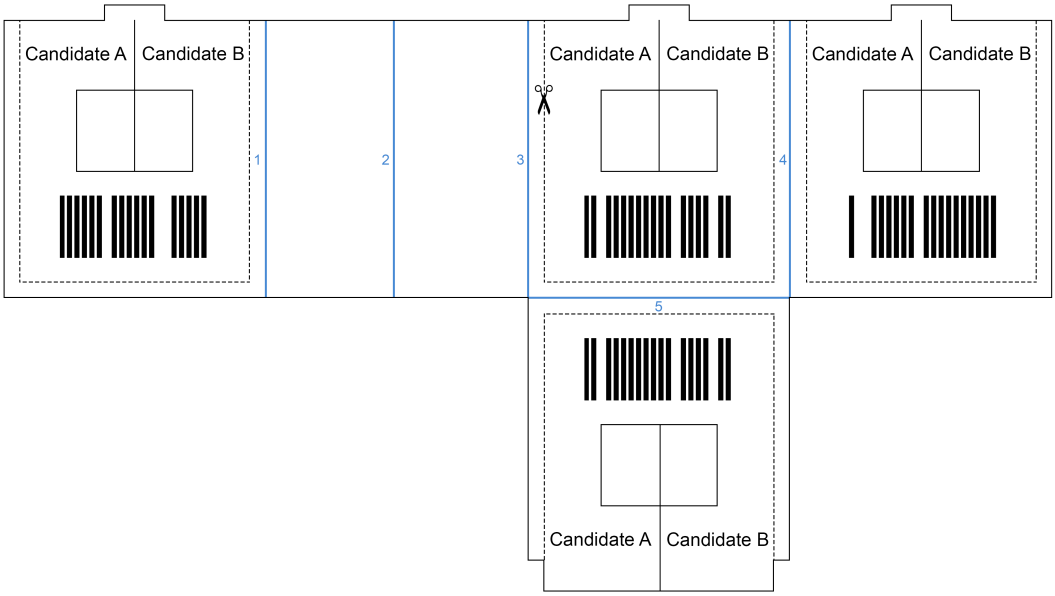

Figure 8.3: The punched ballot before being folded. The voter starts by folding lines 4 and 5 — both mountain folds — in the order of their choice, leaving the central ballot on top. They can then fold either 2 behind the central ballot — with mountain fold — or 1 behind and 3 on top of the central ballot — mountain and valley fold respectively. They then end up with a stack of ballots with the central rectangles aligned, the receipt sticking up at the top, and empty white paper on either the left or the right.

8.4.3 Design choices

The main goal of this design is to lower the probability of mechanical user error that comes from having a succession of actions to do in the previous design. The strings (which should be of a single colour, unlike in Figure 8.2) are but one method of linking together each set of masking tape. Once again, this seemingly non-optimal choice comes from the constraint of having all ballots indistinguishable when cast. Using alternatives like partially adhesive stickers or tear tape might make it simpler and more usable, but creating a tape pattern that links each set while keeping the ballots indistinguishable is a complex endeavour. Having symmetrical tape patterns on a recto-verso ballot is another option, but also decreases the usability. With this design, each ballot cast has a single piece of tape attached with a string that is cut at one end, not revealing whether it was a left ballot or not. It is important that the labels on the different strings be indistinguishable (Audit or Vote, instead of Audit A/Audit B). This is to ensure that they can hang outside the ballot during the cutting/casting process, preventing the ballots inside from being distinguishable while not allowing officials near the ballot box to check what the voter chose.

8.5 Punched ballot

8.5.1 Protocol

This last ballot stems from a different design, that seeks to reduce the user burden by making it simpler for the voter. In this case, the voter makes a single action to get their selection. In its simplest form, an already folded ballot is given to the voter who goes in a privacy booth. There, they can examine it — and unfold/refold it if wanted — before inserting it in a metal frame. They then come out of the booth where an official checks that the frame is correct, before punching a hole in the zone corresponding to the candidate of their choice. The ballots are then separated and cast by cutting them away as with the previous methods, while the voter keeps their receipt.

As in the other two proposed solutions, the ballot has three main parts and the receipt. By folding along the lines, the voter can align the three ballots in two different ways, such that two ballots are facing one way or the other. This means that, when they punch a hole, they give two votes to one candidate or the other.

If the ballot does not come pre-folded, the voter starts by doing the mandatory folding (which corresponds to folding, in turn, over line 5 and then line 4, each time leaving the central ballot on top). Two options are then possible. Either the left ballot will be facing the same direction as the central ballot, in which case punching A on this side results in two votes for A, or it will be facing the other direction, in which case, because of symmetry, punching A results in two votes for B. For the first option, the voter starts by folding line 3 over the central ballot, and then line 1 to leave the left ballot on top. For the second option, they simply need to fold line 2 below the central ballot.

Voting with a folded ballot means that there is an excess of paper on one side, which is to be hidden by the metal frame (to preserve the secrecy of on which side there is an excess of paper, which indicates which way the ballot is folded).

8.5.2 Constraint satisfaction

Constraint 1) depends on the voter not having the opportunity to unfold the ballot and punch holes independently on different parts of the unfolded ballot inside the privacy booth. As long as this is true, a single hole is punched, which, because of the folding, creates at least one ballot for A and one for B. If it is possible to unfold the ballot and fold it again differently (for example by folding not along the folding lines but elsewhere), it becomes necessary to check alignment with the metal frame. This can easily be done through the protruding bits at the top of the ballot.

Constraint 2) is satisfied as, once the ballot is folded and set into the frame, there is no way to know how the third ballot hidden inside is oriented, and the visible holes are always one for A and the other for B.

Constraint 5) is satisfied as the receipt corresponds, by the necessity of the folding, to the central ballot — which, unlike with the previous designs, is not the ballot corresponding to the voter's "true vote". Constraint 3) is satisfied because the voter can choose to fold one way or another, which, combined with their choice of vote, determines which hole is punched on the receipt. The other constraints are satisfied as with the previous two solutions.

8.6 Advantages and drawbacks of the solutions proposed

The translucent design has multiple advantages:

- The voter can easily choose which ballot to audit, as with the masked ballot.
- It allows concurrent elections by having multiple voting rectangles aligned vertically (present on the receipt in reverse order).
- It is quite familiar to many voters — or at least more so than the masked ballot.
- The correctness of the ballot can be checked by a voting official or a machine that simply measures the intensity of light reflected through the translucent rectangle.
- It is easy to fold it correctly.

It also has a few drawbacks:

- The folding instructions are non-trivial.
- While the voter isn't saddled with voting multiple times for a race, the folding confronts the voter with the complexity of the Three-Ballot system.
- It requires the officials to check for translucency.
- If the rectangle is big, it might be possible to identify the vote if they are not entirely coloured.

The masked ballot has similar features, but removes some of the complexity by leaving two choices: vote A or B, and audit A or B, and pull the corresponding strings. The drawbacks are that it requires more complex (and expensive) ballots, and cannot be extended to concurrent races.

Manufacturing issues for the masked ballot are nontrivial and could become a source of confusion and error if the adhesive or strings have any uncertainty. This approach is the most open to partially or completely unreadable ballots due to to problems such as hanging chads, as it depends on adhesives to work and strings not to be snagged incorrectly.

The punched ballot is — for the voter and the officials — the simplest of the three systems, requiring only one step to set up the ballot and one step to vote. It removes the direct choice of who to audit by making it dependent on the orientation of the frame. If it comes pre-folded, all there is to do is orient it carefully and punch the correct hole.

However, there are known problems with punched ballots [HS03, BHI02], and this system also requires a bit more equipment. It has one additional drawback, which is that it doesn't allow blank/null votes.

8.7 Attacks on the proposed systems

The main attacks against Three-Ballot concern either multiple races on a single ballot [HSS09, CKW08] or small numbers of voters [App06]. The first is avoided here by having a single race per ballot — as is already the case in a number of voting systems. The second is mostly a matter of choosing where to use this technology.

However, the designs shown here make certain new attacks possible. For example, in certain cases, the receipt is easy to see for a voting official. However, even knowing which voter has what kind of receipt does not allow an adversary to arbitrarily change votes, as they still have no information on which ballot belongs to whom. It can only inform them when a very small proportion of voters kept a receipt for candidate A, making the attack shown in [App06] a bit easier. This attack is especially relevant on the first two designs due to the green dot and the fact that the official is effectively checking whether the voter is auditing A or B. As they cannot simultaneously see the bar-codes, it is a limited flaw.

A much more significant attack depends on the printing of the ballots. As the voter does not choose which ballot (and hence which unique identifier) gets a receipt, knowing all the bar-codes on the left-side ballot gives an adversary knowledge over which bar-codes are safe to modify and which aren't. There is thankfully a fix: when picking the ballot, the official gives the voter three pairs of bar-codes on stickers. They then watch as the voter puts both stickers from one pair on the left ballot and the receipt, and one sticker from each of the two other pairs on the remaining ballots, before shredding the two stickers left. As the bar-codes are not human-readable, this method should be safe unless the process is systematically filmed with good cameras.

In parallel to this, to check that the printing process happened correctly, there should be the option of taking whatever ballot sheet is given to the voter and putting it in a pile to be audited (either by voter choice or randomly assigned), before giving them another ballot sheet. In the case of bar-code stickers, this should happen after they are pasted on the ballots. The discarded ballots can be checked publicly after the election to make sure that they weren't manipulated, and should, of course, be held securely in the meantime.

One more potential risk comes from the possibility of folding one ballot into another. This can be prevented in practice by using methods such as the tab present on the punched ballot, which allows an official to count the number of stacked ballots.

This brings us to what is a real vulnerability that is generally hard to address: it is possible to prove that one voted one way by filming the whole process, which is becoming increasingly relevant in the age of ballot selfies [Hor15]. There are once again solutions, as long as the voter — or the person spying on them — can't film continuously out of the privacy booth. The first is allowing users to get back to the ballot distribution table, spoiling their ballot, and start the whole process again (making what happened the first time in the privacy booth irrelevant). The second can be done with the third design, where only the folding and inserting of the ballot in the frame is done in the privacy booth. Once outside, the voter can easily flip the frame and vote differently.

8.8 Discussion

Cryptographic solutions to improve security typically come at a huge cost to usability, and sometimes even at the cost of accuracy. They often require careful encoding and multiple confusing actions. Moreover, most of the systems based on Three-Ballot left behind the initial paper-based advantage to use more involved electronic devices. With the systems proposed in this chapter, we sought to provide an alternative that requires no technology more complex than a hole puncher. The systems all have different properties, but they seek to make the inner workings of Three-Ballot more visible and understandable, to confront the voter and give them a better model of the process. Making the security part intuitive to voters has already been the focus of one previous voting system relying on cryptography, Selene [RRI16], as understanding how it works can increase both compliance and performance when dealing with secure systems [MMD14].

Additionally, all the designs shown can easily be made accessible to people with visual impairments by embossing the bar-codes and by adding Braille next to where the printed text is. In that case, additional auditing is needed to make sure that the text in Braille corresponds faithfully to the printed text.

Two main questions remain:

- How does one accommodate races with many different candidates while keeping usable simple ballots?
- What is the simplest way to handle many concurrent races?

For the former, the designs shown here can potentially be adapted to one or two more candidates, but one quickly gets to the geometric limits of paper folding. For the latter, the simplest solution is to make voters vote for each race independently, casting ballots and getting new ones repeatedly. There is also the possibility of having a long strip of ballots all attached to each other, but care has to be taken to prevent someone mixing and matching: parts of one ballot could be used to give a multiple-vote advantage on another race.

The exercise of designing such ballots is one way in which this chapter proposes to push opportunities for secure ballots forward, opening the possibility of further designs which explore more complex folding and geometrical patterns. The other is that this chapter presents actual usable ballot designs that could be deployed today to greatly increase the actual security and integrity of secret ballots for voters, although this is always a complex endeavour [BS18]. The designs shown could also be used as a stepping stone to familiarise voters with the concepts of verifiable voting, before deploying more complex systems. The original author of three ballot voting was sceptical about its practicability; this chapter, then, celebrates that Three-Ballot voting can be used by people in a simple and verifiable way.

Voting in the last two chapters was generally assumed to be in contexts common to many citizens of western countries, with a polling station handling a few hundred to a few thousand voters before aggregating the results on a regional or national scale. However, there are many different voting contexts, and the best solution in one setting can be over-engineered in another one. Going away from large election, the next chapter will then focus on a very specific setting called the boardroom voting problem.

9

Low-Tech Boardroom Voting

9.1 Issue statement and contributions

The previous chapter gave a solution for votes that requires some specially designed ballots and specialised equipment, which is compensated by the fact that it is targeted at large elections. However, most of our interactions with voting do not happen in the context of large elections, but instead in rooms where all voters are present. A board of directors might vote whether to adopt a new corporate policy, a committee of professors might vote whether to grant tenure to a colleague, or commercial decisions might be taken at a meeting of shareholders.

This kind of election has been organised and studied for centuries, with a good example being the papal conclave, where cardinals of the Catholic church meet to elect the pope. The rules governing the conclave are still mostly based on the papal bull[a] *Ubi Periculum* [Pop74], a text written in 1274 and made into canon law in 1298 [CM98]. Although it describes with great detail the way the electors should interact with the outside world, and establishes a necessity of a super-majority of two-thirds for a winner to be elected, it makes no mention of how the vote itself is to happen. Successive papal bulls have partially changed the rules in many ways, such as by increasing the number of electors over time. More recent rulings also forbid the presence of any audio-visual recording equipment [Pop96], and establish some formal rules for the election, such as the format — secret ballots, with explicit constraints on their size and design — and the chain of custody. However, the rules never properly consider the issues of privacy and verifiability in the presence of a skilled adversary.

Besides the papal election, a common occurrence of boardroom voting is in the context of shareholder meetings to decide on commercial strategies, where intimidation and fraud are known to be frequent [Bar09]. A few solutions have been proposed in the past fifteen years, always based on electronic means, whether they need smartphones [vB14], block-chains [MSH17], authenticated communication channels [Gro04], or even insecure devices [ACW13]. Despite these advances — and maybe because they all require some

This chapter is based on research done with Alan Sherman, Enis Golaszewski and Ted Selker.

[a] A papal bull is a formal decree emitted by the Catholic pontiff.

technical savvy — many votes today still happen by writing a name on a piece of paper, folding it and putting it in a hat, which has many issues both *privacy*- and *integrity*-wise.

Contributions. This chapter takes a look at boardroom voting, with three objectives:

- Define the problem of boardroom voting, with the general assumptions and adversarial model.
- Give a large set of primitives that can be used as building blocks to create secure protocols for boardroom voting, while not requiring any electronic devices. 8 of the primitives considered either come from other voting systems or are part of the folklore, while 7 haven't been studied or used for voting, to the best of our knowledge.
- Propose one protocol that uses those primitives to obtain verifiable voting while staying usable.

9.2 Definitions and adversarial model

A *boardroom election* is an election that takes place in a single room, which we shall call the *boardroom*. A crucial property of such elections is that all voters can see and hear each other. While there is no rigid maximum number of voters, we imagine a typical boardroom election to involve approximately four to forty voters. The election is administered by an untrusted voter or their untrusted assistants, also present in the room, which we shall call the Election Authority (EA). The election begins and ends in the boardroom. The process might be supported by some materials, such as paper ballots, marking devices, tape, stamps, and other objects that can be readily acquired in advance.

We seek solutions that are simple and practical, afford ballot privacy, and provide outcome integrity verifiable by the voters present. In particular, we seek solutions that do not require the use of complex technology, such as laptops or sophisticated cryptographic software. These requirements do not exclude the use of cryptography, but require that any cryptography be carried out in a "low-tech" fashion (e.g. implementing a cryptographic commitment by covering a character string with black tape).

The system should satisfy the security requirements of *ballot privacy* and *outcome integrity*. *Privacy*, in that no one should have the ability to link a marked ballot to the voter who cast it, not even with the cooperation of corrupt voters. Ballot privacy protects against undue influence, including vote selling and coercion. *Outcome integrity* means that the voters can verify that (1) they cast their ballot as intended, (2) the ballots were collected as cast, and (3) the ballots were counted as collected. We distinguish between two types of outcome verifiability: *Weak verifiability* means that a voter can convince themselves if outcome integrity is violated. *Strong verifiability* means that the voter can additionally convince others of such malfeasance.

Ideally, the system should resist delay and disruption, and it should not be possible for a corrupt voter to convince other voters with a false claim of malfeasance (that is, the system should resist discreditation attacks).

9.2.1 Assumptions

Boardroom. We assume that the boardroom has sufficient size, light, and acoustics that the voters can all be present in the room, see each other, and hear each other. Cameras

and electronic devices — including cell phones — are in general not permitted in the room, and we assume that none are hidden or otherwise present in the room. Similarly, we assume that it is not possible to peer into the room from outside, for example, using a telescope aimed through a window.

The situation, however, is crowded and cosy enough that voter can see what nearby voters are doing or writing at their seat. There can be a place in the room that offers privacy — for example, by using a privacy screen — where voters can go, one at a time, to carry out certain voting steps.

The only people present in the room are the *voters* and, possibly, a small number of people acting as the election authority. Neither the voters nor the election authority are trusted. For example, some voters may wish to sell their votes, and some may wish to maliciously discredit an outcome they dislike.

During the election, communications among people in the room outside of those required for the election procedure are not allowed. We acknowledge, however, that it would be impossible to stop all such communications completely, possibly including ones sent through covert channels (e.g. hand gestures). We assume that such illicit communications are either detected or have a limited bandwidth.

Adversary. The adversary can have many possible goals, such as the following:

- Influence the result of the election.
- Delay or even prevent the election.
- Find out how certain voters voted.
- Frame a specific voter for trying to disrupt the election.

The adversary might be a voter or member of the election authority, and there could be multiple adversaries at the same time, either acting in concert or each for a different — and potentially opposed — goal. In any case, the adversaries have complete knowledge of the election system and all procedures.

To achieve their goals, the adversary has access to financial means and can be assumed to have copies of the materials used in the election — insofar as they are not unique. They can also try to bribe or coerce one or more of the voters. As they are in the boardroom setting, they can also look over other people's shoulders and look at what they write.

Finally, at least to some limited extent, the adversary is considered capable of executing certain sleight-of-hand activities. This can be by dropping two ballots into a ballot box instead of one without being detected or being able to make a ballot disappear (into their sleeve, for example). This can affect the distribution or collection of physical materials unless additional protections are enforced.

Deterrence. In the context of boardroom voting, deterrence can play a particularly important role for two reasons. First, the adversaries are assumed to be in the room — as there should be no contact with the external world — so any indication that someone has malicious intentions can be costly. This is why, unlike with electronic attacks that could be hard to trace, failed attacks in this context have a very real cost[b]. This is linked to a

[b]The cost could be to reputation, with doubt lingering on potential suspects, or the attack could lead to an investigation wih unknown outcome.

second argument: it is possible to have back-up voting systems. The ideal system would be resistant to all attacks, fast, and easy to use. More realistically, secure systems tend to be less usable and slower, which isn't a problem by itself. As long as there is a protocol in place establishing that any attack detected leads to using a more secure system[c], the delay from a detected attack can be controlled. As long as attacks are noticeable, the worst they can do is delay the procedure somewhat. The existence of a back-up system can then allow the use of faster and more usable alternatives that only need to detect attacks, instead of preventing or correcting them.

This kind of reasoning becomes most useful when it comes to verifiable protocols, and to protocols in which a voter can detect but not prove that there was an attack. An adversary could claim an attack when there is none. Even in systems where one can prove the presence of an attack, adversaries obtain one more avenue of attack, by having the possibility to manipulate their own ballot (or receipt) in a way that allows them to fake the attack. Protocols that can gauge the scope of an attack become very useful in such a situation, and simple methods can address this problem. For example, there could be a threshold for the number of voters needed to annul and redo an election. A simple threshold would be for the margin of victory to be at least three times the number of reported altered ballots[d], multiplied by the reciprocal of the probability of finding out that a ballot was altered. If each voter has probability $\frac{1}{3}$ of finding out that their ballot was altered (as in Three-Ballot if everything is done correctly), then a margin of victory of 9 votes can be challenged if at least two voters report errors, and one reported error is enough to challenge any smaller margin[e].

With the goals clarified, let's start looking at the building blocks we can use to build low-tech secure protocols.

9.3 Building Blocks

The goal in this section is to give a list of primitives that can be used as building blocks for physical protocols. The list, although far from exhaustive, includes primitives that are not immediately useful. The protocols in the next section will be based on a combination of the blocks shown here, but we hope that the list could give rise to other protocols that optimise for different properties. Each primitive brings a property, such as resistance to cameras or the possibility of secure commitment, and comes with a cost, often concerning usability.

9.3.1 Overview of building blocks

We give a quick overview of all the building blocks before going over each in detail in the following subsections. The following list is all that is needed to understand the

[c]Repeated votes on the same subject can have different outcomes, which can give more motivation to an adversary. The effects of multiple sequential votes in case an attack is detected is however beyond the purview of this work.

[d]Twice the number of altered ballots wouldn't be enough, as each altered ballot can add or remove two from the margin.

[e]Because of the number of voters in a boardroom election, systems where the probability of detecting an error is close to 1 have a great advantage, as a margin of 4 is enough to prevent a single falsely reported attack from delaying the process.

protocols in Section 9.4. We start with nine primitives that are known and used in various voting systems:

- Private voting areas, such as booths, which give privacy both to voters and potential adversaries.
- Locked boxes, which prevent sleight-of-hand and can also be used for proof of authorship.
- Random drawing methods, such as taking an item from a bag, which can create a random permutation between a set of voters and a set of items.
- Cut-and-choose, which audits only some of the items at the voter's choice, statistically guaranteeing the integrity of other items.
- Opaque coverings, such as scratch-off or masking tape, which can hide information and serve for commitment.
- Visual cryptography, where two translucent unreadable images reveal a secret when superposed, which can be used for secret sharing.
- Tear-off systems, as in Prêt à Voter, which can be used to split secrets and for verifiable voting.
- Invisible ink of various kinds which can be used for commitment or for manipulating ballots in plain sight without compromising neither the chain of evidence nor the privacy of voters.
- Blind signatures, which allow trustees to verify the authenticity of a document without knowing its contents.

Besides those first nine primitives, we can mention seven additional ones which, to the best of our knowledge, haven't been used in voting systems.

- Scales under ballot boxes, to prevent some sleight-of-hand attacks by enforcing that every voter casts exactly one ballot.
- Polarising filters, which can prevent the use of most hidden cameras.
- Folded ballots, inspired by Prêt à Voter, where the candidate list is folded to make it temporarily invisible. The ballots can be voted on in plain sight as long as the folding is done in private, to give voters privacy while keeping the ballots in plain sight.
- Candidate wheels, which extend the previous ballots for multiple candidates while making multiple audits possible.
- Parallel election ballots, also a derivative of folded ballots. They can be used to force voters to mark two ballots in the exact same way without knowing how they voted.
- Visual secrets, which are random visual patterns that can be printed on ballots (or other supports). They allow a voter to recognise the pattern later and identify their ballot, while being hard to describe, making them impossible to use for coercion.

Readers can now read the detailed explanation of each building block or instead skip to the voting protocols in Section 9.4.

9.3.2 Private voting areas

In a boardroom election, all voters vote in the same room and can observe each other to gather information on voting decisions. To protect voters from malicious observation, the voting area can include private spaces established through means such as opaque panels, curtains, booths, or other visually obfuscating elements, as is already used in many countries to guarantee the voter's privacy while they fill in their ballot. An adversary unable to observe voters casting their votes must rely on potentially more costly, risky, or conspicuous methods. Attempts to encroach on other voters' voting areas should be detected with a high probability, either by an honest voter or by a third party monitoring the election. Private voting areas complicate attacks relying on passive observation while providing voters with some assurance that nobody is watching them cast their vote. However, they can also facilitate more advanced attacks by masking malicious activity, and they are potentially vulnerable to hidden cameras, especially if the adversary can be the one setting up the voting area. They should then carefully balance protecting voters' privacy from spying adversaries and shielding adversaries from the scrutiny of other voters.

In practice, private booths can be used in boardroom voting, although they have multiple issues: they take a significant amount of space, are vulnerable to cameras, but mostly induce a big cost time-wise, unless there are many booths present, and require people moving around, all of which lowers usability. An alternative is to use more portable voting areas, where voters are still partially observable while casting their ballots. In this case, the goal is to offer private manipulation of an item. The simplest case in our context is putting the ballot under the table and writing a name without being able to see the ballot before folding it (and, most importantly, without one's neighbours being able to see what we're writing). A slightly more involved but more secure means is to have a sheet of fabric on the table, under which the voter can manipulate, sign and fold their ballot. Similar systems are also used in conjunction with electronic voting in parliaments, where the hand of the voter is hidden in a box with three buttons inside, as was mentioned in Chapter 7.

9.3.3 Locked boxes

Small items, such as ballots, tokens, pens, or stamps ,often change hands in our context, creating opportunities for an adversary to steal or alter them. Identical small boxes with a lock on them can be a good way to ensure the integrity of items either as they are changing hands or for a certain duration. There is quite a bit of leeway on how to use them, depending on the type of lock. For example, they can easily solve *commitment problems* — and they are generally the baseline example used when designing electronic commitment schemes [Nao91]. A first way to use them is for an agent to put their lock on the box and give them to the voter without giving the key. Once the ballot is put inside the box, the voter closes the lock, and no-one can modify the ballot before it gets back to the agent unless they manage to get the key. Multiple keys can also be used at the same time on a given box. Alternatively, different items (such as empty unique ballots) can be placed in a set of locked boxes, with a key for each box. Then, either the boxes or the keys can be shuffled, before one is attributed to each voter. Although they provide very useful security features, locked boxes add a non-negligible level of complexity as well as a time cost.

9.3.4 Random drawing methods

Many secure voting schemes require the generation of random permutations. This can be done quite easily physically, and examples abound, such as drawing random items from a bag, common in board-games (as in Scrabble where one draws letters). This can be profitably used in boardroom voting, to generate a random permutation between a set of voters and any set of items — such as ballots, papers with secret numbers written on them, coloured pens, or any other small tool. One must be careful to ensure that the items are all identical to the touch with no removable identifiers (such as a sticky piece of paper attached on one side that can be discreetly removed when the item is taken from the bag). To alleviate this problem, locked boxes can be used, and both can be easily combined. To get stronger guarantees on the random draw when one uses locked boxes, there can be a succession of shuffles performed by different agents, with all the boxes being visible between shuffles. Although drawing items from a bag has a negligible cost (in time, material and complexity), this latter more secure method is more time-intensive.

9.3.5 Cut-and-choose

Cut-and-choose is a mainstay of auditing procedure, and corresponds to making duplicates of required items, then drawing some (either at random or chosen by an auditor) and examining them thoroughly in public to ensure that they are not maliciously altered. By taking a few items at random, one can make sure that, if a large proportion of all items are deficient, this will be detected and the vote will be stopped. It can also be used to reveal part of a secret which is split into multiple sections, as has been done in David Chaum's protocol for electronic cash [Cha83]. The main drawback of that kind of method is that it can be quite confusing, and requires more materials (more ballots or tools so that some can be removed and publicly examined). The auditing process also adds a time cost.

9.3.6 Removable opaque coverings

A second way to implement low-tech *commitment* is to use removable opaque coverings. They can come in a few different formats, depending on the properties required. For example, one can use simple masking tape, to prevent others from seeing what one wrote until after a certain time. A simple scheme based on this idea is for every voter to iteratively write who they vote for, cover the name with masking tape, and sign their name somewhere at the bottom of the sheet — not covered by masking tape — before giving it to the next voter. Once everyone is done, there is a single sheet with everyone's vote, which an external auditor can then tally without knowing individual votes[f]. There are also alternatives to masking tape, such as scratch-off systems [KZ10], which require more manufacturing but cannot be put back in their initial state once someone reveals the secret for the first time. Partially see-through envelopes — with a transparent window at a given location within an opaque body can also be of use.

Such covering methods can work especially well when combined with random auditing systems like Cut-and-choose. For example, in certain schemes, two scratch-off cards can be used, with the voter auditing one at random to check that it is correctly made, and using the second [ZCC+13].

[f] This simple scheme has vulnerabilities, such as the possibility for collusion between the auditor and an adversary inside the room to compare information and obtain individual votes, and is just an example.

9.3.7 Visual cryptography

Visual cryptography is a way to distribute secrets and perform secret sharing, first introduced by Naor and Shamir in 1995 [NS95], and widely extended afterwards [NY02]. It works by creating two images on transparent sheets of paper, such that the overlap creates readable information. By setting one image to be composed of random pixels — as in Figure 9.1, — this allows the separation of the initial secret in two physical parts which only reveal information when together. This can be used to separate initial secrets into halves to share with different voters or auditors, but it requires the printing of such secrets in advance.

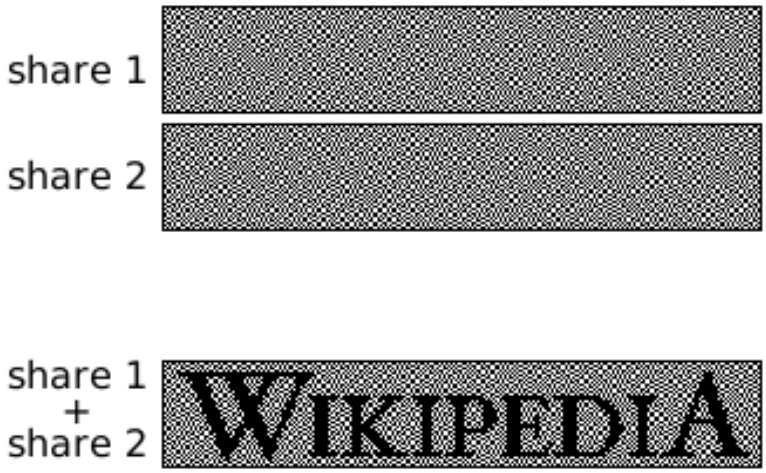

Figure 9.1: An example of visual cryptography, where the superposition of both top strips creates the bottom strip. Public domain image, courtesy of Wikimedia Commons user Blokhead.

9.3.8 Prêt à Voter-style tearing off

To obtain verifiability and preserve privacy, it can be useful to split in half the ballots used — or the information sheets distributed to voters. This can, for example, be done as in the Prêt à Voter system, with the candidate order shown on the side and being torn off and discarded. The Prêt à Voter system requires some electronic components, so it cannot be used directly in our setting. That said, similar methods can also be used to split secrets in half, and can be used in many protocols. It has negligible costs, although it can lower usability depending on how much there is to tear off. It can also introduce new attacks if the tearing off process creates identifiable edges.

9.3.9 Invisible ink

Invisible ink can be used to strengthen ballot confidentiality in multiple ways and has been used in some voting systems, such as Scantegrity [CCC+08, CEC+08]. We define invisible ink as any ink that is not visible to the human eye without the use of special tools or chemical reactions. We also consider invisible ink that automatically becomes visible after a specified period of time, or that disappears after a certain time. The first interest of invisible ink is that it limits the risk of onlookers managing to figure out what a voter is writing while they are writing. A second feature is that it allows the resulting secret

to be kept in plain sight during the rest of the voting protocol, including during shuffles, reducing opportunities to alter the ballot. Usability-wise, this has little cost to the voters, but requires more advanced manufacturing and increases costs.

9.3.10 Blind signatures

Blind signatures support the aims of ballot confidentiality and ballot verifiability. David Chaum described a physical implementation of blind signatures through the use of envelopes with carbo-paper linings, allowing a trustee's signature to transfer to a contained document without opening the envelope [Cha83]. Voters begin by marking their ballots and slipping them into carbon-paper-lined envelopes, turning them over to one or more trustees. The trustees verify the identity and the eligibility of the voters as well as the integrity of the envelopes, applying a certifying signature to the envelopes if all seems well. All certified envelopes go into a container which a trustee then shuffles such that an adversary cannot easily correlate any envelope to a voter. Once shuffled, a voter or trustee removes the envelopes from the hat and opens them, tallying the choices.

Opaque, carbon-paper-lined envelopes prevent honest-but-curious trustees from observing the ballots contained within envelopes. Since the envelopes are sealed, malicious trustees damaging the envelope seals will be detected with a high probability. However, they can potentially ruin certain ballots. The whole process also reduces usability as it requires multiple exchanges of envelopes, and necessitates some uncommon items (carbon-lined envelopes).

9.3.11 Scales

Putting the ballot box on a scale to measure its weight over time can prevent certain attacks by enforcing the fact that the ballot count increases by exactly one each time someone casts a vote[g]. Ideally, the ballot box rests on a scale away from the group of voters — the weight display being visible to everyone — with people going to cast their votes one at a time. At least one attack is prevented by this method, which works as follows. An adversary coerces one voter into not voting, faking the insertion of the ballot into the ballot box — quite easy with sleight-of-hand, especially if there is more than one person next to the ballot box. The adversary then votes twice by inserting two ballots instead of one, through sleight-of-hand. This requires having a second ballot ready, although that is often possible. By obtaining the missing ballot, the adversary can make sure that the total number of ballots is constant while preventing someone from voting how they want, and this even without direct contact between them and their target. This type of attack is preventable by forcing the ballots to remain in plain sight until they are inserted into a ballot box that shows the additional weight. The main drawback of this method is that it requires more equipment, and reduces usability as people have to move around to cast their ballot into the fixed ballot box.

Transparent ballot boxes address the same problem, although they mostly prevent someone from not voting at all — inattentive voters could be fooled by two envelopes being cast at the same time. They have the small inconvenience of making it potentially feasible to follow each envelope during the shuffling process, especially when there are few voters.

[g] An earlier version of this idea could have been used in Ancient Athens, where pottery bits called *ostraka* with votes recorded on them were visibly — and potentially audibly — dropped in tall urns [Can18].

9.3.12 Polarising filters

A second way to reinforce confidentiality is to use polarised light filters, either on ballots or on a screen used to distribute some common secret. The main advantage of this method is that it makes filming the boardroom with hidden cameras much harder, as polarised cameras tend to be bulkier or more expensive [Pru15], and applying a filter before-hand can fail as the adversary needs to know the direction of the polarisation. This method has fewer applications, as it is mostly useful when using a common screen, or small devices that react to polarised light. From a usability standpoint, it only requires polarised glasses, which can easily be found for less than 10€, doesn't really increase the time taken to vote, and slightly raises the complexity.

9.3.13 Folded paper ballots

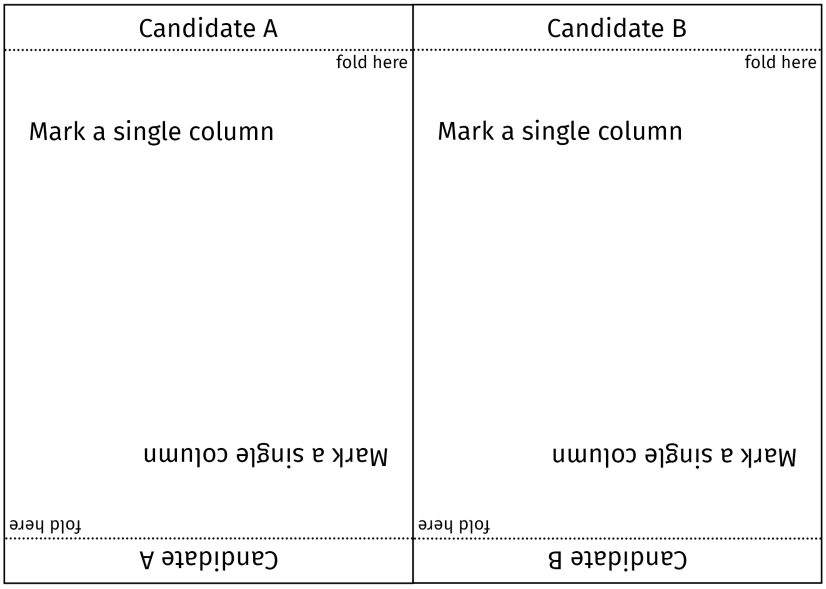

Figure 9.2: A folded paper ballot for a binary choice between two candidates. The voter obtains a ballot, checks that there are no identifying markers on it and makes a mental note of where each label is. They then fold the edge of the ballot on top of the label and rotate it such that only they know the position of the original labels. They can then put a mark on the zone corresponding to the candidate of their choice before casting the ballot, all in plain sight of other voters.

To protect ballot confidentiality, we propose folded paper ballots. For example, we can have a paper ballot consisting of two labelled columns where each column corresponds to a particular choice, which is labelled at both the top and bottom of the ballot, as in Figure 9.2. A voter wishing to mark a ballot makes a mental note of the labels for each column and folds both the top and bottom portions of the ballot down over the labels, then randomly rotates the ballot several times before applying their mark. When rotating their ballot, a voter should take measures to prevent the adversary from observing the number of rotations. One way to prevent an adversary from easily observing the number

of rotations is to rotate the ballot beneath a table or out of sight. An adversary that does not know the number of times a voter rotated a ballot cannot easily discern their choice by observation alone, as the ballot is symmetrical, the folds being present on both top and bottom. The folding can either be temporary or designed to resist some attacks by making part of the paper adhesive, preventing an adversary from unfolding the ballot and glancing at it discreetly. To make temporary folding more secure (and resistant against quickly unfolding and flashing the label at someone), the top and bottom parts can be folded twice.

A folded paper ballot consisting of two columns only supports two choices. To support additional choices, we suggest the use of polygonal paper ballots with folding edges, as on the right of Figure 9.3. A voter wishing to mark a polygonal paper ballot must fold each of the edges, apply a random number of rotations, and apply their mark to the sector corresponding to their choice. As the number of choices in an election increase, the polygonal paper ballot must provide an equal number of edges which may prove impractical when there are many choices, as voters might struggle to remember which sector corresponds to their choice. If that happens — no matter the number of candidates — the voter can either unfold the different labels[h] or request a different ballot if the folding is permanent.

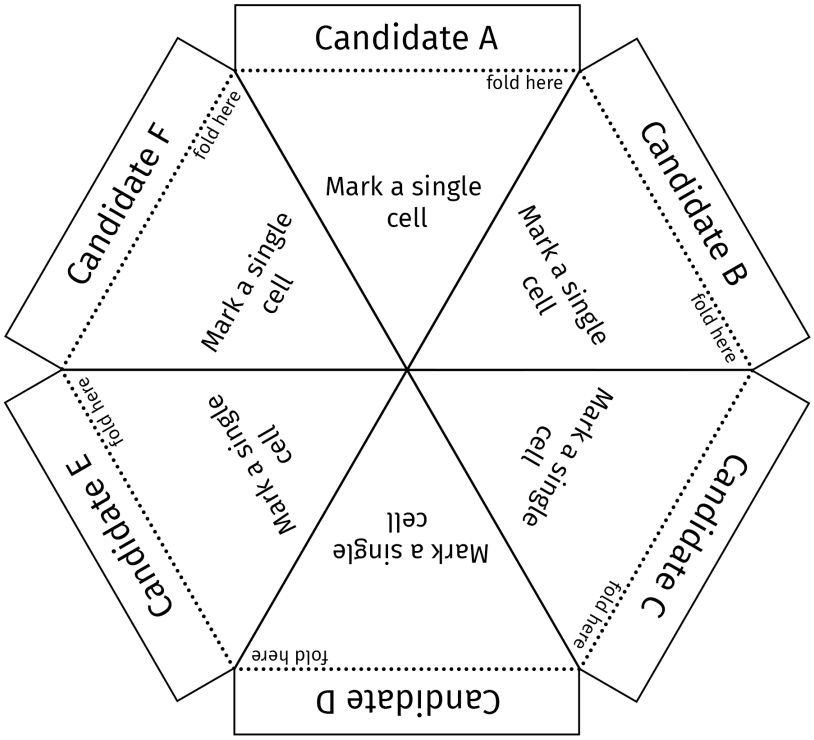

Figure 9.3: An example of folded paper ballot for a choice between six candidates. The procedure works as with the binary design: voters fold the edges of the ballot and rotate it, put a mark on it and cast the ballot, all in plain sight of other voters.

[h] Unfolding a single label could give away information to a watchful adversary, although it can also be used for misdirection.

While folded ballots defeat an adversary relying purely on observation, they do not protect ballots against other types of attacks. Any party able to gain access to ballots before or during distribution can mark or otherwise modify ballots to distinguish them later. Using marked ballots, an adversary can defeat both ballot confidentiality and ballot anonymity by correlating marked ballots to individual voters. To prevent correlation of a voter's mark to their ballot, all voters should make their marks using stamps, paper punches, or some other method that does not produce signature-like marks. Moreover, the initial distribution of ballots should be random, in case an adversary made a preliminary mark on the ballots (potentially with invisible ink), and voters should check that their ballot has no identifying marks.

9.3.14 Candidate wheels

An alternative to the polygonal design when there are many candidates is the paper wheel. Instead of having a sheet of paper with two possibilities side by side, voters get a continuous band, as in Figure 9.4. This allows them to rotate the wheel such that the candidate of their choice is in the position of their choice. They can then fold the labels, rotate the wheel while keeping track of their choice, and put their mark on it before casting the ballot. Voters can potentially fold the wheel in two places to get a flat double-sheet of paper, making it easier to write on it and cast it into a ballot box. This should be done after they fold the labels and rotate the wheel, however, to prevent an adversary from noticing where the creases are and where they made their mark to obtain how they voted from the order of the candidates. Putting the candidates in random order on the ballot can be useful in general[i], but is not sufficient by itself to prevent the attack.

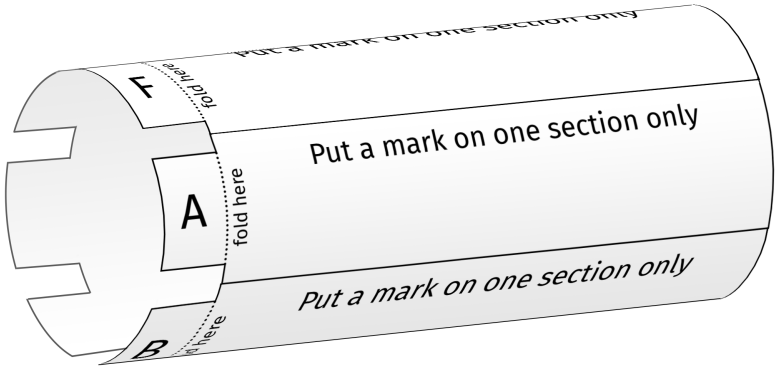

Figure 9.4: The candidate wheel, where the voter folds all the labels, rotates the wheel and puts a mark on the cell of their choice before flattening the ballot and casting it into the ballot box.

[i] A random order (and not just a cyclical shift) decreases both the effects of voter errors [Sle03] and position bias [Blu84].

9.3.15 Parallel elections

If voters suspect that someone could maliciously handle the ballots during the opening of the ballot box and the tallying phase, they might not want to trust the process to a single person. This is especially true when the supporters of different candidates are all distrustful of the other parties. To address this issue, one obvious solution is to hold multiple parallel elections on the same subject, where the guarantor of each ballot box is a different candidate, giving each a strong incentive to prevent attacks and making bribery of the person in charge of the ballot box more difficult. The problem with this solution is that it leaves open the possibilities of new attacks: an adversary could create problems in a single ballot box to get different tallies. This could sow discord, create additional delays or cast doubt on a target guarantor. When voters can detect that their vote was counted incorrectly but can't prove it, it also makes false accusations much easier.

The delaying/denial of service attack is relatively simple to solve, using the method explained in section 9.2.1: if the number of voters who announce that their votes were changed is such that it could change the outcome — potentially adding a margin for those who didn't notice — another vote should happen. As this second vote should be organised with a less usable but more secure system, it should not be in the interest of the adversary as it shows that someone is acting maliciously, without changing the end result[j]. The deterrent should thus be sufficient.

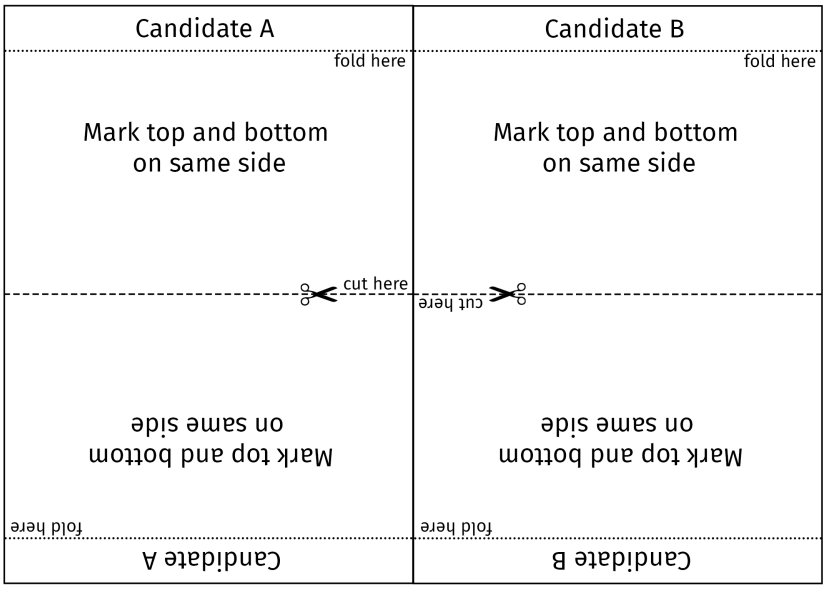

Figure 9.5: A ballot design for parallel elections that forces voters to vote for the same candidate in both elections. Voters have to make two marks on the same side, before cutting the ballot in two along the horizontal line and casting each in a different ballot box.

The second problem is related but more insidious, especially if voters can prove that their vote was altered. In this case, an adversary could vote differently in each election to

[j]The only gain for the adversary in this case is the delay caused by using a different system.

show a discrepancy. The problem then becomes: how to force voters to cast their different ballots for the same candidate? There are quite a few options, so we will mention two of the simplest. The first uses a ballot nearly identical to the binary folded ballot, except that the space where the voter is supposed to make a mark is split into two, vertically, as in Figure 9.5. Once the paper is folded, they make two marks on the same side, which can be checked by other people in the room, before cutting the ballot in half and casting each half in a different ballot box. This can be generalised partially by taking inspiration from the candidate wheel. One possibility is to have two candidate wheels pasted together side by side, with the same ordering of candidates. That is, each row of the wheel has four sections: label, voting space, voting space and label, with a vertical line separating the voting spaces all along the wheel. Once the labels are folded, the voters rotate the wheels, make two marks side by side and cut the wheels in half along the central vertical line, casting each half-wheel into a different ballot box. In this case, each half-wheel is essentially identical to the candidate wheel shown in Figure 9.4.

Another, less useful, possibility is to have a sheet of paper with four columns, in the same order as previously, except that the left side corresponds to a vote for the first candidate, and the right side to a vote for the other (with the labels indicating the left and right candidates on each row), as in Figure 9.6. Once they have folded the labels on the left and right parts and rotated the sheet, voters simply have to put their mark on all the spaces in the left column, or all the spaces in the right, while being watched by others. They can then cut apart each row and cast each one into a different ballot box.

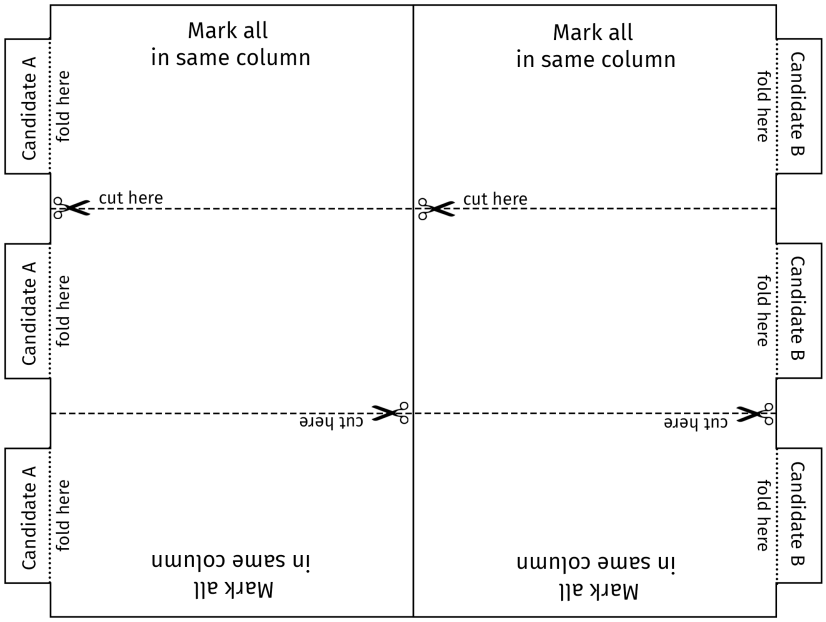

Figure 9.6: A ballot that allows parallel elections with three different ballot boxes. Voters fold the labels on each side, rotate the ballot as previously, put a mark on all cells in a column of their choice, cut the ballot in three can cast each third in a different ballot box.

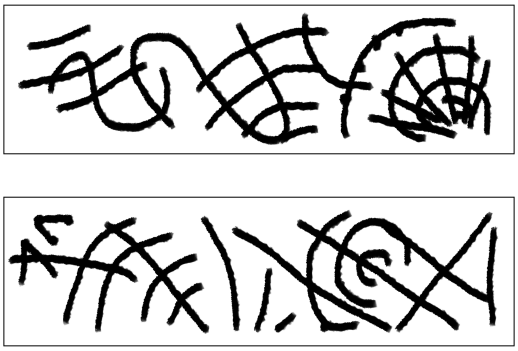

Figure 9.7: Two examples of visual secrets, easy to distinguish and remember but hard to describe orally. The patterns shown here are still relatively simple, and more complex ones could be used.

9.3.16 Visual secrets

To obtain verifiability in a boardroom context, one possibility is to have a secret that is present on the ballot given to the voter such as a secret string under a scratch-off protection, and all ballots are revealed publicly after the votes are all cast. As long as a coerced voter cannot communicate the secret to the adversary before the ballots are all revealed, they are safe, as they can tell the adversary that they voted according to any other revealed ballot, without the possibility to prove anything. There is one caveat: when the adversary coerces multiple voters who all happen to say they got the secret corresponding to the same ballot. In such a case, the adversary knows that at least some of them are lying[k]. However, if they can communicate their secret to an adversary before the public reveal, they can be held accountable to their votes, negating their privacy.

One solution is then to focus on secrets that are easily recognisable but hard to communicate. There is one simple way to do this, thanks to our visual pattern recognition. The idea is then to use a set of images built with similar patterns although visually different. For example, 30 different images of lions could be taken among a set of 1000, making it hard to describe any image with high precision succinctly. Alternatively, abstract patterns could also be used, as in Figure 9.7. One simple way of doing this is to give a sheet of stickers to each user, with many variations on a given design. This way they can select the design of their choice, remove it from the sheet under the table and apply it to their ballot[l].

One drawback of visual secrets is that some people might forget the pattern (or confuse it for another), but this is true for any pattern that needs to be remembered, and humans have excellent abilities for visual recognition — going above recognition for strings — especially with short-term memory [NS67].

[k]There is one potential fix for such situations that drastically reduces the probability of coerced voters saying they have the same secret, but it has a high usability cost and could potentially induce other security weaknesses. It works by agreeing to add a certain number of votes in favour of each candidate to the total, casting the corresponding ballots, and removing the corresponding number from the tally.

[l]This has two problems: first, the sheet with the remaining stickers can be kept, which creates a vulnerability. Second, the chosen sticker itself could be seen by an adversary during the operation, especially if the voter is coerced into cooperating.

9.3.17 Stamps

The visual secrets mentioned previously can be used in multiple ways, but the obvious one is to make a mark on a ballot that only the voter can remember, allowing them to track their ballot when it becomes public. The question is then how to distribute visual secrets securely, and how to apply them on a ballot in a way that is not immediately traceable by an adversary, as the solution shown above with stickers is vulnerable.

One solution is to use customised stamps, as in Figure 9.8. The stamps can be put in a bag, using a random draw method. To essentially eliminate the risk of an adversary seeing the pattern, two additional precautions should be taken. First, the stamp should use invisible ink, such that the visual secret is not visible to the voter's neighbours as soon as they apply the stamp. Second, it should be a stamp that rotates when pushed down — called self-inking stamps — making the pattern visible only when one presses on the stamp. This means that the voter can look at the pattern by pressing it in their hands before their eyes, but prevents neighbours from seeing the pattern due to the limited angle at which the pattern is visible. Moreover, it makes showing the pattern to an adversary much more conspicuous, reducing the interest in coercing the voter to do that. Care should be taken that the stamps are only used on the ballots and put back on the table afterwards, to prevent voters from keeping a proof of how they voted. Another possibility is to make the stamps freely available at the center of the table once they are used, so that coerced voters can fabricate fake evidence that they voted one way[m].

This method requires custom-made stamps, which is moderately costly. However, they can be re-used a few times, even more so if only a subset of the stamps is taken each time. It also adds some complexity as it requires two actions on the part of the voter (checking the pattern and stamping the ballot).

Figure 9.8: Type of stamp suited for this kind of practice, in which the pattern is hidden until the stamp is pressed. Image licensed from pngimg.com and Leila Gabasova.

[m] As long as they manage to stamp a few sheets of paper with different stamps, they have a high chance of getting one with the result they are supposed to have, and it would be hard for the adversary to confront them in the boardroom before the vote corresponding to each pattern is made public.

9.3.18 Summary of properties

The properties announced earlier — both advantages and drawbacks — are summed up in Table 9.1.

Method	Advantageous property	Drawbacks
Private voting areas	Prevents checking how voter votes	Gives privacy to adversary, lowers usability and speed
Locked boxes	Prevents sleight-of-hand, allows commitment	Lowers usability and speed
Random draw	Creates random permutation	Can be made non-random if adversary can alter items
Cut-and-choose	Enables auditing to ensure integrity of items	Increases complexity and costs in time and materials needed
Opaque coverings	Makes commitment possible	Increases complexity and cost of materials (if scratch-off)
Visual cryptography	Allows secret sharing	Requires preprinted sheets
Tear-off	Can be used to split secrets for verifiable voting	Lowers usability, can make identifiable receipts if used badly
Invisible ink	Commitment possible while remaining in plain sight	Increases complexity and cost of materials
Blind signatures	Allows verification by a trustee without disclosing result	Increases complexity and cost, requires multiple exchanges
Scale	Enforces the every voter votes exactly once	Small increase in complexity, lower speed
Polarising filters	Prevents the use of hidden cameras	Greatly increases complexity and cost, reduces usability slightly
Folded ballots	Gives voter privacy while voting in plain sight	Slightly decreases speed, can create errors, is best on binary races
Candidate wheel	Extends folded ballots for multiple candidates	Lowers usability, requires uncommon items (circular paper bands)
Parallel elections	Makes attacks on the ballot box harder	Increased complexity, time requirements and confusion
Visual secrets	Creates an identifying mark that cannot be communicated	Can lead to errors if people misremember
Stamps	Allows the use and printing of a visual secret in plain sight	Moderately costly, only partially reusable, slightly complex

Table 9.1: Table summarising the properties of all the building blocks mentioned. The methods in bold are the ones we proposed for which we couldn't find any previous reference of use in voting — suggested or applied.

9.4 Voting Protocols

We can combine some of the building blocks we proposed in the previous section to build a secure boardroom election. With foldable ballots, randomised stamps, random draws, invisible ink, and a ballot box on a scale, we can already create an initial protocol that has good properties for binary votes, in the context where voters are seated around a table.

9.4.1 Voting protocol

- Every voter receives a ballot with foldable top and bottom, split in half along the middle, with each side corresponding to a candidate. They fold the edges and rotate the ballot under the table until they are confident that only they know which side corresponds to each candidate.
- A set of randomised stamps is put in the middle of the table, such that everyone can observe them and see that they do not have any identifying external features. The stamps all feature a random abstract pattern, and they are filled with invisible ink that becomes visible after a few minutes.
- The stamps are put in a bag one by one under scrutiny of the voters, after which the bag is slightly shaken and passed around the table as each voter takes one stamp out of the bag.
- Voters check the pattern on their stamps and stamp their ballot on the side of their choice.
- One by one, voters cast their ballot into a ballot box on a scale at the center of the room.
- The ballot box is shaken and the ballots are all taken out, unfolded, and spread on a table until the ink appears.
- If any voter doesn't see their stamp or thinks their ballot has been altered, they manifest themselves.
- If the number of voters reporting a problem is less than half the margin of victory, the winner is elected. Otherwise, the election starts again using a more secure system.

Properties. The building blocks used ensure that the protocol has the following properties:

- Each voter can check that their vote was counted correctly, as long as they remember the pattern on the unique stamp.
- Adversaries can't know which side of another voter's ballot is for a given candidate, assuming that unfolding and flashing the label or looking under the table while they are rotating their ballots are both too conspicuous.
- Ballots are in plain sight from their distribution till they are shuffled in the ballot box, except for the moment when they are rotating under the table. This makes it much harder for an adversary to replaced a coerced person's ballot by an already filled one.

The user experience is also quite simple: a voter acquires a ballot, folds it and rotates it a few times under the table. They then take a stamp from a bag, look at it, stamp their ballot and cast it into a ballot box.

Drawbacks. This initial protocol also has a few issues:

- The protocol only works for binary votes, with voters seated around a table.
- There is a small risk that a voter could make a mistake on the number of rotations they performed, although the chance of making such mistakes
- Although voters can notice if their ballots have been altered, they cannot prove it. They can also make a mistake when checking their ballots — although the mistake can also be due to an eventual error they made when voting.
- The protocol requires specific kinds of invisible ink, that become visible after a short period of time, and consistently enough that there is no risk that a voter's ballot becomes readable before it is shuffled.
- Besides the material cost, using time-delay inks also introduces a delay after the vote, while people wait for the ink to become visible.
- A third problem with the kind of ink needed is that it should not dry (or become visible) when it is inside the stamp and in contact with air.
- Cameras could be used to record what is under the table — although the many different angles needed make it difficult.

There is also at least one attack against it, inspired by chain voting attacks [KRMC10], although it is limited in scope. An adversary could acquire a stamp, and discreetly stamp their own ballot as the ballots being manipulated under the table, before exchanging their ballots under the table with a coerced person next to them. They could then check the labels of their new ballot and vote on it. This allows each adversary to coerce exactly one voter.

9.4.2 Improvements and variants

A few potential improvements can address the issues mentioned, all being mutually compatible.

Preventing chain voting attacks. The attack described previously can only happen as long as the adversaries can exchange ballots, which can be solved by changing the private voting area. Instead of rotating their ballots under the table, voters could do it under a piece of fabric on the table.

Alternatives to appearing ink. All the problems with the ink mentioned above could be solved by using either chemically or thermally activated inks. This requires a bit more equipment during the tallying phase, but it could reduce overall costs by using simpler inks and reduce the delay.

Many candidates. The protocol only handles binary elections, but it can be extended to more candidates by using the polygonal ballots described in the subsection 9.3.13 on folded ballots.

Voting without seats In the case where the boardroom is mostly filled with people, without enough space for everyone to be seated at the same table, the protocol described above becomes unworkable. One possibility in such a case is to have a voting station with observers from different factions. The voting station can be as simple as a table with a stack of ballots, a bag of unused stamps, a piece of fabric under which one can rotate the ballot, and a ballot box. Having every person voting sequentially instead of in parallel slows down the process somewhat, although not by a large factor as there is only a small number of voters — and two or three voting stations can be organised.

9.4.3 Handling split factions

When the voters know that there is a high risk of attack during the vote, it can be hard to settle on an election authority, as each faction could cast doubts on the integrity of the other. This is especially true when it comes to handling the ballot box and the chain of evidence, with the risk of sleight-of-hand attacks.

If there are two main factions, we can use the primitive explained in subsection 9.3.15 on parallel elections. Specifically, two ballot boxes can be used. Instead of a supposedly impartial election authority handling the vote — who could potentially be bribed and whose probity would be questioned no matter the winner — two election authorities are used at the same time, each taking care of one ballot box.

In such a case, the deciding factor for annulling the result becomes the disagreement between the two votes. To prevent people from willingly introducing errors, we now have to force voters to cast identical ballots in both ballot boxes. The simplest way for this is to use the kind of ballot shown in Figure 9.5 (in case of binary votes), or the candidate wheel described. As the protocol calls for invisible ink, voters also have to stamp the same column (or row). This can be done by going through two rounds of stamping: first with a common stamp that simply puts a visible black disk, then with the unique stamp. Other people in the room can check that each voter stamps two black disks on the same side, and that people only stamp with invisible ink next to a black disk.

9.5 Discussion

Low-tech voting protocols that do not require any electronic means have been in use in boardroom contexts for seven centuries, although at an average rate of once every ten years for papal elections. That kind of constraint can, however, be applicable to other situations, such as meetings about sensitive subjects where votes must be held but all electronic items are forbidden to limit the risk of surveillance. One building block addresses some concerns about surveillance by making hidden cameras much harder to use but isn't applied in the protocol as it is an expensive and very specific counter-measure.

We started by describing the boardroom election problem, and have shown a list of fifteen low-tech primitives that can be used to organise such elections, seven of which haven't been used in this context to the best of our knowledge. Finally, we created a protocol that uses seven of the primitives mentioned, and improvements on it that use three other primitives.

The protocol's resistance to attacks comes mostly from the deterrence aspect: although it prevents a large range of attacks, its main feature is that it can detect some others. That kind of attack detection can be enough in the context of boardroom elections, where the

cost of the election itself is low, and back-up systems can be available. The simpler system can then be used in a lazy fashion, with a fall-back on the more secure system in case an attack is detected.

Despite these simplifications, the protocol proposed is still relatively complex, requiring some specialised equipment (mostly stamps with invisible ink). Unlike in the previous chapter, one issue with the designs shown here is that they are alas not adaptable to people with visual impairments, as they rely on the principle of visual secrets. It seems reasonable that some schemes could be made adaptable, however. Indeed, there are no doubts other low-tech primitives that could be formalised and used to create alternative protocols, hopefully ones with stronger properties.

The next step in this investigation would be to test the primitives mentioned and some protocols, along with some variants, in a real-world setting. The main open question left is whether there exists a verifiable voting system that gives a proof of attack and requires neither cryptography — as does Prêt-à-Voter — while having a high probability of attack detection — unlike Three-Ballot — as an attack on one or two ballots would already be dangerous in the context of boardroom voting.

This concludes the third and last part of this thesis, and it is time to look back at the different chapters to summarise the main open questions we mentioned, as well as the potential research directions that could improve our understanding of the subjects investigated in this work.

Moving forward

As we are done with the main body of this thesis, it is time to look back on the questions we asked, and look forward at the answers still missing. First, we have quite a list of open questions from the first two parts on authentication.

1 Open questions on authentication

Chapter 2 gave general results and a good candidate for a code structure that is easy to transcribe. It also raised a few questions when it comes to transcribing codes. Some are quite general, such as the effects on transcription accuracy and speed of different syllabic patterns, font choice, case, and colour. A cross-cultural study on transcribing ability with a wide variety of linguistic groups (with languages having alphabets, alphasyllabaries, abjads and logograms) would also be much welcome. Other questions are more specific, such as understanding the effect of chunking, with a variety of input interfaces (chunked input boxes for example), and seeing the difference with the chunked behaviours we observed.

There are few theoretical questions directly about Chapter 3, as the algorithms shown there already correct 91% of allowable typos. Its security guarantees could be improved or, alternatively, a new algorithm could potentially be developed that reaches similar correction rates with security features closer to [CWP+17]. It would also be interesting to extend the algorithm based on discrete logarithms, as the one shown has good properties but can only correct a small number of typos. The main future work in that direction rather lies in figuring how to implement those algorithms in practice. To get an idea of the work to do, we recently developed methods to check the prevalence of client-side hashing — already a good improvement over most of what is done, which has strong advantages, among which the ability for the client to check whether the hashing is correctly done. We showed that of the 50 biggest websites, only 8 implemented such security, corresponding exactly to the 8 websites based in the People's Republic of China. We also worked on automated method to check the presence of such features, and proposed different potential paths to web-wide acceptance.

Chapter 4, on the other hand, leaves open some questions of real importance. First, a second usability study is needed to ascertain the performance of the mental password manager in real-world conditions. Hopefully, the performances would be better than an-

nounced as the speed was still improving when the first study had to stop after four days. Second, a better framework to analyse the security of such algorithms would be useful, as the analyses so far have to assume a powerful adversary to get worst case bounds. Third and most importantly, looking for alternative algorithms seems a worthwhile pursuit, especially if some could breach the 15s barrier (in the average case), making them solid contenders in practice. On the other hand, lower bounds on the capabilities of human computation could disprove the possibility of that kind of performance, and such bounds could be obtained with a deeper analysis of the data shown in Chapter 6, or rather a follow-up study. As flaws have been shown in the original model, this new study would ideally allow the creation of an empirically-based model.

As for the passphrase generation method shown in Chapter 5, three main questions remain. First, how well do the results shown apply to long-term memory? Second, what is the optimal number of words to show, and from which dictionary should we take them? Third, is this method viable with other languages, specifically ones with fewer words?

2 New uses for voting tools

The voting methods shown in Chapter 8 and Chapter 9 have their interests, and other systems could probably be developed using similar primitives, to improve the usability and make them easier to understand. That said, there is a much bigger question that has received little attention so far: how does one use the new systems that are being developed? Nearly all of the political institutions in use today were theorised before the 20th century, when constraints were different, both technologically and geographically, with the urban population jumping from 15% to more than 50%.

The new methods developed today allow fundamentally different political systems, and although verifiable voting would be a nice addition to most democracies, it is not a radical departure from the systems that are currently used despite showing their limits in handling strong social tension. Upending traditional thought, recent rigorous research by Mirko Canevaro has shown that public consensus was effectively reachable in the Greek *poleis*, even in relatively large populations of more than ten thousand voters[n] [Can18]. With appropriate institutions and political culture, 95% assent on political decisions wasn't an anomaly but the norm. The much larger populations in contemporary states have been an obstacle to any form of direct democracy, but this could change with distributed decision systems. This requires both advanced deliberative systems — as a balanced public debate is often a requirement for perceived legitimacy — and voting systems, such as RSV, which can make many decisions in parallel while being representative of the general will when there is one.

Despite a lot of research and many technologies appearing on the subject, there's been a lag when it comes to modern institutions, and research in that direction could lead to very positive outcomes. A central obstacle in the way of implementing new institutions is

[n] Of course, the set of voters was more homogeneous than today, with women, metics, and non-citizens being deprived of the right to vote. However, society was still quite mixed when it came to class-based structure, with a Gini index between citizens of 0.71, comparable to 1950s USA [Obe11]. Moreover, the demographic importance of slavery in Athens has been frequently overstated. Although the voters only represented 12.5% of the total population, the biggest disenfranchised group wasn't the slaves (33%), but people of all origins who were too young to vote, making up 54% of the population [Obe11], as opposed to a quarter in today's Western countries.

that older and bigger democracies tend to suffer from heavy political inertia: one needs just recall the decades it took to adopt secret-ballots in many democracies after it was successfully used in Australia. Through a collaboration with the Humboldt Institute for Internet and Society, we are currently planning a larger-scale test of RSV coupled with an e-ID system in a German municipality of thirty thousand people, with the vote planned for the summer of 2019. That said, expanding the results of such demonstrations would be a tough fight. Instead of France, the USA or Germany, smaller and younger countries then seem to be better testing grounds. Estonia is known for its will to experiment on that front, but Taiwan has also been experimenting with a wide array of public technologies for debating and decision-making. Kosovo also seems to have the required public will to experiment and could be another test-bed for new democratic technologies. Focusing our efforts on getting positive results from these countries could establish the advantages provided by the inclusion of new democratic technologies, making easier future campaigns targeting larger states.

3 Using social behaviours for science

In this thesis, usability has been a concern and the main idea has been to adapt existing technologies or create new ones to make them accessible to most people. Let's try to come back to the start by doing the opposite: using usability to create technological tools that are not made for general users, but depend on general social behaviours to achieve their means instead. A very simple application of this idea comes from a cryptosystem that we're currently developing [BK19c], inspired by work by Aumann, Ding and Rabin [ADR02]. Their initial idea was to create everlasting security, in which a message that isn't decrypted in a short delay becomes impossible to decrypt, even with infinite computing power. The problem is that this requires a large source of public randomness, with a bandwidth that is far from reachable with existing public random beacons (such as the NIST one [FIP11]). As it happens, the number of potential sources is quite limited, and video streaming is the best bet. The problem with this idea is that most videos are shared in a many-to-one way: Youtube represents 11% of total Internet traffic, but only grows at the pace of 40GB/s, less than a thousandth of its traffic [Cul18, Cis18]. As it happens, there is one candidate that has the required bandwidth as well as many advantages from a socio-technical point of view: the pornographic live-streaming infrastructure, that is ever expanding in capacity and already accounts for a few percentage points of the total internet bandwidth. This cryptosystem has limited applications, as it only becomes relevant if most of the cryptosystems in use today become vulnerable. It is, however, just one example of using public behaviour for technical outcomes.

Semi-public data. A more critical research avenue that we are currently exploring is in the field of citizen science, specifically when it comes to *semi-public data*. Such data is present all over the Internet, and our prime example is the collection of movie ratings found on IMDb.com. The data there is collected from a variety of user contributions and aggregated, and has multiple anomalies, caused by inherent bias or sometimes targeted down-voting campaigns [Rey17]. We discovered by chance one particular feature that could potentially allow the automatic discovery of propaganda movies (specifically in the case of recent holocaust-denialism movies). The problem is that automatic extraction of this data is against the service provider's terms and conditions, and they do not make the data publicly available (even after a motivated request), although they sometimes share it with researchers close to the development team. There are many qualified researchers who

do not have access to that data through personal relations, or employers ready to defend them legally in cases of intellectual property infringement claims, and making the data available to them would allow — among other things — much stricter analysis of the bias we see in recommendation systems.

It turns out that many countries have different kinds of legal loopholes that would make this data available for researchers, with one big caveat: it requires crowdsourcing. The idea is quite simple: the data is split into a large set of elements (such as ratings for one particular movie), and a distributed set of users input them one by one. That kind of process would be protected in both the USA and the European Union under multiple kinds of legal reasonings[o]. This is where usability comes into place, as the interface issues are prominent with an unusual constraint: it must be entirely content blind for legal reasons. Of course, one could ask volunteers to copy and paste (or retype rather, for legal reasons) the content into text-boxes, but this is far from optimal and would deter many users. There is then a real problem of how to make it as easy as possible for users, and we are currently looking at multiple designs that satisfy both legal and usability constraints. Moreover, making it semi-competitive with rewards for finding specific patterns — such as the most biased movies — is also under consideration.

4 Final words

We have gone full circle over this thesis. What started as a small experiment to improve ballot usability gave us better code structures that are faster and easier to transcribe. This pushed us towards online authentication, whether it comes to creating better passphrases, mentally managing a large set of distinct passwords securely, or ignoring the typing mistakes in them from the server's side. This led us to some fundamental questions on human mental computation, of which we have but scratched the surface. We finally came back to the original ballots, spurred by the low usability of a design whose dependence on probabilities makes it highly counter-intuitive.

In all the results shown, the goal was to make the designs as simple as possible, both for the end-user and for the practitioner who implement and invariably modify them. One main question remains then: how to make sure they — or equivalently secure systems — are used in practice?

[o] In France for example, the database owner could potentially make claims under *sui generis* protection, but this requires having a registered office in the EU, or under contract law if they were to change their terms and conditions, but that would be extremely hard to do without making them equivalent to non-disclosure agreements, which could be considered void if used in the same way.

List of Figures

List of Tables

List of Algorithms

Author and General Bibliographies

As the research work presented in this thesis started about 18 months before the manuscript had to be sent, most of the results are still unpublished or under review. Here is a detailed breakdown of the work shown in each chapter, followed by the author's bibliography (by type), and the complete thesis bibliography.

1. Some of the ideas present at the end of Chapter 1 correspond to work submitted with Siargey Kachanovich, Ted Selker and Florentin Waligorski as [BKSW19].

2. Chapter 2 is based on work done with Leila Gabasova and Ted Selker, and has been accepted to be published as [BGS19].

3. Chapter 3 is based on work that is currently waiting for reviews [Bla19d].

4. Chapter 4 is based on work done with Ted Selker and Eli Sennesh, and the corresponding paper [BGSS19] is currently waiting for reviews. Some of it was published as [BGSS18].

5. Chapter 5 is based on work done with Clement Malaingre and Ted Selker, with additional help from Amira Lakhdar — an intern I supervised. The results in it were published as [BMS18b] and [BMS18a].

6. Chapter 6 is based on work done with Ted Selker and Florentin Waligorski which hasn't been submitted yet.

7. Chapter 7 is partially based on work partially done with Ted Selker, and incorporates ideas and passages from [BS18, Bla16, Bla18, Bla17].

8. Chapter 8 is based on work done with Ted Selker and some help from Leila Gabasova, which is currently being reviewed as [BS19].

9. Chapter 9 is based on work done with Alan Sherman, Enis Golaszewski and Ted Selker, which hasn't been submitted yet.

10. Some of the ideas mentioned in Moving forward have been published as [Bla19a] or submitted as [BK19c] and [BCS19].

Journal Papers (by the author)

[MRB14] J. A. Makowsky, E. V. Ravve, and N. K. Blanchard, "On the location of roots of graph polynomials," *European Journal of Combinatorics*, vol. 41, pp. 1–19, 2014.

[Bla18] N. K. Blanchard, "Building trust for sample voting," *International Journal of Decision Support System Technology*, 2018.

International peer-reviewed conferences (by the author)

[BS17] N. K. Blanchard and N. Schabanel, "Dynamic Sum-Radii Clustering," in *WALCOM: Algorithms and Computation: 11th International Conference and Workshops – WALCOM*, S.-H. Poon, M. S. Rahman, and H.-C. Yen, Eds. Springer International Publishing, 2017, pp. 30–41.

[BMS18a] N. K. Blanchard, C. Malaingre, and T. Selker, "Improving security and usability with guided word choice," *34th Annual Computer Security Applications Conference – ACSAC*, 2018.

[BS18] N. K. Blanchard and T. Selker, "Improving voting technology is hard: the trust-legitimacy-participation loop and related problems," in *Workshop on Socio-Technical Aspects in Security and Trust – STAST*, 2018.

[GBS+18] L. Gabasova, N. K. Blanchard, B. Schmitt, W. Grundy, and New Horizons COMP team, "Progressive metaheuristics for high-dimensional radiative transfer model inversion," *European Planetary Science Congress*, 2018.

[Bla19a] N. K. Blanchard, "CIVICS: Changing Incentives for Voters in International Cooperation through Sampling," in *Smolny Conference*, 2019.

[BGS19] N. K. Blanchard, L. Gabasova, and T. Selker, "Consonant-Vowel-Consonant for Error-Free Code Entry," in *HCI International Conference*, 2019.

[GBS+19] L. R. Gabasova, N. K. Blanchard, B. Schmitt, W. M. Grundy, C. B. Olkin, J. R. Spencer, L. A. Young, K. E. Smith, H. A. Weaver, A. Stern, and New Horizons COMP team, "Pluto surface composition from spectral model inversion with metaheuristics," 2019.

[BKSW19] N. K. Blanchard, S. Kachanovich, T. Selker, and F. Waligorski, "Reflexive memory authenticator: a proposal for effortless renewable biometrics," in *2nd International Workshop on Emerging Technologies for Authorization and Authentication*, 2019.

[BCS19] N. K. Blanchard, X. Coquand, and T. Selker, "Moving to client-sided hashing for online authentication," in *Workshop on Socio-Technical Aspects in Security and Trust – STAST*, 2019.

National peer-reviewed conferences (by the author)

[BS16] N. K. Blanchard and N. Schabanel, "Clustering Dynamique par Rayon ," in *ALGOTEL 2016 - 18èmes Rencontres Francophones sur les Aspects Algorithmiques des Télécommunications*, Bayonne, France, 2016. https://hal.archives-ouvertes.fr/hal-01304598

[Bla16] N. K. Blanchard, "Vote par sondage uniforme incorruptible," in *ALGOTEL 2016 - 18èmes Rencontres Francophones sur les Aspects Algorithmiques des Télécommunications*, Bayonne, France, 2016. https://hal.archives-ouvertes.fr/hal-01304599

[Bla17] N. K. Blanchard, "Building trust for sample voting," in *Proceedings of TeSS*, G. Camilleri, G. Cèze, F. Dupin de St-Cyr, and P. Zaraté, Eds., 2017.

[BMS18b] N. K. Blanchard, C. Malaingre, and T. Selker, "Mots de passe : le choix humain plus sécurisé que la génération aléatoire," in *ALGOTEL 2018 - 20èmes Rencontres Francophones sur les Aspects Algorithmiques des Télécommunications*, Roscoff, France, 5 2018. https://hal.archives-ouvertes.fr/hal-01781239

[BGSS18] N. K. Blanchard, L. Gabasova, T. Selker, and E. Sennesh, "Créer de tête de nombreux mots de passe inviolables et inoubliables," in *ALGOTEL 2018 - 20èmes Rencontres Francophones sur les Aspects Algorithmiques des Télécommunications*, Roscoff, France, 5 2018. https://hal.archives-ouvertes.fr/hal-01781238

[Bla19b] N. K. Blanchard, "Comment corriger efficacement les typos dans les mots de passe," in *ALGOTEL 2019 - 21èmes Rencontres Francophones sur les Aspects Algorithmiques des Télécommunications*, 2019.

Submitted papers (by the author)

[Bla19c] N. K. Blanchard, "La Démocratie Hasardeuse (book)," 2019.

[Bla19d] N. K. Blanchard, "Secure and efficient password typo tolerance," 2019.

[BFLR19] N. K. Blanchard, E. Fischer, O. Lachish, and F. Reidl, "Longest paths in 2-edge-connected cubic graphs," 2019.

[BK19b] N. K. Blanchard and S. Kachanovich, "A note on the inflating enclosing ball problem," 2019.

[BK19a] N. K. Blanchard and S. Kachanovich, "Counting authorised paths in constrained control-flow graphs," 2019.

[BK19c] N. K. Blanchard and S. Kachanovich, "Usable everlasting encryption using the pornography infrastructure," 2019.

[BGSS19] N. K. Blanchard, L. Gabasova, T. Selker, and E. Sennesh, "Cue-Pin-Select, a Secure and Usable Offline Password Scheme," 2019. https://hal.archives-ouvertes.fr/hal-01781231

[BS19] N. K. Blanchard and T. Selker, "Usable paper-based verifiable voting systems," 2019.

Complete list of references cited in this thesis

[RL19] N. Rotam and R. Locar, "Report: Data breach in biometric security platform affecting millions of users," vpnMentor, Tech. Rep., 2019. https://web.archive.org/web/20190815090536/https://www.vpnmentor.com/blog/report-biostar2-leak/

[Aba14] J. Abawajy, "User preference of cyber security awareness delivery methods," *Behaviour & Information Technology*, vol. 33, no. 3, pp. 237–248, 2014.

[ARH+13] A. Abaza, A. Ross, C. Hebert, M. A. F. Harrison, and M. S. Nixon, "A survey on ear biometrics," *ACM Computing Survey*, vol. 45, no. 2, pp. 22:1–22:35, Mar. 2013.

[ABF+17] Y. Acar, M. Backes, S. Fahl, D. Kim, M. L. Mazurek, and C. Stransky, "How internet resources might be helping you develop faster but less securely," *IEEE Security Privacy*, vol. 15, no. 2, pp. 50–60, March 2017.

[AFM16] Y. Acar, S. Fahl, and M. L. Mazurek, "You are not your developer, either: A research agenda for usable security and privacy research beyond end users," in *IEEE Cybersecurity Development – SecDev*, Nov 2016, pp. 3–8.

[AAB+17] A. Acquisti, I. Adjerid, R. Balebako, L. Brandimarte, L. F. Cranor, S. Komanduri, P. G. Leon, N. Sadeh, F. Schaub, M. Sleeper, Y. Wang, and S. Wilson, "Nudges for privacy and security: Understanding and assisting users choices online," *ACM Computing Survey*, vol. 50, no. 3, pp. 1–41, Aug. 2017.

[ABD+15] D. Adrian, K. Bhargavan, Z. Durumeric, P. Gaudry, M. Green, J. A. Halderman, N. Heninger, D. Springall, E. Thomé, L. Valenta, B. VanderSloot, E. Wustrow, S. Zanella-Béguelin, and P. Zimmermann, "Imperfect forward secrecy: How diffie-hellman fails in practice," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '15. New York, NY, USA: ACM, 2015, pp. 5–17.

[AM10] C. R. Adsett and Y. Marchand, "Syllabic complexity: A computational evaluation of nine european languages," *Journal of Quantitative Linguistics*, vol. 17, no. 4, pp. 269–290, 2010.

[AG15] B. S. Akkoca and M. Gokmen, "Automatic smile recognition from face images," *23nd Signal Processing and Communications Applications Conference – SIU*, pp. 1985–1988, 2015.

[AR16] N. Alkaldi and K. Renaud, "Why do people adopt, or reject, smartphone password managers?" in *Proceedings of EuroUSEC*, 2016, eprint on Enlighten: Publications.

[AAA+01] M. R. Alvarez, S. Ansolabehere, E. Antonsson, J. Bruck, S. Graves, T. Palfrey, R. Rivest, T. Selker, A. Slocum, and C. Stewart III, "Voting: what is, what could be," *Pasadena, CA: Caltech/MIT Voting Technology Project*, 2001.

[Ama18] "Amazon AWS S3 cost calculator," Amazon, 2018, accessed: 2017-12-18. https://calculator.s3.amazonaws.com/index.html

[And83] J. A. Anderson, "Cognitive and psychological computation with neural models," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-13, no. 5, pp. 799–815, 9 1983.

[ASI05] S. Ansolabehere and C. S. Stewart III, "Residual votes attributable to technology," *Journal of Politics*, vol. 67, no. 2, pp. 365–389, 2005.

[Ant18] Antelle. (2018) Argon2 in browser. https://web.archive.org/web/20180119222301/http://antelle.net/argon2-browser/

[AIA14] S. Aoyama, K. Ito, and T. Aoki, "A finger-knuckle-print recognition algorithm using phase-based local block matching," *Information Sciences*, vol. 268, pp. 53 – 64, 2014.

[App06] A. W. Appel, "How to defeat rivest's threeballot voting system," 2006. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.116.5156&rep=rep1&type=pdf

[ACW13] M. Arnaud, V. Cortier, and C. Wiedling, "Analysis of an electronic boardroom voting system," in *E-Voting and Identify*, J. Heather, S. Schneider, and V. Teague, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 109–126.

[ABTV11] C. Ashby, A. Bhatia, F. Tenore, and J. Vogelstein, "Low-cost electroencephalogram (eeg) based authentication," in *5th International IEEE/EMBS Conference on Neural Engineering – NER*. IEEE, 2011, pp. 442–445.

[Ash92] M. H. Ashcraft, "Cognitive arithmetic: A review of data and theory," *Cognition*, vol. 44, no. 1-2, pp. 75–106, 1992.

[Auf19] S. Auffret. (2019) En suisse, on encourage les hackers à pirater le système de vote électronique. https://web.archive.org/web/20190302022910/https://www.lemonde.fr/pixels/article/2019/02/25/les-hackers-invites-a-pirater-le-systeme-de-vote-electronique-suisse_5428208_4408996.html

[ADR02] Y. Aumann, Y. Z. Ding, and M. O. Rabin, "Everlasting security in the bounded storage model," *IEEE Transactions on Information Theory*, vol. 48, no. 6, pp. 1668–1680, 2002.

[AZM+13] A. Awad, H. Zawbaa, H. Mahmoud, E. Hassan, R. Fayed, and A. E. Hassanien, "A robust cattle identification scheme using muzzle print images," in *Federated Conference on Computer Science and Information Systems*, 01 2013, pp. 529–534.

[BH74] A. Baddeley and G. J. Hitch, *Working memory*. Academic Press, 1974, vol. 8, pp. 47–90.

[BL16] M. Balinski and R. Laraki, "Instaurons le «jugement majoritaire»," *Commentaire*, no. 2, pp. 413–415, 2016.

[Bar09] R. W. Barrett, "Elephant in the boardroom: Counting the vote in corporate elections," *Valparaiso University Law Review*, vol. 44, p. 125, 2009.

[BRS18] D. Basin, S. Radomirovic, and L. Schmid, "Alethea: A provably secure random sample voting protocol," in *IEEE 31st Computer Security Foundations Symposium – CSF*, July 2018, pp. 283–297.

[BMK+18] G. C. Batista, C. C. Miers, G. P. Koslovski, M. A. Pillon, N. M. Gonzalez, and M. A. Simplicio, "Using Externals IdPs on OpenStack: A Security Analysis of OpenID Connect, Facebook Connect, and OpenStack Authentication," in *IEEE 32nd International Conference on Advanced Information Networking and Applications – AINA*, vol. 00, 5 2018, pp. 920–927.

[BGI+13] A. Baujard, F. Gavrel, H. Igersheim, J.-F. Laslier, and I. Lebon, "Vote par approbation, vote par note," *Revue économique*, vol. 64, no. 2, pp. 345–356, 2013.

[BIL+14] A. Baujard, H. Igersheim, I. Lebon, F. Gavrel, and J.-F. Laslier, "Who's favored by evaluative voting? An experiment conducted during the 2012 french presidential election," *Electoral Studies*, vol. 34, pp. 131–145, 2014.

[Baw18] H. Baweja. (2018) Srinagar's 7% voter turnout means a rejection of everything 'indian'. https://web.archive.org/web/20180608104936/https://www.hindustantimes.com/opinion/opinion-srinagar-s-7-turnout-means-a-rejection-of-everything-indian/story-gJ38rJG8X61VwjQODvC8LN.html

[Bea14] E. Beaulieu, "From voter ID to party ID: How political parties affect perceptions of election fraud in the US," *Electoral Studies*, vol. 35, pp. 24–32, 2014.

[BKMF05] R. Bednarik, T. Kinnunen, A. Mihaila, and P. Fränti, "Eye-movements as a biometric," in *Image Analysis*, H. Kalviainen, J. Parkkinen, and A. Kaarna, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 780–789.

[Bee91] D. Beetham, *The Legitimation of Power*. Palgrave Macmillan, 1991.

[Bel87] F. S. Bellezza, *Mnemonic Devices and Memory Schemas*. New York, NY: Springer New York, 1987, pp. 34–55.

[BCD02] C. BenAbdelkader, R. Cutler, and L. S. Davis, "Person identification using automatic height and stride estimation," in *ICPR*, 2002.

[BDT03] J. Bendor, D. Diermeier, and M. Ting, "A behavioral model of turnout," *The American Political Science Review*, vol. 97, no. 2, pp. 261–280, 2003. http://www.jstor.org/stable/3118208

[vB14] P. von Bergen, "A mobile application for boardroom voting," Master's thesis, Bern University of Applied Sciences, Biel, Switzerland, 2014.

[BLM01] M. Bernard, C. H. Liao, and M. Mills, "The effects of font type and size on the legibility and reading time of online text by older adults," in *CHI '01 Extended Abstracts on Human Factors in Computing Systems*, ser. CHI EA '01. New York, NY, USA: ACM, 2001, pp. 175–176.

[BDPvA08] G. Bertoni, J. Daemen, M. Peeters, and G. van Assche, "On the indifferentiability of the sponge construction," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2008, pp. 181–197.

[BDK16] A. Biryukov, D. Dinu, and D. Khovratovich, "Argon2: new generation of memory-hard functions for password hashing and other applications," in *IEEE European Symposium on Security and Privacy – EuroS&P*. IEEE, 2016, pp. 292–302.

[Bis09] J. M. Bishop, "A cognitive computation fallacy? Cognition, computations and panpsychism," *Cognitive Computation*, vol. 1, no. 3, pp. 221–233, 2009.

[Bis19] S. Biswas. (2019) India election 2019: The mystery of 21 million missing women voters. https://web.archive.org/web/20190314015224/https://www.bbc.com/news/world-asia-india-47521208

[Bla05] P. E. Black, "Fisher-yates shuffle," *Dictionary of algorithms and data structures*, vol. 19, 2005.

[BC90] A. Blais and K. R. Carty, "Does proportional representation foster voter turnout?" *European Journal of Political Research*, vol. 18, no. 2, pp. 167–181, 1990.

[BBD13] J. Blocki, M. Blum, and A. Datta, "Naturally rehearsing passwords," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2013, pp. 361–380.

[BBDV17] J. Blocki, M. Blum, A. Datta, and S. Vempala, "Towards human computable passwords," *8th Innovations in Theoretical Computer Science Conference – ITCS 2017*, 2017.

[BZ16] C. Blum and C. I. Zuber, "Liquid democracy: Potentials, problems, and perspectives," *Journal of Political Philosophy*, vol. 24, no. 2, pp. 162–182, 2016.

[BV17] M. Blum and S. Vempala, "The complexity of human computation: A concrete model with an application to passwords," *CoRR*, vol. abs/1707.01204, 2017. http://arxiv.org/abs/1707.01204

[BV15] M. Blum and S. S. Vempala, "Publishable humanly usable secure password creation schemas." in *3rd AAAI Conference on Human Computation and Crowdsourcing*, 2015.

[Blu84] N. J. Blunch, "Position bias in multiple-choice questions," *Journal of Marketing Research*, pp. 216–220, 1984.

[BSR+12] H. Bojinov, D. Sanchez, P. Reber, D. Boneh, and P. Lincoln, "Neuroscience meets cryptography: Designing crypto primitives secure against rubber hose attacks," in *21st USENIX Security Symposium*. Bellevue, WA: USENIX, 2012, pp. 129–141. https://www.usenix.org/conference/usenixsecurity12/technical-sessions/presentation/bojinov

[Bon12] J. Bonneau, "The science of guessing: Analyzing an anonymized corpus of 70 million passwords," in *IEEE Symposium on Security and Privacy*, 5 2012, pp. 538–552.

[BHvOS12] J. Bonneau, C. Herley, P. C. van Oorschot, and F. Stajano, "The quest to replace passwords: A framework for comparative evaluation of web authentication schemes," in *IEEE Symposium on Security and Privacy*. IEEE, 2012, pp. 553–567.

[BS12] J. Bonneau and E. Shutova, "Linguistic properties of multi-word passphrases," in *International Conference on Financial Cryptography and Data Security*. Springer, 2012, pp. 1–12.

[BHSY00] S. E. Bonner, R. Hastie, G. B. Sprinkle, and M. S. Young, "A review of the effects of financial incentives on performance in laboratory tasks: Implications for management accounting," *Journal of Management Accounting Research*, vol. 12, no. 1, pp. 19–64, 2000.

[BA17] T. Booth and K. Andersson, "Stronger authentication for password credential internet services," in *3rd International Conference on Mobile and Secure Services – MobiSecServ*. IEEE, 2017, pp. 1–5.

[BBW08] H. Borgen, P. Bours, and S. D. Wolthusen, "Visible-spectrum biometric retina recognition," in *International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, Aug 2008, pp. 1056–1062.

[BRB12] C. Borghesi, J.-C. Raynal, and J.-P. Bouchaud, "Election turnout statistics in many countries: similarities, differences, and a diffusive field model for decision-making," *PloS one*, vol. 7, no. 5, 2012.

[BMLZ17] E. Borleffs, B. A. M. Maassen, H. Lyytinen, and F. Zwarts, "Measuring orthographic transparency and morphological-syllabic complexity in alphabetic orthographies: a narrative review," *Reading and Writing*, vol. 30, no. 8, pp. 1617–1638, 10 2017.

[Bor06] D. Borsboom, "The attack of the psychometricians," *Psychometrika*, vol. 71, no. 3, p. 425, 2006.

[Bou16a] P. Boucher, "What if blockchain technology revolutionised voting," *Unpublished manuscript, European Parliament*, 2016.

[Bou16b] A. Boult. (2016) Petition calling for second EU referendum was created by a Leave voter – and he's not happy that it's been hijacked by Remain. https://web.archive.org/web/20160702162755/https://www.telegraph.co.uk/news/2016/06/27/petition-calling-for-second-eu-referendum-was-created-by-a-leave/

[BBD01] S. Bowler, D. Brockington, and T. Donovan, "Election systems and voter turnout: Experiments in the united states," *Journal of Politics*, vol. 63, no. 3, pp. 902–915, 2001.

[BHF08] K. W. Bowyer, K. Hollingsworth, and P. J. Flynn, "Image understanding for iris biometrics: A survey," *Computer Vision and Image Understanding*, vol. 110, no. 2, pp. 281–307, May 2008.

[Boz99] S. Boztas, "Entropies, guessing, and cryptography," Department of Mathematics, Royal Melbourne Institute of Technology, Tech. Rep., 1999.

[BL15] M. M. Bradley and P. J. Lang, "Memory, emotion, and pupil diameter: Repetition of natural scenes," *Psychophysiology*, vol. 52, no. 9, pp. 1186–1193, 2015.

[BK88] G. Brassard and S. Kannan, "The generation of random permutations on the fly," *Information Processing Letters*, vol. 28, no. 4, pp. 207–212, Jul. 1988.

[BCT07] J. Bringer, H. Chabanne, and Q. Tang, "An application of the naccache-stern knapsack cryptosystem to biometric authentication," in *2007 IEEE Workshop on Automatic Identification Advanced Technologies*, June 2007, pp. 180–185.

[BS00] S. Brostoff and M. A. Sasse, "Are passfaces more usable than passwords? a field trial investigation," in *People and Computers XIV — Usability or Else!: Proceedings of HCI*, S. McDonald, Y. Waern, and G. Cockton, Eds. London: Springer London, 2000, pp. 405–424.

[BSMK16] M. Brysbaert, M. Stevens, P. Mandera, and E. Keuleers, "How many words do we know? practical estimates of vocabulary size dependent on word definition, the degree of language input and the participant's age," *Frontiers in Psychology*, vol. 7, p. 1116, 2016.

[BHI02] C. S. Bullock, III and M. Hood III, "One person - no vote; one vote; two votes: voting methods, ballot types, and undervote frequency in the 2000 presidential election," *Social Science Quarterly*, vol. 83, no. 4, pp. 981–993, 2002.

[Bur04] T. Burchardt, "Capabilities and disability: the capabilities framework and the social model of disability," *Disability & society*, vol. 19, no. 7, pp. 735–751, 2004.

[BDN+12] W. E. Burr, D. F. Dodson, E. M. Newton, R. A. Perlner, T. W. Polk, S. Gupta, and E. A. Nabbus, *Electronic Authentication Guideline: Recommendations of the National Institute of Standards and Technology - Special Publication 800-63-1.* USA: CreateSpace Independent Publishing Platform, 2012.

[BDP04] W. E. Burr, D. F. Dodson, and T. W. Polk, "Nist special publication 800-63-2," *Electronic Authentication Guideline*, vol. 1, 2004.

[But99] B. Butterworth, *The Mathematical Brain.* Macmillan, 1999.

[But02] B. Butterworth, "Mathematics and the brain," 2002, opening Address to The Mathematical Association.

[BCW96] B. Butterworth, L. Cipolotti, and E. K. Warrington, "Short term memory impairment and arithmetical ability," *The Quarterly Journal of Experimental Psychology Section A*, vol. 49, no. 1, pp. 251–262, 1996.

[BZGJ01] B. Butterworth, M. Zorzi, L. Girelli, and A. R. Jonckheere, "Storage and retrieval of addition facts: The role of number comparison," *The Quarterly Journal of Experimental Psychology Section A*, vol. 54, no. 4, pp. 1005–1029, 2001, pMID: 11765730.

[CG16] J. Cancela and B. Geys, "Explaining voter turnout: A meta-analysis of national and subnational elections," *Electoral Studies*, vol. 42, pp. 264–275, 2016.

[Can18] M. Canevaro, *Majority Rule vs. Consensus: The Practice of Democratic Deliberation in the Greek Poleis.* Edinburgh University Press, 09 2018, pp. 101–156.

[CJ16] K. Cao and A. K. Jain, "Hacking mobile phones using 2D printed fingerprints," Michigan State University, Tech. Rep., 2016.

[CM07] D. Card and E. Moretti, "Does voting technology affect election outcomes? Touch-screen voting and the 2004 presidential election," *The Review of Economics and Statistics*, vol. 89, no. 4, pp. 660–673, 2007.

[Cİ13] M. Carreras and Y. İrepoğlu, "Trust in elections, vote buying, and turnout in latin america," *Electoral Studies*, vol. 32, no. 4, pp. 609–619, 2013.

[CMP12] C. Castelluccia, D. Markus, and D. Perito, "Adaptive Password-Strength Meters from Markov Models," in *19th Annual Network & Distributed System Security Symposium – NDSS.* San Diego, United States: ISOC, 2 2012. https://hal.inria.fr/hal-00747824

[Cen14] Centrify, "Centrify password survey: Summary," Centrify, Tech. Rep., 2014, accessed: 2017-12-20. https://www.centrify.com/resources/5778-centrify-password-survey-summary/

[CSM+17] H. Chang, L. Sprute, E. A. Maloney, S. L. Beilock, and M. G. Berman, "Simple arithmetic: not so simple for highly math anxious individuals," *Social cognitive and affective neuroscience*, vol. 12, no. 12, pp. 1940–1949, 2017.

[CTY06] N. Chater, J. B. Tenenbaum, and A. Yuille, "Probabilistic models of cognition: Conceptual foundations," *Trends in Cognitive Sciences*, vol. 10, no. 7, pp. 287–291, 2006, special issue: Probabilistic models of cognition.

[CAA+16] R. Chatterjee, A. Athayle, D. Akhawe, A. Juels, and T. Ristenpart, "pASSWORD tYPOS and how to correct them securely," in *IEEE Symposium on Security and Privacy.* IEEE, 2016, pp. 799–818.

[CWP+17] R. Chatterjee, J. Woodage, Y. Pnueli, A. Chowdhury, and T. Ristenpart, "The typtop system: Personalized typo-tolerant password checking," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '17. New York, NY, USA: ACM, 2017, pp. 329–346.

[Cha16] D. Chaum, "Random-sample voting," 2016.

[Cha04] D. Chaum, "Secret-ballot receipts: True voter-verifiable elections," *IEEE security & privacy*, vol. 2, no. 1, pp. 38–47, 2004.

[Cha83] D. Chaum, "Blind signatures for untraceable payments," in *Advances in Cryptology*, D. Chaum, R. L. Rivest, and A. T. Sherman, Eds. Boston, MA: Springer US, 1983, pp. 199–203.

[CCC+08] D. Chaum, R. Carback, J. Clark, A. Essex, S. Popoveniuc, R. L. Rivest, P. Y. Ryan, E. Shen, and A. T. Sherman, "Scantegrity II: End-to-End Verifiability for Optical Scan Election Systems using Invisible Ink Confirmation Codes," *17th USENIX Security Symposium*, vol. 8, pp. 1–13, 2008.

[CEC+08] D. Chaum, A. Essex, R. Carback, J. Clark, S. Popoveniuc, A. Sherman, and P. Vora, "Scantegrity: End-to-end voter-verifiable optical-scan voting," *IEEE Security & Privacy*, vol. 6, no. 3, pp. 40–46, 2008.

[CRS05] D. Chaum, P. Y. A. Ryan, and S. A. Schneider, "A practical voter-verifiable election scheme," in *10th European Symposium on Research in Computer Security – ESORICS*, 2005, pp. 118–139.

[Cha81] D. L. Chaum, "Untraceable electronic mail, return addresses, and digital pseudonyms," *Commun. ACM*, vol. 24, no. 2, pp. 84–90, Feb. 1981.

[CDJ05] Y. Chen, S. C. Dass, and A. K. Jain, "Fingerprint quality indices for predicting authentication performance," in *Audio- and Video-Based Biometric Person Authentication*, T. Kanade, A. Jain, and N. K. Ratha, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 160–170.

[CBvO07] S. Chiasson, R. Biddle, and P. C. van Oorschot, "A second look at the usability of click-based graphical passwords," in *Proceedings of the 3rd Symposium on Usable Privacy and Security*, ser. SOUPS '07. New York, NY, USA: ACM, 2007, pp. 1–12.

[CvOB06] S. Chiasson, P. C. van Oorschot, and R. Biddle, "A usability study and critique of two password managers." in *USENIX Security Symposium*, 2006, pp. 1–16.

[CFSW12] E. Chin, A. P. Felt, V. Sekar, and D. Wagner, "Measuring user confidence in smartphone security and privacy," in *Proceedings of the 8th Symposium on Usable Privacy and Security*, ser. SOUPS '12. New York, NY, USA: ACM, 2012, pp. 1–16.

[CTI+18] B. Choudhury, P. Then, B. Issac, V. Raman, and M. Haldar, "A survey on biometrics and cancelable biometrics systems," *International Journal of Image and Graphics*, vol. 18, p. 1850006, 01 2018.

[CKW08] J. Cichoń, M. Kutyłowski, and B. Węglorz, "Short ballot assumption and threeballot voting protocol," in *International Conference on Current Trends in Theory and Practice of Computer Science*. Springer, 2008, pp. 585–598.

[Cis18] Cisco, "Global cloud index: Forecast and methodology, 2016–2021," Cisco, Tech. Rep., 2018. https://web.archive.org/web/20190320151956/https://www.cisco.com/c/en/us/solutions/collateral/service-provider/global-cloud-index-gci/white-paper-c11-738085.html

[CJE+06] L. S. Clair, L. Johansen, W. Enck, M. Pirretti, P. Traynor, P. McDaniel, and T. Jaeger, "Password exhaustion: Predicting the end of password usefulness," in *International Conference on Information Systems Security*. Springer, 2006, pp. 37–55.

[Cod15] S. Cody, "Do only the eyes have it? Predicting subsequent memory with simultaneous neural and pupillometry data," Master's thesis, The Ohio State University, 2015.

[CM98] J. M. Colomer and I. McLean, "Electing popes: approval balloting and qualified-majority rule," *Journal of Interdisciplinary History*, vol. 29, no. 1, pp. 1–22, 1998.

[UK 17] UK Electoral Commission, "Understanding public attitudes towards elections and voting," 10 2017.

[Con01] R. D. Congleton, "Rational ignorance, rational voter expectations, and public policy: A discrete informational foundation for fiscal illusion," *Public Choice*, vol. 107, no. 1, pp. 35–64, 4 2001.

[Cra16] L. F. Cranor. (2016) Time to rethink mandatory password changes. https://web.archive.org/web/20190302102425/https://www.ftc.gov/news-events/blogs/techftc/2016/03/time-rethink-mandatory-password-changes

[CF16] H. Crossman and D. Fischer, "Participatory budgeting and transparency in municipal finances," *Journal of Accounting, Ethics and Public Policy*, vol. 17, no. 3, 2016.

[Cul18] C. Cullen, "Global internet phenomena report," Sandvine, Tech. Rep., 2018.

[CHJ+14] C. Culnane, J. Heather, R. Joaquim, P. Y. A. Ryan, S. Schneider, and V. Teague, "Faster print on demand for Prêt à Voter," in *Electronic Voting Technology Workshop/Workshop on Trustworthy Elections – EVT/WOTE*, 2014. https://www.usenix.org/conference/evtwote14/workshop-program/presentation/culnane

[CYMC16] M. T. Curran, J.-k. Yang, N. Merrill, and J. Chuang, "Passthoughts authentication with low cost EarEEG," in *IEEE 38th Annual International Conference of the Engineering in Medicine and Biology Society – EMBC*. IEEE, 2016, pp. 1979–1982.

[Cze06] M. Cześnik, "Voter turnout and democratic legitimacy in central eastern Europe," *Polish Sociological Review*, no. 156, 2006.

[Dam64] F. J. Damerau, "A technique for computer detection and correction of spelling errors," *Communications of the ACM*, vol. 7, no. 3, pp. 171–176, 1964.

[DBC+14] A. Das, J. Bonneau, M. Caesar, N. Borisov, and X. Wang, "The tangled web of password reuse." in *NDSS*, vol. 14, 2014, pp. 23–26.

[DMC16] R. Das, E. Maiorana, and P. Campisi, "Eeg biometrics using visual stimuli: A longitudinal study," *IEEE Signal Processing Letters*, vol. 23, no. 3, pp. 341–345, 2016.

[DH14] S. Deering and R. Hinden. (2014) Rfc 2460-internet protocol, version 6 (ipv6) specification, 1998. http://www.ietf.org/rfc/rfc2460.txt

[Deh92] S. Dehaene, "Varieties of numerical abilities," *Cognition*, vol. 44, no. 1, pp. 1–42, 1992.

[DHvdG+13] D. Demirel, M. Henning, J. van de Graaf, P. Y. A. Ryan, and J. A. Buchmann, "Prêt à Voter providing everlasting privacy," in *E-Voting and Identify - 4th International Conference, Vote-ID*, 2013, pp. 156–175.

[Wor17] World Forum for Democracy, "Is populism a problem? Evaluation data," Council of Europe, Tech. Rep., 2017.

[DG11] F. Deravi and S. P. Guness, "Gaze trajectory as a biometric modality." in *Biosignals*, 2011, pp. 335–341.

[DNBB10] M. O. Derawi, C. Nickel, P. Bours, and C. Busch, "Unobtrusive user-authentication on mobile phones using biometric gait recognition," in *6th International Conference on Intelligent Information Hiding and Multimedia Signal Processing*. IEEE, 2010, pp. 306–311.

[DORS08] Y. Dodis, R. Ostrovsky, L. Reyzin, and A. Smith, "Fuzzy extractors: How to generate strong keys from biometrics and other noisy data," *SIAM Journal on Computing*, vol. 38, no. 1, pp. 97–139, 2008. https://doi.org/10.1137/060651380

[Doe18] K. Doel, "Bad password habits die hard, shows splashdata's 8th annual worst passwords list," SplashData, Tech. Rep., 2018. https://web.archive.org/web/20190129025352/https://www.prweb.com/releases/bad_password_habits_die_hard_shows_splashdata_s_8th_annual_worst_passwords_list/prweb15987071.htm

[Doe17] K. Doel, "Starwars premieres (as a terrible password!) on splashdata's annual list of "worst passwords of the year"," SplashData, Tech. Rep., 2017. https://web.archive.org/web/20180109121442/http://www.prweb.com/releases/2017/12/prweb15029008.htm

[DR12] D. Donno and N. Roussias, "Does cheating pay? The effect of electoral misconduct on party systems," *Comparative Political Studies*, vol. 45, pp. 575–605, 05 2012.

[DMCS10] A. Donovan, R. Muth, B. Chen, and D. Sehr, "Pnacl: Portable native client executables," *Google White Paper*, 2010.

[Dot11] K. Dotson. (2011) Third largest bitcoin exchange bitomat lost their wallet, over 17,000 bitcoins missing. https://web.archive.org/web/*/https://siliconangle.com/2011/08/01/third-largest-bitcoin-exchange-bitomat-lost-their-wallet-over-17000-bitcoins-missing/

[DK15] M. Dürmuth and T. Kranz, "On password guessing with gpus and fpgas," in *Technology and Practice of Passwords*, S. F. Mjølsnes, Ed. Cham: Springer International Publishing, 2015, pp. 19–38.

[ESC05] D. I. Eastlake, J. Schiller, and S. Crocker. (2005) Rfc4086: Randomness requirements for security. https://tools.ietf.org/html/rfc4086

[ERLM15] S. Eberz, K. B. Rasmussen, V. Lenders, and I. Martinovic, "Preventing lunchtime attacks: Fighting insider threats with eye movement biometrics," in *22nd Annual Network and Distributed System Security Symposium – NDSS*, 2015. https://www.ndss-symposium.org/ndss2015/preventing-lunchtime-attacks-fighting-insider-threats-eye-movement-biometrics

[Ein17] W. Einhäuser, *The Pupil as Marker of Cognitive Processes*. Singapore: Springer Singapore, 2017, pp. 141–169.

[EG14] C. Enguehard and J.-D. Graton, "Machines à voter et élections politiques en France: étude quantitative de la précision des bureaux de vote," *Cahiers Droit, Sciences & Technologies*, vol. 4, no. 4, pp. 159–198, 2014.

[EE15] J. Everett and B. Earp, "A tragedy of the (academic) commons: interpreting the replication crisis in psychology as a social dilemma for early-career researchers," *Frontiers in Psychology*, vol. 6, p. 1152, 2015. https://www.frontiersin.org/article/10.3389/fpsyg.2015.01152

[Fay05] J. A. Fay, "Elderly electors go postal: Ensuring absentee ballot integrity for older voters," *Elder Law Journal*, vol. 13, p. 453, 2005.

[FGNT15] H. Ferradi, R. Géraud, D. Naccache, and A. Tria, "When organized crime applies academic results: a forensic analysis of an in-card listening device," *Journal of Cryptographic Engineering*, vol. 6, 10 2015.

[FIP11] M. J. Fischer, M. Iorga, and R. Peralta, "A public randomness service," in *Proceedings of the International Conference on Security and Cryptography – SECRYPT*. IEEE, 2011, pp. 434–438.

[FH07] D. Florêncio and C. Herley, "A large-scale study of web password habits," in *Proceedings of the 16th international conference on World Wide Web*. ACM, 2007, pp. 657–666.

[FHC07] D. Florêncio, C. Herley, and B. Coskun, "Do strong web passwords accomplish anything?" in *Proceedings of the 2Nd USENIX Workshop on Hot Topics in Security*, ser. HOTSEC'07. Berkeley, CA, USA: USENIX Association, 2007, pp. 1–6. http://dl.acm.org/citation.cfm?id=1361419.1361429

[FHvO14a] D. Florêncio, C. Herley, and P. C. van Oorschot, "An administrator's guide to internet password research." in *LISA*, vol. 14, 2014, pp. 35–52.

[FHvO14b] D. Florêncio, C. Herley, and P. C. van Oorschot, "Password portfolios and the finite-effort user: Sustainably managing large numbers of accounts," in *23rd USENIX Security Symposium*. San Diego, CA: USENIX Association, 2014, pp. 575–590. https://web.archive.org/web/20170823094633/https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/florencio

[FB08] A. Forget and R. Biddle, "Memorability of persuasive passwords," in *CHI '08 Extended Abstracts on Human Factors in Computing Systems*, ser. CHI EA '08. New York, NY, USA: ACM, 2008, pp. 3759–3764.

[Fow13] A. Fowler, "Electoral and policy consequences of voter turnout: Evidence from compulsory voting in Australia," *Quarterly Journal of Political Science*, vol. 8, no. 2, pp. 159–182, 2 2013.

[Fra15] A. V. Frane, "Are per-family type i error rates relevant in social and behavioral science?" *Journal of Modern Applied Statistical Methods*, vol. 14, no. 1, p. 5, 2015.

[FRS18] T. Frye, O. J. Reuter, and D. Szakonyi, "Hitting them with carrots: Voter intimidation and vote buying in Russia," *British Journal of Political Science*, pp. 1–25, 2018.

[FFC+06] L. S. Fuchs, D. Fuchs, D. L. Compton, S. R. Powell, P. M. Seethaler, A. M. Capizzi, C. Schatschneider, and J. M. Fletcher, "The cognitive correlates of third-grade skill in arithmetic, algorithmic computation, and arithmetic word problems." *Journal of Educational Psychology*, vol. 98, no. 1, p. 29, 2006.

[GP04] S. Gabel and S. Peters, "Presage of a paradigm shift? Beyond the social model of disability toward resistance theories of disability," *Disability & Society*, vol. 19, no. 6, pp. 585–600, 2004.

[GHS06] D. Gafurov, K. Helkala, and T. Søndrol, "Biometric gait authentication using accelerometer sensor," *Journal of Computers*, vol. 1, pp. 51–59, 2006.

[GZ19] S. D. Galbraith and L. Zobernig, "Obfuscated fuzzy hamming distance and conjunctions from subset product problems," Cryptology ePrint Archive, Report 2019/620, 2019, https://eprint.iacr.org/2019/620.

[GNR+13] C. Galdi, M. Nappi, D. Riccio, V. Cantoni, and M. Porta, "A new gaze analysis based soft-biometric," in *Pattern Recognition*, J. A. Carrasco-Ochoa, J. F. Martínez-Trinidad, J. S. Rodríguez, and G. S. di Baja, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 136–144.

[GNRW16] C. Galdi, M. Nappi, D. Riccio, and H. Wechsler, "Eye movement analysis for human authentication: a critical survey," *Pattern Recognition Letters*, vol. 84, pp. 272–283, 2016.

[GF15] E. Ganuza and F. Francés, "The participatory challenge: Deliberation and democratic inclusion in participatory budgeting," *Participations*, no. 1, pp. 167–190, 2015.

[GL14] S. Garfinkel and H. R. Lipford, *Usable Security: History, Themes, and Challenges*, ser. Synthesis Lectures on Information Security, Privacy, and Trust. Morgan & Claypool Publishers, 2014. https://books.google.fr/books?id=HPS9BAAAQBAJ

[Gat04] K. A. Gates, "Advocating alternative futures: Screening biometrics and related technologies in science-fiction cinema," in *Annual meeting of the International Communication Association, New Orleans*, vol. 27, 2004.

[GF06] S. Gaw and E. W. Felten, "Password management strategies for online accounts," in *Proceedings of the Second Symposium on Usable Privacy and Security*, ser. SOUPS '06. New York, NY, USA: ACM, 2006, pp. 44–55.

[GXQ+17] C. Ge, L. Xu, W. Qiu, Z. Huang, J. Guo, G. Liu, and Z. Gong, "Optimized password recovery for sha-512 on gpus," in *IEEE International Conference on Computational Science and Engineering – CSE – and Embedded and Ubiquitous Computing – EUC*, vol. 2. IEEE, 2017, pp. 226–229.

[GB18] R. L. German and K. S. Barber, "Consumer attitudes about biometric authentication," University of Texas at Austin Center for Identity, Tech. Rep., 2018.

[Gey06] B. Geys, ""Rational" theories of voter turnout: A review," *Political Studies Review*, vol. 4, no. 1, pp. 16–35, 2006. https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1478-9299.2006.00034.x

[Gia15] D. Giannetti, "Secret voting in the italian parliament," *Secrecy and publicity in votes and debates*, pp. 108–130, 2015.

[GD14] H. Gjermundrød and I. Dionysiou, "Recirculating lost coins in cryptocurrency systems," in *Business Information Systems Workshops*, W. Abramowicz and A. Kokkinaki, Eds. Cham: Springer International Publishing, 2014, pp. 229–240.

[GSD16] M. Golla, T. Schnitzler, and M. Dürmuth, "Will any password do? Exploring rate-limiting on the web," in *Who Are You ?! Adventures in Authentication*, 2016.

[GMM15] C. A. Gomes, D. Montaldi, and A. Mayes, "The pupil as an indicator of unconscious memory: Introducing the pupil priming effect," *Psychophysiology*, vol. 52, no. 6, pp. 754–769, 2015.

[GTAF03] S. Gonzalez, C. M. Travieso, J. B. Alonso, and M. A. Ferrer, "Automatic biometric identification system by hand geometry," in *IEEE 37th Annual International Carnahan Conference onSecurity Technology*, Oct 2003, pp. 281–284.

[GOKdJN15] E. González-Ocantos, C. Kiewiet de Jonge, and D. W. Nickerson, "Legitimacy buying: The dynamics of clientelism in the face of legitimacy challenges," *Comparative Political Studies*, vol. 48, no. 9, pp. 1127–1158, 2015.

[Goo15] D. Goodin. (2015) Once seen as bulletproof, 11 million+ ashley madison passwords already cracked. https://web.archive.org/web/20180803014106/https://arstechnica.com/information-technology/2015/09/once-seen-as-bulletproof-11-million-ashley-madison-passwords-already-cracked/

[GTP09] G. Goudelis, A. Tefas, and I. Pitas, "Emerging biometric modalities: a survey," *Journal on Multimodal User Interfaces*, vol. 2, no. 3, p. 217, Sep 2009.

[Gre17] S. Gressin. (2017) The equifax data breach: What to do. https://web.archive.org/web/20190304122541/https://www.consumer.ftc.gov/blog/2017/09/equifax-data-breach-what-do

[Gri12] M. Grissinger, "Avoiding confusion with alphanumeric characters," *Pharmacy and Therapeutics*, vol. 37, no. 12, pp. 663–665, 2012.

[GS04] K. Grönlund and M. Setälä, "Low electoral turnout: An indication of a legitimacy deficit?" in *European Consortium for Political Research Joint Sessions of Workshops*, 2004.

[Gro04] J. Groth, "Efficient maximal privacy in boardroom voting and anonymous broadcast," in *Financial Cryptography*, A. Juels, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 90–104.

[GM16] A. Guillevic and F. Morain, "Discrete Logarithms," in *Guide to pairing-based cryptography*, N. E. Mrabet and M. Joye, Eds. CRC Press - Taylor and Francis Group, Dec. 2016, p. 42. https://hal.inria.fr/hal-01420485

[HSGMS02] L. Q. Ha, E. I. Sicilia-Garcia, J. Ming, and F. J. Smith, "Extension of zipf's law to words and phrases," in *Proceedings of the 19th International Conference on Computational Linguistics - Volume 1*, ser. COLING '02. Stroudsburg, PA, USA: Association for Computational Linguistics, 2002, pp. 1–6.

[HG11] M. S. Handcock and K. J. Gile, "Comment: On the concept of snowball sampling," *Sociological Methodology*, vol. 41, no. 1, pp. 367–371, 2011.

[HR17] R. Hanifatunnisa and B. Rahardjo, "Blockchain based e-voting recording system design," in *11th International Conference on Telecommunication Systems Services and Applications – TSSA*. IEEE, 2017, pp. 1–6.

[HV12] M. Hanmandlu and S. Vasikarla, "Online biometric authentication using facial thermograms," in *IEEE Applied Imagery Pattern Recognition Workshop – AIPR*, Oct 2012, pp. 1–6.

[HPT+10] M. J. Hanmer, W.-H. Park, M. W. Traugott, R. G. Niemi, P. S. Herrnson, B. B. Bederson, and F. C. Conrad, "Losing fewer votes: the impact of changing voting systems on residual votes," *Political Research Quarterly*, vol. 63, no. 1, pp. 129–142, 2010.

[Har46] G. W. Hartmann, "Further evidence on the unexpected large size of recognition vocabularies among college students." *Journal of educational psychology*, vol. 37, no. 7, p. 436, 1946.

[Har19] K. Hartnett. (2019) Cryptography that can't be hacked. https://www.quantamagazine.org/how-the-evercrypt-library-creates-hacker-proof-cryptography-20190402/

[Har16] C. J. Harvey, "Changes in the menu of manipulation: Electoral fraud, ballot stuffing, and voter pressure in the 2011 russian election," *Electoral Studies*, vol. 41, pp. 105 – 117, 2016.

[HAK13] H. Hasan and S. Abdul-Kareem, "Fingerprint image enhancement and recognition algorithms: A survey," *Neural Computing and Applications*, vol. 23, 11 2013.

[HPM15] G. Hatzivasilis, I. Papaefstathiou, and C. Manifavas, "Password hashing competition-survey and benchmark." *IACR Cryptology ePrint Archive*, vol. 2015, p. 265, 2015.

[HA14] Y. M. Hausawi and W. H. Allen, "An assessment framework for usable-security based on decision science," in *Human Aspects of Information Security, Privacy, and Trust: Second International Conference – HAS*, T. Tryfonas and I. Askoxylakis, Eds. Cham: Springer International Publishing, 2014, pp. 33–44.

[Hec95] J. C. Heckelman, "The effect of the secret ballot on voter turnout rates," *Public Choice*, vol. 82, no. 1-2, pp. 107–124, 1995.

[HSS09] K. J. Henry, D. R. Stinson, and J. Sui, "The effectiveness of receipt-based attacks on three-ballot," *IEEE Transactions on Information Forensics and Security*, vol. 4, no. 4, p. 699, 2009.

[HS10] W. Herbranson and J. Schroeder, "Are birds smarter than mathematicians? Pigeons (Columba livia) perform optimally on a version of the monty hall dilemma," *Journal of comparative psychology*, vol. 124, pp. 1–13, 02 2010.

[HS03] M. C. Herron and J. S. Sekhon, "Overvoting and representation: An examination of overvoted presidential ballots in broward and miami-dade counties," *Electoral Studies*, vol. 22, no. 1, pp. 21–47, 2003.

[HB16] B. Herzog and Y. Balmas, "Great crypto failures," in *Virus Bulletin Conference*, 2016.

[HK18] L. K. Ho and N. Katuk, "Users' Acceptance Study of OAuth Manager Module for Social Login in Mobile Environment," *Journal of Telecommunication, Electronic and Computer Engineering – JTEC*, vol. 10, no. 2-4, pp. 41–45, 2018.

[Hoe63] W. Hoeffding, "Probability inequalities for sums of bounded random variables," *Journal of the American Statistical Association*, vol. 58, no. 301, pp. 13–30, 3 1963. http://www.jstor.org/stable/2282952?

[Hol16] J. Holland. (2016, 11) Donald trump won only by undermining an already broken democracy. https://web.archive.org/web/20190416121149/https://www.thenation.com/article/donald-trump-only-won-by-undermining-an-already-broken-democracy/

[HB01] N. J. Hopper and M. Blum, "Secure human identification protocols," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2001, pp. 52–66.

[Hor15] D. A. Horwitz, "A picture's worth a thousand words: Why ballot selfies are protected by the first amendment," *SMU Science and Technology Law Review*, vol. 18, p. 247, 2015.

[HKB+15] J. H. Huh, H. Kim, R. B. Bobba, M. N. Bashir, and K. Beznosov, "On the memorability of system-generated pins: Can chunking help?" in *Eleventh Symposium On Usable Privacy and Security SOUPS*. Ottawa: USENIX Association, 2015, pp. 197–209.

[HRS+97] C. Hulme, S. Roodenrys, R. Schweickert, G. D. A. Brown, S. Martin, and G. Stuart, "Word-frequency effects on short-term memory tasks: Evidence for a redintegration process in immediate serial recall." *Journal of Experimental Psychology: Learning, Memory, and Cognition*, vol. 23, no. 5, p. 1217, 1997.

[Hun76] D. Hunter, "An upper bound for the probability of a union," *Journal of Applied Probability*, vol. 13, no. 3, pp. 597–603, 1976.

[Hus08] A. Huszti, "A secure electronic voting scheme," *Periodica Polytechnica Electrical Engineering*, vol. 51, no. 3-4, pp. 141–146, 2008.

[IM07] S. Inoue and T. Matsuzawa, "Working memory of numerals in chimpanzees," *Current Biology*, vol. 17, no. 23, 2007.

[Pon17] Ponemon Institute, "The cost of credential stuffing," Ponemon Institute, Tech. Rep., 2017.

[Dem16] (2016) Global forum on modern direct democracy. Democracy International. https://web.archive.org/web/20180312035936/https://www.democracy-international.org/global-forum-modern-direct-democracy-2016

[IRI19] IRIS. (2019) Corporate info. https://web.archive.org/web/20190308135058/http://www.iris.com.my/corporate_info.html

[IWS04] B. Ives, K. R. Walsh, and H. Schneider, "The domino effect of password reuse," *Communications of the ACM*, vol. 47, no. 4, pp. 75–78, Apr. 2004.

[JPG+16] D. Jaeger, C. Pelchen, H. Graupner, F. Cheng, and C. Meinel, "Analysis of publicly leaked credentials and the long story of password (re-) use," in *Proc. Int. Conf. Passwords*, 2016.

[Jar14] C. Jara, "Democratic legitimacy under strain? Declining political support and mass demonstrations in Chile," *European Review of Latin American and Caribbean Studies/Revista Europea de Estudios Latinoamericanos y del Caribe*, pp. 25–50, 2014.

[JJY08] H.-K. Jee, S.-U. Jung, and J.-H. Yoo, "Liveness detection for embedded face recognition system," in *International Journal of Computer and Information Engineering*, vol. 2, no. 6, 2008.

[Jef07] D. Jefferson, "What happened in Sarasota county?" *Voting Technologies*, vol. 37, no. 2, 12 2007. https://web.archive.org/web/20190424175222/https://www.nae.edu/7665/WhatHappenedinSarasotaCounty

[JGK+03] W. Jensen, S. Gavrila, V. Korolev *et al.*, "Picture password: A visual login technique for mobile devices," National Institute of Standards and Technology, Tech. Rep., 2003.

[JL10] P. N. Johnson-Laird, "Mental models and human reasoning," *Proceedings of the National Academy of Sciences*, vol. 107, no. 43, pp. 18 243–18 250, 2010. http://www.pnas.org/content/107/43/18243

[Jon03] D. W. Jones, "A brief illustrated history of voting," *University of Iowa Department of Computer Science.*, 2003. http://homepage.divms.uiowa.edu/~jones/voting/pictures/

[JH06] D. W. Jones and M. Hall, "Technologists as political reformers: Lessons from the early history of voting machines," in *Society for the History of Technology Annual Meeting, Las Vegas*, vol. 13, 2006.

[JJB06] H. Jones, J. Juang, and G. Belote, "Threeballot in the field," 2006.

[Ju18] C. Ju. (2018) You can't hack a piece of paper: Jake braun talks us election security. https://web.archive.org/web/20180408041605/http://chicagopolicyreview.org/2018/04/01/you-cant-hack-a-piece-of-paper-jake-braun-talks-u-s-election-security/

[JL97] W.-S. Juang and C.-L. Lei, "A secure and practical electronic voting scheme for real world environments," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, pp. 64–71, 09 1997.

[JA09] M. Just and D. Aspinall, "Personal choice and challenge questions: a security and usability assessment," in *Proceedings of the 5th Symposium on Usable Privacy and Security*. ACM, 2009, p. 8.

[JA10] M. Just and D. Aspinall, "Challenging challenge questions: An experimental analysis of authentication technologies and user behaviour," *Policy & Internet*, vol. 2, no. 1, pp. 99–115, 2010.

[KM11] A. Kafkas and D. Montaldi, "Recognition memory strength is predicted by pupillary responses at encoding while fixation patterns distinguish recollection from familiarity," *The Quarterly Journal of Experimental Psychology*, vol. 64, no. 10, pp. 1971–1989, 2011.

[Kal00] B. Kaliski, "Pkcs# 5: Password-based cryptography specification version 2.0," RFC Editor, Tech. Rep., 2000.

[KDFK15] R. Kang, L. Dabbish, N. Fruchter, and S. Kiesler, "My data just goes everywhere: user mental models of the internet and implications for privacy and security," in *Symposium on Usable Privacy and Security – SOUPS*. USENIX Association Berkeley, CA, 2015, pp. 39–52.

[Kar19] E. Karlsten. (2019) 13 MEPs pressed the wrong button on crucial copyright vote. https://web.archive.org/web/20190410124538/https://medium.com/@emanuelkarlsten/13-meps-pressed-the-wrong-button-on-crucial-copyright-vote-f1ccbd2e3b0a

[KSC11] A. Karole, N. Saxena, and N. Christin, "A comparative usability evaluation of traditional password managers," in *Proceedings of the 13th International Conference on Information Security and Cryptology*, ser. ICISC'10. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 233–251. http://dl.acm.org/citation.cfm?id=2041036.2041056

[KD14] S. Karthika and P. Devaki, "An efficient user authentication using captcha and graphical passwords-a survey," *International Journal of Science and Research*, p. 123, 2014.

[KKK12] P. Kasprowski, O. V. Komogortsev, and A. Karpov, "First eye movement verification and identification competition at btas 2012," in *IEEE 5th International Conference on Biometrics: Theory, Applications and Systems – BTAS*. IEEE, 2012, pp. 195–202.

[KSS09] M. Keith, B. Shao, and P. Steinbart, "A behavioral analysis of passphrase design and effectiveness," *Journal of the Association for Information Systems*, vol. 10, no. 2, p. 2, 2009.

[KSS07] M. Keith, B. Shao, and P. J. Steinbart, "The usability of passphrases for authentication: An empirical field study," *International journal of human-computer studies*, vol. 65, no. 1, pp. 17–28, 2007.

[Kel18a] R. Keller. (2018, 8) Missouri officials look to improve election security. https://web.archive.org/web/20180807045000/https://www.columbiatribune.com/news/20180806/missouri-officials-look-to-improve-election-security

[KKM+12] P. G. Kelley, S. Komanduri, M. L. Mazurek, R. Shay, T. Vidas, L. Bauer, N. Christin, L. F. Cranor, and J. Lopez, "Guess again (and again and again): Measuring password strength by simulating password-cracking algorithms," in *IEEE Symposium on Security and Privacy*. IEEE, 2012, pp. 523–537.

[Kel18b] P. Kellner. (2018) In January 2019 Britain will officially switch from a pro-Brexit to an anti-Brexit country, and this is how we know. https://web.archive.org/web/20190210025342/https://www.independent.co.uk/voices/final-say-remain-leave-second-referendum-brexit-no-deal-crossover-day-a8541576.html

[KRMC10] J. Kelsey, A. Regenscheid, T. Moran, and D. Chaum, "Attacking paper-based e2e voting systems," in *Towards Trustworthy Elections*. Springer, 2010, pp. 370–387.

[Ken38] M. G. Kendall, "A new measure of rank correlation," *Biometrika*, vol. 30, no. 1/2, pp. 81–93, 1938.

[KB81] G. Keren and S. Baggen, "Recognition models of alphanumeric characters," *Perception and Psychophysics*, vol. 29, no. 3, pp. 234–246, 1981.

[KTR13] D. Khader, Q. Tang, and P. Y. A. Ryan, "Proving Prêt à Voter receipt free using computational security models," in *Electronic Voting Technology Workshop / Workshop on Trustworthy Elections – EVT/WOTE*, 2013. https://www.usenix.org/conference/evtwote13/workshop-program/presentation/khader

[Kha18] I. A. Khan. (2018) ECP cancels Shangla polls over low turnout of women. https://web.archive.org/web/20180811164226/https://www.dawn.com/news/1426325

[Kha19] S. Khandelwal. (2019) Facebook caught asking some users passwords for their email accounts. https://web.archive.org/web/20190404071339/https://amp.thehackernews.com/thn/2019/04/facebook-email-password.html

[KB84] D. E. Kieras and S. Bovair, "The role of a mental model in learning to operate a device," *Cognitive Science*, vol. 8, no. 3, pp. 255–273, 1984.

[Kim03] B. Kim, "Help America Vote Act," 2003.

[Kim95] H.-J. Kim, "Biometrics, is it a viable proposition for identity authentication and access control?" *Computers & Security*, vol. 14, no. 3, pp. 205–214, 1995.

[KH16] B. A. King and K. Hale, *Why Don't Americans Vote? Causes and Consequences*. ABC-CLIO, 2016. https://books.google.fr/books?id=-bCJDAAAQBAJ

[Kin01] R. F. King, "Counting the votes: South Carolina's stolen election of 1876," *Journal of Interdisciplinary History*, vol. 32, no. 2, pp. 169–191, 2001.

[KMI18] KMIZ, "Boone county clerk apologizes for primary election results delay," 8 2018. https://web.archive.org/web/20180808063008/https://www.abc17news.com/news/boone-county-clerk-apologizes-for-primary-election-results-delay/778723208

[KFB05] K. Kollreider, H. Fronthaler, and J. Bigun, "Evaluating liveness by face images and the structure tensor," in *IEEE 4th Workshop on Automatic Identification Advanced Technologies – AutoID*, Oct 2005, pp. 75–80.

[KSK+11] S. Komanduri, R. Shay, P. G. Kelley, M. L. Mazurek, L. Bauer, N. Christin, L. F. Cranor, and S. Egelman, "Of passwords and people: Measuring the effect of password-composition policies," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '11. New York, NY, USA: ACM, 2011, pp. 2595–2604.

[KZK09] A. Kong, D. Zhang, and M. Kamel, "A survey of palmprint recognition," *Pattern Recognition*, vol. 42, no. 7, pp. 1408–1418, Jul. 2009.

[Kra19] J. H. Krantz. (2019) Psychological research on the net. https://psych.hanover.edu/research/exponnet.html

[KEA+14] H. Krasnova, N. Eling, O. Abramova, P. Buxmann *et al.*, "Dangers of facebook login for mobile apps: Is there a price tag for social information?" in *ICIS*, 2014.

[Kre19] B. Krebs. (2019) Facebook stored hundreds of millions of user passwords in plain text for years. https://web.archive.org/web/20190322091235/https://krebsonsecurity.com/2019/03/facebook-stored-hundreds-of-millions-of-user-passwords-in-plain-text-for-years/

[Kre18] B. Krebs. (2018) Twitter to all users: Change your password now! https://web.archive.org/web/20190402093127/https://krebsonsecurity.com/2018/05/twitter-to-all-users-change-your-password-now/

[KSMD08] H. Kruger, T. Steyn, B. D. Medlin, and L. Drevin, "An empirical assessment of factors impeding effective password management," *Journal of Information Privacy and Security*, vol. 4, no. 4, pp. 45–59, 2008.

[KKJ+13] H. Kumar, S. Kumar, R. Joseph, D. Kumar, S. K. S. Singh, and P. Kumar, "Rainbow table to crack password using md5 hashing algorithm," in *IEEE Conference on Information & Communication Technologies – ICT*. IEEE, 2013, pp. 433–439.

[KRC06] C. Kuo, S. Romanosky, and L. F. Cranor, "Human selection of mnemonic phrase-based passwords," in *Proceedings of the second symposium on Usable privacy and security*. ACM, 2006, pp. 67–78.

[KZ10] M. Kutylowski and F. Zagórski, "Scratch, click & vote: E2E voting over the internet," in *Towards Trustworthy Elections, New Directions in Electronic Voting*, 2010, pp. 343–356.

[Lam12] P. Lambert. (2012, 6) The case of case-insensitive passwords. https://web.archive.org/web/20190310221858/https://www.zdnet.com/article/the-case-of-case-insensitive-passwords/

[LMQ14] H. Larreguy, J. Marshall, and P. Querubin, "What is the effect of turnout buying? Theory and evidence from Mexico," *Harvard University (Cambridge, MA).*, 2014.

[LFZS09] A. H. Lashkari, S. Farmand, O. B. Zakaria, and R. Saleh, "Shoulder surfing attack in graphical password authentication," *International Journal of Computer Science and Information Security – IJCSIS*, vol. 6, no. 2, 2009. http://arxiv.org/abs/0912.0951

[Las16] Lastpass, "Psychology of passwords survey," Lastpass, Tech. Rep., 2016.

[LBP51] S. Le Baron Payne, "The art of asking questions," 1951.

[Leb99] C. Lebiere, "The dynamics of cognition: An act-r model of cognitive arithmetic," *Kognitionswissenschaft*, vol. 8, no. 1, pp. 5–19, 3 1999.

[LK10] C. Lee and J. Kim, "Cancelable fingerprint templates using minutiae-based bit-strings," *Journal of network and computer applications*, vol. 33, no. 3, pp. 236–246, 2010.

[Lee14] K. Lee, "Four methods to create a secure password you'll actually remember," 2014, accessed: 2017-12-18. https://web.archive.org/web/20190123014846/https://www.lifehacker.com.au/2014/07/four-methods-to-create-a-secure-password-youll-actually-remember/

[LS09] S. Lee and K. M. Sivalingam, "An efficient one-time password authentication scheme using a smart card," *International Journal of Security and Networks*, vol. 4, no. 3, pp. 145–152, 2009.

[dLB07] K. M. M. de Leeuw and J. Bergstra, *The History of Information Security: A Comprehensive Handbook*. Elsevier Science, 2007. https://books.google.fr/books?id=pQBrsonDp6cC

[LH14] K. Lerman and T. Hogg, "Leveraging position bias to improve peer recommendation," *PloS one*, vol. 9, no. 6, 2014.

[LP07] D. K. Levine and T. R. Palfrey, "The paradox of voter participation? A laboratory study," *American political science Review*, vol. 101, no. 1, pp. 143–158, 2007.

[Lev07] J. Levitt, "The truth about voter fraud," Brennan Center for Justice, Tech. Rep., 2007. https://papers.ssrn.com/sol3/papers.cfm?abstract_id=1647224

[LK11] S. Li and A. C. Kot, "Attack using reconstructed fingerprint," in *IEEE International Workshop on Information Forensics and Security – WIFS*. IEEE, 2011, pp. 1–6.

[LHAS14] Z. Li, W. He, D. Akhawe, and D. Song, "The emperor's new password manager: Security analysis of web-based password managers," in *23rd USENIX Security Symposium*. San Diego, CA: USENIX Association, 2014, pp. 465–479. https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/li_zhiwei

[Lia19] S. Liao. (2019) ji32k7au4a83 is a surprisingly bad password. https://web.archive.org/web/20190306042434/https://www.theverge.com/platform/amp/tldr/2019/3/5/18252150/bad-password-security-data-breach-taiwan-ji32k7au4a83-have-i-been-pwned

[LJY02] E. Lim, X. Jiang, and W. Yau, "Fingerprint quality and validity analysis," in *Proceedings of the 2002 International Conference on Image Processing – ICIP*, 2002, pp. 469–472.

[Lip16] P. Lipa, "The security risks of using "forgot my password" to manage passwords," 2016, accessed: 2017-12-18. https://web.archive.org/web/20170802185615/https://www.stickypassword.com/blog/the-security-risks-of-using-forgot-my-password-to-manage-passwords/

[Lip00] T. Lipman, "The future general practitioner: out of date and running out of time." *The British journal of general practice: the journal of the Royal College of General Practitioners*, vol. 50, no. 458, pp. 743–746, 2000.

[LCGM10] G. Little, L. B. Chilton, M. Goldman, and R. C. Miller, "Exploring iterative and parallel human computation processes," in *Proceedings of the ACM SIGKDD Workshop on Human Computation*, ser. HCOMP '10. New York, NY, USA: ACM, 2010, pp. 68–76.

[Lof72] G. R. Loftus, "Eye fixations and recognition memory for pictures," *Cognitive Psychology*, vol. 3, no. 4, pp. 525–551, 1972.

[Low16] A. Low. (2016, 10) Brexit is not the will of the british people – it never has been. https://web.archive.org/web/20190216133955/https://blogs.lse.ac.uk/brexit/2016/10/24/brexit-is-not-the-will-of-the-british-people-it-never-has-been/

[MCTK10] W. Ma, J. Campbell, D. Tran, and D. Kleeman, "Password entropy and password quality," in *4th International Conference on Network and System Security*, 9 2010, pp. 583–587.

[Mai18] V. Maike, "The portrayal of Russia in US media in the aftermath of the 2016 election hacking scandal," Master's thesis, Saint Petersburg State University, 2018.

[MLRC16] E. Maiorana, D. La Rocca, and P. Campisi, "Eigenbrains and eigentensorbrains: Parsimonious bases for eeg biometrics," *Neurocomputing*, vol. 171, pp. 638–648, 2016.

[MM12] D. Malone and K. Maher, "Investigating the distribution of password choices," in *Proceedings of the 21st international conference on World Wide Web*. ACM, 2012, pp. 301–310.

[MM07] S. Marcel and J. d. R. Millán, "Person authentication using brainwaves (EEG) and maximum a posteriori model adaptation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 29, no. 4, 2007.

[Mar12] J. Marquardson, "Password policy effects on entropy and recall: Research in progress," in *Americas Conference on Information Systems*, 2012.

[MW09] W. Mason and D. J. Watts, "Financial incentives and the performance of crowds," in *Proceedings of the ACM SIGKDD workshop on human computation*. ACM, 2009, pp. 77–85.

[MMYH02] T. Matsumoto, H. Matsumoto, K. Yamada, and S. Hoshino. (2002) Impact of artificial "gummy" fingers on fingerprint systems. https://web.archive.org/web/20190301195928/https://cryptome.org/gummy.htm

[MKV+13] M. L. Mazurek, S. Komanduri, T. Vidas, L. Bauer, N. Christin, L. F. Cranor, P. G. Kelley, R. Shay, and B. Ur, "Measuring password guessability for an entire university," in *Proceedings of the 2013 ACM SIGSAC Conference on Computer Communications Security*, ser. CCS '13. New York, NY, USA: ACM, 2013, pp. 173–186.

[McC14] J. A. McCabe, "Learning and memory strategy demonstrations for the psychology classroom," 2014. http://goblues.org/faculty/professionaldevelopment/files/2012/01/McCabe-2014-Learning-Memory-Demos1.pdf

[MSH17] P. McCorry, S. F. Shahandashti, and F. Hao, "A smart contract for boardroom voting with maximum voter privacy," in *Financial Cryptography and Data Security*, A. Kiayias, Ed. Cham: Springer International Publishing, 2017, pp. 357–375.

[MR18] S. McCulley and V. Roussev, "Latent typing biometrics in online collaboration services," in *Proceedings of the 34th Annual Computer Security Applications Conference*, ser. ACSAC '18. New York, NY, USA: ACM, 2018, pp. 66–76.

[McG78] K. O. McGraw, "The detrimental effects of reward on performance: A literature review and a prediction model," *The hidden costs of reward: New perspectives on the psychology of human motivation*, pp. 33–60, 1978.

[McK18] T. McKay. (2018) Is the fbi's fitness app a cop? https://web.archive.org/web/20181108180322/https://gizmodo.com/is-the-fbis-fitness-app-a-cop-1829413814

[McK01] M. McKenna, *Building "a closet of prayer" in the new world: the story of the "Australian ballot"*. Menzies Centre for Australian Studies, 2001.

[MKS+16] W. Melicher, D. Kurilova, S. M. Segreti, P. Kalvani, R. Shay, B. Ur, L. Bauer, N. Christin, L. F. Cranor, and M. L. Mazurek, "Usability and security of text passwords on mobile devices," in *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, ser. CHI '16. New York, NY, USA: ACM, 2016, pp. 527–539.

[Mem17] N. Memon, "How biometric authentication poses new challenges to our security and privacy [in the spotlight]," *IEEE Signal Processing Magazine*, vol. 34, no. 4, pp. 196–194, 2017.

[Mer15] R. Merchant. (2015) Dashlane study: US internet users drowning in online accounts – with further tidal wave approaching. https://web.archive.org/web/20160820001652/https://www.prweb.com/releases/2015/07/prweb12860738.htm

[Mic18] E. Michel. (2018) Macron with a comfortable majority undermined by record low turnout. https://web.archive.org/web/20190507082755/https://cise.luiss.it/cise/2017/06/22/macron-with-a-comfortable-majority-undermined-by-record-low-turnout/

[Mil56] G. A. Miller, "The magical number seven, plus or minus two: some limits on our capacity for processing information." *Psychological review*, vol. 63, no. 2, p. 81, 1956.

[Mil95] W. R. Miller, "Harrison county methods: Election fraud in late nineteenth-century Texas," *Locus*, vol. 7, pp. 111–28, 1995. http://archive.today/2019.02.18-013902/http://courses.missouristate.edu/bobmiller/populism/texts/harrison_county_methods.htm

[Ms19] H. Ming-sho, *CHALLENGING BEIJING'S MANDATE OF HEAVEN: Taiwan's Sunflower Movement and Hong Kong's Umbrella Movement*. TEMPLE University Press, 2019.

[MYL14] D. Moon, J.-H. Yoo, and M.-K. Lee, "Improved cancelable fingerprint templates using minutiae-based functional transform," *Security and Communication Networks*, vol. 7, no. 10, pp. 1543–1551, Oct. 2014.

[MT79] R. Morris and K. Thompson, "Password security: A case history," *Communications of the ACM*, vol. 22, no. 11, pp. 594–597, Nov. 1979.

[MG17] T. Moura and A. Gomes, "Blockchain voting and its effects on election transparency and voter confidence," in *Proceedings of the 18th Annual International Conference on Digital Government Research*. New York, NY, USA: ACM, 2017, pp. 574–575.

[MBSS13] C. Mulliner, R. Borgaonkar, P. Stewin, and J.-P. Seifert, "Sms-based one-time passwords: attacks and defense," in *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer, 2013, pp. 150–159.

[MDAB10] S. J. Murdoch, S. Drimer, R. Anderson, and M. Bond, "Chip and PIN is broken," in *IEEE Symposium on Security and Privacy*, May 2010, pp. 433–446.

[MMD14] F. Mwagwabi, T. McGill, and M. Dixon, "Improving compliance with password guidelines: How user perceptions of passwords and security threats affect compliance with guidelines," in *47th Hawaii International Conference on System Sciences – HICSS*, vol. 00, 1 2014, pp. 3188–3197.

[NS67] R. N. Shepard, "Recognition memory for words, sentences, and pictures," *Journal of Verbal Learning and Verbal Behavior*, vol. 6, pp. 156–163, 02 1967.

[NFRE13] M. Naber, S. Frässle, U. Rutishauser, and W. Einhäuser, "Pupil size signals novelty and predicts later retrieval success for declarative memories of natural scenes," *Journal of vision*, vol. 13, no. 2, pp. 11–11, 2013.

[NM96] J. H. Nagel and J. E. McNulty, "Partisan effects of voter turnout in senatorial and gubernatorial elections," *American Political Science Review*, vol. 90, no. 4, pp. 780–793, 1996.

[NY02] M. Nakajima and Y. Yamaguchi, "Extended visual cryptography for natural images," in *The 10-th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision'2002, WSCG 2002, University of West Bohemia, Campus Bory, Plzen-Bory, Czech Republic, February 4-8, 2002*, 2002, pp. 303–310. http://wscg.zcu.cz/wscg2002/Papers_2002/A73.pdf

[Nao91] M. Naor, "Bit commitment using pseudorandomness," *Journal of Cryptology*, vol. 4, no. 2, pp. 151–158, Jan 1991. https://doi.org/10.1007/BF00196774

[NS95] M. Naor and A. Shamir, "Visual cryptography," in *Advances in Cryptology — EUROCRYPT'94*, A. De Santis, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1995, pp. 1–12.

[Nea09] T. Neale. (2009) How doctors can stay up to date with current medical information. https://web.archive.org/web/20190128025936/https://www.kevinmd.com/blog/2009/12/doctors-stay-date-current-medical-information.html

[Nic08] S. Nichter, "Vote buying or turnout buying? Machine politics and the secret ballot," *American political science review*, vol. 102, no. 1, pp. 19–31, 2008.

[NOP+06] C. M. Nielsen, M. Overgaard, M. B. Pedersen, J. Stage, and S. Stenild, "It's worth the hassle!: the added value of evaluating the usability of mobile systems in the field," in *Proceedings of the 4th Nordic conference on Human-computer interaction: changing roles*. ACM, 2006, pp. 272–280.

[NF82] D. A. Norman and D. Fisher, "Why alphabetic keyboards are not easy to use: Keyboard layout doesn't much matter," *Human Factors*, vol. 24, no. 5, pp. 509–519, 1982.

[Nor04] P. Norris, *Electoral engineering: Voting rules and political behavior*. Cambridge university press, 2004.

[Nor09] P. Norvig, "Natural language corpus data," *Beautiful Data*, pp. 219–242, 2009.

[NS71] D. Noton and L. Stark, "Scanpaths in saccadic eye movements while viewing and recognizing patterns," *Vision Research*, vol. 11, no. 9, 1971.

[Nur99] H. Nurmi, *Voting paradoxes and how to deal with them*. Springer Science & Business Media, 1999.

[OTB18] P. O'Donnell, T. J. Toohey, and P. Blechner, "Complaint for Michael Terpin v. AT&T," 2018.

[O'N14] M. E. O'Neill, "PCG: A family of simple fast space-efficient statistically good algorithms for random number generation," *ACM Transactions on Mathematical Software*, 2014.

[Obe11] J. Ober, "Wealthy hellas," *The Journal of Economic Asymmetries*, vol. 8, no. 1, pp. 1–38, 2011.

[ÖTC16] G. Öğütçü, Ö. M. Testik, and O. Chouseinoglou, "Analysis of personal information security behavior and awareness," *Computers & Security*, vol. 56, pp. 83–93, 2016.

[Ohl16] J. D. Ohlin, "Did russian cyber interference in the 2016 election violate international law," *Texas Law Review*, vol. 95, p. 1579, 2016.

[Old68] R. C. Oldfield, *Language: selected readings*. Penguin, 1968, vol. 10.

[Oli13] M. Oliver, "The social model of disability: Thirty years on," *Disability & society*, vol. 28, no. 7, pp. 1024–1026, 2013.

[vOS06] P. C. van Oorschot and S. Stubblebine, "On countering online dictionary attacks with login histories and humans-in-the-loop," *ACM Transactions on Information and System Security – TISSEC*, vol. 9, no. 3, pp. 235–258, Aug. 2006.

[OvdB04] A.-M. Oostveen and P. van den Besselaar, "Security as belief user's perceptions on the security of e-voting systems." in *Electronic Voting in Europe - Technology, Law, Politics and Society*, 01 2004, pp. 73–82.

[Orm17] H. Orman, "Secure voting: A call to arms," *IEEE Internet Computing*, vol. 21, no. 5, pp. 67–71, 2017.

[Pal05] R. Palmer. (2005) ES&S files legal action to maintain choice in Ohio's voting system selection process. https://web.archive.org/web/20190425162611/https://www.businesswire.com/news/home/20050502006105/en/ESS-Files-Legal-Action-Maintain-Choice-Ohios

[PS09] J. J. Palmieri and T. A. Stern, "Lies in the doctor-patient relationship," *Primary care companion to the Journal of clinical psychiatry*, vol. 11, no. 4, p. 163, 2009.

[PCCB16] V. M. Patel, R. Chellappa, D. Chandra, and B. Barbello, "Continuous user authentication on mobile devices: Recent progress and remaining challenges," *IEEE Signal Processing Magazine*, vol. 33, no. 4, pp. 49–61, July 2016.

[PGY16] P. Perrineau, G. Grunberg, and C. Ysmal, *Europe at the Polls: The European Elections of 1999*. Springer, 2016.

[Per02] N. Persily, "In defense of foxes guarding henhouses: the case for judicial acquiescence to incumbent-protecting gerrymanders," *Harvard Law Review*, vol. 116, no. 2, pp. 649–683, 2002.

[PLB14] N. Phetmak, W. Liwlompaisan, and P. Boonma, "Travel password: A secure and memorable password scheme," in *Intelligent Information and Database Systems: 6th Asian Conference – ACIIDS*, N. T. Nguyen, B. Attachoo, B. Trawiński, and K. Somboonviwat, Eds. Cham: Springer International Publishing, 2014, pp. 402–411.

[PJGS12] D. R. Pilar, A. Jaeger, C. F. A. Gomes, and L. M. Stein, "Passwords usage and human memory limitations: A survey across age and educational background," *PLoS One*, vol. 7, no. 12, 12 2012, pONE-D-12-21406[PII]. http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3515440/

[PS02] B. Pinkas and T. Sander, "Securing passwords against dictionary attacks," in *Proceedings of the 9th ACM Conference on Computer and Communications Security*, ser. CCS '02. New York, NY, USA: ACM, 2002, pp. 161–170.

[Pin14] M. Pinola, "Your clever password tricks aren't protecting you from today's hackers," 2014. https://web.archive.org/web/20190203093823/https://lifehacker.com/your-clever-password-tricks-arent-protecting-you-from-t-5937303

[PS00] R. Plamondon and S. N. Srihari, "Online and off-line handwriting recognition: a comprehensive survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 1, pp. 63–84, Jan 2000.

[Pli00] J. O. Pliam, "On the incomparability of entropy and marginal guesswork in brute-force attacks," in *International Conference on Cryptology in India*. Springer, 2000, pp. 67–79.

[Pom94] C. Pomerance, "The role of smooth numbers in number theoretic algorithms," in *International Congress of Mathematicians*. Citeseer, 1994.

[Poo97] R. S. Poore, "Passwords: Obsolete authenticators or cutting edge?" *Information Systems Security*, vol. 6, no. 1, pp. 10–13, 1997.

[Pop74] Pope Gregory X, "Ubi Periculum," 07 1274. https://web.archive.org/web/20190331231157/https://www.ewtn.com/library/COUNCILS/LYONS2.HTM

[Pop96] Pope John Paul II, "Universi dominici gregis on the vacancy of the apostolic see and the election of the roman pontiff," in *Apostolic Constitution*, 02 1996. https://web.archive.org/web/20190303083040/http://w2.vatican.va/content/john-paul-ii/en/apost_constitutions/documents/hf_jp-ii_apc_22021996_universi-dominici-gregis.html

[Por82] S. N. Porter, "A password extension for improved human factors," *Computers & Security*, vol. 1, no. 1, pp. 54–56, 1982.

[Pot05] B. Potter, "Are passwords dead?" *Network Security*, vol. 2005, no. 9, pp. 7–8, 2005.

[PLV+02] R. W. Proctor, M.-C. Lien, K.-P. L. Vu, E. E. Schultz, and G. Salvendy, "Improving computer security for authentication of users: Influence of proactive password restrictions," *Behavior Research Methods, Instruments, & Computers*, vol. 34, no. 2, pp. 163–169, 2002.

[Pro11] E. Protalinski. (2011) Facebook passwords are not case sensitive. https://web.archive.org/web/20180422141217/https://www.zdnet.com/article/facebook-passwords-are-not-case-sensitive-update/

[Pru15] D. Prutchi. (2015) Dolpi - two low-cost, raspi-based polarization cameras for humanitarian demining and other applications. https://web.archive.org/web/20190406042503/https://hackaday.io/project/6958-dolpi-raspi-polarization-camera/details

[Pta07] T. Ptacek. (2007) Enough with the rainbow tables: What you need to know about secure password schemes. https://web.archive.org/web/20160310152217/https://www.nccgroup.trust/us/about-us/newsroom-and-events/blog/2007/july/enough-with-the-rainbow-tables-what-you-need-to-know-about-secure-password-schemes/

[Que60] D. E. Queller, "Thirteenth-century diplomatic envoys: Nuncii and procuratores," *Speculum*, vol. 35, no. 2, pp. 196–213, 1960.

[RTB03] C. Rallings, M. Thrasher, and G. Borisyuk, "Seasonal factors, voter fatigue and the costs of voting," *Electoral Studies*, vol. 22, no. 1, pp. 65 – 79, 2003.

[RAS+17] M. S. A. N. Ranak, S. Azad, M. Safwan Fathi Bin, Z. Kamal, and M. Rahman, "An analysis on vulnerabilities of password retrying," *5th International Conference on Software Engineering & Computer System*, 2017.

[RR06] B. Randell and P. Y. A. Ryan, "Voting technologies and trust," *IEEE Security & Privacy*, vol. 4, no. 5, pp. 50–56, 2006.

[RK18] S. Rass and S. König, "Password security as a game of entropies," *Entropy*, vol. 20, no. 5, p. 312, 2018.

[Ray78] K. Rayner, "Eye movement latencies for parafoveally presented words," *Bulletin of the Psychonomic Society*, vol. 11, no. 1, pp. 13–16, 1978.

[RS06] A. Reynolds and M. Steenbergen, "How the world votes: The political consequences of ballot design, innovation and manipulation," *Electoral Studies*, vol. 25, no. 3, pp. 570–598, 2006.

[Rey17] M. Reynolds. (2017) You should ignore film ratings on imdb and rotten tomatoes. https://web.archive.org/web/20190202092552/https://www.wired.co.uk/article/which-film-ranking-site-should-i-trust-rotten-tomatoes-imdb-metacritic

[Rie16] F. Rietta. (2016) Use Bcrypt or Scrypt Instead of SHA* for Your Passwords, Please! https://web.archive.org/web/20190119220456/https://rietta.com/blog/2016/02/05/bcrypt-not-sha-for-passwords/

[RAK16] I. Rigas, E. Abdulin, and O. Komogortsev, "Towards a multi-source fusion approach for eye movement-driven recognition," *Information Fusion*, vol. 32, pp. 13–25, 2016.

[REF12] I. Rigas, G. Economou, and S. Fotopoulos, "Biometric identification based on the eye movements and graph matching techniques," *Pattern Recognition Letters*, vol. 33, no. 6, pp. 786–792, Apr. 2012.

[Rij18] D. Rijmenants. (2018) Hand ciphers. https://web.archive.org/web/20190610130334/http://users.telenet.be/d.rijmenants/en/handciphers.htm

[RO68] W. H. Riker and P. C. Ordeshook, "A theory of the calculus of voting," *The American Political Science Review*, vol. 62, no. 1, pp. 25–42, 1968. http://www.jstor.org/stable/1953324

[Riv92] R. Rivest, "The md5 message-digest algorithm," RFC, Tech. Rep., 1992.

[RS07] R. L. Rivest and W. D. Smith, "Three voting protocols: Threeballot, vav, and twin," in *Proceedings of the USENIX Workshop on Accurate Electronic Voting Technology*, ser. EVT'07. Berkeley, CA, USA: USENIX Association, 2007, pp. 16–16. http://dl.acm.org/citation.cfm?id=1323111.1323127

[Rob07] C. Roberts, "Biometric attack vectors and defences," *Computers & Security*, vol. 26, no. 1, pp. 14–25, 2007.

[RR17] J. J. Roberts and N. Rapp. (2017) Exclusive: Nearly 4 million bitcoins lost forever, new study says. https://web.archive.org/web/*/http://fortune.com/2017/11/25/lost-bitcoins/

[RLCM98] J. A. Robinson, V. W. Liang, J. A. M. Chambers, and C. L. MacKenzie, "Computer user verification using login string keystroke dynamics," *IEEE Transactions on Systems, Man, and Cybernetics - Part A*, vol. 28, no. 2, pp. 236–241, 1998.

[RZT+15] M. Rodic, X. Zhou, T. Tikhomirova, W. Wei, S. Malykh, V. Ismatulina, E. Sabirova, Y. Davidova, M. G. Tosto, J.-P. Lemelin, and Y. Kovas, "Cross-cultural investigation into cognitive underpinnings of individual differences in early arithmetic," *Developmental Science*, vol. 18, no. 1, pp. 165–174, 2015. https://onlinelibrary.wiley.com/doi/abs/10.1111/desc.12204

[RS13] D. Ron and A. Shamir, "Quantitative analysis of the full bitcoin transaction graph," in *Financial Cryptography and Data Security*, A.-R. Sadeghi, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 6–24.

[RKSP18] D. Root, L. Kennedy, M. Sozan, and J. Parshall, "Election security in all 50 states," *Center for American Progress, February*, vol. 28, 2018.

[Ros16] A. Rossberg, "Webassembly: high speed at low cost for everyone," in *ML16: Proceedings of the 2016 ACM SIGPLAN Workshop on ML*, 2016.

[Run14] M. Rundle. (2014) Bitcoin worth 4.6 million pounds is buried under a landfill in south wales. https://web.archive.org/web/20160806085246/http://www.huffingtonpost.co.uk/2013/11/27/bitcoin-buried-wales-46-million_n_4350267.html

[Rya11] P. Y. A. Ryan, "Prêt à Voter with confirmation codes," in *Electronic Voting Technology Workshop / Workshop on Trustworthy Elections – EVT/WOTE*, 2011. https://www.usenix.org/conference/evtwote-11/pr%C3%AAt-%C3%A0-voter-confirmation-codes

[RBH+09] P. Y. A. Ryan, D. Bismark, J. Heather, S. Schneider, and Z. Xia, "Prêt à Voter: a voter-verifiable voting system," *IEEE transactions on information forensics and security*, vol. 4, no. 4, pp. 662–673, 2009.

[RRI16] P. Y. A. Ryan, P. B. Rønne, and V. Iovino, "Selene: Voting with transparent verifiability and coercion-mitigation," in *Financial Cryptography and Data Security – FC, Revised Selected Papers*, 2016, pp. 176–192.

[SVK18] S. Samadi, S. Vempala, and A. T. Kalai, "Usability of humanly computable passwords," in *6th AAAI Conference on Human Computation and Crowdsourcing*, 2018.

[San18] S. Sandoval, "Friends of Wendy Noren remember her as a brilliant, dedicated woman," *KOMU 8 news*, 2018.

[SCM08] A. O. Santin, R. G. Costa, and C. A. Maziero, "A three-ballot-based secure electronic voting system," *IEEE Security & Privacy*, vol. 6, no. 3, pp. 14–21, 2008.

[Sau12] F. Sauvadet, "Rapport fait au nom de la commission des lois constitutionnelles, de la législation et de l'administration générale de la république sur la proposition de loi de M. François Sauvadet (n°107) visant à reconnaître le vote blanc aux élections," Assemblée Nationale, Tech. Rep., 2012.

[SBE09] S. Schechter, A. J. B. Brush, and S. Egelman, "It's no secret. measuring the security and reliability of authentication via "secret" questions," in *30th IEEE Symposium on Security and Privacy*. IEEE, 2009, pp. 375–390.

[Sch99] N. O. Schiller, "Masked priming of sublexical units segments vs syllables," in *Advances in phonetics : proceedings of the international phonetic sciences conference – IPS*, F. Steiner, Ed., 1999.

[Sch98] B. Schneier. (1998, 10) Memo to the amateur cipher designer. https://web.archive.org/web/20190301142015/https://www.schneier.com/crypto-gram/archives/1998/1015.html

[SJT04] R. Scully, R. W. Jones, and D. Trystan, "Turnout, participation and legitimacy in post-devolution wales," *British Journal of Political Science*, vol. 34, no. 3, pp. 519–537, 2004.

[SMK+17] S. M. Segreti, W. Melicher, S. Komanduri, D. Melicher, R. Shay, B. Ur, L. Bauer, N. Christin, L. F. Cranor, and M. L. Mazurek, "Diversify to survive: Making passwords stronger with adaptive policies," in *13th Symposium on Usable Privacy and Security – SOUPS*. Santa Clara, CA: USENIX Association, 2017, pp. 1–12. https://www.usenix.org/conference/soups2017/technical-sessions/presentation/segreti

[Sel04] T. Selker, "Process can improve electronic voting: a case study of an election," Caltech/MIT Voting Technology Project, Tech. Rep., 2004.

[SC05] T. Selker and S. Cohen, "An active approach to voting verification," Caltech/MIT Voting Technology Project, Tech. Rep., 2005.

[SGA+14] T. Selker, D. Gillette, L. Avendano, S. Hoque, K. Liu, M. Pham, and M. Vroomen, "Research in accessible voting report," US Election Assistance Commission, Tech. Rep., 2014.

[SRP06] T. Selker, E. Rosenzweig, and A. Pandolfo, "A methodology for testing voting systems," *Journal of usability studies*, vol. 2, no. 1, pp. 7–21, 2006.

[Sha49] C. E. Shannon, "Communication theory of secrecy systems," *Bell system technical journal*, vol. 28, no. 4, pp. 656–715, 1949.

[Sha18] Shape, "2018 credential spill report," Shape Security, Tech. Rep., 2018.

[SKK+12] R. Shay, P. G. Kelley, S. Komanduri, M. L. Mazurek, B. Ur, T. Vidas, L. Bauer, N. Christin, and L. F. Cranor, "Correct horse battery staple: Exploring the usability of system-assigned passphrases," in *Proceedings of the 8thsymposium on usable privacy and security*. ACM, 2012, p. 7.

[SKD+14] R. Shay, S. Komanduri, A. L. Durity, P. S. Huh, M. L. Mazurek, S. M. Segreti, B. Ur, L. Bauer, N. Christin, and L. F. Cranor, "Can long passwords be secure and usable?" in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '14. New York, NY, USA: ACM, 2014, pp. 2927–2936.

[SKD+16] R. Shay, S. Komanduri, A. L. Durity, P. S. Huh, M. L. Mazurek, S. M. Segreti, B. Ur, L. Bauer, N. Christin, and L. F. Cranor, "Designing password policies for strength and usability," *ACM Transactions on Information and System Security – TISSEC*, vol. 18, no. 4, pp. 1–34, May 2016.

[SKK+10] R. Shay, S. Komanduri, P. G. Kelley, P. G. Leon, M. L. Mazurek, L. Bauer, N. Christin, and L. F. Cranor, "Encountering stronger password requirements: User attitudes and behaviors," in *Proceedings of the 6th Symposium on Usable Privacy and Security*, ser. SOUPS '10. New York, NY, USA: ACM, 2010, pp. 1–20.

[She15] K. Sherwin, "Password creation: 3 ways to make it easier," 2015, accessed: 2017-12-18. https://web.archive.org/web/20181030222401/https://www.nngroup.com/articles/password-creation/

[SLMM09] S. W. Shin, M.-K. Lee, D. Moon, and K. Moon, "Dictionary attack on functional transform-based cancelable fingerprint templates," *ETRI journal*, vol. 31, no. 5, pp. 628–630, 2009.

[Sim04] B. Simons, "Electronic voting systems: The good, the bad, and the stupid," *Queue*, vol. 2, no. 7, pp. 20–26, Oct. 2004.

[SA11] S. Singh and G. Agarwal, "Integration of sound signature in graphical password authentication system," *International Journal of Computer Applications*, vol. 12, no. 9, pp. 11–13, 2011.

[SS13] Y. N. Singh and S. K. Singh, "A taxonomy of biometric system vulnerabilities and defences," *International Journal of Biometrics*, vol. 5, 01 2013.

[SHA+13] Y. Sintomer, C. Herzberg, G. Allegretti, A. Röcke, and M. L. Alves, "Participatory budgeting worldwide," *Dialog Global*, no. 25, pp. 1–93, 2013.

[SE07] N. Sklavos and C. Efstathiou, "SecurID authenticator: On the hardware implementation efficiency," in *14th IEEE International Conference on Electronics, Circuits and Systems – ICECS*. IEEE, 2007, pp. 589–592.

[Sle03] S. M. Sled, "Vertical proximity effects in the California recall election," Caltech/MIT Voting Technology Project, Tech. Rep., 2003.

[SRRM16] I. Sluganovic, M. Roeschlin, K. B. Rasmussen, and I. Martinovic, "Using reflexive eye movements for fast challenge-response authentication," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '16. New York, NY, USA: ACM, 2016, pp. 1056–1067.

[SWL15] D. F. Smith, A. Wiliem, and B. C. Lovell, "Face recognition on consumer devices: Reflections on replay attacks," *IEEE Transactions on Information Forensics and Security*, vol. 10, pp. 736–745, 2015.

[Spr11] M. Sprengers, "Gpu-based password cracking," Master's thesis, Radboud University Nijmegen, 2011.

[SFD+14] D. Springall, T. Finkenauer, Z. Durumeric, J. Kitcat, H. Hursti, M. MacAlpine, and A. J. Halderman, "Security analysis of the estonian internet voting system," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2014, pp. 703–715.

[Sta11] F. Stajano, "Pico: No more passwords!" in *International Workshop on Security Protocols*. Springer, 2011, pp. 49–81.

[Ste12] A. Stenton, "The contribution of the computer to improving L2 oral production. An examination of the applied and theoretical research behind the SWANS authoring programme," *Etudes en Didactique des Langues*, no. 19, 2012.

[SBK+17] M. Stevens, E. Bursztein, P. Karpman, A. Albertini, and Y. Markov, "The first collision for full sha-1," in *Advances in Cryptology – CRYPTO*, J. Katz and H. Shacham, Eds. Cham: Springer International Publishing, 2017, pp. 570–596.

[SI06] C. Stewart III, "Residual vote in the 2004 election," *Election Law Journal*, vol. 5, no. 2, pp. 158–169, 2006.

[SL12] G. Stewart and D. Lacey, "Death by a thousand facts: Criticising the technocratic approach to information security awareness," *Information Management & Computer Security*, vol. 20, no. 1, pp. 29–38, 2012.

[Str06b] C. E. Strauss, "A critical review of the triple ballot voting system, part 2: Cracking the triple ballot encryption," 2006. https://www.cs.princeton.edu/~appel/voting/Strauss-ThreeBallotCritique2v1.5.pdf

[Str06a] C. E. Strauss, "A critical review of the triple ballot (3ballot) scheme, part 1," 2006. https://www.cs.princeton.edu/~appel/voting/Strauss-TroubleWithTriples.pdf

[SCL12] H.-M. Sun, Y.-H. Chen, and Y.-H. Lin, "oPass: A user authentication protocol resistant to password stealing and password reuse attacks," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 2, pp. 651–663, 2012.

[SB12] S.-T. Sun and K. Beznosov, "The Devil is in the (Implementation) Details: An Empirical Analysis of OAuth SSO Systems," in *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, ser. CCS '12. New York, NY, USA: ACM, 2012, pp. 378–390.

[SS15] A. Sundström and D. Stockemer, "Regional variation in voter turnout in Europe: The impact of corruption perceptions," *Electoral Studies*, vol. 40, pp. 156–169, 2015.

[TAM+13] M. M. Taha, T. A. Alhaj, A. E. Moktar, A. H. Salim, and S. M. Abdullah, "On password strength measurements: Password entropy and password quality," in *International Conference on Computing, Electrical and Electronics Engineering – ICCEEE*. IEEE, 2013, pp. 497–501.

[TBA14] M. A. Tariq, J. Brynielsson, and H. Artman, "The security awareness paradox: A case study," in *Proceedings of the 2014 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, ser. ASONAM '14. Piscataway, NJ, USA: IEEE Press, 2014, pp. 704–711. http://dl.acm.org/citation.cfm?id=3191835.3191974

[Tay09] J. G. Taylor, "Cognitive computation," *Cognitive Computation*, vol. 1, no. 1, pp. 4–16, 3 2009.

[Tef17] P. Teffer. (2017) Build trust before you introduce e-voting, says estonian president. https://web.archive.org/web/20180917074334/https://euobserver.com/digital/138394

[TAT07] U. Topkara, M. J. Atallah, and M. Topkara, "Passwords decay, words endure: Secure and re-usable multiple password mnemonics," in *Proceedings of the 2007 ACM Symposium on Applied Computing*, ser. SAC '07. New York, NY, USA: ACM, 2007, pp. 292–299.

[Top10] A. Toponce, "Password cards," 2010, accessed: 2017-12-18. https://web.archive.org/web/20121101053014/http://pthree.org/2010/09/21/password-cards/

[Top11] A. Toponce, "Strong Passwords NEED Entropy," 2011, accessed: 2017-12-18. https://web.archive.org/web/20180223215746/https://pthree.org/2011/03/07/strong-passwords-need-entropy/

[Tra18] D. S. Tran, *Unrigging American Elections: Reform Past and Prologue*, ser. Elections, Voting, Technology. Springer International Publishing, 2018. https://books.google.fr/books?id=gcl-DwAAQBAJ

[THP+05] M. W. Traugott, M. J. Hanmer, W.-H. Park, P. S. Herrnson, R. G. Niemi, B. B. Bederson, and F. G. Conrad, "The impact of voting systems on residual votes, incomplete ballots, and other measures of voting behavior," in *Annual Conference of the Midwest Political Science Association*. Citeseer, 2005, pp. 7–10.

[TL12] G. Tucker and C. Li, "Cloud computing risks," in *Proceedings on the International Conference on Internet Computing – ICOMP*, 2012, p. 1.

[UAA+17] B. Ur, F. Alfieri, M. Aung, L. Bauer, N. Christin, J. Colnago, L. F. Cranor, H. Dixon, P. Emami Naeini, H. Habib, N. Johnson, and W. Melicher, "Design and evaluation of a data-driven password meter," in *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, ser. CHI '17. New York, NY, USA: ACM, 2017, pp. 3775–3786.

[UBS+16] B. Ur, J. Bees, S. M. Segreti, L. Bauer, N. Christin, and L. F. Cranor, "Do users' perceptions of password security match reality?" in *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, ser. CHI '16. New York, NY, USA: ACM, 2016, pp. 3748–3760.

[UNB+15] B. Ur, F. Noma, J. Bees, S. M. Segreti, R. Shay, L. Bauer, N. Christin, and L. F. Cranor, "I added '!'at the end to make it secure": Observing password creation in the lab," in *Proceedings of the 11th symposium on usable privacy and security*, 2015.

[UB14] Y. Uzunay and K. Bicakci, "Trusted3ballot: Improving security and usability of three ballot voting system using trusted computing," in *5th International Conference on Intelligent Systems, Modelling and Simulation – ISMS*. IEEE, 2014, pp. 534–539.

[Van10] A. Vance. (2010) If your password is 123456, just make it hackme. https://web.archive.org/web/20181023160454/https://www.nytimes.com/2010/01/21/technology/21password.html

[VSV+16] K. Vassil, M. Solvak, P. Vinkel, A. H. Trechsel, and M. R. Alvarez, "The diffusion of internet voting. Usage patterns of internet voting in Estonia between 2005 and 2015," *Government Information Quarterly*, vol. 33, no. 3, pp. 453–459, 2016.

[Vin15] P. Vinkel, *Remote electronic voting in Estonia: legality, impact and confidence*. TUT Press, 2015.

[WSH+18] B. Wang, J. Sun, Y. He, D. Pang, and N. Lu, "Large-scale election based on blockchain," *International conference on identification, information and knowledge in the internet of things*, vol. 129, pp. 234–237, 2018.

[WY05] X. Wang and H. Yu, "How to break md5 and other hash functions," in *Advances in Cryptology – EUROCRYPT*, R. Cramer, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 19–35.

[WRBW16] R. Wash, E. Rader, R. Berman, and Z. Wellmer, "Understanding password choices: How frequently entered passwords are re-used across websites," in *12th Symposium on Usable Privacy and Security – SOUPS*. Denver, CO: USENIX Association, 2016, pp. 175–188. https://www.usenix.org/conference/soups2016/technical-sessions/presentation/wash

[WDCJ09] C. S. Weir, G. Douglas, M. Carruthers, and M. Jack, "User perceptions of security, convenience and usability for ebanking authentication tokens," *Computers & Security*, vol. 28, no. 1-2, pp. 47–62, 2009.

[WPPS05] T. Westeyn, P. Pesti, K.-h. Park, and T. Starner, "Biometric Identification using Song-Based Blink Patterns Blinking as a Biometric," in *HCI International*, 2005.

[WM11] M. E. Whitman and H. J. Mattord, *Principles of Information Security*, 4th ed. Boston, MA, United States: Course Technology Press, 2011.

[Whi18] Z. Whittaker. (2018) Github says bug exposed some plaintext passwords. https://web.archive.org/web/20190331110732/https://www.zdnet.com/article/github-says-bug-exposed-account-passwords/

[WT99] A. Whitten and J. D. Tygar, "Why Johnny can't encrypt: A usability evaluation of PGP 5.0," in *Proceedings of the 8th Conference on USENIX Security Symposium - Volume 8*, ser. SSYM'99. Berkeley, CA, USA: USENIX Association, 1999, pp. 14–14. http://dl.acm.org/citation.cfm?id=1251421.1251435

[WWB+05] S. Wiedenbeck, J. Waters, J.-C. Birget, A. Brodskiy, and N. Memon, "Passpoints: Design and longitudinal evaluation of a graphical password system," *International journal of human-computer studies*, vol. 63, no. 1-2, pp. 102–127, 2005.

[WZ14] F. Wiemer and R. Zimmermann, "High-speed implementation of bcrypt password search using special-purpose hardware," in *International Conference on ReConFigurable Computing and FPGAs – ReConFig*. IEEE, 2014, pp. 1–6.

[WCD+17] J. Woodage, R. Chatterjee, Y. Dodis, A. Juels, and T. Ristenpart, "A new distribution-sensitive secure sketch and popularity-proportional hashing," in *Advances in Cryptology – CRYPTO*, J. Katz and H. Shacham, Eds. Cham: Springer International Publishing, 2017, pp. 682–710.

[YG09] R. Yampolskiy and V. Govindaraju, "Taxonomy of behavioural biometrics," *Behavioral Biometrics for Human Identification: Intelligent Applications*, 01 2009.

[YBAG04] J. Yan, A. Blackwell, R. Anderson, and A. Grant, "Password memorability and security: Empirical results," *IEEE Security and Privacy*, vol. 2, no. 5, pp. 25–31, Sep. 2004.

[Yan01] J. J. Yan, "A note on proactive password checking," in *Proceedings of the 2001 Workshop on New Security Paradigms*, ser. NSPW '01. New York, NY, USA: ACM, 2001, pp. 127–135.

[YLC+16] W. Yang, N. Li, O. Chowdhury, A. Xiong, and R. W. Proctor, "An empirical study of mnemonic sentence-based password generation strategies," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '16. New York, NY, USA: ACM, 2016, pp. 1216–1229.

[Yil17] M. Yildirim, "Security and usability in password authentication," phdthesis, University of Sussex, 2017.

[YPR10] Y. Yue, R. Patel, and H. Roehrig, "Beyond position bias: Examining result attractiveness as a source of presentation bias in clickthrough data," in *Proceedings of the 19th international conference on World Wide Web*. ACM, 2010, pp. 1011–1018.

[YAS10] A. Yuksel, L. Akarun, and B. Sankur, "Biometric identification through hand vein patterns," in *International Workshop on Emerging Techniques and Challenges for Hand-Based Biometrics*, Aug 2010, pp. 1–6.

[ZCC+13] F. Zagórski, R. Carback, D. Chaum, J. Clark, A. Essex, and P. L. Vora, "Remotegrity: Design and use of an end-to-end verifiable remote voting system," in *Applied Cryptography and Network Security - 11th International Conference – ACNS*, 2013, pp. 441–457.

[ZVJ09] G. Zayaraz, V. Vijayalakshmi, and D. Jagadiswary, "Securing biometric authentication using DNA sequence and Naccache Stern Knapsack cryptosystem," in *International Conference on Control, Automation, Communication and Energy Conservation*, June 2009, pp. 1–4.

[Zha17] B. Zhang, "Random sample voting implementation technical report," Random Sample Voting Project, Tech. Rep., 2017.

[ZMR10] Y. Zhang, F. Monrose, and M. K. Reiter, "The security of modern password expiration: An algorithmic framework and empirical analysis," in *Proceedings of the 17th ACM conference on Computer and communications security*. ACM, 2010, pp. 176–186.

[ZXL+12] Y. Zhang, P. Xia, J. Luo, Z. Ling, B. Liu, and X. Fu, "Fingerprint attack against touch-enabled devices," in *Proceedings of the Second ACM Workshop on Security and Privacy in Smartphones and Mobile Devices*, ser. SPSM '12. New York, NY, USA: ACM, 2012, pp. 57–68.

[ZCPR03] W.-Y. Zhao, R. Chellappa, P. J. Phillips, and A. Rosenfeld, "Face recognition: A literature survey," *ACM Computing Surveys*, vol. 35, pp. 399–458, 12 2003.

[ZH90] M. Zviran and W. J. Haga, "Cognitive passwords: The key to easy access control," *Computers & Security*, vol. 9, no. 8, pp. 723–736, 1990.