Usable everlasting encryption using the pornography infrastructure

Nikola K. Blanchard[1] and Siargey Kachanovich[2]

[1] Digitrust, Loria, Université de Lorraine Nikola.K.Blanchard@gmail.com
www.koliaza.com
[2] Université Côte d'Azur, INRIA Sophia-Antipolis, France

Abstract. Nine years before Snapchat and its ephemeral messages, Aumann, Ding, and Rabin introduced the idea of everlasting security: an encryption that could not be decrypted after a certain date, no matter the adversary's computing power. Their method is efficient but not adapted to real-life constraints and cannot effectively be used today. This paper proposes a new solution that makes use of the already existing communications from pornography distribution networks. The method proposed has the advantage of being usable off-the-shelf by individuals with limited technical skills.

Keywords: Usable security · Random beacon · Bounded memory model · One-time pad

1 Introduction

Between January 5th, 2010 and February 3rd, 2010, Chelsea Manning downloaded more than 491 000 confidential documents, smuggled them through security hidden in a Lady Gaga CD, then onto an SD card hidden in a camera, before sending them to WikiLeaks through Tor [Mad13]. Later in 2016, an anonymous[3] whistle-blower nicknamed "John Doe" managed to communicate 2.6TB of confidential financial documents — known as the Panama Papers — to the International Consortium of Investigative Journalists [OOWJ16]. In both events, a more attentive supervisor could probably have prevented or at least detected the leak as it was happening. The methods used by Chelsea Manning are most probably by now obsolete, which poses the question of the technological tools available to the whistle-blowers. An issue is that the whistle-blowing attempts of highest importance come from agents within large institutional and corporate actors, which are best suited to detect leaks by storing and decrypting their agents' communications.

Irrespective of the ethical considerations, it then seems natural to look for methods, through which an agent within a large organisation could exfiltrate —

[3] The whistle-blower remained anonymous through the usage of Tor and limited communications.

ideally to report criminal activities. This agent's main issue then becomes the communication of data, while being watched by an overpowering adversary.

There are encryption methods available today that are currently impossible to decrypt without a key — e.g. any of the current asymmetric encryption standards[4]. For a whistle-blower, however, asymmetrical cryptography cannot be considered entirely secure, as the current methods all depend on computational hardness of certain problems, and this hardness is still just an assumption. An adversary capable of storing the encrypted messages for an arbitrary duration could theoretically either spend the computational resources when they are cheaper, or wait for the development of more efficient decryption algorithms. Furthermore, due to Shannon's theorems, we know that the only way to avoid a leak of information is to use a one-time pad with the same length as the original message [Sha49]. We then require an external source to obtain a one-time pad.

Previous work. In an article written in 2002 [ADR02], Aumann, Ding and Rabin introduced the first everlasting encryption algorithm, where an adversary that cannot decrypt the message in a bounded time after its emission will have an exponentially small probability of being able to decrypt it afterwards. The peculiarity of their model is that it holds even against an adversary that obtains unbounded computational capabilities right after the cut-off time. For this, the authors make two central assumptions:

- There exists a common source of public random bits.
- No adversary can store more than a constant fraction of these random bits.

This seems quite reasonable, as it corresponds to the existence of a common source of public random bits that has a very high throughput to avoid being stored. However, implementing it in practice reveals a few challenges. An idea initially mentioned in the original paper was to create a constellation of satellites that beam random bits at a high rate. This is alas costly, and not directly usable by the average user. An alternative proposed by the same authors was to access and compress a large amount of textual web pages, but this creates new problems.

Contributions. Our goals in this paper are three-fold. First, we formalise the objections above in an agent model. Second, we propose a list of alternative entropy sources and show that the choice is in fact quite limited. We show that we can use socially generated data as a primitive for cryptography and that the main candidate seems to be pornography. We also detail how to use it in an appropriate way. Finally, we look at the social implications of using such a system, and why it has advantages beyond the simple cryptographic ones. One central advantage comes from the very taboo surrounding pornography, making this choice a strength of the protocol.

[4] Notwithstanding side-channel attacks, infected client machines and the possibility that an actor could have secretly found an algorithm to polynomially solve supposed hard problems.

2 Model and intuition

Our goal is to study a model that is adapted to real world use — for example two agents in different organisations wanting to exchange securely. Keeping our introductory example, it could be an agent of a government trying to exfiltrate some files in a whistle-blowing attempt by communicating them to a journalist.

2.1 Agents

We consider four agents:

- Alice is the agent with the data, who wants to send data to Bob without getting caught or attracting too much attention.
- Bob just wants to receive Alice's data.
- Eve is the supervisor of Alice as well as many other agents, and wants to prevent data leaks. Eve has large capabilities: she can intercept all messages from her agents, and store all the encrypted ones for potential later decryption. However, she cannot store all the information exchanged in clear because of size constraints. Eve also monitors the information exchanged in clear and tries to detect anomalous patterns.
- Carlos represents the rest of the internet.

Here, realistically, Eve could be the head of Alice's organisation. She would then be able to impose rules such that most of the communications have to be in clear-text — or easily decrypted — with some allowances for higher security to protect her agents' privacy, at the cost of storing those for future use if she has any hint of wrong-doing. Eve stores every communication between Alice and Bob, and inspects all the exchanges between Alice and Carlos, but can't store them as long as Alice is statistically indistinguishable from her colleagues. Eve possibly has the capability of decrypting some encrypted messages, but this requires time and is expensive.

We could also consider a slightly more complex model where Bob also has a supervisor. Because of symmetries in the protocol, both models are close to equivalent and we focus on the simpler one.

2.2 Intuition behind the protocol

In such a model, a very simple protocol is available. Using asymmetric encryption — which we can initially assume to be secure for at least a limited time — Alice sends a set of pseudo-random numbers to Bob. Those are pointers to a common source of random bits held by Carlos. After they confirm having the same data — e.g. by the use of a checksum — Alice and Bob can exchange using the equivalent of a one-time pad. As long as Eve doesn't store the whole data required, she won't be able to decrypt Alice's messages.

The hard task is then to find a good and discreet source of random bits. Alternatively, a public source of low-entropy bits can also work as long as a randomness extractor is used, as in the original paper [ADR02].

3 Entropy sources

3.1 Constraints

Before listing potential sources, we have to look at the constraints we're facing.

Throughput. The first constraint is quite simple: the data throughput has to be large enough to make storing it be impossible, or at least extremely expensive. In practice, this evolves with the hardware costs, but 1 TB/s is a good initial target to counter all but the largest state actors, as the global cloud storage increases at the rate of 8TB/s [Cis18a]. A smaller throughput could still be secure — storing it becoming only extremely expensive — while 10TB/s would be more than enough for the foreseeable future.

Canonicity. The second constraint is the canonical nature of the entropy source. As Alice and Bob agree on a set of pointers, these pointers need to target the same data stream, and both agents must obtain the same data when they try to access it.

Accessibility. The third constraint is that Alice and Bob should be able to access Carlos' random bits, following Alice's pointers, but in a way that do not set Alice or Bob apart from their colleagues. As such, the data should be common enough, in the sense that it is accessed on a regular basis by a large number of people.

3.2 Original sources

The original paper investigated two main sources of entropy: satellite-based random beacons, and random web-page compression [ADR02]. Both have some weaknesses, however, which we will mention before looking at potential replacements.

Satellites. The first entropy source imagined relies on a satellite — or a small constellation of satellites — that beams random bits. The source being unique, canonicity is evident. An issue is that, as storage cost decreased exponentially since the original proposal, a single satellite is far from being an option. Using SpaceX's Starlink project as an example of satellite constellation, a total of 1600 satellites would be required to achieve the required bound [dPCC18], for a total cost above one billion dollars[5]. Depending on whether the satellite system is only used for this purpose, the specialised receiving equipment — which could be costly — could set Alice apart and make her obvious to Eve. Thus, neither throughput nor accessibility is achieved.

[5] Moreover, one could wonder about the entities capable of funding such a system have any interest in doing so.

Random beacons. Although not mentioned in the original paper as it didn't exist yet, a simpler possibility would be to use a public random beacon, such as the NIST randomness beacon [FIP11]). This source only produces 512 bits per minute today, but its throughput could be increased. An issue is that, even if it achieved 1TB/s of random bits, the total demand from clients accessing the beacon would probably be much lower than that in practice. As such, Eve could simply store all the random bits requested from the beacon. A coordinated effort could be done to spam the system with bogus requests, but seems unlikely to succeed[6]. Moreover, accessing such a service — or going through an anonymising service such as Tor — would make Alice suspect.

Web page compression. A second original method goes by accessing random web pages, compressing them, and using this data as a one-time pad. A naive implementation of this method already fails for simple throughput reasons: considering only pages indexed by Google and ignoring — for now — multimedia content, the total throughput is far from enough. The total index size of Google, for example, is still storable by a powerful adversary [vdBBdK16]. This method also requires both agents to agree on a large set of web pages and to have common access to them (without local variability of content due to redirections, which can be hard to foresee. Accessing these pages might trigger Eve's detection mechanism because of their sheer quantity and potential lack of pattern.

3.3 Multimedia sources

As the entropy sources mentioned all have inherent issues, the obvious solution is to use the multimedia content present online. Specifically, one can use video sources, as they comprise more than half of the 500TB/s of internet throughput [Cis18b]. We can consider two kinds of traffic: *upstream*, with the communication going from a computer to the network. This is contrasted with *downstream*, which means the communication from the network to computers. Here, we must be careful, as most of the downstream traffic is many-to-one, with the same content being distributed to many clients from a single source. Youtube, for example, represents more than 11% of global downstream traffic but is still quite storable — and is stored in practice — as its total sizes only increases at a rate of 40GB/s [Cul18]. We are then left with multiple candidates, all in the same category: many-to-many video streaming services with many different sources of content. We will focus mainly on the upstream throughput, as it is one-to-many and hence presents higher entropy.

Twitch. Twitch, a streaming platform with a focus on video-game streaming, is the first candidate. It has a sufficiently high throughput, with more than 5% of all upstream traffic and 2.2 million active content creators each month [Cul18]. This could be good enough, but Aumann *et al.*'s original model mentions a

[6] Akin to the famous but fruitless attempt made against the ECHELON system [Dyk01].

problem with this kind of source: the adversary can modify the data before storing it[7]. As it turns out, Twitch data being mostly video-game live-stream from a few major video-games, it can be theoretically compressible with extreme compression ratios, as long as Eve has a highly advanced data model. Because of this, it fails to achieve sufficiently high throughput for our purposes.

Video chat. Our second candidate lies in direct video chats and calls, corresponding to Skype, WhatsApp and competing services. This has a large throughput — 8% of upstream internet traffic — and isn't easily compressible without high losses. However, it suffers from two problems. First, it is generally not accessible to people outside the call, unless they have advanced surveillance capabilities, making it fail the accessibility constraint. Second, they are distributed and make it hard to create any canonical indexing.

Pornography . Our last candidate, as strange as it seems, is live pornography. Besides its non-technical advantages, it is the first candidate to truly satisfy all our constraints. It is hard to estimate its throughput accurately, but first order approximations seem to exceed our expectations. At least 4% of Google search requests concern pornography, and the largest live pornography web site (livejasmin.com) is consistently ranked in the top 50 most visited web sites worldwide — behind two other pornography web sites, according to Amazon Alexa. By itself — and it is only one of many alternatives — this web site already has an upstream throughput counted in GB/s, with thousands of performers at any point[8], generally with high definition streams [Pre17]. It is also easily accessible — sometimes at a cost on paying sites — all that is left is making it canonical.

4 Protocol to use the live pornography infrastructure

4.1 Protocol overview

We have found a high-throughput, hard-to-compress and accessible source, but Alice and Bob still need to agree on the indexing. Here is one potential protocol to address this, with details on each part shown afterwards:

1. Alice sends Bob an initial message using any asymmetric encryption system. This messages contains the parameters for the data, in the form of a set of n complex pointers. Each pointer has data corresponding to a web site[9], the

[7] This is in opposition to Maurer, where the adversary with bounded memory can only choose to store or discard data, instead of storing the result of computations done on this data, potentially to compress it [Mau92,ADR02].

[8] This does not even address response streams from viewers, which can increase the total throughput by one or two orders of magnitude if we manage to make them canonical.

[9] A single web site could be used, making this first element obsolete, but it is safer to include it to allow for a diversity of web sites, making it more adaptable and giving the system access to a larger throughput.

index of a video stream on that web site, a temporal marker for the start of the stream and a duration. The pointers do not correspond to existing videos but to streams in the near future. The time delay depends on the agents' constraints, going from 5 minutes to a few days.
2. Alice and Bob both record the streams pointed to, and extract entropy from those by taking a common start time and using a randomness extractor.
3. They obtain n streams of similar length, and truncate these to equalise the lengths.
4. If Alice only managed to download n' streams instead of n, she still applies the protocol with n' streams instead.
5. Alice computes a parity stream by XORing her n streams and sets it aside to send to Bob. This will allow Bob to have some redundancy in the case of one stream being not correctly acquired.
6. Alice splits her n streams into blocks of small equal length. She then hashes all the i-th blocks together and concatenates them to obtain a one-time pad.
7. Alice hashes each of her streams independently to get n checksums.
8. Alice sends a message to Bob containing the checksums, the encrypted message and the parity stream she set aside.
9. As Bob is supposed to have download the same streams, he checks the hashes to ensure that they are indeed equal to Alice's. If Bob is missing one of the streams, he recomputes them using the parity stream.
10. Bob hashes the streams block-wise — exactly as Alice did — to obtain the same one-time pad and decrypt Alice's message.

This protocol has the advantage of requiring only two rounds of one-sided communications from Alice to Bob, with a delay between the two to have the time to record the streams. It also integrates some fault tolerance, as bugs are inevitable. However, it glosses over the pointers, which we must now investigate.

4.2 Making a canonical pointer

As we stated above, each pointer is composed of four elements: a web site URL, the duration of the video to record, the index of a stream on that web site, and a frame at which both agents are supposed to start recording within that stream. The web site's URL and the duration of the video are easy to define. For instance, we can assume for simplicity that each web site is well-defined by its URL [10]. The other two elements — stream index and starting frame — require more work.

Stream index. The index of a stream on the web site is harder to agree on, as the stream is not well-defined at the time Alice sends her message. Even worse, streams are generally ordered in a variable way. For example, because Alice's and Bob's accesses are asynchronous, some streams may disappear in the interval between, hence changing the order of the streams. As such, we must

[10] URLs can change depending on countries because of redirections, but the underlying streams tend to be the same.

give a pointer that statistically will point towards a single stream, even if both agents do not look it up simultaneously. Finally, the probability of selecting a stream from all streams should be as close as possible to uniform to maintain the throughput guarantees.

One potential solution could be for Alice to select a large number x and send it to Bob along with the pointer. If the web site has k streams when Alice accesses it, she selects the stream number ($x \bmod k$). Due to the strong variability in the number of streams, this still fails with high probability. We can then pick a large constant c and take the stream number $\left(x \bmod \left\lfloor \frac{k}{c} \right\rfloor\right)$. By taking $\left\lfloor \frac{k}{c} \right\rfloor$ instead of k, we reduce the probability of it changing between the two different accesses.

The number of streams tends to be relatively stable on a given web site, with the previous method being able to absorb most of the minor variability. However, this is still not sufficient for our purpose, as the order between two streams can change much faster than the number of streams. This is especially true when the streams are ordered by number of viewers. A solution is then to choose a second criterion that is independent of the one being used for the sorting. Both Alice and Bob then select the first stream in the list after number $\left(x \bmod \left\lfloor \frac{k}{c} \right\rfloor\right)$ that satisfies this criterion[11].

Starting frame. Alice and Bob require the exact same streams for Bob to be able to decrypt the message. This means that they must agree on a starting frame for the video. In practice, they can simply record and download a certain quantity of video, and then manually choose a starting frame. This means that they can easily agree on the start time with a margin of a few seconds, without having to agree on a shared clock beforehand[12].

Along with her pointer, Alice then sends a random value — a nonce — and both she and Bob hash frames from the video until they get a hash whose first bits coincide with the random value. By setting an appropriate precision for this nonce, they can get the same start frame with high probability even if they started recording at slightly different times. However, this means that the delay between the times when Alice and Bob start recording has to be at least one order of magnitude smaller than the duration they expect to record[13]. In practice, assuming the delay in accessing the stream is less than 30 seconds, recording 10 minutes of video is more than enough.

With a canonical stream, a set duration and a starting frame, Alice and Bob can both extract entropy from the stream (or losslessly compress them with a good algorithm). By setting $n = 10$ and sending a single parity stream, we already have good guarantees: even if Eve manages to store 10% of all streams, she still has a probability at most 9×10^{-9} of being able to decrypt the message.

[11] If we are looking at response streams, we can use this method recursively.

[12] This would be made even harder by the fact that Alice and Bob can have different latencies.

[13] A similar method could be used for the end time, but getting a duration is safer and increases the chance of all n streams having broadly similar bit-length.

5 Social aspects

In addition to its cryptographic feasibility, the protocol that we showed has multiple advantages that are not directly technical.

Immediate employability. The first advantage of the protocol is that it can be used directly, without advanced tools or the creation of a large source of entropy. With the values shown previously, anyone could send a secure email to their interlocutor, with a list of web sites, stream indices, different times to access them, duration and nonces. As the system can tolerate a 30 second difference, it is doable manually without requiring automation of any task. The only step left is agreeing on a starting frame and computing the one-time pad, which can be done using off-the-shelf tools.

Plausible deniability. A second advantage is entirely dependent on being based on pornography, and the fact that pornography is still seen as potentially shameful, but socially tolerated in Western societies. The very shame associated with it for most people gives Alice plausible deniability. Even if Eve catches her downloading the streams onto her work computer, Alice can plead that she was only doing it for their intrinsic interest, and not as a tool to exfiltrate secrets. This is statistically true, as 57% of respondents to an online study admitted to having accessed pornography web sites from their office [McD18].

Firing all the people caught doing so because of security risks would be costly to Eve because of the sheer number of false positives. Any suspicious behaviour of Alice — such as hiding a USB key on which the pornography streams are recorded — becomes partially justified without incriminating Alice further than for the lighter offence of watching pornography at work.

Public reaction to surveillance. Finally, although government surveillance of citizens' online activity is now partially tolerated, the public is much more critical of it when it comes to intimate subjects. The public outcry after Edward Snowden's revelations that government agencies stored and accessed sensitive personal data is an example of this [Hil14]. This strongly negative image would make it harder for most countries' agencies to receive the massive funding required to store even a portion of the streams considered, especially in Europe after the implementation of new directives on the right to be forgotten.

6 Issues and extensions

Encoding variability. One issue with the protocol we proposed is that the video stream can be different due to variations in the stream's encoding. There are multiple ways to address this depending on Alice and Bob's respective constraints. The first is to hash the received frames (by groups of a few dozen frames), exchange the hashes, and only keep the common ones. However, this requires an additional intermediate round of communication going from Bob to Alice. An

alternative is to use locality-sensitive hashing (LSH) [SC08] to ensure that the stream is similar on both ends. This strongly reduces entropy, but maintains the 2-round one-sided communication structure.

Improving stream agreement. With the proposed protocol, if the number of streams k changes between the different access times, the probability of disagreement is $\frac{1}{c}$. This can be further improved at a small cost to Alice and Bob. Instead of checking k upon getting to the web site, they instead refresh the counter and track its evolution for a minute. Many methods then become available, but to limit ourselves to the simplest available to Alice and Bob, they can just keep track of the maximum and minimum reached by k, and then use stream $\left(x \bmod \left\lfloor \frac{k_{\max}+k_{\min}}{2c} \right\rfloor\right)$. This increases their agreement probability to at least $\gamma^2 \times \frac{c-1}{c}$, where γ is the proportion of overlap on the time they spent looking at the evolution of k.

Increasing the fault tolerance. The protocol can only correct one missing stream in its current state. It can potentially be extended to tolerate more erroneous or missing streams. Thanks to the checksums sent by Alice, Bob can eliminate the erroneous streams — or even try to fix them, which can be done with probability $\frac{1}{2}$ if Bob's starting frame is different from Alice's, but he recorded a bit more than the expected duration. We can then reduce the problem to that of fixing missing streams instead of erroneous ones. To go further, one could use double error correction as in RAID 6 and derivative systems [JJFT09]. It would also be possible to use a method based on Shamir's secret sharing to create some simple redundancy [Sha79], by sending some additional data with Alice's second round message.

The question to ask is therefore: how many missing or erroneous streams should be tolerated, compared to the number of total streams? This proportion should be quite higher than the proportion of public data that Eve is supposed to be able to store, as otherwise it increases her chance of decrypting the message. However, a high fault tolerance ratio could be used by Alice if she has a single opportunity to send the message and wants to be sure that it gets decrypted. This would still be secure if it is a rare occurrence, as Eve would not have the incentives to invest into the storage necessary to decrypt just a few messages with strong redundancy.

Addressing already compromised encryption methods. If we decide to push Aumann *et al.*'s fears of an all-powerful adversary further, we can get a slightly different model that is in practice quite realistic. It is imaginable that Eve could already have functioning attack schemes against certain encryption methods. This is consistent with recent events, such as what happened with the NIST SP800-90 Dual Elliptic curve PRNG [Hal13]. In such a case, Alice can send two or three pointers, each in its own message encrypted with a different encryption algorithm. She should, however, be careful with the parity checks to prevent the decryption of one source from revealing the whole secret.

7 Discussion

The main contribution of this paper is a protocol to exfiltrate secrets that can realistically be used today by people with limited technical skills. It is a concern that our protocol can be used by actors with nefarious intents as well as well-meaning whistle-blowers. As such, it is not politically neutral. We believe, however, that it would have limited effect on governmental and industrial espionage. Indeed, those activities generally benefit from advanced technical expertise and highly-refined toolkits due to the powerful financial interests involved. On the other hand, whistle-blowers generally do not benefit from such a support network and would be the primary users of this technology.

There are alternatives that satisfy the constraints presented in this paper and could be used instead of our protocol. For example, we could use slightly altered data from P2P networks. We could also create a large entropy source by introducing some noise into the content distribution system of a P2P network. A similar idea could work by slightly modifying the Scuttlebutt protocol [vRDGT08]. Contrary to our system, these alternatives are not directly employable today, as they require the cooperation of a large set of users. Moreover, they would require a higher technical ability to implement.

Other entropy source might also appear as the result of two things: the evolution of our online behaviour and the increasing traffic from the deployment of the Internet of Things (IoT). Recent tendencies in hardware costs for data storage and global throughput are currently working in our advantage. If these trends continue, sources that today represent a small percentage of global throughput than pornography could become sufficient for our protocol in the future.

Social behaviours online have garnered a lot of interest, especially when it comes to security [MKV+13,DBC+14] or to subjects that can be taboo [MTC+14]. Besides just analysing these behaviours, we were interested in how we could build security features based on the social behaviours without directly affecting them. This is but one example and there might be many more social effects awaiting to be used as primitives to improve our security and privacy online.

References

[ADR02] Yonatan Aumann, Yan Zong Ding, and Michael O. Rabin. Everlasting security in the bounded storage model. *IEEE Transactions on Information Theory*, 48(6):1668–1680, 2002.

[Cis18a] Cisco. Global cloud index: Forecast and methodology, 2016–2021. Technical report, Cisco, 2018.

[Cis18b] Cisco. Global visual networking index: Forecast and trends, 2017–2022. Technical report, Cisco, 2018.

[Cul18] Cam Cullen. Global internet phenomena report. Technical report, Sandvine, 2018.

[DBC+14] Anupam Das, Joseph Bonneau, Matthew Caesar, Nikita Borisov, and XiaoFeng Wang. The tangled web of password reuse. In *NDSS*, volume 14, pages 23–26, 2014.

[dPCC18] Inigo del Portillo, Bruce G. Cameron, and Edward F. Crawley. A technical comparison of three low earth orbit satellite constellation systems to provide global broadband. In *69th International Astronautical Congress 2018*, 2018.

[Dyk01] Peter Dykstra. Net activists launch campaign to jam 'Echelon', 2001.

[FIP11] Michael J. Fischer, Michaela Iorga, and René Peralta. A public randomness service. In *Proceedings of the International Conference on Security and Cryptography – SECRYPT*, pages 434–438. IEEE, 2011.

[Hal13] Thomas C. Hales. The NSA back door to NIST. *Notices of the AMS*, 61(2):190–19, 2013.

[Hil14] Kashmir Hill. NSA responds to Snowden claim that intercepted nude pics 'routinely' passed around by employees, 2014.

[JJFT09] Chao Jin, Hong Jiang, Dan Feng, and Lei Tian. P-Code: A new RAID-6 code with optimal properties. In *Proceedings of the 23rd international conference on Supercomputing*, pages 360–369. ACM, 2009.

[Mad13] Chase Madar. *The passion of Bradley Manning: The story behind the Wikileaks whistleblower*. Verso Books, 2013.

[Mau92] Ueli M. Maurer. Conditionally-perfect secrecy and a provably-secure randomized cipher. *Journal of Cryptology*, 5(1):53–66, 1992.

[McD18] Tommie McDonald. How many people watch porn at work will shock you, 2018.

[MKV+13] Michelle L. Mazurek, Saranga Komanduri, Timothy Vidas, Lujo Bauer, Nicolas Christin, Lorrie Faith Cranor, Patrick Gage Kelley, Richard Shay, and Blase Ur. Measuring password guessability for an entire university. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer Communications Security*, CCS '13, pages 173–186, New York, NY, USA, 2013. ACM.

[MTC+14] Antoine Mazières, Mathieu Trachman, Jean-Philippe Cointet, Baptiste Coulmont, and Christophe Prieur. Deep tags: toward a quantitative analysis of online pornography. *Porn Studies*, 1(1-2):80–95, 2014.

[OOWJ16] Frederik Obermaier, Bastian Obermayer, Vanessa Wormer, and Wolfgang Jaschensky. About the panama papers. *Süddeutsche zeitung*, 2016.

[Pre17] Linda Pressly. Cam-girls: Inside the romanian sexcam industry. *BBC News, Bucharest*, 2017.

[SC08] Malcolm Slaney and Michael Casey. Locality-sensitive hashing for finding nearest neighbors [lecture notes]. *IEEE Signal processing magazine*, 25(2):128–131, 2008.

[Sha49] Claude E. Shannon. Communication theory of secrecy systems. *Bell system technical journal*, 28(4):656–715, 1949.

[Sha79] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[vdBBdK16] Antal van den Bosch, Toine Bogers, and Maurice de Kunder. Estimating search engine index size variability: a 9-year longitudinal study. *Scientometrics*, 107(2):839–856, 2016.

[vRDGT08] Robbert van Renesse, Dan Dumitriu, Valient Gough, and Chris Thomas. Efficient reconciliation and flow control for anti-entropy protocols. In *Proceedings of the 2nd Workshop on Large-Scale Distributed Systems and Middleware*, LADIS '08, pages 1–7, New York, NY, USA, 2008. ACM.