Safe passwords made easy to use

Nicolas K. Blanchard [1], Leila Gabasova [2], Clément Malaingre [3], Ted Selker[4], Eli Sennesh[5]

[1]IRIF, Université Paris Diderot

[2]Institut de Planétologie et d'Astrophysique de Grenoble

[3]Teads France

[4]University of California, Berkeley

[5]Northeastern University

Stanford EE Computer Systems Colloquium November 28th, 2018

Passwords are bad, m’kay ?

Too many passwords

State of password use:

- Average user has $\sim$ 100 accounts
- Creates 50 passwords per year on average
- Often counterproductive constraints, avoided by users (e.g. 1@MyPassword)

Too many passwords

State of password use:

- Average user has $\sim$ 100 accounts
- Creates 50 passwords per year on average
- Often counterproductive constraints, avoided by users (e.g. 1@MyPassword)

Because of this:

- High rate of re-use (75% of users)
- Lots of sharing (40% of users)
- Frequent loss of passwords (40% to 60% reinitialised every 3 months)

Authentication methods

Multiple alternatives to secure access:

- Biometrics: have been durably hackable
- Defer to a service (Facebook connect): trust issues
- Physical devices: introduce other vulnerabilities
- Password managers: single point of failure
- Passwords re-use: extremely vulnerable

Authentication methods

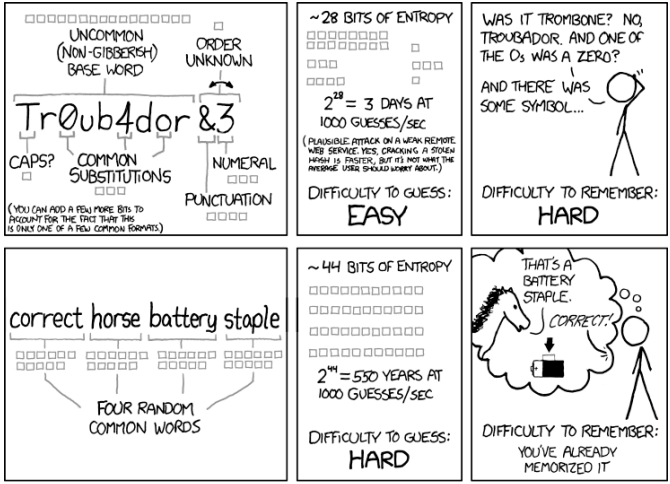
Multiple alternatives to secure access:

- Biometrics: have been durably hackable
- Defer to a service (Facebook connect): trust issues
- Physical devices: introduce other vulnerabilities
- Password managers: single point of failure
- Passwords re-use: extremely vulnerable

Methods to make passwords better:

- Salt + variable ending: soon vulnerable
- Blum's algorithm: costly
- Passphrases: not compatible with constraints

Passwords vs Passphrases



It seems we’re stuck with passwords!

Constraints

Constraints for a good password management algorithm:

- High entropy for each password
- High residual entropy against stolen clear-text passwords
- Memorable even without frequent use (hence deterministic)
- Easy to understand by non-Turing-award-winners
- Compatible with frequent constraints

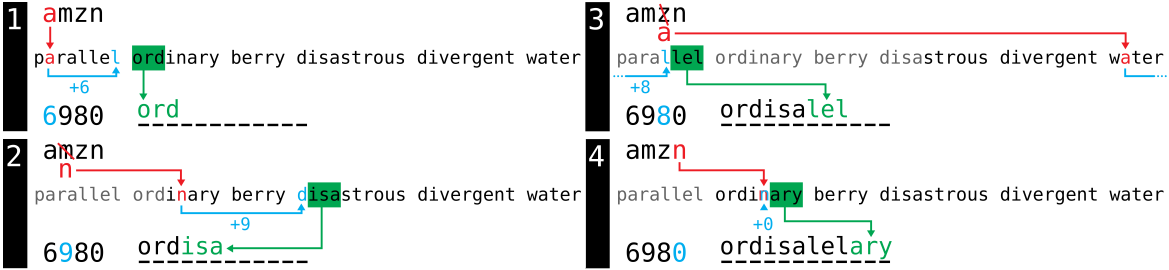
Idea: mentally extract entropy from a large secret

Cue-Pin-Select

High level view :

- Create one high-enropy passphrase and a 4-digit PIN
- Create a 4-letter cue for each service
- Deterministically extract 4 trigrams from the sentence using the PIN and the cue

Example run


1
amzn
parallel ordinary berry disastrous divergent water
+6
6980
ord
2
amzn
parallel ordinary berry disastrous divergent water
+9
6980
ordisa
3
amzn
parallel ordinary berry disastrous divergent water
+8
6980
ordisalel
4
amzn
parallel ordinary berry disastrous divergent water
+0
6980
ordisalelary

Main Algorithm

Data: Passphrase P of at least 6 random words
PIN K of 4 random digits
service name N
Result: String S of 12 characters
begin
  From N, create string M of four characters
  $L \longleftarrow$ Length(P), $V \longleftarrow 0$, $S \longleftarrow$ ""
  for $i = 0\ ;\ i < 4\ ;\ i++$ **do**
    $X \longleftarrow M[i]$
    while $X \notin P$ **do**
      $X \longleftarrow$ letter following X in the alphabet
    $V \longleftarrow$ index of next occurrence of $X \in P$ after V
    $V \longleftarrow V + K[i] + 3 \mod (L)$
    $S \longleftrightarrow$ Concatenate $(S, P[V-2], P[V-1], P[V])$
  Print S

Algorithm 1: Cue-Pin-Select

Security analysis

Bruteforcing Cue-Pin-Select

Today's standard for web services : 36-42 bits (30 years at 1000 tries/s).

Brute-force against Cue-Pin-Select :

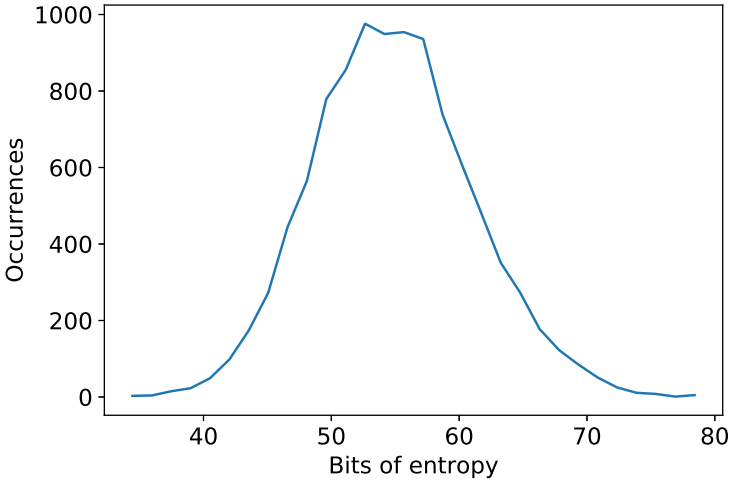
- Naive against a password $\rightarrow$ 56 bits
- Optimised dictionary against a password $\rightarrow$ 52 bits
- Naive against passphrase $\rightarrow$ 210 bits
- Dictionary against passphrase $\rightarrow$ 111 bits

Clear-text attacks

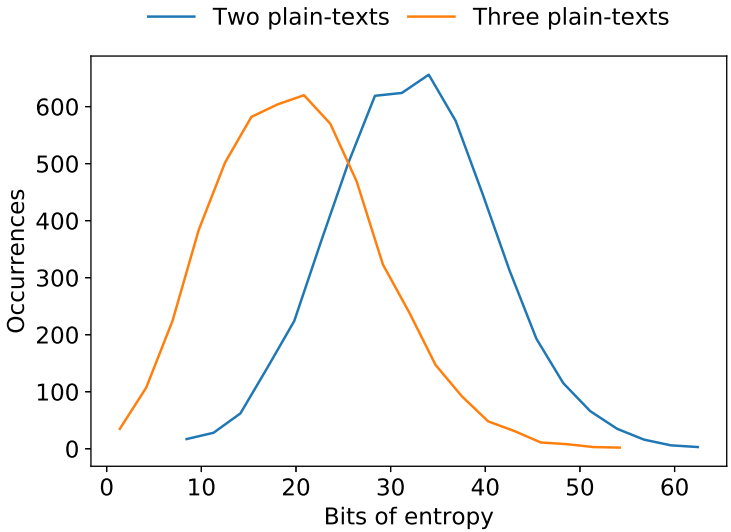
To simplify analysis, Very strong adversary model, who knows:

- 1+ passwords
- Length of the passphrase
- Position of each revealed trigram in the sentence

Residual entropy (empirical on 10 000 tries)



Residual entropy (empirical on 10 000 tries)



User experiment

After 4-day experiment:

- High initial cost (82s on average), and multiple errors initially
- Quick speed-up, down to 42s after two days, with pen and paper
- Increase when shift to mental computation only (86s)
- Speed-up over the last day (down to 57s), no errors
- Large variability, 24s-71s

Adaptability

Algorithm can be extended to handle:

- Number and special characters
- Length constraints
- Frequent changes

Cue-Pin-Select Summary

Cue-Pin-Select:

- 52 bits security per password
- Guaranteed resistance to single clear-text attack, probable resistance to 2-3 clear-text
- Can create 500+ passwords without high risk of strong partial collision
- Quick learning process to get under 1 min
- According to models, strongly memorable
- Natural extension to handle frequent constraints
- Other extension to improve security

How to choose a passphrase ?

Current methods to make passphrase

First possibility: let people choose them

Problems:

- Sentences from literature (songs/poems)
- Famous sentences (2.55% of users chose the same sentence in a large experiment)
- Low entropy sentences with common words

Current methods to make passphrase

First possibility: let people choose them

Problems:

- Sentences from literature (songs/poems)
- Famous sentences (2.55% of users chose the same sentence in a large experiment)
- Low entropy sentences with common words

Second possibility: random generation

Limits :

- Small dictionary if we want to make sure people know all words
- Harder to memorise

What if we take the best of both world ?

Passphrase choice experiment

We show 20 or 100 words to users, they have to pick – and remember – six.

Questions :

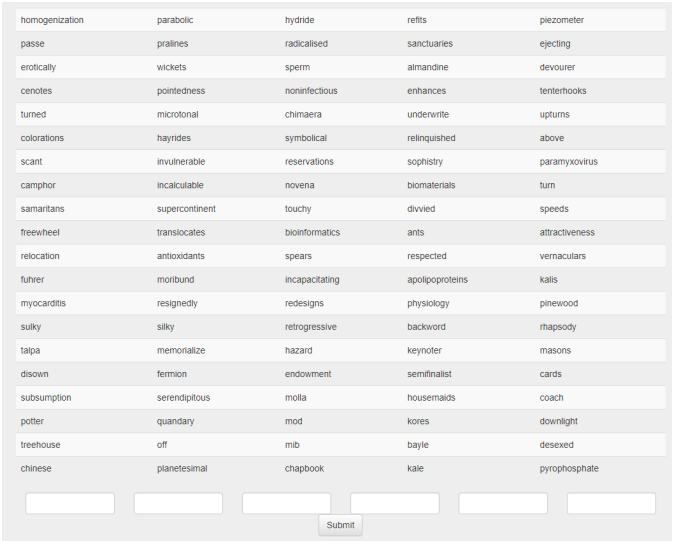
- What factors influence their choices ?
- What is the effect on entropy ?
- What are the most frequent mistakes ?
- How is memorisation affected ?

Protocol

Simple protocol :

- Show a list of 20/100 random words from a large dictionary
- Ask to choose and write down 6 words (imposed on the control group)
- Show them the sentence and ask them to memorise, with little exercise to help them.
- Distractor task: show them someone else's word list and ask to guess the word choice
- Ask them to write the initial sentence

Interface



homogenization	parabolic	hydride	refits	piezometer
passe	pralines	radicalised	sanctuaries	ejecting
erotically	wickets	sperm	almandine	devourer
cenotes	pointedness	noninfectious	enhances	tenterhooks
turned	microtonal	chimaera	underwrite	upturns
colorations	hayrides	symbolical	relinquished	above
scant	invulnerable	reservations	sophistry	paramyxovirus
camphor	incalculable	novena	biomaterials	turn
samaritans	supercontinent	touchy	divvied	speeds
freewheel	translocates	bioinformatics	ants	attractiveness
relocation	antioxidants	spears	respected	vernaculars
fuhrer	moribund	incapacitating	apolipoproteins	kalis
myocarditis	resignedly	redesigns	physiology	pinewood
sulky	silky	retrogressive	backword	rhapsody
talpa	memorialize	hazard	keynoter	masons
disown	fermion	endowment	semifinalist	cards
subsumption	serendipitous	molla	housemaids	coach
potter	quandary	mod	kores	downlight
treehouse	off	mib	bayle	desexed
chinese	planetesimal	chapbook	kale	pyrophosphate

Submit

Choosing models

Three main models to analyse user's choice

Uniform : every word with equal probability

Smallest : Take the six most frequent words from the list shown

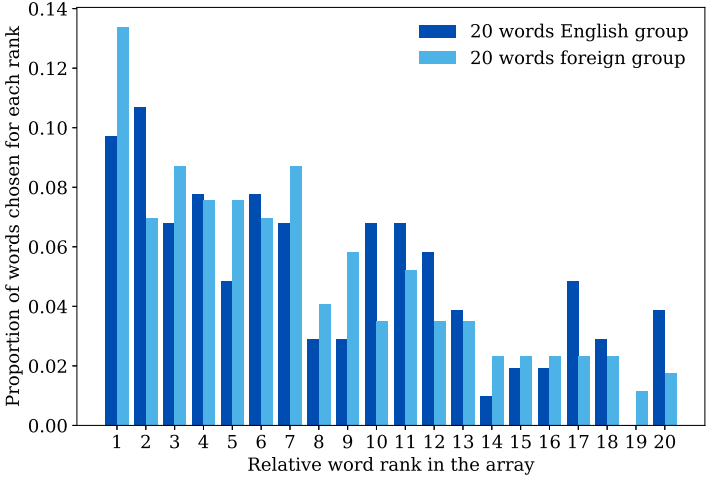
Corpus : every word taken with probability proportional to its use in natural language. The word of rank r_k is taken with probability :

$$\frac{\frac{1}{r_k}}{\sum_{i=1}^{n} \frac{1}{r_i}}$$

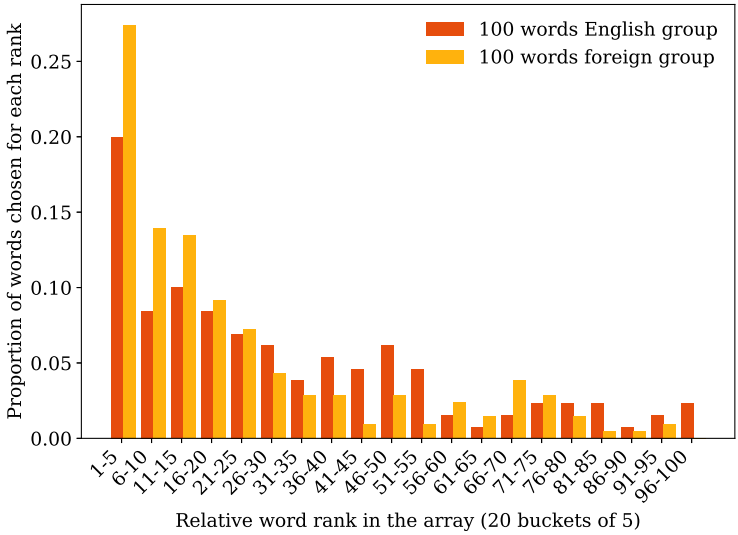
Error comparison

Section	Correct	Typo	Variant	Order	Miss	Wrong
1:20	19/47	6	8	6	26	5
1:100	26/51	10	5	3	16	4
Control	6/26	11	11	10	31	12
2:20	14/29	1	2	8	0	3
2:100	15/26	4	2	3	1	4

Semantic bias



Semantic bias

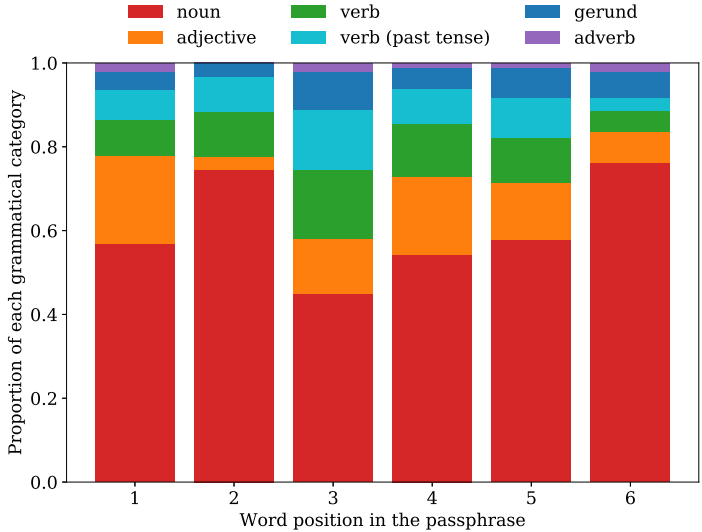


Syntactic bias

Syntactic effects :

- Average frequency ($< 50\%$) of meaningful sentences
- 65 different syntactic structures for 99 sentences
- Single frequent structure: six nouns in a row

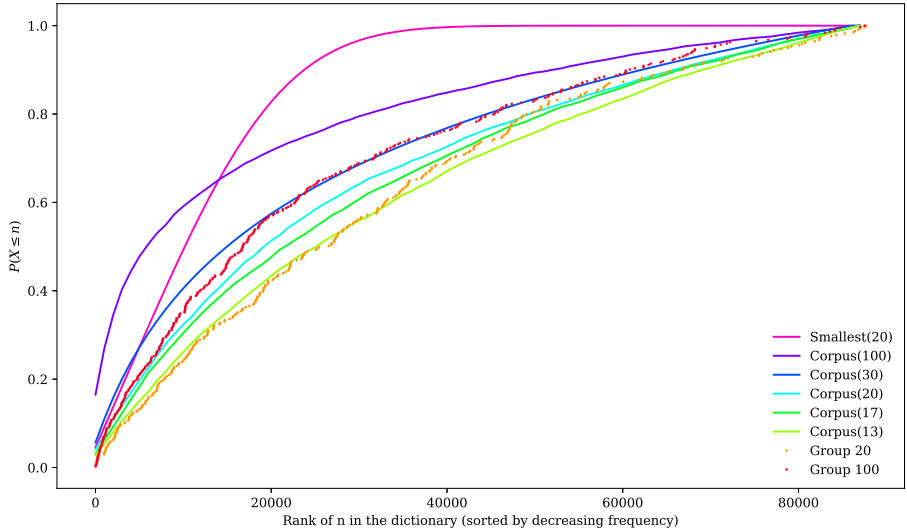
Syntactic bias



Entropy comparison

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		

Entropy curves



Conclusion

Passphrase choice method

Advantage with 100-word list:

- Secure: 97% of maximal entropy, 30% increase over uniform with limited dictionary
- Memorable: error rate divided by 4
- Lightweight: <1MB tool, can and should be used inside a browser

Questions

Questions on passphrase choice:

- What is the optimal number of words to show ?
- Is it interesting to take even bigger dictionaries ?
- Can this method and Cue-Pin-Select be applied to languages with small vocabularies (Esperanto)
- What is the best way to model user choice ?

Future password research

Our recent work:

- Typo-tolerant password checkers
- Culturally neutral codes and passwords for e-voting

Planned research:

- Better attack models with fewer assumption to prove higher resistance
- Mental computing cost model to test password algorithms without user experiments
- Alternative to Cue-Pin-Select that works in <30s