Local Password validation using Self-Organizing Maps

ESORICS 2014

Diogo Monica and Carlos Ribeiro (diogo.monica, carlos.ribeiro)@tecnico.ulisboa.pt)

Talk Outline

- Background and Motivation
- Our approach
- Performance
- Conclusions

Background and Motivation

Background

Passwords are still the **#1** way of doing authentication

- Leaks have shown that:
  - Users are bad at choosing passwords
  - There is prevalent password re-use across services

Background

Password validation heuristics are not working

- Leaks have shown that:
  - promote weak variations that computers are good at guessing
  - positive reinforcement of bad decisions (password strength meters)

Background

Password storage is evolving

- The prevalence of bcrypt, scrypt and variations have made brute-force hard
  - Time-memory trade off (TMTO) resistance
  - GPU unfriendliness (rapid random reads)
- Targeted (dictionary) attacks becoming the norm
  - Some of them based on the knowledge of what the most common passwords are

Background

- Password managers are still not prevalent
- There will always be passwords that people need to chose and memorize (laptop, offline access, password manager's password, etc)

Motivation

- Give better password strength feedback
- Remove unnecessary jumps and hoops (strict rules)
- Promote the use of passwords that are easy to remember but hard to guess

Password Frequency

Core idea

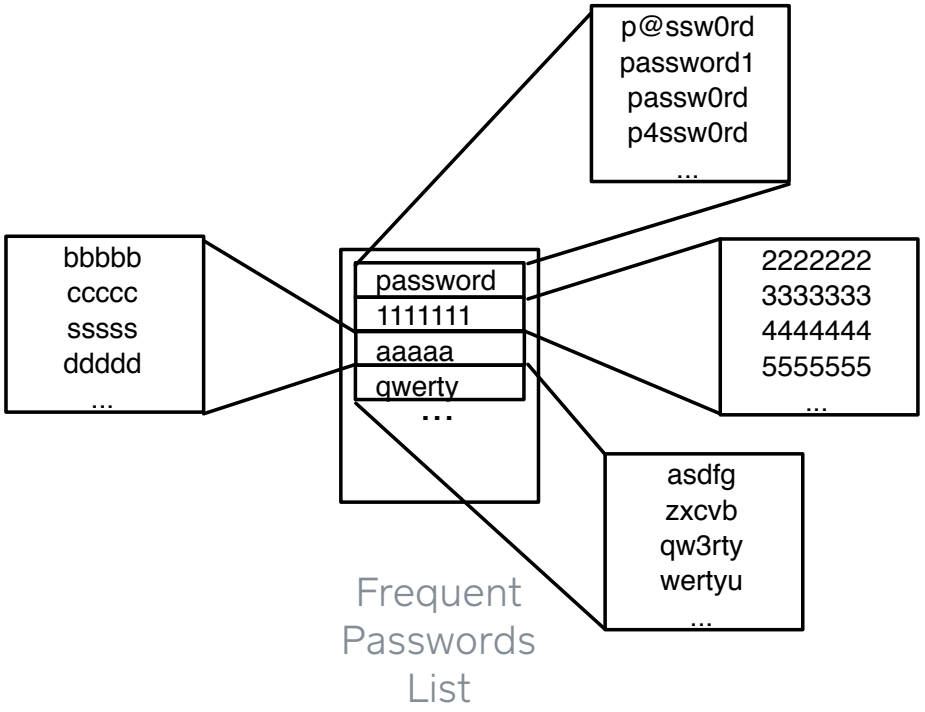
- Use the frequency of **past password appearance** to hamper the effectiveness of statistical dictionary attacks
- Some organizations already use basic versions of this method
  - Twitter prohibits the use of the 370 most common passwords

http://techcrunch.com/2009/12/27/twitter-banned-passwords/

Why is it hard?

- Lack of access to representative data-sets
- Computationally expensive
- Offline access (local validation)
  - Efficient client distribution (updates)
  - Compression
  - Potential leak of candidate passwords

Dealing with variations



Our goal

Design a popularity based classification scheme that:

- Resists common password variations (**generalization capability**)
- Allows for offline operation (no centralized authority)
- Is easily distributable through end user's systems
  - Total size must be small
  - Should not compromise password security
- Testing candidate passwords should be easy and inexpensive
  - Time, cpu, memory

Our approach

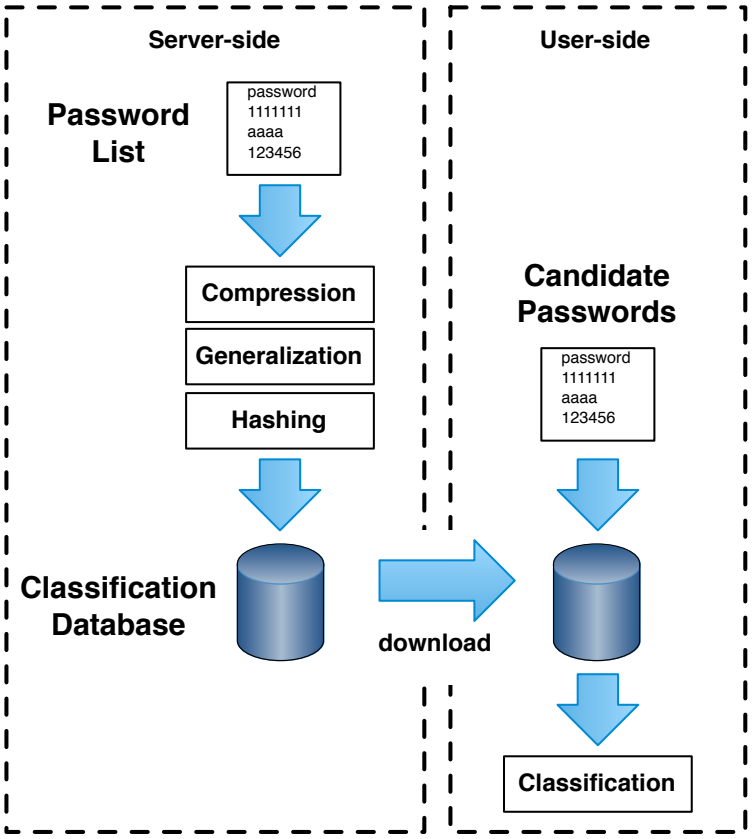
Our approach

Server-side:

- Compression
- Generalization
- Hashing

Client-side:

- Classification



Compression

Requirements

- Compress database to allow distribution (e.g. mobile phones)
- Compression should not destroy topological proximity of the passwords

Self-Organizing Maps

- Clustering tool
- Unsupervised neural network
- Reflects in the output space topological proximity relations in the input space
- It provides us with an easy way of doing password “generalization”

Self-Organizing Maps

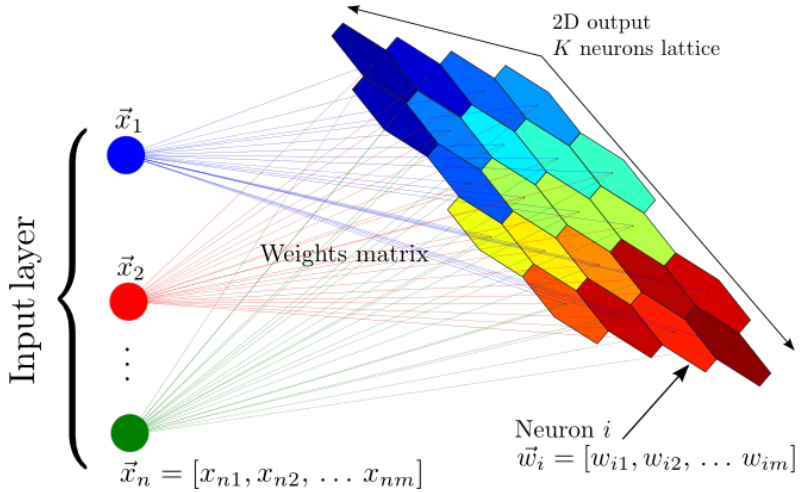
Training process

High-level Process:

- Determine which node has a model closer to the input password (BMU)
- Find the set of all nodes in the lattice neighborhood of the BMU
- Update the models of all nodes in the lattice neighborhood of the BMU, to make them approximate the input password.

Output:

The resulting map is a summary replica of the input space, with a much lower number of elements but maintaining its topological relations.

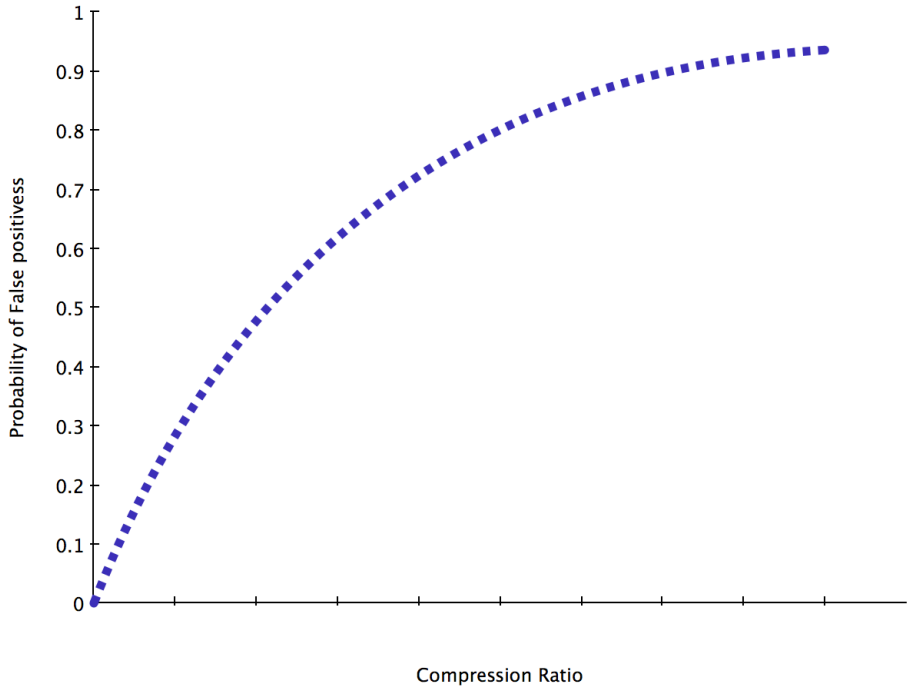


Concrete details of the training procedure can be found in the paper

Self-Organizing Maps

Compression Ratio

- Compression ration is $\frac{N}{M}$
  - N is the number of input passwords
  - M is the number of nodes in the SOM.
- Important to node that $p_{miss} = 0$ for any chosen compression ratio.



pmiss is the probability of wrongly classifying a password whose occurrence is higher than the threshold as safe

Self-Organizing Maps

Similarity Measure

- Defines the topological characteristics of the input space to be preserved in the output space

$$d('password','p@ssw0rd') =$$

$$\sqrt{(97-64)^2+(111-48)^2}\cdot 2^2 = 284.48$$

$$d(p_1,p_2) = \sqrt{\sum_{j=1}^{n}(p_1(j)-p_2(j))^2}\cdot hamm(p_1,p_2)^{\beta}$$

Euclidean distance between ASCII codes

Hamming distance

Beta pulls the overall measure of dissimilarity towards a simple human-related overlap distance

Classification

- Chose a popularity threshold
- Determine the BMU of the candidate password
- If the popularity level of the BMU is above the threshold the password is rejected.

Generalization

Generalization

- At this point we have a map that can be used for password classification
- Similar passwords are adjacent to each other
- By imposing popularity leakage from local maxima to neighboring nodes, we increase the generalization capability of the network.

Generalization

Non-linear low pass filtering of the popularity levels

Popularity label of node (x,y)

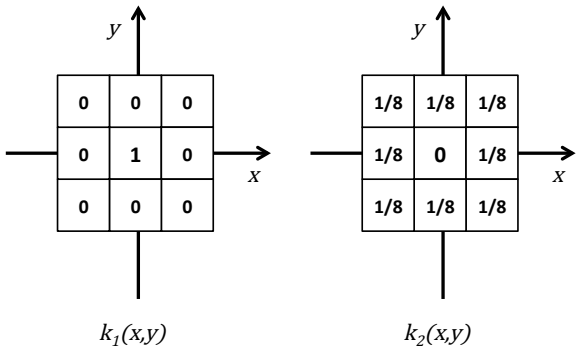
$$\phi(x, y)$$

We apply:

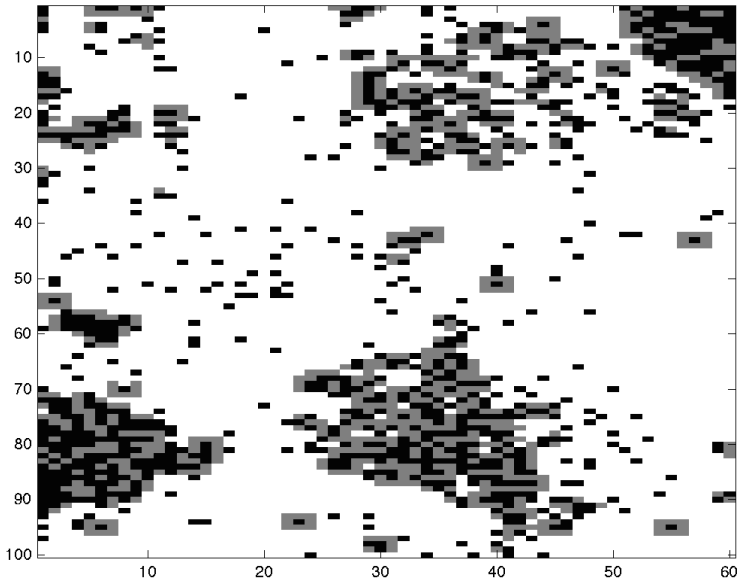
$$\phi(x, y) * K(x, y)$$

With:

$$K(x, y) = max(k_1(x, y), k_2(x, y))$$



Smoothing kernel



Hashing

Hashing

- The models in the SOM are a compressed summary of the training passwords
  - Security problem
- We will need to find a hash that ensures non-invertibility of models
  - Can’t destroy the topological proximity

Hashing

- Locality Preserving Hashes
  - Deterministically invertible
- Cryptographic Hashes
  - Destroy topological proximity by definition

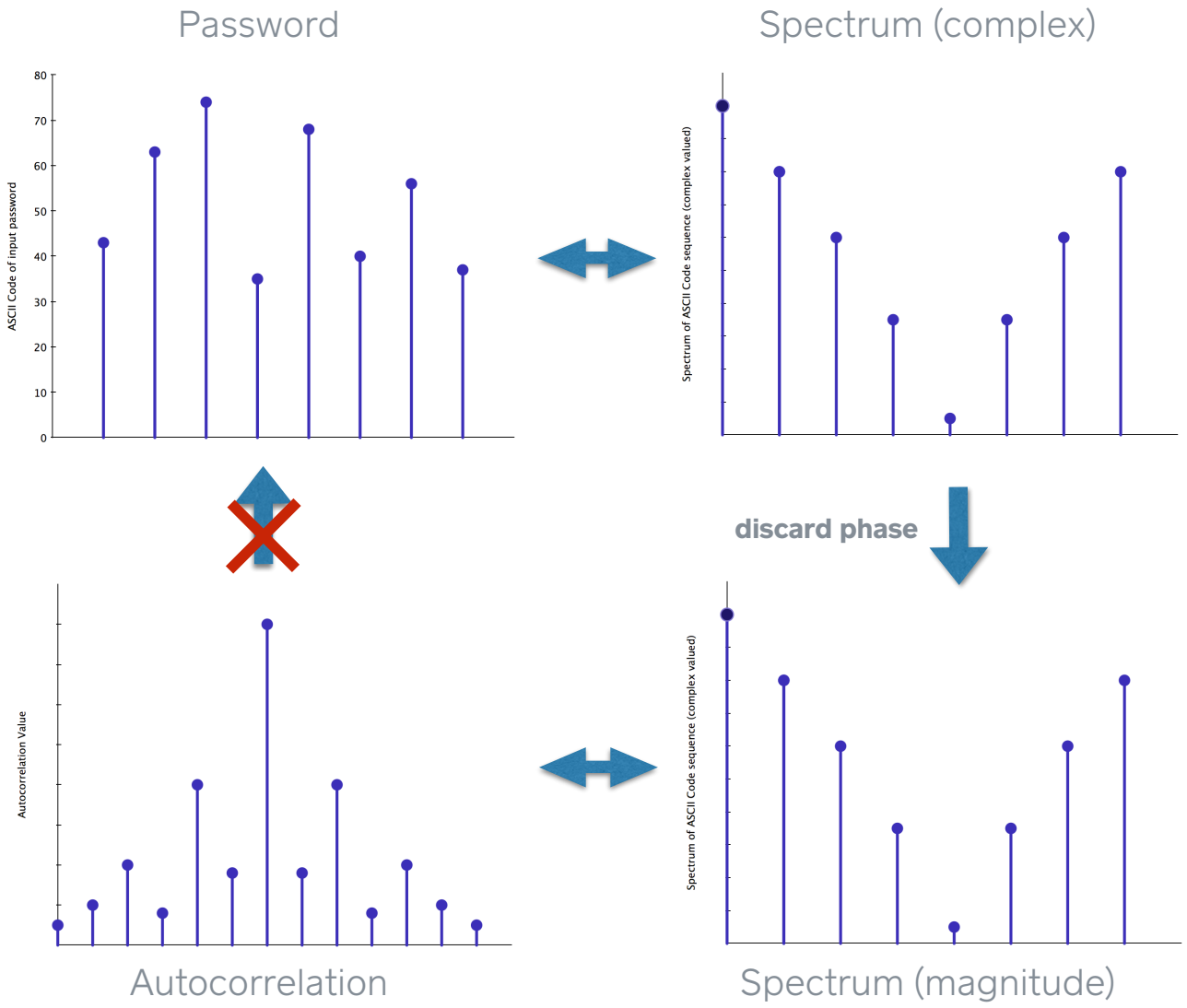
Hashing

Discrete Fourier Transform

- A linear vector projection that we understand
- We can remove the phase information, thus avoiding invertibility

$$p_{miss} = 0$$

- The power spectrum of a password is always closer to itself than to the power spectrum of a different password



Performance

CM-Sketch

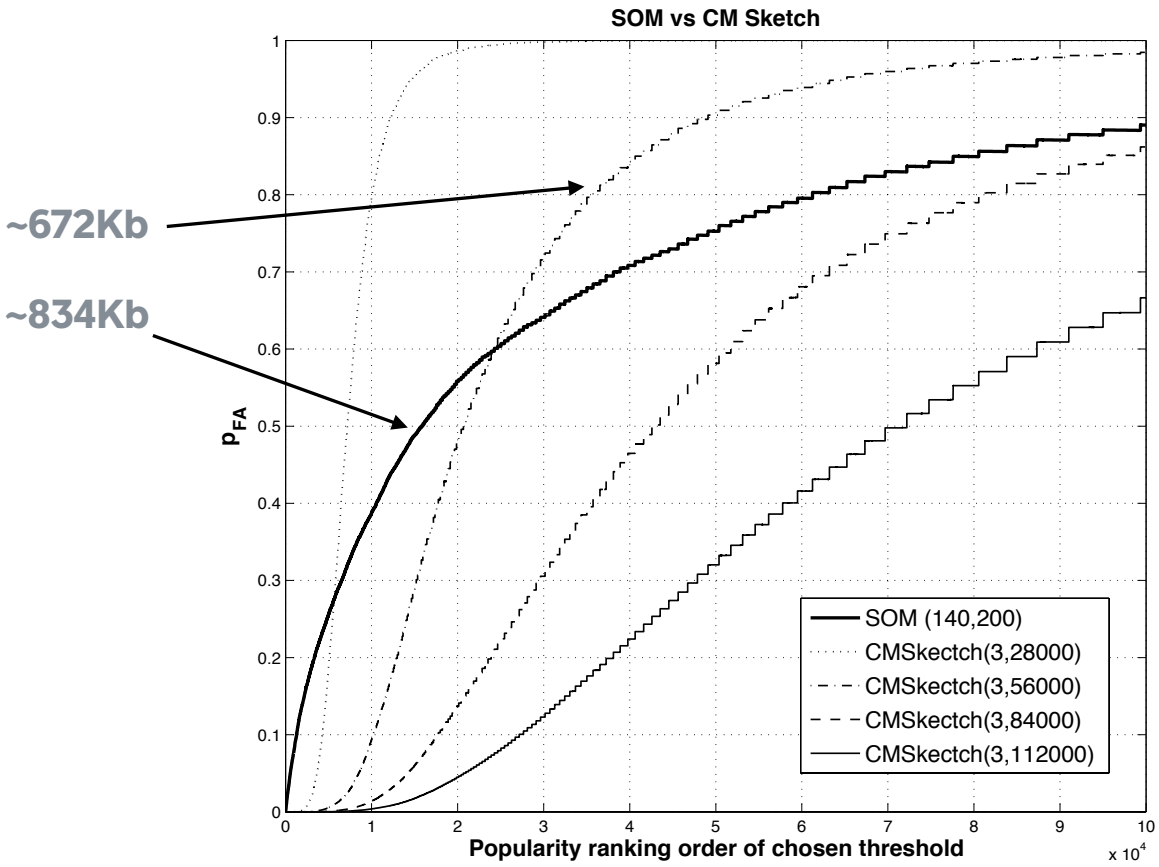
- Oracle to identify undesirably popular passwords
- Uses a count-min sketch
- No false negatives ($p_{miss} = 0$)
- Centralized operation
- No generalization features

http://research.microsoft.com/pubs/132859/popularityiseverything.pdf

Performance

Compression rate and statistical performance

- CM-sketch false positives are pure statistical classification errors
- SOM false positives may result from the desirable generalization properties



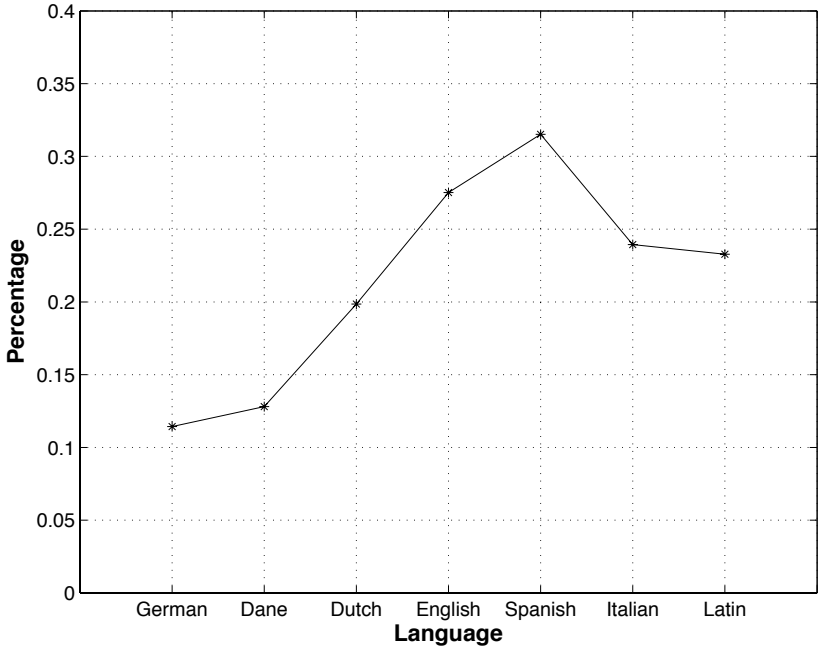
Performance

Generalization capability and consistency

- Is the SOM generalizing in a useful way?
- Is the SOM generalizing too much?

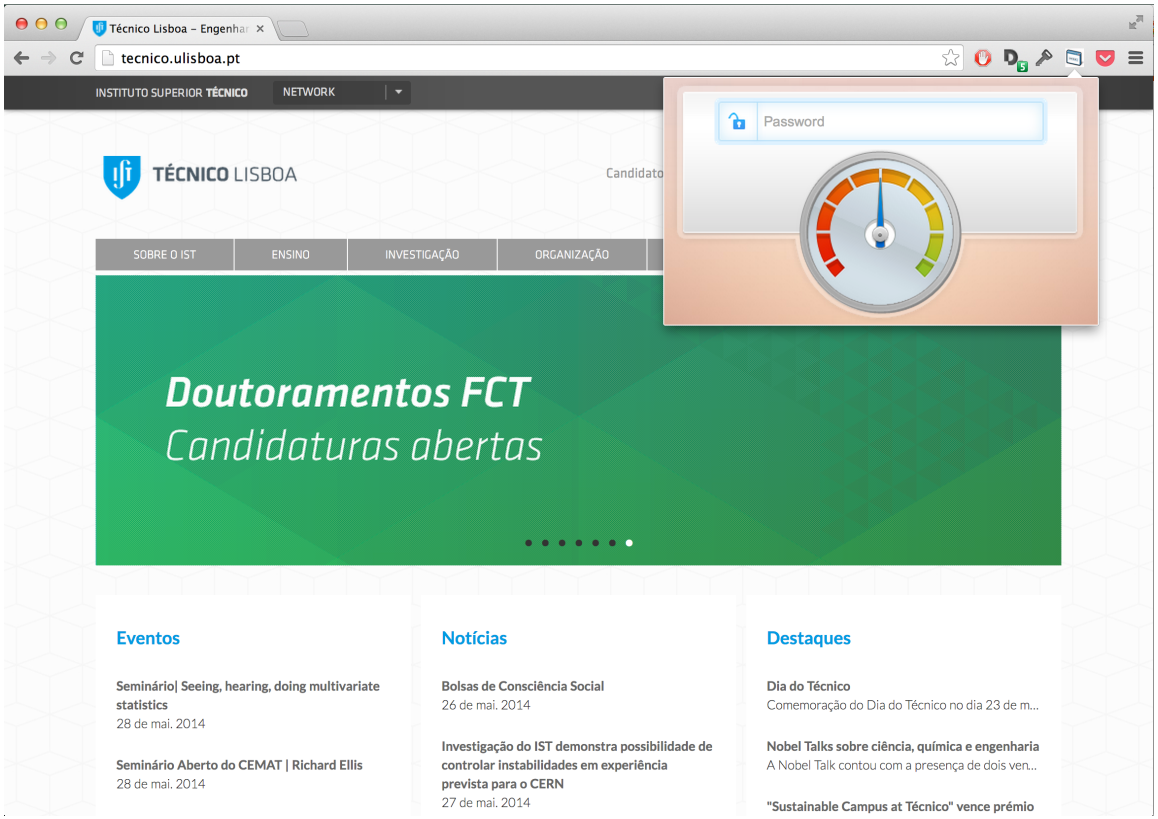
Passwords	Flagged as Dangerous
500 most probable	100%
All mutations	84%
Random passwords	11.3%

John the Ripper mutations testing



Testing different language dictionaries

Implementation



Conclusions

- Presented a scheme for password validation
  - Envisaged for local, decentralized operation
  - Possesses generalization capabilities
- No claims of optimality were made, but it was shown that our approach is feasible
- The solution was implemented and tested it empirically

Thank you

Diogo Monica (@diogomonica)

Why didn't you do the DFT at the beginning?

- Our inability to come up with a similarity measure that has human meaning when working with hashes
- Having the hashes at the beginning would make the training computationally more expensive
- By doing them at the end, the hashing function can be improved without changing anything in the SOM training
- We verified via Monte Carlo simulations that there is practically no loss in terms of topological proximity when doing the hashes at the end, making this a non-issue

Why did you chose this similarity measure?

- We are able to easily calculate distances between models and input passwords
- We have the ability of doing fractional approximation
- Our similarity measure captures a human “closeness” criteria

Why did you chose DFTs

- They are very fast to compute
- They are informationally non-invertible (if we discard the phase component)
- They maintained the topological proximity of our SOM
- Something that we can reason about because we understand what it means
- The only hashing mechanism that we found that works