

Fast Dictionary Attacks on Passwords Using Time-Space Tradeoff

Tyler Combs

The University of Oklahoma

tyler.combs@ou.edu

April 14, 2015

Overview

Introduction

- What is the problem?

- Dictionary Attacks

- Previous Work

Filtering

- Markovian

- Finite Automaton

Indexing Algorithms

Experiment

Conclusions

Analysis

What is the problem?

$$P(\alpha) = \prod_{x \in \alpha} \mathcal{V}(x)$$

Dictionary Attacks

Helman

Rainbow

Markov Chains

- ▶ Markov chains are commonly used in natural language processing. Most notably speech recognition systems.
- ▶ Markov models have been used to generate passwords for users.
- ▶ Given a set of states $S = \{s_1, s_2, \dots, s_n\}$ there is some probability p_{ij} that denotes the probability of transitioning from state s_i to state s_j
- ▶ The probability of transitioning to the next state depends only on the current state

Markov Chains

The order of a Markov chain of order n is defined as:

$$P(X_n = x_n | X_{n-1} = x_{n-1}, X_{n-2} = x_{n-2}, \dots, X_1 = x_1) = \\ P(X_n = x_n | X_{n-1} = x_{n-1}, \dots, X_{n-m} = x_{n-m})$$

Zero-order Markov Chain:

$$P(X_n = x_n | X_{n-1} = x_{n-1}, X_{n-2} = x_{n-2}, \dots, X_1 = x_1) = \\ P(X_n = x_n)$$

First-order Markov Chain:

$$P(X_n = x_n | X_{n-1} = x_{n-1}, X_{n-2} = x_{n-2}, \dots, X_1 = x_1) = \\ P(X_n = x_n | X_{n-1} = x_{n-1})$$

Zero-order Markov Model

In a zero-order Markov model, each character is generated given its underlying probability distribution. This is based on the frequency of the letter in the users natural language. Formally the zero-order model can be written as:

$$P(\alpha) = \prod_{x \in \alpha} \mathcal{V}(x)$$

where: where

- ▶ $P(\cdot)$ is the markovian probability distribution
- ▶ α is a string of characters
- ▶ $\mathcal{V}(\cdot)$ is the frequency of a letter occurring in English
- ▶ x is an individual character

First-order Markov Model

In a First-order Markov model, each ordered pair is assigned a probability and each character is generated by looking at the previous character. The first-order Markov model can be written as:

$$P(x_1 x_2 x_3 \cdots x_n) = \mathcal{V}(x_1) \prod_{i=1}^{n-1} \mathcal{V}(x_{i+1} | x_i)$$

where: where

- ▶ $P(\cdot)$ is the Markovian probability distribution
- ▶ x_i are individual characters
- ▶ $\mathcal{V}(\cdot)$ is the frequency of a letter or ordered pair occurring in English

Markov Dictionary

A probability distribution is not a dictionary. To create a dictionary, discretize the probabilities into two levels using a threshold θ

Zero-order dictionary

$$\mathcal{D}_{\mathcal{V},\theta} = \{\alpha : \prod_{x \in \alpha} \mathcal{V}(x) \geq \theta\}$$

First-order dictionary

$$\mathcal{D}_{\mathcal{V},\theta} = \{x_1 x_2 \cdots x_n : \mathcal{V}(x_1) \prod_{i=1}^{n-1} \mathcal{V}(x_{i+1}|x_i) \geq \theta\}$$

Markov Dictionary

The zero-order model is better for abrivtations and acrynyms. For example, a user picks their favorite song lyric and the first letter of each word creates their password.

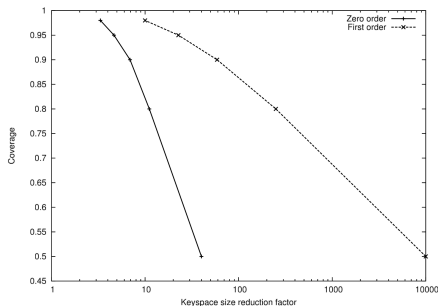


Figure: Convergence vs reduction in Keyspace size ($|\mathcal{K}|$) for 8-character sequences

Deterministic Finite Automaton

- ▶ A DFA or Deterministic Finite Automaton is a finite state machine that accepts or rejects a string
- ▶ A regular expression can be constructed from a DFA
- ▶ Humans are not random with how they use numerals and special characters
 - ▶ Numbers tend to be at the end of a password: password1
 - ▶ Capital letters are typically at the beginning of a password: Password
 - ▶ there are typically more lowercase letters in passwords than uppercase letters, numerals, or special characters

Dictionary using a DFA

An improved dictionary is one where strings are both accepted by a Markovian filter and accepted by at least one DFA from some set of DFA's. The updated dictionary is defined as:

$$\mathcal{D}_{\mathcal{V}, \theta, \langle M_i \rangle} = \{ \alpha : \prod_{x \in \alpha} \mathcal{V}(x) \geq \theta, \text{ and } \exists i : M_i \text{ accepts } \alpha \}$$

where

- ▶ A is the set of 26 uppercase characters
- ▶ a is the set of 26 lowercase characters
- ▶ n is the set of 10 numerals
- ▶ s is the set of 5 special characters
 $\{space, hyphen, underscore, period, comma\}$

Indexing Algorithms

- ▶ The goal is to create an algorithm that will efficiently enumerate the passwords in a given password space. Given i as input return the i^{th}
- ▶ In the rainbow attack the reduction function maps from ciphertext space to $\{0, 1, \dots, |\mathcal{K} - 1|\}$
- ▶ Composed with a mapping from $\{0, 1, \dots, |\mathcal{K} - 1|\}$ to a key in \mathcal{K} .
- ▶ Makes no assumption about keyspace other than its size
- ▶ Use the rainbow attack with a "smart" way to choose the keyspace

Dictionary Modification

Modify the dictionary to only consider fixed length strings. This allows for different threshold values θ for each length.

$$\mathcal{D}_{\mathcal{V},\theta,\ell} = \{\alpha : |\alpha| = \ell \text{ and } \prod_{x \in \alpha} \mathcal{V}(x) \geq \theta\}$$

Discretization

The algorithm also needs to discretize the probability distribution of the strings. First, turn the dictionary into a sum rather than a product.

Transform the product

$$\prod_{x \in \alpha} \mathcal{V}(x) \geq \theta$$

$$\log\left(\prod_{x \in \alpha} \mathcal{V}(x)\right) \geq \log(\theta)$$

$$\log(\mathcal{V}(x_1)\mathcal{V}(x_2) \cdots \mathcal{V}(x_n)) \geq \log(\theta)$$

$$\log(\mathcal{V}(x_1)) + \log(\mathcal{V}(x_2)) + \cdots + \log(\mathcal{V}(x_n)) \geq \log(\theta)$$

Discretization

To arrive at a discrete version of the modified dictionary:

$$\mathcal{D}_{\mathcal{V},\theta,\ell} = \{\alpha : |\alpha| = \ell \text{ and } \sum_{x \in \alpha} \mu(x) \geq \lambda\}$$

Where where

- ▶ $\mu(x) = \log(\mathcal{V}(x))$
- ▶ $\lambda = \log(\theta)$

Discretization

- ▶ $\mu(x) = \log(\mathcal{V}(x))$
- ▶ Discretize the values of the μ function to the nearest multiple of some μ_0
- ▶ Narayanan and Shmatikov use a μ_0 that yields around 1000 different discrete values

Zero-Order Markovian Dictionary

First-Order Markovian Dictionary

DFA Dictionary

Any Keyspace \mathcal{K}

Hybrid Markovian/DFA

Multiple Keyspaces

Optimizations

Experiment

- ▶ Measure coverage of rainbow attack vs hybrid attack
- ▶ 142 real user passwords
- ▶ 6-Character alphanumeric sequences for the rainbow attack ($|\mathcal{K}| = 36^6 \approx 2 * 10^9$)
- ▶ 70 regular expressions

Results

Category	Count	Rainbow	Hybrid
Length at most 5	63	29	63
Length 6	21	10	17
Length 7	18	0	10
Length 8, A* or a*	9	0	6
Others	31	0	0
Total	142	39(27.5%)	96(67.6%)
only length ≥ 6	79	10(12.7%)	33(41.8%)

Figure: Passwords recovered in Hybrid attack vs. Rainbow attack

Conclusions

- ▶ One of many attacks targeting human weakness
- ▶ Some possible defences against dictionary attacks for human memorable passwords
 - ▶ Graphical passwords
 - ▶ Biometric information
- ▶ Are these actually safer?

Analysis