Password Cracking

eCrime and Internet Service Abuse +
Applied Cryptography
Combined Lecture

UIUC - Joshua Reynolds

Let U be the set of users

And let p be the user's password

$\forall u_i \in U$,

store $\{u_i, C_i\}$

where $C_i = E_p(\text{Nonce})$

P. Oechslin’s Attack

$E_p(\text{Nonce})$

P. Oechslin’s Attack

Does it work with hashed passwords?

P. Oechslin’s Attack

Does it work with hashed passwords?

E_p(Nonce) vs. $H(p)$

P. Oechslin’s Attack

E_p(Nonce) vs. H(p)

1. Both take in a secret p and output a value

P. Oechslin’s Attack

$E_p(\text{Nonce})$ vs. $H(p)$

1. Both take in a secret p and output a value
2. Both make it cryptographically hard to recover p from that output value

P. Oechslin’s Attack

E_p(Nonce) vs. H(p)

1. Both take in a secret p and output a value indistinguishable from random
2. Both make it cryptographically hard to recover p

The attack will work on either method

We will use the H(k) notation today

Math

Let H() be a cryptographic hash function

Let P be the set of all possible passwords.

Let p_i be a password from the set P.

Let $h_i = H(p_i)$

Let H be the set of the corresponding h_i for all p_i

$|H| == |P|$

Hash Table

```
For every $p_i$ in P:
    Compute & Store $h_i$
```

Time complexity to generate: $\Theta(|P|)$

Time complexity to lookup: $\Theta(\log(|P|))$

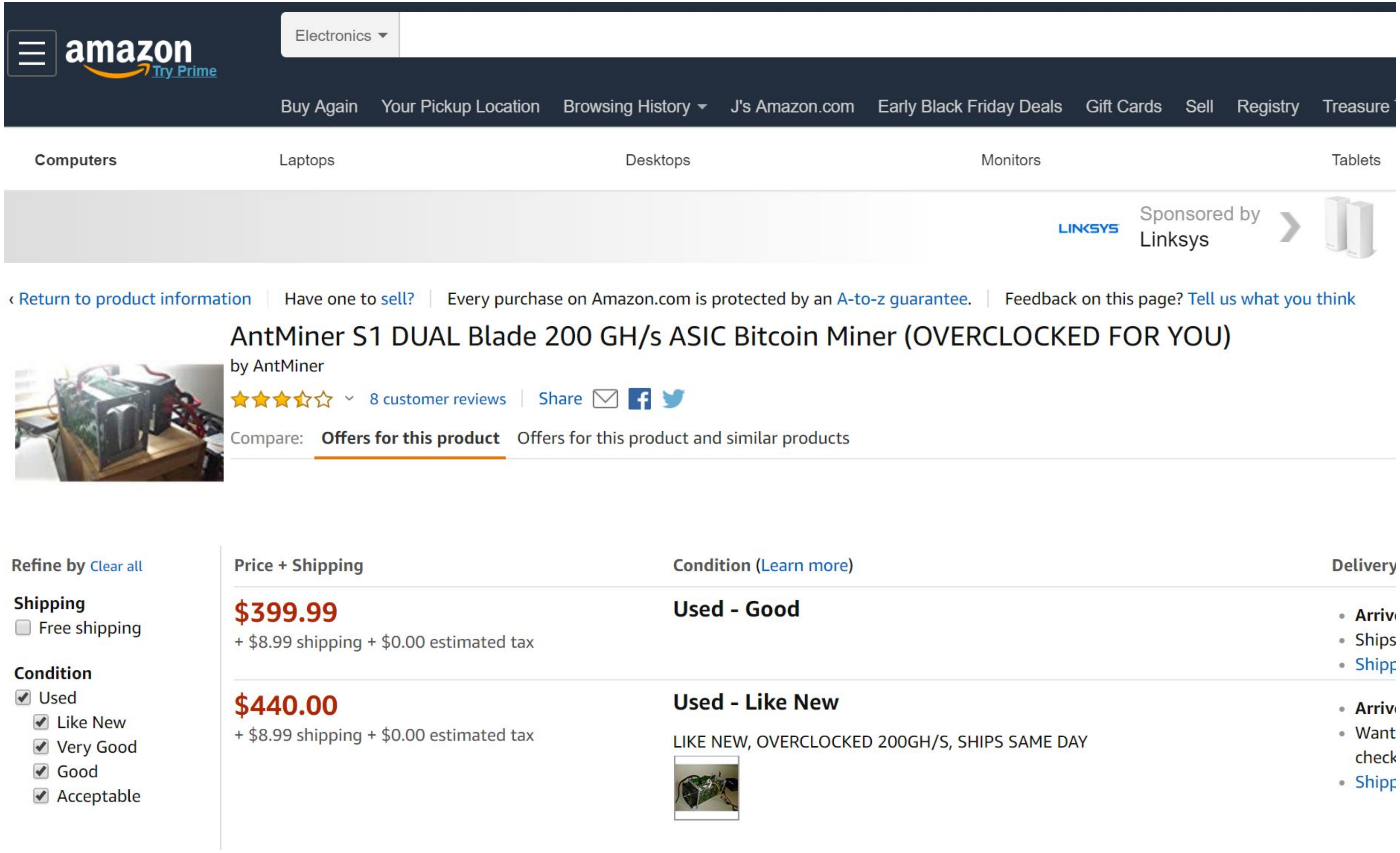
Space complexity: $\Theta(|P|)$

How big is |P| ?

Consider an exactly 7-character password with upper and lower case letters.

Permutation(48,7) = 587,068,342,272

How long would it take to generate every SHA-256 hash?



Electronics

Buy Again | Your Pickup Location | Browsing History | J's Amazon.com | Early Black Friday Deals | Gift Cards | Sell | Registry | Treasure

Computers | Laptops | Desktops | Monitors | Tablets

Sponsored by Linksys

‹ Return to product information | Have one to sell? | Every purchase on Amazon.com is protected by an A-to-z guarantee. | Feedback on this page? Tell us what you think

AntMiner S1 DUAL Blade 200 GH/s ASIC Bitcoin Miner (OVERCLOCKED FOR YOU)

by AntMiner

8 customer reviews | Share

Compare: **Offers for this product** Offers for this product and similar products

Refine by Clear all

Shipping
- [ ] Free shipping

Condition
- [x] Used
  - [x] Like New
  - [x] Very Good
  - [x] Good
  - [x] Acceptable

Price + Shipping	Condition (Learn more)	Delivery
$399.99 + $8.99 shipping + $0.00 estimated tax	**Used - Good**	Arriv / Ships / Shipp
$440.00 + $8.99 shipping + $0.00 estimated tax	**Used - Like New** LIKE NEW, OVERCLOCKED 200GH/S, SHIPS SAME DAY	Arriv / Want check / Shipp

How big is |P| ?

Consider an exactly 7-character password with upper and lower case letters.

Permutation(48,7) = 587,068,342,272

200Gh = 200,000,000 hashes per second

P(48,7)/200Gh = ~50 Minutes on a $400 machine

How big is |P| ?

Consider an exactly 7-character password with upper and lower case letters.

Sha-256 hash = 32 bytes

32 bytes * |P| = 17 TB

Idea

Can we trade lookup time for storage reduction?

Hash Chains

Define a reduction function R that maps from hash-output space back into P.

For each chain, select a random p_i from P as your starting point.

    $\text{Chain}_i = [p_i \rightarrow R(H(p_i)) \rightarrow \{\text{repeat } t \text{ times}\} \rightarrow \text{final\_value}]$

Repeat until you have coverage.

Reduction Function

Its input is the output of a hash function: 0x59ae5403928df849394...

Its output is a string that is a possible password: “a#2%33pq”

Simple example, treat the last 7 bytes of the hash as ascii.

Hash Chains

p_i= “1234567”

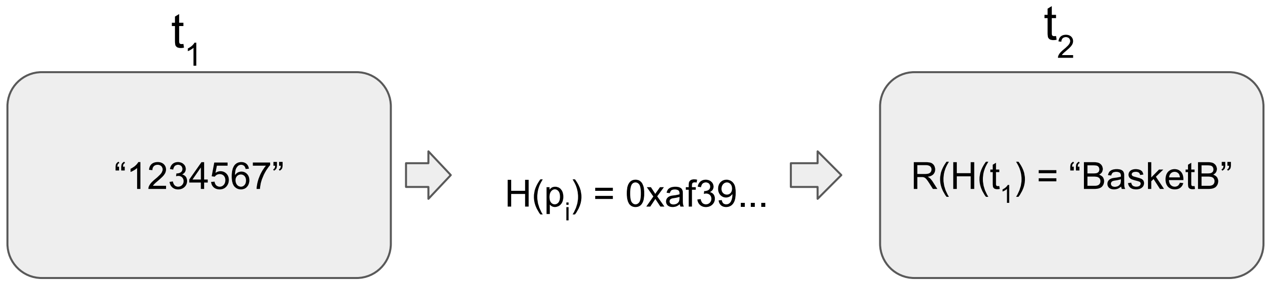
Hash Chains

$$\begin{tikzcd} \boxed{p_i = \text{“1234567”}} \arrow[r] & H(p_i) = \text{0xaf39...} \end{tikzcd}$$

Hash Chains

$$\begin{tikzcd}
\boxed{p_i = \text{“1234567”}} \arrow[r] & H(p_i) = \text{0xaf39...} \arrow[r] & \boxed{R(H(p_i) = \text{“BasketB”}}
\end{tikzcd}$$

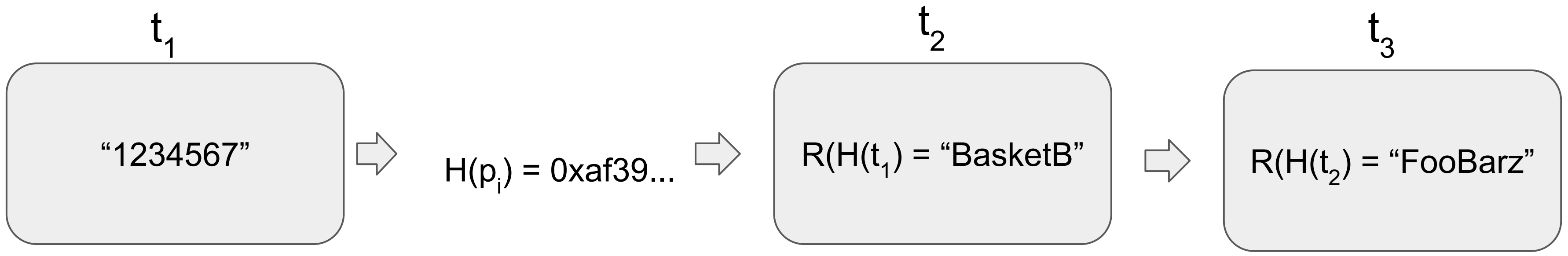
Hash Chains



Hash Chains

$$\begin{tikzcd} \overset{t_1}{\boxed{p_i = \text{“1234567”}}} \arrow[r] & H(p_i) = \text{0xaf39...} \arrow[r] & \overset{t_2}{\boxed{R(H(t_1) = \text{“BasketB”}}} \arrow[r] & \overset{t_3}{\boxed{R(H(t_2) = \text{“FooBarz”}}} \end{tikzcd}$$

Hash Chains



**If I store t_1 then I can re-compute the whole chain.*

**If I store t_n (the last link) then I can check if any password is in the chain.*

Hash Chains Lookup

Start with a hash - apply R() to it and see if it matches any of the stored end-links

Hash Chains Lookup

Start with a hash - apply R() to it and see if it matches any of the stored end-links

Keep applying R(H()) up to n times (the number of links in the hash chains) and checking if the result matches any of your stored end-links

Hash Chains Lookup

Start with a hash - apply R() to it and see if it matches any of the stored end-links

Keep applying R(H()) up to n times (the number of links in the hash chains) and checking if the result matches any of your stored end-links

If you find a match, the true password lies in that chain.

Hash Chains Lookup

Start with a hash - apply R() to it and see if it matches any of the stored end-links

Keep applying R(H()) up to n times (the number of links in the hash chains) and checking if the result matches any of your stored end-links

If you find a match, the true password lies in that chain.

If you never see a match, your chains do not contain the answer.

Collisions

Hash chains collide when they contain duplicate values.

Once two chains have a duplicate value, all subsequent values must also be the same. They will contain duplicate information.

The greater coverage of P you try to get, the more collisions will happen.

There can also be cycles within a single chain.

Choosing a Smart Reduction Function

-Avoid Cycles/Collision

-Map to likely values in P

-Different Reduction Function for each chain

Rainbow Tables

Build a series of unique reduction functions R_i applied at t_i on the chain.

Collisions will diverge unless they are collisions in the exact same step on the chain. Collisions are $|R|$ times less likely to occur.

Salting

Hash(Password|Salt)

Dilutes the value of pre-computed tables

Why Not Just Ask Nicely?

Why Not Just Ask Nicely?

To: you@gmail.com

Re: Dan from Google - Your Account

Your google account has been {penalty} for {reasons}!

Sign in here to fix this urgent problem!

<a href=evil.com>google.com</a>

-Dan from Google Customer Support

Idea

Choose high-value accounts to crack.

Idea

Choose high-value accounts to crack.

Are you then safe if you are not a valuable target?

Idea

Choose high-value accounts to crack.

Are you then safe if you are not a valuable target?

Do you plan not to be a valuable target forever?

Another Idea

How good of random string generators are people?

Another Idea

How good of random string generators are people?

Please enter a password

-> afableyellow

Another Idea

How good of random string generators are people?

Please enter a password

-> afableyellow

Error: your password must contain at least 1 number and 1 special character

Another Idea

How good of random string generators are people?

Please enter a password

-> afableyellow

Error: your password must contain at least 1 number and 1 special character

-> afableyellow1!

Registration Complete

Smart Guessing

Dictionary words

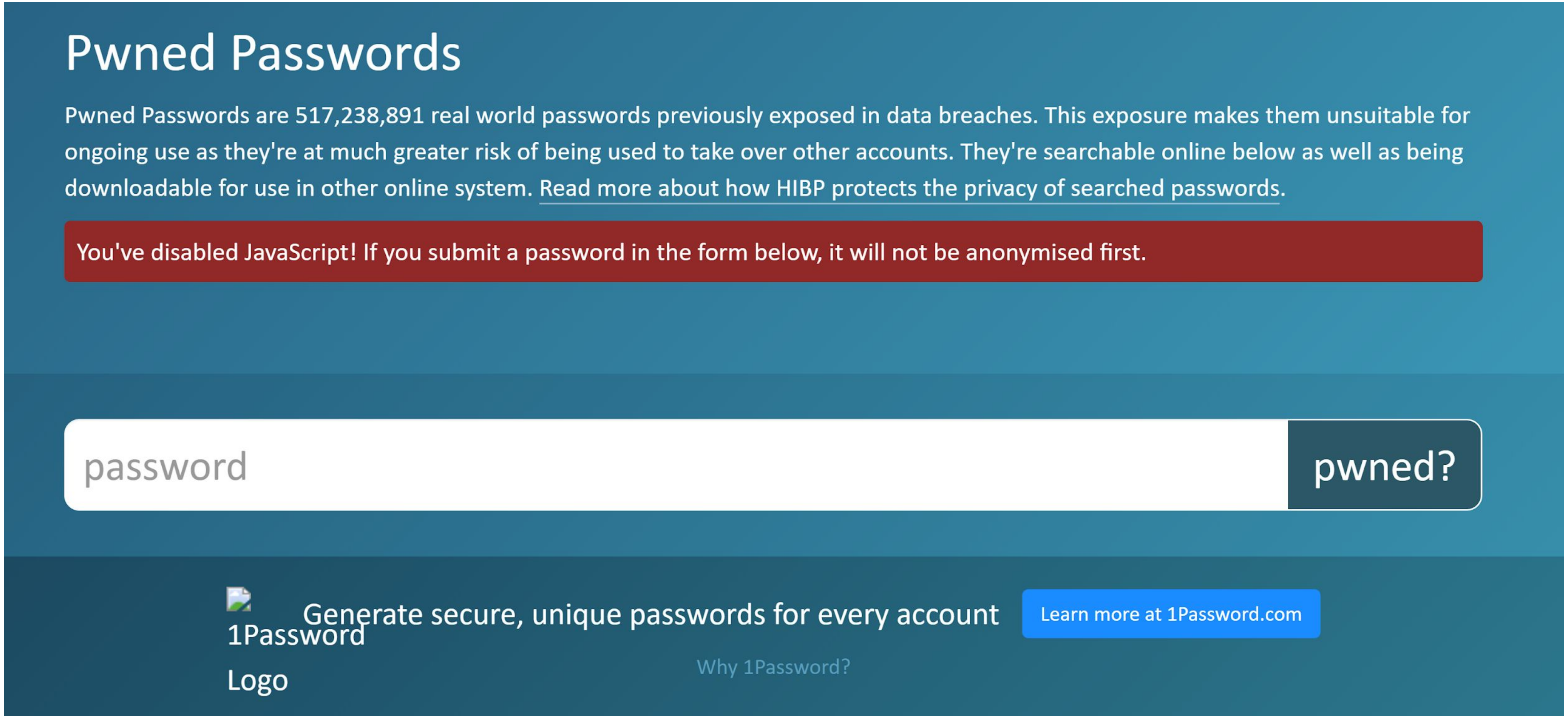
Dictionary words with substitutions

Keyboard patterns

Patterns (like number and special char at the end)

Leaked Password Lists

517,238,891 Leaked Passwords



Pwned Passwords

Pwned Passwords are 517,238,891 real world passwords previously exposed in data breaches. This exposure makes them unsuitable for ongoing use as they're at much greater risk of being used to take over other accounts. They're searchable online below as well as being downloadable for use in other online system. Read more about how HIBP protects the privacy of searched passwords.

You've disabled JavaScript! If you submit a password in the form below, it will not be anonymised first.

password
pwned?
1Password Logo
Generate secure, unique passwords for every account
Learn more at 1Password.com
Why 1Password?

https://haveibeenpwned.com

Password Crackers

HashCat

JohnTheRipper

Ophcrack

Aircrack

DavidGrohl

Can we avoid the server ever seeing the password?

Can we avoid the server ever seeing the password?

ZK{(password): H = H(salt, password)}

Can we avoid the server ever seeing the password?

ZK{(password): H = H(salt, password)}

But you have to reveal the salt as a public parameter

Attackers can start working on cracking tables for individual accounts right now!

Preventing Pre-Computation *(Jarecki et al. 2018)*

Preventing Pre-Computation *(Jarecki et al. 2018)*

$$\mathrm{ZK}\Big\{\Big(\mathit{pwd}\Big) : \mathrm{H} = \mathrm{H}(\mathrm{H}(\mathrm{pwd}, \mathrm{H}'(\mathrm{pwd})^{\mathrm{b}}), \mathrm{pwd})\Big\}$$

Preventing Pre-Computation *(Jarecki et al. 2018)*

$$\mathrm{ZK}\left\{\left(\mathit{pwd}\right): \mathrm{H} = \mathrm{H}(\mathrm{H}(\mathrm{pwd},\mathbf{H'(pwd)^{b}}), \mathrm{pwd})\right\}$$

b is known only to the verifier (server)

$\mathbf{X^{b}}$ can be obtained by anyone for any X

The real prover sends **H'(pwd)** and gets back $\mathbf{H'(pwd)^{b}}$

Preventing Pre-Computation *(Jarecki et al. 2018)*

$$\mathrm{ZK}\left\{\left(\mathit{pwd}\right): \mathrm{H} = \mathrm{H}(\mathrm{H}(\mathrm{pwd},\mathbf{H'(pwd)^{b}}), \mathrm{pwd})\right\}$$

b is known only to the verifier (server)

$\mathbf{X^{b}}$ can be obtained by anyone for any X

The real prover sends **H'(pwd)** and gets back **H'(pwd)**$^{\mathbf{b}}$

Attackers have to guess full entropy b before they can generate any lookup tables