Lecture 23 — Password Cracking, Bitcoin Mining

Patrick Lam & Jeff Zarnett
patrick.lam@uwaterloo.ca, jzarnett@uwaterloo.ca

Department of Electrical and Computer Engineering
University of Waterloo

October 15, 2020

scrypt is the algorithm behind DogeCoin.

The reference:

Colin Percival, "Stronger Key Derivation via Sequential Memory-Hard Functions".
Presented at BSDCan'09, May 2009.

`http://www.tarsnap.com/scrypt.html`

Acceptable Practices for Password Storage

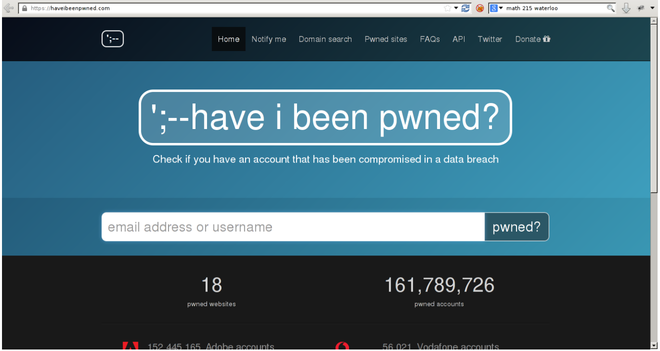
- not **plaintext!**
- hashed and salted

One-way function:

- $x \mapsto f(x)$ easy to compute; but
- $f(x) \overset{?}{\mapsto} x$ hard to reverse.

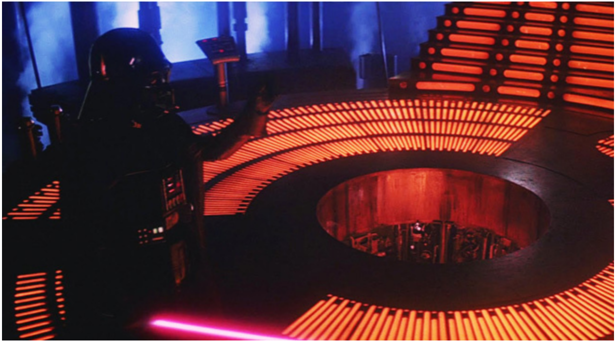
Examples: SHA1, scrypt.

Perhaps passwords have already been leaked!



The first thing is to try really common passwords.

You just might get a hit!



"All too easy."

How can we reverse the hash function?

- Brute force.

GPUs (or custom hardware) are good at that!

The Arms Race: Making Cracking Difficult

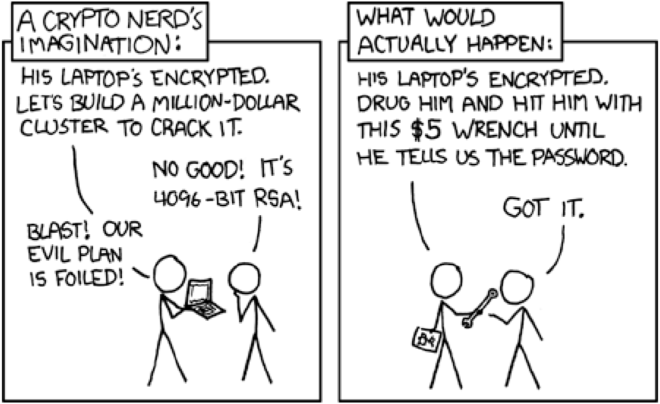
Historically: force repeated iterations of hashing.

Main idea behind scrypt (hence DogeCoin):

- make hashing expensive in time and space.

Implication: require more circuitry to break passwords.
Increases both # of operations and cost of brute-forcing.

Of course, there's always this form of cracking:



(Source: xkcd 538)

Formalizing "expensive in time and space"

Definition

A memory-hard algorithm on a Random Access Machine is an algorithm which uses $S(n)$ space and $T(n)$ operations, where $S(n) \in \Omega(T(n)^{1-\varepsilon})$.

Such algorithms are expensive to implement in either hardware or software.

Generalizing to functions

Next, add a quantifier:
move from particular algorithms to underlying functions.

A *sequential memory-hard function* is one where:

- the fastest sequential algorithm is memory-hard; and
- it is impossible for a parallel algorithm to asymptotically achieve lower cost.

Exhibit. ReMix is a concrete example of a sequential memory hard function.

The scrypt paper concludes with an example of a more realistic (cache-aware) model and a function in that context, BlockMix.

Across the Rainbow Bridge

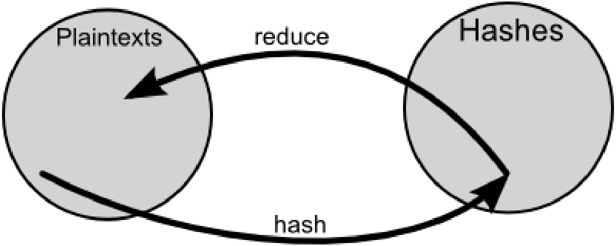
If designed well, hash functions aren't easy to brute force.

What if we remembered some previous work?

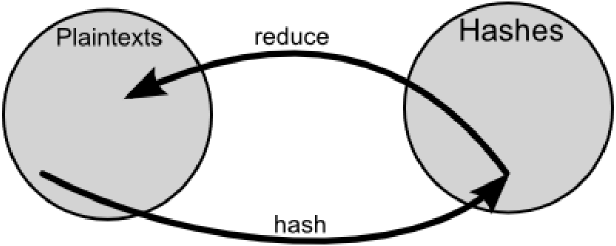
It isn't practical, or possible, to store the hashes for every possible plaintext.

The rainbow table is a compromise between speed and space.

The “reduction” function maps hashes to plaintext:



Example: `123456` $\rightarrow$ `d41d8cd98f00b204e9800998ecf8427e` $\rightarrow$ `418980`



We should do this to develop some number of chains.

This is the sort of task you could do with a GPU, because they can do a reduction relatively efficiently.

Or, y'know, just download them

Once we have those developed for a specific input set and hash function, they can be re-used forever.

You do not even need to make them yourself anymore (if you don't want) because you can download them on the internet... they are not hard to find.

They are large, yes, but in the 25 - 900 GB range.

I mean, Fallout 76 had a day one patch of 52 GB, and it was a disaster of a game.

How to Use Rainbow Tables

1. Look for the hash in the list of final hashes; if there, we are done
2. If it's not there, reduce the hash into another plaintext and hash the new plaintext
3. Go back to step 1
4. If the hash matches a final hash, the chain with the match contains the original hash
5. Having identified the correct chain, we can start at the beginning of the chain with the starting plaintext and hash, check to see if we are successful (if so, we are done); if not, reduce and try the next plaintext.

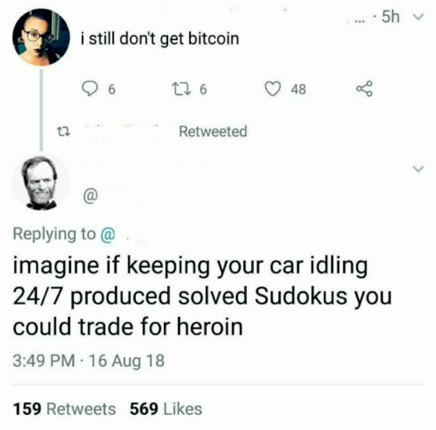
Is it Fast?

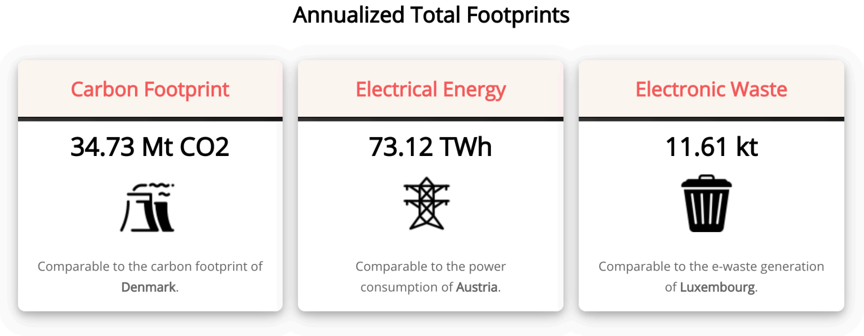
Like generation, checking the tables can also be done efficiently by the GPU.

Some numbers from
`http://www.cryptohaze.com/gpurainbowcracker.php.html`:

- Table generation on a GTX295 core for MD5 proceeds at around 430M links/sec.
- Cracking a password 'K#n&r4Z': real: 1m51.962s, user: 1m4.740s. sys: 0m15.320s

Yikes.


· 5h
i still don't get bitcoin
6
6
48
Retweeted
@
Replying to @
imagine if keeping your car idling 24/7 produced solved Sudokus you could trade for heroin
3:49 PM · 16 Aug 18
159 Retweets 569 Likes



https://digiconomist.net/bitcoin-energy-consumption As of Dec 2019

Basis: SHA-256

Started with CPU... then GPU... then what?

Custom hardware: optimize exactly what you need.

First FPGA, then ASIC...

Please Don't

We've uncovered why you should not mine Bitcoin.

Not cost efficient; way too much competition.

Not a hardware course, but this is the logical extension of GPU...