2550 Intro to cybersecurity

L3: Hash Functions

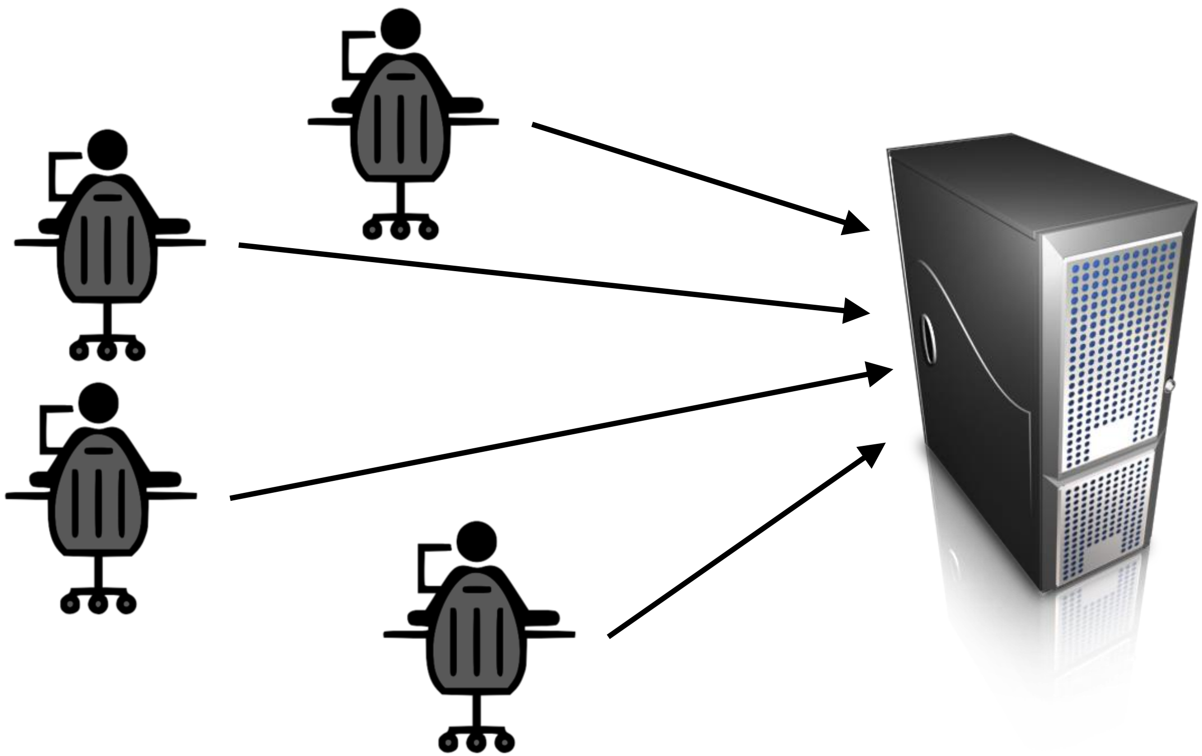
Ran Cohen & abhi shelat

Agenda

- What are hash functions
- Security notions
- Popular hash functions
- Slow hashing

Recap: Authentication

- Clients authenticate to a server using passwords
- Server storing pwds in the clear - exposed
- Common solution: stores hashed pwds



username1	h(pwd1)
username2	h(pwd2)
username3	h(pwd3)
username4	h(pwd4)

⋮

Basic notation

- A set is a collection of distinct elements
- $x \in X$ means: an element x belongs to the set X
- $|X|$ stands for the size (cardinality) of the set X
- $\{0,1\}^n$ is the set of all binary strings of length n
  - $\{0,1\}^1$ is 0 and 1
  - $\{0,1\}^2$ is $00, 01, 10, 11$
  - $\{0,1\}^3$ is $000, 001, 010, 011, 100, 101, 110, 111$
- There are 2^n binary strings of length n, i.e. $|\{0,1\}^n| = 2^n$
- $\{0,1\}^*$ is the set of all binary strings of finite length, i.e.,

$$\{0,1\}^* = \bigcup_{n \in \mathbb{N}} \{0,1\}^n$$

What's a hash function

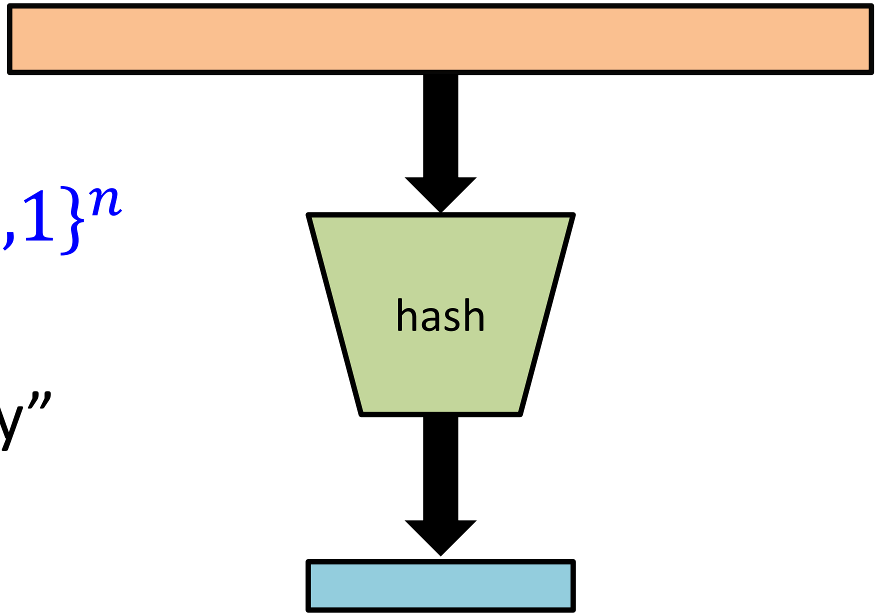
A hash function maps strings of arbitrary length to a fixed-length output (digital fingerprint)

$h: \{0,1\}^* \to \{0,1\}^n$ for some $n \in \mathbb{N}$

Desired properties:

- Compressing
- Given $x \in \{0,1\}^*$ and $y \in \{0,1\}^n$ can verify whether $y = h(x)$
- Output distributed "randomly" (minimize collisions)

Arbitrary length input

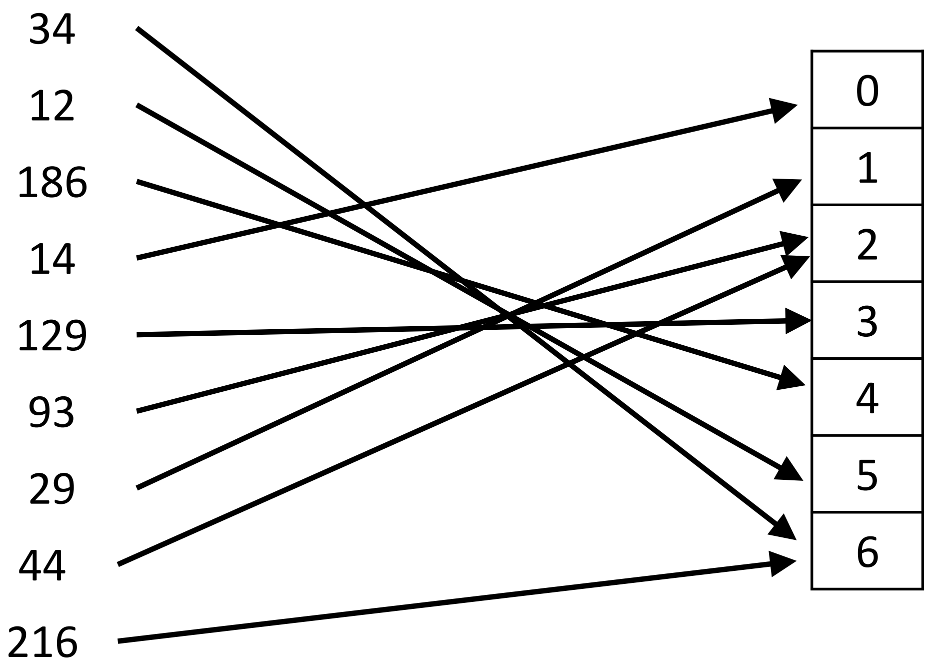


Fixed length output (digest)

Non-cryptographic hash

Heavily used in data structures

- Emphasis: reduce collisions
- Enables, e.g., constant-time lookup
- Example: $x \mapsto x \; mod \; 7$



Non-cryptographic hash

Is it good enough?

- In data structures collision resistance is a desire (better performances) but not crucial

  For security it is a necessity:
  the ability to find collisions yields attacks

- In data structures we can assume that elements are chosen independently of the hash function

  In security we consider adversaries that choose inputs with the explicit goal of finding collisions

Cryptographic hash

Three security flavors for $h: \{0,1\}^* \rightarrow \{0,1\}^n$:

- Collision resistant (CR)
  it is "hard" to find $x \neq x'$ such that $h(x) = h(x')$
- Second preimage resistance / Target-collision resistant (TCR)
  given x it is "hard" to find x' such that $h(x) = h(x')$
- Preimage resistance / One way (OW)
  given y it is "hard" to find x such that $h(x) = y$

Yes: If consider only long inputs (at least $2n$ bits)
No: is short inputs allowed

CR $\Rightarrow$ $\not\Leftarrow$ TCR $\Rightarrow$ $\not\Leftarrow$ OW

Generic attacks #1

Given a hash function $h: \{0,1\}^* \rightarrow \{0,1\}^n$

How many values should be tested to find a collision?

There are 2^n potential output values for h

$\Rightarrow$ a collision is guaranteed after testing $2^n + 1$ values (by the pigeonhole principle)

Wikipedia

Generic attacks #2

Given a hash function $h\colon \{0,1\}^* \to \{0,1\}^n$

How many values should be tested to find a collision with good probability?

About $\sqrt{2^n} = 2^{n/2}$ (by the birthday paradox)

- Choose m random strings $x_1, \ldots, x_m$
- There are $\frac{m(m-1)}{2} \approx m^2$ pairs
- For $i \neq j$, $h(x_i) = h(x_j)$ with probability $1/n$
- Expected number of pairs that collide is $\approx \frac{m^2}{n}$
- For $m = \sqrt{2^n}$ it is ≈ 1

Common hash functions

Hash function	Year	Digest length	Security
MD4	1990	128 bits	64 bits
MD5	1992	128 bits	64 bits
SHA-1	1995	160 bits	80 bits
SHA-2	2001	256/512 bits	128/256 bits
SHA-3	2015	256/512 bits	128/256 bits

Is it really an attack?

The birthday paradox yields random collisions – is it meaningful

Yes!!!!

Can generate fire/recommendation letters with same hash

Mr Ran Cohen is a bad teacher and should be fired

Is it really an attack?

The birthday paradox yields random collisions – is it meaningful

Yes!!!!

Can generate fire/recommendation letters with same hash

Mr	Ran Cohen is a	bad	teacher	and	should	be	fired
Mister		horrible	instructor		is ought to		terminated
Dr		poor	lecturer				canned
Dr.							

There are $4 \cdot 3 \cdot 3 \cdot 2 \cdot 3 = 72$ variations

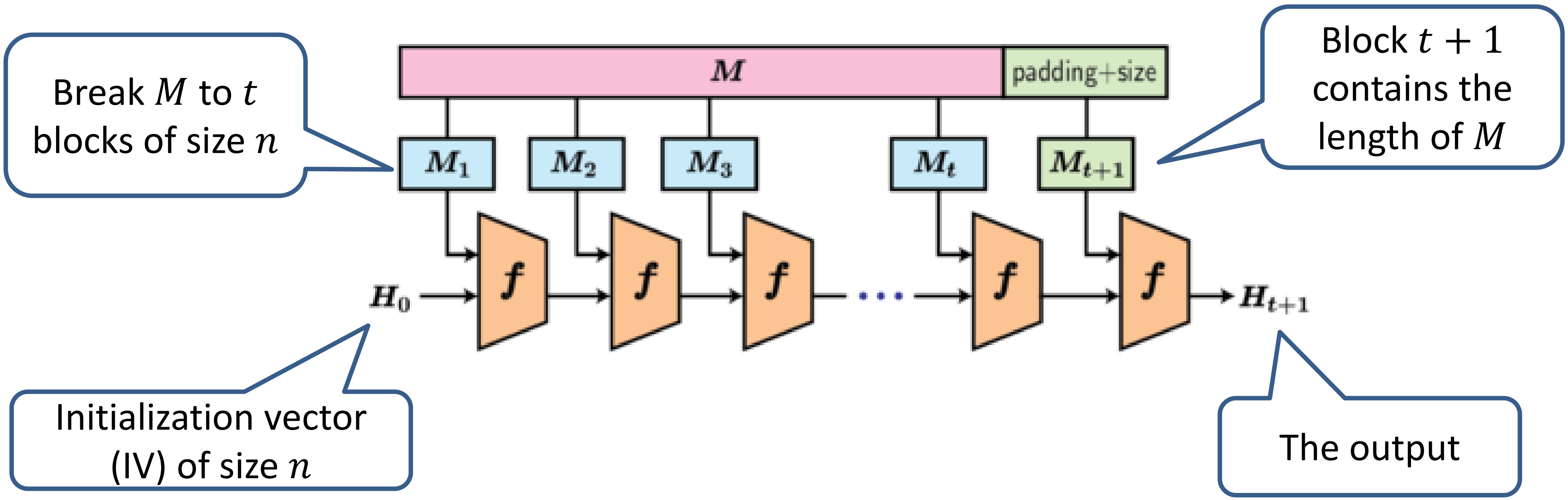
Generate letter 1 where 64 words have synonyms $\Rightarrow$ 2^{64} variations

Similarly generate 2^{64} variations of letter 2

$\approx 2^{128}$ pairs

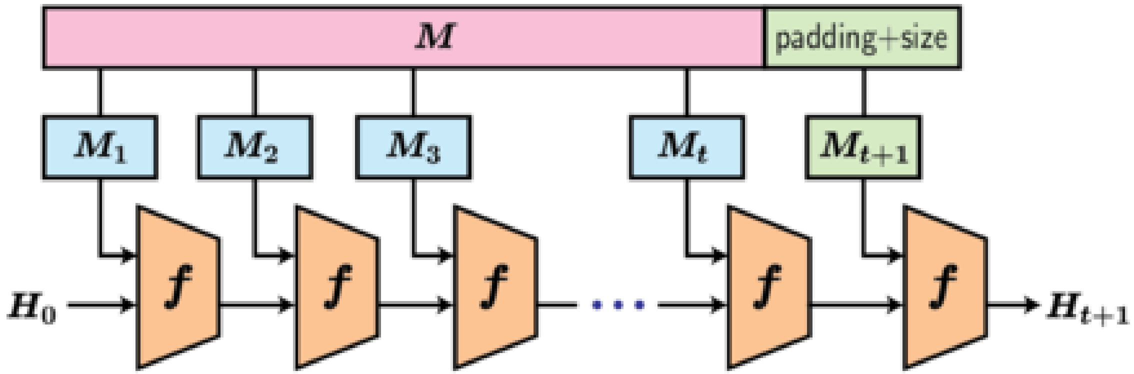
Domain extension

- Let $f: \{0,1\}^{2n} \to \{0,1\}^n$ be a compressing function
- The Merkle-Damgård transform constructs a hash function $h: \{0,1\}^* \to \{0,1\}^n$ as follows



Theorem: if f is CR then h is CR

Domain extension



Theorem: if f is CR then h is CR

Proof idea: assume that h is not CR and prove that f is not CR (this will contradict the assumption).

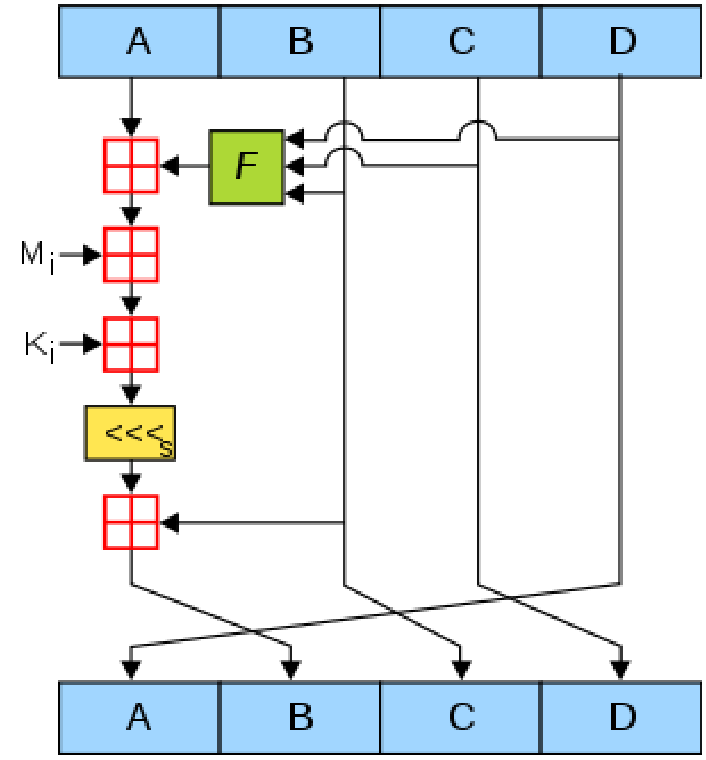
If h is not CR we can find $x \neq x'$ such that $h(x) = h(x')$

Case 1: if $|x| \neq |x'|$ then there is a collision for f in the last block (that contains the size)

Case 2: if $|x| = |x'|$ then go backwards and check the blocks. A collision for f must exist otherwise $x = x'$

MD5

- Message Digest
- Ron Rivest introduced MD4 (1990)
- Weaknesses found in MD4
- Strengthened to MD5 (1992)
- Generates 128 bit digest
- Based on MD transform
- Very fast, very popular
- Conjectured to offer 2^{64} security
- Completely broken (still used)



Wikipedia

MD5 history

1993	compression function collisions (pseudo collisions)
1996	free-start collision
2004	practical collision attack (1 hour on cluster)
2005	collision in 8 hours on laptop
2006	collision in 1 minute
2007	colliding X.509 Certificates for different identities
2007	extracting passwords from APOP using MD5 collisions
2009	Rogue CA Certificate of RapidSSL This allows issuing new “valid” certificates to anyone

Flame super malware

- Very complex malware
- Used for targeted cyber-espionage in middle east
- Operated 2010-2012
- Discovered May 28, 2012 by Kaspersky Lab
- Implemented a novel collision attack on MD5
- Impersonated a legitimate security update from Microsoft

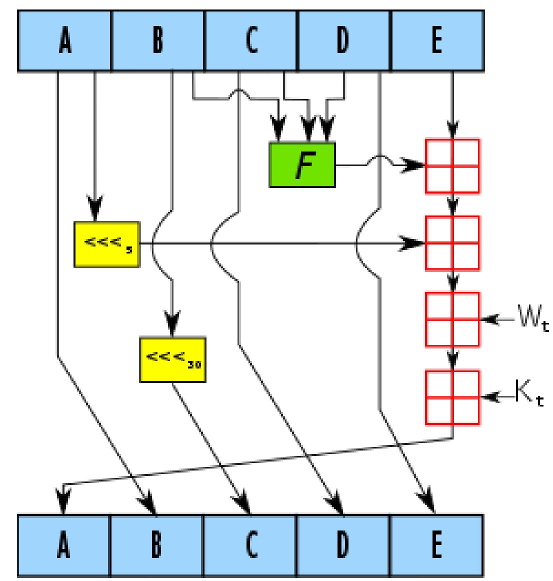
siliconangle.com

wired.com

SHA1

- Secure Hash Algorithms
- SHA-0 introduced in 1993 by NIST
- Based on MD5
- Digest length 160 bits
- NIST announced SHA-0 is vulnerable
- Introduces SHA-1 in 1995

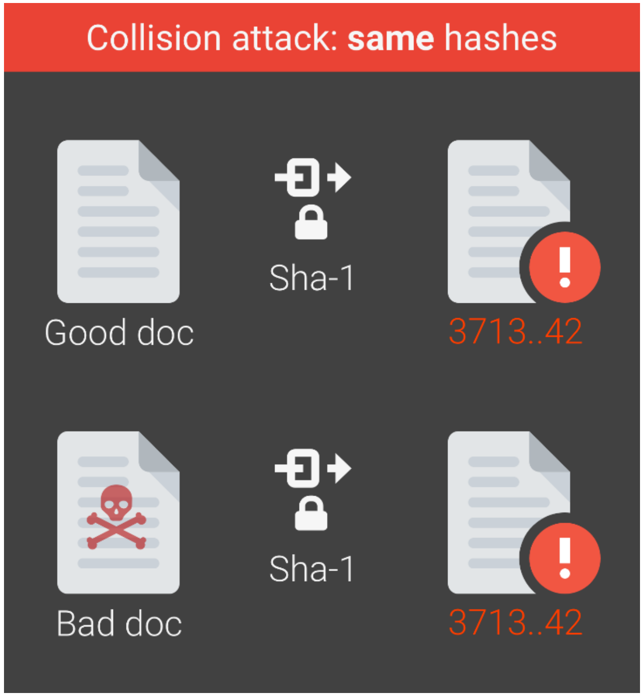
NIST
National Institute of Standards and Technology



Wikipedia

SHA-1 history

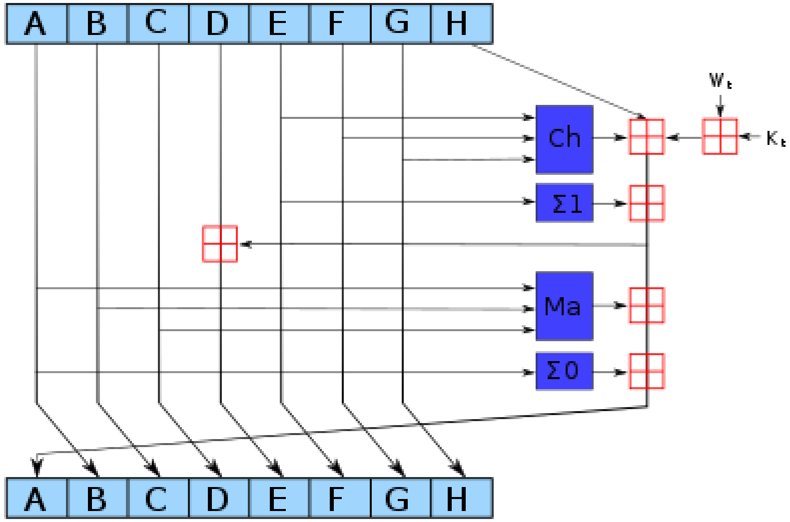
1995	SHA-1 introduced, collision should take $\approx 2^{80}$ steps
2005	Collision in 2^{69} steps
2015	Free-start collision in 2^{57} steps
2017	First collision found in $\approx 2^{63}$ steps



shattered.io

SHA-2

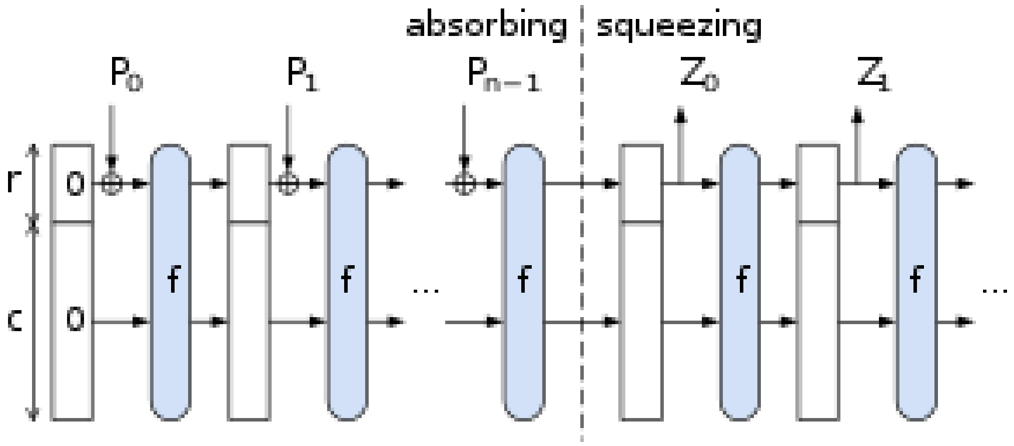
- Designed by NSA in 2001
- Various digest length:
  SHA-224, SHA-256, SHA-384, SHA-512
- Based on MD transform
- Known attacks only on weakened version
- Bitcoin uses SHA-256



Wikipedia

SHA-3

- In 2007 NIST announced a competition for new hash
- The winner Keccak designed in 2008
- Accepted as standard in 2015
- NOT based on MD transform
- Novel sponge construction



Wikipedia

Slow hashing

- MD5/SHA-1/SHA-2/SHA-3 are fast & deterministic
- This is good for many applications:
  - Digital signatures
  - Message authentication codes (MACs)
  - File integrity
  - Commitment schemes
  - Many more
- NOT GOOD FOR STORING PASSWORDS

The slower the better

- Recall last lecture
  - Offline brute force attacks
  - Rainbow tables
  - Time-memory tradeoffs
- Idea #1 – stored passwords should always be salted
  - digest = Hash(salt + password)
  - Store (salt, digest)
- Idea #2 – evaluating the hash function should be slow
  - Multiple iterations
  - Use memory

Bcrypt

- Designed in 1999
- Always uses salt
- Adjustable number of rounds
  - Cost $r \Rightarrow 2^r$ rounds

```
$2a$10$N9qo8uLOickgx2ZMRZoMyeIjZAgcfl7p92ldGxad68LJZdL17lhWy
\__/\/ \____________________/\_____________________________/
 Alg Cost      Salt                        Hash
```

Wikipedia

128 bits

192 bits

scrypt

- Designed in 2009
- Requiring large amounts of memory
  - Generates a vector of pseudorandom strings
  - Access the vector in a pseudorandom manner
- Either need to store the vector in RAM, or generate on the fly (which is computationally expensive)
- Used for blockchains such as Litecoin

Recap

- Discussed what's a hash function
- Different security notions
- Generic attacks
- The Merkle-Damgård transform
- Fast hash functions
- Slow hash functions