Hash functions: in data structures

- hash function is a compression function
- arbitrary input length
- $H : \{0,1\}^* \mapsto \{0,1\}^n$, typically $n = 128, 160, 256, etc$
- used in data structures and algorithms (set, map, etc.)
  - elements stored in a table of size k.
  - find and add operations in $O(1)$ time (amortized)
  - Key x stored in cell at index $H(x)$
  - Find x by computing $H(x)$ and looking at corresponding cell
- *collision*: $x \neq x' : H(x) = H(x')$
- collisions should not be too frequent
- good: distributes elements „evenly” accross the table

Hash functions: cryptography

- compress
- few collisions
- avoiding collisions
  - algo: good for running times, no strict avoidance, rather minimization
  - crypto: it's a must
- No special interest in values of x and $H(.)$ in algo
- Attacker may choose x in crypto
- Cryptographic hash: more of a challenge

Cryptographic hash functions

Definition

A collision of function $H(.)$ is a pair $x \neq x'$ with $H(x) = H(x')$.
A function $H(.)$ is collision-free, if any PPT attacker has only negligible probability of finding a collision.
A function $H(.)$ is a hash function if $H : \{0,1\}^* \mapsto \{0,1\}^n$.

Weaker security assumptions

1. Collision-free
2. *Second-preimage resistance*: for a given x, no PPT attacker can find another $x' \neq x : H(x') = H(x)$
3. *Preimage resistance*: for a given $y = H(x)$ which is obtained from a random (unknown) x, no PPT adversary can find $x' : H(x') = y$ („one-way function")

Cryptographic hash functions

Definition

A collision of function $H(.)$ is a pair $x \neq x'$ with $H(x) = H(x')$.
A function $H(.)$ is collision-free, if any PPT attacker has only negligible probability of finding a collision.
A function $H(.)$ is a hash function if $H : \{0,1\}^* \mapsto \{0,1\}^n$.

Weaker security assumptions

1. Collision-free
2. *Second-preimage resistance*: for a given x, no PPT attacker can find another $x' \neq x : H(x') = H(x)$
3. *Preimage resistance*: for a given $y = H(x)$ which is obtained from a random (unknown) x, no PPT adversary can find $x' : H(x') = y$ („one-way function")

Cryptographic hash functions

Design principles

- Collision-free
- Second-preimage resistance
- Preimage resistance
- *Avalanche effect*: small change in input $\Rightarrow$ large change in output
    - *Strict avalanche criterion:* changing an input bit $\Rightarrow$ changes all output bits with prob. 1/2
    - *Bit independence criterion:* $\forall i, j, k$: changing input bit $i \Rightarrow$ change in output bits j, k independent

Attacks

Birthday attack

Let $H : \{0,1\}^* \mapsto \{0,1\}^n$ be a hash function. By computing (roughly) $2^{n/2}$ hash values, we will find a collision with prob. 1/2.

- Faster than brute force
- $\Longrightarrow n \geq 160$
- „Breaking" a hash function usually means an attack which beats the birthday attack

Merkle-Damgård construction

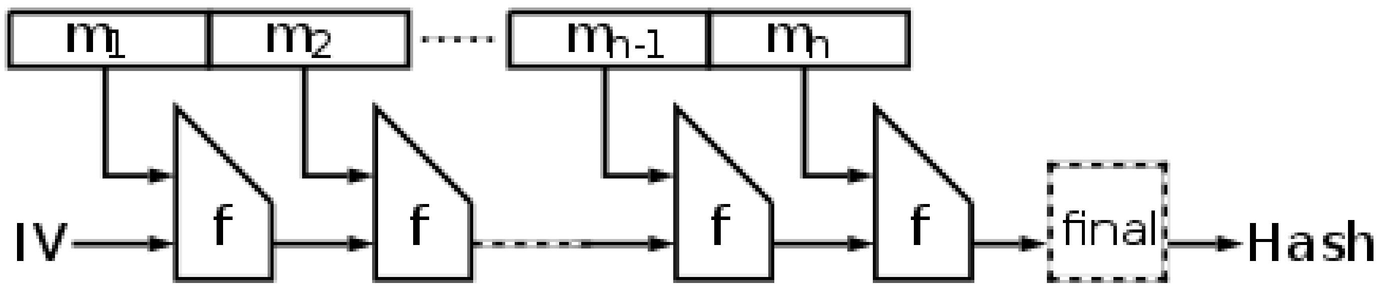
Basic idea

- In practice, input length is usually fixed
- this construction enables the use of arbitrary inputs
- Let $h : \{0,1\}^{2n} \mapsto \{0,1\}^n$ a hash function with fixed length inputs, $m \in \{0,1\}^*$, ahol $|m| = \ell < 2^n$
- construction uses chaining
- $\Rightarrow H(.)$ obtained with arbitrary inputs

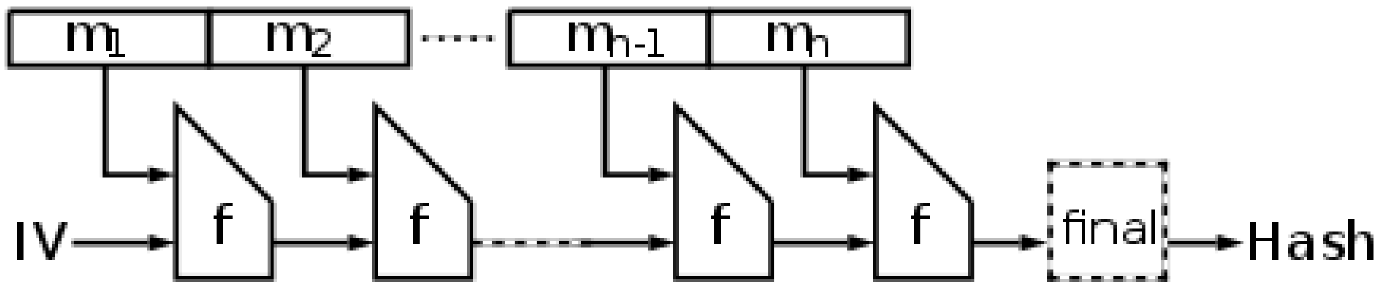
Merkle-Damgård construction

Merkle-Damgård transform

1. Split m into blocks of length n: $b := \lceil \frac{\ell}{n} \rceil$ és $m = (m_1|m_2|\ldots|m_b)$
2. Let $m_{b+1} := \ell \in \{0,1\}^k$, $z_0 := IV$
3. For $i = 1, \ldots, b+1$, compute $z_i := h(z_{i-1}|m_i)$
4. $H(m) := z_{b+1}$



Merkle-Damgård construction

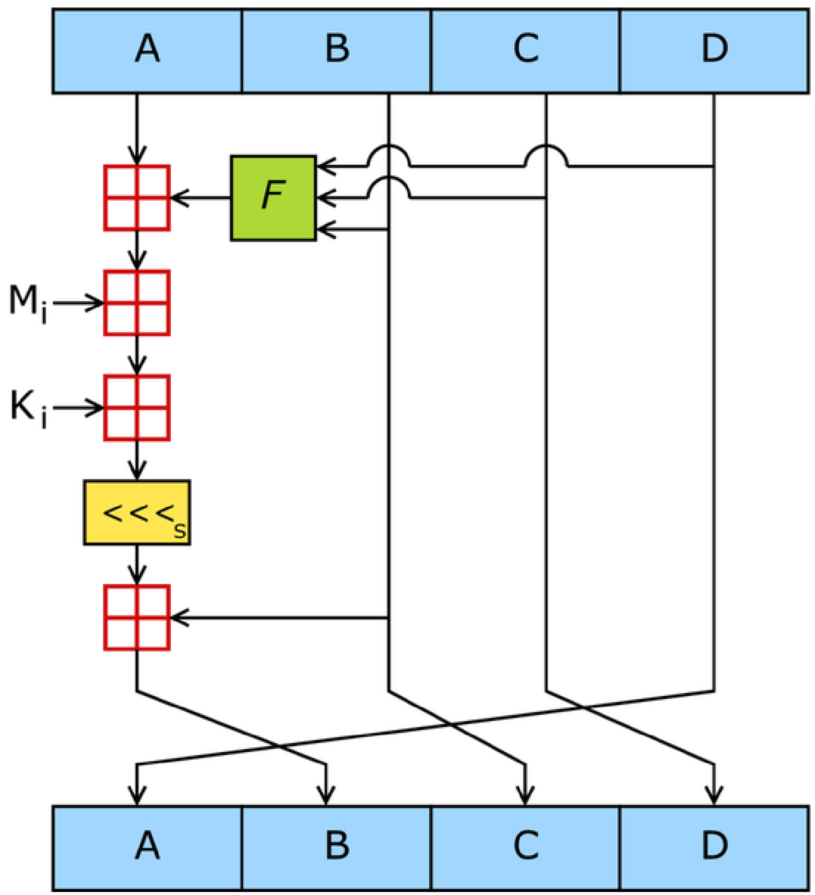


Properties

- in practice: suffices to have a hash for fixed input length
- theoretically: any compression ratio is fine
- $IV : z_0$ free to choose
- $h(.)$ collision-free $\Rightarrow H(.)$ collision-free

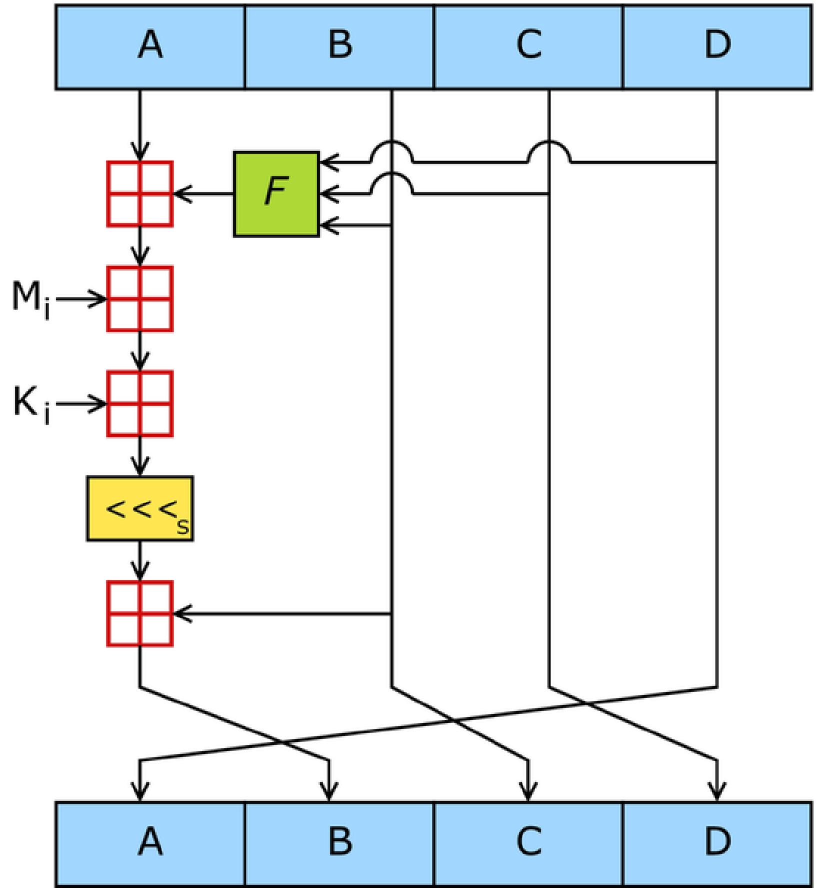
MD5 - description

- 512-to-128 bit compression and Merkle-Damgård
- Works on 32-bit words
- m broken up onto 512(=16*32)-bit blocks
- Operates on 128(=4*32)-bit „states”
- A, B, C, D fixed
- 4 rounds, 16 operations per round
- 4 non-linear F:
  1. $F(B, C, D) = (B \wedge C) \vee (\neg B \wedge D)$
  2. $G(B, C, D) = (B \wedge D) \vee (C \wedge \neg D)$
  3. $H(B, C, D) = B \oplus C \oplus D$
  4. $I(B, C, D) = C \oplus (B \vee \neg D)$
- M_i message block
- K_i costant, s shift parameter (varies for each operation)



MD5 – analysis

- Historical importance, collisions can be found!
- 128 bit output $\Longrightarrow$ birthday attack
- 1992 - MD5 published
- 1993 - „pseudo-collision” in compression function (IV -based attack)
- 1996 - collision in compression function
- 2004 - MD5CRK, distributed birthday attack
- 2004 - hash collision in under an hour (analytic attack)
- 2005 - collision in two X.509 certificates, different key, same MD5 hash
- 2010 - first published one/block collision



SHA family of hash functions

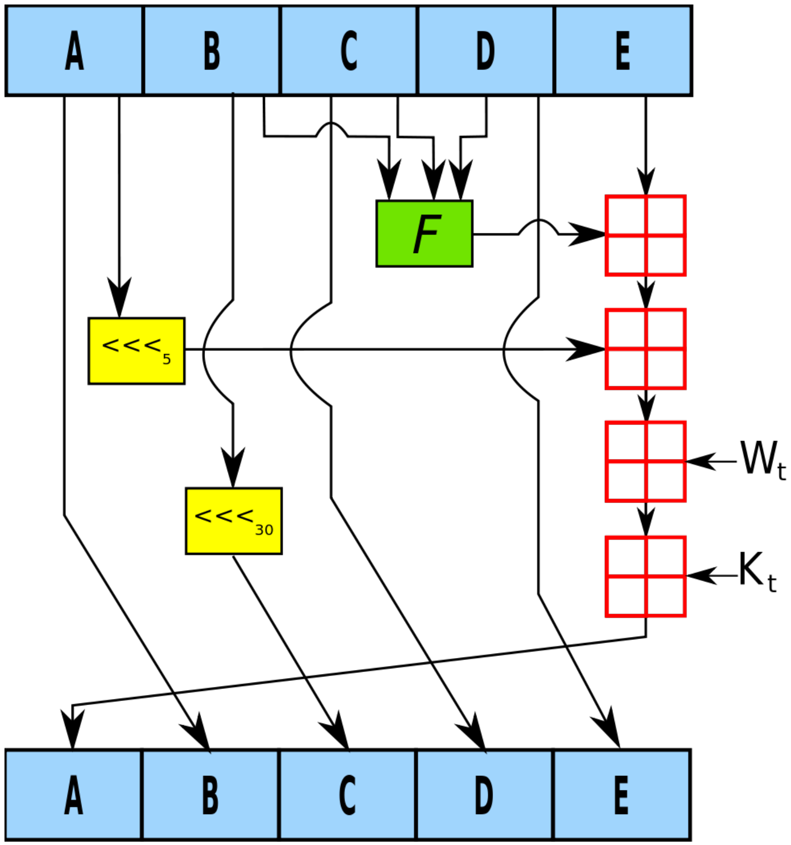
- SHA – Secure Hash Algorithm
- U.S. NSA, U.S. NIST

SHA-0 (1993)
- 160-bit output, 32-bit words, 80 rounds
- operations: $\oplus, \boxplus, \wedge, \vee, \lll$
- collision...

SHA-1 (1995)
- 160-bit output, 32-bit words, 80 rounds
- more resistant, theoretical attack in 2^{61} time (2011)

SHA-2 (2001) = SHA-256/SHA-512
- 256/512-bit output, 32/64-bit words, 64/80 rounds
- no (known) collisions

SHA-3 (2014-)
- different design
- alternative to SHA-2



SHA-1, original diagram for Wikipedia created

by User:Matt Crypto

NIST hash competition (2007 – 2012)

Similar to AES competition

Oct. 2008	deadline for submissions
Dec. 2008	Round 1: 51 candidates remain
Feb. 2009	NIST conference: submitted candidates
Jul. 2009	Round 2: 14 candidates
Aug. 2010	CRYPTO 2010: analyze round 2 candidates
Dec. 2010	Finalists announced

- performance: modest hardware requirements
- security: crypto/design weaknesses
- analysis: cryptanalysis by the entire crypto community
- diversity: various modes of operations and internal states

Dec. 2012	winner: Keccak
Aug. 2013	NIST announces changes compared to the standardized hash for „better security/performance”
Aug. 2015	Keccak is new SHA-3 hash standard

SHA-3/Keccak

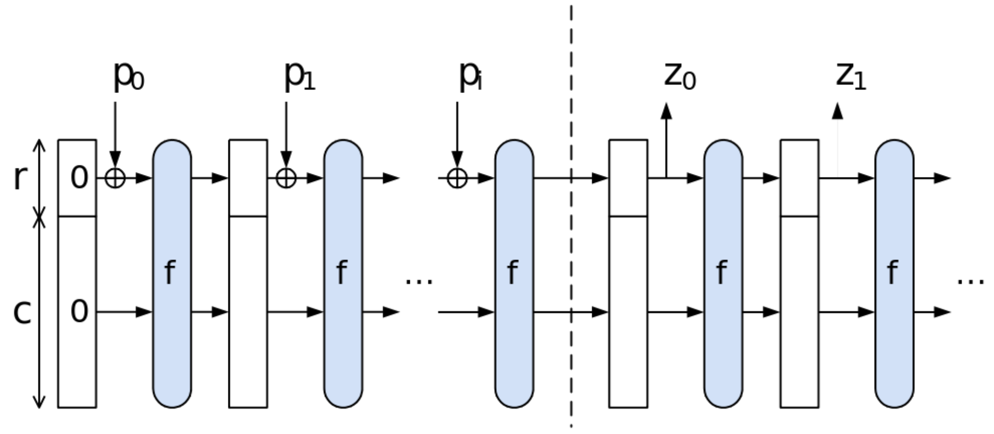


Diagram of a sponge construction from http://sponge.noekeon.org/

- Bertoni, Daemen, Peteers , Van Assche
- *Sponge construction* – permutation f of fixed length + padding rule:

1. m padded and broken up into r-bit blocks p_i
2. absorption: iteration of f: XOR of p_i with output of f from the previous block. All blocks are „absorbed" into internal state
3. squeeze out: extract output blocks z_i from (continuously updated) internal state.

Security vs. Integrity

- Secure communication
  - Alice sends message to Bob
  - in an open communication channel
  - security
  - **tools:** encryption
- Integrity
  - Alice sends message to Bob
  - in an open communication channel
  - authenticity (ID of caller, email address)
  - integrity
  - notice change in the message
  - preventing change not a crypto challenge (physical countermeasures)
  - **tool:** ???

Security vs. Integrity

- Secure communication
  - Alice sends message to Bob
  - in an open communication channel
  - security
  - **tools:** encryption
- Integrity
  - Alice sends message to Bob
  - in an open communication channel
  - authenticity (ID of caller, email address)
  - integrity
  - notice change in the message
  - preventing change not a crypto challenge (physical countermeasures)
  - **tool:** ???

Security vs. message authentication

- Stream cipher
  - Let $c := E_k(m) = G(k) \oplus m$ a ciphertext with $G(.)$ a PRG
  - Changing a single bit in $c \implies$ changes same bit in m
  - Still secure
  - Similar with one-time pad
- Block cipher
  - OTR and CTR: same modification possible
  - more sophisticated for ECB, CBC
- encryption alone does not provide integrity
  - c hides m-et
  - BUT attacker can still mess around and modify c, thereby also m.
  - Any c corresponds to an m...
- need a new „layer”

Security vs. message authentication

- Stream cipher
  - Let $c := E_k(m) = G(k) \oplus m$ a ciphertext with $G(.)$ a PRG
  - Changing a single bit in $c \Longrightarrow$ changes same bit in m
  - Still secure
  - Similar with one-time pad
- Block cipher
  - OTR and CTR: same modification possible
  - more sophisticated for ECB, CBC
- encryption alone does not provide integrity
  - c hides m-et
  - BUT attacker can still mess around and modify c, thereby also m.
  - Any c corresponds to an m...
- need a new „layer”

Message Authentication Code (MAC): definition

Problem

- secret key shared between communicating parties
- authenticated message sent
- Has it been modified?
- Message Authentication Code (MAC)

Message Authentication Code (MAC): definition

Definition

A Message Authentication Code is a triple $(Gen, Mac, Vrfy)$ *with:*

- Gen *key generation: for security parameter* 1^n*, returns a key* k *with* $|k| \geq n$
- Mac *tag generation: for key* k *and message* $m \in \{0,1\}^*$ *returns a MAC-tag* $t := Mac_k(m)$
- $Vrfy$ *verification: for key* k*, tag* t *and message* m*, returns a bit* $b := Vrfy_k(m, t)$ *(*$b = 1$*, if* t *is a valid MAC-tag for* m*, otherwise 0.)*

The system fullfils the following correctnes definition:

$$Vrfy_k(m, Mac_k(m)) = 1.$$

MAC – security definition

What is an attack like?

- The attacker can:
  1. query MACs from Alice for various messages (examine how the message affects the tag)
  2. do some computation
  3. *forge* a MAC: a valid tag for a for some new message m (never queried before)
- security means that the attacker cannot perform the above attack efficiently

Definition

An authentication method is secure against adaptve chosen plaintext attack if any PPT adversary can only genarte a valid tag t for a message $m$with negligable probability even after querying several tags t' for messages $m' \neq m$.

MAC – security definition

Definition

An authentication method is secure against adaptive chosen plaintext attack if any PPT adversary can only genarte a valid tag t for a message $m$with negligable probability even after querying several tags t' for messages $m' \neq m$.

- too strong?
  - can query anything
  - any valid tag is a successful „break”
- in practice, only meaningful messages are of interest
- What’s „meaningful”
- *Replay attack*
  - solutions: timestamps or counter with m
  - drawback: synchronization or storage issues

MAC construction for fixes length meassages

Fixed length MAC

Let $PRF : \{0,1\}^n \mapsto \{0,1\}^n$ PRF. Then the following is a secure MAC.

Gen: $k \in_R \{0,1\}^n$

Mac: For key k and message $m \in \{0,1\}^n$, let the tag $t := PRF_k(m)$

$Vrfy$: Output 1 $\Leftrightarrow t = PRF_k(m)$

- if PRF secure, $\Rightarrow$ MAC secure
- drawback: fixed length m
- randomization (+ a few additional tricks): arbitrary length

MAC from hash: HMAC

HMAC

Let $h : \{0,1\}^{2n} \mapsto \{0,1\}^n$ and $H : \{0,1\}^* \mapsto \{0,1\}^n$ the Merkle-Damgård constructed hash. Let $IV, ipad, opad \in \{0,1\}^n$ fixed.

$$
\begin{aligned}
Gen&: k \in_R \{0,1\}^n \\
Mac&: t := h(h(IV|k \oplus opad)|H_{IV}(k \oplus ipad|m)) \\
Vrfy&: \text{output } 1 \Leftrightarrow t = Mac_k(m)
\end{aligned}
$$

- if h collision-free, then HMAC secure
- Merkle-Damgard not secure against so-called length extension attack