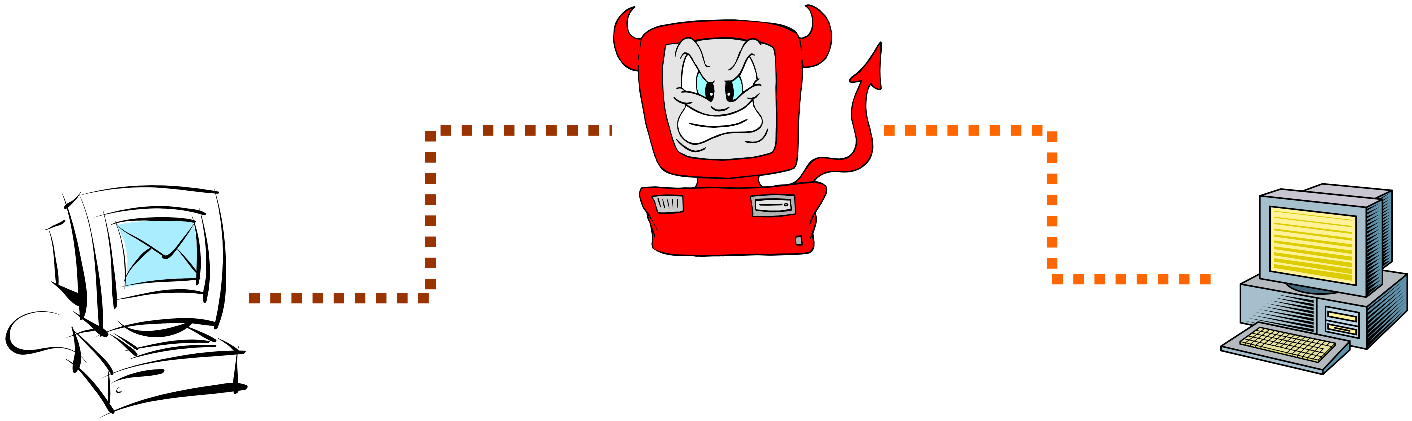
ENEE 459-C
Computer Security

Message authentication

Data Integrity and Source Authentication



- Encryption does not protect data from modification by another party.
  - Why?
- Need a way to ensure that data arrives at destination in its original form as sent by the sender and it is coming from an authenticated source.

Hash Functions

- A hash function maps a message of an arbitrary length to a m-bit output
  - output known as the fingerprint or the message digest

- What is an example of hash functions?
  - Given a hash function that maps Strings to integers in $[0,2^{32}-1]$
  - $F(x) = A x + b \mod q$, where $x = 0,1,\ldots,T$ where $T >> q$
  - Hash function used in the hash table data structure

Using Hash Functions for Message Integrity

- Method 1: Uses a Hash Function h, assuming an authentic (adversary cannot modify) channel for short messages
  - Transmit a message M over the normal (insecure) channel
  - Transmit the message digest h(M) over the secure channel
  - When receiver receives both M’ and h, how does the receiver check to make sure the message has not been modified?

- This is insecure. How to attack it?
- A hash function is a many-to-one function, so collisions can happen.

Non-crypto Hash (1)

- Data $X = (X_0,X_1,X_2,\ldots,X_{n-1})$, each X_i is a bit
- **hash**$(X) = X_0+X_1+X_2+\ldots+X_{n-1}$
- What is the compression of this hash?
- Show it does not satisfy preimage resistance
- Show it does not satisfy collision resistance

Non-crypto Hash (2)

- Data $X = (X_0,X_1,X_2,\ldots,X_{n-1})$
- Suppose hash is
  - $h(X) = nX_0+(n-1)X_1+(n-2)X_2+\ldots+1\cdot X_{n-1}$
- What is the compression of this hash?
- Show it does not satisfy preimage resistance
- Show it does not satisfy collision resistance

Non-crypto Hash (3)

- Cyclic Redundancy Check (CRC)
- Essentially, CRC is the remainder in a long division calculation
- Find a collision (modulo x^8+1)
- Good for detecting burst **errors**
- Easy to construct collisions
- CRC sometimes mistakenly used in crypto applications (WEP)

Cryptographic Hash Functions

Given a function $h: X \rightarrow Y$, then we say that h is:

- preimage resistant (one-way):

  if given $y \in Y$ it is computationally infeasible to find a value $x \in X$ s.t. $h(x) = y$

- 2-nd preimage resistant (weak collision resistant):

  if given $x \in X$ it is computationally infeasible to find a value $x' \in X$, s.t. $x' \neq x$ and $h(x') = h(x)$

- collision resistant (strong collision resistant):

  if it is computationally infeasible to find two distinct values $x', x \in X$, s.t. $h(x') = h(x)$

Relations between properties

- collision resistance $\Rightarrow$ 2nd preimage resistance
- 2nd preimage resistance ? preimage resistance

Find collisions for crypto-hashes?

- The brute-force birthday attack aims at finding a collision for a cryptographic function h
  - Randomly generate a sequence of plaintexts $X_1, X_2, X_3, \ldots$
  - For each X_i compute $y_i = h(X_i)$ and test whether $y_i = y_j$ for some $j < i$
  - Stop as soon as a collision has been found
- If there are m possible hash values, the probability that the i-th plaintext does not collide with any of the previous $i - 1$ plaintexts is $1 - (i - 1)/m$
- The probability F_k that the attack fails (no collisions) after k plaintexts is

$$F_k = (1 - 1/m)(1 - 2/m)(1 - 3/m) \ldots (1 - (k - 1)/m)$$

- Using the standard approximation $1 - x \approx e^{-x}$

$$F_k \approx e^{-(1/m + 2/m + 3/m + \ldots + (k-1)/m)} = e^{-k(k-1)/2m}$$

- The attack succeeds with probability p when $F_k = 1 - p$, that is,

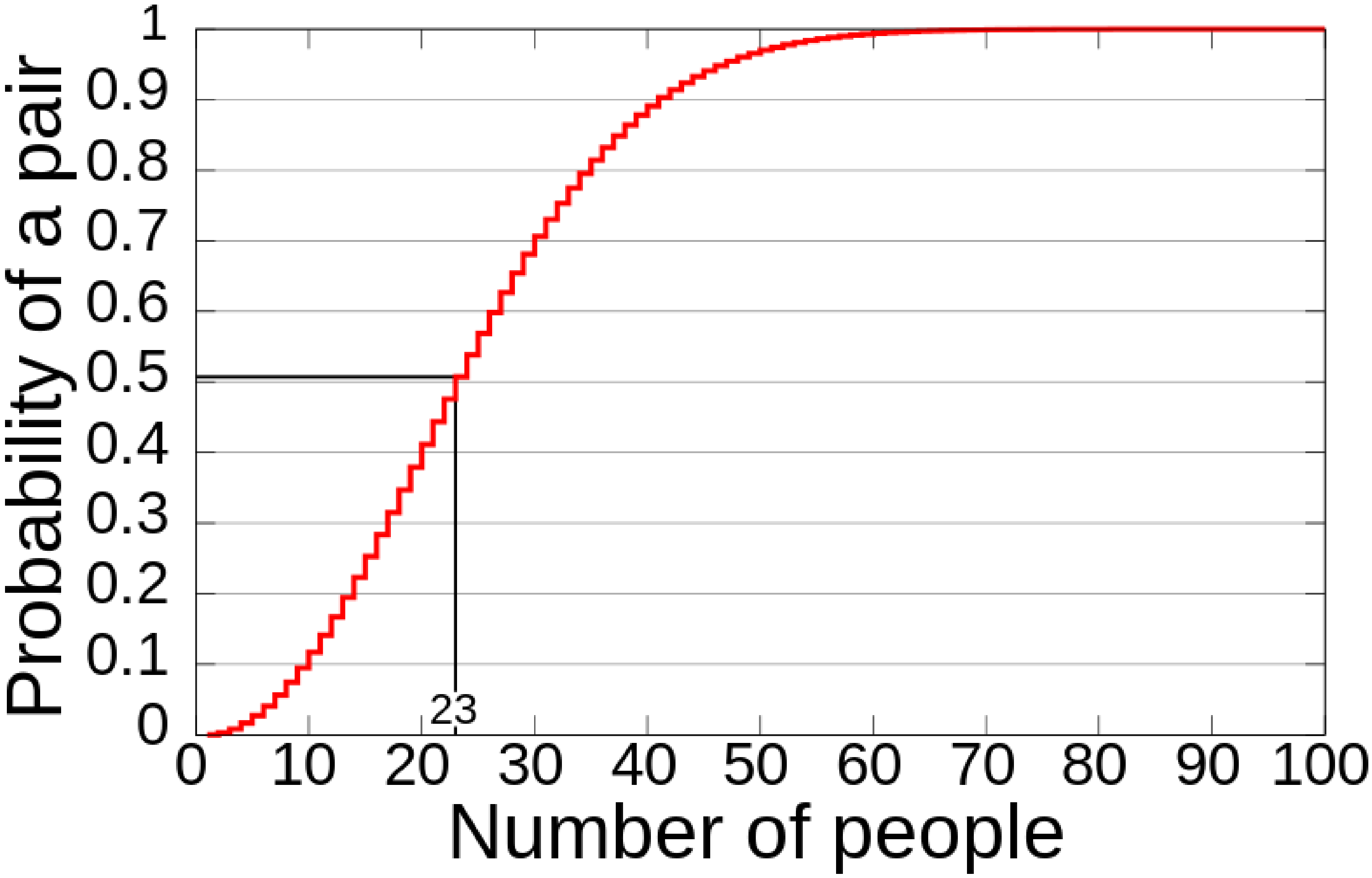
$$e^{-k(k-1)/2m} = 1 - p$$

- For p=1/2

$$k \approx 1.17\, m^{1/2}$$

- For m = 365, p=1/2, k is around 24

Birthday attack



Applications: Online Bid Example

- Suppose Alice, Bob, Charlie are bidders
- Alice plans to bid A, Bob B and Charlie C
  - They do not trust that bids will be secret
  - Nobody willing to submit their bid
- Solution?
  - Alice, Bob, Charlie submit **hashes** h(A),h(B),h(C)
  - All hashes received and posted online
  - Then bids A, B and C revealed
- Hashes do not reveal bids (which property?)
- Cannot change bid after hash sent (which property?)

Online Bid

- This protocol is not secure!
- A forward search attack is possible
  - Bob computes h(A) for likely bids A
- How to prevent this?
- Alice computes h(A,R), R is random
  - Then Alice must reveal A and R
  - Bob cannot try all A and R

Applications: Securing storage

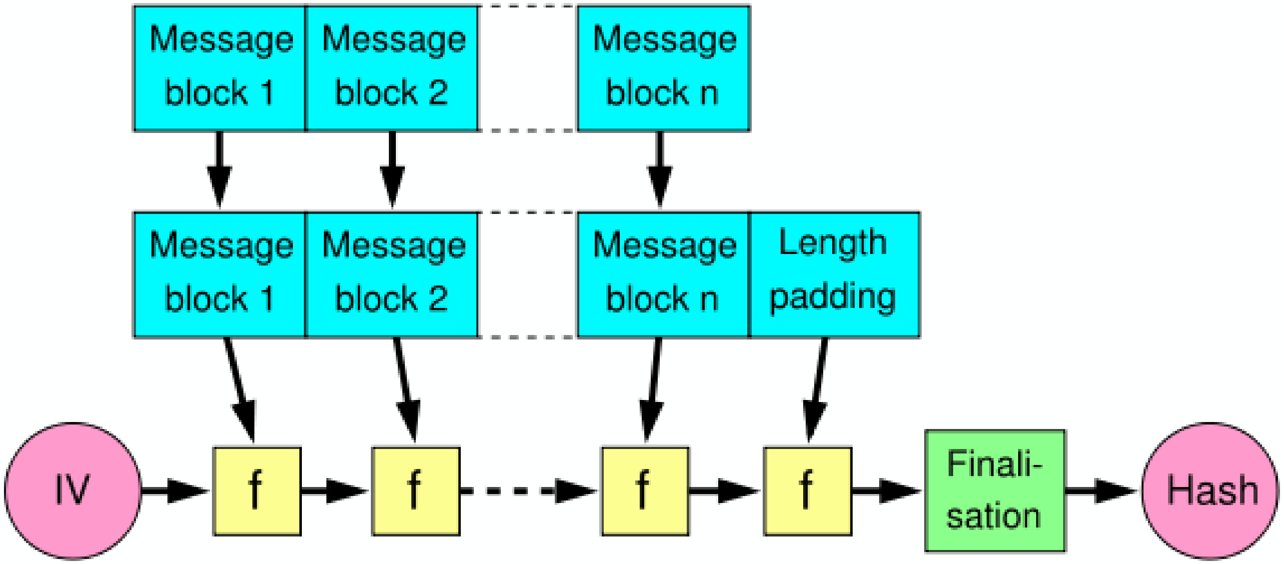
- Bob has files f1,f2,…,fn
- Bob sends to Amazon S3 the hashes
  - h(r||f1),h(r||f2),…,h(r||fn)
  - The files f1,f2,...,fn
- Bob stores randomness r (and keeps it secret)
- Every time Bob **reads** a file f1, he also reads h(r||fi) and verifies
- Any problems with **writes**?

Well Known Hash Functions

- MD5
  - output 128 bits
  - collision resistance completely broken by researchers in China in 2004
- SHA1
  - output 160 bits
  - considered insecure for collision resistance
- SHA2 (SHA-224, SHA-256, SHA-384, SHA-512)
  - outputs 224, 256, 384, and 512 bits, respectively
  - No real security concerns yet
- SHA3
  - Recently proposed
  - Not meant to replace SHA2

Merkle-Damgard Construction for Hash Functions

- Message is divided into fixed-size blocks and padded
- Uses a compression function f, which takes a chaining variable (of size of hash output) and a message block, and outputs the next chaining variable
- Final chaining variable is the hash value



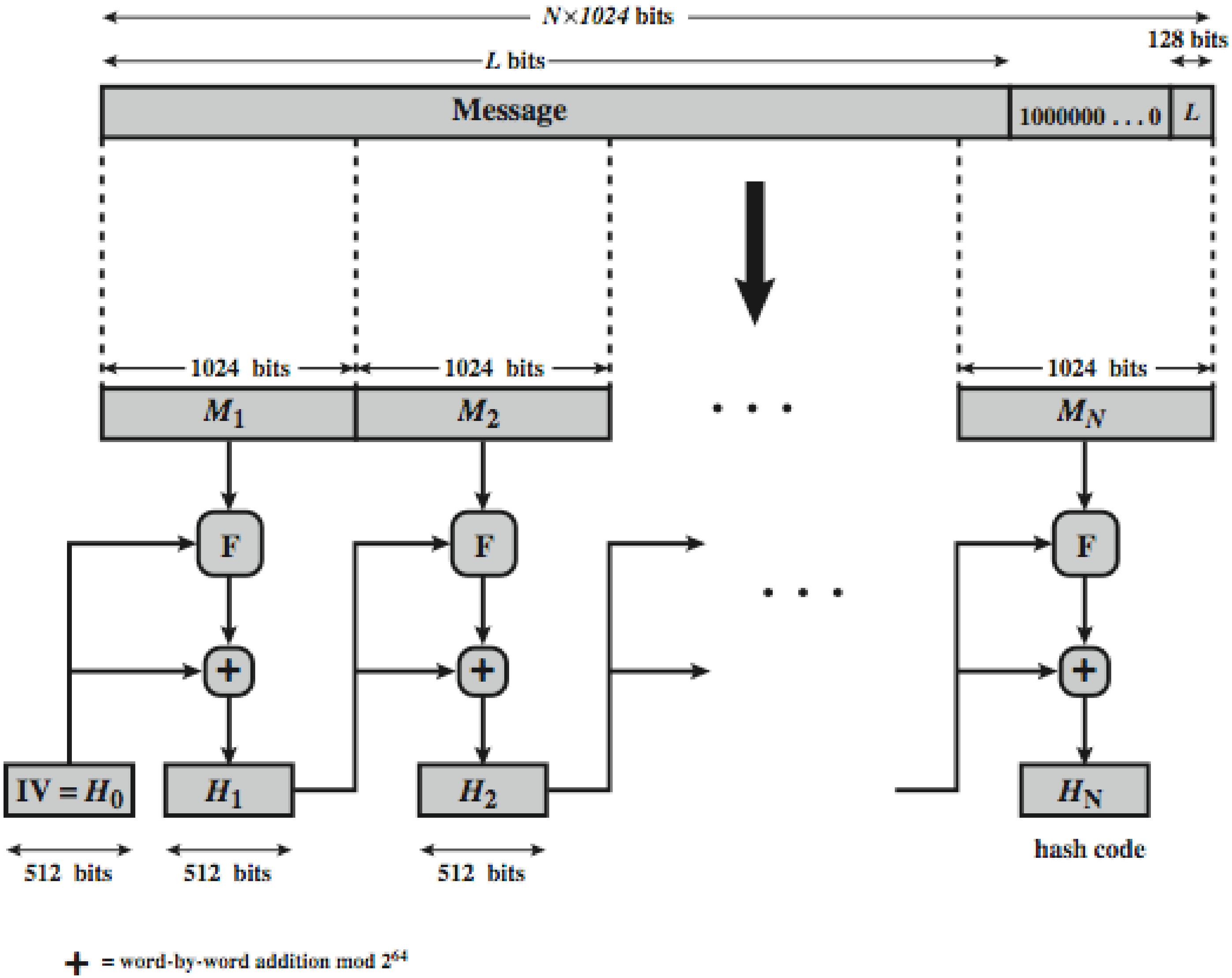
Merkle's meta-method

- any collision resistant compression function f can be extended to a CRHF
- Merkle's meta-method provides an efficient way to construct CRHF from f
  - n bit output, r bit chain variable
  - collision for h would imply collision for f for some stage i

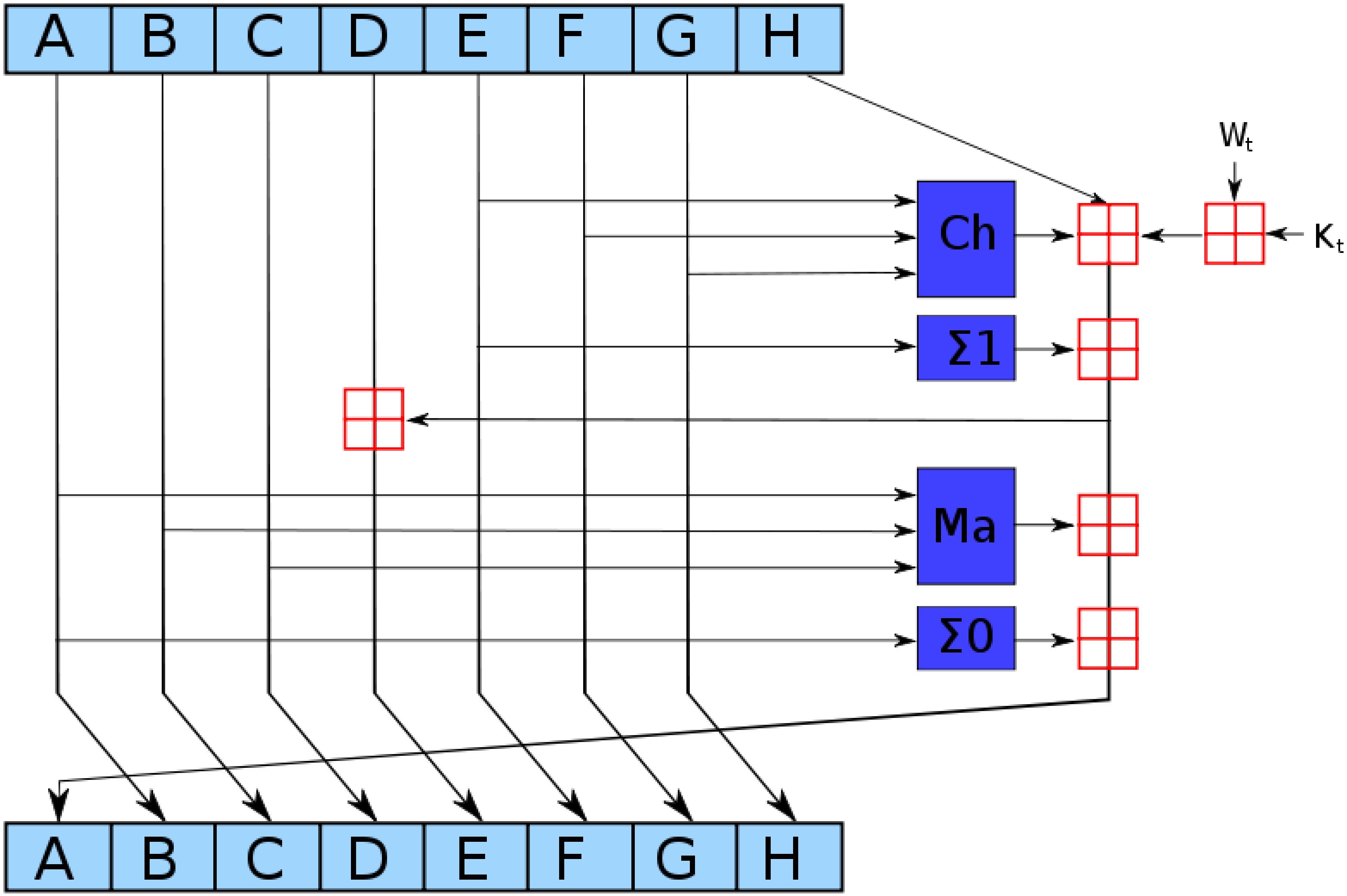
Message-Digest Algorithm 5 (MD5)

- Developed by Ron Rivest in 1991
- Uses 128-bit hash values
- Still widely used in legacy applications although considered insecure
- Various severe vulnerabilities discovered
- Collisions found by Marc Stevens, Arjen Lenstra and Benne de Weger

SHA-2 overview



SHA-2 internals



Limitation of Using Hash Functions for Authentication

- Require an authentic channel to transmit the hash of a message
  - Without such a channel, it is insecure, because anyone can compute the hash value of any message, as the hash function is public
  - Such a channel may not always exist
- How to address this?
  - use more than one hash functions
  - use a key to select which one to use