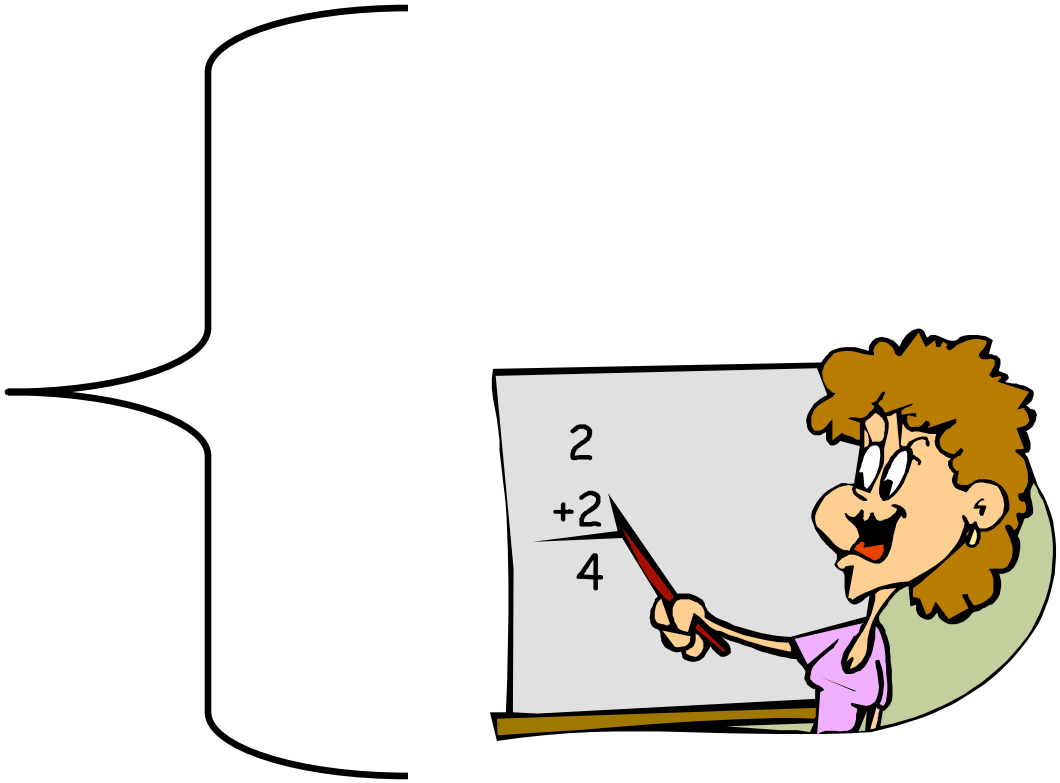
Cristina Nita-Rotaru

CS355: Cryptography

Lecture 15: Hash functions and message authentication codes.

Some Math …



Functions

Definition

Given two sets, X and Y, a function $f : X \rightarrow Y$, a function f from set X to set Y, is a relation which uniquely associates members of set X with members of set Y.

Terminology

X is called domain

Y is called range or codomain.

For $y = f(x)$ where $x \in X$ and $y \in Y$, y is called the image of x and x is called the preimage of y.

The number of functions from X to Y is $|Y|^{|X|}$

Image of a Function

Definition

Given a function $f : X \rightarrow Y$, then Im(f), called image of f, is the set of all $y \in$ that have at least one preimage.

Examples

$f:R \rightarrow R$, $f(x) = x^2$

Im(f) = [0, inf)

Injections and Surjections

Definition

Given a function $f : X \rightarrow Y$, then:

f is a one-to-one function (or injection) if each element in Y is the image of at most one element in X (I.e. $x1 \neq x2 \Rightarrow f(x1) \neq f(x2)$). Example: $f(x) = x$

f is a onto function (or surjection) if Im(f) =Y, for each y in Y there exists x in X s.t. $f(x) = y$.

Example: $f(x) = x - 2$

Bijection and Inverse of a Function

Definition

f is a bijection if f is one-to-one and onto.

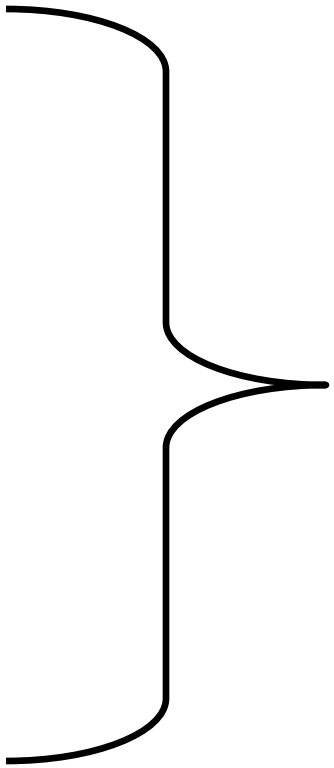
Definition

Given a function $f : X \rightarrow Y$, its inverse $f^{-1}(x)$ is defined by $f(f^{-1}(x)) = f^{-1}(f(x)) = x$

Theorem

The inverse function of a function f exists if and only if f is a bijection.

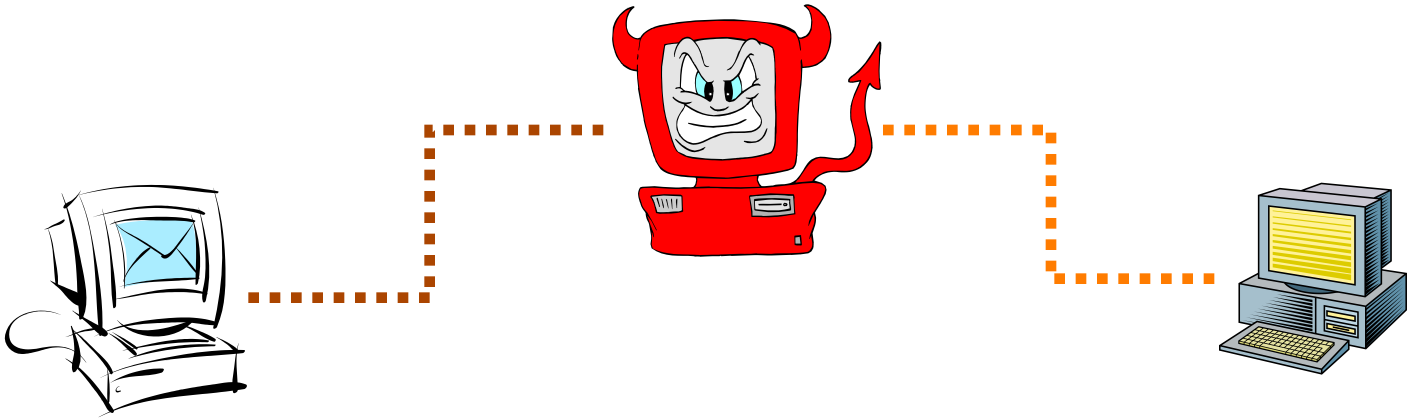
End Math



Security Services

- ✓ Confidentiality
- Integrity
- Authentication
- Non-repudiation
- Access control
- Availability

Data Integrity and Source Authentication



- Encryption does not protect data from modification by another party.
- 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

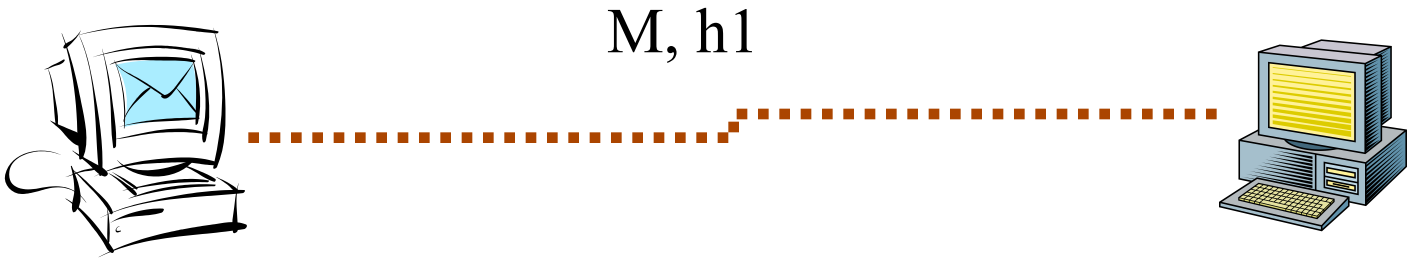
- Map a message of variable length n bits to a fingerprint of fixed length m bits, with m < n (output referred as **message digest**).
- A hash is a many-to-one function, so **collisions can happen**.
- Two fundamental properties: **compression and easy to compute**.

Uses of Hash Functions

- Data integrity
- Message authentication
- One-time passwords
- Digital signature
- Timestamping
- Certificate revocation management

Data Integrity with Hash Functions

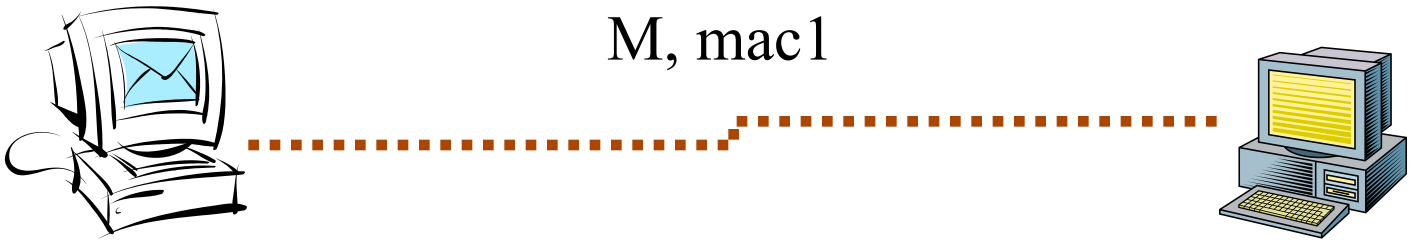
- Hash function is public:
  - Sender computes the value h1 = h(M) and sends it along with the message
  - Receiver computes h2 = h(M)
  - Checks if h1 = h2 ?
  - Yes accept the message, no reject the message



Source Authentication with MACs

- Hash function is public and the key shared between the sender and the receiver is secret
- The output of MAC can not be produced without knowing the secret key
  - Sender computes mac1 = MAC(M, H, K) and sends it along with the message M
  - Receiver computes mac2 = MAC(M, H, K) and checks if mac1 = mac2 ? Yes: accept the message, no: reject

    Because the mac1 could have been generated only by someone that knew the secret key K, this mechanism provides also data source authentication.



Classification

- MDC (manipulation detection codes) or MIC (message integrity codes)
  - Do not use a key
  - One-Way Hash Functions (OWHFs)
  - Collision Resistant Hash Functions (CRHFs)
- MAC (message authentication codes)
  - Use also a key
  - Used for both authentication and integrity

Requirements for Hash Functions

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

- preimage resistance (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 resistance (weak collision resistance):

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

- collision resistance (strong collision resistance):

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

Bruteforce Attacks on Hash Functions

Attacking one-wayness

- Goal: given $h:X \rightarrow Y$, $y \in Y$, find x such that $h(x)=y$
  - Algorithm: pick a random set X_0 of q values in X, for each $x \in X_0$, return x if $h(x)=y$, after all q values have been evaluated, return fail
  - m is the size of the output function
  - A Las Vegas randomized algorithm
  - When h is a random instance of all functions mapping X to Y, the average-case success probability is

$$\varepsilon = 1 - \left(1 - 1/|Y|\right)^q \approx \frac{q}{|Y|}$$

  - Let $|Y|=2^m$, to get ε to be close to 0.5, $q \approx 2^{m-1}$

Bruteforce Attacks on Hash Functions

Attacking collision resistance

- Goal: given h, find x, x' such that h(x)=h(x')
  - Algorithm: pick a random set X_0 of q values in X
    for each $x \in X_0$, computes $y_x=h(x)$
    if $y_x=y_{x'}$ for some $x' \neq x$ then return (x,x') else fail
  - The average success probability is $\varepsilon = 1 - e^{-\frac{q(q-1)}{2|Y|}}$
  - Let $|Y|=2^m$, to get ε to be close to 0.5, $\mathbf{q \approx 2^{m/2}}$
  - This is the birthday attack.

Las Vegas Randomized Algorithms

- A Las Vegas randomized algorithm may fail to give an answer, but when it does give an answer, the answer is always correct
- Such an algorithm has worst-case success probability ε if the algorithm returns a correct answer with probability at least ε
- Such an algorithm has average-case success probability ε if the probability that the algorithm returns a correct answer, averaged over all problem instances, is at least ε

Choosing the Length of Hash Outputs

- Because of the birthday attack, the length of hash outputs in general should double the key length of block ciphers
  - SHA-256, SHA-384, SHA-512 to match the new key lengths (128,192,256) in AES

Birthday Paradox

- Given a group of people, the minimum number of people such that two will share the same birthday with probability > 0.5 is only 23.
- 1 - (1- 1/365)(1-2/365)… (1-22/365) = 1- 0.493 = 0.507

General Problem

- Given a random variable that is an integer with uniform distribution between 1 and n and a selection of k instances, k<n of the random variable, what is the probability P(n, k) that there is at least one duplicate?

- Solution:

$$P(n,k) = 1 - \frac{n!}{(n-k)!n^k}$$

Solution (cont.)

$$P(n,k) = 1 - \frac{n!}{(n-k)!n^k} = 1 - [(1-\frac{1}{n}) \times (1-\frac{2}{n}) \times ... \times (1-\frac{k-1}{n})]$$

$$1 - x \le e^{-x} \quad \text{implies} \quad P(n,k) = 1 - \frac{n!}{(n-k)!n^k} > 1 - e^{-\frac{k \times (k-1)}{2n}}$$

$$P(n,k) > 0.5 \quad \text{implies} \quad \frac{1}{2} = 1 - e^{-\frac{k \times (k-1)}{2n}}$$

For large k, $(k-1)k \approx k^2$, we obtain

$$k = \sqrt{2 \ln 2n} = 1.18\sqrt{n} \approx \sqrt{n}$$

Why 23?

For the birthday problem, n = 365

$$k \approx \sqrt{n} \approx \sqrt{365} \approx 22.54 \approx 23$$

Going Back to Hash Functions…

Hash functions map n bits to m bits, m < n. Given h(x), if h is applied to k random inputs, what is the minimum k such that there exists x' such that h(x') = h(x) with probability > 0.5?

In other words: how many hash computations needs an attacker to try to break the 2-nd pre-image resistance condition?

Answer: $k = 2^{(m-1)}$

Moreover …

Given a hash function h that maps n bits to m bits, m < n. Apply h to k random inputs to produce the set X and apply again h to k additional random inputs to produce the set Y. What is the minimum k such that I found a match between the two sets with probability > 0.5?

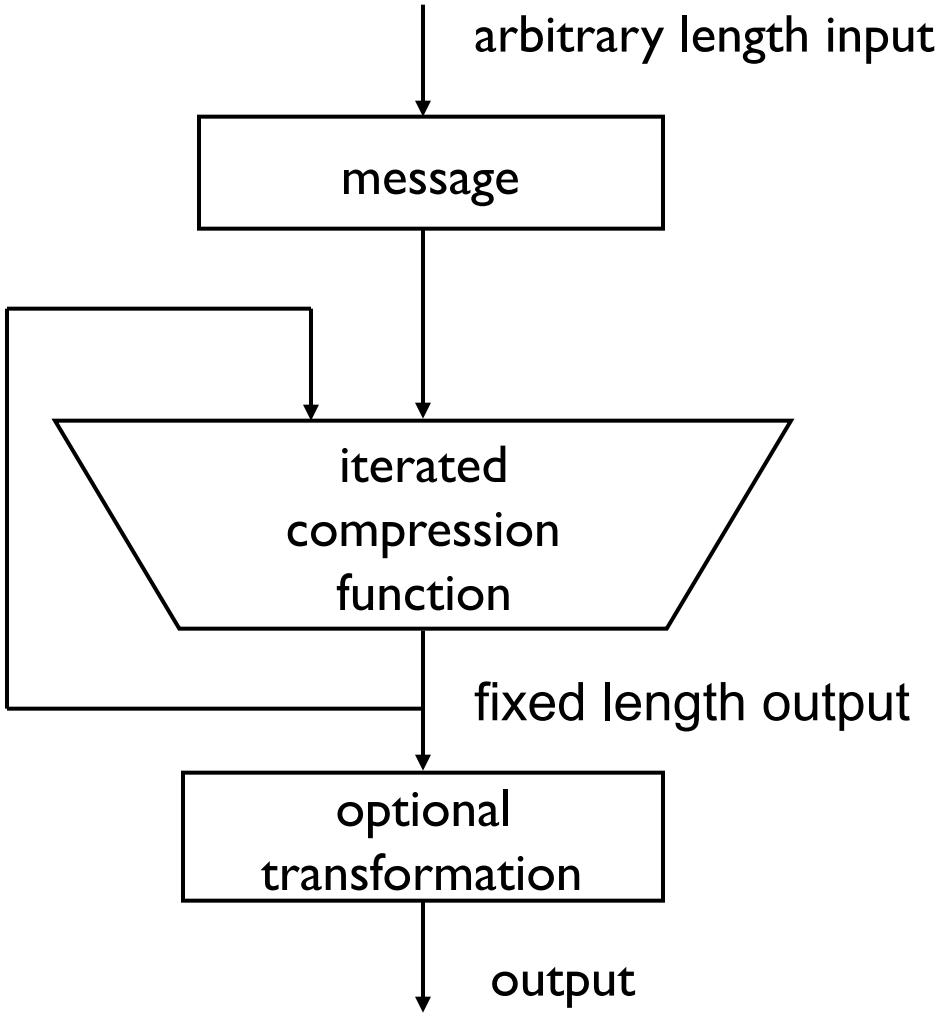
In other words: how much work does an attacker to do to find a collision (break the collision resistance property)?

Answer: $k = 2^{m/2}$

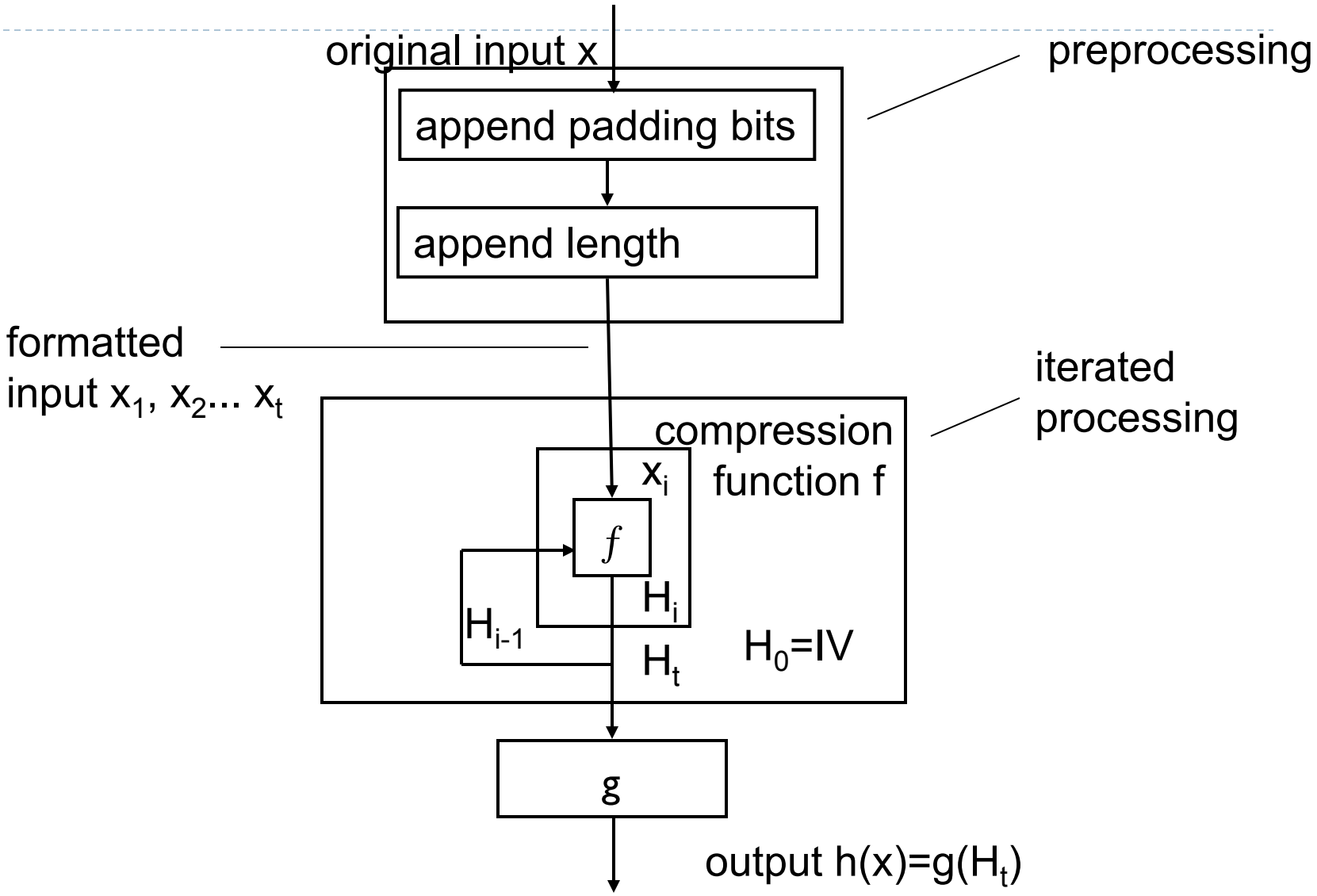
Birthday Attacks on Collision Resistance

- Attacks runs in $O(2^{m/2})$ and works against all the unkeyed hash function
- Steps:
  - Attacker generates $2^{m/2}$ variations of a valid message all with essentially the same meaning
  - Attacker also generates $2^{m/2}$ variations of a desired fraudulent message
  - Two sets of messages are compared to find pair with same hash (probability > 0.5 by birthday paradox)

Model for Iterated Hash Functions



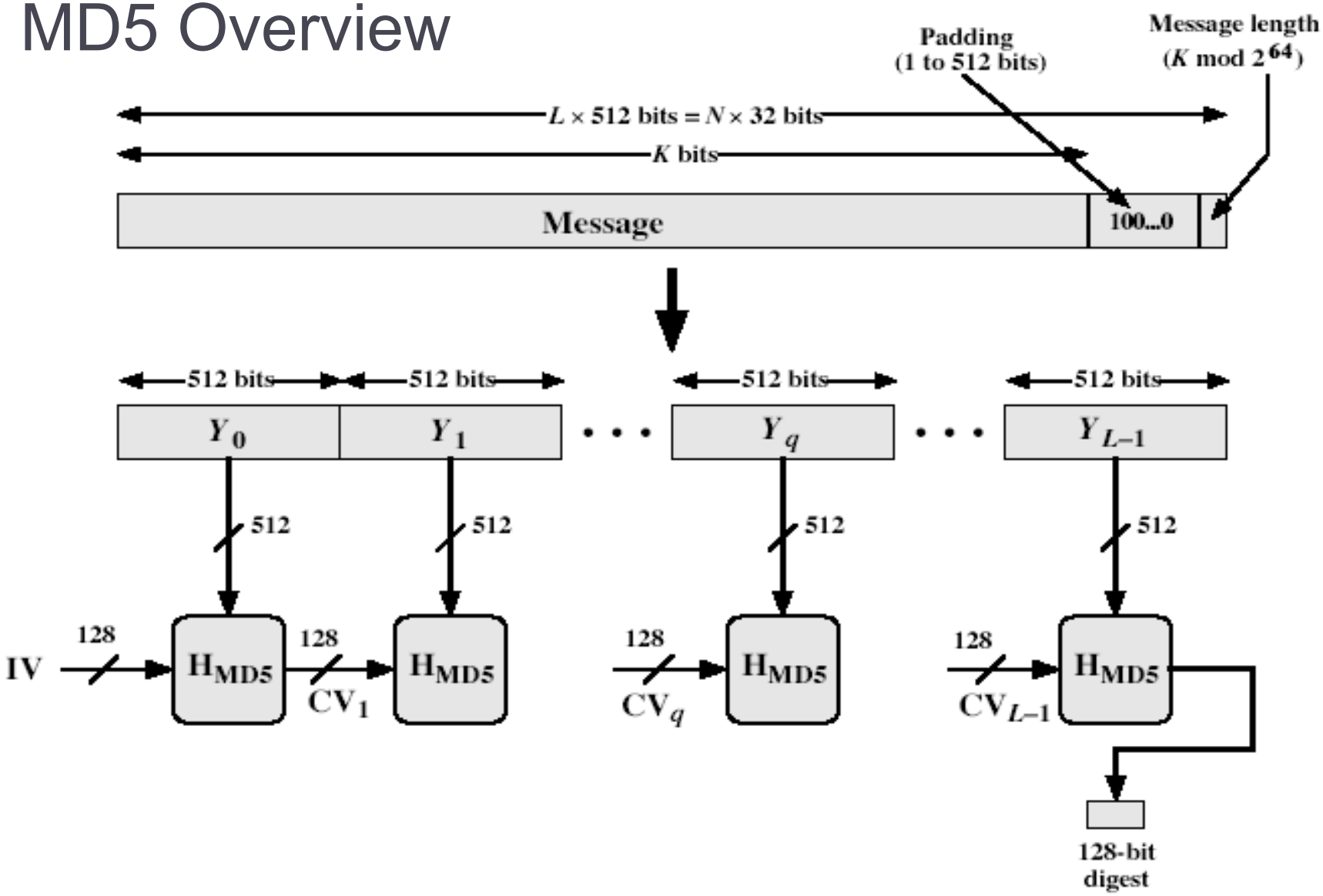
Details


original input x
preprocessing
append padding bits
append length
formatted
input x_1, x_2... x_t
iterated
processing
compression
function f
x_i
f
H_i
H_{i-1}
H_t
H_0=IV
g
output h(x)=g(H_t)

MD2, MD4 and MD5

- Family of one-way hash functions (Ron Rivest)
- MD2: produces a 128-bit hash value, perceived as slower and less secure than MD4 and MD5
- MD4: produces a 128-bit hash of the message, using bit operations on 32-bit operands for fast implementation, specified as Internet standard RFC1320
- MD5: produces a **128-bit output**, specified as Internet standard in RFC1321; till relatively recently was widely used.

MD5 Overview



MD5 Details

- The message is padded (1 followed by 0s) such that its L ≡ 448 mod 512
- Append a 64-bit (treated as unsigned int) representing the length of the message before padding (actually length mod $2^{64)}$
- Initialize the 4-word (128-bit) buffer (A,B,C,D)

  A = 01 23 45 67

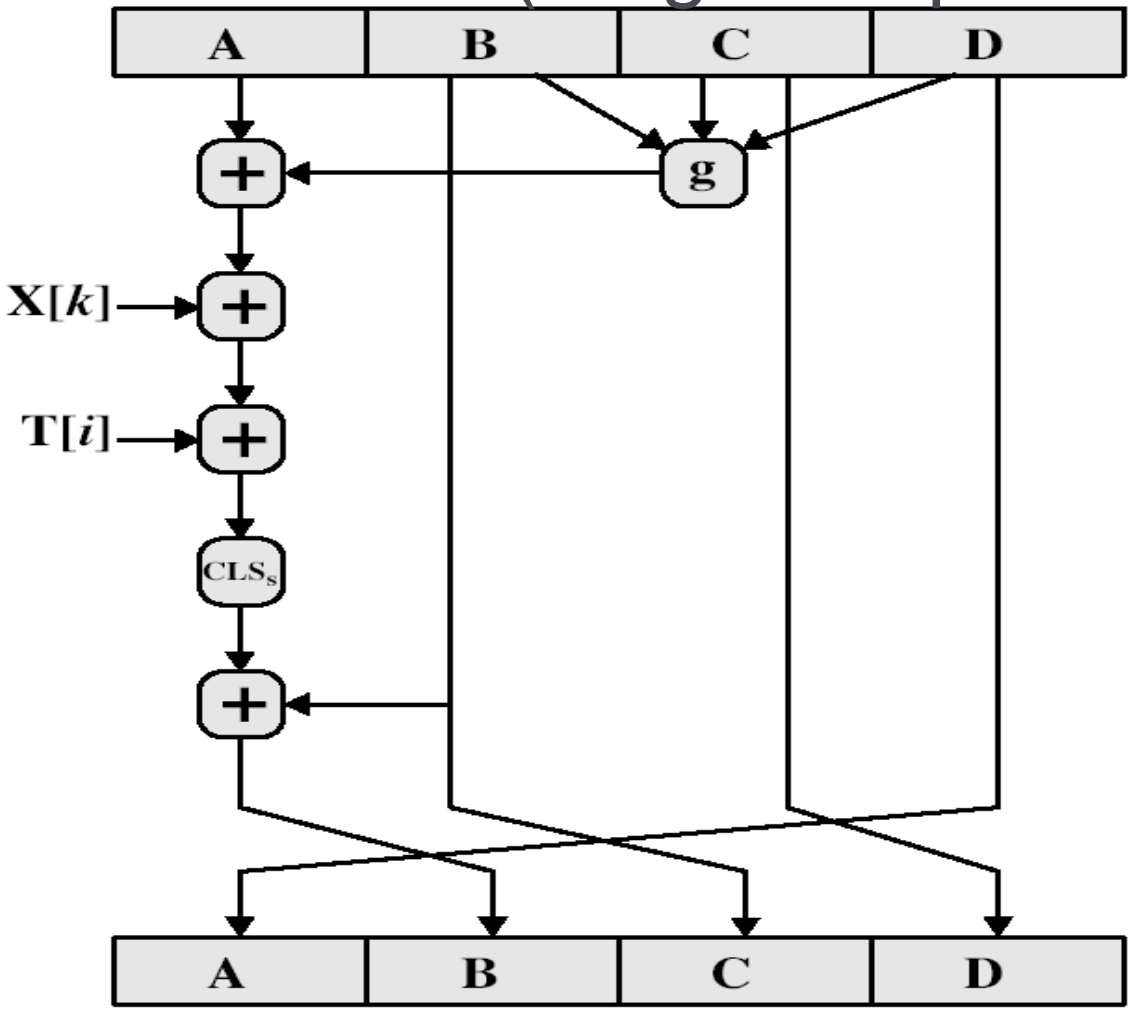
  B = 89 AB CD EF

  C = FE DC BA 98

  D = 76 54 32 10
- The message is processed in 16-word (512-bit) chunks, using **4 rounds of 16 bit operations each**

MD5 Compression Function (Single Step

- Output: 128 bits
- Using 4 rounds
- Each round: 16 steps



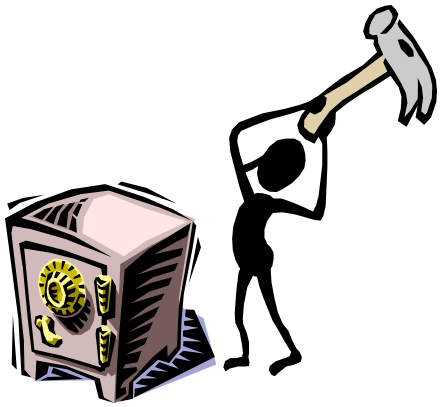
MD5 Compression Function

- Each round has 16 steps of the form:

  `a = b+((a+g(b,c,d)+X[k]+T[i])<<<s)`
- a,b,c,d are the 4 words of the buffer, but used in varying permutations
- g(b,c,d) is a different nonlinear function in each round (F,G,H,I); Example: round 1 g(b,c,d)=(b∧c)∨(neg(b) ∧d)
- T[i] is a constant value derived from sin function
- X[k] derived from a 512-block of the message

MD5 Cryptanalysis

- Known attacks:
  - Berson (1992): for a single-round MD5, he used differential cryptanalysis to find two messages producing the same hash. Attack does not work for 4-round MD5.
  - Boer & Bosselaers(1993): found a pseudo collision (use a particular IV), unable to extend to full MD5
  - Dobbertin (1996) created collisions on MD5 compression function only on the first block, but initial constants prevent exploit on a bigger message
  - Wang, Feng, Lai, Yu (2004) found collisions of MD5
    - works on any IV
    - easy to find multiple collisions
- Brute force biggest concern:
  - $2^{m/2}$, since m =128 , 2^{64}



SHA1 (Secure Hash Algorithm)

- SHA was designed by NIST and is the US federal standard for hash functions, specified in FIPS-180 (1993).
- SHA-1, revised version of SHA, specified in FIPS-180-1 (1995).
- It produces 160-bit hash values.
- NIST have issued a revision FIPS 180-2 that adds 3 additional hash algorithms: SHA-256, SHA-384, SHA-512, designed for compatibility with increased security provided by AES.

SHA1 Overview

- As in MD5 message is padded such as its length is a multiple of 512 bits
- Initialize a 5-word (160-bit) buffer

  Word A: 67 45 23 01

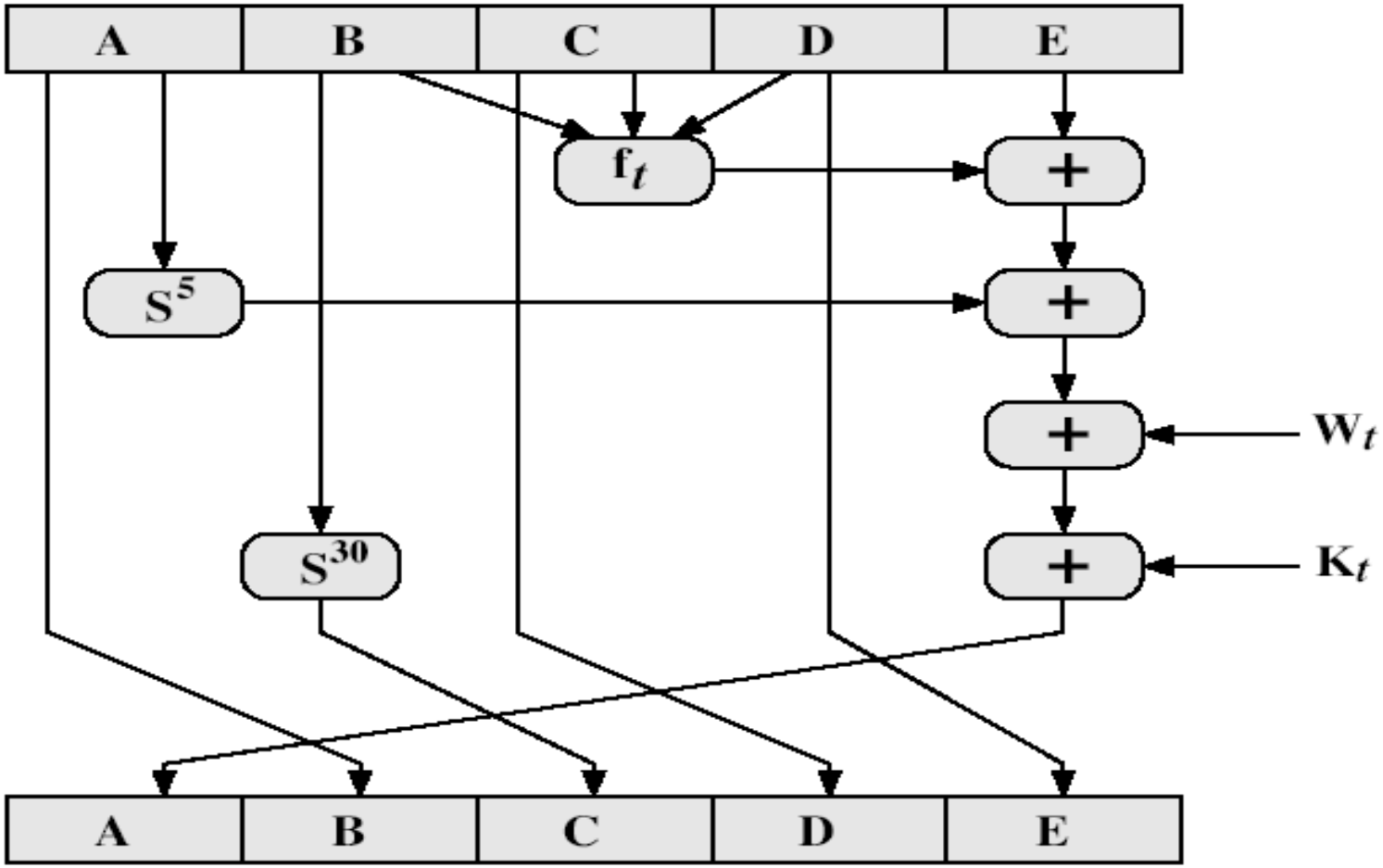
  Word B: EF CD AB 89

  Word C: 98 BA DC FE

  Word D: 10 32 54 76

  Word E: C3 D2 E1 F0
- Message is processed in 16-word (512-bit) chunks:
  - expand 16 words into 80 words by mixing & shifting
  - use **4 rounds of 20 operations** on message block and buffer

SHA-1 Compression Function (Single Step)



SHA-1 Compression Function

- Each round consists of 20 steps, updates the buffer as follows:

  (A,B,C,D,E) <- (E+f(t,B,C,D)+(A<<5)+W_t+K_t),A,
  (B<<30),C,D)
- t is the step number
- f(t,B,C,D) is a non-linear function for round
- W_t is derived from the message block
- K_t is a constant value derived from the sin function
- S^k is circular left shift by k bits

SHA-1 Cryptanalysis

- SHA1 uses 20 steps instead of 16 (MD5) in each round.
- SHA1 shuffles and mixes them using rotates & XOR's to form a more complex input that makes finding collisions more difficult.
- Brute force attack is harder (160 vs 128 bits for MD5)
- Wang, Yin, and Yu (2005) found ways to find collisions using no more than 2^{69} hash evaluations
- Wang, Yao and Yao (2005) found collisions using no more than $\mathbf{2^{63}}$ hash evaluations
- NIST made a request for the design of a new hash function

NIST Request

- ``A process to develop and standardize one or more new hash algorithms to augment and revise FIPS 180-2, Secure Hash Standard, is being initiated by the National Institute of Standards and Technology (NIST). As a first step in this process, NIST is publishing draft minimum acceptability requirements, submission requirements, and evaluation criteria for candidate algorithms to solicit public comment. It is intended that the revised hash function standard will specify one or more additional unclassified, publicly disclosed hash algorithms that are available royalty-free worldwide, and are capable of protecting sensitive government information well into the foreseeable future.''
- Entries for the competition must be received by ***October 31, 2008***.

NIST Hash Competition

- **Oct. 31, 2008**: Received sixty-four entries from cryptographers around the world
- **Dec. 2008**: Selected fifty-one first-round candidates
- **July 2009**: Selected fourteen second-round candidates
- **December 2010**: Announced five third-round candidates – BLAKE, Grøstl, JH, Keccak and Skein, to enter the final round of the competition.
- **October 2, 2012**: Based on the public comments and internal review of the candidates, **NIST announced Keccak as the winner of the SHA-3 Cryptographic Hash Algorithm Competition on October 2, 2012, and ended the five-year competition.**

Keccak

- Designed by a team of cryptographers from Belgium and Italy:
- Guido Bertoni (Italy) of STMicroelectronics,
- Joan Daemen (Belgium) of STMicroelectronics,
- Michaël Peeters (Belgium) of NXP Semiconductors, and
- Gilles Van Assche (Belgium) of STMicroelectronics.

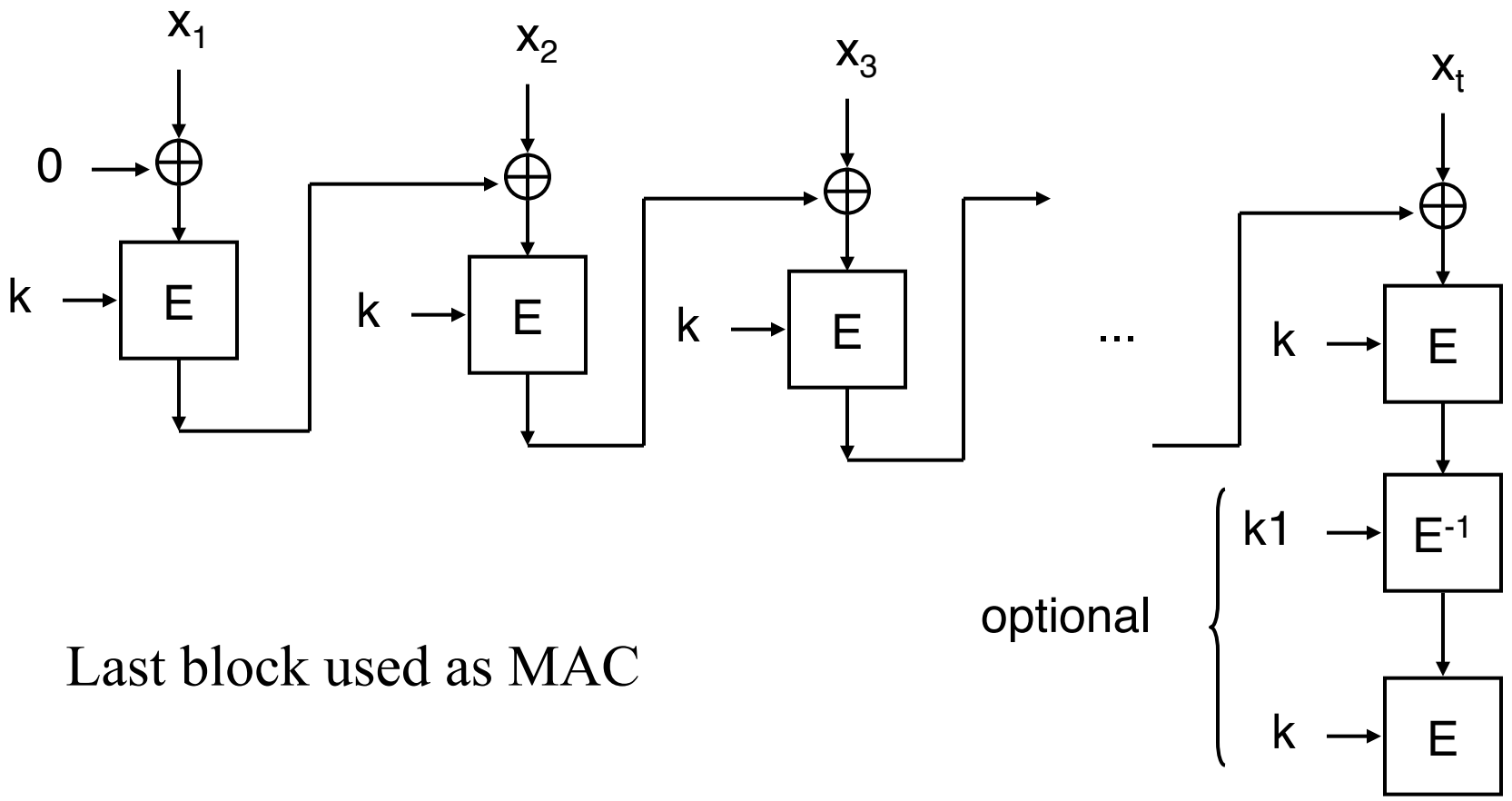
Why Hash is Not Enough?

- Hash functions can provide data integrity, but no indication about where is data coming from or who generated the hash output (hash function is public)
- Data source authentication (also referred as message authentication) is needed, otherwise anybody can inject traffic
- Mechanism? Involve a secret key
- NOTE: MACs do not prevent all traffic injections (for example do not prevent replay attacks)

Requirements for MAC

- Involve a secret key
- Easy of computation if secret key k is known.
- MAC is a many-to-one function so collisions are possible (different messages have same MAC)
- Similar to hash functions requirements:
  - Compression: M has n bits, $h_k(M)$ has fixed length m, m < n
  - Knowing a message and MAC, is infeasible to find another message with same MAC
  - MACs should be uniformly distributed
  - MAC should depend equally on all bits of the message

MAC Based on Symmetric Encryption: CBC-MAC



Last block used as MAC

Security of CBC-MAC

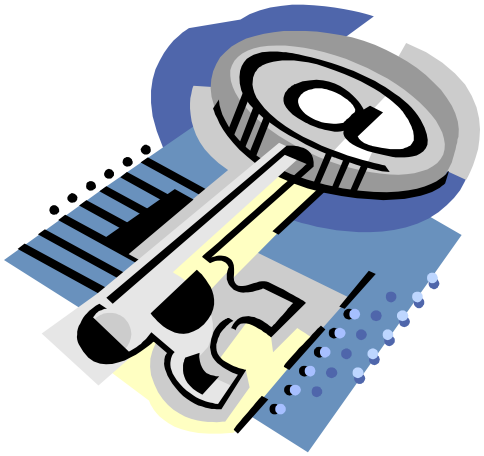
- The basic CBC-MAC is secure only for messages of a fixed number of blocks.
- Attack against CBC-MAC: Having (x_1,H_1) and (x_2,H_2) and requesting $((x_1 \,||z),H_3)$ it's possible to construct a new message s.t. $(x_2 \,||(H_1 \oplus z \oplus H_2),H_3)$ is valid, where x1, x2, and z have the length equal to the block size of the block cipher used to generate the MAC.
- Attack prevention: the optional step prevents forgery without impacting intermediate stages.

Attacks Exploiting the Cipher

- These attacks apply against MACs that use symmetric ciphers
- Exploit properties of the cipher to find out collisions
- Examples:
  - Complementation property
  - Weak keys
  - Key collisions

Keyed Hash Functions as MACs

- Create a MAC using a hash function rather than a block cipher because hash functions are generally faster.
- Uses a public hash function and a secret symmetric key
- Current standard is HMAC, specified in FIPS 198 (2002)



HMAC Goals

- Use available hash functions without modification.
- Preserve the original performance of the hash function without incurring a significant degradation.
- Use and handle keys in a simple way.
- Allow easy replacement of the underlying hash function in the event that faster or more secure hash functions are later available.
- Have a well-understood cryptographic analysis of the strength of the authentication mechanism based on reasonable assumptions on the underlying hash function.

Why not only one Hash?

- Construction with key at the beginning:

  hmac = hash (K || M)

  Problem: allows for length extension attacks

- Construction with key at the end:

  hmac = hash (M || K)

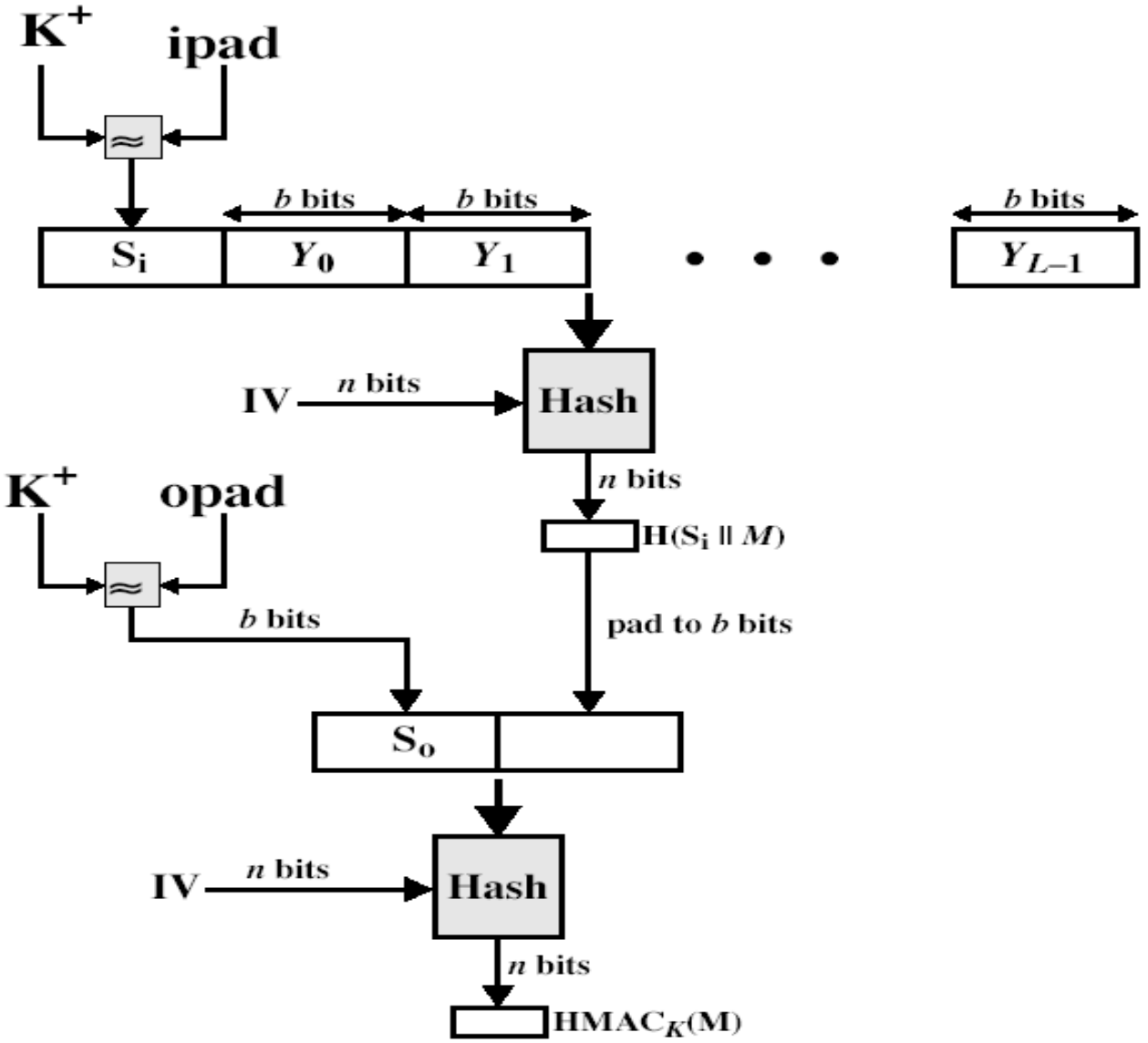
  Problem: allows for key-recovery attacks

HMAC

$$HMAC_K = Hash[(K^+ \oplus opad) \,||\, Hash[(K^+ \oplus ipad)||M)]]$$

- K^+ is the key padded out to input block size of the hash function and opad, ipad are specified padding constants
- Key size: $L/2 < K < L$
- MAC size: at least L/2, where L is the hash output

HMAC Overview



HMAC Security

- Security of HMAC relates to that of the underlying hash algorithm
- If used with a secure hash function and according to the specification (key size, and use correct output), not known practical attacks against HMAC
- In general, HMAC be attacked as follows:
  - brute force on the key space
  - attacks on the hash function itself
    - birthday attack, although the use of key makes this attack more difficult
    - attacks against the compression function

A Word about Data Integrity in a Non-Malicious Environment

- Goal: protect against accidental or non-malicious errors on noisy channels subject to transmission errors
- Error detection codes and error correction codes
- NOTE: with these methods, anybody can forge packets, the requirement is different
- Methods:
  - Checksum
  - CRC (Cyclic redundancy codes)

WEP Details

- RC4 is a stream cipher: based on key k and initialization vector (IV) v, generates a keystream RC4(v,k)
- To send a message M from A to B
  - Compute integrity checksum (CRC32): c(M)
  - plaintext P = {M, c(M)}
  - Encrypt P using RC4: ciphertext C = P $\oplus$ RC4(v,k)
  - Transmit **C' = v, (P $\oplus$ RC4(v,k))**
- To decipher an encrypted message C', the encryption process is reversed

Some Observations

- The integrity check does not depend on a key, but just on the message M, so anybody can create a pair M and CRC32(M)
- The WEP standard specifies 64-bit key = **40** bit key and 24 IV. Some vendors implemented 128-bit keys (24 IV and **104** bit key).
- The IV is sent in clear, so is available to the attacker as well.

Risk of Keystream Reuse

$$C1 = P1 \oplus RC4(v, k)$$
$$C2 = P2 \oplus RC4(v, k)$$
$$C1 \oplus C2 = P1 \oplus P2$$

- If P1 or P2 is also known by the attacker, the other plaintext is easy to compute
- If n ciphertexts using the same keystream are available makes reading traffic easier (frequency analysis, etc)
- *Find plaintext P and the encryption C with keystream k, then it is easy to decipher any ciphertext C' encrypted with the same keystream k.*

Is Keystream Reused?

- The pseudorandom keystream is based on the shared key k and the initialization vector IV. Since the key k is secret and is difficult to be changed for every packet, changing the IV is important to prevent keystream reuse.
- The IV is sent in clear, so is available to the attacker as well.
- The WEP standard recommends, but does not require that the IV be changed every packet, also does not say anything about how to select the IV.
- An implementation can reuse the same IV for all packets without risking non-compliance !

24-bit IV Space

- Busy access point sending 1500 byte packets, at an average of 2 Mbps, exhausts the IV space in **half a day.**
- Random generation of IV can produce collisions every **5000** packets (due to the *birthday paradox)*.
- Many implementations use for IV a counter that is incremented for each packet sent and reset every time the card is inserted in the computer.

Exploiting Keystream Reuse

- Methods to obtain pairs (plaintext, ciphertext):
  - IP fields predictable: login sequences, recognize shared libraries transfer
  - Send email and wait for the user to check it via wireless links
  - Send data to access-points that have access control disables and observe the encrypted data

Dictionary Attack

- Goal: Decrypt traffic
- How: Store keystream in a table, indexed by IV.
- Remember the IV is sent it clear
- When the attacker sees a packet with an IV stored already in the table, look up the corresponding keystream, XOR it against the packet, and read the data!
- Table is at most 1500 * 2^24 bytes = 24 GB

Packet Modification

- CRC32 is linear: $c(M \oplus D) = c(M) \oplus c(D)$
- Message M was transmitted, and the ciphertext was C and the IV was IV, C and IV are known to the adversary.
- Attacker can find C' s. t. it decrypts to M',M' = M ⊕ D
  D = arbitrarily chosen by the attacker
- $C' = C \oplus \langle D, c(D) \rangle$
  $= RC4(v,k) \oplus \langle M, c(M) \rangle \oplus \langle D, c(D) \rangle$
  $= RC4(v,k) \oplus \langle M \oplus D, c(M) \oplus c(D) \rangle$
  $= RC4(v,k) \oplus \langle M', c(M \oplus D) \rangle$
  $= RC4(v,k) \oplus \langle M', c(M') \rangle$