

# Cryptography CS 555



## Lecture 9: Cryptographic Hash Functions and HMAC

Department of Computer Sciences  
Purdue University

# Lecture Outline

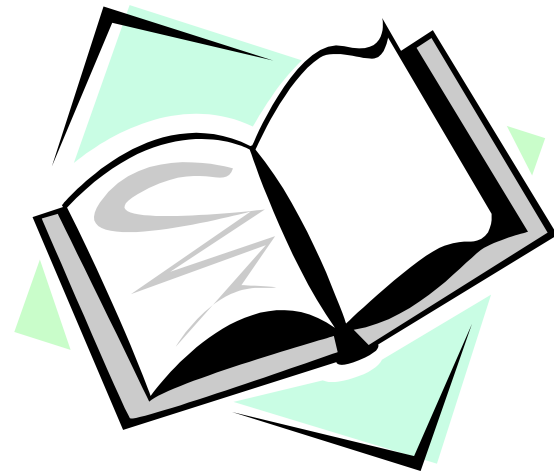
- SHA1, RIPEMD160
- HMAC
- Attacks on hash functions and HMAC



# Recommended Reading

Stallings, Chapter 12: 12.1, 12.2  
and 12.3

Stinson, Chapter 4: 4.1, 4.2 and  
4.3

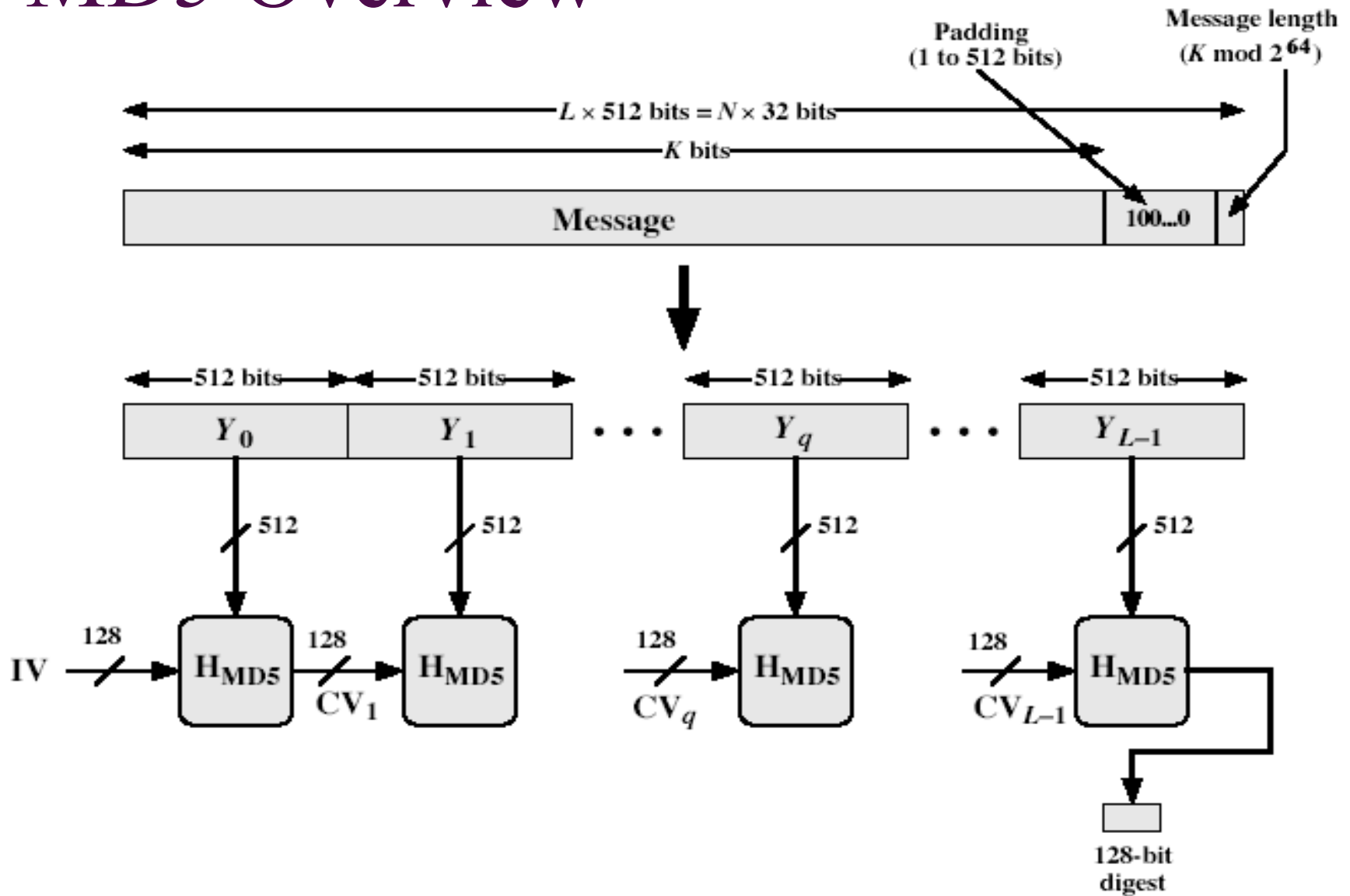


# Requirements for Hash Functions

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

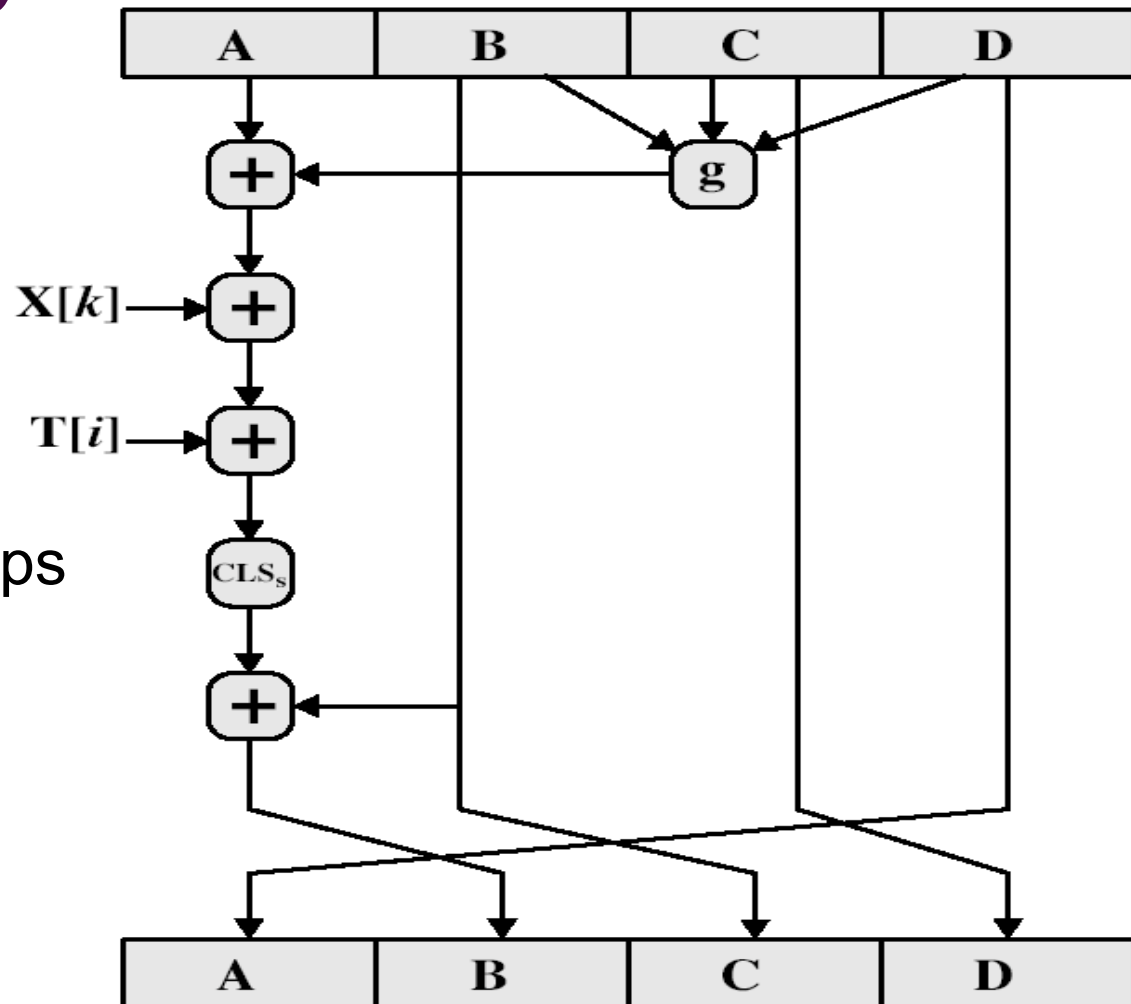
- **preimage resistance (one-way):**  
if given  $y \in Y$  it is computationally infeasible to find a value  $x \in X$  s.t.  $h(x) = y$
- **2-nd preimage resistance (weak collision resistance):**  
if given  $x \in X$  it is computationally infeasible to find a value  $x' \in X$ ,  $x' \neq x$  s.t.  $h(x') = h(x)$
- **collision resistance (strong collision resistance):**  
if it is computationally infeasible to find any two distinct values  $x', x \in X$ , s.t.  $h(x') = h(x)$

# MD5 Overview



# MD5 Compression Function (Single Step)

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



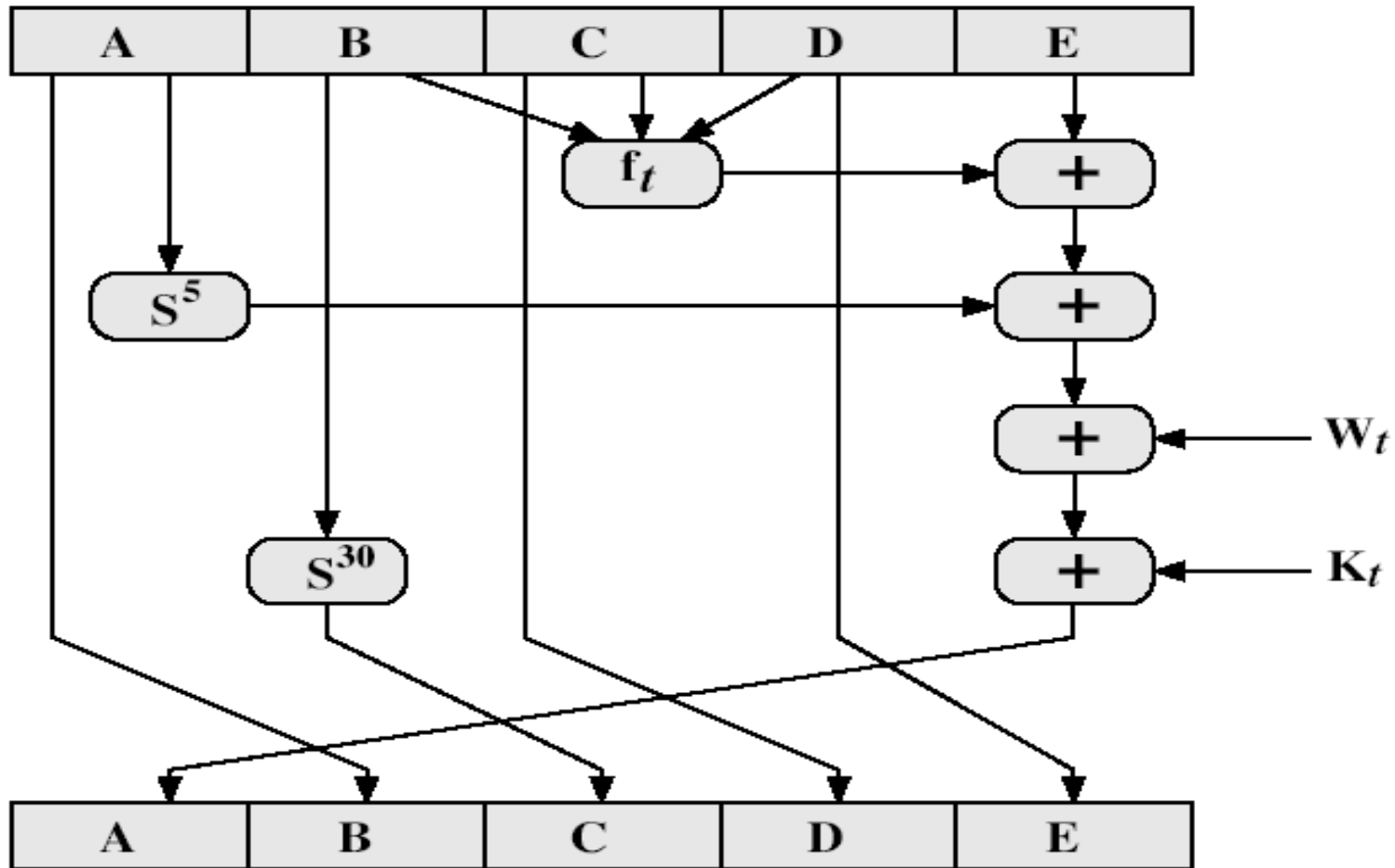
# SHA1 (Secure Hash Algorithm)

- 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:  
 $(A, B, C, D, E) \leftarrow (E + f(t, B, C, D) + (A \ll 5) + W_t + K_t), A, (B \ll 30), C, D)$
- $t$  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

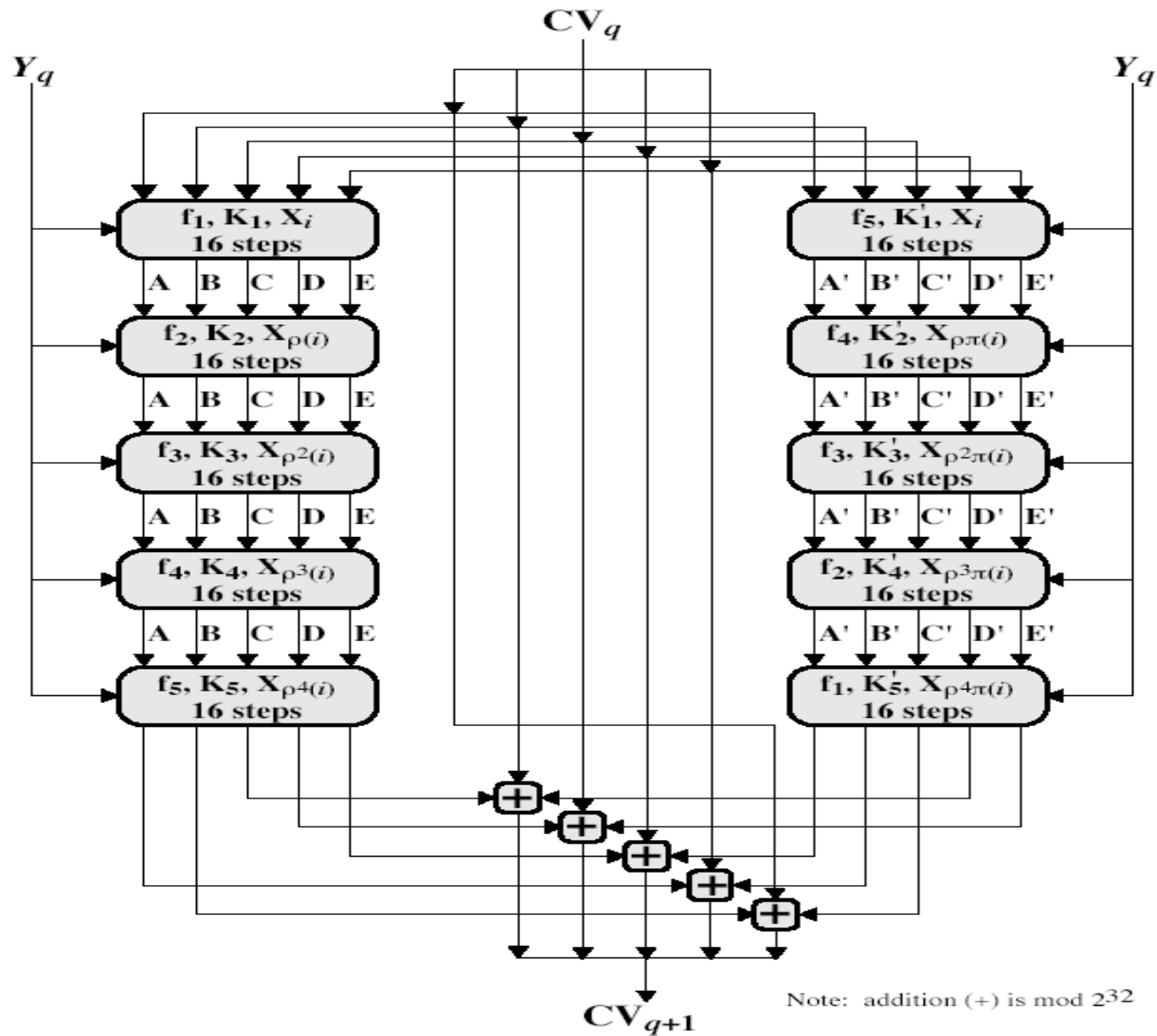
- SHA1 uses 20 instead of 16 steps (MD5) in each round.
- SHA1 shuffles and mixes them using rotates & XOR's to form a more complex input that makes finding collisions more difficult.
- Brute force attack is harder (160 vs 128 bits for MD5)
- Not vulnerable to any known attacks (as opposed to MD5)



# RIPEMD-160

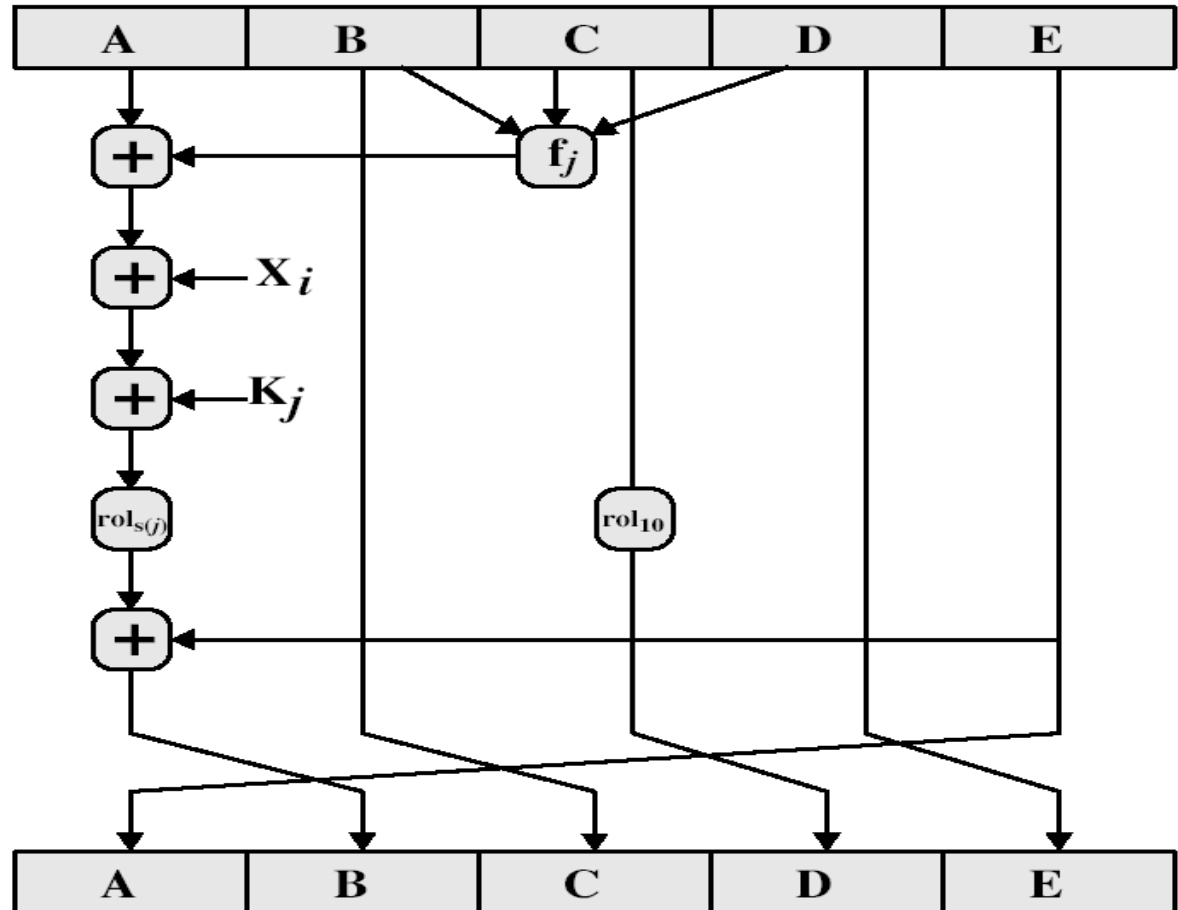
- RIPEMD-160 was developed in Europe by researchers involved in attacks on MD4/5
- Output is 160 bits
- As MD5, message is padded such that  $\text{length} \equiv 448 \pmod{512}$ , and 64-bit length value is appended
- Initialise 5-word (160-bit) buffer (A,B,C,D,E) as follows:
  - A: 67 45 23 01
  - B: EF CD AB 89
  - C: 98 BA DC FE
  - D: 10 32 54 76
  - E: C3 D2 E1 F0
- Process message in 16-word (512-bit) chunks
- Uses 10 rounds of 16 steps on message block and buffer, in **2 parallel lines of 5**

# RIPEMD-160 Round



# RIPEMD-160 Compression Function (Single Step)

$K_j$  constant used  
 in step  $j$   
 Rot denotes  
 circular left shift  
 $X_i$  derived from a  
 512 message block



# RIPEMD-160, MD5 & SHA-1

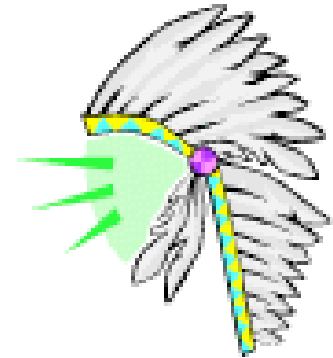
- Brute force attack similar to SHA1, (160 bits)
- Not vulnerable to known attacks
- Slower than MD5 (more steps)
- SHA-1 optimised for big endian CPU's while RIPEMD-160 & MD5 are optimised for little endian CPU's

# { Endianness }

- Defines byte ordering, i.e. which end of a multi-byte integer appears in the lowest address byte of the integer.
- **Big endian** means the **most significant byte is stored first**. Sun and network format are big endian
- **Little endian** means the **least significant byte is stored first**. Intel is little endian

Example: 0x87654321

Address	Big Endian	Little Endian
Buf	0x87	0x21
Buf + 1	0x65	0x43
Buf + 2	0x43	0x65
Buf + 3	0x21	0x87



# 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
  - both authentication and integrity

# Why Hash is Not Enough?

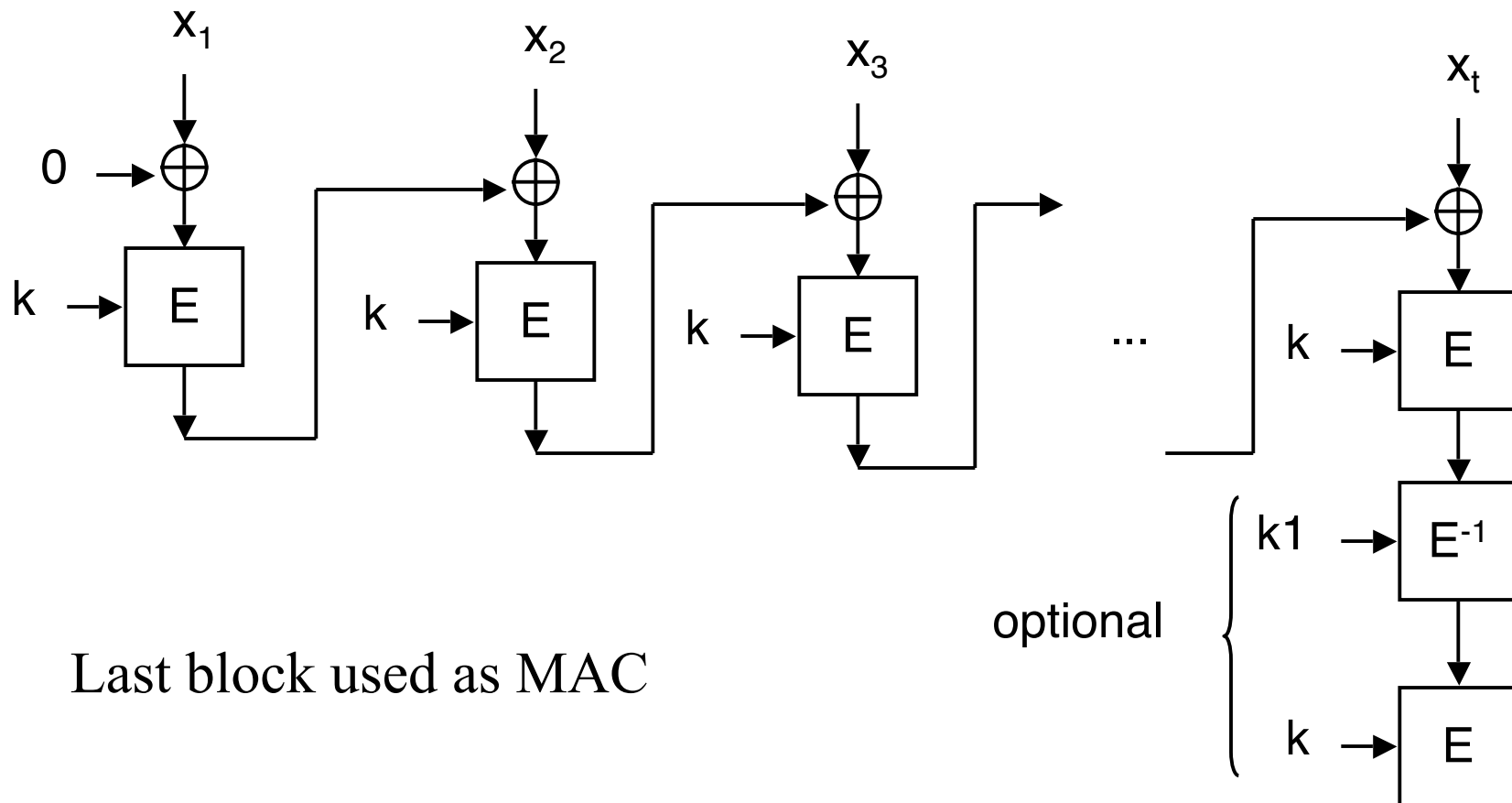
- Hash functions can provide data integrity, but no indication about where is data coming from or who generated the hash output (hash function is public)
- Data source authentication (also referred as message authentication) is needed, otherwise anybody can inject traffic
- Mechanism? Involve a secret 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

optional

# 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  $x_1$ ,  $x_2$ , 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.

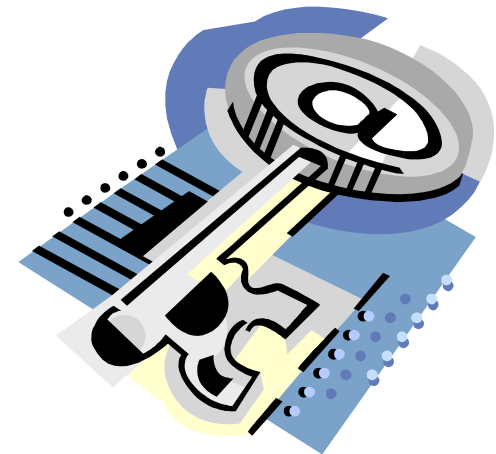
# Attacks Exploiting the Cipher

- These attacks apply against MACs that use symmetric ciphers
- Exploit properties of the cipher to find out collisions
- Examples:
  - Complementation property
  - Weak keys
  - Fixed points (limited practical impact  $(E_k(x))=x$  for all  $x$ )
  - 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

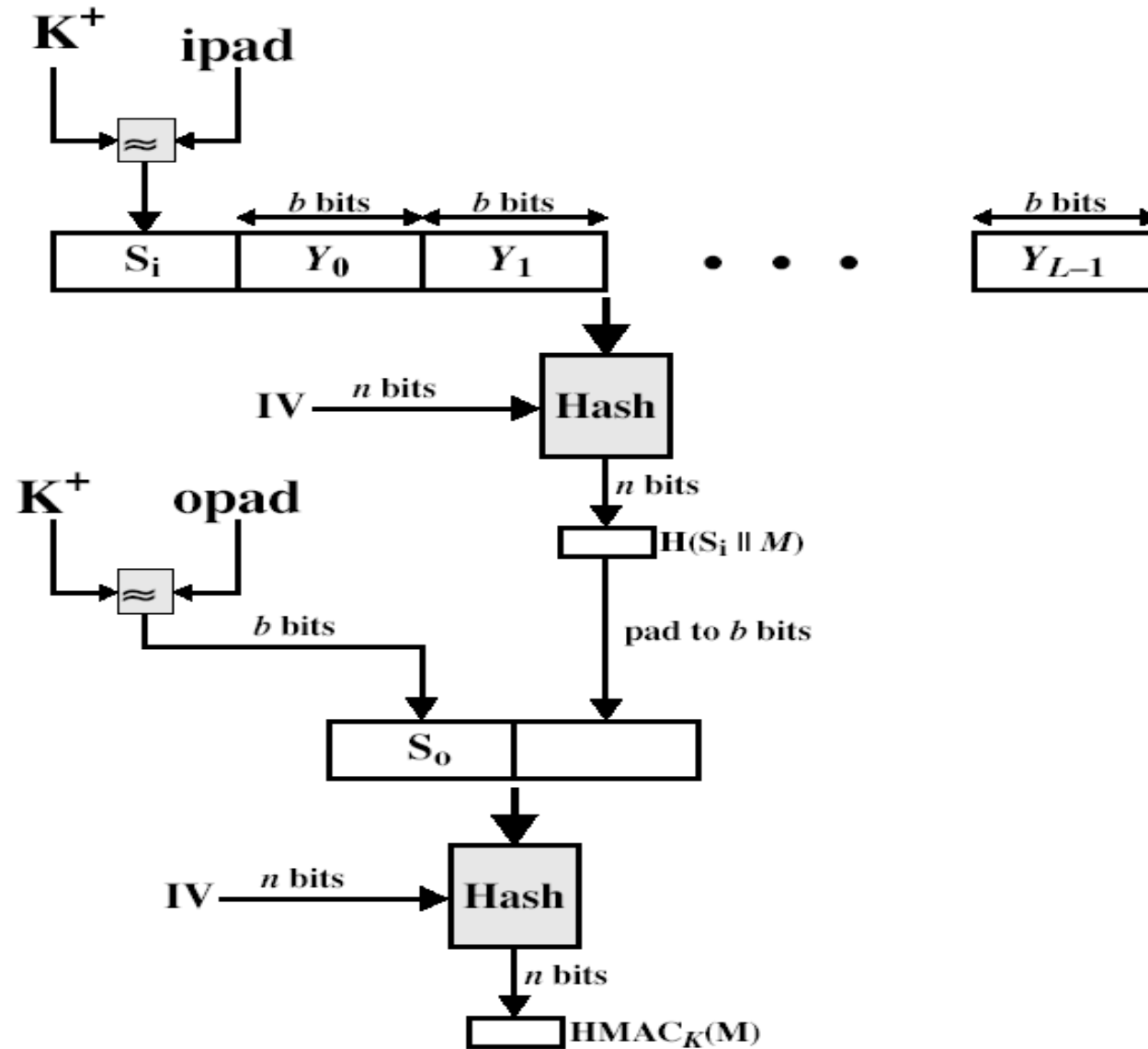
- Use available hash functions without modification.
- Preserve the original performance of the hash function without incurring a significant degradation.
- Use and handle keys in a simple way.
- Allow easy replacement of the underlying hash function in the event that faster or more secure hash functions 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.

# HMAC

$$\text{HMAC}_K = \text{Hash}[(K^+ \oplus \text{opad}) \parallel \text{Hash}[(K^+ \oplus \text{ipad}) \parallel M]]$$

- $K^+$  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 (s.t. SHA1) 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)



# Summary of Attacks on Hash and HMAC

- Birthday attacks
- Pseudo-collisions
- Attacks against compression function
- Chaining attacks
- Attacks based on properties of the underlying cipher

# Birthday Attacks

- 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 a pair with same hash (probability  $> 0.5$  by birthday paradox)

# Pseudo-collisions

- Find collisions while allowing different IVs for the different message inputs
- Provide upper bounds for security, used to analyze security of hash functions, sometimes can be extended to full attacks.
- Not a practical concern, use specified IV
- Example of trivial collision for random IVs
  - $H(IV, x_1x_2) = f(f(IV, x_1), x_2)$
  - $IV' = f(IV, x_1)$   $h(IV', x_2) = h(IV, x_1x_2)$
  - We obtained a pseudo-collision, independent of the strength of function  $f$

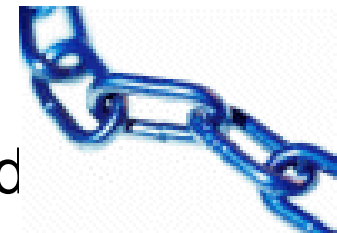


# Attacks on Compression Function $f$

- Setup:
  - focus is on a fixed step  $I$  of the iterative hash function,
  - the entire message consists of one block
  - hash output is the compression function output
- Why bother?
  - Because sometimes this attack can be extended to similar attack on the iterated hash function (same complexity), although sometimes an iterated hash function can be weaker than its compression function

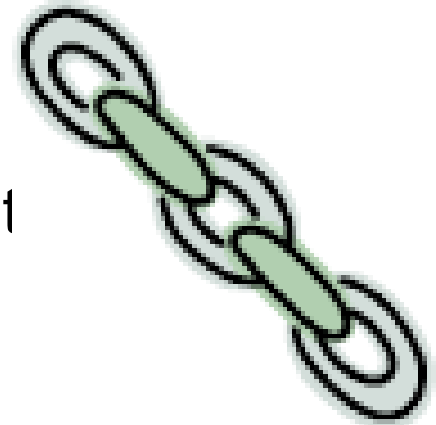
# Chaining Attacks

- Based on the iterative nature of the hash function
- **Correcting-block chaining attacks:** produce a collision by using a designed block that produces a chaining value that determines the output to be a specific  $h(x)$
- **Meet-in-the-middle chaining attacks:** similar to birthday attack, but seek collision on intermediate result instead the final output



# Chaining Attacks (cont.)

- **Fixed point:**
  - a fixed point is a pair  $(H_{i-1}, x_i)$  s.t.  
 $f(H_{i-1}, x_i) = H_{i-1}$
  - insertion of arbitrary number of identical block  $x_i$  at this chain point (the overall hash output remains unchanged)
- **Differential chaining attacks:**  
differential cryptanalysis applied to hash functions



# 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
  - Fixed points (limited practical impact ( $E_k(x))=x$  for all  $x$ )
  - Key collisions



# Summary

- Current hash standard SHA1 (160 bits)
- HMAC is the MAC standard
- Birthday attacks has implications on the length of the output of hash functions (brute force)



# Next lecture...

- Number theory
- Recommended reading
  - Stallings: number theory chapter
  - Wagstaff

