COM 5336 Cryptography
Lecture 10
Hash, MAC, HMAC

Scott CH Huang

Contents

- Hash functions
- Message authentication (three methods)
  - message encryption
  - message authentication code (MAC)
  - hash function
- Case studies
  - MD5, SHA-1, RIPEMD-160
- HMAC

Compare Information w/o Leaking

- Alice tells her boss that an annoying guy, but she wants to keep his identity confidential.
- Bob also tell his boss that a guy is annoying him a lot.
- Alice and Bob would like to determine whether they are complaining about the same guy, but they don’t want to reveal his identity. What should they do?

Hash Functions

- A Hash Function is a function h s.t.
  - Compression: it condenses a variable-length message M to a fixed-sized fingerprint
  - Ease of computation: $h(x)$ is easy to compute for any given x
- usually hash function is public and not require a secret key
  - However, MAC is keyed

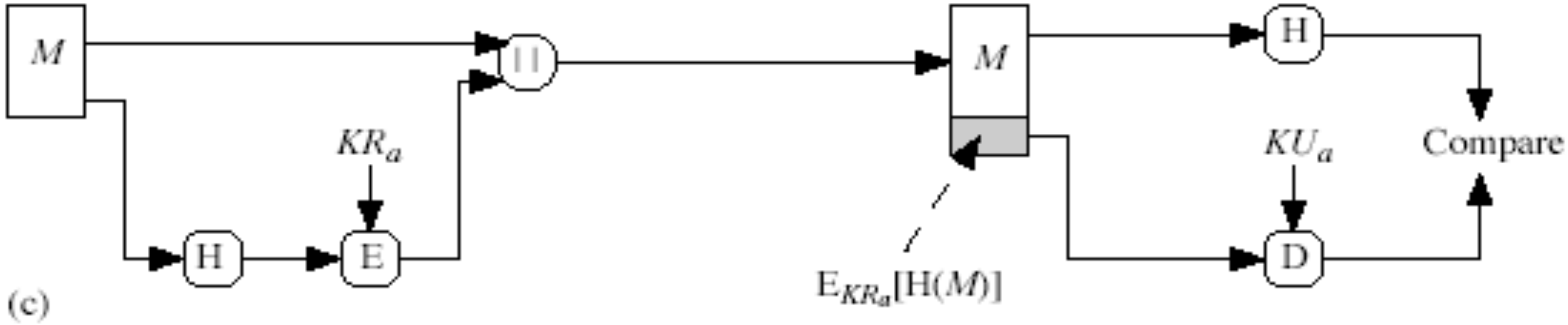
Basic Properties

- One-way-ness
  - Given any hashed value y, it’s computationally infeasible to find an x s.t h(x)=y
- Weak collision resistance
  - Given any x, it’s computationally infeasible to find x’ s.t. h(x)=h(x’)
- Strong collision resistance
  - Computationally infeasible to find x,x’ s.t. h(x)=h(x’)

Relation Between Properties

- Strong collision resistance => weak collision resistance
- Strong collision resistance =>? One-way
  - No!
  - Let g be a collision resistant hash function, g:{0,1}*→{0,1}^{n}, define h as follows
  - h(x)=1||x, if x is n-bit long
  - =0||g(x), otherwise

Hash Functions & Digital Signatures



Use of Hash Functions

- Hash can be used to detect changes to message
  - can use in various ways with message
  - most often to create a digital signature

Birthday Attacks

- A.k.a *Birthday Paradox*
- Suppose there are 13 ppl in the class. The probability that at least one student has the same birthday as the teacher is

$$1 - \left(1 - \frac{1}{365}\right)^{13} \doteq 3.5\%$$

- However, the probability that at least two students have the same birthday $\geq 19.2\%$

Birthday Attacks

Suppose there are n students in the class and there are H=365 days in a year.

$$\begin{aligned}
&\Pr[\geq 2 \text{ students have same bday}] \\
&= 1 - \Pr[\text{everybody has different bdays}] \\
&= 1 - \left(1 - \frac{1}{H}\right)\left(1 - \frac{2}{H}\right)\cdots\left(1 - \frac{n-1}{H}\right) \\
&\geq 1 - \left(1 - \frac{1}{H}\right)\left(1 - \frac{1}{H}\right)^2\cdots\left(1 - \frac{1}{H}\right)^{n-1} \\
&= 1 - \left(1 - \frac{1}{H}\right)^{\frac{n(n-1)}{2}} = 1 - \left(1 - \frac{1}{H}\right)^{H\frac{n(n-1)}{2H}} \\
&\approx 1 - e^{-\frac{n(n-1)}{2H}} \approx 1 - e^{-n^2/2H}
\end{aligned}$$

Birthday Attacks

Conversely, let p= collision probability. What's the smallest number n to make collision happen w/ probability p?

$$\begin{aligned} \frac{n^2}{2H} &\approx \ln\frac{1}{1-p} \\ n &\approx \sqrt{2H\ln\frac{1}{1-p}} = O\left(\sqrt{H}\right) \end{aligned}$$

If we wish to make collision happen w/ a fixed string, then

$$\begin{aligned} 1-p &= \left(1-\frac{1}{H}\right)^n \approx e^{-\frac{n}{H}} \\ n &\approx H\ln\frac{1}{1-p} = O(H) \end{aligned}$$

Birthday Attacks

- 64-bit hash is not secure
  - because of Birthday Attack (a.k.a. Birthday Paradox)
- Birthday attack vs hash function:
  - opponent generates $2^{m/2}$ variations of a valid message all with essentially the same meaning
  - opponent 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)
  - have user sign the valid message, then substitute the forgery which will have a valid signature
- Conclusion: we need to use larger MACs

Block Ciphers as Hash Functions

- Can use block ciphers as hash functions
  - using H_0=0 and zero-pad of final block
  - compute: $H_i = E_{M_i} [H_{i-1}]$
  - and use final block as the hash value
  - similar to CBC but without a key
- Resulting hash is too small (64-bit)
  - both due to direct birthday attack
  - and to "meet-in-the-middle" attack
- Other variants also susceptible to attack

Hash Algorithms

- Similarities in the evolution of hash functions & block ciphers
  - increasing power of brute-force attacks
  - leading to evolution in algorithms
  - from DES to AES in block ciphers
  - from MD4 & MD5 to SHA-1 & RIPEMD-160 in hash algorithms
    - New attacks vs SHA-1 and RIPEMD-160 appeared recently.
- Likewise tend to use common iterative structure as do block ciphers

Message Authentication

- Data integrity of a message
  - Make sure what is sent is what is received.
- Validating identity of originator
  - The claimed sender is the actual sender
- Verifying sequencing and timeliness
- Counter repudiation of the source
  - Sender cannot deny having sent a message.

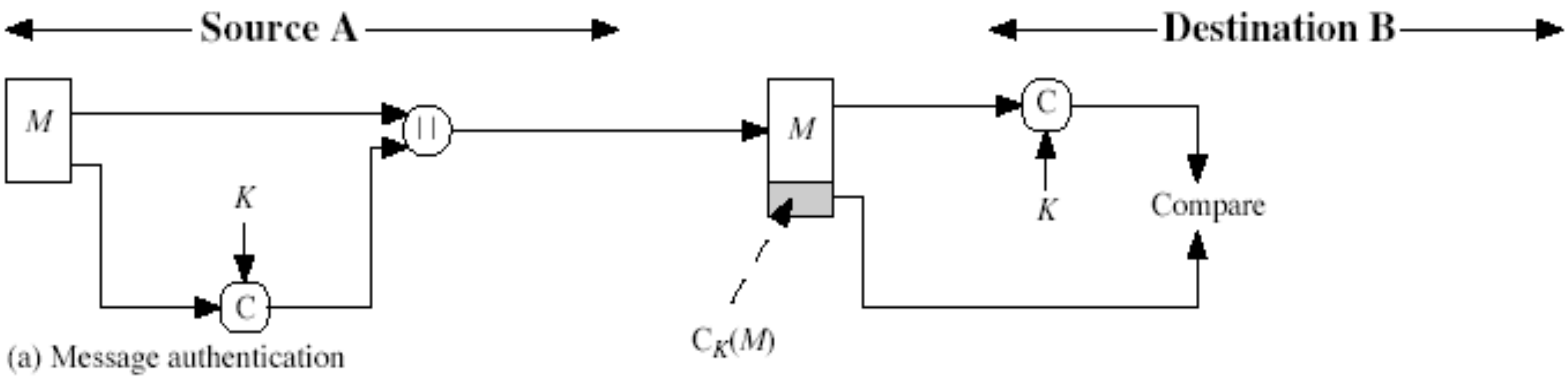
Authentication via Message Encryption

- Message encryption by itself also provides a measure of authentication
- If symmetric encryption is used then:
    - Validation of the sender:
        - receiver know sender must have created it
            - since only sender and receiver know the key used
    - Confidentiality:
        - know content cannot have been altered

Message Authentication Code

- Generated by an algorithm that creates a small fixed-sized block
  - depending on both message and some key
  - Similar to encryption though need not be reversible
- Appended to message as a **signature**
- Receiver performs same computation on message and checks it matches the MAC
- It provides assurance (to the receiver) that message is unaltered and comes from sender

MAC



(a) Message authentication

Advantage of MAC

- It provides validation of the sender
- Why use a MAC, not encryption?
  - sometimes only validation of the sender is needed
  - sometimes need validation of the sender to persist longer than the encryption (e.g., for archival use)
- Alternatively, one can also use encryption for confidentiality
  - generally use separate keys for each
  - can compute MAC either before or after encryption
  - is generally regarded as better done before
- Note that an MAC does not provide non-repudiation
  - Receiver could forge message
  - Sender could deny message

MAC Properties

- An MAC is a cryptographic checksum

  MAC = $C_K(M)$

  - It condenses a variable-length message M,
    - using a secret key K
  - to a fixed-sized authenticator
- It is a many-to-one function
  - potentially many messages have same MAC
  - but finding these are very difficult

Requirements for MACs

1. Knowing a message and its MAC, is infeasible to find another message with the same MAC
2. MACs should be uniformly distributed
3. MACs should depend equally on all bits of the message

Using Symmetric Ciphers for MACs

- Can use any block cipher chaining mode and use final block as a MAC
- **Data Authentication Algorithm (DAA)** is a widely used MAC based on DES-CBC
  - using IV=0 and zero-pad of final block
  - encrypt message using DES in CBC mode
  - and send just the final block as the MAC
    - or the leftmost M bits (16≤M≤64) of final block
- But final MAC is now too small for security

Drawbacks

- Non-repudiation cannot hold here:
  - The receiver can send a message to itself using the secrete key and claim it was originated from the sender.
  - No judge can decide who sent the message originally since both have the secrete key.

Authentication via Public Key Encryption

- Encryption using secrete key,
- Decryption with public key
  - Validation of the sender
  - Non-repudiation

Authentication via Message Encryption

- Sender **signs** message using private-key then encrypts with recipient’s public key
  - Confidentiality
  - Validation of sender
  - Non-repudiation of both sender and receiver
    - Other than the receiver cannot read message
  - but at cost of two public-key uses on message

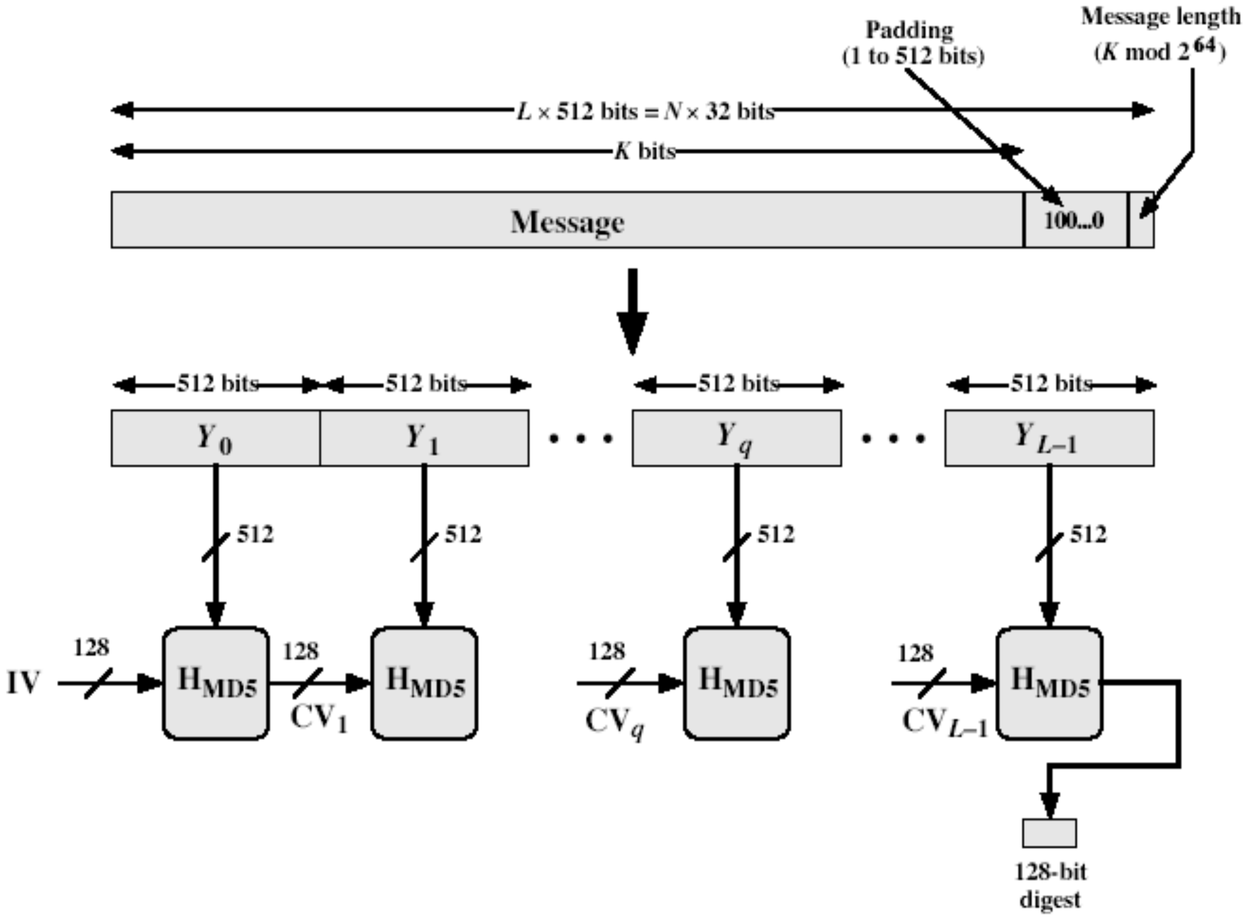
MD5

- Designed by Ronald Rivest (the R in RSA)
- Latest in a series of MD2, MD4
- Produces a 128-bit hash value
- Until recently was the most widely used hash algorithm
  - in recent times have both brute-force & cryptanalytic concerns
- Specified as Internet standard RFC1321

MD5 Overview

1. Pad message so its length is 448 mod 512
2. Append a 64-bit length value to message
3. Initialise 4-word (128-bit) MD buffer (A,B,C,D)
4. Process message in 16-word (512-bit) blocks:
   - using 4 rounds of 16 bit operations on message block & buffer
   - add output to buffer input to form new buffer value
5. Output hash value is the final buffer value

MD5 Overview



MD5 Compression Function

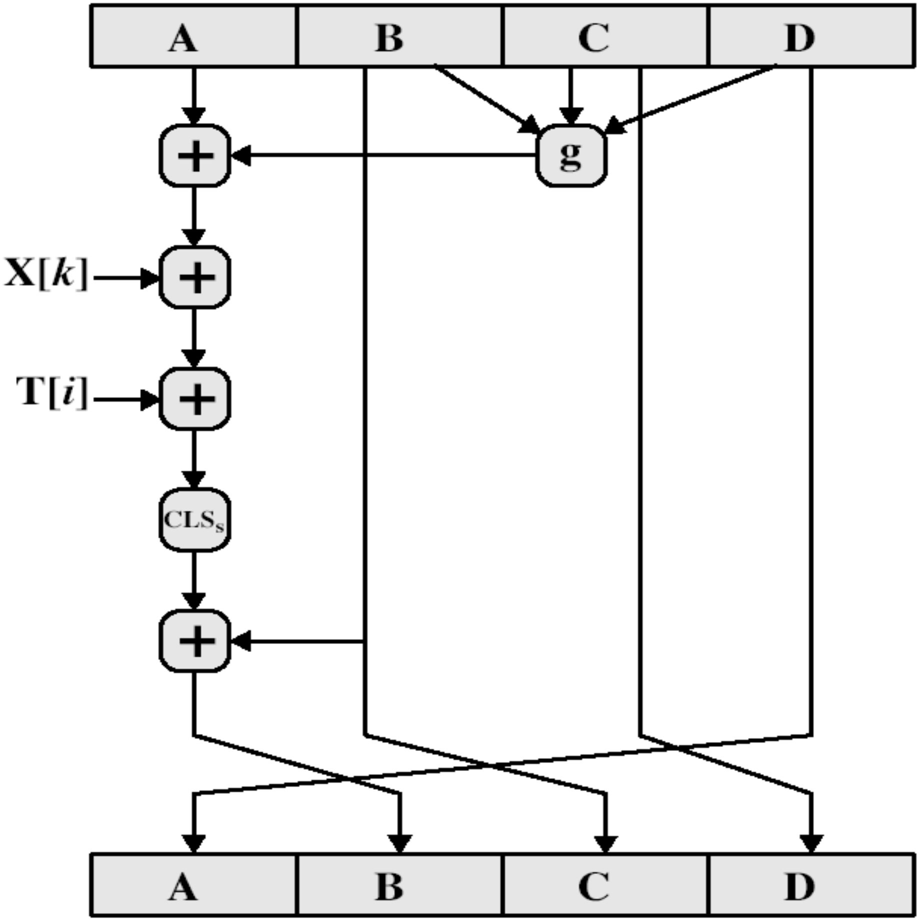
- Each round has 16 steps of the form:

  $a = b+((a+g(b,c,d)+X[k]+T[i]) \lll s)$
- a,b,c,d refer to the 4 words of the buffer, but used in varying permutations
  - note this updates 1 word only of the buffer
  - after 16 steps each word is updated 4 times
- where g(b,c,d) is a different nonlinear function in each round (F,G,H,I)
- T[i] is a constant value derived from sin

The g function

- g function is either one of the following F,G,H,I functions, according to their rounds.
- F(b,c,d)=(b & c) || (~b & d)
- G(b,c,d)=(b & d) || (c & ~d)
- H(b,c,d)=b XOR c XOR d
- I(b,c,d)=c XOR (b || ~d)

MD5 Compression Function



Strength of MD5

- MD5 hash is dependent on all message bits
- Rivest claims security is good as can be
- Known attacks are:
  - Berson 92 attacked any 1 round using differential cryptanalysis (but can't extend)
  - Boer & Bosselaers 93 found a pseudo collision (again unable to extend)
  - Dobbertin 96 created collisions on MD compression function (but initial constants prevent exploit)
- Xiaoyun Wang (Tsinghua Univ) cracked MD5 completely in 2004!

Secure Hash Algorithm (SHA-1)

- SHA was designed by NIST & NSA in 1993, revised 1995 as SHA-1
- US standard for use with DSA signature scheme
  - standard is FIPS 180-1 1995, also Internet RFC3174
  - nb. the algorithm is SHA, the standard is SHS
- Produces 160-bit hash values
- Now the generally preferred hash algorithm
- Based on design of MD4 with key differences

SHA Overview

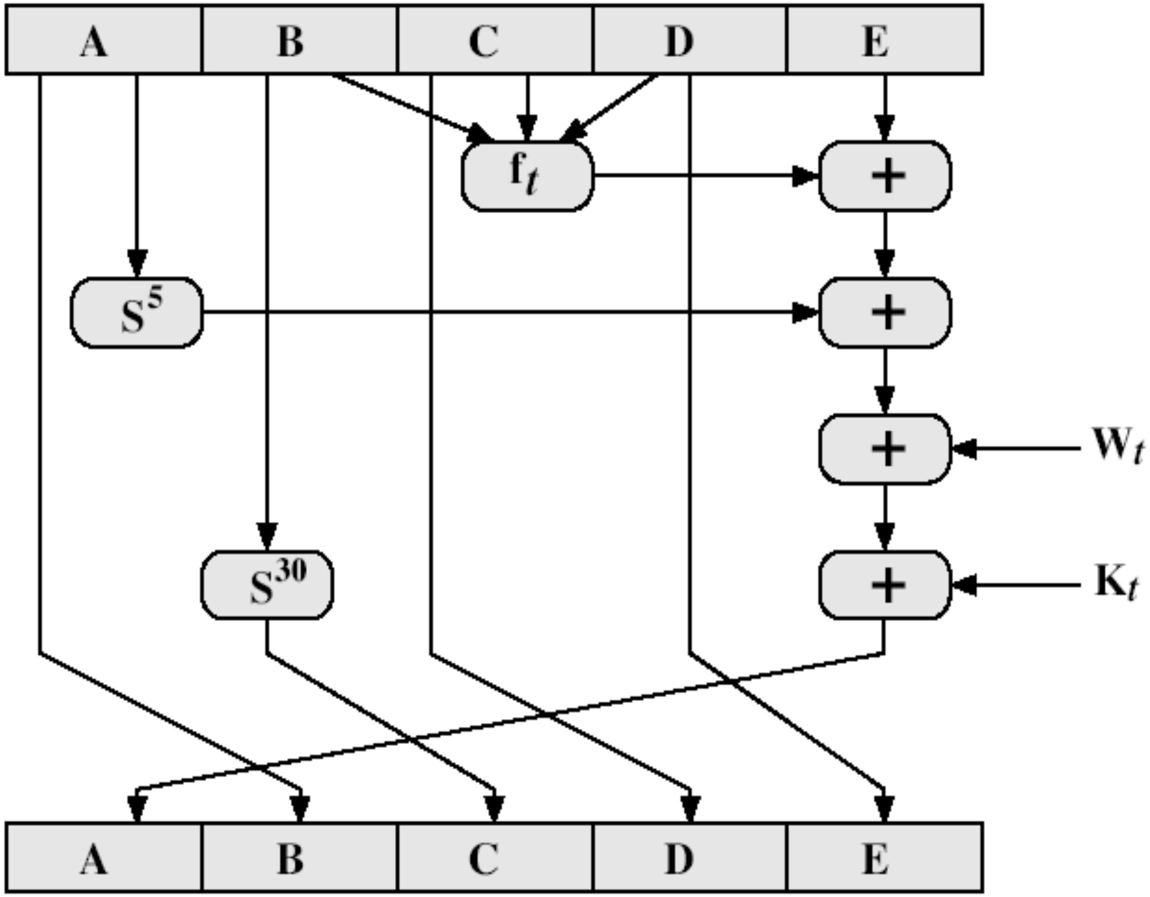
1. Pad message so its length is 448 mod 512
2. Append a 64-bit length value to message
3. Initialise 5-word (160-bit) buffer (A,B,C,D,E) to (67452301,efcdab89,98badcfe,10325476,c3d2e1f0)
4. Process message in 16-word (512-bit) chunks:
   - expand 16 words into 80 words by mixing & shifting
   - use 4 rounds of 20 bit operations on message block & buffer
   - add output to input to form new buffer value
5. Output hash value is the final buffer value

SHA-1 Compression Function

- Each round has 20 steps which replaces the 5 buffer words thus:

  $(A,B,C,D,E) <-(E+f(t,B,C,D)+(A<<5)+W_t+K_t),A,(B<<30),C,D)$
- a,b,c,d refer to the 4 words of the buffer
- t is the step number
- f(t,B,C,D) is nonlinear function for round
- W_t is derived from the message block
- K_t is a constant value derived from sin

SHA-1 Compression Function



SHA-1 verses MD5

- Brute force attack is harder (160 vs 128 bits for MD5)
- A little slower than MD5 (80 vs 64 steps)
- Both designed as simple and compact
- Optimised for big endian CPU's (vs MD5 which is optimised for little endian CPU’s)
- Also cracked completely by Xiaoyun Wang in 2005!

Revised Secure Hash Standard

- NIST have issued a revision FIPS 180-2
- Adds 3 additional hash algorithms
- SHA-256, SHA-384, SHA-512
- Designed for compatibility with increased security provided by the AES cipher
- Structure & detail is similar to SHA-1
- Hence analysis should be similar

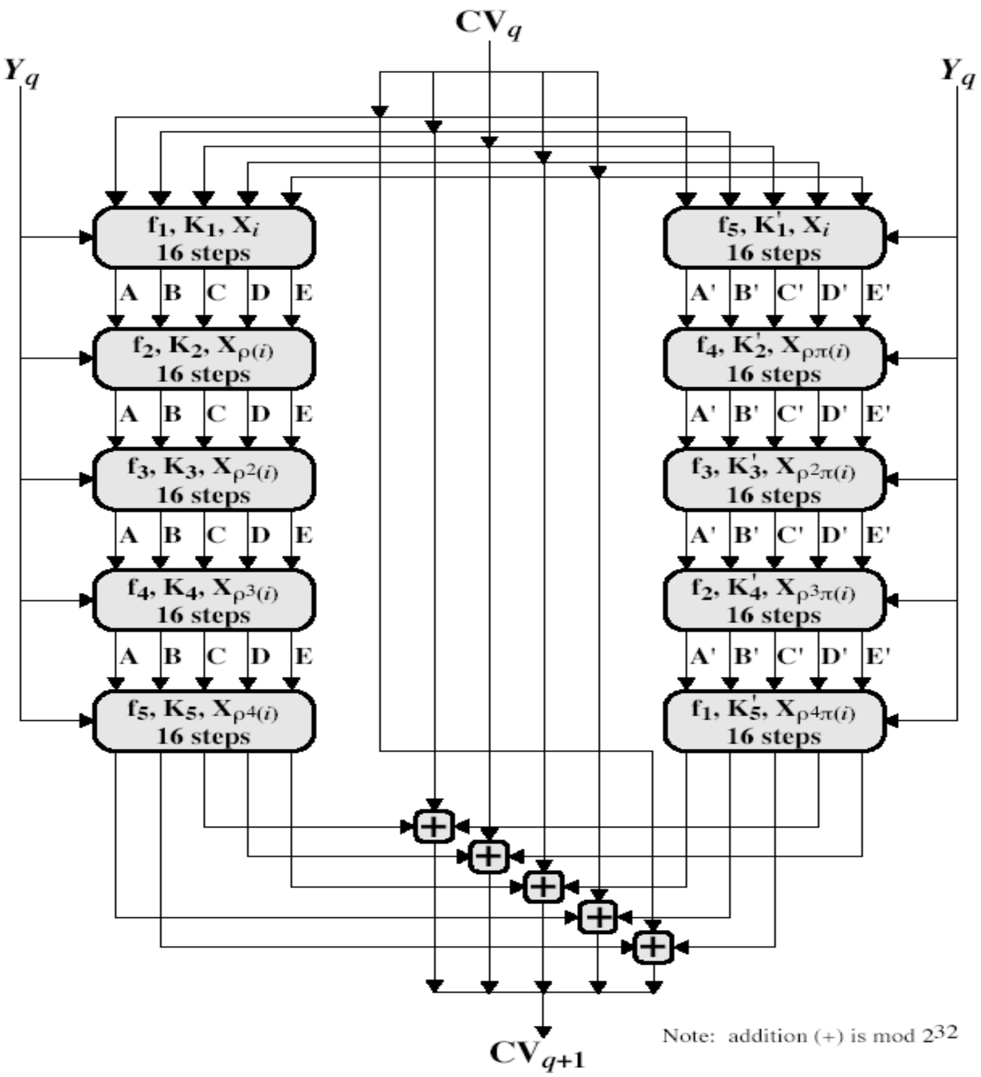
RIPEMD-160

- RIPEMD-160 was developed in Europe as part of RIPE project in 96
- By researchers involved in attacks on MD4/5
- Initial proposal strengthen following analysis to become RIPEMD-160
- Somewhat similar to MD5/SHA
- Uses 2 parallel lines of 5 rounds of 16 steps
- Creates a 160-bit hash value
- Slower, but probably more secure, than SHA

RIPEMD-160 Overview

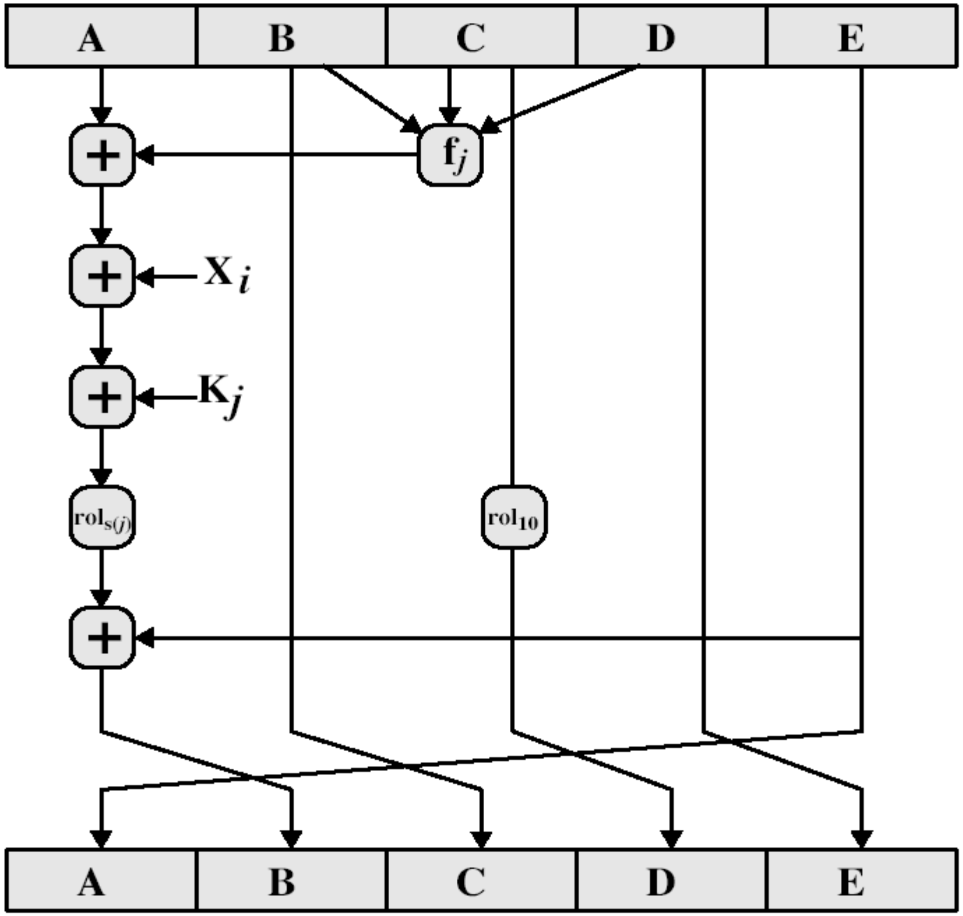
1. Pad message so its length is 448 mod 512
2. Append a 64-bit length value to message
3. Initialize 5-word (160-bit) buffer (A,B,C,D,E) to (67452301,efcdab89,98badcfe,10325476,c3d2e1f0)
4. Process message in 16-word (512-bit) chunks:
   - use 10 rounds of 16 bit operations on message block & buffer – in 2 parallel lines of 5
   - add output to input to form new buffer value
5. Output hash value is the final buffer value

RIPEMD-160 Round


CV_q
Y_q
Y_q
f_1, K_1, X_i 16 steps
f_5, K'_1, X_i 16 steps
A B C D E
A' B' C' D' E'
$f_2, K_2, X_{\rho(i)}$ 16 steps
$f_4, K'_2, X_{\rho\pi(i)}$ 16 steps
A B C D E
A' B' C' D' E'
$f_3, K_3, X_{\rho^2(i)}$ 16 steps
$f_3, K'_3, X_{\rho^2\pi(i)}$ 16 steps
A B C D E
A' B' C' D' E'
$f_4, K_4, X_{\rho^3(i)}$ 16 steps
$f_2, K'_4, X_{\rho^3\pi(i)}$ 16 steps
A B C D E
A' B' C' D' E'
$f_5, K_5, X_{\rho^4(i)}$ 16 steps
$f_1, K'_5, X_{\rho^4\pi(i)}$ 16 steps
CV_{q+1}

Note: addition (+) is mod 2^{32}

RIPEMD-160 Compression Function



RIPEMD-160 Design Criteria

- Use 2 parallel lines of 5 rounds for increased complexity
- For simplicity the 2 lines are very similar
- Step operation very close to MD5
- Permutation varies parts of message used
- Circular shifts designed for best results

RIPEMD-160 verses MD5 & SHA-1

- Brute force attack harder (160 like SHA-1 vs 128 bits for MD5)
- Slower than MD5 (more steps)
- All designed as simple and compact
- SHA-1 optimised for big endian CPU's vs RIPEMD-160 & MD5 optimised for little endian CPU’s
- Also cracked by Xiaoyun Wang in 2004.

Hash Functions & MAC Security

- Brute-force attacks exploiting
  - strong collision resistance hash have cost $2^{m/2}$
    - have proposal for h/w MD5 cracker
    - 128-bit hash looks vulnerable, 160-bits better
  - MACs with known message-MAC pairs
    - can either attack keyspace (cf key search) or MAC
    - at least 128-bit MAC is needed for security

Hash Functions & MAC Security

- **Cryptanalytic attacks** exploit structure
  - like block ciphers want brute-force attacks to be the best alternative
- Have a number of analytic attacks on iterated hash functions
  - $CV_i = f[CV_{i-1}, M_i]; H(M)=CV_N$
  - typically focus on collisions in function f
  - like block ciphers is often composed of rounds
  - attacks exploit properties of round functions

Keyed Hash Functions as MACs

- Creating a MAC using a hash function rather than a block cipher
  - because hash functions are generally faster
  - not limited by export controls unlike block ciphers
- Hash includes a key along with the message
- Original proposal:

```
KeyedHash = Hash(Key||Message)
```

  - some weaknesses were found with this
- Eventually led to development of HMAC

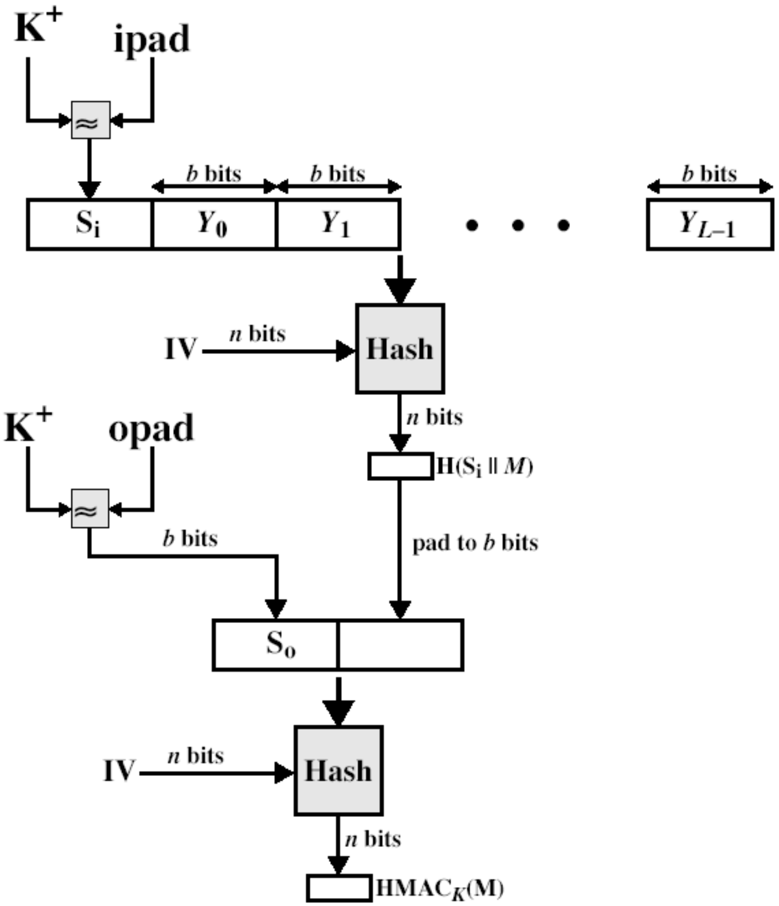
HMAC (Hash-based MAC)

- Specified as Internet standard RFC2104
- Uses hash function on the message:

$$\mathrm{HMAC}_K = \mathrm{Hash}[(K^+ \text{ XOR opad}) \,||\, \mathrm{Hash}[(K^+ \text{ XOR ipad})||M)]]$$

- K^+ is the key padded out to size
- opad, ipad are specified padding constants
- Overhead is just 3 more hash calculations than the message needs alone
- Both MD5, SHA-1, RIPEMD-160 (or any other hash functions) can be used

HMAC Overview



HMAC Security

- Know that the security of HMAC relates to that of the underlying hash algorithm
- Attacking HMAC requires either:
  - brute force attack on key used
  - birthday attack (but since keyed would need to observe a very large number of messages)
- Choose hash function used based on speed verses security constraints

Recent Collision Attacks to Hash Functions

- Collision attacks were announced at Crypto2004
  - in SHA-0 by Antoine Joux
  - in MD4, MD5, HAVAL-128, and RIPEMD by Xiaoyun Wang, with co-authors Dengguo Feng, Xuejia Lai, and Hongbo Yu
  - http://www.cryptography.com/cnews/hash.html
- Most recently SHA-1 is broken by
  - Xiaoyun Wang, Lisa Yiqun Yin, Hongbo Yu http://www.schneier.com/blog/archives/2005/02/sha1_broken.html

Collision Attacks

- The attackers can construct two messages with the same hash value, but the attacker can't pick what the hash will be.
- To exploit a collision attack, one may construct two messages of the same hashing value. Therefore, while signing one but later deny it by presenting the alternative with the same signature.

The Impact of New Collision Attacks

- Practical use of the hashing functions are not directly affected.
- Potentially Harmful in a more general use of the hash functions
- Two alternatives:
  - Use unbroken hash functions: SHA-512?
  - Apply new (and weaker) concept of hash function security in applications.