Notice

This lecture note (Cryptography and Network Security) is prepared by Xiang-Yang Li. This lecture note has benefited from numerous textbooks and online materials. Especially the “Cryptography and Network Security” 2nd edition by William Stallings and the “Cryptography: Theory and Practice” by Douglas Stinson.

You may not modify, publish, or sell, reproduce, create derivative works from, distribute, perform, display, or in any way exploit any of the content, in whole or in part, except as otherwise expressly permitted by the author.

The author has used his best efforts in preparing this lecture note. The author makes no warranty of any kind, expressed or implied, with regard to the programs, protocols contained in this lecture note. The author shall not be liable in any event for incidental or consequential damages in connection with, or arising out of, the furnishing, performance, or use of these.
Cryptography and Network Security

Hash Algorithms

Xiang-Yang Li
Hash Function

- Map a message to a smaller value

- Requirements
  - Be applied to a block of data of any size
  - Produced a fixed length output
  - $H(x)$ is easy to compute (by hardware, software)
  - **One-way**: given code $h$, it is computationally infeasible to find $x$: $H(x)=h$
  - **Weak collision resistance**: given $x$, computationally infeasible to find $y$ so $H(x)=H(y)$
  - **Strong collision resistance**: Computationally infeasible to find $x, y$ so $H(x)=H(y)$
Hash Algorithms

- see 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
- likewise tend to use common iterative structure as do block ciphers
Basic Uses of Hash Function

Six basics usages

- $E_k(M||H(M))$
  - Confidentiality and authentication
- $M||E_k(H(M))$
  - Authentication
- $M||E_{KRa}(H(M))$
  - Authentication and digital signature
- $E_k(M||E_{KRa}(H(M)))$
  - Authentication, digital signature and confidentiality
- $M||H(M||S)$
  - Authentication (S shared by both sides)
- $E_k(M||H(M||S))$
  - Confidentiality and authentication
Birthday Attacks

- If 64-bits hash code is used
  - On average, how many messages need to try to find one match the intercepted hash code?

- Birthday paradox
  - A will sign a message appended with m-bits hash code
  - Attacker generates some variations of fraud message, also variations of good message
  - Find pair of message each from the two sets messages
    - Such that they have the same hash code
  - Give good message to A to get signature
  - Replace good message with fraud message
Analysis

- Using birthday attack, given 64-bits hash code
  - How many message variations needed so the success probability is large, say 90%?
Examples

Simple hash functions

- XOR of the input message
  - $H(M) = X_1 \oplus X_2 \oplus \ldots \oplus X_{m-1} \oplus X_m$

- But not secure
  - $Y_m = H(M) \oplus Y_1 \oplus Y_2 \oplus \ldots \oplus Y_{m-1}$ has same hash value as
  - $(X_1 X_2 \ldots X_{m-1} X_m)$, where $Y_i$ is any value
Based on DES, block chaining technique

- Rabin, 1978
- Divide message $M$ into fix-sized blocks $M_i$
  - Assume total $n$ data blocks
- $H_0 =$ initial value
- $H_i =$ $E_{m_i}[H_{i-1}]$
- $H_n$ is the hash value

Birthday attack still applies

- If still 64-bits code used
More Attacks

- Birthday attack applied if chosen plaintext
- Meet in the middle attack if known plaintext
  - Known signed hash code $G$
  - Construct $n-2$ desired message block $Q_i$
  - Compute $H_i = E_{Q_i}[H_{i-1}]$
  - Generate $2^{m/2}$ random blocks $X$
    - For each $X$, Compute $H_{n-1} = E_X[H_{n-2}]$
  - Generate $2^{m/2}$ random blocks $Y$
    - For each $Y$, Compute $H'_{n-1} = D_Y[G]$
  - Find $X, Y$ such that $H_{n-1} = H'_{n-1}$
  - Then $Q_1, Q_2, \ldots Q_{n-2}, X, Y$ is a fraud message
Security

- The size of hash code determines security
  - 128bits is not secure
  - Currently, most use 160 bits hash code
    - Now recommend 256 bits
- Attack MAC
  - Objective is to find valid \((x, C_k(x))\) pair
  - Attack the key space: roughly \(2^k\), \(k = \text{key size}\)
  - Attack the MAC value
More Hash Algorithms

- **Algorithms**
  - Message Digest: MD5 (was mostly widely used)
  - Secure Hash Algorithm: SHA-1 (from MD4)
  - RIPEMD-160
  - HMAC
**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

\[ L \times 512 \text{ bits} = N \times 32 \text{ bits} \]

Padding (1 to 512 bits)

Message length \((k \mod 2^{64})\)

\[ \text{Message} \]

\[ \text{Y}_0 \quad \text{Y}_1 \quad \ldots \quad \text{Y}_q \quad \ldots \quad \text{Y}_{L-1} \]

\[ \text{H}_{\text{MD5}} \quad \text{H}_{\text{MD5}} \quad \text{H}_{\text{MD5}} \quad \text{H}_{\text{MD5}} \]

IV 128

\[ \text{CV}_1 \quad \text{CV}_q \quad \text{CV}_{L-1} \]

128-bit digest

Cryptography and Network Security 16
MD5 Compression Function

- each round has 16 steps of the form:
  \[ a = b + ((a + g(b, c, d) + X[k] + T[i]) \ll s) \]

- \( a, b, c, d \) 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 \( \sin \)
MD5 Compression Function
MD4

- precursor to MD5
- also produces a 128-bit hash of message
- has 3 rounds of 16 steps vs 4 in MD5
- design goals:
  - collision resistant (hard to find collisions)
  - direct security (no dependence on "hard" problems)
  - fast, simple, compact
  - favours little-endian systems (eg PCs)
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)
- Conclusion is that MD5 looks vulnerable soon
Bad news

- Chinese authors (Wang, Feng, Lai, and Yu) reported a family of collisions in MD5
  - (fixing the previous bug in their analysis), and also reported that their method can efficiently ($2^{40}$ hash steps) find a collision in SHA-0.
  - August Crypto 2004,

- MD5 is fatally wounded; its use will be phased out. SHA-1 is still alive but the vultures are circling. A gradual transition away from SHA-1 will now start. The first stage will be a debate about alternatives, leading to a consensus among practicing cryptographers about what the substitute will be.
Why collisions are bad

- An example of what you might do with this.
  - You could request an SSL certificate (for your real identity) from a certificate authority. After the response comes back, you can then use that response (which is based on the MD5 of your identity+key) to "authenticate" a carefully chosen different certificate, one which claims that you are LargeBankOrSoftwareCorp., but which has the same MD5 as your real identity. You can then present this to other people in order to convince them that you are someone whom you are not.

- Another example,
  - core internet routers use md5 to exchange passwords. I simply sniff the md5sum, and if I can find a string that generates the same sum, easily, I can send my own routing update that takes down the internet. More examples, since a LOT of applications use md5, but you get the idea.
Further detail

- Obviously the above attack isn't quite so simple, but this research makes it *possible*. Before, it was believed to be sufficiently difficult to find a collision, that nobody worried about it. Now they are saying its feasible to do it in hours.

- The question hanging around right now is that these researchers managed to find collisions easily, but not for an arbitrary string. The questions is how long before someone modifies this method to find any collision. That is how much time the world has to move away.

- More at
What to do next

- The U.S. National Institute of Standards and Technology is having a competition for a new cryptographic hash function.
- The phrase "one-way hash function" might sound arcane and geeky, but hash functions are the workhorses of modern cryptography.
- Submissions will be due in fall 2008, and a single standard is scheduled to be chosen by the end of 2011.
- We have an interim solution in SHA-256.
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
   \((67452301,\text{efcdab89},98\text{badcfe},10325476,\text{c3d2e1f0})\)
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:
  \[(A, B, C, D, E) \leftarrow (E + f(t, B, C, D) + (A \ll 5) + W_t + K_t), A, (B \ll 30), C, D)\]

- \(a, b, c, d\) 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)
- not vulnerable to any known attacks (compared to MD4/5)
- 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)
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. 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:
   - 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

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)
- not vulnerable to known attacks, like SHA-1 though stronger (compared to MD4/5)
- 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
Keyed Hash Functions as MACs

- have desire to create 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:
  \[ \text{KeyedHash} = \text{Hash}(\text{Key} | \text{Message}) \]
  - some weaknesses were found with this
- eventually led to development of HMAC
HMAC

- specified as Internet standard RFC2104
- uses hash function on the message:
  \[ \text{HMAC}_K = \text{Hash}[(K^+ \text{ XOR opad}) || \text{Hash}[(K^+ \text{ XOR ipad}) || M])] \]
- where \( K^+ \) is the key padded out to size
- and \text{opad}, \text{ipad} are specified padding constants
- overhead is just 3 more hash calculations than the message needs alone
- any of MD5, SHA-1, RIPEMD-160 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
Summary

- have considered:
  - some current hash algorithms: MD5, SHA-1, RIPEMD-160
  - HMAC authentication using hash function