Lecture 8b

Authentication and Hash Functions (2)

Random Oracle Model

- Ideal Hash Function
  - Only feasible or efficient way to determine the value $h(x)$ is to actually evaluate the function $h(.)$ at x
- Another insecure hash function
  - $h(x||z) = y = ax + bz \bmod n$
  - Compute $h(x||z) = y$ and $h(u||v) = w$
  - Knowing y and w you can calculate any other hash value
  - In particular $h(rx + su || rz + sv) = ry + sw \bmod n$
- Random oracle
  - We do not know the function $h(x)$
  - We only have "oracle" access to the hash value
  - The output of the oracle is completely random and independent of other outputs

Average case success in finding the Preimage

- Suppose you perform q trials and the hash size is $M=2^k$ (the output of the hash function is k bits)
- The probability that none of the messages satisfy $h(x) = y$ is
  - $(1 - 1/M)^q$
- The probability that at least one of them will satisfy $h(x) = y$ is
  - $1 - (1 - 1/M)^q$
- You can perform a similar calculation for computing the average success in finding the second preimage in q trials

The Birthday Paradox

- How many people do you need before the odds that two of them have the same date of birth is greater than 50%?
  - 183?

Success of Collisions

- Hash functions are many to one mappings
- There always exists some $x_1 \neq x_2$ such that $h(x_1) = h(x_2)$
- Algorithm to find a collision in q trials
  - Choose any $X_o \in X$ such that $|X_o| = q$
  - For each $x_i \in X_o$ find $h(x_i) = y_i$
  - Sort the y_i's to see if there is a collision
    - Example $h(x_7) = y_7 = h(x_{29}) = y_{29}$

Probability of successful collision in q trials

- Suppose we have two hash values y_1 and y_2
- What is the probability that they are identical?
  - Probability that $y_2 \neq y_1$ is (1 – 1/M)
  - Probability that $y_2 = y_1$ is 1 – (1 – 1/M)
- For q such values, you can show that the probability of no collisions is $\left(1-\frac{1}{M}\right)\left(1-\frac{2}{M}\right)\left(1-\frac{3}{M}\right)\cdots\left(1-\frac{q-1}{M}\right) = \prod_{i=1}^{q-1}\left(1-\frac{i}{M}\right)$
- If i/M is small, then $1 - i/M \cong e^{-i/M}$

Probability of successful collision in q trials - II

- Make simplifying approximations
- Let ε be the probability of at least one collision in q trials
- Then $$\varepsilon = 1 - e^{\frac{-q(q-1)}{2M}}$$
- Alternatively $$q \approx \sqrt{2M \ln \frac{1}{1-\varepsilon}}$$
- If $\varepsilon = 0.5$, we see that $q \cong 1.17\sqrt{M}$

Hashing the square root of the number of elements is sufficient to give us a collision with probability 0.5

Example

- Suppose the length of the hash code is 40 bits.
- The number of possible hash values is $M = 2^{40}$
- If we want a collision with a probability of 0.5, we need to check q plaintexts where

$$q = 1.17\sqrt{M}$$

- In other words, if you check roughly 2^{20} plaintexts, there is a 50% chance that you will find two of them with $x_1 \neq x_2$ but $h(x_1) = h(x_2)$

The birthday paradox and the security of hash functions

- The probability that at least two people in a group of 23 have the same birthday is 0.5
- Because of this, the hash code should have a length of at least 160 bits (square root of 2^{160} is 2^{80} – remember skipjack)
- Today, we usually prefer a hash code of length 256 bits

How can we create Secure Hash Functions

- Merkle-Damgard Construction
  - The name "MD" comes from here
  - MD4, MD5 are the earliest
- Sponge functions
  - SHA-3 is based on sponge functions

Hash Algorithms

- (1) MD5
  - Developed by Ron Rivest
  - Size of the hash code is 128 bits (not secure anymore)
  - Specified in IETF RFC 1321
  - Some cryptanalytic attacks based on differential cryptanalysis have made it weaker
  - It is used very commonly in VPNs
- (2) SHA (Secure Hash Algorithm) (or SHA-1)
  - Standardized in the NIST FIPS PUB 180-1 in 1995
  - Design is very similar to MD5
  - Size of the hash code is 160 bits

Hash Algorithms (2)

- SHA
  - Input is a multiple of 512 bits
  - Uses bitwise operations (AND, OR, NOT, XOR etc.) and also cyclic shifts
  - Slower than MD5 but works well on 32 bit architectures
  - More secure than MD5
  - Cryptanalytic attacks?
- Available since 2001 (SHA-2)
  - SHA-256, SHA-384, and SHA-512

Hash Algorithms (3)

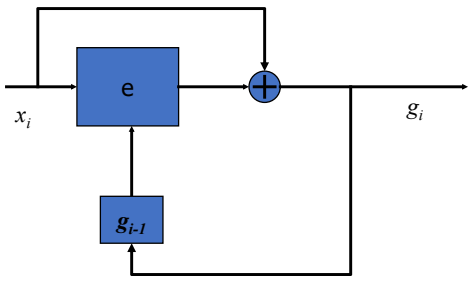
- RIPEMD
  - Developed by the European RACE project
  - Hash code length is 160 bits
  - Input is a multiple of 512 bits
  - Similar to SHA and more secure but slower than MD5
    - Runs two parallel algorithms in each round and combines the outputs
  - Cryptanalysis is more difficult, but used less
- SHA-3 (⊂Keccak) (variable length outputs: 224, 256, 384, or 512 bits)
  - Uses Sponge functions

Hash function using block ciphers

- Compute a running variable as follows:
  - $g_i = f(g_{i-1}, x_i)$
- Set g_0 = IV
- Use the previous running variable as the key to the next so that the key changes with each iteration
- Assumption here is that the key size, the plaintext block and ciphertext block are all the same size

Hash functions using block ciphers

- Example: $x = x_1 || x_2 || x_3 || x_4 || \ldots x_n$
- g_0 = IV (initial vector)
- $g_i = e_{g(i-1)}(x_i) \oplus x_i$; The hash value is g_n



Message Authentication Codes

Message Authentication Codes (MACs)

- Also called cryptographic checksums or keyed hash functions
- MAC is usually written as $C_k(x)$
- Depends on a key that is shared between Alice and Bob
- MAC has a fixed length that is much smaller than the message length

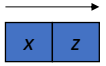
General Idea of a MAC

Alice

Insecure channel

Bob

Create x
Calculate $C_k(x) = z$
Send $x||z$

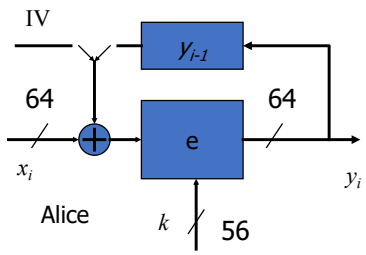


Get x, z
Calculate $C_k(x) = z^*$
Check if $z^* = z$

Security of a MAC

- The key size should be large enough to prevent brute force attacks
  - Length of k should be > 80 bits
- Given x and $C_k(x)$, Oscar should not be able to find or generate x_1 such that $C_k(x_1) = C_k(x)$
  - Similar to the collision property of hash functions
- $C_k(x)$ should be uniformly distributed
  - If there are two messages x_1 and x_2, and the size of the MAC is m bits, then the probability that $C_k(x_1) = C_k(x_2)$ should be 2^{-m}
- Changing the message x should result in a completely random MAC (similar to block cipher principles)

MAC based on CBC mode of DES



- This is NIST FIPS PUB 113 (ANSI X9.17 standard)
- Also called Data Authentication Algorithm

Let $x = x_0 || x_1 || x_2 || x_3 || x_4 || \ldots x_n$
Set IV = 0 (all zeros)
Compute $y_0, y_1, y_2, y_3, y_4, \ldots y_n$
Here $y_0 = e_k(x_0), y_1 = e_k(x_1 \oplus y_0), \ldots$

$$MAC\ is\ C_k(x) = y_n$$

HMAC

- Idea: Use hash functions to generate a MAC
  - $C_k(x) = h(x||k)$
  - Actually $\text{HMAC}_k(x) = h[\ (k^+ \oplus \texttt{opad})||h[(k^+ \oplus \texttt{ipad})||x]]$
  - k^+ is a key padded with zeros to make it the same length b as the message block
  - Ipad is 00110110 repeated $b/8$ times
  - Opad is 01011010 repeated $b/8$ times
- Can be proved to be secure – can only be broken if the hash function itself is insecure

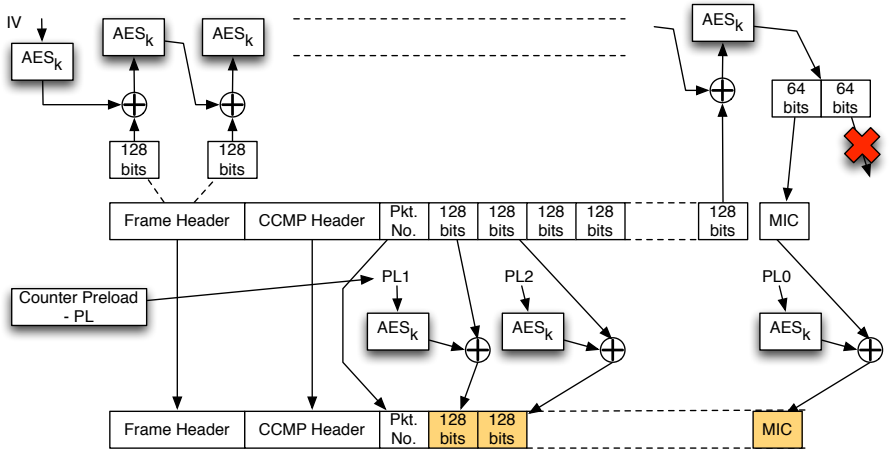
Why HMAC?

- Hash functions are faster in software compared to block ciphers
- There are no export restrictions on Hash Functions
- Issued as RFC 2104
- Used in SSL and mandatory in IPsec
- Since the key is secret, on a 1 Gbps link it would take 250,000 years to get enough data to break HMAC using the birthday paradox or by brute force
- HMAC uses SHA or RIPEMD-160 and they are sufficiently secure

Other MACs

- Authenticated Encryption
  - Uses CTR mode encryption and a CBC-MAC
    - Also called CCM
  - GMAC
    - Galois counter mode MAC
    - Uses a mode of operation we have not explored in this class
- CBC MAC is used in many wireless standards
  - 802.11i for wireless LANs
  - 802.15.4 for wireless PANs
- CMAC
  - https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-38B.pdf

AES CCMP Block Diagram for WiFi



Block Chains (very simplified)

- You have already heard the word “block chaining”
  - CBC mode
- Block chains are somewhat different, but have similar constructions using hash functions
- What do you need to verify if x_k has not been modified?