[CS309] Foundations of Cryptography and Security Networks

Instructor: Dr. Dibyendu Roy — Autumn Semester 2024
Notes by: Tanuj Saini (202251141) — Week 7 Lecture

1 Composition Function

A compression function is an essential element in cryptographic hash functions. It accepts an input message of length $(m+t)$ and produces an output of fixed length m, where $t \geq 1$:

$$h : \{0,1\}^{m+t} \rightarrow \{0,1\}^m$$

The security of the hash function H is contingent upon the security of the compression function h.

Given an input $x \in \{0,1\}$ with a length denoted by $|x|$ (where $|x| \geq m+t+1$): Construct y from x using a public function such that $|y| \equiv 0 \mod (t)$.

- If $|x| \equiv 0 \mod (t)$, then $y = x$.
- If $|x| + d \equiv 0 \mod (t)$, then $y = x \parallel 0^x$.
- Otherwise, $y = y_1 \parallel y_2 \parallel \ldots \parallel y_r$ such that $|y| = t$ for $1 \leq i \leq r$.

Let $Z_0 = IV$, and compute:

$$Z_1 = h(Z_0 \parallel y_1)$$

Continuing in this manner, we have:

$$Z = h(Z_{r-1} \parallel y_r)$$

2 Merkle–Damgård Construction

The Merkle-Damgård construction is commonly used in cryptographic hash function design. It breaks the input message into fixed-size blocks and iteratively applies a compression function to produce the final hash value.

$$h : \{(0,1)\}^* \rightarrow \{(0,1)\}^m$$

The compression function:

$$\text{Compress} : \{(0,1)\}^{m+t} \rightarrow \{(0,1)\}^m$$

Let $n = |x|$ and define:

$$k = \left\lfloor \frac{n}{t} - 1 \right\rfloor$$

$$d = k(t-1) - n$$

For $i = 1$ to k:

$$y_i = x_i$$

Construct:

$$y_m = x_k \parallel O^{(d)}$$

$$y_{m+1} = \text{binary}(d)$$

Initialization:

$$z_1 = O^{m+1} \parallel y_1$$

$$g_1 = \text{compress}(z_1)$$

For $i = 1$ to k:

$$z_{i+1} = g_i \parallel 1 \parallel y_{i+1}$$

$$g_{i+1} = \text{compress}(z_{i+1})$$

Final output:

$$h(x) = g_{k+1}$$

Return $h(x)$.

3 Secure Hash Functions (SHA)

The Secure Hash Algorithm (SHA) was developed by the National Institute of Standards and Technology (NIST) and was first introduced as a federal information processing standard (FIPS 180) in 1993. Over time, revisions were made, with SHA-1 being published in 1995 as FIPS 180-1. Other versions in the SHA family include SHA-224, SHA-256, SHA-384, and SHA-512.

3.1 SHA-1

The Secure Hash Algorithm 1 (SHA-1) is a cryptographic hash function designed by the National Security Agency (NSA) and published by the National Institute of Standards and Technology (NIST) as part of the Digital Signature Algorithm (DSA). SHA-1 outputs a fixed-length, 160-bit (20-byte) message digest from an input of arbitrary length, typically represented as a 40-character hexadecimal number.

3.2 Algorithm

SHA-1 is constructed using the Merkle-Damgård design and processes input in blocks. The core steps of the SHA-1 algorithm are outlined as follows:

1. Message Padding: The input message is padded to ensure its bit length is congruent to 448 mod 512. Padding involves appending a '1' bit, followed by enough '0' bits, and finally adding the 64-bit representation of the message's original length.

2. Message Parsing: The padded message is split into 512-bit blocks.

3. Initial Hash Values: SHA-1 starts with five 32-bit registers, denoted H_0, H_1, H_2, H_3, H_4, initialized as follows:

$$
\begin{aligned}
H_0 &= \text{0x67452301} \\
H_1 &= \text{0xEFCDAB89} \\
H_2 &= \text{0x98BADCFE} \\
H_3 &= \text{0x10325476} \\
H_4 &= \text{0xC3D2E1F0}
\end{aligned}
$$

4. Processing Each Block: For each 512-bit block, the following steps are carried out:

   (a) The block is divided into sixteen 32-bit words, and an additional 64 words are generated through bitwise operations.

   (b) 80 rounds are executed, involving mixing the message schedule words with the hash values, using bitwise operations such as AND, OR, XOR, and rotations.

   (c) The result from each block's processing updates the hash values H_0 to H_4.

5. Final Output: After processing all the blocks, the concatenation of H_0, H_1, H_2, H_3, H_4 produces the final 160-bit hash value.

3.3 SHA-1 Rounds

SHA-1 involves 80 rounds of processing for each block, with four constants used based on the round number:

$$
K_t = \begin{cases}
\text{0x5A827999} & \text{for } 0 \leq t \leq 19 \\
\text{0x6ED9EBA1} & \text{for } 20 \leq t \leq 39 \\
\text{0x8F1BBCDC} & \text{for } 40 \leq t \leq 59 \\
\text{0xCA62C1D6} & \text{for } 60 \leq t \leq 79
\end{cases}
$$

The core loop of the algorithm applies bitwise logical functions F_t based on the round number t, utilizing word additions and rotations.

4 SHA-256

SHA-256, part of the SHA-2 family, is a cryptographic hash function that generates a fixed-size 256-bit (32-byte) hash value from an arbitrary-length input. Designed by the NSA and published by NIST, SHA-256 operates similarly to SHA-1 but with notable differences in block size and operations.

4.1 Steps of SHA-256

1. Message Padding:
   - The input message is padded so that its bit length is congruent to 448 mod 512.
   - Padding involves appending a single '1' bit, followed by '0' bits, and ending with a 64-bit representation of the original message length.
2. Message Parsing: The padded message is split into 512-bit blocks, which are processed sequentially.
3. Initial Hash Values: Eight 32-bit registers (H_0, H_1, H_2, H_3, H_4, H_5, H_6, H_7) are initialized to constants derived from the fractional parts of the square roots of the first eight primes:

$$\begin{aligned}
H_0 &= \text{0x6a09e667} \\
H_1 &= \text{0xbb67ae85} \\
H_2 &= \text{0x3c6ef372} \\
H_3 &= \text{0xa54ff53a} \\
H_4 &= \text{0x510e527f} \\
H_5 &= \text{0x9b05688c} \\
H_6 &= \text{0x1f83d9ab} \\
H_7 &= \text{0x5be0cd19}
\end{aligned}$$

4. Processing Each Block:
   (a) Message Schedule: A sequence of 64 words W_t is generated from each block, with the first 16 directly derived from the block and the remaining 48 produced by:

$$W_t = \sigma_1(W_{t-2}) + W_{t-7} + \sigma_0(W_{t-15}) + W_{t-16}$$

   where:

$$\begin{aligned}
\sigma_0(x) &= (x \ggg 7) \oplus (x \ggg 18) \oplus (x \ggg 3) \\
\sigma_1(x) &= (x \ggg 17) \oplus (x \ggg 19) \oplus (x \ggg 10)
\end{aligned}$$

(b) Compression Function: For each round t:

$$T_1 = H + \Sigma_1(E) + Ch(E, F, G) + K_t + W_t$$
$$T_2 = \Sigma_0(A) + Maj(A, B, C)$$

where:

$$\begin{aligned}
\Sigma_0(x) &= (x \ggg 2) \oplus (x \ggg 13) \oplus (x \ggg 22) \\
\Sigma_1(x) &= (x \ggg 6) \oplus (x \ggg 11) \oplus (x \ggg 25) \\
Ch(x, y, z) &= (x \wedge y) \oplus (\neg x \wedge z) \\
Maj(x, y, z) &= (x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z)
\end{aligned}$$

Variables are updated as follows:

$$H = G, G = F, F = E, E = D + T_1, D = C, C = B, B = A, A = T_1 + T_2$$

5. Final Hash: After all blocks are processed, the final hash is obtained by concatenating the updated values:

$$\text{Hash} = H_0 \parallel H_1 \parallel H_2 \parallel H_3 \parallel H_4 \parallel H_5 \parallel H_6 \parallel H_7$$

The final output is a 256-bit (32-byte) value.

4.2 Round Constants

SHA-256 uses 64 constant values K_t, one for each round, which are derived from the first 32 bits of the fractional parts of the cube roots of the first 64 primes.

5 Diffie-Hellman Key Exchange

The diagram below represents a symmetric key encryption setup:

The Diffie-Hellman Key Exchange Algorithm introduced a new paradigm in cryptography, commonly referred to as Public Key Cryptography.

G represents a cyclic group defined as $\langle g \rangle$: $(G, *)$

1. For any $a, b \in G$, then $a * b \in G$.
2. There exists an identity element $e \in G$ such that:

$$e * a = a * e = a, \forall a \in G.$$

3. Each $a \in G$ has an inverse $a^{-1} \in G$ such that:

$$a * a^{-1} = a^{-1} * a = e$$

4. G is associative.

6 RSA

$\Phi(m)$: the number of integers less than m that are coprime with m.

Example: $\Phi(m) = 4 \to 1, 3, 5, 7$

For a prime number p, $\Phi(p) = p - 1$. p: prime

For powers of a prime, $\Phi(p^k) = p^k - p^{k-1}$.

Alternatively, $\Phi(p^k) = p^k \left(1 - \frac{1}{p}\right)$.

If $\gcd(a, m) = 1$, then: $S = x \mod m$ S consists of $\{r_1, r_2, \ldots, r_m\}$.

Multiplying S by a gives: $\{ar_1, ar_2, \ldots, ar_m\}$.

If $ar_i = ar_j$, then $r_i = r_j$. This is possible only if $r_i \neq r_j$.

Since $\gcd(a, m) = 1$, we can express this as:

$$1 = a \cdot b + m \cdot s$$

Thus, $\exists$ some b such that $a \cdot b \equiv 1 \mod m$.

If $ar_i \equiv ar_j \mod m$ and $r_i \neq r_j$, we get:

$$b \cdot ar_i \equiv b \cdot ar_j \mod m.$$

Hence, $r_i \equiv r_j \mod m$, and therefore, $ar_i \neq ar_j \mod m$.

6.1 Fermat's Little Theorem

For a prime p, if a is an integer not divisible by p, then:

$$a^{p-1} \equiv 1 \pmod{p}$$

7 Mitigating Man-in-the-Middle Attacks

7.1 Endpoint Authentication

- Digital signatures can verify the integrity and authenticity of messages.
- Public-key cryptography: Use the private key for signing and the public key for verification.

7.2 Secure Key Exchange Protocols

- Diffie-Hellman key exchange: Enables secure generation of shared secrets.
- RSA key exchange: The shared secret is encrypted using the recipient's public key.

8 Euler's Totient Function and Theorem

8.1 Definition and Properties

- Definition: $\Phi(m)$ counts the number of integers less than m that are coprime with m.
- Properties:
  - For a prime p, $\Phi(p) = p - 1$.
  - For two distinct primes p and q, $\Phi(pq) = (p-1)(q-1)$.
  - For powers of a prime p, $\Phi(p^k) = p^k - p^{k-1}$.

8.2 Euler's Theorem

If a and m are coprime, then $a^{\Phi(m)} \equiv 1 \mod m$. This is essential for RSA and other cryptographic algorithms.