[CS309] Introduction to Cryptography and Network Security

Course Instructor: Dr. Dibyendu Roy | Autumn 2024-2025
Scribed by: Kunj Thakkar (202251142) | Lecture (Week 7)

1 Compression function in hashing

- $h : \{0,1\}^{m+t} \rightarrow \{0,1\}^m$ is a hash function that takes inputs of length $m + t$ and produces output of length m.
- **Goal**: make $H : \{0,1\}^* \rightarrow \{0,1\}^m$ from h. This means that H takes input of any length and produces output of length m.

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

$$SecondPreimage, preimage \rightarrow O(2^m)$$

$$Collision \rightarrow O(2^{m/2})$$

Algorithm 1: Compress

Assumption: The function `Compress`: $\{0,1\}^{m+t} \rightarrow \{0,1\}^m$ is defined as a compression function.

Input:

- x: A string whose length is greater than $m + t + 1$.

Output:

- $h(x)$: The hash value produced from the input string x.

Procedure:

1. Pad x with zeros to form a new string y such that the length of y is a multiple of t.
2. Split y into parts as $y = y_1 \| y_2 \| \ldots \| y_r$, where each y_i is of length t, except possibly the last one.
3. Set the initial value $z_0 \leftarrow IV$.
4. For $i = 1$ to r, perform the following:
   - Update $z_i \leftarrow \text{compress}(z_{i-1} \| y_i)$.

---

2 Secure Hash Function(SHA)

SHA was proposed as standard hashing function by NSIT in 1993, and adopted by FIPS 180.SHA-I is slight modification to SHA, it was published in 1995 as FIPS 180-1(and SHA was then referred as SHA-0)

2.1 Types of SHA

Secure Hash Functions (SHA)

SHA-1	**SHA-256**	**SHA-512**
Message Size: $< 2^{64}$ bits	Message Size: $< 2^{64}$ bits	Message Size: $< 2^{128}$ bits
Block Size: 512 bits	Block Size: 512 bits	Block Size: 1024 bits
Word Size: 32 bits	Word Size: 32 bits	Word Size: 64 bits
Message Digest Size: 160 bits	Message Digest Size: 256 bits	Message Digest Size: 512 bits

2.2 SHA-1

In SHA-1 message size should be less than 2^{64} bits. If the message size is less than 2^{64}, then padding is applied such that y becomes multiple of 512 bits by appending single '1' and then remaining '0s'

Algorithm 2: SHA-1 Process

Input: x: The input message
Output: $h(x)$: The hash value of x

$n \leftarrow |x|$;
$K \leftarrow \left\lfloor \frac{n}{t-1} \right\rfloor$;
$d \leftarrow K(t-1) - n$;
for $i = 1$ *to* $K - 1$ **do**
  $y_i \leftarrow x_i$;
$y_K \leftarrow x_K \| 0^d$;
$y_{K+1} \leftarrow \text{binary}(d)$;
$Z_1 \leftarrow 0^{m+1} \| y_1$;
$g_1 \leftarrow \text{compress}(Z_1)$;
for $i = 1$ *to* K **do**
  $Z_{i+1} \leftarrow g_i \| 1 \| y_{i+1}$;
  $g_{i+1} \leftarrow \text{compress}(Z_{i+1})$;
$h(x) \leftarrow g_{K+1}$;
return $h(x)$;

2.3 SHA-2

SHA-2 is a family of cryptographic hash functions, including SHA-256 and SHA-512, which produce 256-bit and 512-bit hash values respectively. Unlike SHA-1, SHA-2 is more secure and resistant to collision attacks. The input message must be less than 2^{64} bits for SHA-256 or 2^{128} bits for SHA-512. Padding ensures the message length is a multiple of the block size (512 bits for SHA-256, 1024 bits for SHA-512), followed by processing in multiple rounds to produce the final hash.

Algorithm 3: SHA-2 Process

Input: x: The input message
Output: $h(x)$: The hash value of x

$n \leftarrow |x|$;
if $n \geq 2^{128}$ *for SHA-512 or* $n \geq 2^{64}$ *for SHA-256* **then**
  return *Error: Message too long* ;
Padding: Add '1' bit, followed by '0's, and append n (original length) ;
Divide the padded message into blocks ;
Initialize hash values $H_0, H_1, \ldots$;
for *each block* **do**
  Process using bitwise operations and constants ;
$h(x) \leftarrow$ Final combination of $H_0, H_1, \ldots$;
return $h(x)$;

3 Euler's Theroem

If $gcd(a, m) = 1$ **then** $a^{\phi(m)} \equiv 1 \pmod{m}$.
Let us assume we have set S, such that:

$$S = \{x | gcd(x, m) = 1\}$$

$$S = \{s_1, s_2, s_3, ..., s_{\phi(m)}\}$$

Set S contains all the numbers which are less than m and are coprime with m.
lets gcd(a,m) = 1 and create another set S_1 such that:

$$S_1 = \{a * s_1, a * s_2, a * s_3, ..., a * s_{\phi(m)}\}$$

Here every element of S_1 is coprime to m as a and s_i are coprime to m.
The number of elements in S and S_1 are equal i.e. $\phi(m)$.

$$|S| = \phi(m)$$

$$|S_1| = \phi(m)$$

If we take the product of all elements in S and S_1, it gives:

$$s_1 \cdot s_2 \cdot \ldots \cdot s_{\phi(m)} \equiv (a \cdot s_1) \cdot (a \cdot s_2) \cdot \ldots \cdot (a \cdot s_{\phi(m)}) \pmod{m}$$

Simplifying the right side gives:

$$s_1 \cdot s_2 \cdot \ldots \cdot s_{\phi(m)} \equiv a^{\phi(m)} \cdot (s_1 \cdot s_2 \cdot \ldots \cdot s_{\phi(m)}) \pmod{m}$$

Since the values $s_1, s_2, \ldots, s_{\phi(m)}$ are non-zero and coprime to m, we can cancel out the terms:

$$a^{\phi(m)} \equiv 1 \pmod{m}$$

This confirms Euler's theorem.

4 Fermats theorem

If p is prime and p is coprime with a then

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

. using Fermat's theorem we cansay that $a^p \equiv a \pmod{p}$
Note: Fermat's theorem will not hold when p does not divide a.