[CS304] Introduction to Cryptography and Network Security

Course Instructor: Dr. Dibyendu Roy — Winter 2023-2024
Scribed by: Archit Verma (202151192) — Lecture (Week 7 and 8)

1 Algorithm for Finding a Second Preimage

Given a hash function $h : X \to Y$ where $|Y| = m$, we aim to find for a given input x and its hash $h(x)$, another input x' such that $h(x) = h(x')$ and $x \neq x'$. Let X_0 be a subset of $X - \{x'\}$, meaning $X_0 \subseteq X - \{x'\}$. $|X_0| = q$, representing the number of elements in X_0. The complexity of the operation is of order n.

2 Collision Finding Algorithm

Consider a hash function $h : X \to Y$, with $|Y| = m$. The task is to find two distinct inputs $x, x' \in X$ such that $h(x) = h(x')$, where $x \neq x'$.

Let $X_0 = \{x_1, x_2, \ldots, x_q\}$ be a subset of X excluding x', and let's evaluate the probability of finding a collision within X_0.

For each pair $x, x' \in X_0$, compute:

$$Y_x = h(x),$$

$$Y_{x'} = h(x').$$

If there exists some $x \neq x'$ such that $Y_x = Y_{x'}$, then return (x, x') as the collision pair. Otherwise, return failure. .

2.1 Probability of Collision

To derive the formula for the probability of collision, let's start with the probability of not having a collision after hashing Q distinct inputs:

$$\text{Probability of no collision} = \frac{m}{m} \times \frac{m-1}{m} \times \frac{m-2}{m} \times \ldots \times \frac{m-(Q-1)}{m}$$

This represents the probability that each of the Q hashed values is different from the previously hashed values. To find the probability of having at least one collision, we take the complement of the probability of no collision:

$$\text{Probability of at least one collision} = 1 - \text{Probability of no collision}$$

Substituting the expression for the probability of no collision:

$$\text{Probability of no collision} = 1 - \frac{(Q-1)(Q)}{2M}$$

This formula represents the probability of at least one collision occurring after hashing Q distinct inputs with a hash function that produces M possible output values.

3 General Formula for Complexity

To derive a general formula for the complexity, we start with the expression:

$$\text{Complexity} = \sqrt{M} \times \frac{1}{\epsilon}$$

where ϵ is the probability of failure. We approximate $\epsilon = 1 - e^{-1/M}$, and rewrite the complexity formula as:

$$\text{Complexity} = \sqrt{M} \times \frac{1}{1 - e^{-1/M}}$$

To find a general formula, we express ϵ in terms of ϵ directly:

$$\epsilon = 1 - e^{-1/M}$$

$$e^{-1/M} = 1 - \epsilon$$

$$-\frac{1}{M} = \ln(1 - \epsilon)$$

$$\frac{1}{M} = -\ln(1 - \epsilon)$$

$$M = \frac{1}{-\ln(1 - \epsilon)}$$

Substituting this into the complexity formula:

$$\text{Complexity} = \sqrt{\frac{1}{-\ln(1 - \epsilon)}} \times \frac{1}{\epsilon}$$

$$\text{Complexity} = \sqrt{\frac{-1}{\ln(1 - \epsilon)}} \times \frac{1}{\epsilon}$$

Taking the reciprocal of the square root and applying the property $\sqrt{\frac{1}{a}} = \frac{1}{\sqrt{a}}$:

$$\text{Complexity} = \frac{1}{\sqrt{-\ln(1 - \epsilon)}} \times \frac{1}{\epsilon}$$

$$\text{Complexity} = \frac{\sqrt{2 \ln\left(\frac{1}{1-\epsilon}\right)}}{\epsilon}$$

Therefore, the general formula for the complexity of the algorithm is:

$$\text{Complexity} = \sqrt{2 \ln\left(\frac{1}{1 - \epsilon}\right)} \times \sqrt{M}$$

This formula gives the complexity in terms of ϵ and M, where ϵ is the probability of failure and M is the size of the hash space.

4 Secure Hash Function

$h : \{0,1\}^* \to \{0,1\}^m$
h is a secure hash function.

- Preimage $\to O(2^m)$
- Collision $\to O(2^{m/2})$

5 Compression Function

$h : \{0,1\}^{m+t} \to \{0,1\}^m$, $t \geq 1$
h is a secure hash function.

- Preimage, 2nd preimage $\to O(2^m)$
- Collision $\to O(2^{m/2})$

6 Target: Construction of H

Construct $h : \{0,1\}^* \to \{0,1\}^m$ from h such that the security of H will completely depend on the security of h.

Given $x \in \{0,1\}^*$, $|x|$: Length of x, $|x| \geq M + T + 1$.

From x, construct by using a public function such that:

$$y = \begin{cases} x & \text{if } m_1 \equiv 0 \mod + \\ x110d & \text{if } |x| + d \equiv 0 \mod 7 \end{cases}$$

Select $IV \in \{0,1\}^*$ (IV is public).

$$Y = Y_1||Y_2||\dots||Y_r$$

such that $|Y_i| = t; 1 \leq i \leq r$.

H:

$$H = \begin{cases} Z_0 = y \\ Z_1 = h(Z_0||y_1) \\ Z_2 = h(Z_1||y_2) \\ \dots \\ Z_r = h(Z_{r-1}||y_r) \end{cases}$$

$Z_r = H(X)$ is known as an iterative hash function.

If h is pre-image resistant, then H is also pre-image resistant.

7 Merkle-Damgard

$h : \{0,1\}^* \to \{0,1\}^m$

Compress $h : \{0,1\}^{m+t} \to \{0,1\}^m$, $t \geq 2$

$$X = X_1||X_2||\dots||X_k$$

$|X_i| = t - 1$

$n = |X|$

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

For $i = 1$ to $k - 1$:

$$y_i = x_i$$

$$y_k = x_k||0^d$$

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

8 Secure Hash Functions

The Secure Hash Algorithm (SHA) is a family of cryptographic hash functions developed by the National Security Agency (NSA) and published by the National Institute of Standards and Technology (NIST). SHA is widely used in various cryptographic applications to ensure data integrity and authenticity.

8.0.1 Key Points:

- **SHA-1**: Produces a 160-bit hash value and is no longer considered secure.
- **SHA-2**: Includes SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224, and SHA-512/256, offering hash values of different lengths.
- **SHA-3 (Keccak)**: The latest member of the SHA family, selected through a public competition, providing security against various attacks.

For SHA-1:

$$|x| \leq 2^{64} - 1$$

$$X \& Y : \text{Bitwise AND Operation}$$

$$X|Y : \text{Bitwise OR Operation}$$

$$X \oplus Y : \text{Bitwise XOR Operation}$$

$$\sim X : \text{Bitwise Complement}$$

$$X + Y \equiv (X + Y) \mod 2^{32}$$

$$d = (447 - |x|) \mod 512$$

$$l = \text{binary}(|x|), \text{max of 64 bits}$$

$$y = x||1||0^d||l$$

$$|x| + d \equiv 447 \mod 512$$
$$|y| \equiv 0 \mod 512$$

$$\text{ROTLs}(x) = \text{circular left shift of } x \text{ by } s \text{ positions}$$

Define the function $F_t(b, c, d)$ as follows:

$$F_t(b,c,d) = \begin{cases} (b\&c)|(\sim b\&d) & \text{if } 0 \leq t \leq 19 \\ b \oplus c \oplus d & \text{if } 20 \leq t \leq 39 \\ (b\&c)|(b\&d)|(c\&d) & \text{if } 40 \leq t \leq 59 \\ b \oplus c \oplus d & \text{if } 60 \leq t \leq 79 \end{cases}$$

$$\text{SHA-1}(X) = \text{SHA-1-PAD}(A)$$

$$y = M_1||M_2||...||M_n, |M_i| = 512$$

Initial hash values:

$$H_0 = 67452301$$
$$H_1 = EFCDAB89$$
$$H_2 = 98B4D\angle FE$$
$$H_3 = 10325476$$
$$H_4 = CBDZEIFO$$

Constants:

$$K_t = \begin{cases} 5A827999 & \text{if } 0 \leq t \leq 19 \\ 6ED9EBA1 & \text{if } 20 \leq t \leq 39 \\ 8F1BBCDC & \text{if } 40 \leq t \leq 59 \\ CA62C1DC & \text{if } 60 \leq t \leq 79 \end{cases}$$

For $i = 1$ to n:

$$M_i = \omega_0||\omega_1||\omega_2||...||\omega_{15}, |\omega_i| = 32$$

For $t = 16$ to 79:

$$W_t = \text{ROTL}'(W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16}) \oplus W_{t-14} \oplus W_{t-16}$$

$$A = H_0, B = H_1, C = H_2, D = H_3, E = H_4$$

For $t = 0$ to 79:

$$\text{temp} = \text{ROTL}^5(A) + F_t + (BCD) + E + W_t + K_t$$
$$E = D, D = C, C = \text{ROTL}^{30}(B), B = A, A = \text{temp}$$

$$H_0 = H_0 + A$$
$$H_1 = H_1 + B$$
$$H_2 = H_2 + C$$
$$H_3 = H_3 + D$$
$$H_4 = H_4 + E$$

Return $(H_0, H_1, H_2, H_3, H_4)$

9 Message Authentication Code (MAC)

Message Authentication Code (MAC) is a cryptographic technique used to ensure the integrity and authenticity of a message. It involves generating a fixed-size digest (also known as a tag) based on the message content and a secret key. This tag is then appended to the message. Upon receiving the message, the recipient can recompute the MAC using the same key and verify if it matches the received MAC. If they match, it indicates that the message has not been tampered with and originates from a trusted source.

Alice $\xrightarrow{K}$ $c = \text{enc}(M, K) \xrightarrow{\sim C} c = \text{enc}(M, K)$

$\text{MAC} = \text{hash}(M, K) \xrightarrow{\sim \text{MAC}}$

$\text{dec}(\sim c, K) = \sim M$

$\text{hash}(\sim M, K) = \text{MAC}_1$

If $\text{MAC}_1 = \sim \text{MAC}$ as $\sim M = M$ accept $\sim M$

10 HMAC

HMAC (Hash-based Message Authentication Code) is a type of MAC that utilizes a cryptographic hash function along with a secret key to produce a message authentication code. It provides a way to verify both the data integrity and the authenticity of a message.

Given: ipad = 36 36...36 (512 bits), opad = $5c\,5c...5c$ (512 bits)

K is the secret key

$$\text{HMAC}_K(x) = \text{SHA-1}(K \oplus \text{opad})||\text{SHA-1}((K \oplus \text{ipad})||x)$$

11 CBC-MAC

CBC-MAC (Cipher Block Chaining Message Authentication Code) is a technique used to authenticate messages of arbitrary length. It involves encrypting the message using a block cipher in CBC mode, where the last block's output is used as the MAC. It provides integrity and authenticity to the message.

Let (x, K) denote the message x and the secret key K.

Let $x = x_1||...||x_n$.

The CBC-MAC process can be described as follows:

$$y_0 = \text{iv}$$

$$\text{for } i = 1 \text{ to } n:$$

$$y_i = \text{Enc}(y_{i-1} \oplus x_i, K)$$

$$\text{return } y_n$$