Cryptography Cheat Sheet

Isaac Piegat

December 5, 2024

Cryptographic Hash Functions (CRHFs)

Description

A hash function maps an input of arbitrary size to a fixed-size output, ensuring unique ”fingerprints” for data. A **CRHF** specifically ensures collision resistance.

Conditions

- **Collision Resistance:** Hard to find two inputs $x \neq x'$ such that $h(x) = h(x')$.
- **Second-Preimage Resistance:** Given x, hard to find $x' \neq x$ where $h(x) = h(x')$.
- **Preimage Resistance:** Given y, hard to find any x where $h(x) = y$.

Weaknesses/Common Attacks

- **Collision Attacks:** Exploiting weaknesses in the hash function to find two inputs with the same output (e.g., MD5, SHA-1).
- **Length Extension Attacks:** Extending valid hashes without knowing the original input.
- **Birthday Attacks:** Using the birthday paradox to find collisions faster than brute force.

One-Way Functions (OWFs)

Description

A function $f(x)$ that is easy to compute but computationally infeasible to invert. Used for securing data such as passwords.

Conditions

- **Easy to Compute:** $f(x)$ is efficient to evaluate.
- **Hard to Invert:** Given $y = f(x)$, it's computationally infeasible to find x.

Weaknesses/Common Attacks

- **Brute-Force Attacks:** If the input space is small, attackers can try all possibilities.
- **Weak Randomness:** Predictable inputs compromise security.
- **Structural Weaknesses:** Poorly designed OWFs can have shortcuts for inversion.

Hashes

Description

Hash functions compress data into fixed-size values for integrity, signatures, and authentication. Cryptographic hashes are a secure subset used in protocols.

Conditions

- **Fixed Output Size:** Maps any input to a specific-length output.
- **Deterministic:** Same input always produces the same output.
- **Avalanche Effect:** A small change in input drastically changes the output.

Weaknesses/Common Attacks

- **Collision Attacks:** Finding two inputs with the same hash.
- **Rainbow Table Attacks:** Precomputed hash chains to reverse hashes.
- **Timing Attacks:** Using timing information to guess the hash process or input.

Shared Key Protocols (SKPs)

Description

Protocols used to securely establish a shared secret key between two parties over an insecure channel. The key is then used for encryption or authentication.

Conditions

- **Authentication:** Each party verifies the identity of the other.
- **Confidentiality:** The key remains secret during exchange.
- **Integrity:** Messages exchanged during the protocol must not be altered.

Weaknesses/Common Attacks

- **Replay Attacks:** Reusing intercepted messages to impersonate one party.
- **Man-in-the-Middle (MitM) Attacks:** Intercepting and modifying communication.
- **Weak Nonces or Counters:** Predictable or reused values compromise security.
- **Reflection Attacks:** Tricking parties into processing their own messages.

Message Authentication Codes (MACs)

Description

A Message Authentication Code (MAC) is a cryptographic tool used to ensure the authenticity and integrity of a message. It uses a shared secret key K known only to the communicating parties.

How It Works

Given a message m and a shared key K:

$$\text{MAC} = \text{MAC}_K(m)$$

The recipient can verify that:

- m has not been tampered with (integrity).
- m originates from someone who knows K (authenticity).

Conditions for a Secure MAC

- **Key-Dependent:** The MAC must depend on the secret key K. Without K, an attacker should not be able to forge a valid MAC.
- **Unforgeability:** It must be computationally infeasible to generate a valid MAC without knowing K.
- **Resistance to Replay Attacks:** Often achieved by including a nonce or timestamp with the message.

Types of MACs

- **HMAC:** Hash-based MACs use a cryptographic hash function (e.g., SHA-256) combined with a secret key.
- **CBC-MAC:** Cipher Block Chaining MACs are based on block ciphers and operate on fixed-size message blocks.
- **GMAC:** A MAC based on Galois Field arithmetic, used in authenticated encryption schemes.

Applications

- Authenticating messages in communication protocols (e.g., SSL/TLS, IPsec).
- Ensuring data integrity in storage systems.
- Preventing tampering in distributed systems.

Weaknesses/Common Attacks

- **Key Compromise:** If K is leaked, the attacker can forge valid MACs.
- **Length Extension Attacks:** A poorly implemented MAC based on hash functions can be vulnerable if intermediate states of the hash are exposed.
- **Reused Nonces:** If a MAC includes a nonce that is reused, it can allow replay attacks.

Combining Encryption and Authentication

There are three primary methods to combine encryption and authentication in cryptographic protocols:

Encrypt-then-MAC (EtM)

- **How It Works:** First, the plaintext is encrypted to produce the ciphertext. Then, a MAC is calculated over the ciphertext using a secret key.
- **Order of Operations:**

$$\text{Ciphertext} = E_k(\text{plaintext})$$

$$\text{MAC} = \text{MAC}_k(\text{Ciphertext})$$

- **Advantages:**
  - Provides strong security.
  - Detects tampering with the ciphertext before decryption.
- **Example Use Case:** IPsec.

MAC-then-Encrypt (MtE)

- **How It Works:** First, a MAC is computed on the plaintext. Then, the plaintext and its MAC are encrypted together.
- **Order of Operations:**

$$\text{MAC} = \text{MAC}_k(\text{plaintext})$$

$$\text{Ciphertext} = E_k(\text{plaintext}\|\text{MAC})$$

- **Advantages:**
  - Protects both the plaintext and the MAC.
- **Disadvantages:**
  - Vulnerable if decryption is performed before verifying the MAC.
- **Example Use Case:** SSL/TLS (pre-TLS 1.3).

Encrypt-and-MAC (E&M)

- **How It Works:** The plaintext is both encrypted and authenticated independently.
- **Order of Operations:**

$$\text{Ciphertext} = E_k(\text{plaintext})$$

$$\text{MAC} = \text{MAC}_k(\text{plaintext})$$

- **Advantages:**
  - Simpler conceptually; failures in one process (encryption or MAC) do not compromise the other.
- **Disadvantages:**
  - Less efficient because encryption and authentication are independent.
- **Example Use Case:** Rarely used in modern protocols.

MACs vs. Hash Functions

Message Authentication Codes (MACs)

- **Description:** A MAC binds a message to a shared secret key K, providing integrity and authenticity.
- **Purpose:** Ensures that the message is from an authenticated sender and has not been tampered with.

Hash Functions

- **Description:** A hash function produces a fixed-length digest of a message but does not use a secret key.
- **Limitation:** Hash functions do not provide authenticity; anyone can compute a valid hash for a modified message.

Why MACs Are Superior

- **Key Binding:** A MAC ensures that the message is bound to the shared secret K, preventing forgery.
- **Tamper Resistance:** An attacker cannot recompute a valid MAC without knowing K.
- **Man-in-the-Middle Defense:** MACs resist MitM attacks, where a hash function might fail due to lack of key-dependency.

Modular Arithmetic

Basic Operations

Modular arithmetic operates within the range $\{0, 1, \ldots, n-1\}$ for a modulus n. The key operations include:

Addition Modulo n

$$(a + b) \mod n = ((a \mod n) + (b \mod n)) \mod n$$

Multiplication Modulo n

$$(a \cdot b) \mod n = ((a \mod n) \cdot (b \mod n)) \mod n$$

Subtraction Modulo n

$$(a - b) \mod n = ((a \mod n) - (b \mod n)) \mod n$$

Distributive Property

For any integers a, b, c:

$$(a \cdot (b + c)) \mod n = ((a \cdot b) + (a \cdot c)) \mod n$$

Fermat's Little Theorem

Statement: If p is a prime number and a is an integer such that $p \not| a$, then:

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

Euler's Totient Function

Definition

The totient function $\phi(n)$ counts the integers between 1 and n that are coprime to n.

Formula for $\phi(n)$

For $n = p_1^{e_1} \cdot p_2^{e_2} \cdot \cdots \cdot p_k^{e_k}$:

$$\phi(n) = n \cdot \prod_{i=1}^{k} \left(1 - \frac{1}{p_i}\right)$$

RSA Key Generation

Steps to Generate Keys

1. Select two large prime numbers p and q. 2. Compute $n = p \cdot q$. 3. Compute $\phi(n) = (p-1) \cdot (q-1)$. 4. Choose e such that $1 < e < \phi(n)$ and $\gcd(e, \phi(n)) = 1$. 5. Compute d such that:

$$e \cdot d \equiv 1 \mod \phi(n)$$

6. Public key: (e, n). Private key: d.

Encryption and Decryption

- **Encryption:** Given message m:

$$c \equiv m^e \mod n$$

- **Decryption:** Recover m from c:

$$m \equiv c^d \mod n$$