**Experiment 10**

**Shashwat Tripathi**

**D20A 64**

**Aim:** Understanding the concept of cryptography and guidelines for using encryption.

**Theory:**

**What is Cryptography?**

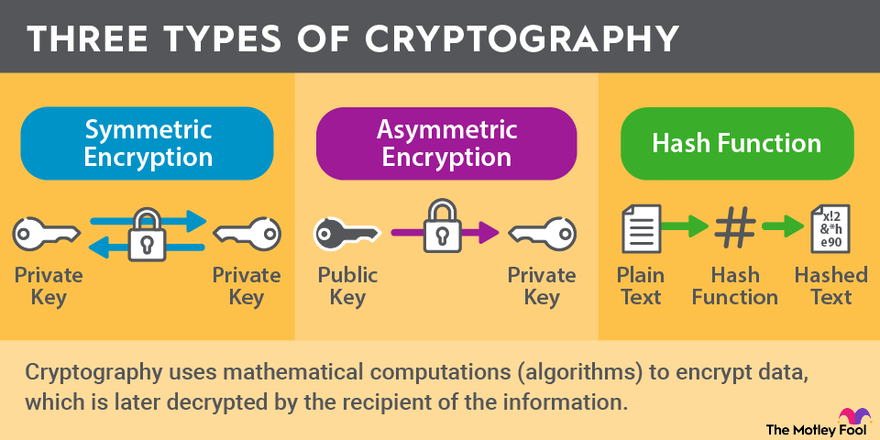
**Cryptography** is the practice of securing information by converting it into an unreadable format, known as **encryption**, to prevent unauthorized access. The purpose of cryptography is to ensure the confidentiality, integrity, authenticity, and non-repudiation of data, whether it is being stored (data at rest) or transmitted (data in transit). By using cryptographic techniques, sensitive data such as passwords, personal details, financial transactions, and communications are protected from attackers and adversaries.

Key concepts in cryptography include:

1. **Plaintext and Ciphertext**:
   * **Plaintext** is the original readable message or data.
   * **Ciphertext** is the encrypted, unreadable version of the plaintext after applying a cryptographic algorithm. The conversion from plaintext to ciphertext is encryption, and reversing this process (turning ciphertext back into plaintext) is decryption.
2. **Keys**:
   * **Keys** are essential in cryptography. They are used to encrypt and decrypt data. A key is a random string of bits used by an encryption algorithm to transform the plaintext into ciphertext or vice versa. Without the correct key, it is nearly impossible to decrypt the ciphertext.
3. **Cryptographic Algorithms**:
   * These are mathematical functions or protocols that are used to perform encryption and decryption. The strength of cryptography largely depends on the robustness of the cryptographic algorithms and the secrecy of the keys.
4. **Basic Objectives of Cryptography**:
   * **Confidentiality**: Ensuring that information is only accessible to those who are authorized.
   * **Integrity**: Ensuring that the information has not been altered or tampered with during transmission or storage.
   * **Authentication**: Verifying the identity of the entities involved in the communication.
   * **Non-repudiation**: Preventing the sender or receiver from denying their involvement in a communication or transaction.

Cryptography is widely applied in various areas such as secure communications (email, messaging), online transactions (e-commerce), protecting sensitive information (personal data, medical records), and ensuring data integrity (digital signatures, hashing).

**Types of cryptography and demonstration**



There are three main types of cryptography based on the key types used for encryption and decryption:

#### **1. Symmetric Key Cryptography (Secret Key Cryptography)**

**Symmetric key cryptography** is a type of cryptography in which both the sender and the receiver use the same key for encryption and decryption. This method is also known as **private key cryptography**. It is faster and simpler than asymmetric cryptography but requires that the key be securely shared between the communicating parties.

* **Example Algorithm**: AES (Advanced Encryption Standard), DES (Data Encryption Standard), and Triple DES.
* **Working Demonstration**:
  1. **Sender**: The sender encrypts the plaintext using a symmetric algorithm (like AES) and a secret key.
  2. **Receiver**: The receiver decrypts the ciphertext using the same secret key to retrieve the original plaintext.

Key: abc123

Plaintext: "Hello World"

Ciphertext (after encryption): "XUO!2@\*OLp"

* **Advantages**:
  + Fast encryption and decryption.
  + Efficient for large amounts of data.
* **Disadvantages**:
  + Secure key distribution is difficult since the same key must be shared by both parties.
  + If the key is compromised, the entire system becomes vulnerable.

#### **2. Asymmetric Key Cryptography (Public Key Cryptography)**

In **asymmetric key cryptography**, two keys are used: a **public key** for encryption and a **private key** for decryption. The public key is shared openly, while the private key is kept secret. It is more secure than symmetric key cryptography, especially for exchanging data between two parties who have never met, but it is computationally slower.

* **Example Algorithms**: RSA (Rivest-Shamir-Adleman), ECC (Elliptic Curve Cryptography), DSA (Digital Signature Algorithm).
* **Working Demonstration**:
  1. **Sender**: The sender encrypts the message using the receiver’s **public key**.
  2. **Receiver**: The receiver decrypts the message using their own **private key**.

Public Key: XYZ789

Plaintext: "Hello World"

Ciphertext (after encryption): "A84%R2i#lm"

Private Key: PQR456 (used for decryption to get the original message)

* **Advantages**:
  + No need to share a secret key between the sender and receiver.
  + The public key can be shared openly without compromising security.
* **Disadvantages**:
  + Slower than symmetric key cryptography due to complex mathematical operations.
  + More computationally expensive.

#### **3. Hash Functions**

**Hashing** is a one-way cryptographic operation that converts data into a fixed-size string of characters, which is typically a digest that represents the data. Hash functions are primarily used for ensuring data integrity. They are not used for encryption or decryption but to verify whether data has been altered. A small change in the input will produce a completely different hash value.

* **Example Algorithms**: SHA (Secure Hash Algorithm), MD5 (Message Digest Algorithm), Bcrypt, and Argon2.
* **Working Demonstration**:
  1. **Sender**: The sender applies a hash function to the data to generate a hash (digest).
  2. **Receiver**: The receiver also hashes the received data and compares it to the original hash value to ensure integrity.

Plaintext: "Hello World"

Hash (SHA-256): "a591a6d40bf420404a011733cfb7b190d62c65bf0bcda32b23ea517ad7c7a2a1"

**Advantages**:

* Quick and efficient for verifying data integrity.
* Hash functions are deterministic: the same input will always result in the same hash.

**Disadvantages**:

* Hashing is not reversible (you cannot retrieve the original data from a hash).
* Weak algorithms like MD5 are vulnerable to attacks like collisions.

**What are the Cryptographic best practices according to OWASP?**

The **Open Web Application Security Project (OWASP)** provides a set of cryptographic best practices to ensure secure implementation of cryptographic mechanisms in applications. These best practices mitigate risks such as data breaches, compromised communications, and security flaws arising from improper cryptography.

Here are some key cryptographic best practices recommended by OWASP:

#### **1. Use Strong Cryptographic Algorithms**

* Always use well-known and thoroughly vetted algorithms such as **AES**, **RSA**, **ECC**, and **SHA-256**.
* Avoid outdated or broken algorithms like **MD5** and **DES** which are vulnerable to attacks such as collision attacks or brute-force decryption.

#### **2. Secure Key Management**

* **Key generation**: Keys must be generated using secure random number generators. Do not use weak, predictable keys.
* **Key storage**: Store cryptographic keys securely, either in hardware security modules (HSMs) or through robust software key management solutions. Do not store keys in the same location as the encrypted data.
* **Key rotation**: Periodically rotate cryptographic keys to minimize the impact of a key compromise.
* **Key sharing**: Use secure key exchange protocols, such as **Diffie-Hellman** or **Elliptic Curve Diffie-Hellman** (ECDH), to safely share keys over insecure networks.

#### **3. Use Strong Hashing Algorithms**

* Use strong cryptographic hashing functions such as **SHA-256** and **SHA-3** for data integrity checks and password hashing.
* When hashing passwords, always use a **salt** (a random value added to the password before hashing) and apply hashing algorithms specifically designed for passwords, such as **Bcrypt**, **Scrypt**, or **Argon2**. This ensures resistance to brute-force and rainbow table attacks.

#### **4. Encrypt Sensitive Data at Rest and in Transit**

* Always encrypt sensitive data, both at rest and in transit, using strong encryption algorithms.
* For data in transit, use **TLS (Transport Layer Security)** to secure communication between clients and servers.
* For data at rest, use full-disk encryption or database-level encryption to protect sensitive information from unauthorized access.

#### **5. Avoid Implementing Custom Cryptography**

* Do not attempt to write custom encryption algorithms or implement your own cryptographic protocols. Instead, rely on well-established libraries like **OpenSSL**, **Bouncy Castle**, and **libsodium**.
* Custom cryptographic solutions often contain vulnerabilities that can be easily exploited by attackers.

#### **6. Implement Secure Random Number Generation**

* Use secure random number generators for generating keys, nonces, salts, and initialization vectors (IVs). Cryptographically secure functions such as java.security.SecureRandom or random.SystemRandom() in Python should be used instead of simpler, predictable generators.

#### **7. Ensure Proper Use of Cryptographic Modes**

* When using block ciphers like AES, avoid using **ECB (Electronic Codebook)** mode, which is insecure. Instead, use secure modes such as **CBC (Cipher Block Chaining)** with proper padding or **GCM (Galois/Counter Mode)**, which provides both encryption and integrity.
* Ensure the correct use of **IVs** and nonces in encryption algorithms. They should be unique and unpredictable to ensure the same plaintext always results in different ciphertext.

#### **8. Secure Authentication and Integrity**

* Use cryptographic techniques like **HMAC (Hash-based Message Authentication Code)** to ensure message integrity and authentication.
* Always validate digital certificates and avoid trusting invalid or self-signed certificates.

**Conclusion:**

Cryptography is an essential tool in securing digital communication and data in today’s interconnected world. By understanding the types of cryptographic systems—symmetric, asymmetric, and hash functions—and applying cryptographic best practices recommended by OWASP, developers can ensure the confidentiality, integrity, and authenticity of sensitive information.