UTT

UNIVERSIDAD TECNOLÓGICA DE TIJUANA

GOBIERNO DE BAJA CALIFORNIA

TEMA:

DATA ENCRYPTION MECHANISMS IN MOBILE APPLICATIONS

PRESENTADO POR:

Valenzuela Esparza Javier Ivan

GRUPO:

10B

MATERIA:

Desarrollo Movil Integral

PROFESOR:

Ray Brunett Parra Galaviz

FECHA:

23/01/2025

DATA ENCRYPTION MECHANISMS IN MOBILE APPLICATIONS

Data encryption mechanisms in mobile applications are essential for protecting users' sensitive information and preventing unauthorized access. Some of the most commonly used encryption mechanisms in mobile apps include:

1. Data-at-Rest Encryption

This encryption protects data stored on the device, such as local databases, files, and user preferences.

- **AES (Advanced Encryption Standard):** One of the most secure and widely used algorithms to protect data-at-rest. AES with a 256-bit key length is highly recommended.
- **Keychain (iOS) / Keystore (Android):** Both platforms provide secure services for storing encryption keys. In iOS, the Keychain stores passwords and other sensitive data. In Android, the Keystore provides secure storage for encryption keys.

2. Data-in-Transit Encryption

This encryption protects data while it is moving, i.e., when it is transmitted between the mobile device and the server.

- **TLS (Transport Layer Security):** The most commonly used protocol for encrypting communication between the client (mobile device) and the server. TLS ensures data is not intercepted during transmission.
- **HTTPS:** Using TLS, mobile applications can make secure network requests over HTTPS, ensuring communications with APIs and servers are protected.

3. Local Database Encryption

Mobile apps may store sensitive data locally, such as user information or configurations. To protect this data, database encryption is critical.

- **SQLCipher:** An extension for SQLite that adds AES-256 encryption to SQLite databases.
- **Realm:** A mobile database engine that supports encryption of data-at-rest.

4. File and Document Encryption

If an application handles confidential documents or files, appropriate encryption should be applied to protect the information stored within those files.

- **AES:** Like data-at-rest encryption, AES is ideal for encrypting files due to its efficiency and security.

5. Authentication and Authorization

In addition to encrypting data, mobile applications must secure access through robust authentication mechanisms.

- **Biometric Authentication:** Using fingerprint or facial recognition to authenticate the user.
- **OAuth 2.0 / OpenID Connect:** Authorization protocols used to securely access server resources without exposing user credentials.
- **MFA (Multifactor Authentication):** Enhances app security by requiring a second form of authentication, such as a code sent via SMS or an authentication app.

6. Key Protection

The security of the encryption key is critical. If a key is compromised, encryption is ineffective.

- **Hardware Security Module (HSM):** Used in high-performance devices or servers to protect cryptographic keys.
- **Secure Enclaves (iOS) / Trusted Execution Environment (TEE) (Android):** Provide secure environments within the device's hardware to generate and store cryptographic keys without them being accessible through software.

7. Password Encryption

User passwords should be stored securely to prevent compromise.

- **PBKDF2 (Password-Based Key Derivation Function 2):** Used to derive a key from a password. It enhances security by applying multiple rounds of encryption.
- **bcrypt:** Another password hashing algorithm that is resistant to brute-force attacks.

8. Protection Against Attacks

- **Code Obfuscation:** Makes reverse engineering of the application code difficult, helping to prevent key or sensitive data extraction.
- **Traffic Analysis:** It is recommended to analyze network traffic using tools like Wireshark or Fiddler to detect potential vulnerabilities in encryption.

Bibliographical Sources.

Security. (s/f). Apple Developer Documentation. Recuperado el 24 de enero de 2025, de https://developer.apple.com/documentation/security

Security. (s/f-b). Android Developers. Recuperado el 24 de enero de 2025, de https://developer.android.com/security