Ingeniería En Desarrollo Y Gestión De Software

Nombre:

García Arreola Howard Isai

Subject:

Desarrollo movil Integral

Actividad:

How and When to Cipher in Mobile

Grupo: 10-B

Profesor:

Ray Brunett Parra

Galaviz

Fecha de Realización:

January 28th, 2025

How and When to Use Ciphering in Mobile

El cifrado es una técnica esencial en el desarrollo móvil para proteger la información sensible de los usuarios y garantizar la integridad y confidencialidad de los datos. Consiste en convertir datos legibles en un formato codificado, accesible solo para aquellos con la clave de descifrado adecuada. Su implementación es crucial tanto para datos en tránsito como para datos en reposo.

- **Datos en tránsito** se refiere a la información que se transmite entre dispositivos o a través de redes, como cuando un usuario envía datos desde una aplicación móvil a un servidor. Es fundamental cifrar estos datos para prevenir interceptaciones por parte de terceros no autorizados. El uso de protocolos como HTTPS asegura que la comunicación entre la aplicación y el servidor esté cifrada, protegiendo la información durante su transferencia.

- **Datos en reposo** son aquellos almacenados en el dispositivo móvil, como bases de datos locales, archivos o preferencias de la aplicación. Cifrar estos datos es vital para protegerlos en caso de que el dispositivo sea perdido o robado. Tanto Android como iOS ofrecen mecanismos integrados para el cifrado de datos en reposo. Por ejemplo, Android proporciona opciones para cifrar el dispositivo completo, asegurando que los archivos y carpetas estén protegidos contra accesos no autorizados.

Al desarrollar aplicaciones móviles, es importante considerar qué datos requieren cifrado y en qué momentos. Información sensible como credenciales de usuario, datos personales, información financiera y cualquier otro dato que pueda comprometer la privacidad del usuario debe ser cifrada tanto en tránsito como en reposo. Además, es recomendable utilizar algoritmos de cifrado robustos y actualizados.

Uso en Jetpack Compose

En Jetpack Compose, el cifrado se puede implementar aprovechando las APIs de Android para gestionar la seguridad de los datos. A continuación, te explico cómo aplicar cifrado en datos en reposo y datos en tránsito dentro de una aplicación Jetpack Compose.

1. Cifrado de Datos en Reposo (Almacenados en el Dispositivo)

Para cifrar datos en almacenamiento local, se puede usar Android Keystore junto con Cipher para encriptar y desencriptar información.

```
object CryptoHelper {
    private const val ALIAS = "MySecretKey"

    fun generateKey() {
        val keyGenerator = KeyGenerator.getInstance(KeyProperties.KEY_ALGORITHM_AES, "AndroidKeyStore")
        val keyGenParameterSpec = KeyGenParameterSpec.Builder(
            ALIAS,
            KeyProperties.PURPOSE_ENCRYPT or KeyProperties.PURPOSE_DECRYPT
        ).setBlockModes(KeyProperties.BLOCK_MODE_CBC)
            .setEncryptionPaddings(KeyProperties.ENCRYPTION_PADDING_PKCS7)
            .build()
        keyGenerator.init(keyGenParameterSpec)
        keyGenerator.generateKey()
    }

    fun getSecretKey(): SecretKey {
        val keyStore = KeyStore.getInstance("AndroidKeyStore").apply { load(null) }
        val keyEntry = keyStore.getEntry(ALIAS, null) as KeyStore.SecretKeyEntry
        return keyEntry.secretKey
    }

    fun encryptData(plainText: String): Pair<ByteArray, ByteArray> {
        val cipher = Cipher.getInstance("AES/CBC/PKCS7Padding")
        cipher.init(Cipher.ENCRYPT_MODE, getSecretKey())
```

```
        val iv = cipher.iv
        val encryptedData = cipher.doFinal(plainText.toByteArray(Charsets.UTF_8))
        return Pair(encryptedData, iv)
    }

    fun decryptData(encryptedData: ByteArray, iv: ByteArray): String {
        val cipher = Cipher.getInstance("AES/CBC/PKCS7Padding")
        cipher.init(Cipher.DECRYPT_MODE, getSecretKey(), IvParameterSpec(iv))
        val decryptedData = cipher.doFinal(encryptedData)
        return String(decryptedData, Charsets.UTF_8)
    }
}
```

2. Cifrado de Datos en Tránsito (Comunicación con Servidor)

Para garantizar la seguridad en la comunicación entre la app y un servidor, se debe de utilizar TLS/SSL en las conexiones HTTP. Esto se logra al usar Retrofit con OkHttp asegurando que la API solo funcione con HTTPS.

```
private val client = OkHttpClient.Builder()
    .connectTimeout(30, TimeUnit.SECONDS)
    .readTimeout(30, TimeUnit.SECONDS)
    .writeTimeout(30, TimeUnit.SECONDS)
    .build()

private val retrofit = Retrofit.Builder()
    .baseUrl("https://tuapi.com/") // Asegurar que es HTTPS
    .client(client)
    .addConverterFactory(GsonConverterFactory.create())
    .build()

interface ApiService {
    @GET("datos")
    suspend fun getSecureData(): Response<List<String>>
}

val apiService: ApiService = retrofit.create(ApiService::class.java)
```

References

Criptografía. (n.d.). Retrieved from https://developer.android.com/privacy-and-security/cryptography

Guía para Android Application Security | Blog | Digital.ai. (n.d.). Retrieved from https://digital.ai/es/catalyst-blog/android-app-security/#Tools-and-Technologies-for-Enhancing-Security