GIE Technical Document

1. Introduction

English

This document provides a technical overview of GIE (Go-based Information Encryptor), a cross-platform desktop application built using the Wails framework. GIE is designed for secure encryption and decryption of files and directories, leveraging robust cryptographic primitives implemented in Go. The application aims to provide a user-friendly interface for managing sensitive data.

Spanish

Este documento proporciona una visión técnica de GIE (Go-based Information Encryptor), una aplicación de escritorio multiplataforma construida utilizando el framework Wails. GIE está diseñada para el cifrado y descifrado seguro de archivos y directorios, aprovechando primitivas criptográficas robustas implementadas en Go. La aplicación tiene como objetivo proporcionar una interfaz fácil de usar para gestionar datos sensibles.

2. Architecture

English

GIE follows a client-server architecture facilitated by the Wails framework. * **Frontend (Client):** Developed with React and TypeScript, responsible for the user interface and interaction. It communicates with the Go backend via Wails' inter-process communication (IPC) mechanisms. * **Backend (Server):** Implemented in Go, handling all core logic, including file operations, cryptographic processes, and system interactions.

Spanish

GIE sigue una arquitectura cliente-servidor facilitada por el framework Wails. * **Frontend (Cliente):** Desarrollado con React y TypeScript, responsable de la interfaz de usuario y la interacción. Se comunica con el backend de Go a través de los mecanismos de comunicación entre procesos (IPC) de Wails. * **Backend (Servidor):** Implementado en Go, maneja toda la lógica central, incluyendo operaciones de archivo, procesos criptográficos e interacciones con el sistema.

3. Cryptographic Design

English

GIE employs a hybrid encryption scheme combining symmetric encryption (AES-CTR) for data confidentiality and a Message Authentication Code (HMAC-SHA256) for data integrity and authenticity. Key derivation is performed using

PBKDF2.

Spanish

GIE emplea un esquema de cifrado híbrido que combina cifrado simétrico (AES-CTR) para la confidencialidad de los datos y un Código de Autenticación de Mensajes (HMAC-SHA256) para la integridad y autenticidad de los datos. La derivación de claves se realiza utilizando PBKDF2.

3.1. Key Derivation (PBKDF2)

English

The encryption and HMAC keys are derived from the user's password using PBKDF2 (Password-Based Key Derivation Function 2) with SHA256 as the pseudorandom function. Separate salts are used for the AES key and the HMAC key to prevent cross-protocol attacks. The number of iterations and key length are configurable based on the chosen encryption level (Low, Normal, High).

The PBKDF2 function is defined as:

$$DK = PBKDF2(PRF, Password, Salt, Iterations, KeyLength)$$

Where: * DK: Derived Key * PRF: Pseudorandom Function (HMAC-SHA256 in this case) * $Password$: User-provided password * $Salt$: Random salt (16 bytes) * $Iterations$: Number of iterations (e.g., 10,000 for Low, 800,000 for Normal, 12,000,000 for High) * $KeyLength$: Desired length of the derived key (16 bytes for AES-128, 32 bytes for AES-256)

Spanish

Las claves de cifrado y HMAC se derivan de la contraseña del usuario utilizando PBKDF2 (Password-Based Key Derivation Function 2) con SHA256 como función pseudoaleatoria. Se utilizan sales separadas para la clave AES y la clave HMAC para prevenir ataques entre protocolos. El número de iteraciones y la longitud de la clave son configurables según el nivel de cifrado elegido (Bajo, Normal, Alto).

La función PBKDF2 se define como:

$$DK = PBKDF2(PRF, Password, Salt, Iterations, KeyLength)$$

Donde: * DK: Clave Derivada * PRF: Función Pseudoaleatoria (HMAC-SHA256 en este caso) * $Password$: Contraseña proporcionada por el usuario * $Salt$: Sal aleatoria (16 bytes) * $Iterations$: Número de iteraciones (ej., 10,000 para Bajo, 800,000 para Normal, 12,000,000 para Alto) * $KeyLength$: Longitud deseada de la clave derivada (16 bytes para AES-128, 32 bytes para AES-256)

3.2. Symmetric Encryption (AES-CTR)

English

AES in Counter (CTR) mode is used for encrypting the file content. CTR mode transforms a block cipher into a stream cipher, allowing for parallel encryption/decryption and direct access to any part of the ciphertext. A unique, randomly generated 16-byte Initialization Vector (IV) is used for each encryption operation.

The CTR mode encryption process can be conceptually represented as:

$$C_i = E_K(Nonce||Counter_i) \oplus P_i$$

Where: * C_i: i-th ciphertext block * E_K: AES encryption function with key K * $Nonce$: A unique value for each encryption (part of the IV) * $Counter_i$: A counter that increments for each block * P_i: i-th plaintext block * $\oplus$: Bitwise XOR operation

Spanish

AES en modo Contador (CTR) se utiliza para cifrar el contenido del archivo. El modo CTR transforma un cifrado por bloques en un cifrado de flujo, permitiendo el cifrado/descifrado paralelo y el acceso directo a cualquier parte del texto cifrado. Se utiliza un Vector de Inicialización (IV) único y generado aleatoriamente de 16 bytes para cada operación de cifrado.

El proceso de cifrado en modo CTR se puede representar conceptualmente como:

$$C_i = E_K(Nonce||Counter_i) \oplus P_i$$

Donde: * C_i: i-ésimo bloque de texto cifrado * E_K: Función de cifrado AES con clave K * $Nonce$: Un valor único para cada cifrado (parte del IV) * $Counter_i$: Un contador que se incrementa para cada bloque * P_i: i-ésimo bloque de texto plano * $\oplus$: Operación XOR a nivel de bits

3.3. Message Authentication Code (HMAC-SHA256)

English

HMAC-SHA256 is used to ensure the integrity and authenticity of the encrypted data and its associated metadata. The HMAC is calculated over the entire encrypted file, including the metadata header (hint length, hint, channel, encryption level code, AES salt, HMAC salt, CTR IV) and the ciphertext. This prevents tampering with both the data and its critical parameters.

The HMAC function is defined as:

$$HMAC_K(m) = H((K \oplus opad)||H((K \oplus ipad)||m))$$

Where: * H: Hash function (SHA256 in this case) * K: Secret key (derived HMAC key) * m: Message (metadata || ciphertext) * $ipad$: Inner padding (0x36 repeated) * $opad$: Outer padding (0x5C repeated) * ||: Concatenation

Spanish

HMAC-SHA256 se utiliza para garantizar la integridad y autenticidad de los datos cifrados y sus metadatos asociados. El HMAC se calcula sobre todo el archivo cifrado, incluyendo la cabecera de metadatos (longitud de la pista, pista, canal, código de nivel de cifrado, sal AES, sal HMAC, IV CTR) y el texto cifrado. Esto evita la manipulación tanto de los datos como de sus parámetros críticos.

La función HMAC se define como:

$$HMAC_K(m) = H((K \oplus opad)||H((K \oplus ipad)||m))$$

Donde: * H: Función hash (SHA256 en este caso) * K: Clave secreta (clave HMAC derivada) * m: Mensaje (metadatos || texto cifrado) * *ipad*: Relleno interno (0x36 repetido) * *opad*: Relleno externo (0x5C repetido) * ||: Concatenación

4. File Format (`.gie`)

English

The `.gie` file format is structured as follows:

Field	Size (Bytes)	Description
Hint Length	2	`uint16`, length of the hint in bytes
Hint	`HintLength`	User-provided hint for the file
Channel	2	`uint16`, a user-defined channel for grouping files
Encryption Level Code	1	`byte`, code representing the encryption strength (0=Low, 1=Normal, 2=High)
AES Key Salt	16	Random salt for AES key derivation
HMAC Key Salt	16	Random salt for HMAC key derivation
CTR IV	16	Random Initialization Vector for AES-CTR
Ciphertext	Variable	Encrypted content of the original file
HMAC Tag	32	SHA256 HMAC of all preceding data (metadata + ciphertext)

Spanish

El formato de archivo `.gie` se estructura de la siguiente manera:

Campo	Tamaño (Bytes)	Descripción
Longitud de la Pista	2	`uint16`, longitud de la pista en bytes

Campo	Tamaño (Bytes)	Descripción
Pista	`LongitudPista`	Pista proporcionada por el usuario para el archivo
Canal	2	`uint16`, un canal definido por el usuario para agrupar archivos
Código Nivel Cifrado	1	`byte`, código que representa la fuerza del cifrado (0=Bajo, 1=Normal, 2=Alto)
Sal Clave AES	16	Sal aleatoria para la derivación de la clave AES
Sal Clave HMAC	16	Sal aleatoria para la derivación de la clave HMAC
IV CTR	16	Vector de Inicialización aleatorio para AES-CTR
Texto Cifrado	Variable	Contenido cifrado del archivo original
Etiqueta HMAC	32	HMAC-SHA256 de todos los datos precedentes (metadatos + texto cifrado)

5. Encryption Process Flow

English

1. **Input:** Original file path, password, hint, encryption level, channel.
2. **Salt Generation:** Generate unique 16-byte salts for AES key and HMAC key.
3. **IV Generation:** Generate a unique 16-byte CTR IV.
4. **Key Derivation:** Derive AES key and HMAC key from the password using PBKDF2 with respective salts and iterations.
5. **Metadata Assembly:** Assemble the metadata header: Hint Length, Hint, Channel, Encryption Level Code, AES Key Salt, HMAC Key Salt, CTR IV.
6. **Temporary Output File:** Create a temporary output file (`.gie.tmp`).
7. **HMAC Initialization:** Initialize HMAC-SHA256 with the derived HMAC key.
8. **Write Metadata & Feed HMAC:** Write the metadata header to the temporary file and simultaneously feed it to the HMAC hasher.
9. **AES-CTR Initialization:** Initialize AES-CTR stream cipher with the derived AES key and CTR IV.
10. **Encrypt & Write Data:** Read the input file in chunks, encrypt each chunk using AES-CTR, write the encrypted chunk to the temporary file, and simultaneously feed it to the HMAC hasher.
11. **Finalize HMAC:** Compute the final HMAC tag.
12. **Write HMAC Tag:** Append the HMAC tag to the temporary file.
13. **Verification:** Attempt to decrypt the temporary file to verify integrity and correctness. If verification fails, the temporary file is deleted.
14. **Rename:** If verification succeeds, rename the temporary file to the final `.gie` file, and attempt to securely delete the original file.

Spanish

1. **Entrada:** Ruta del archivo original, contraseña, pista, nivel de cifrado, canal.
2. **Generación de Sales:** Generar sales únicas de 16 bytes para la clave AES y la clave HMAC.
3. **Generación de IV:** Generar un IV CTR único de 16 bytes.
4. **Derivación de Claves:** Derivar la clave AES y la clave HMAC de la contraseña usando PBKDF2 con sus respectivas sales e iteraciones.
5. **Ensamblaje de Metadatos:** Ensamblar la cabecera de metadatos: Longitud de la Pista, Pista, Canal, Código de Nivel de Cifrado, Sal de la Clave AES, Sal de la Clave HMAC, IV CTR.
6. **Archivo de Salida Temporal:** Crear un archivo de salida temporal (`.gie.tmp`).
7. **Inicialización de HMAC:** Inicializar HMAC-SHA256 con la clave HMAC derivada.
8. **Escribir Metadatos y Alimentar HMAC:** Escribir la cabecera de metadatos en el archivo temporal y simultáneamente alimentarla al hasher HMAC.
9. **Inicialización de AES-CTR:** Inicializar el cifrado de flujo AES-CTR con la clave AES derivada y el IV CTR.
10. **Cifrar y Escribir Datos:** Leer el archivo de entrada en bloques, cifrar cada bloque usando AES-CTR, escribir el bloque cifrado en el archivo temporal y simultáneamente alimentarlo al hasher HMAC.
11. **Finalizar HMAC:** Calcular la etiqueta HMAC final.
12. **Escribir Etiqueta HMAC:** Añadir la etiqueta HMAC al archivo temporal.
13. **Verificación:** Intentar descifrar el archivo temporal para verificar la integridad y corrección. Si la verificación falla, el archivo temporal se elimina.
14. **Renombrar:** Si la verificación tiene éxito, renombrar el archivo temporal al archivo `.gie` final e intentar eliminar de forma segura el archivo original.

6. Decryption Process Flow

English

1. **Input:** Encrypted `.gie` file path, password, expected channel.
2. **Read Metadata:** Read the metadata header (Hint Length, Hint, Channel, Encryption Level Code, AES Key Salt, HMAC Key Salt, CTR IV) from the `.gie` file.
3. **Channel Verification:** Verify if the `Channel` in the file matches the `expectedChannel` provided by the user.
4. **Key Derivation:** Derive AES key and HMAC key from the password using PBKDF2 with the salts read from the file.
5. **HMAC Initialization:** Initialize HMAC-SHA256 with the derived HMAC key.

6. **Feed Metadata & Ciphertext to HMAC:** Feed the metadata header and the entire ciphertext (excluding the final HMAC tag) from the `.gie` file to the HMAC hasher.
7. **Read Stored HMAC Tag:** Read the 32-byte HMAC tag from the end of the `.gie` file.
8. **HMAC Verification:** Compare the computed HMAC with the stored HMAC tag. If they do not match, the file is considered corrupted or the password incorrect, and decryption is aborted.
9. **AES-CTR Initialization:** If HMAC verification passes, initialize AES-CTR stream cipher with the derived AES key and CTR IV.
10. **Temporary Output File:** Create a temporary output file (`.tmp`).
11. **Decrypt & Write Data:** Read the ciphertext in chunks, decrypt each chunk using AES-CTR, and write the decrypted chunk to the temporary output file.
12. **Rename:** Rename the temporary file to the original file name (by removing the `.gie` extension), and attempt to securely delete the original `.gie` file.

Spanish

1. **Entrada:** Ruta del archivo `.gie` cifrado, contraseña, canal esperado.
2. **Leer Metadatos:** Leer la cabecera de metadatos (Longitud de la Pista, Pista, Canal, Código de Nivel de Cifrado, Sal de la Clave AES, Sal de la Clave HMAC, IV CTR) del archivo `.gie`.
3. **Verificación de Canal:** Verificar si el `Canal` en el archivo coincide con el `canalEsperado` proporcionado por el usuario.
4. **Derivación de Claves:** Derivar la clave AES y la clave HMAC de la contraseña usando PBKDF2 con las sales leídas del archivo.
5. **Inicialización de HMAC:** Inicializar HMAC-SHA256 con la clave HMAC derivada.
6. **Alimentar Metadatos y Texto Cifrado a HMAC:** Alimentar la cabecera de metadatos y todo el texto cifrado (excluyendo la etiqueta HMAC final) del archivo `.gie` al hasher HMAC.
7. **Leer Etiqueta HMAC Almacenada:** Leer la etiqueta HMAC de 32 bytes del final del archivo `.gie`.
8. **Verificación de HMAC:** Comparar el HMAC calculado con la etiqueta HMAC almacenada. Si no coinciden, el archivo se considera corrupto o la contraseña incorrecta, y el descifrado se aborta.
9. **Inicialización de AES-CTR:** Si la verificación de HMAC pasa, inicializar el cifrado de flujo AES-CTR con la clave AES derivada y el IV CTR.
10. **Archivo de Salida Temporal:** Crear un archivo de salida temporal (`.tmp`).
11. **Descifrar y Escribir Datos:** Leer el texto cifrado en bloques, descifrar cada bloque usando AES-CTR y escribir el bloque descifrado en el archivo de salida temporal.
12. **Renombrar:** Renombrar el archivo temporal al nombre del archivo original (eliminando la extensión `.gie`) e intentar eliminar de forma segura el archivo

`.gie` original.

7. Conceptual Diagrams (Textual Description)

English

- **Overall Architecture Diagram:** A simple block diagram showing "Frontend (React/TypeScript)" communicating with "Backend (Go)" via "Wails IPC", with the Backend interacting with "File System" and performing "Cryptography".
- **Encryption Flow Diagram:** A flowchart illustrating the steps from "Input File + Password" to "Derived Keys + IVs", then "Metadata + Ciphertext Generation", "HMAC Calculation", and finally "Output .gie File".
- **Decryption Flow Diagram:** A flowchart illustrating the steps from "Input .gie File + Password" to "Metadata Extraction", "Key Derivation", "HMAC Verification", "Decryption", and finally "Output Original File".

Spanish

- **Diagrama de Arquitectura General:** Un diagrama de bloques simple que muestra "Frontend (React/TypeScript)" comunicándose con "Backend (Go)" a través de "Wails IPC", con el Backend interactuando con el "Sistema de Archivos" y realizando "Criptografía".
- **Diagrama de Flujo de Cifrado:** Un diagrama de flujo que ilustra los pasos desde "Archivo de Entrada + Contraseña" hasta "Claves Derivadas + IVs", luego "Generación de Metadatos + Texto Cifrado", "Cálculo de HMAC" y finalmente "Archivo .gie de Salida".
- **Diagrama de Flujo de Descifrado:** Un diagrama de flujo que ilustra los pasos desde "Archivo .gie de Entrada + Contraseña" hasta "Extracción de Metadatos", "Derivación de Claves", "Verificación de HMAC", "Descifrado" y finalmente "Archivo Original de Salida".

8. Conclusion

English

GIE provides a robust and secure solution for file and directory encryption, built on modern cross-platform technologies. Its design prioritizes both security through strong cryptographic practices and usability through a responsive graphical interface.

Spanish

GIE proporciona una solución robusta y segura para el cifrado de archivos y directorios, construida sobre tecnologías multiplataforma modernas. Su diseño prioriza tanto la seguridad a través de prácticas criptográficas sólidas como la usabilidad a través de una interfaz gráfica responsiva.