Modelado y Programación

Esquema de Secreto Compartido de Shamir

Introducción

El esquema de secreto compartido de Shamir hace posible que un sólo dato pueda ser ocultado de manera que, a partir de él, se generan n diferentes datos y que con, al menos $t \leq n$ cualesquiera de ellos sea posible recuperar el dato original.

Es bien sabido que r puntos en el plano determinan un único polinomio de grado $r-1$, a saber, el polinomio que interpola los r puntos. Bastan, por ejemplo, dos puntos para determinar la línea que los une, o tres puntos para encontrar la única parábola que pasa por ellos. Así, en el esquema de Shamir de secreto compartido, el dato que se pretende ocultar y compartir, digamos K, se considera como el término independiente de un polinomio de grado $t-1$:

$$P(x) = c_{t-1}x^{t-1} + c_{t-2}x^{t-2} + \ldots + c_1 x + K$$

Así, $P(x)$ puede ser reconstruido a partir de cualesquiera t puntos en los que haya sido evaluado. Para construir $P(x)$ basta generar $t-1$ coeficientes de manera aleatoria, los que, junto con el secreto K, constituyen todos los coeficientes de P. Se procede luego a evaluar a P en $n \geq t$ puntos diferentes, obteniéndose así los n puntos: $\{(x_1, P(x_1)), (x_2, P(x_2)), \ldots (x_n, P(x_n))\}$ que pueden ser distribuidos entre las n personas autorizadas para acceder al secreto K. Si se logran reunir, al menos t de los n puntos distribuidos, es posible calcular el polinomio interpolante de grado $t-1$ que pasa por los puntos reunidos, cuyo término independiente es K, el secreto que se deseaba compartir.

El propósito del proyecto final, es elaborar un programa que implemente el esquema de Shamir para compartir la clave necesaria para descifrar un archivo que suponemos contiene información confidencial. Dicha información debe poder ser accedida siempre que estén presentes, al menos u, de los n miembros de un grupo de personas con autorización para acceder a ella.

A partir de un documento u otro tipo de archivo que suponemos contiene información confidencial, se debe generar una versión cifrada de él. Para ello se utilizará un mecanismo de cifrado simétrico, por ejemplo AES (Advanced Encryption Standard). El documento claro (como llamaremos al original) será cifrado por la entidad que lo produce usando una contraseña sólo conocida por dicha entidad y que debe mantenerse en secreto aún cuando ya no será utilizada. A partir de esta contraseña se debe generar la clave de cifrado K, con la que será cifrado el documento claro. Además de producir el documento cifrado, la clave será considerada como el término independiente de un polinomio de grado $t-1$, mismo que será evaluado en n puntos diferentes. Tanto el documento cifrado, como las n evaluaciones del polinomio, pueden ser distribuidos entre las personas con acceso autorizado al documento claro. Si luego se logran reunir al menos $t \leq n$ de estas personas, usando el programa, pueden descifrar el documento cifrado recuperando el claro.

Entrada

El programa debe funcionar en dos modalidades, para cifrar (opción **c**) y para descifrar (opción **d**).

Cifrar. Se debe proporcionar, en la línea de llamada:

1. La opción **c**.
2. El nombre del archivo en el que serán guardadas las n evaluaciones del polinomio.
3. El número total de evaluaciones requeridas ($n > 2$).

4. El número mínimo de puntos necesarios para descifrar ($1 < t \leq n$).
5. El nombre del archivo con el documento claro.
6. Ya con el programa en ejecución se debe solicitar al usuario una contraseña (sin hacer eco en la terminal).

Descifrar. Se debe proporcionar, en la línea de llamada:

1. La opción **d**.
2. El nombre del archivo con, al menos, *t* de las *n* evaluaciones del polinomio.
3. El nombre del archivo cifrado.

Salida

Dependiendo de la opción seleccionada el programa debe entregar lo que se especifica.

Cifrar.

El archivo con el documento cifrado usando AES.
El archivo con *n* parejas $(x_i , P(x_i))$ de las evaluaciones del polinomio.

Descifrar.

El archivo con documento claro y con el nombre original.

Proceso

El estándar de cifrado de datos AES admite claves de cifrado de 256 bits, longitud recomendada para documentos clasificados de seguridad nacional[1]. Es improbable, sin embargo, que el usuario del programa que produce el documento confidencial utilice una contraseña de 256 bits (32 caracteres), así que para garantizar la longitud de la clave usada para cifrar con AES, es necesario pasar la contraseña por un proceso que, a partir de ella, obtenga una clave de la longitud requerida. Lo adecuado para ello es utilizar una función de dispersión (*hash*) criptográfica, como SHA-256, para la que hasta ahora el criptoanálisis por ataque de colisión o de pre-imagen ha resultado poco práctico. Con esto en mente el proceso de cifrado y obtención de los n fragmentos es el siguiente:

1. A partir de la contraseña p tecleada por el usuario obtener su código *hash* con SHA-256, es decir *sha(p)*, con longitud de 256 bits.
2. Establecer $K = sha(p)$ como el término independiente del polinomio.
3. Obtener las *n* evaluaciones del polinomio.
4. Usar *K* como la llave de cifrado de AES para cifrar el documento claro.
5. Salvar tanto el documento cifrado (en el que de alguna manera se rescata el nombre original del archivo) como las *n* evaluaciones del polinomio.
6. El archivo cifrado debe tener un nombre cuya extensión lo distinga del claro (por ejemplo, poniéndole `.aes`). El archivo de fragmentos debe poseer el mismo nombre que el cifrado, pero diferente extensión (por ejemplo `.frg`). Éste archivo debe tener *n* líneas de texto, cada una con una pareja $(x_i , P(x_i))$ diferente. A cada miembro del grupo autorizado para descifrar le debe corresponder una línea del archivo.

Implementación

Para la implementación se pueden utilizar bibliotecas de software libre disponibles para el cifrado con

[1] Hataway, Lynn, "National Policy on the use of the Advanced Encryption Standard (AES) to protect national security systems and national security information", *CNSS Policy No. 15, Fact Sheet No. 1*, Committee on National Security Systems, junio de 2003.

AES, para SHA-256 y para manipular números enteros muy grandes. El polinomio de interpolación debe ser implementado desde cero.
Se pueden hacer equipos de, a lo más, dos personas.
El programa debe estar bien documentado usando las herramientas que para ello provea la plataforma de desarrollo (javadoc, doxygen). Cada función (método) debe tener una descripción de su funcionamiento, los parámetros que recibe, los resultados que entrega, las pre y post condiciones. El programa debe ser robusto (no morir sin explicación, soportar errores del usuario, manejo de condiciones excepcionales). Cada módulo (clase) elaborada debe tener sus pruebas unitarias. Se debe prestar atención al funcionamiento correcto y robusto y a la eficiencia (en ese orden). El código mal formateado o ininteligible será penalizado.