Memoria de Seguimiento

Cerradura Digital de alta seguridad para cajas fuertes

Pablo Castillo Martínez
Tutor: Jorge Dávila Muro

Índice

Resultados del trabajo realizado

Hasta la fecha, se ha realizado un trabajo de investigación sobre las cajas fuertes y sistemas de seguridad modernos. Desde los aspectos físicos, como materiales que se utilizan, a mecanismos de cerradura, tanto mecánicos como eléctricos (a pesar de que mucho de éstos componentes internos existen bajo secreto en la empresa que los manufactura).

Aparte de éstos componentes físicos, también se han estudiado las distintas normas y estándares que existen actualmente para los sistemas de seguridad, tanto en España por el BOE, como normas UNE, como normas aplicables en los Estados Unidos (sobre materiales necesarios, composición del material, medidas necesarias para proporcionar gran seguridad…).

Sobre la cerradura, se han estudiado los componentes que forman una cerradura tradicional (cerradura de tambor y cerradura de combinación), y cómo se puede incorporar una cerradura a nuestro proyecto ofreciendo una mayor seguridad.

Se pretende implementar una cerradura basada en los protocolos de autenticación que se tienen hoy en día en Internet.

De forma general, se pretende que el usuario del sistema de seguridad disponga de una “llave” eléctrica capaz de comunicarse con la caja fuerte.

Para ello, se ha programado un sistema inicial basado en HOTP (HMAC-based One-time Password Algorithm). Los pasos que sigue son:

- Generar un contador aleatorio de 8 Bytes, y comprobar que no ha sido generado anteriormente para que no ningún atacante pueda aprovecharse de un mensaje generado varias veces.

- Establecer una conexión entre usuario y caja fuerte a través del algoritmo de Diffie-Hellman (**DH**). Generará una clave entre caja fuerte y sistema del usuario. Por las características de DH, a pesar de compartir mensajes en claro, no es posible revertir para obtener los valores privados de cada parte, por las propiedades de la aritmética modular, en concreto de la propiedad de clases de equivalencia módulo n, ambas partes llegarán a la misma clave final K, ya que bajo un módulo p (primo) común:

$$A^b \bmod p = g^{ab} \bmod p = g^{ba} \bmod p = B^a \bmod p$$

Siendo los valores A y B públicos de cada parte, de la forma A = g ^ a mod p / B = g ^ b mod p, y a y b los privados de cada parte.

Como antes, los valores privados se guardan en un archivo para no volver a generarlos de nuevo.

A continuación, se podrá establecer la conexión, ya que ambos comparten la misma clave.

Se han seguido los requisitos del RFC (Request For Comments) asociado a DH, es decir, RFC 2631 y 4419

- Presentar al usuario con una interfaz que le deje introducir el contador de su llave (en un sistema real, este valor es privado. El valor de la llave aumenta cada vez que se pide una conexión, falle o no. El contador de la caja sólo aumenta tras una conexión acertada. El contador del usuario puede estar adelantado, y se debe sincronizar en una ventana de n valores o impedir el acceso).

- A continuación, se calcula el HMAC (Hash Based Message Authentication Code) con la función hash SHA-512, que da un output de 512 bits o 64 Bytes. Normalmente una función hash se usa para calcular el “resumen” de un mensaje o un archivo, el valor único asociado a éste. Pero como éstos mensajes pueden ser manipulados, se usa el HMAC.

Los componentes necesarios para construir el HMAC son los siguientes:

- Función criptográfica ***H*** con la que se "hashean" los datos bajo una función de compresión sobre bloques de tamaño ***B*** de datos (SHA-512 da una salida de 64 Bytes).
- Clave Secreta ***K***.
- Tamaño ***B*** bytes de bloques con los que se operan.
- Tamaño ***L*** bytes de salida del algoritmo.

Además, contamos con dos constantes, ***ipad*** = el byte 0x36 repetido *B* veces, y ***opad*** = el byte 0x5C repetido *B* veces. La operación es la siguiente:

$$\mathrm{HMAC}(K, m) = \mathrm{H}\left(\left(K' \oplus opad\right) \parallel \mathrm{H}\left(\left(K' \oplus ipad\right) \parallel m\right)\right)$$

$$K' = \begin{cases} \mathrm{H}(K) & K \text{ is larger than block size} \\ K & \text{otherwise} \end{cases}$$

Genera el HMAC necesario. Los pasos en detalle son los siguientes:

1. Insertar ceros al final de *K* para crear un string de *B* bytes (Si *K* tiene un tamaño de 20 Bytes y *B*=64, se insertan 44 Bytes de 0x00).
2. XOR del paso 1 con el ipad.
3. "Append"/inserta al string de texto junto al resultado del paso 2.
4. Aplicar *H* a la salida del paso 3.
5. XOR del paso 1 con el *opad*.
6. "Append"/inserta la salida *H* del paso 4 a la salida del paso 5.

Aplicar *H* a la salida anterior y devolver el resultado.

Se han seguido los requisitos del RFC (Request For Comments) asociado a HMAC, es decir, RFC 2104

- Después de conseguir HMAC, calculamos HOTP. A partir del HMAC, podemos conseguir contraseñas de un sólo uso, capaces de cambiar en cada uso para una caja fuerte, siempre que los miembros de la comunicación estén sincronizados con el mismo valor del contador.

Por defecto se usa SHA-1 en la implementación estándar, y se hablará de él en los ejemplos a continuación por facilidad para entenderlo, pero en mi implementación se usa SHA-512, que es mucho más seguro y no presenta ningún riesgo de seguridad.

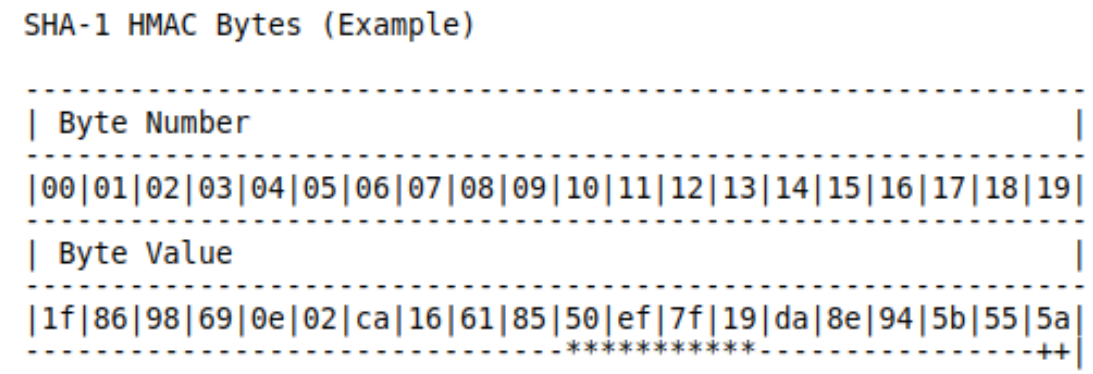
Si se usa SHA-1, el output es de 160 bits, ese valor se trunca entonces para obtener un valor que sea fácilmente introducible por el usuario, de la forma:

$$HOTP(K,C) = Truncar(HMAC\text{-}SHA\text{-}1(K,C))$$

Se trunca de la siguiente forma: Poniendo de ejemplo la salida de SHA-1 de 160 bits (20 Bytes), cogemos los 4 últimos bits del último Byte (El Byte 19, si se empieza a contar desde el Byte 0).

Con éstos últimos bits, los pasamos a decimal. Teniendo 2^4 números posibles para elegir desde un subgrupo (Del 0 al 15). Teniendo el grupo que nos haya salido, cogemos los 3 siguientes grupos de Bytes, y tenemos un número de 4 Bytes (Por ejemplo, 0x50ef7f19).

Ahora éste número se puede pasar a decimal, se hace la operación de módulo igual a la longitud que queremos que introduzca el usuario y que calcule el sistema (Para 8 números de contraseña, 0x50ef7f19 (pasado a decimal) % 10^8).




```
* The last byte (byte 19) has the hex value 0x5a.
* The value of the lower 4 bits is 0xa (the offset value).
* The offset value is byte 10 (0xa).
* The value of the 4 bytes starting at byte 10 is 0x50ef7f19,
  which is the dynamic binary code DBC1.
* The MSB of DBC1 is 0x50 so DBC2 = DBC1 = 0x50ef7f19 .
* HOTP = DBC2 modulo 10^6 = 872921.
```

El método de truncamiento dinámico, junto a un contador distinto en cada uso, da lugar a salidas uniformes e independientemente distribuidas.

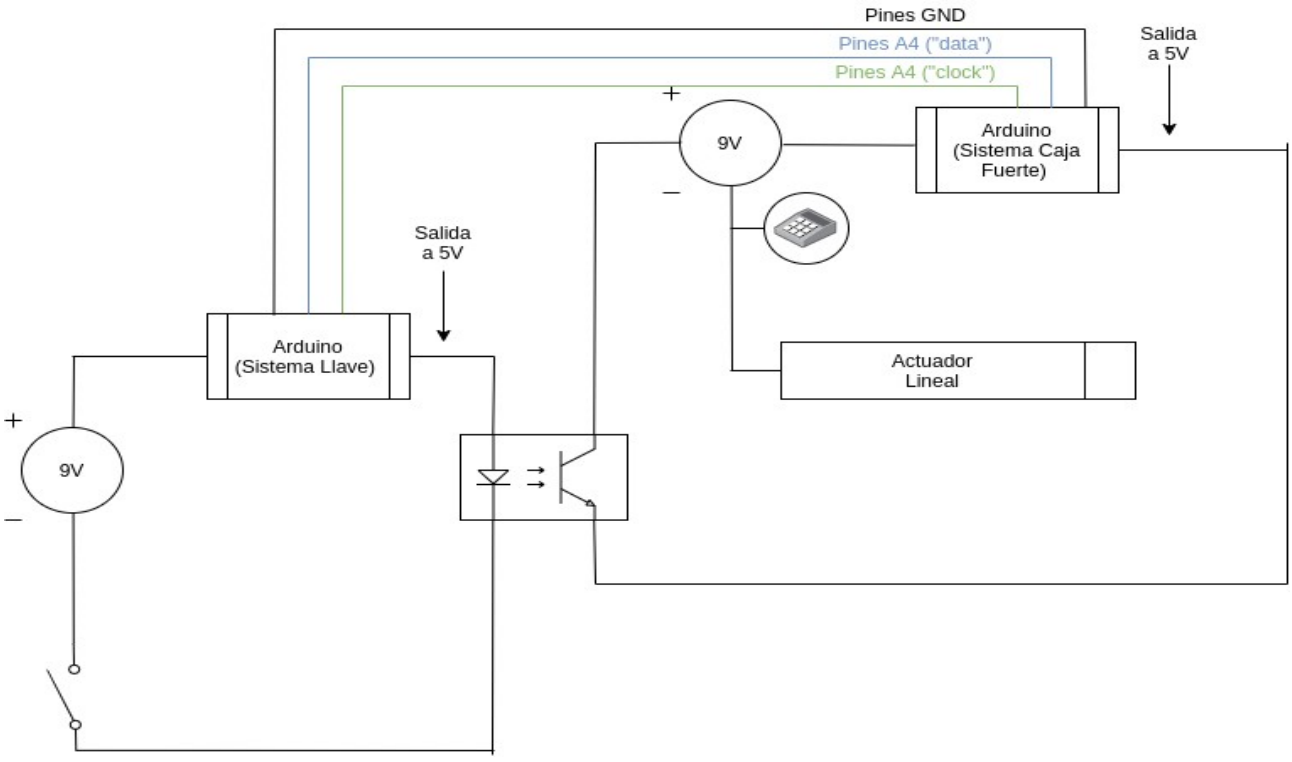
En nuestro caso, al tener una salida de 512 bits, 64 Bytes, no se puede hacer lo mismo. Disponemos de 64 Bytes de salida, que son 64 grupos, del 0 al 63. Para cubrir éstos grupos, se necesita cubrir como máximo hasta el 60, ya que el grupo de 4 Bytes máximo elegible es del 60 al 63. A partir del 61, perdemos Bytes. Se ha decidido coger los últimos 8 bits, y hacer un módulo 60 para encontrar todos los valores en éste rango. Es decir, si el último Byte es ff (11111111 == 255), corresponde al grupo 15 (255 mod 60), y se eligen los grupos del 15 al 18.

Se han seguido los requisitos del RFC (Request For Comments) asociado a HOTP, es decir, RFC 4226 y 6238

- A partir de los grupos de Bytes, se pasan a decimal, se cogen tantos valores como dígitos queramos que tenga la combinación de un solo uso de la cerradura (entre 6 y 8 números, una mezcla de seguridad y facilidad de introducir).

TODA esta serie de operaciones las hacen tanto la caja fuerte como la cerradura, y que el resultado sea el mismo depende de los valores de los contadores y que estén sincronizados.

También se comenzó a diseñar dos circuitos (uno para llave de usuario, otra para caja fuerte), con los conocimientos básicos de electrónica de los que dispongo, juntando los dos en uno sólo:



Éste ha sido el trabajo hasta ahora, de la forma más resumida posible.

Explicación y justificación de las modificaciones al Plan de Trabajo

Revisión de la lista de objetivos del trabajo

La lista de objetivos planteada al inicio del proyecto está avanzando según lo acordado; ya nos encontramos trabajando en el objetivo de codificar, y salvo algunos detalles que faltan implementar, todo ha ido según lo acordado y los objetivos han sido alcanzados.

No se necesita ninguna modificación, al menos en éste momento.

Revisión de la lista de tareas

La codificación en Python ha cumplido con la fecha acordada aproximadamente (se propuso el 17 de octubre, se empezó realmente el 19 de octubre, viendo el repositorio de github sobre los commits realizados. Éstos dos días se usaron para buscar más información de componentes y seguridad, antes de empezar con la implementación).

El progreso ha sido rápido, tras una semana centrado en el código ya se cuenta con un modelo funcional, mejorable, pero cumple las expectativas y ha servido de mucho para hacer debug y unos tests iniciales sobre HMAC y HOTP.

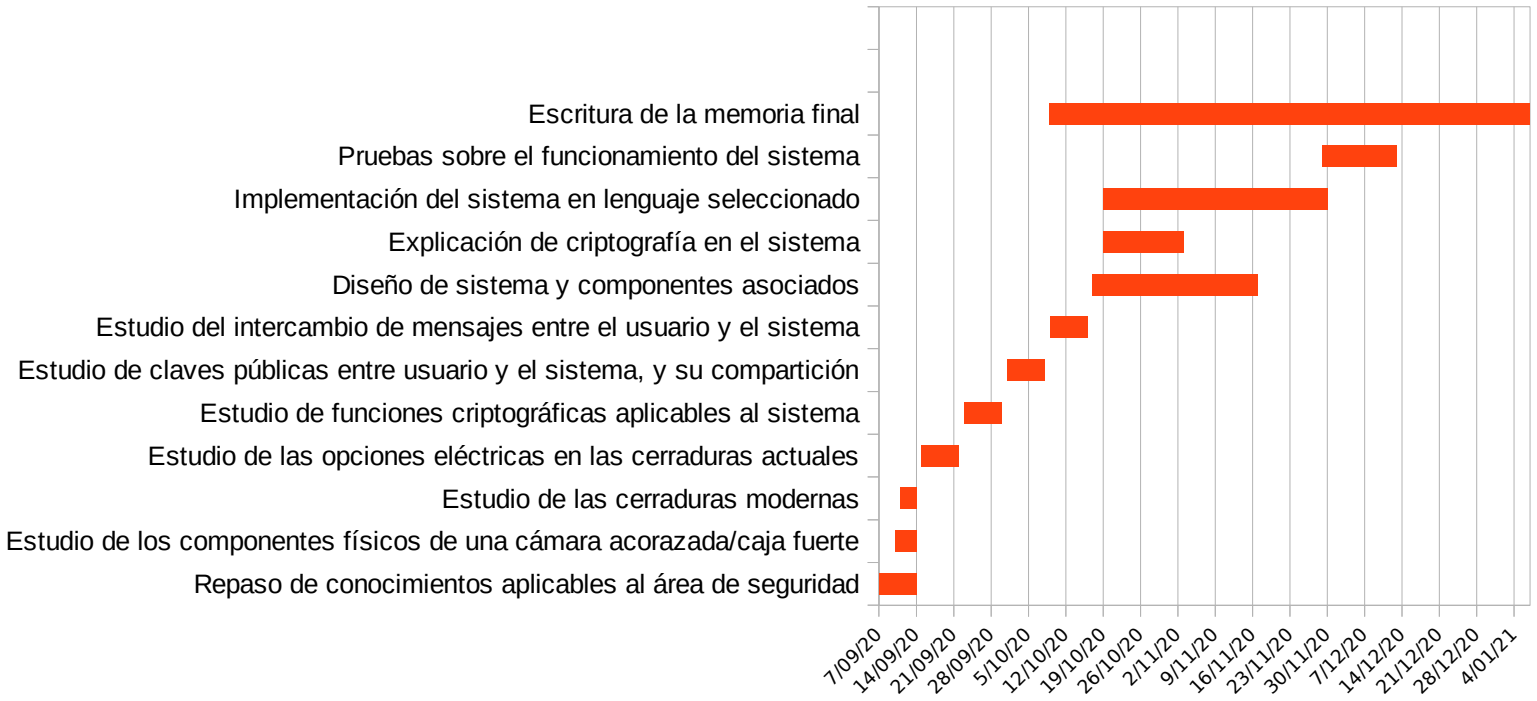
Ciertas funciones de diseño de componentes se han podido juntar y diseñar a la vez, como el circuito y las baterías que se usan, de una manera sencilla.

Hace falta modificar el plan y añadir una entrada llamada “explicación de la seguridad en el sistema”, para dedicar más tiempo a estudiar en detalle la seguridad total que proporcionan estas medidas.

Pero en general se han cumplido las metas dentro de sus márgenes de tiempo, aunque varias tareas se podrían juntar en una sola,”diseño del sistema y de sus componentes asociados”.

Revisión del diagrama de Gantt

Pocos cambios realizados, aparte de poner bien la fecha en que se comenzó a programar, y juntar todas las tareas del diseño del sistema en una sola, sumando tiempos, y la explicación de la criptografía en el sistema.



Borrador del índice

Hasta ahora, se está trabajando con el siguiente índice:

Primeros dos capítulos

A continuación se presentan dos capítulos trabajados, introducción y trabajos previos:

Introducción

La aplicación de nuevos avances tecnológicos a los mecanismos diseñados para mantener alejadas de nuestras posesiones a personas sin el permiso adecuado siempre ha estado a la orden del día. Desde cerraduras tradicionales, pasando por las que aprovechan los efectos del electromagnetismo y la corriente para evitar un forzado tradicional, a la tecnología de lector de tarjetas, biometría, claves... para probar nuestra identidad, las cerraduras han avanzado a pasos agigantados.

Pero igual que ha avanzado la tecnología para darnos seguridad de la mano de la comodidad (la posibilidad de no llevar llaves encima, de poder conectar nuestros teléfonos para abrir una puerta...), también han avanzado las habilidades y conocimientos de aquellos interesados por romper estas defensas.

Pero, ¿cómo se adaptan las cerraduras electrónicas a cajas fuertes? Existe un amplio mercado basado en la venta de cajas fuertes (de distintas categorías, tamaños, funciones...) e instalación de bóvedas bancarias con características como contraseña, lector de ID, conexión móvil, adicional a las combinaciones o llaves tradicionales. La protección que otorgan está regulada por distintos estándares.

Si bien es cierto que las cerraduras más modernas [1] (con contraseña electrónica, con lector ID, con conexión Bluetooth al móvil...) bien aplicadas por marcas de confianza son igual de seguras que las cerraduras tradicionales, suelen destacar marcas comerciales menores, poseyendo unos materiales deficientes en comparación, o mayor facilidad para forzarlas por fallos de diseño. No por ser digital proporciona una seguridad adicional.

Pero en esta área cabe aún espacio donde innovar. Por ejemplo, añadir componentes criptográficos basados en un secreto entre la cerradura y el usuario que haga que la combinación a introducir sea diferente en cada uso que se realice. Esta cerradura tiene que proporcionar la misma seguridad que los métodos tradicionales, ser difícil de sustraer o interceptar los mensajes de clave y, además, registrar todas las combinaciones o claves usadas para no utilizarlas de nuevo.

Éste trabajo pretende diseñar e implementar un sistema de alta seguridad para una caja fuerte, con técnicas que son actualmente usadas en la seguridad y criptografía modernas.

También se pretende estudiar la fiabilidad que puedan proporcionar éstas técnicas tras algunas de ellas llevar múltiples décadas en uso.

Trabajos previos

Si se pretenden aplicar componentes de seguridad de tipo criptográfico a los sistemas tradicionales, primero es necesario una breve introducción sobre la historia de éstos en el mundo moderno.

Si lo que nos interesa es un sistema con una clave numérica secreta, una combinación, capaz de cambiar tras cada uso, debemos investigar los principales mecanismos que existen actualmente para generar contraseñas de un sólo uso.

En el mundo comercial, por ejemplo, destacan los servicios proporcionados por Google (y empresas que permiten el uso de éste en su software) y su Authenticator [2]. Aparte de la información normal de usuario y password, se genera una “contraseña” de un sólo uso que se manda a un canal seguro proporcionado por el usuario para probar que es él mismo el que accede, ya que dispone de información conocida para el acceso, y puede iniciar sesión en el canal seguro.

Éste es el mundo de la autenticación de múltiples factores, o AMF [3] (que también suele ser conocida por autenticación de dos factores, ya que normalmente sólo se usan dos para confirmar la identidad del usuario). Combinando diferentes componentes que debe poseer el usuario (algo que se tiene, algo que se sabe, o algo que se es), se consigue una seguridad total mayor.

Éste y otros métodos operan bajo los algoritmos HMAC (Hash-based message authentication code) y HOTP (HMAC-based one-time Password). Ambos operan bajo sus respectivos RFC, request for comments, documentos del IETF, Internet engineering task force, que dictan con detalle cómo y con qué parámetros formar éstos algoritmos, que se explicarán en más detalle en la parte de codificación del proyecto.

Los RFC de más interés son los siguientes: RFC 4226, RFC 6238 (ambos de HOTP), RFC 2104 y RFC 4868 (ambos de HMAC). [4][5][6][7]

A continuación, se pasará a describir con detalle las características físicas de los mecanismos de caja fuerte/cerradura.

A continuación, la bibliografía que se ha usado en todo el proyecto hasta ahora:

Bibliografía

[1] Cerradura inteligente, https://en.wikipedia.org/wiki/Electronic_lock

[2] Google Authenticator, https://es.wikipedia.org/wiki/Google_Authenticator

[3] Autenticación de múltiples factores, https://es.wikipedia.org/wiki/Autenticaci%C3%B3n_de_m%C3%BAltiples_factores

[4] RFC 4226, https://tools.ietf.org/html/rfc4226

[5] RFC 6238, https://tools.ietf.org/html/rfc6238

[6] RFC 2104, https://tools.ietf.org/html/rfc2104

[7] RFC 4868, https://tools.ietf.org/html/rfc4868

[8] Caja de caudales, definición de la RAE, https://dle.rae.es/caja#2s7HuNa

[9] Orden INT/317/2011, de 1 de febrero, sobre medidas de seguridad privada, https://www.boe.es/buscar/act.php?id=BOE-A-2011-3171

[10] Información en detalle sobre las cerraduras, https://www.locks.ru/germ/informat/schlagehistory.htm

[11] Intercambio mediante Diffie-Hellman, https://en.wikipedia.org/wiki/Diffie%E2%80%93Hellman_key_exchange

[12] RFC 2631, https://tools.ietf.org/html/rfc2631

[13] RFC 3526, https://tools.ietf.org/html/rfc3526

[14] Especificaciones de Arduino, https://www.arduino.cc/en/Main/ArduinoBoardUnoSMD

[15] Imperfect Forward Secrecy: How Diffie-Hellman Fails in Practice, https://weakdh.org/imperfect-forward-secrecy-ccs15.pdf

[16] Ejemplo de comunicación entre Arduinos, https://www.arduino.cc/en/Tutorial/LibraryExamples/MasterWriter

[17] Ejemplo de activación de actuador lineal, https://www.firgelliauto.com/blogs/tutorials/how-do-you-control-a-linear-actuator-with-an-arduino

[18] Repositorio de Github asociado al trabajo, https://github.com/PCMSec/TFG