

Memoria de Seguimiento

Cerradura Digital de alta seguridad para cajas fuertes

Pablo Castillo Martínez
Tutor: Jorge Dávila Muro

Índice

Resultados del trabajo realizado.....	4
Explicación y justificación de las modificaciones al Plan de Trabajo.....	12
Revisión de la lista de objetivos del trabajo.....	12
Revisión de la lista de tareas.....	12
Revisión del diagrama de Gantt.....	13
Borrador del índice.....	14
Primeros dos capítulos.....	14
Introducción.....	15
Trabajos previos.....	16
Bibliografía.....	17

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.

El primer caso necesita de una llave (algo que se tiene). El funcionamiento se basa en una combinación de varios pines apoyados sobre un muelle, todos a distintas alturas. En su estado habitual, prohíben el movimiento del cilindro interior, hasta que se introduce el perfil de llave correcto, que hace que cada pin se levante a la altura adecuada para poder mover el cilindro, y poder abrir o cerrar la cerradura.

Se precisa de un elemento físico que no comprueba quien porta la llave, por lo que no es una buena idea para un sistema de alta seguridad.

En una cerradura de combinación, nos encontramos con una serie de discos que necesitan de una alineación correcta para que un mecanismo interior “salte” y libere el cierre. Si se introduce un dígito incorrecto, mueve los discos interiores para que no se abra.

Este mecanismo se basa en algo que se sabe, una combinación, pero un usuario “enemigo” que observe qué combinación es podrá meterse de nuevo, a menos que se cambie, y que el cambio sea rápido y casi instantáneo.

Por lo tanto, se pretende implementar un sistema basado 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 $A = g^a \bmod p$ / $B = g^b \bmod p$, 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:

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

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

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:

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

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).

SHA-1 HMAC Bytes (Example)

Byte Number
00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19
Byte Value
1f 86 98 69 0e 02 ca 16 61 85 50 ef 7f 19 da 8e 94 5b 55 5a
*****++

Raihi, et al.

Informational

[

C 4226

HOTP Algorithm

Decemb

- * 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 finámico, 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.

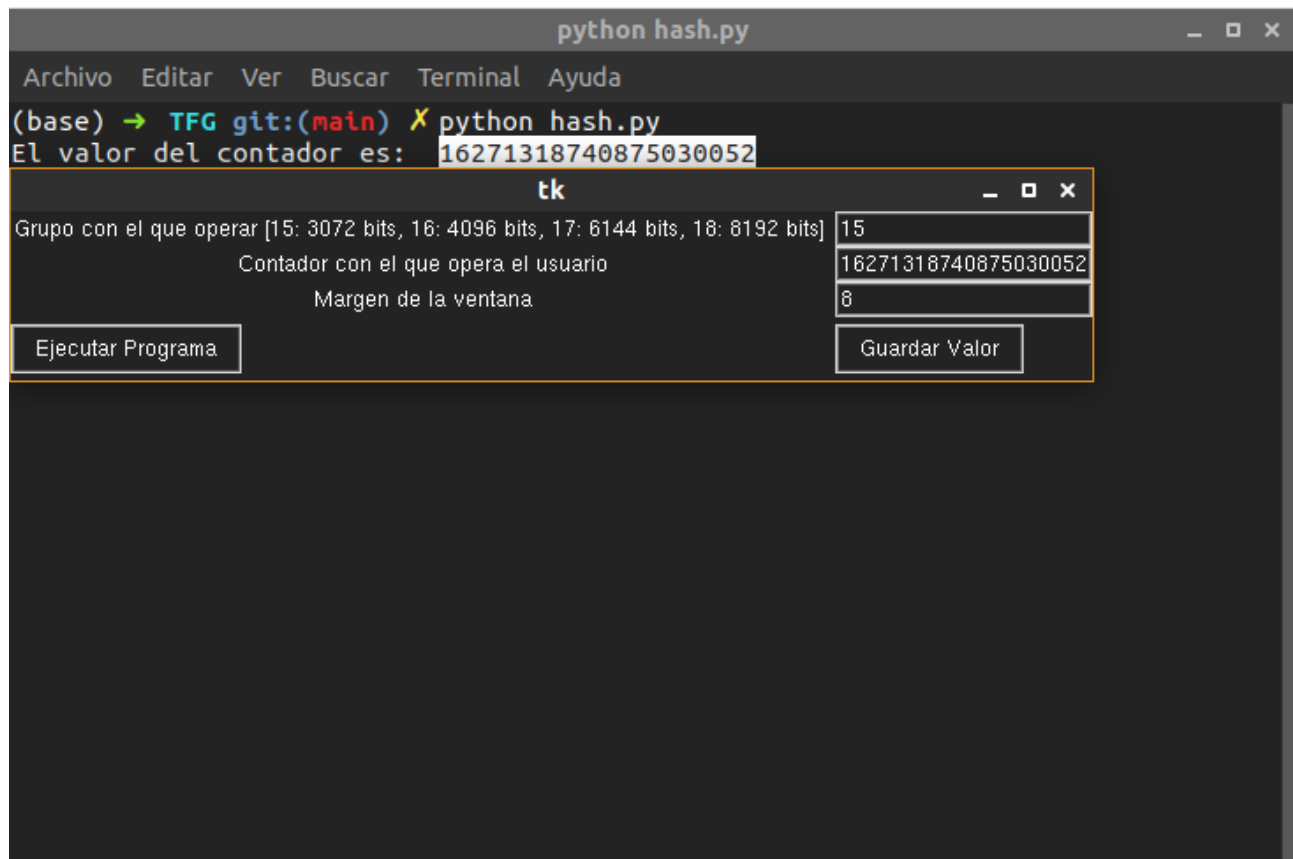
Se han seguido los requisitos del RFC (Request For Comments) asociado a HOTP, es decir, RFC 4226 y 6238

- 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.

Ya está terminada una implementación de éstas operaciones, totalmente funcional, en Python. Requiere unos detalles finales, como ocultar los contadores de cada sistema para que el usuario no accede a ellos de forma normal.

Un ejemplo de su uso:



Nos da como resultado (abreviado para no ocupar mucho espacio):

Trazas de log del sistema

Valor privado de a:

513217249502504875914937383042243799584589314614951999981728419
659469531360809791169745878095058461906876188989834855959027127
646476613291032942868414839946634836092942360893085711731835865
781188860915437940082635704274977414410153231490461519912268331
722446398603230133088409712191689879222851616658936793670078944
725270719513018106581074849711365187247926020309842530693409227
652066455421505187017902086836081722894040195079222050992775085
088093655232380391325363920876264855458798000047434430356110498
261902236520955706599980521643719502132901291378977263481466421
708732086683240014191406065348518689243090860267355640011742830
147810055873340302482002080403514512876850551459385157286711808
534807954826887114753464732384303684643936675576879145208110916

806792653424661955850731000597727722068399780091828079271478929
780570569256836084630802744045303353733877007020907227824344003
3823246235801615901129245207756959596963330

Valor privado de b:

256861659970398075488053184192367008997408143521258254761623294
961286754361452102973157725300853848778727289430328529339667405
883550730671208861918469272782750315656948838381223420338439249
882542969685132318046085312162875649393515812415490362792446641
212949400607597883381294127395901700297362901976160270425814827
032491205414692368924920048051060690413174522790643784722977958
943341883122090594415525151467506191476147384205259327362059864
333629612226125338084596185878517037023467080866325343789630326
243550358296930602092864484799739253884019985386512218906834007
569258762320247104933798282270620353727638637450590539304299522
484848973269454492278228141014642135293603593945020736638332176
698483149382263964654802298216389759957784970517415841500793022
006332996623209312128359387569593636810299044748665707139875936
199817485171257432849120580653421429352028075498110950389144931
694168516782443360090444801132794653404249

[Abreviado]

Valor de la clave para a:

184128963218365763929611009820689988382847630388425718255404311
390369173819492296557696886298355191215280445934493544293070176
764537943924814778983556861234806225064957166607465365882857252
234646634165308759976553452626634371614927624834316881098946178
054706784674491522940716833718725019185821221527292497614543194
877104024847892200589439780074251061749363691836565529102950619
869600030990117022961195279086518744505069858265665893662458015
472478363338273200251995197223046620868435413272671362838754834
332960929127634691456236330920277334476985484791141659998587573
094616388249858735907484175636230964842534584152513471359547254
794140354630103919539861841643878854390927165046510945043904187
775872019541025307438674228438066751639840479958014823021764806
017053162086000277602269814510226276224817784979210413597067280
526295796167764333602876519712301287632862760849502801823034786
7055325856346403504368857621291412589772557

Valor de la clave para b:

184128963218365763929611009820689988382847630388425718255404311
390369173819492296557696886298355191215280445934493544293070176
764537943924814778983556861234806225064957166607465365882857252

234646634165308759976553452626634371614927624834316881098946178
054706784674491522940716833718725019185821221527292497614543194
877104024847892200589439780074251061749363691836565529102950619
869600030990117022961195279086518744505069858265665893662458015
472478363338273200251995197223046620868435413272671362838754834
332960929127634691456236330920277334476985484791141659998587573
094616388249858735907484175636230964842534584152513471359547254
794140354630103919539861841643878854390927165046510945043904187
775872019541025307438674228438066751639840479958014823021764806
017053162086000277602269814510226276224817784979210413597067280
526295796167764333602876519712301287632862760849502801823034786
7055325856346403504368857621291412589772557

Los valores son iguales, comparten la misma clave

Tamaño del HMAC resultante: 64 Bytes

Resumen HMAC resultante:

468956c5170995970f91c8ce8758566d82fe6742e6ac40cf80a55736f9d72c47fd
19775c842bd4a49073749966dc41a7c89da729b8534c3eecf8f70e3d3839d4

Último Byte del HMAC: d4

Representacion binaria de d : 0b1101

Representacion binaria de 4 : 0b100

Binario del último Byte: 1101100

Valor en decimal del grupo de BITS resultante: 108

Se elige el grupo de Bytes 54

4 Bytes que nos salen: 4c3eecf8

Resultado decimal de los Bytes obtenidos: 1279192312

Primeros 8 dígitos de la contraseña: 12791923

Los contadores de ambos son iguales, calculando HOTP para ambos

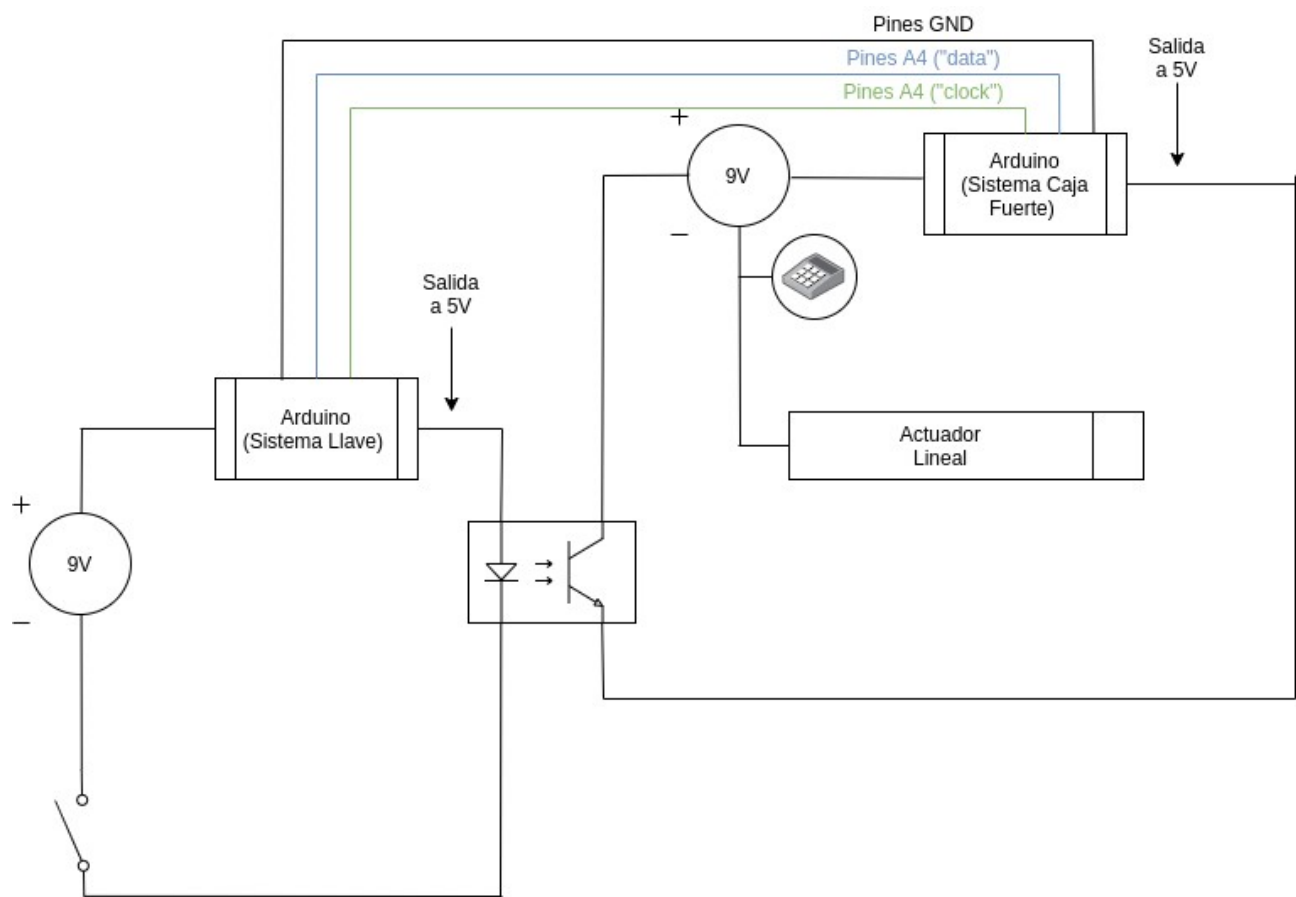
[Abreviado, en este paso el usuario hace las mismas operaciones para conseguir la clave final]

Resultado decimal de los Bytes obtenidos: 1279192312

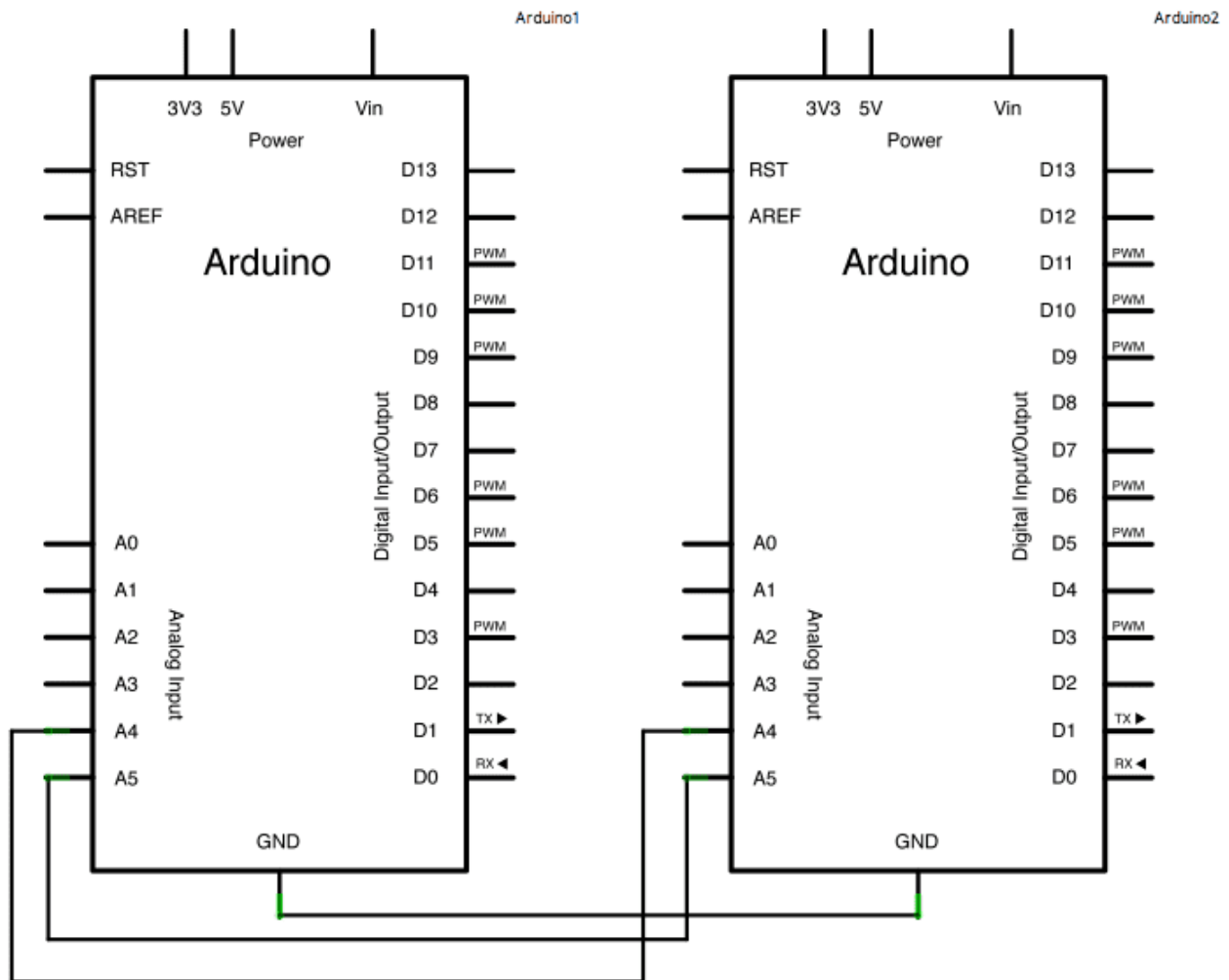
Primeros 8 dígitos de la contraseña: 12791923

Los valores coinciden, se da acceso

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:



A continuación, la comunicación entre dos Arduinos a través de sus pines. Se ha decidido que sea de manera directa en ésta versión inicial:



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.

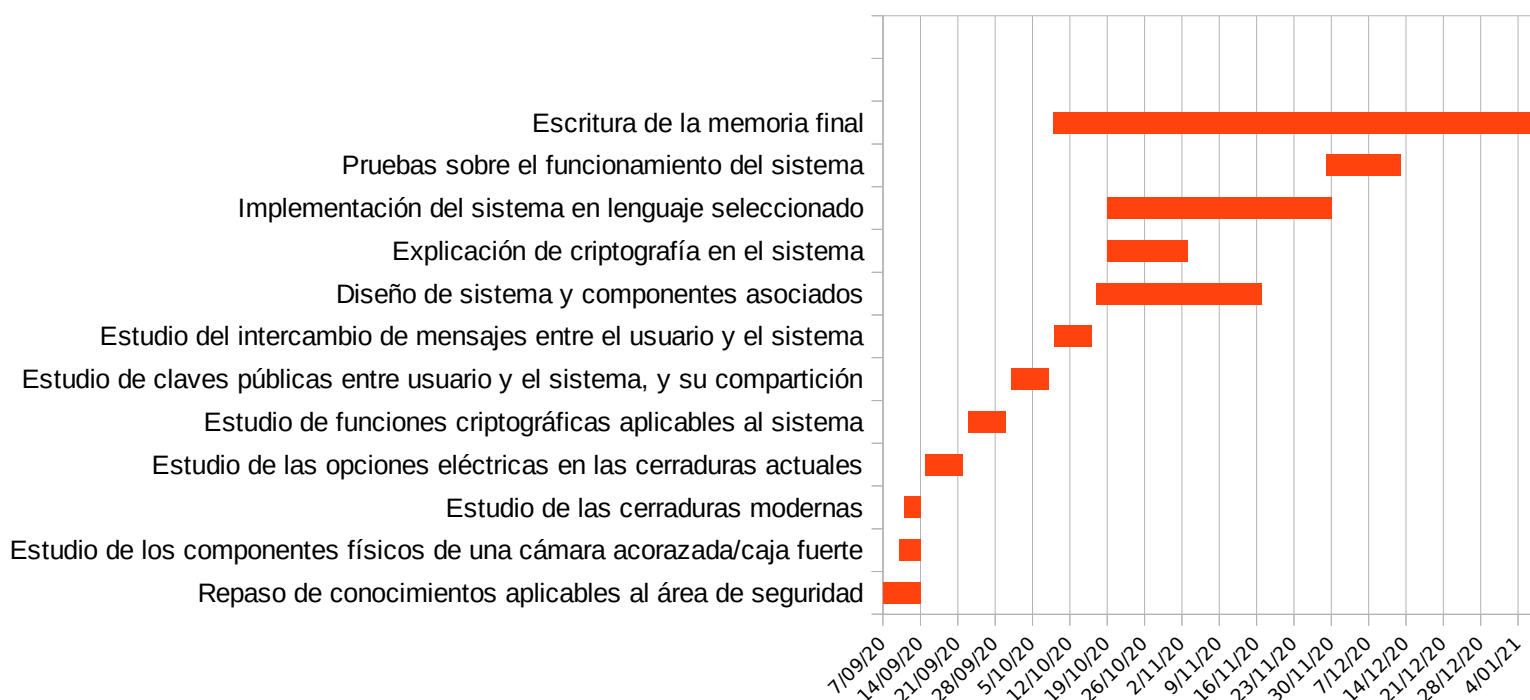
Ciertas funciones de diseño de componentes se han podido juntar y diseñar a la vez, como el circuito y las baterías que se usan, de una manera sencilla.

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:

Resumen.....	3
Abstract.....	4
Introducción.....	5
Trabajos previos.....	6
Especificaciones físicas.....	7
Materiales.....	7
Cerradura.....	7
Mecanismo de apertura.....	9
Conclusión.....	10
Diseño del sistema.....	11
Criptografía.....	11
Diffie-Hellman.....	12
Especificaciones de HMAC/HOTP.....	13
HMAC.....	13
HOTP.....	14
Función Hash SHA-512.....	17
Acceso al Sistema.....	17
Seguridad asociada a la parte criptográfica.....	19
Diffie-Hellman.....	19
HMAC.....	19
HOTP.....	20

Componentes del sistema.....	21
Sistema Llave.....	21
Sistema Caja Fuerte.....	22
Circuitos.....	22
Seguridad del circuito.....	24
Implementación en código.....	25
Clase diffieHellman.py.....	25
Funciones del archivo.....	27
Clase hash.py.....	31
Funciones del archivo.....	33
Pruebas.....	42
Pruebas Iniciales.....	42
Grupo 15.....	42
Grupo 16.....	43
Grupo 17.....	43
Grupo 18.....	44
Pruebas avanzadas.....	45
Prueba 1.....	46
Prueba 2.....	49
Prueba 3.....	54
Bibliografía.....	55
Anexo A: Componentes.....	58

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>