JITEL 2017
XIII Jornadas de Ingeniería Telemática
27-29 Septiembre 2017, Valencia (España)

Actas de las XIII Jornadas de Ingeniería Telemática (JITEL 2017), Valencia (España), 27-29 de Septiembre de 2017.

ISBN: 978-84-9048-595-8
DOI: http://dx.doi.org/10.4995/JITEL2017.2017.6562

Establecimiento de claves seguro mediante códigos sonoros en dispositivos móviles

Ricardo Ruiz Tueros, Isaac Agudo
Departamento de Lenguajes y Ciencias de la Computación
Universidad de Málaga
{rrt,isaac}@lcc.uma.es

***Resumen*—El objetivo de este trabajo ha sido investigar el uso de canales sonoros para el intercambio de claves en teléfonos inteligentes. Para ello, se ha realizado un diseño y un análisis del problema a resolver, teniendo en cuenta factores como la necesidad de sincronización, los parámetros de escucha, la longitud de la clave que podemos obtener, los ataques a los que sería vulnerable el protocolo, la necesidad de cierta persistencia a nivel local, el uso de la nube, etc.**

Se ha realizado un análisis de tecnologías relacionadas, eligiendo aquellas que mejor se adaptan a los requisitos del problema y se ha implementado un prototipo capaz de realizar un intercambio de claves y autenticar mutuamente a un par de dispositivos, creando así un canal seguro a través del cual puedan comunicarse en el futuro, sin necesidad de que una tercera parte confiable que certifique la identidad de las partes.

***Palabras Clave*—Autenticación, Seguridad, Sonido , Intercambio de claves, iOS, Nube**

I. INTRODUCCIÓN

La motivación inicial detrás de este trabajo es la implementación de una aplicación para teléfonos inteligentes que permita establecer una clave compartida entre dos dispositivos de forma que los respectivos usuarios tengan un control total sobre el proceso. Para ello se han analizado diferentes aproximaciones al problema y se ha implementado un prototipo que resuelve el problema usando canales sonoros.

Existe una gran variedad de protocolos de intercambio de claves [1] en la literatura, si bien uno de lo más conocidos y usados en la actualidad, por ejemplo en los protocolos TLS e IPSEC, es Diffie Hellman (DH) [2]. Uno de los requisitos para que DH funcione correctamente es que los usuarios implicados en el protocolo tengan la certeza de que han calculado la misma clave compartida, para de esa forma evitar un ataque de Man-in-the-middle (MitM). Una posible solución al problema se basa en la autenticación de las claves públicas DH, tal y como ocurre en TLS cuando se utiliza el intercambio de claves basado en DH. Esto requiere de una tercera parte confiable (Autoridad Certificadora) que certifique la identidad de las partes, mediante un certificado de clave pública, y del establecimiento de relaciones de confianza que pueden ser complejas de administrar. Nuestra propuesta se basa en la utilización de un canal *"fuera de banda"* , que nos permita realizar la autenticación de los dispositivos sin necesidad usar claves autenticadas. Esto se engloba dentro de las técnicas de "Key Fingerprint Verification" usando canales fuera de banda [3].

Nuestro objetivo final no es saber quien es la otra persona, sino tener la certeza de que cuando queramos enviar algo a un dispositivo que ya hemos enlazado previamente, solo ese dispositivo será capaz de leerlo. Por tanto, se cuenta con una primera fase donde la autenticación en el intercambio de claves está fundamentada solo en la proximidad y la capacidad que tienen los usuarios de observar el intercambio de información, y una segunda fase de comunicación donde la autenticación de basa en la clave negociada en la primera fase.

Si bien inicialmente se valoraron diferentes protocolos inalámbricos para comunicaciones de corto alcance (Bluetooh, NFC, etc.) como canal fuera de banda, se descartó esta aproximación debido a dos factores principales:

- El uso de antenas de gran tamaño y/o potencia puede ampliar el rango de comunicaciones, de forma que el sentido de proximidad se pierda.
- Los usuarios no tienen una constancia directa de cuando se están comunicando los dispositivos ni pueden identificar visualmente que dispositivos son los que se están emparejando. Esto último es un problema reconocido en los sistema de pago sin contacto[4].

Existen diversas técnicas para el emparejamiento de dispositivos móviles que no recurren al uso de tecnologías de radio frecuencia. Un ejemplo sería el uso del canal háptico (vibración) [5], es decir, las vibraciones del teléfono. Una de las ventajas o limitaciones, según los requisitos que se tengan en cuenta, de este canal es que solo funciona entre un par de dispositivos y además, estos tienen que estar

en contacto. Esto podría ser bueno porque evitaría ataques de intermediarios pero puede resultar intrusivo al requerir que se coloquen los dos dispositivos en contacto.

Como alternativa se buscó un canal de comunicación aún más básico, las personas intercambian secretos mediante susurros, esto es, en definitiva sonido... ¿Y si un dispositivo pudiera "susurrar" un secreto a otro?. El canal sonoro cumple las condiciones que necesitamos: Es un canal de dispersión entre esos dos dispositivos que puede ser controlado sencillamente, mediante el volumen, y que es percibido por los usuarios de forma directa. Es independiente de la plataforma y fácilmente accesible desde cualquier sistema operativo y hardware, ya que el sonido sigue siendo el mismo (o al menos tan sólo ligeramente distinto), sumado a que hoy en día casi cualquier dispositivo digital incorpora un micrófono y un altavoz lo hacen el canal fuera de banda ideal para nuestro esquema.

Los canales sonoros también se han utilizado con otros propósitos. Por ejemplo, [6] utiliza el sonido ambiente recibido por dos dispositivos móviles como segundo factor en la autenticación, si el sonido ambiente grabado por los dos dispositivos es el mismo, podemos asumir que se encuentran en la misma localización. Si la calidad del sonido ambiente no es buena, se podrían utilizar dispositivos externos que emitan un patrón predecible para los dispositivos cuya escucha garantice que ambos se encuentran en la misma localización. El problema de esta aproximación es que requiere del despliegue de un sistema de "balizas" que emitan este tipo de señales.

II. TECNOLOGÍAS RELACIONADAS

A continuación se analizan algunas tecnologías relacionadas con el trabajo realizado.

Signal360 [1] (Anteriormente SonicNotify) es una tecnología propietaria en la que participa Oracle, permite que el dispositivo reciba notificaciones con información por ultrasonidos, se orienta a la obtención de información adicional de un evento concreto, por ejemplo, los grupos que está actuando en un festival de música o para fines de marketing y publicidad, sorteos en eventos, compartición de imágenes mediante redes sociales, etc... Desgraciadamente al ser una solución propietaria no cuenta con un SDK (Software Development Kit) o librerías de carácter público.

Audio Modem [2] ha sido desarrollada por la empresa francesa Appdilium, en su página se analizan las distintas frecuencias que podrían ser utilizadas y razona la utilización de un rango de frecuencias de 18.4kHz-20.8kHz (ultrasonidos). Además, utilizan su propia implementación para la modulación de onda, siguiendo el algoritmo DBPSK [7] que permite hacer más robusto el canal gracias a la redundancia de datos, así como facilitar la demodulación con un oyente de tipo no coherente.

En la actualidad las tecnologías basadas en ultrasonidos son el objetivo de grandes empresas, prueba de ello es la publicación de la API Nearby de Google para Android e iOS [3] que combina Bluetooth, Bluetooth Low Energy, WiFi y ultrasonidos para intercambiar códigos de emparejamiento entre dispositivos.

Al utilizar ultrasonidos, los usuarios no pueden identificar a los dispositivos que participan en el emparejamiento. De hecho, recientemente, investigadores de la Universidad de Brunswick han descubierto aplicaciones Android en Google Play Store que hacen uso de técnicas de ultrasonidos en anuncios, para poder relacionar cuales son los dispositivos del usuario con objeto de realizar campañas de marketing dirigido sin el conocimiento del usuario, es lo que se conoce como *"ultrasound Cross-Device Tracking (uXDT)"* [8]. Todo esto nos lleva a buscar tecnologías audibles de forma que los usuarios sean conscientes de los intercambios realizados.

Para el desarrollo del prototipo se recurrió a "Chirp" (ver sección IV) que tiene como finalidad el intercambio de contenido multimedia entre dos o más dispositivos cercanos mediante sonidos similares a los módem analógicos. En realidad lo que se envía a través del sonido es un enlace al archivo, que ha sido alojado en su servidor desde el dispositivo, al recibirlo el otro usuario lo descarga en su aplicación y lo visualiza casi de forma instantánea. Chirp incluye un SDK tanto en Android como en iOS así como en plataformas web. Chirp permite el uso de la SDK durante un periodo de tiempo limitado de forma gratuita a los desarrolladores que lo soliciten. Este fue uno de los puntos que decantó la elección de esta tecnología.

III. SOLUCIONES PROPUESTAS

En el desarrollo de nuestro prototipo hemos asumido que el atacante no tiene acceso de forma simultanea a los dos canales de comunicación que utilizamos: el canal sonoro implementado usando la SDK de Chirp y un canal a través de Internet implementado usando Firebase (ver sección IV). Con respecto al canal de comunicaciones a través de Internet asumimos que todos los usuarios tienen un ID temporal diferente único, por lo que el atacante no puede apoderarse de la sesión de un usuario existe, aunque si puede leer los mensajes de todos los usuarios y enviar mensajes con su propio ID a cualquier usuario del sistema. También puede conseguir nuevos IDs simplemente estableciendo una nueva conexión. Con respecto al canal sonoro o canal fuera de banda, asumimos que el atacante no puede enviar ningún código sonoro, ya que en ese caso, el usuario receptor podría identificar que la fuente del sonido no se corresponde con el dispositivo del otro usuario y abortar el emparejamiento. Estamos por tanto ante un atacante pasivo en el canal fuera de banda, que si podría enviar tráfico en el canal de comunicaciones pero no puede secuestrar sesiones existentes.

Durante el desarrollo del prototipo se han tenido en cuenta diferentes diseños para el protocolo, en función de las capacidades del atacante. También se ha tenido en cuenta una limitación de Chirp en cuanto al envío de información, que no puede superar los 50 bits[4].

[1] http://newatlas.com/sonicnotify-audio-signals/21385/
[2] https://applidium.com/en/news/data_transfer_through_sound/
[3] https://developers.google.com/nearby/
[4] http://developers.chirp.io/docs/online-and-offline-operation

La primera aproximación (Figura 1) consiste en el envío directo de la clave mediante un código sonoro. Esta opción es la más simple de implementar pero presenta el problema de que un usuario que se encuentre en la misma zona de influencia puede tener acceso a la clave e interceptar todas las comunicaciones, por tanto solo sería segura antes atacantes sin acceso al canal fuera de banda.

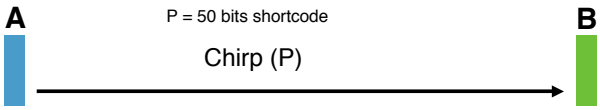

Fig. 1. Primera iteración del protocolo

También se podría impersonar al segundo usuario frente al primero en el canal de comunicaciones al disponer de la clave de comunicaciones. Por tanto solo es recomendable en escenarios donde solo los dos dispositivos tengan acceso al canal sonoro. Por ejemplo, en entornos con mucho ruido de fondo, ajustando el nivel de volumen, la distorsión del ruido haría que un dispositivo que estuviera fuera del rango visual de los usuarios no fuera capaz de recibir correctamente la clave P. Sin embargo, en entornos sin ruido, y con un volumen suficientemente alto, el atacante podría ser capaz de capturar la clave sin ser visto. Otro problema es que la clave al tener solo 50 bits como máximo tiene una entropía limitada.

La primera idea que podríamos plantear para mejorar la situación sería usar un protocolo DH donde cada usuario enviara su clave pública usando el canal fuera de banda, desgraciadamente el limite de dicho canal hace que esta aproximación sea insegura. Por tanto, se definió una segunda aproximación (Figura 2) que intenta resolver esos problemas, implementando el protocolo Diffie Hellman dentro del canal de comunicaciones con una verificación de la clave intercambiada usando el canal fuera de banda.

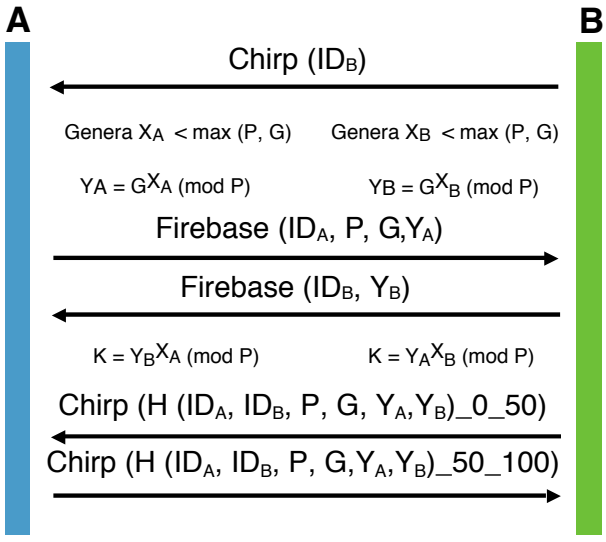

Fig. 2. Segunda iteración del protocolo

En concreto, se realiza la autenticación mediante códigos sonoros de las identidades de los usuarios en canal de comunicaciones, así como de las claves públicas y parámetros del protocolo Diffie Hellman. Para ello se comparte el hash de todos los elementos mencionados anteriormente por el canal fuera de banda, aunque debido a la limitación de 50 bits, se divide el código de verificación en dos partes, teniendo que enviar cada usuario una de las dos partes para que el otro la verifique.

En este protocolo un atacante no podría impersonar a ninguno de los usuarios legítimos, ya que aunque intercepte el ID del primer usuario no podría utilizarlo dentro del canal de comunicaciones con lo que sus mensajes sería descartados por el segundo usuario. Tampoco podría impersonar al segundo usuario porque no sería capaz de enviar el código de autenticación de vuelta usando el canal fuera de banda.

Aunque este esquema es más seguro que el anterior, el principal problema que presenta es que se necesita enviar al menos tres mensajes con códigos sonoros, uno para iniciar la comunicación con el ID y dos más para la autenticación con el HMAC, lo cual introduce problemas de sincronía entre ambos canales y además resulta poco práctico debido al tiempo que tardan en enviarse los mensajes por el canal fuera de banda. Se podría reducir el envío de mensajes a dos, solo para la verificación de la clave, pero en ese caso habría que acompañar a las claves públicas de ambos usuarios de alguna información que le permitiera relacionar ambos mensajes. Una forma de calcular un identificador único compartido entre ambos sería usar una función HASH sobre los dos parámetros siguientes:

1) **Hora actual:** En horas y minutos, de esta forma sabemos que la sincronización se está llevando a cabo en un instante de tiempo determinado. Esto también permitiría evitar *"Replay Attacks"* [9].
2) **Redes WiFi disponibles:** Podemos comprobar que ambos dispositivos se encuentran próximos si detectan las mismas redes WiFi. Para ello utilizamos el SSID de las redes con mayor señal. De esta forma podemos también mitigar los conocidos como *"Wormhole Attacks"* [10] en redes ad-hoc.

Enviado las claves públicas en difusión, acompañadas de este identificador, por el canal de comunicaciones, cada usuario podría identificar la clave pública de la otra parte y completar el intercambio de claves.

Si queremos implementar un prototipo con un solo mensaje por el canal fuera de banda, siempre será posible que el receptor de ese mensaje sea impersonado por un atacante que se encuentre próximo a ambos usuarios. Es decir, el iniciador nunca podrá tener la certeza que el dispositivo que está viendo es el que responde a través del canal de comunicaciones. Lo más que se puede hacer en ese escenario es una validación mutua a posteriori de la identidad de ambos usuarios, usando otro canal fuera de banda como puede ser el visual. Una opción es enviar información personal de cada usuario, como el nombre o alguna foto, que permita a cada usuario reconocerla en el dispositivo de la otra parte. Aún así, un atacante que tenga acceso, aunque solo sea de escucha, a ambos canales de

forma simultanea sería capaz de leer todos los mensajes.

Bajo al asunción de que el atacante no puede observar el canal de fuera de banda, se puede mantener la misma usabilidad de la primera versión, pero usando una clave de mayor calidad. Para ello se definió una tercera aproximación (Figura 3) que hace uso de la función PBKDF2 [11] (Password Based Key Derivation Function) para a partir de la clave de 50 bits enviada por el canal fuera de banda generar una clave de mayor extensión, usando como *"salt"* la hora y las redes wifi disponibles.

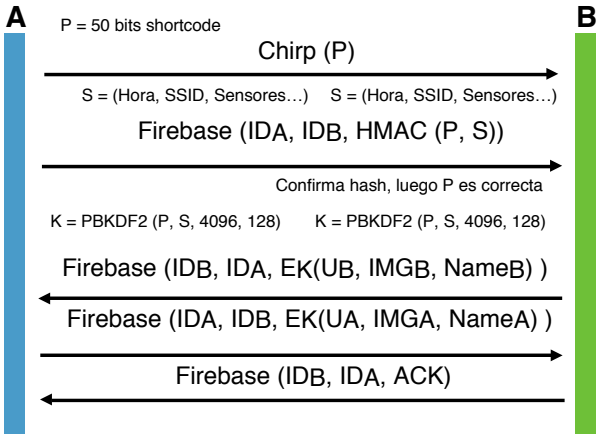

Fig. 3. Tercera iteración del protocolo

Una vez calculada la clave por ambos dispositivos se procede a enviar la información privada del usuario (nombre e imagen) cifradas, así como un código HMAC que servirá para autenticar al receptor y confirmar que la clave se ha calculado correctamente. Si el código HMAC es correcto se procede a añadir la información transmitida como un nuevo contacto y establecer una ID para el canal de comunicación con el mismo. En caso contrario la información se descarta y no se responde con la información propia al mensaje recibido.

La información de los contactos añadidos se almacena de forma local en la aplicación y no es accesible por el servidor, por otra parte las IDs de los usuarios se generan aleatoriamente en cada comunicación, por lo que se almacenan las ID utilizadas por el emisor y el receptor en la sincronización y se emplean (en un orden determinado) para establecer la ID del canal de comunicación entre ellos.

El objetivo de esta sincronización es que únicamente se envíen al servidor dos tipos de información: IDs aleatorias de comunicación y mensajes cifrados. De esta forma, aunque alguien pudiera atacar y observar la información del servidor no podría conocer el contenido de los mensajes enviados, ni siquiera podría identificar usuarios concretos, ya que en cada canal de comunicación utilizan ID distintos, garantizamos así no solo la confidencialidad de los mensajes sino también la privacidad de los usuarios.

En la Tabla I se puede ver un resumen comparativo de los tres esquemas que se han mencionado.

Tabla I
COMPARATIVA DE LOS DISTINTOS PROTOCOLOS PLANTEADOS

Protocolo	Bits de clave	Mensajes	Resiste ambos canales
$Claro$	50	1	No
DH	> 50	3	Si
$PBKDF2$	$50 + salt$	1	No

IV. TECNOLOGÍAS UTILIZADAS

La plataforma sobre la que se desarrolla el prototipo es iOS, esto se debe principalmente a que la primera tecnología que tratamos de utilizar fue Audio Modem que trabaja sobre iOS, aunque posteriormente se optó por Chirp pensando en la portabilidad de la aplicación a otros sistemas operativos.

A. Chirp

Tal y como se ha adelantado, Chirp[5] es una tecnología que permite la comunicación de información entre dispositivos utilizando el canal sonoro. Para ello emplean una asociación unívoca entre un byte y una determinada frecuencia, de forma que el carácter 'a' podría, por ejemplo, tener asociada la frecuencia de 500Hz, el carácter 'b' la de 510Hz sucesivamente.

Distinguen dos tipos de mensajes en su SDK:

1) **Shortcodes:** Son mensajes de un máximo de 50 bits, enviados en forma de 10 caracteres que no pasan por el servidor de Chirp y que se envían a través del canal sonoro de un dispositivo a otro siguiendo la codificación en frecuencias anteriormente explicadas.
2) **Mensajes de diccionario:** Son estructuras de datos más complejas que se envían con una clave, esta clave es, precisamente, un *"shortcode"* de los anteriormente mencionados. Estos mensajes con diccionario son, en última instancia, una clave de 10 caracteres que identifica un mensaje en formato JSON con la información estructurada.

Los mensajes con diccionario, a diferencia de los *"shortcode"*, sí necesitan subirse al servidor de Chirp. En su funcionamiento, el usuario final recibe mediante el canal sonoro un *"shortcode"* que utiliza como *"token"* frente al servidor de Chirp para obtener la estructura de datos en formato JSON.

Además, la SDK de Chirp también permite entre otras cosas la visualización en tiempo real de la onda sonora captada por el micrófono.

B. Firebase

Firebase[6] es una tecnología relativamente nueva, que comenzó como un proyecto independiente y ha sido comprada por Google. Aunque ahora ha integrado muchos servicios con Google podemos definir Firebase en su origen como una "Base de datos NoSQL online basada en JSON". En la actualidad incorpora gran cantidad de servicios adicionales en colaboración con Google como

[5]Chirp: http://chirp.io
[6]Firebase: https://firebase.google.com/

Analytics, Autenticación, Almacenamiento de ficheros, Hosting, Monetización de aplicaciones mediante Google Admob.

Dentro de Firebase la información se estructura en subramas en forma de arbol, con un nodo inicial: nuestro identificador de Firebase. Esto permite definir fácilmente ciertas políticas de control de acceso, por ejemplo, dando acceso a un elemento se concede acceso a todos los elementos de la rama.

C. CyptoSwift

CryptoSwift[7], tal y como su nombre sugiere, es una librería criptográfica en el lenguaje de programación Swift (utilizado conjuntamente con Objective-C a lo largo del desarrollo del proyecto iOS).

Se trata de un proyecto de código abierto en Github que incluye las operaciones básicas de criptografía que se utilizan actualmente, entre ellas: Funciones hash (MD5, SHA1.. 512), CRC, Algoritmos de cifrado (AES128..256, ChaCha20, Rabbit), Códigos de autenticación HMAC (MD5, SHA1..256, Poly1305), Modos de operación en bloque (ECB, CBC, CTR...), Funciones de derivación de claves (PBKDF 1 y 2) y Esquemas de relleno (PKCS5/PKCS7).

Para las aislar las operaciones criptográficas del control de la aplicación hemos creado un interfaz en una clase llamada *"CustomCrypto"* que contiene los métodos necesarios para realizar todas las operaciones criptográficas con CryptoSwift ofreciendo el resultado esperado al control de la aplicación.

D. JSQMessages

Por último, JSQMessages[8] es otro proyecto *"open source"* que conforma una librería de gráficos y control para crear de forma sencilla los elementos básicos y con un diseño estandarizado de una ventana de chat en iOS. Entre los elementos se incluyen diferentes burbujas con mensajes: texto, contenedoras de vídeos o imágenes, localización geográfica. Todo a nivel de interfaz de usuario.

Dado que se trata de una librería compleja y la elaboración de un chat es sólo una excusa para poner de manifiesto la solución teórica propuesta hemos hecho un uso reducido de esta librería limitándonos a la creación de ventanas de chat, burbujas de mensajes e indicadores de "escribiendo...".

V. NUESTRO PROTOTIPO: CHATCHAT

Como aplicación prototipo para la utilización del canal sonoro con Chirp e implementación del protocolo propuesto hemos elaborado una aplicación de chat para dispositivos iOS, que hemos denominado ChatChat.

En la pantalla principal (Figura 4) podemos apreciar tres botones:

1) **General:** Chat general compartido por todos los usuarios de ChatChat. No tiene cifrado.
2) **Profile (Perfil):** Permite editar nuestro perfil, nombre de usuario e imagen.
3) **Contacts (Contactos):** Permite establecer un chat con un usuario sincronizado o sincronizar la información de dos usuarios, poniendo en práctica nuestro protocolo

La ventana de chat general (Figura 5) muestra en burbujas los mensajes enviados al "Chat General" de Firebase. En este chat general pueden enviar mensajes todos los usuarios, usando un nuevo ID cada vez que entremos. Estos mensajes "anónimos" se almacenarán en Firebase en claro y serán visibles para el resto de usuarios de la sala. Nótese que si salimos y volvemos a entrar en el chat general Firebase nos asigna un nuevo ID aleatorio, por lo que no hay forma de identificar los mensajes pertenecientes a un mismo dispositivo o usuario.

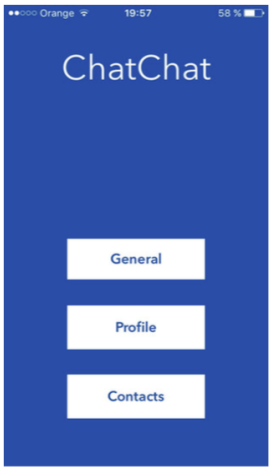

Fig. 4. Menú principal

Fig. 5. Pantalla de chat

En la ventana de perfil (Figura 6) podemos editar la información de usuario: nuestro nombre y nuestra imagen. Esta información se almacena en un fichero local y es persistente aunque cerremos la aplicación. Ésta es también la información que se envía al otro dispositivo después de un intercambio correcto y es la que deben usar los usuarios para verificar que no hay un impostor.

Si en el menú principal pulsamos sobre el botón *"Contacts"* iremos a la ventana de contactos existentes y sincronización de los mismos (Figura 7). En la parte central encontramos un botón *"Add New"* que al pulsarlo pondrá en marcha nuestro protocolo de sincronización, generando el *"shortkey"* de chirp para reproducirlo a través del altavoz del dispositivo, calculando la clave con PBKDF (fecha + WiFi) y enviando los mensajes correspondientes a través de Firebase. En esta ventana y sin haber pulsado el botón nos encontraríamos en la situación de receptor de la sincronización (inicialmente ambos), estando suscrito a una rama de mensajes de Firebase que se utiliza en exclusiva para gestionar las sincronizaciones. En la parte inferior de la pantalla, de un color azul más claro, hemos incluido la visualización en tiempo real de la captación de sonido del micrófono.

Después de un intercambio exitoso en ambos dispos-

[7]CryptoSwift: https://github.com/krzyzanowskim/CryptoSwift
[8]https://github.com/jessesquires/JSQMessagesViewController/tree/master

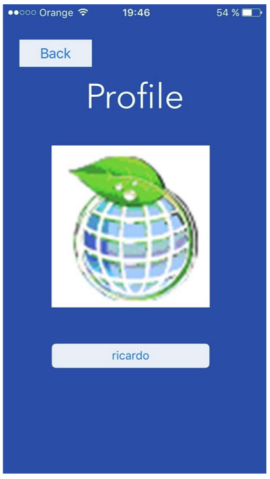

Fig. 6. Pantalla de perfil.

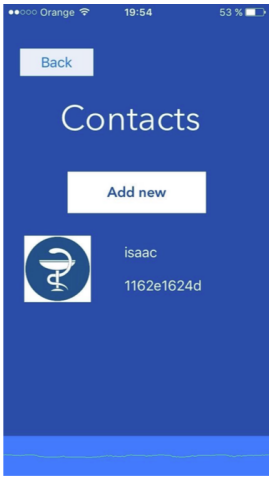

Fig. 7. Pantalla de contactos.

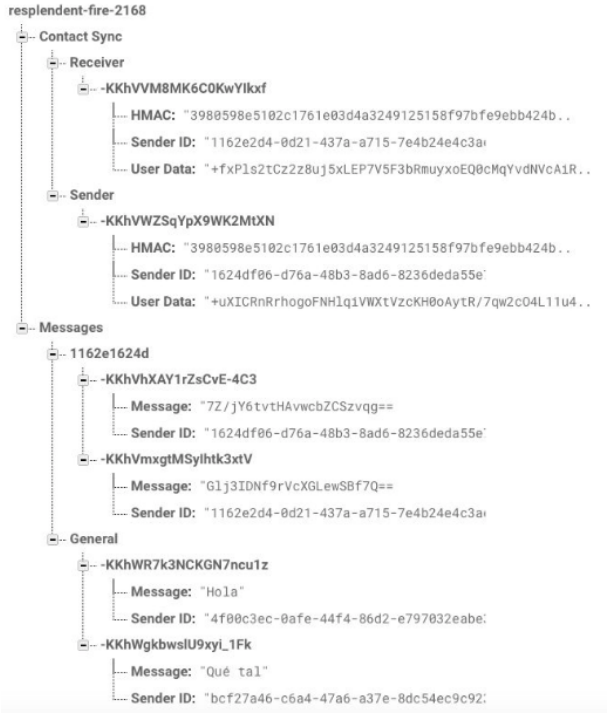

Fig. 8. Esquema de chat de Firebase

itivos se agregará el dispositivo opuesto como contacto. El número inferior al nombre de usuario es la ID del canal (construida a partir de la ID de ambos contactos en la sincronización) que será la rama en Firebase en la que ambos realizarán la comunicación. Si tocamos la imagen del contacto agregado podemos acceder a tener una conversación con él en una ventana de chat similar a la del canal general.

Distinguimos en nuestro esquema de Firebase (Figura 8) dos grandes ramas:

1) **Contact Sync:** Es la rama en la que se envían los mensajes de sincronización. Tiene dos subramas: Receiver y Sender, en función de quién haya iniciado la sincronización con Chirp y quien haya recibido el mensaje sonoro. Se envía la información de usuario cifrada, el HMAC y el ID aleatorio
2) **Messages:** Es la rama dedicada al envío de mensajes, distinguimos una rama General en la que podemos observar que se envían los mensajes en claro (texto ”Hola” y ”Que tal”) y una rama con un ID de canal coincidente con el ID que aparecía debajo del nombre del contacto agregado constituido por el ID aleatorio que ambos tenían. En esta subrama (canal de contactos sincronizados) podemos observar que los mensajes se envían efectivamente cifrados

De esta forma, por el servidor de Firebase únicamente pasan dos tipos de información: IDs aleatorios cada vez que establecemos un canal de comunicación (que garantizan la privacidad de los usuarios al no poder identificarse con los mensajes enviados) y mensajes cifrados. También se almacenan mensajes en claro, pero únicamente en el canal general.

VI. CONCLUSIONES Y TRABAJO FUTURO

Podemos concluir sobre este proyecto que los resultados que pretendíamos alcanzar son viables con la tecnología actual y se ha conseguido implementar un prototipo funcional de código abierto que está disponible en la plataforma GitHub [9].

Por desgracia, la usabilidad está reñida con el nivel de seguridad que se deseé obtener por lo que uno de los puntos a mejorar sería la tecnología para el envío de información utilizando el canal sonoro, que aún está en vías de desarrollo y, como hemos visto, en Chirp tiene una extensión máxima de 50 bits.

En cuanto a las posibles mejoras sobre el prototipo elaborado, una de las deficiencias de nuestra propuesta es que no proporcionan ”Perfect Forward Secrecy (PFS)” de forma directa. Es decir, si un atacante consiguiera hacerse con la clave privada de un canal, mediante el robo de uno de los dispositivos, podría descifrar todos los mensajes pasados de ese canal. Una forma fácil de solucionar este problema, sería usar la clave negociada en la sincronización de los dispositivos solamente como mecanismo de autenticación y no para establecer un canal confidencial. De esta forma, se requeriría una fase extra que podría consistir en un protocolo Diffie-Hellman donde al final, ambos usuarios utilizan la clave secreta negociada usando el canal sonoro para confirmar que no ha habido intermediarios en el protocolo Diffie-Hellman.

Otra línea de trabajo futuro sería integrar este desarrollo con alguna aplicación de mensajería instantánea, como por ejemplo Telegram.

REFERENCIAS

[1] Boyd C., Mathuria A: ”Key establishment protocols for secure mobile communications: A selective survey”. In: Boyd C., Dawson E. (eds) Information Security and Privacy. ACISP 1998. Lecture Notes in Computer Science, vol 1438. Springer, Berlin, Heidelberg

[9] ChatChat: https://github.com/RicardoRuizTueros/ChatChat

[2] Whitfielf Diffie, Martin E. Hellman, , "New Directions in Cryptography", 644 IEEE Transactions on Information Theory, Vol. IT-22, No. 6, 1976
[3] N. Unger et al., "SoK: Secure Messaging," 2015 IEEE Symposium on Security and Privacy, San Jose, CA, 2015, pp. 232-249. doi: 10.1109/SP.2015.22
[4] Roland, Michael, and Josef Langer. "Cloning Credit Cards: A Combined Pre-play and Downgrade Attack on EMV Contactless." WOOT. 2013.
[5] Sebastian U. et al. "Tactile One-Time Pad: Leakage-Resilient Authentication for Smartphones", Ruhr-University Bochum, Germany, 2015
[6] Nikolaos K et al. "Sound-Proof: Usable Two-Factor Authentication Based on Ambient Sound", 24th USENIX Security Symposium, Washintong DC, 2015
[7] F. Gardner "A BPSK/QPSK Timing-Error Detector for Sampled Receivers", IEEE Transactions on Communications. Volume: 34, Issue: 5, 1986
[8] Daniel A. et al. "Privacy Threats through Ultrasonic Side Channels on Mobile Devices", Technische Universität Braunschweig, Brunswick, Germany. 2017
[9] Han G. et al. "A Formal Analysis for Capturing Replay Attacks in Cryptographic Protocols", Informatics and Mathematical Modelling, Technical University of Denmark and Dipartimento di Informatica, Universita di Pisa
[10] Mariano G, Adrián P. "Detection of wormhole attacks in wireless sensor networks using range-free localization", IEEE 17th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD). 2012
[11] K. Moriarty et al. "PKCS 5: Password-Based Cryptography Specification Version 2.1", RFC 8018, Internet Engineering Task Force (IETF), 2017