Universidad de La Laguna

Sección de Matemáticas | Facultad de Ciencias

Ashneet Khandpur Singh

Estudio de la Herramienta Criptográfica Signal

Study of the Cryptographic Signal Tool

Trabajo Fin de Grado
Grado en Matemáticas
La Laguna, Marzo de 2018

Dirigido por

Pino Caballero Gil
Candelaria Hernández Goya

Pino Caballero Gil
Departamento de Ingeniería Informática y de Sistemas
Universidad de La Laguna
38271 La Laguna, Tenerife

Candelaria Hernández Goya
Departamento de Ingeniería Informática y de Sistemas
Universidad de La Laguna
38271 La Laguna, Tenerife

Agradecimientos

Me gustaría expresar mi reconocimiento y agradecimiento a todas aquellas personas que, gracias a su colaboración, han contribuido a la realización de este Trabajo Fin de Grado:
En primer lugar, mi sincero agradecimiento a Pino y Candelaria, tutoras de este proyecto, por su comprensión ante dificultades de día a día , consejos y dedicación durante el desarrollo de este trabajo.
Por otro lado, agradecer a mis padres, mi hermana y amigos, la paciencia, apoyo y comprensión que me han otorgado siempre que lo he necesitado, dándome fuerza a poder conseguir mis objetivos, confiando en mí.

Ashneet Khandpur Singh

Resumen Abstract

Resumen

La presente memoria se centra en el nuevo protocolo criptográfico de seguridad, Signal, que proporciona cifrado extremo a extremo para mensajería instantánea. El protocolo está en uso en varias aplicaciones populares como WhatsApp, Facebook Messenger y Google Allo. El objetivo principal de este trabajo es comprender Signal y su funcionamiento. Para ello, se estudian todos los algoritmos y cifrados que lo componen: Diffie-Hellman, Double Ratchet, Curve25519, AES-256 y HMAC-SHA256. Y finalmente, se realiza un breve análisis de la seguridad del protocolo y una implementación beta de curvas elípticas.

Palabras clave: *protocolo – Signal – cifrado extremo a extremo – algoritmos – WhatsApp*

Abstract

The present project focuses on the new cryptographic security protocol, Signal, that provides end-to-end encryption for instant messaging. The protocol is in use in many popular apps like WhatsApp, Facebook Messenger and Google Allo. The main goal of this paper is to understand Signal and how it functions. For that purpose, a study has been done on the algorithms and ciphers that are part of the protocol: Diffie-Hellman, Double Ratchet, Curve25519, AES-256 and HMAC-SHA256. Also, a brief analysis has been performed on the security of Signal. And finally, an implementation of elliptic curves.

Keywords: *protocol – Signal – end-to-end encryption – algorithms – WhatsApp.*

Contenido

1

Introducción

En este capítulo se incluye una introducción acerca del planteamiento del proyecto, así como su estructura.

1.1. Motivación

En la actualidad, las tecnologías se han vuelto una parte esencial de nuestras vidas. Para entender el por qué, solo basta con mirar a nuestro alrededor y ver que en todo momento y contexto estamos rodeados por ellas.
La seguridad y privacidad son cuestiones que cada vez preocupan más a los usuarios de estas tecnologías. Es así que ahora ha surgido un nuevo nicho en las aplicaciones de comunicaciones centradas en la seguridad de la información del usuario. Aplicaciones de mensajería instantánea como *Signal Private Messenger* ofrecen encriptación y la capacidad de destruir los mensajes enviados, optando los usuarios a tener un mayor control de su información.
Esto ha sido posible gracias a la herramienta criptográfica Signal, desarrollada por Open Whisper Systems, que estudiaremos a lo largo de este documento.

1.2. Objetivos

El principal objetivo de este proyecto es realizar un estudio de los distintos cifrados y algoritmos que componen el protocolo criptográfico Signal, para después poder comprender su aplicación en las diferentes tecnologías que se utilizan en el día a día, entre ellas, WhatsApp, Facebook Messenger y Google Allo.

1.3. Fases

El desarrollo del proyecto se ha dividido en diferentes partes que se han desarrollado a lo largo del período de ejecución planteado.

- ◇ Recopilación de información en diferentes medios.
- ◇ Análisis extendido del funcionamiento de cada uno de los algoritmos contenidos en el protocolo.
- ◇ Una vez familiarizado con el protocolo, se realizó un estudio del mismo en las diferentes tecnologías y como éstas usan sus características.
- ◇ Estudio de la seguridad de la herramienta criptográfica Signal.
- ◇ Implementación beta de curvas elípticas.

1.4. Estructura de la memoria

Además de este primer capítulo introductorio, esta memoria consta de otros seis capítulos. En el segundo y tercero se estudian cada uno de los cifrados y algoritmos, de clave pública y privada, de los que consta el protocolo Signal, así como el algoritmo Diffie-Hellman, Double Ratchet, AES256, Curve25519 y HMAC-SHA256. En el cuarto se realiza una descripción breve de Signal y las tecnologías que lo implementan, entre ellas, WhatsApp, una de las aplicaciones más popular del mundo. El capítulo cinco se centra en la seguridad del protocolo, estudiando sus características y algunos ataques que pueden ocurrir. Posteriormente, se realiza una implementación de curvas elípticas usando el lenguaje de programación C. El último capítulo finaliza con las conclusiones obtenidas de este estudio.

2

Gestión de claves

Al conjunto de procedimientos y técnicas que permiten que lo escrito solamente sea legible para quien sepa descifrarlo se les denomina *criptosistemas*, que usan un algoritmo que se compone de dos fases, la primera encripta el mensaje con cierta clave, transformándolo de tal modo que sea ilegible, a menos que se tenga la clave de desencriptado. La segunda fase del algoritmo se encarga del uso de esta clave de desencriptado, que será secreta y nos posibilita la obtención del mensaje. Podemos distinguir dos tipos de criptosistemas según sus claves: simétrico o de clave privada, cuando se usa la misma clave para encriptar y desencriptar, y asimétrico o de clave pública, cuando usa dos claves diferentes. En este caso la clave de encriptado se hace pública, mientras que la clave de desencriptado se mantiene privada. Estos criptosistemas de clave pública serán el objeto de estudio de este capítulo.

2.1. Intercambio de claves de Diffie-Hellman

El protocolo criptográfico de Diffie-Hellman (en honor a sus creadores, Whitfield Diffie y Martin Hellman [12] [6]) permite acordar una clave secreta entre dos partes que no han tenido contacto previo, a través de un canal inseguro y enviando únicamente dos mensajes. La clave secreta resultante no puede ser descubierta por un atacante, aunque éste obtenga los dos mensajes enviados por el protocolo.
Este algoritmo resuelve el siguiente dilema:
Dos partes, Alice y Bob, quieren compartir una clave secreta para su uso en el cifrado simétrico, pero su único medio de comunicación es inseguro. Cada parte de información que intercambian es observada por su adversario Eve. La seguridad, para que Alice y Bob puedan compartir la clave sin que sea asequible a Eve, radica en la extrema dificultad del Problema del Logaritmo Discreto (PLD)

para cuerpos finitos ($\mathbb{F}_p^*$). Si se usa un número primo p muy grande, el problema será muy difícil de resolver incluso computacionalmente.

2.1.1. El problema del logaritmo discreto

Definición 2.1. *Sea G un grupo finito. El Problema del Logaritmo Discreto es el problema de resolver una ecuación del tipo*

$$a^x = b \tag{2.1}$$

con $x \in \mathbb{N}$ *y* $a, b \in G$.

La dificultad de este problema varía según el grupo G que elijamos. Por ejemplo, si G es $\mathbb{Z}_n$, el problema es resolver una ecuación de congruencias del tipo $ax \equiv b$ mód n, que no es más que resolver la ecuación diofántica $ax + ny = b$ utilizando el Algoritmo de Euclides.
Sin embargo, si G es el grupo de las unidades de $\mathbb{Z}_n$ ó $\mathbb{F}_p$, el problema es considerablemente más difícil, lo que garantiza la seguridad de los sistemas criptográficos como DH, ElGamal, curvas elípticas y DSA (Digital Signature Algorithm).
Además de todo esto, el grupo G escogido tiene que tener un algoritmo rápido de cálculo para realizar las operaciones definidas en él.

Ejemplo 2.2. Sea $\mathbb{F}_{17}^*$ el grupo multiplicativo de los enteros módulo 17. Supongamos que tenemos que resolver la ecuación

$$3^x \equiv 13(\text{mód}17)$$

Como $13 \equiv 3^4$ (mód 17), el logaritmo discreto de 13 en base 3 es 4.

Este problema del logaritmo discreto es una operación inversa a la exponenciación en un grupo. La exponenciación modular es una operación sencilla y requiere de métodos eficientes de calcularla como el que se expone a continuación.

Algoritmo de Exponenciación Rápida

Supongamos que se desea calcular a^b (mód m). En primer lugar, hay que escribir el exponente en binario:

$$b = 2^0 b_0 + 2^1 b_1 + ... + 2^{n-1} b_{n-1} = \sum_{i=0}^{n-1} 2^i b_i$$

Se calculan los productos a^{2^j} con $j = 0$ hasta $n - 1$, siendo n el número de bits que representan el valor b en binario.
De los productos calculados, sólo se toman en cuenta en los que en la posición j del valor b en binario aparece un 1.

Ejemplo 2.3. Sea $\mathbb{F}_{221}^*$ el grupo multiplicativo de los enteros módulo 221. Se quiere calcular 12^{37}.
Utilizando el algoritmo anterior, $a = 12$ y $b = 37$, la representación en binario de 37 es 100101.
Por lo tanto, $x = 12^{37}$ (mód 221)$= 12 \cdot 183 \cdot 1$(mód 221) $= 207$

j	b_j	a^{2^j}
0	1	12
1	0	144
2	1	183
3	0	118
4	0	1
5	1	1

Los siguientes son algunos de los algoritmos existentes para el cálculo del logaritmo discreto [19].

- ◇ Fuerza bruta
- ◇ Index-Calculus
- ◇ Baby Step-Giant Step
- ◇ Algoritmo ρ de Pollard
- ◇ Algoritmo de Pohlig-Hellman

2.1.2. Algoritmo Diffie-Hellman

La idea básica del algoritmo de Diffie-Hellman es:

- ◇ Alice y Bob se ponen de acuerdo en un número primo p y un entero g ($\neq 0$) módulo p. Estos dos valores son públicos.
- ◇ Alice escoge un número secreto a, sólo conocido por ella, calcula g^a(mód p) y envía el resultado a Bob. Éste realiza el mismo procedimiento pero con el número secreto b. Ahora, tenemos que

$$A \equiv g^a(\text{mód p}) \qquad \text{y} \qquad B \equiv g^b(\text{mód p})$$

  Eve conoce los valores A y B, ya que han sido enviados por un canal de comunicación inseguro.
- ◇ Finalmente, Alice y Bob usan sus números secretos para calcular

$$B^a(\text{mód p}) \qquad \text{y} \qquad A^b(\text{mód p})$$

El resultado que obtienen Alice y Bob son iguales, puesto que

$$(g^a(\text{mód p}))^b(\text{mód p}) = g^{ab}(\text{mód p})$$
$$(g^b(\text{mód p}))^a(\text{mód p}) = g^{ba}(\text{mód p})$$

El valor resultante a ambos lados de la comunicación será la clave generada. Veamos lo explicado anteriormente con un ejemplo:

Ejemplo 2.4. Alice y Bob deciden usar el número primo $p = 941$ y el entero $g = 627$. Alice elige la clave secreta $a = 3474$y calcula $A \equiv 627^{347}(\text{mód } 941) = 390$. Similarmente, Bob elige $b = 781$ y calcula $B \equiv 627^{781}(\text{mód } 941) = 691$. Alice envía a Bob su resultado A y Bob envía a Alice su resultado B. Estos dos resultados A y B son públicos, ya que han sido enviados por un canal inseguro, mientras que a y b solo son conocidos por Alice y Bob, respectivamente. Entonces, Alice y Bob son capaces de calcular

$$470 \equiv 627^{347*781} \equiv A^b \equiv B^a(\text{mód } 941)$$

y, por tanto, 470 es la clave secreta compartida por los dos.

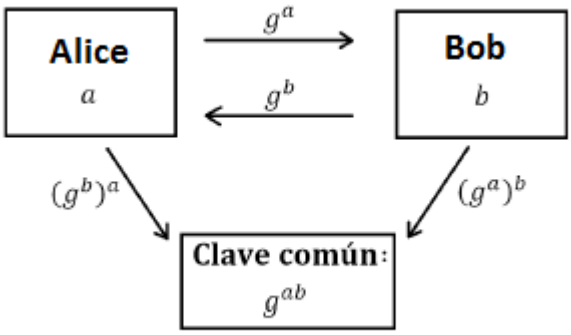


Figura 2.1: Esquema del algoritmo Diffie-Hellman

El algoritmo de Diffie-Hellman fue publicado en 1976 [12]. Actualmente se conoce que es vulnerable a ataque de 'hombre en medio' (Man in the Middle, abreviado MitM): un atacante podría situarse entre ambas partes y acordar una clave simétrica con cada una de la partes, haciéndose pasar por la parte A de

cara a la parte B y viceversa. Una vez establecidas las dos claves simétricas, el atacante haría de puente entre las dos partes, descifrando toda la comunicación y volviéndola a cifrar para enviársela a la otra parte. Por esta razón, la limitación fundamental de este protocolo es que no hay autenticación de los mensajes enviados, es decir, no se puede tener la certeza de quien se está comunicando con una parte.
El protocolo DH es muy usado en los diferentes sistemas de seguridad implementados en Internet, como SSL (Secure Socket Layer) y VPN (Virtual Private Network). Además se usa en WhatsApp (mensajería por móvil), OpenSSL (software libre de cifrados y firma) y Tor (red para proteger anonimato).

2.1.3. Triple Diffie-Hellman handshake

El protocolo Signal utiliza una variante del intercambio de claves de Diffie-Hellman llamada Extended Triple Diffie-Hellman (X3DH). Es un protocolo de establecimiento de claves, especialmente adecuado para su uso en la comunicación asíncrona, donde una parte puede estar *'offline'*(desconectada) durante el intercambio de mensajes. X3DH usa tres fases del algoritmo Diffie-Hellman en una forma que permite a los usuarios comunicarse de forma segura, proporcionando confidencialidad, integridad y autenticidad (propiedades de seguridad que se estudiarán en el capítulo 4).
Triple Diffie-Hellman involucra tres partes: Alice, Bob y un servidor, y tiene tres fases:

- Alice quiere enviar a Bob datos iniciales usando encriptación, y además establece una clave secreta compartida, que será usada para la comuniación bidireccional.

- Bob permite a Alice establecer una clave compartida con él y envía datos cifrados. Sin embargo, Bob puede estar desconectado en el momento que Alice intenta intercambiar la clave. Para que este intercambio sea posible, se necesita una tercera parte, el servidor.

- El servidor almacena los mensajes de Alice, que Bob recupera posteriormente.

X3DH está diseñado de una manera que previene al servidor de comprometer comunicaciones entre el usuario.

2.2. Double Ratchet Algorithm

Es un algoritmo para la gestión de claves, que permite que tras un intercambio inicial de clave, se gestione la renovación continua de claves de sesión de

corta duración. Fue desarrollado por Trevor Perrin y Moxie Marlinspike en 2013 e introducido como parte del Protocolo Signal en febrero del 2014 [24].
Inicialmente este algoritmo era llamado Axolotl Ratchet, pero los desarrolladores lo renombraron a Double Ratchet Algorithm (DRA) para diferenciarlo del protocolo entero, ya que algunos utilizaban el nombre Axolotl refiriéndose al Protocolo Signal.
El algoritmo Double Ratchet combina un ratchet (trinquete) criptográfico basado en el intercambio de claves de Diffie-Hellman con una función de derivación de claves (Key Derivation Function, abreviado KDF), como por ejemplo, una función hash (véase 3.1).

Cadenas KDF

Una función de derivación de claves (KDF) es un componente básico y esencial de sistemas criptográficos. Es una función criptográfica que toma datos de entrada y deriva de ellos una o más claves criptográficamente secretas. El uso más común de funciones de derivación de claves es en contraseñas.
Se usa el término de cadena KDF cuando una parte de la salida (output) de una función de derivación de clave es usada como una clave de salida y la otra para reemplazar la clave KDF, que puede ser usada con otra entrada (input). Una cadena KDF protege claves de mensajes de Alice y Bob incluso cuando un adversario tenga datos de entrada de la función de derivación de clave, porque una clave nunca es utilizada dos veces. La figura 2.2 representa una cadena KDF procesando tres entradas y produciendo tres claves de salida.

En una sesión Double Ratchet entre Alice y Bob, cada uno almacena una clave KDF para tres cadenas: cadena raíz (root chain), cadena de envío (sending chain) y cadena recibida (la cadena enviada por Alice coincide con la cadena recibida por Bob, y viceversa).

Mientras Alice y Bob intercambian mensajes, también intercambian nuevas claves públicas Diffie-Hellman, y los datos secretos de salida se convierten en entradas a la *root chain*. Las claves de salida de la *root chain* se convierten en nuevas claves KDF para las cadenas de envío y recepción. A esto se llama Diffie-Hellman ratchet.
Las cadenas de envío y recepción avanzan a medida que se envía y recibe cada mensaje. Sus claves de salidas son usadas para cifrar y descifrar mensajes. Esto se denomina *symmetric-key ratchet*.
Combinando estos dos ratchets obtenemos el Double Ratchet:

- Cuando un mensaje es enviado o recibido, se aplica el *symmetric-key ratchet* a la cadena de envío o recepción para derivar la clave del mensaje.

◇ Cuando se recibe una nueva clave pública, se realiza un paso DH ratchet antes del symmetric-key ratchet para reemplazar las claves de cadena.

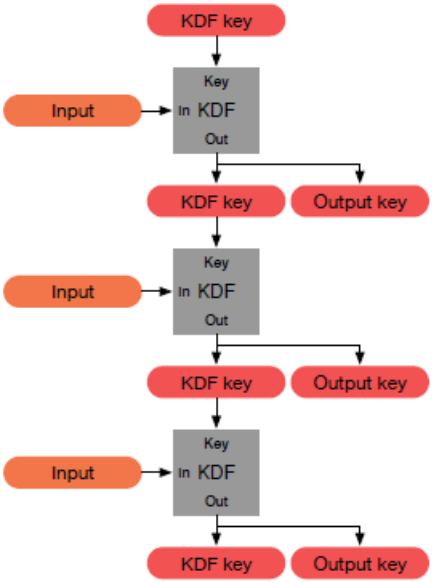


Figura 2.2: Cadena KDF

El algoritmo Double Ratchet está diseñado para proporcionar seguridad contra un atacante que graba mensajes cifrados y luego compromete al remitente o receptor un tiempo más tarde. Esta seguridad podría ser eliminada si se pudieran recuperar las claves eliminadas por un atacante con acceso de bajo nivel al dispositivo comprometido.
El DRA puede ser usado como parte de un protocolo criptográfico para proporcionar cifrado punto a punto para mensajes instantáneos.
Algunas de las aplicaciones que usan este algoritmo son: Facebook Messenger, Google Allo, Signal, WhatsApp, Wire y G Data Secure Chat.

2.3. Elliptic Curve Criptography

2.3.1. Curvas elípticas

Definición 2.5. *Sean* $\mathbb{K}$ *un cuerpo y* $a_1, a_2, a_3, a_4, a_6 \in \mathbb{K}$. *Una curva elíptica* E *sobre* $\mathbb{K}$ *es una curva proyectiva no singular, admitiendo una ecuación definida sobre* $\mathbb{K}$, *denominada* ***ecuación general de Weierstrass***

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \tag{2.2}$$

junto con un punto $\mathcal{O}$ *qe se denomina* ***punto del infinito***.

Observación. El punto del infinito, denotado $\mathcal{O}$ se considerará situado en lo alto del eje y. Así, una recta pasará por el infinito cuando sea vertical. En el desarrollo anterior, corresponde al punto $P = [0, 1, 0]$, ya que si hacemos $z = 0$, obtenemos $x^3 = 0$, luego $x = 0$.

Antes de continuar con la proposición, es necesario recordar el siguiente concepto.

Definición 2.6. *Diremos que un cuerpo* $\mathbb{K}$ *(o un anillo) tiene* ***característica*** n *si* n *es el menor número natural tal que* $1 + 1 + \ldots^n \ldots + 1 = 0$, *es decir, si* $n1_A = 0$ *para todo* $a \in A$. *Si esta suma nunca se anulase, se dice que el cuerpo tiene característica cero. La notación habitual es* $char(K) = n$.

Proposición 2.7. *Sea* $\mathbb{K}$ *un cuerpo con* $char(K) \neq 2, 3$. *Usando transformaciones lineales de las variables, la ecuación 2.2 se puede expresar como*

$$y^2 = x^3 + Ax + B, \tag{2.3}$$

con $A, B \in \mathbb{K}$ *y* $4A^3 + 27B^2 \neq 0$. *Esta ecuación se denomina* ***ecuación reducida de Weierstrass***.

Demostración. Partiendo de la ecuación generalizada 2.2 podemos dividir entre 2 y completar cuadrados:

$$(y + \frac{a_1x}{2} + \frac{a_3x}{2})^2 = x^3 + (a_2 + \frac{a_1^2}{4})x^2 + (a_4 + \frac{a_1a_3}{2})x + (a_6 + \frac{a_3^2}{4}).$$

Haciendo un cambio de variable

$$y_1 = y + \frac{a_1x}{2} + \frac{a_3x}{2}$$

y poniendo

$$a_2' = a_2 + \frac{a_1^2}{4}, a_4' = a_4 + \frac{a_1a_3}{2}, a_6' = a_6 + \frac{a_3^2}{4},$$

obtenemos

$$y_1^2 = x^3 + a_2'x^2 + a_4'x + a_6'.$$

Como $char(K) \neq 3$ podemos completar cubos tomando $x_1 = x + \frac{a_2'}{3}$ obteniendo:

$$y_1^2 = x_1^3 + Ax_1 + B,$$

que es lo que queríamos.

Nota:. La cantidad $\Delta = 4A^3 + 27B^2$ se llama el *discriminante* de la ecuación de Weierstrass. La curva definida por una ecuación de Weierstrass es una curva no singular si y solamente si el discriminante es no nulo.

En la Figura 2.5 se muestran las gráficas de dos curvas elípticas sobre $\mathbb{R}$. La primera tiene un único punto en el eje de abscisas mientras que la segunda tiene tres.

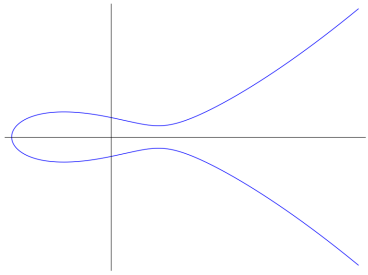


Figura 2.3: $E_1 : Y^2 = X^3 - 3X + 3$

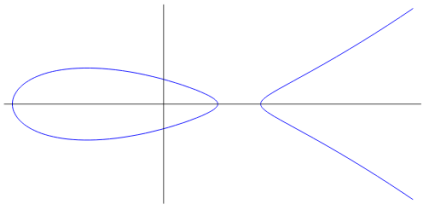


Figura 2.4: $E_2 : Y^2 = X^3 - 6X + 5$

Figura 2.5: Gráficas de dos curvas elípticas definidas sobre $\mathbb{R}$

Suma de puntos

La finalidad de este apartado es demostrar que las curvas elípticas forman un grupo abeliano. Para ello necesitamos definir una operación de suma y de-

mostrar las propiedades pertinentes. A partir de ahora la notación $P = (x, y)$ denotará un punto afín de la curva elíptica E.

Proposición 2.8. *(Elliptic curve point addition and point doubling)[21]*
Sea E una curva elíptica definida por la ecuación 2.3. Sean $P_1 = (x_1, y_1)$ y $P_2 = (x_2, y_2)$ puntos de E con $P_1, P_2 \neq \mathcal{O}$. Entonces si $x_1 = x_2$ pero $y_1 \neq y_2$ ó $P_1 = P_2$ e $y_1 = 0$, entonces $P_1 + P_2 = \mathcal{O}$. En otro caso, $P_1 + P_2 = P_3 = (x_3, y_3)$, con

$$x_3 = m^2 - 2x_1,$$

$$y_3 = m(x_3 - x_1) - y_1$$

donde

$$m = \begin{cases} \frac{3x_1^2 + A}{2y_1} & six_1 = x_2 \\ \frac{y_2 - y_1}{x_2 - x_1} & six_1 \neq x_2 \end{cases}$$

Proposición 2.9. *Sea E una curva elíptica. La suma de puntos definida anteriormente cumple las siguientes propiedades:*

1. *Elemento neutro $P + \mathcal{O} = \mathcal{O} + P = P$ para todo $P \in E$.*
2. *Elemento inverso Dado $P \in E$, $P' \in E$ tal que $P + P' = \mathcal{O}$. Este punto P' suele denotarse $-P$.*
3. *Asociatividad $(P_1 + P_2) + P_3 = P_1 + (P_2 + P_3)$ para todo $P_1, P_2, P_3 \in E$.*
4. *Conmutatividad $P_1 + P_2 = P_2 + P_1$ para todo $P_1, P_2 \in E$.*

Curvas elípticas sobre cuerpos finitos

Para aplicar la teoría de curvas elípticas a la criptografía, trabajaremos con curvas elípticas definidas sobre cuerpos finitos $\mathbb{K} = \mathbb{F}_p$.

Definición 2.10. *Se define una curva elíptica sobre $\mathbb{F}_p$ a la ecuación de la forma*

$$E : Y^2 = X^3 + AX + B$$

con $A, B \in \mathbb{F}_p$ satisfaciendo $A^3 + 27B^2 \neq 0$ y denotamos por

$$E(\mathbb{F}_p) = (x, y) : x, y \in \mathbb{F}_p verificay^2 = x^3 + Ax + B \cup \mathcal{O}$$

a los puntos en E con coordenadas en $\mathbb{F}_p$.

Ejemplo 2.11. Consideramos la curva elíptica $E : Y^2 = X^3 + 3X + 8$ *sobre* $\mathbb{F}_{13}$. *Podemos encontrar los puntos de* $E(\mathbb{F}_p)$ *sustituyendo todos los posibles valores* $X = 0, 1, 2, 3, ..., 12$ *y comprobando para qué valor* X *la cantidad* $X^3 + 3X + 8$ *es un cuadrado módulo* 13.
Por ejemplo, $X = 0$ *da* 8, *y* 8 *no es un cuadrado módulo* 13.
Seguimos con $X = 1$, $1 + 3 + 8 = 12$, *entonces, vemos que* 12 *es un cuadrado módulo* 13 *y tiene 2 raíces cuadradas.*

$$5^2 \equiv 12 \textit{(mód 13)}$$

$$8^2 \equiv 12 \textit{(mód 13)}$$

Con esto obtenemos dos puntos $(1, 5)$ *y* $(1, 8)$ *en* $E(\mathbb{F}_p)$.
Continuando de esta forma, completamos la lista,

$$E(\mathbb{F}_p) = \mathcal{O}, (1, 5), (1, 8), (2, 3), (2, 10), (9, 6), (9, 7), (12, 2), (12, 11)$$

$\therefore E(\mathbb{F}_p)$ *consiste de 9 puntos.*

Teorema 2.12 (Teorema de Hasse). *Sea* E *una curva elíptica sobre* $\mathbb{F}_p$. *Entonces,*

$$E(\mathbb{F}_p) = p + 1 - t_p$$

con t_p *verificando* $|t_p| \leq 2\sqrt{p}$.

La demostración a este teorema se encuentra en [27].

Definición 2.13. *La cantidad* $t_p = p + 1 - E(\mathbb{F}_p)$ *del teorema 2.12 se llama traza de Frobenius para* $E/\mathbb{F}_p$.

En función del valor de su traza de Frobenius, las curvas elípticas pueden ser catalogadas según estas formas: supersingular, anómala y curva FR. Las distintas denominaciones permiten, entre otras cosas, diferenciar si son vulnerables a distintos tipos de ataques al logaritmo discreto elíptico mediante emparejamientos, conocidos como algoritmos de reducción.

2.3.2. Criptografía con curvas elípticas

En 1985, Koblitz y Miller propusieron, de manera independiente, la utilización del grupo de puntos de una curva elíptica definida sobre un cuerpo finito como base para criptosistemas basados en la dificultad de romper el logaritmo discreto. [20]

Algoritmo del logaritmo discreto sobre curvas elípticas

Definición 2.14. *Sea E una curva elíptica sobre $\mathbb{F}_p$ y sean P y Q puntos en $E(\mathbb{F}_p)$. El problema del logaritmo discreto en curvas elípticas (ECDLP) es el problema de encontrar un entero n tal que $Q = nP$. Por analogía con el DLP en $\mathbb{F}_p^*$, denotamos a este entero n por*

$$n = log_P(Q)$$

y llamamos a n el logaritmos discreto elíptico de Q con respecto a P.

Protocolo Diffie-Hellman basado en curvas elípticas

Elliptic curve Diffie-Hellman (ECDH) es un protocolo de intercambio de claves que permite a dos individuos que disponen de una clave pública y otra privada, establecer un secreto compartido. Este secreto puede ser directamente usado como clave o para derivar una clave. En particular, es una variante del protocolo Diffie-Hellman (DH) usando curvas elípticas. Veamos los pasos para obtener ese secreto compartido:

- ◇ *Tanto el emisor como el receptor B acuerdan una curva elíptica E sobre un cuerpo finito $\mathbb{Z}_p$ suficientemente segura y acuerdan un punto $G \in E(\mathbb{Z}_p)$ tal que el subgrupo generado por G sea de un orden grande.*
- ◇ *A elige un entero aleatorio a y envía $P_A = aG$.*
- ◇ *B elige un entero aleatorio b y envía $P_B = bG$.*
- ◇ *A calcula $aP_B = abG$.*
- ◇ *B calcula $bP_A = abG$.*
- ◇ *Finalmente se tiene que la coordenada x de abG es el secreto compartido.*

Observacion. La seguridad de este secreto radica en la dificultad de resolver el problema del logaritmo discreto, es decir, conociendo G, aG y $bG \in E(\mathbb{Z}_p)$, calcular abG. Para que esto no sea factible, p debe ser suficientemente grande.

Criptosistema ElGamal

El cifrado ElGamal es un algoritmo de criptografía asimétrica basado en la idea de Diffie-Hellman.
Fue descrito por Taher Elgamal en 1984 [16]. La seguridad del algoritmo se basa en la suposición que la función utilizada es de un solo sentido y la dificultad de calcular un logaritmo discreto.
El cifrado consta de tres componentes: generación de claves, algoritmo de cifrado y algoritmo de descifrado.

A continuación se describe la versión elíptica del cifrado ElGamal.

Dados un cuerpo finito $\mathbb{F}_p$, una curva E, y un punto P, Alice y Bob deciden usar un primo p particular.
Alice escoge n_A como su clave secreta y $Q_A = n_A P$ como su clave pública.
El mensaje que desea enviar Bob es el punto $M(\mathbb{F}_p)$. Escoge el entero k para ser su clave secreta y calcula

$$C_1 = kPC_2 = M + kQ_A.$$

Entonces, envía los dos puntos (C_1, C_2) a Alice, que calcula

$$C_2 - n_A C_1 = (M + kQ_A) - n_A(kP) = M + k(n_A P) - n_A(kP) = M$$

para recuperar el mensaje.

Criptosistema RSA

Este algoritmo fue ideado en 1977 por Ron Rivest, Adi Shamir y Leonard Adleman (las letras RSA son las iniciales de sus apellidos). Es uno de los criptosistemas de clave pública más empleado en la actualidad, sencillo de comprender e implementar, aunque la longitud de sus claves es bastante considerable, ya que ha pasado desde sus 200 bits originales a los 2048 bits que se usan actualmente. RSA hace uso de dos de los algoritmos más importantes de la historia: el cálculo del Máximo Común Divisor de Euclides (Grecia, 450-377 A.C.) y el Pequeño Teorema de Fermat (Francia, 1601-1665). Su seguridad se basa en la diffcultad de factorizar numeros primos de gran tamaño, ya que aunque en principio, se puede deducir la clave secreta conociendo la clave pública, solo podría hacerse por medio de la factorización de números de gran longitud (centenares de cifras), lo que hace el problema intratable.

Versión con curvas elípticas [23]:
En este esquema se representan los puntos de una curva elíptica de la forma $y^2 = x^3 + ax + b$ sobre $\mathbb{Z}_n$ como $E_n(b)$. Para generar la clave pública el usuario B escogerá dos números primos grandes (p, q) tales que $p = q = 2$ (mód 3) y, como en la versión clásica, calculará y publicará (e, n), donde $n = p \cdot q$ y mantendrá en secreto las claves privadas $(p, q, \varphi(n), d)$.
Cada vez que A quiere enviar un mensaje m a B deberá seguir los siguientes pasos:

- *El ususario A divide su mensaje m en dos partes $m = (m_1, m_2)$ donde $m_1, m_2 \in \mathbb{Z}_n$.*

- *el usuario A determina el valor b de la curva que forma $m \in E_n(b)$. Específicamente, calcula $b = m_2^2 - m_1^3 (módn)$.*

◇ *Cifra el punto m calculando* $c = E(m) = e \cdot m$ *sobre* $E_n(b)$

◇ *Envía el texto cifrado* $c = (c_1, c_2)$ *a B.*

El usuario B para descifrar el mensaje c deberá hacer:

◇ *A partir del mensaje cifrado* $c = (c_1, c_2)$ *el usuario B puede determinar el valor de b puesto que este no cambia en el proceso de cifrado. Específicamente, calcula* $b = c_2^2 - c_1^3$ *(mód n) y construye la curva* $y^2 = x3 + ax + b$.
◇ *A partir de la clave privada calcula* $m = D(c) = d \cdot c$ *sobre* $E_n(0, b)$.

2.3.3. Curve25519

Curve25519 es una curva elíptica que establece claves entre dos partes tal que ningún atacante pasivo pueda obtener la clave ([3]) y ofrece alta seguridad y gran velocidad. Se usa principalmente en esquemas de establecimientos de claves Diffie-Hellman con curvas elípticas (ECDH).

Teorema 2.15. *Sea p un número primo con* $p \geq 5$. *Sea A un entero tal que* $A^2 - 4$ *no es un cuadrado módulo p.*
Se define E como la curva elíptica $y^2 = x^3 + Ax^2 + x$ *sobre el cuerpo* $\mathbb{F}_p$, *y* $X_0 : E(\mathbb{F}_{p^2}) \to \mathbb{F}_{p^2}$ *como sigue:* $X_0(\infty) = 0$; $X_0(x, y) = x$. *Sean n un entero y q un elemento de* $\mathbb{F}_p$. *Entonces existe un único* $s \in \mathbb{F}_p$ *tal que* $X_0(nQ) = s$ *para todo* $Q \in E(\mathbb{F}_{p^2}), X_0(Q) = q$.

En particular, se define p como el primo $2^{255} - 19$ *y* $\mathbb{F}_p$ *como el cuerpo* $Z/p = Z/(2^{255}-19)$. *Sea* $A = 486662$, *y E la curva elíptica* $y^2 - x^3 + Ax^2 + x$ *sobre* $\mathbb{F}_p$. *La función* $X_0 : E(\mathbb{F}_{p^2}) \to \mathbb{F}_{p^2}$ *es definida como* $X_0(\infty) = 0; X_0(x, y) = x$. *Dado* $n \in 2^{254} + 8\left\{0, 1, 2, 3, ..., 2^{251} - 1\right\}$ *y* $q \in \mathbb{F}_p$, *la función Curve25519 produce s en el teorema 2.15.*
El conjunto de claves públicas de Curve25519 es, por definición,

$$\left\{q : q \in \left\{0, 1, ..., 2^{256} - 1\right\}\right\}.$$

El conjunto de claves privadas de Curve25519 es

$$\left\{n : n \in 2^{254} + 8\left\{0, 1, 2, 3, ..., 2^{251} - 1\right\}\right\}.$$

∴ *Curve25519(n,q) es definida como s.*

Algunas de la características de la función Curve25519 son:

◇ *Claves cortas, tanto secretas como públicas.*

- ◇ *Velocidad muy alta en varias plataformas.*
- ◇ *Validación de claves.*
- ◇ *Código pequeño.*
- ◇ *Sin variabilidad temporal, lo que le hace inmune a ataques basados en el tiempo.*

2.3.4. Seguridad de los criptosistemas basados en curvas elípticas

Durante los primeros años de la criptografía basada en curvas elípticas, los diseñadores de estos criptosistemas tendían a elegir curvas con una estructura particular que les permitiera computar su número de puntos de manera rápida. Resulta, por lo tanto, desaconsejable seleccionar curvas elípticas basándose en una estructura suplementaria que facilite el conteo de puntos y/o simplifique la aritmética en ellas. Una solución natural sería pues la de evitar tales curvas, eligiendo para el protocolo una curva de forma totalmente aleatoria y verificando a posteriori que no sea susceptible a ataques de reducción, en cuyo caso se desecharía para repetir el proceso.
La otra posibilidad pasa por tomar un cuerpo finito lo suficientemente grande como para que el criptosistema sea seguro no sólo en el grupo de puntos de la curva elíptica, sino también en el correspondiente grupo de llegada de los ataques de reducción.
ECC proporciona un nivel de seguridad similar a RSA o sistemas basados en el DLP sobre $\mathbb{F}_p^*$, pero con unos operandos mucho más pequeños (160-256 bits en lugar de 1024-3072 bits aproximadamente).
La ECC tiene sus aplicaciones en el cifrado, firmas digitales, intercambio de claves, en el protocolo Transport Layer Security (TLS) y en el conjunto de algoritmos criptográficos denominado Suite B, propuesto por la Agencia Nacional de Seguridad de los Estados Unidos (NSA). También el bitcoin y otras criptodivisas se basan en esta criptografía de curva elíptica.

3

Protección de integridad y confidencialidad

En este capítulo estudiaremos diferentes criptosistemas simétricos que proporcionan al protocolo Signal las características de seguridad como integridad y confidencialidad, que se estudiarán en el siguiente capítulo.

3.1. HMAC-SHA256

Una de las herramientas fundamentales de la criptografía moderna son las funciones resumen criptográficas, a menudo llamadas funciones resumen de una vía. Su motivación es la de dar una representación corta de cualquier secuencia binaria de entrada.

Definición 3.1. *Una función resumen o función hash h es una función computacionalmente eficiente que envía secuencias binarias de una longitud arbitraria a una secuencia binaria de una longitud prefijada, que recibe el nombre de valor-resumen.*

Las funciones hash deben satisfacer las siguientes condiciones:

1. *Dependencia bits: El resumen de un mensaje o documento debe depender de todos los bits del mensaje, de modo que si se cambia un único bit del mensaje, su resumen debe cambiar la mitad de sus bits.*

2. *Resistencia a la primera preimagen: Dado un resumen h, es computacionalmente intratable encontrar m de la forma $r(m) = h$.*

3. *Resistencia a la segunda preimagen: Dado un mensaje concreto m, no se puede encontrar otro mensaje m' cuyo resumen coincida con m, es decir, que $r(m) = r(m')$.*

4. *Resistencia a colisiones: Debe ser difícil encontrar colisiones, i.e, encontrar dos mensajes cualesquiera m y m' cuyos resúmenes coincidan, es decir,* $r(m) = r(m')$.

Un uso típico de las funciones hash se encuentra en el proceso de generación las firmas digitales.

Definición 3.2. *Las funciones hash criptográficas son unidireccionales (one-way functions), es decir, son difíciles de invertir, pero fáciles de calcular. Una función no invertible f es una función con las siguientes propiedades:*

1. *Calcular la función* $f : \mathcal{K} \to \mathbb{Z}_n$ *lleva tiempo polinomial, fácil de calcular.*
2. *Calcular una preimagen x para una* $y = h(x)$ *arbitraria lleva más que tiempo polinomial, difícil de invertir.*

Para que una función resumen que dé como salida valores-resumen de n bits (típicamente, $n = 128$ *o* 160*) tenga propiedades deseables, la probabilidad de que una secuencia binaria escogida aleatoriamente tenga como imagen un valor-resumen determinado ha de ser* 2^{-n}*. Para su uso criptográfico, una función resumen h tiene que verificar que sea computacionalmente imposible encontrar dos entradas distintas que den el mismo valor-resumen (que colisionen, es decir, encontrar x e y tales que* $h(x) = h(y)$*) y que, dado un determinado valor-resumen y, sea computacionalmente imposible encontrar una contraimagen x tal que* $h(x) = y$.
Bajo ciertas hipótesis con respecto a la longitud de las cadenas de entrada, las funciones hash criptográficas se pueden utilizar como medios de autenticación bastante eficientes. El valor hash en estos casos recibe varios nombres: imprint, digital fingerprint, message digest.

3.1.1. Secure Hash Algorithm (SHA)

El Algoritmo de Hash Seguro (SHA) es una familia de funciones hash diseñadas por la Agencia de Seguridad Nacional (NSA) de EEUU publicadas por el Instituto Nacional de Estándares y Tecnología (NIST).

La función ***SHA-0*** *fue publicada por el NIST en 1993 como FIPS-180 y proporcionnaba un resumen de 160 bits. SHA-0 fue retirada poco después al presentar fallos significativos en su diseño y no ha vuelto ha ser utilizada.*

SHA-1 *fue publicada por el NIST en 1995. También proporciona un resumen de 160 bits con bloques de 512 bits del mensaje original.*
La función SHA-1 es la más utilizada porque se emplea en certificados digitales y otros protocolos criptográficos. Sin embargo, se han encontrado debilidades que

hacen desaconsejable su uso en aplicaciones a medio o largo plazo.

*La serie de funciones resumen **SHA-2** es la sucesora de SHA-1 y consta de cuatro algoritmos: SHA-224, SHA-256, SHA-384 y SHA-512, cada uno de ellos proporciona resúmenes del número de bits indicado en su nombre [10]. Al incrementar el tamaño de los resúmenes, disminuye la cantidad de posibles colisiones y la seguridad, por tanto, aumenta.*

Y la más reciente, SHA-3, seleccionada en una competición libre de funciones hash celebrada por el NIST en 2012.
SHA-256 y SHA-512 son funciones hash con palabras de tamaño 32 y 64 bits, respectivamente. Sus estructuras son virtualmente idénticas, diferenciándose únicamente por el número de iteraciones. SHA-256 es mucho más lenta que SHA-1, y SHA-224 y SHA-384 son simples versiones de truncado de las dos anteriores funciones. SHA-384 es relativamente inservible. Se recomienda trabajar con SHA-256 y SHA-512.

3.1.2. Códigos de autenticación de mensajes (MAC)

Dos partes comunicándose por un canal inseguro necesitan un método por el cual se detecte cualquier intento de modificar la información enviada de uno al otro. Este mecanismo, llamado MAC, está basado por una clave compartida entre las dos partes.
Los valores MAC se calculan mediante la aplicación de una función hash criptográfica con clave secreta k, que sólo conocen el remitente y el destinatario, pero no los atacantes.

***Definición 3.3.** Un MAC (Message Authentication code) es una familia de funciones h_k parametrizada por una clave secreta k, donde las funciones cumplen con las siguientes características:*

1. *Fácil de calcular: para una función h_k y dado un valor k y una entrada x, $h_k(x)$ es fácil de calcular.*
2. *Compresión: h_k mapea una entrada x de tamaño arbitrario a una salida $h_k(x)$ de longitud fija n.*
3. *Resistencia a calcularse: dados cero o más pares $MAC\,(x_i, h_k(x_i))$, es computacionalmente difícil calcular una nueva pareja $(x, h_k(x))$ tal que $h_k(x) = h_k(x_i)$ para alguna x diferente de x_i.*

El propósito de un MAC es, informalmente, el de aumentar, sin el uso de mecanismos adicionales, la confianza respecto a la fuente del mensaje y a su integridad. Los MAC tienen dos parámetros distintos, el mensaje de entrada y una

clave secreta incluida como parte del mensaje de entrada. El tamaño de la clave determinará la seguridad del algoritmo.

3.1.3. HMAC

Un código de autenticaciíon de mensajes con clave-hash (HMAC) es un MAC que usa una clave criptográfica secreta en combinación con una función hash criptográfica [2].

Definición 3.4. *Sea H una función hash criptográfica. Asumimos que H es cualquie función hash criptográfica. H toma entradas de cualquier longitud y produce l-bit de salida (l=128 para MD5 y l=160 para SHA-1).*
Sea m el mensaje a ser autenticado y K la clave secreta.
Denotamos por B al tamaño del bloque de entrada de la función hash (B=64).
Definimos dos cadenas distintas y fijas ipad y opad ('i' y 'o' son mnemotécnicos de inner y outer):

$$ipad = el\ byte\ 0 \times 36\ repetido\ B\ veces$$
$$opad = el\ byte\ 0 \times 5c\ repetido\ B\ veces$$

Entonces,

$$HMAC(K,m) = H((K' \oplus opad) \| H(K' \oplus ipad) \| m))$$

donde $\|$ denota concatenación y $\oplus$ denota disyunción exclusiva (XOR).

Si en la H anterior usamos SHA-256, obtenemos el algoritmo HMAC-SHA256, que opera en bloques de datos de 512-bit.
La seguridad de este algoritmo depende de las propiedades criptográficas de la función hash H. El ataque más común contra HMACs es la fuerza bruta para descubrir la clave secreta.

3.2. AES-256

En 1997 el NIST (United States National Institute for Standards and Technology) organiza una competición libre para escoger un sustituto al DES (Data Encryption Standard) [22]. Los requisitos para los algoritmos presentados eran:

- ◇ *Longitud de bloque = 128 bits*
- ◇ *Longitud de clave = 128, 192 y 256 bits*
- ◇ *Número de iteraciones flexibles: 10, 12 y 14*
- ◇ *Operaciones a nivel de byte, con palabras de 4 bytes*

Después de varios años de análisis y conferencias internacionales, NIST anunció en 2001 la elección del Rijndael, desarrollado por los criptógrafos belgas Joan Daemon y Vincent Rijmen. Rijndael es una familia de cifrados con distintas longitudes de clave y tamaños de bloques. El cifrado AES (Advanced Encryption Standard) es casi idéntico al cifrado en bloque de Rijndael. El tamaño de bloque y clave en Rijndael varía entre 128, 192 y 256 bits, mientras en el AES el tamaño del bloque es de 128 bits.
Los criterios considerados en la selección del AES estaban basados en:

- ◇ *Seguridad (esfuerzo criptoanalítico)*
- ◇ *Eficacia computacional*
- ◇ *Adecuación hardware y software*
- ◇ *Simplicidad de diseño*
- ◇ *Flexibilidad*
- ◇ *No es de tipo Feistel(*)*
- ◇ *Fácilmente paralelizable*
- ◇ *Posible implementación en tarjeta inteligente*
- ◇ *Requisitos de licencia*

Nota:. (*)
También llamado Red de Feistel, es un método de cifrado en bloque con una estructura particular, utilizado en el algoritmo estándar de cifrado anterior al AES (DES).

AES es un algoritmo de clave simétrica, i.e la misma clave es usada para cifrar y descifrar los datos. Este algoritmo es similar a DES en que cifra y descifra repitiendo una operación básica varias veces.
La entrada y salida en el algoritmo AES cada una consiste de secuencias de 128 bits. Para la entrada o salida de una clave de cifrado denotado por a, los bytes en el resultante vector serán referenciados como a_n, donde n tomará valores de los siguientes rangos:

$$\begin{aligned} Longitud\ de\ clave &= 128 bits, 0 \leq n < 16 \\ Longitud\ de\ clave &= 192 bits, 0 \leq n < 24 \\ Longitud\ de\ clave &= 256 bits, 0 \leq n < 32 \end{aligned}$$

$$a_n = input_{8n}, input_{8n+1}, ..., input_{8n+7}$$

Estos bytes son interpretados como elementos $GF(2^8)$ usando una representación polinómica:

$$b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x + b_0 = \sum_{i=0}^{7} b_i x^i$$

Por ejemplo,

$$\{63\} = \{01100011\} = x^6 + x^5 + x + 1.$$

El Estado

Cada operación del algoritmo AES tiene como entrada la salida de la operación anterior. Este resultado intermedio se denomina ***Estado*** *(ó State en inglés).*
El Estado consiste de 4 filas de bytes, cada una conteniendo Nb bytes, donde Nb es la longitud de bloque dividido entre 32.
Cada byte corresponde a dos índices, r siendo el número de fila en el rango $0 \leq r < 4$ *y c el número de columna en el rango* $0 \leq c < Nb = 4$.
Esto permite a cada byte en el Estado a ser referido como $S_{r,c}$.
En el comienzo del Cifrado y del Cifrado inverso, el vector de entrada, in, es copiado al vector Estado de acuerdo al esquema siguiente:

$$S_{r,c} = in_{r+4c},\ para\ 0 \leq r < 4, 0 \leq c < Nb$$

Y en el final:

$$out_{r+4c} = s_{r,c},\ para\ 0 \leq r < 4, 0 \leq c < Nb$$

in_0	in_4	in_8	in_{12}
in_1	in_5	in_9	in_{13}
in_2	in_6	in_{10}	in_{14}
in_3	in_7	in_{11}	in_{15}

Tabla 3.1: Bytes de entrada

$S_{0,0}$	$S_{0,1}$	$S_{0,2}$	$S_{0,3}$
$S_{1,0}$	$S_{1,1}$	$S_{1,2}$	$S_{1,3}$
$S_{2,0}$	$S_{2,1}$	$S_{2,2}$	$S_{2,3}$
$S_{3,0}$	$S_{3,1}$	$S_{3,2}$	$S_{3,3}$

Tabla 3.2: Matriz del Estado

out_0	out_4	out_8	out_{12}
out_1	out_5	out_9	out_{13}
out_2	out_6	out_{10}	out_{14}
out_3	out_7	out_{11}	out_{15}

Tabla 3.3: Bytes de salida

3.2.1. Operaciones en AES (en $GF(2^8)$)

Suma

La operación suma coincide con la operación XOR de los bits, denotada por $\oplus$.

$$1 \oplus 1 = 0, 1 \oplus 0 = 1, 0 \oplus 0 = 0$$

Ejemplo 3.5.

$$57 \oplus 83 = D4$$

En notación binaria:

$$\{01010111\} \oplus \{10000011\} = \{11010100\}$$

Y en notación polinómica:

$$(x^6 + x^4 + x^2 + x + 1) + (x^7 + x + 1) = x^7 + x^6 + x^4 + x2$$

La resta de polinomios es idéntica a la suma.

Multiplicación

Esta operación corresponde con la multiplicación de polinomios módulo un polinomio irreducible de grado 8. Para el algoritmo AES, este polinomio irreducible es

$$m(x) = x^8 + x^4 + x^3 + x + 1$$

.

Ejemplo 3.6.

$$57 \cdot 83 = c1$$

$$(x^6 + x^4 + x^2 + x + 1) \otimes (x^7 + x + 1) =$$

$$x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1 \equiv (x^7 + x^6 + 1) mód (x^8 + x^4 + x^3 + x + 1)$$

Inverso de un elemento

El inverso en $GF(2^8)$ es la operación básica de la transformación SubByte.

Definición 3.7. *Dado una cuerpo finito $GF(2^m)$ y el polinomio irreducible correspondiente $P(x)$, el inverso A^{-1} de un elemento $(\neq 0)$ $A \in GF(2^m)$ es:*

$$A^{-1}(x) \cdot A(x) = 1 mód P(x)$$

La tabla 3.1 contiene todos los inversos en $GF(2^8)$ módulo $P(x) = x^8 + x^4 + x^3 + x + 1$ en notación hexadecimal.

	Y															
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	00	01	8D	F6	CB	52	7B	D1	E8	4F	29	C0	B0	E1	E5	C7
1	74	B4	AA	4B	99	2B	60	5F	58	3F	FD	CC	FF	40	EE	B2
2	3A	6E	5A	F1	55	4D	A8	C9	C1	0A	98	15	30	44	A2	C2
3	2C	45	92	6C	F3	39	66	42	F2	35	20	6F	77	BB	59	19
4	1D	FE	37	67	2D	31	F5	69	A7	64	AB	13	54	25	E9	09
5	ED	5C	05	CA	4C	24	87	BF	18	3E	22	F0	51	EC	61	17
6	16	5E	AF	D3	49	A6	36	43	F4	47	91	DF	33	93	21	3B
7	79	B7	97	85	10	B5	BA	3C	B6	70	D0	06	A1	FA	81	82
X 8	83	7E	7F	80	96	73	BE	56	9B	9E	95	D9	F7	02	B9	A4
9	DE	6A	32	6D	D8	8A	84	72	2A	14	9F	88	F9	DC	89	9A
A	FB	7C	2E	C3	8F	B8	65	48	26	C8	12	4A	CE	E7	D2	62
B	0C	E0	1F	EF	11	75	78	71	A5	8E	76	3D	BD	BC	86	57
C	0B	28	2F	A3	DA	D4	E4	0F	A9	27	53	04	1B	FC	AC	E6
D	7A	07	AE	63	C5	DB	E2	EA	94	8B	C4	D5	9D	F8	90	6B
E	B1	0D	D6	EB	C6	0E	CF	AD	08	4E	D7	E3	5D	50	1E	B3
F	5B	23	38	34	68	46	03	8C	DD	9C	7D	A0	CD	1A	41	1C

Figura 3.1: Inversos en $GF(2^8)$.

Ejemplo 3.8. Usando la tabla 3.1, el inverso de

$$x^7 + x^6 + x = (11000010)_2 = (c2)_{16} = (xy)$$

está dado por el elemento en la fila c y columna 2:

$$(2F)_{16} = (00101111)_2 = x^5 + x^3 + x^2 + x + 1$$

Lo verificamos multiplicando

$$(x^7 + x^6 + x) \cdot (x^5 + x^3 + x^2 + x + 1) = 1 mód P(x)$$

3.2.2. Algoritmo AES

Como se vio anteriormente, la longitud del bloque de entrada, bloque de salida y el Estado es 128 bits. La longitud de la clave de cifrado, k, es 128, 192 0 256 bits y es representada por $N_k = 4, 6o8$*, que refleja el número columnas en la clave de cifrado. El número de iteraciones utilizadas durante el algoritmo depende del tamaño de clave, que se puede ver en la siguiente tabla:*

	Longitud de clave (N_k)	Tamaño de bloque (N_b)	Nº de iteraciones (N_r)
AES-128	4	4	10
AES-192	6	4	12
AES-256	8	4	14

Tabla 3.4: Nº de iteraciones utilizadas en el algoritmo.

Para cifrado y cifrado inverso, el algoritmo AES usa iteraciones compuestas por 4 distintas transformaciones:

- ◇ ***SubBytes()****. Sustitución de cada byte por su recíproco usando una tabla de sustitución (S-box).*
- ◇ ***ShiftRows()****. Desplazamiento de bytes.*
- ◇ ***MixColumns()****. Multiplicación de cada columna por una matriz.*
- ◇ ***AddRoundKey()****. Suma de la subclave y la información del estado intermedio actual.*

- *Cifrado*

El esquema general del algoritmo consta de: expansión de claves, ronda inicial, ronda estándar, ronda final y tiene la estructura siguiente.

P1: Expansión de claves
P2: Iteración 0: AddRoundKey(S)
P3: Transformación 1: SubBytes(S)
Transformación 2: ShiftRows(S)
Transformación 3: MixColumns(S)
Transformación 4: AddRoundKey(S)
P4: SubBytes(S)
ShiftRows(S) AddRoundKey(S)
Ahora, explicaremos cada una de estas transformaciones:

1. *Transformación SubBytes()*
   Es una sustitución no lineal de un byte que opera independientemente sobre cada byte del Estado usando una tabla de sustitución (S-box). Esta S-box,

que es invertible, es generada por dos transformaciones:

a) *Con el inverso multiplicativo en* $GF(2^8)$.
Nota:El elemento 0 *no tiene inverso, sin embargo, en AES su inverso es él mismo.*
b) *Se aplica la transformación lineal en forma de matriz:*

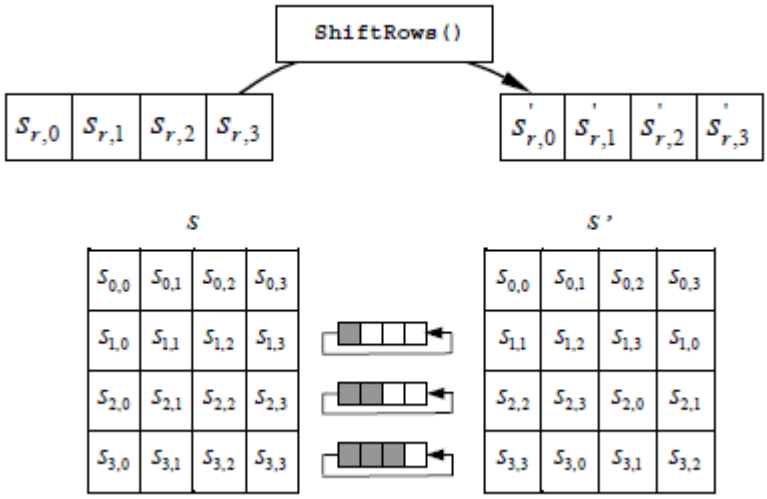
$$\begin{bmatrix} b'_0 \\ b'_1 \\ b'_2 \\ b'_3 \\ b'_4 \\ b'_5 \\ b'_6 \\ b'_7 \end{bmatrix} = \begin{bmatrix} 1&0&0&0&1&1&1&1 \\ 1&1&0&0&0&1&1&1 \\ 1&1&1&0&0&0&1&1 \\ 1&1&1&1&0&0&0&1 \\ 1&1&1&1&1&0&0&0 \\ 0&1&1&1&1&1&0&0 \\ 0&0&1&1&1&1&1&0 \\ 0&0&0&1&1&1&1&1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

		y														
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
	1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
	2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
	3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
	4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
	5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
	6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
	7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
	8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
	9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
	a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
	b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
	c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
	d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
	e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
	f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Figura 3.2: Caja S

La caja S usada en esta transformación es presentada en forma hexadecimal. Ejemplo: Si $S_{1,1} = 53$, *el resultado será la intersección de la fila 5 con columna 3, i.e* $S'_{1,1} = ed$.

2. *Transformación ShiftRows()*
   El paso ShiftRows opera sobre las filas de la matriz Estado. Consiste en desplazar la segunda fila de la matriz estado por tres bytes a la derecha, la tercera fila por dos bytes a la derecha y la cuarta fila por un byte a la derecha. La primera fila no se cambia en esta transformación.



3. *Transformación MixColumns()*
   Opera en la matriz del estado columna por columna, tratando cada columna como un polinomio de cuatro términos.
   Las columnas son consideradas como polinomios sobre $GF(2^8)$ *y multiplicadas módulo* $x^4 + 1$ *con un polinomio fijo* $a(x)$*, dado por*

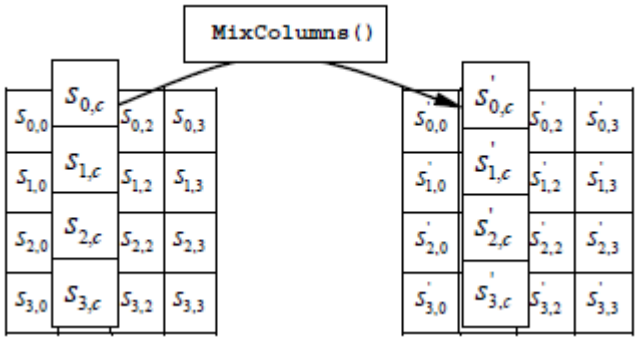
$$a(x) = 03x^3 + 01x^2 + 01x + 02$$

Esto se puede escribir como una multiplicación de matrices. Sea S'(x)=a(x)⊗S(x) :

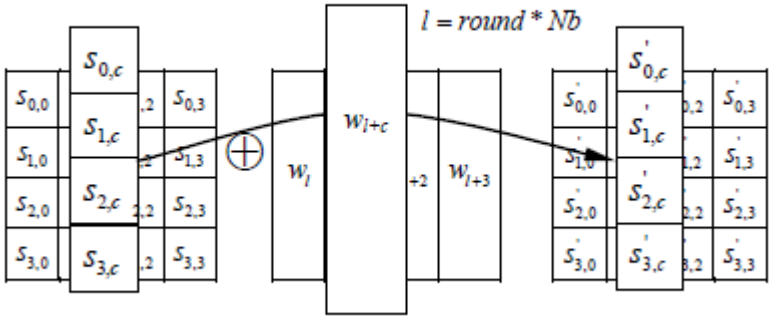
$$\begin{bmatrix} S'_{0,c} \\ S'_{1,c} \\ S'_{2,c} \\ S'_{3,c} \end{bmatrix} = \begin{bmatrix} 02\ 03\ 01\ 01 \\ 01\ 02\ 03\ 01 \\ 01\ 01\ 02\ 03 \\ 03\ 01\ 01\ 02 \end{bmatrix} \begin{bmatrix} S_{0,c} \\ S_{1,c} \\ S_{2,c} \\ S_{3,c} \end{bmatrix}$$

Como resultado de esta multiplicación, los cuatro bytes en una columna son remplazados por:

$S'_{0,c} = (0, 2 \cdot S_{0,c}) \oplus (0, 3 \cdot S_{1,c}) \oplus S_{2,c} \oplus S_{3,c}$
$S'_{1,c} = S_{0,c} \oplus (02 \cdot S_{1,c}) \oplus (03 \cdot S_{2,c}) \oplus S_{3,c}$
$S'_{2,c} = S_{0,c} \oplus S_{1,c} \oplus (02 \cdot S_{2,c}) \oplus (03 \cdot S_{3,c})$
$S'_{3,c} = (03 \cdot S_{0,c}) \oplus S_{1,c} \oplus S_{2,c} \oplus (02 \cdot S_{3,c})$



Transformación AddRoundKey()
Esta transformación consiste en una OR-EXCLUSIVO entre el Estado Intermedio 4 y una subclave que se genera a partir de la clave original de cifrado.



- Expansión de claves

Con la clave de cirado K, la expanxión de claves genera un total de $Nb \cdot (Nr+1)$ palabras de 4 bytes pues en cada una de las $Nr+1$ iteraciones hace falta una subclave de $Nb \cdot 4$ bytes.
el resultado consiste de un vector lineal de palabras de cuatro bytes, donotado por $[w_i]$, con i en el rango $0 \leq< Nb \cdot (Nr+1)$.
Existen diferentes procedimientos (muy similares) para los tres tipos de claves en AES de 128, 192 y 256 bits.

- Descifrado o Cifrado inverso

Es similar al de cifrado, pero algunas operaciones son las transformaciones inversas de las operaciones realizadas en el cifrado, y las subclaves se usan en orden inverso. La transformación Addroundkey(S) es su propio inverso, ya que solo usa la operación XOR.

InverseSubByte

Como la tabla de sustitución es biyectiva, es posible construir un s-box inverso:

		y														
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
x	0	52	09	6A	D5	30	36	A5	38	BF	40	A3	9E	81	F3	D7	FB
	1	7C	E3	39	82	9B	2F	FF	87	34	8E	43	44	C4	DE	E9	CB
	2	54	7B	94	32	A6	C2	23	3D	EE	4C	95	0B	42	FA	C3	4E
	3	08	2E	A1	66	28	D9	24	B2	76	5B	A2	49	6D	8B	D1	25
	4	72	F8	F6	64	86	68	98	16	D4	A4	5C	CC	5D	65	B6	92
	5	6C	70	48	50	FD	ED	B9	DA	5E	15	46	57	A7	8D	9D	84
	6	90	D8	AB	00	8C	BC	D3	0A	F7	E4	58	05	B8	B3	45	06
	7	D0	2C	1E	8F	CA	3F	0F	02	C1	AF	BD	03	01	13	8A	6B
	8	3A	91	11	41	4F	67	DC	EA	97	F2	CF	CE	F0	B4	E6	73
	9	96	AC	74	22	E7	AD	35	85	E2	F9	37	E8	1C	75	DF	6E
	A	47	F1	1A	71	1D	29	C5	89	6F	B7	62	0E	AA	18	BE	1B
	B	FC	56	3E	4B	C6	D2	79	20	9A	DB	C0	FE	78	CD	5A	F4
	C	1F	DD	A8	33	88	07	C7	31	B1	12	10	59	27	80	EC	5F
	D	60	51	7F	A9	19	B5	4A	0D	2D	E5	7A	9F	93	C9	9C	EF
	E	A0	E0	3B	4D	AE	2A	F5	B0	C8	EB	BB	3C	83	53	99	61
	F	17	2B	04	7E	BA	77	D6	26	E1	69	14	63	55	21	0C	7D

InverseShiftRow

Se realiza el mismo proceso que en el cifrado pero desplazando a la derecha los bytes de las filas que conforman la matriz del estado actual. En este caso también la primera fila no se cambia.

InverseMixColumn

Para esta transformación buscamos el inverso del polinomio a(x) (descrito en la transformación mixcolumn):

$$a^{-1}(x) = 0bx^3 + 0dx^2 + 09 + 0e$$

Sea $S'(x) = a^{-1} \otimes S(x)$, en forma matricial este producto queda como:

$$\begin{bmatrix} \mathrm{S'}_{0,c} \\ \mathrm{S'}_{1,c} \\ \mathrm{S'}_{2,c} \\ \mathrm{S'}_{3,c} \end{bmatrix} = \begin{bmatrix} 0e\ 0b\ 0d\ 09 \\ 09\ 0e\ 0b\ 0d \\ 0d\ 09\ 0e\ 0b \\ 0b\ 0d\ 09\ 0e \end{bmatrix} \begin{bmatrix} \mathrm{S}_{0,c} \\ \mathrm{S}_{1,c} \\ \mathrm{S}_{2,c} \\ \mathrm{S}_{3,c} \end{bmatrix}$$

Como resultado de esta multiplicación, los cuatro bytes en una columna son remplazados por:

$$\begin{aligned} S'_{0,c} &= (0e \cdot S_{0,c}) \oplus (0b \cdot S_{1,c}) \oplus (0d \cdot S_{2,c}) \oplus (09 \cdot S_{3,c}) \\ S'_{0,c} &= (09 \cdot S_{0,c}) \oplus (0e \cdot S_{1,c}) \oplus (0b \cdot S_{2,c}) \oplus (0d \cdot S_{3,c}) \\ S'_{0,c} &= (0d \cdot S_{0,c}) \oplus (09 \cdot S_{1,c}) \oplus (0e \cdot S_{2,c}) \oplus (0b \cdot S_{3,c}) \\ S'_{0,c} &= (0b \cdot S_{0,c}) \oplus (0d \cdot S_{1,c}) \oplus (09 \cdot S_{2,c}) \oplus (0e \cdot S_{3,c}) \end{aligned}$$

3.2.3. Seguridad en el AES

El AES tiene 10 iteraciones para claves de 128 bits, 12 iteraciones para claves de 192 bits y 14 para claves de 256 bits.
Hasta 2005, los mejores ataques conocidos son sobre versiones reducidas a 7 iteraciones para claves de 128 bits, 8 iteraciones para claves de 192 bits y 9 iteraciones para claves de 256 bits.
El riesgo es que se puede encontrar alguna manera de mejorar los ataques y de ser así, el cifrado podría ser roto.

Nota:. Un algoritmo se considera roto si existe algún ataque por fuerza bruta.

4

Tecnologías que implementan Signal

Signal es un protocolo desarrollado por Open Whisper Systems en 2013 como parte de su aplicación de mensajería SMS *TextSecure*, que actualmente se llama Signal.
El objetivo principal del protocolo es poder enviar un mensaje de forma segura, aunque el destinatario no esté conectado en ese momento. Es un protocolo de seguridad que proporciona cifrado extremo a extremo (end-to-end encryption), *deniability* (repudio), *forward secrecy* (confidencialidad directa) y *future secrecy* (confidencialidad futura). Estas propiedades se estudiarán a lo largo de este capítulo. Otras de las propiedades importantes es *usability* (facilidad de uso).
Signal ha sido adaptado por aplicaciones de mensajería instantánea como WhatsApp, Facebook Messenger, y Google Allo entre muchas otras; las dos primeras tienen al menos un billón de usuarios activos.
El protocolo consiste de las siguientes fases:

- ◇ Fase de registro
- ◇ Fase de verificación de claves
- ◇ Fase de establecimiento de sesión (envío/recepción de un primer mensaje)
- ◇ Fase de mantenimiento de sesión (envío/recepción de mensajes siguientes)
- ◇ Fase de envío de respuesta

En la siguiente sección estudiaremos cómo se aplica el protocolo a WhatsApp y otras aplicaciones.

4.1. WhatsApp

WhatsApp Messenger permite a las personas intercambiar mensajes (incluyendo conversaciones, imágenes, vídeos, mensajes de voz y documentos) y realizar llamadas de WhatsApp alrededor del mundo.
Los mensajes y llamadas de Whatsapp entre un remitente y destinatario que

usan el software de cliente WhatsApp, publicado después del 31 de marzo de 2016, son de cifrado extremo a extremo [29]. Además, la nueva característica, *WhatsApp Status* (las historias que duran 24 horas), añadida en febrero de 2017 usa el protocolo para asegurar su contenido.

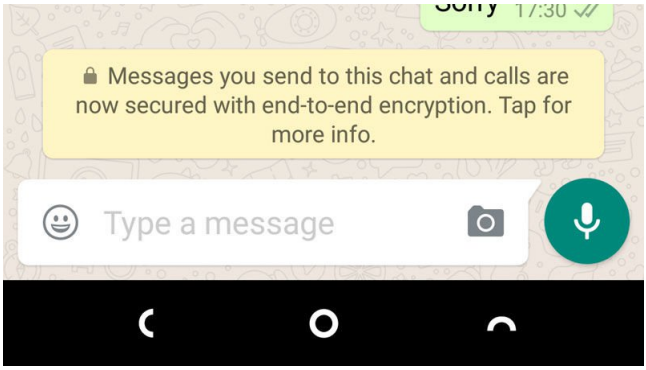


Figura 4.1: WhatsApp's end-to-end encryption

Este protocolo de cifrado extremo a extremo está diseñado para prevenir a terceras partes y WhatsApp de tener acceso a los mensajes y llamadas.
La integración de Signal con WhatsApp ofrece a sus usuarios las siguientes propiedades:

- Confidencialidad, esto es que las comunicaciones están cifradas;
- Integridad, que significa que cualquier alteración en un mensaje será detectada y no se producirá la transacción. Esto último implica que existe un código de autenticación de mensaje (MAC);
- Autenticación, aunque esto hay que activarlo, ya que está desactivado por defecto. Mediante la autenticación cada participante es capaz de verificar la identidad de la otra persona;
- Forward secrecy, es decir, si alguna vez en el futuro se compromete la clave privada, no será posible descifrar mensajes antiguos. Del mismo modo se dispone de backward secrecy, siendo lo contrario del anterior, si una clave privada antigua se ve comprometida, no será posible descifrar mensajes futuros. Estas dos propiedades se consiguen con claves efímeras o *ephemeral keys*. Este tipo de claves están en constante cambio y renegociándose continuamente, de modo que alguien que consiga una clave no podrá usarla.

◇ Message unlinkability (Asincronía), los mensajes son asíncronos e independientes, pueden llegar en distinto orden, pueden perderse, y aún así el sistema seguirá siendo consistente;
◇ Por último, deniability, i.e, dada una copia de una conversación y todas las claves criptográficas, no hay evidencia de que un mensaje dado sea autorizado por un usuario particular.
Cuando nos registramos por primera vez en WhatsApp se intercambian una serie de claves con los servidores de manera que quedamos autenticados con él. El sistema hace uso de tres tipos de claves públicas:
Una clave larga Curve25519 generada en el momento de instalación, para identificar al dispositivo, *Identity Key Pair*; otra clave media Curve25519 generada durante la instalación, firmada digitalmente por la anterior y que cambia con el tiempo, *Signed Pre Key*; y otra que se usa solo una vez en cada utilización del servicio, *One-Time Pre Keys*.
Además de las claves públicas, se usan las claves de sesión:
Una clave de 32-byte utilizada para crear las *Chain Keys*,*Root Key*; una clave de 32-byte usada para crear las *Message Keys*, *Chain Key*; y una clave de 80 bytes, *Message Key*, utilizada para cifrar todo el tráfico de las comunicaciones:

- 32 bytes para una clave AES-256
- 32 bytes para una clave HMAC-SHA256
- y 16 para un IV, un vector de inicialización.

Para comunicar con otro usuario de WhatsApp, el cliente WhatsApp, en este caso el emisor, necesita establecer una sesión cifrada. Para ello, el emisor almacena las claves Identity Key del receptor como $I_{recipient}$; Signed Pre Key como $S_{recipient}$; y One-Time Pre Key como $O_{recipient}$. El emisor genera una clave efímera Curve25519 $E_{initiator}$ y carga su propia clave Identity Key como $I_{initiator}$.
Luego, calcula un *master_secret*=

$$ECDH(I_{initiator}, S_{recipient})||ECDH(E_{initiator}, I_{recipient})||$$

$$ECDH(E_{initiator}, S_{recipient})||ECDH(E_{initiator}, O_{recipient})$$

Si no hay ninguna clave One-time Pre Key, el ECDH final se omite.
El emisor utiliza la función HKDF para crear una Root Key y Chain Keys de *master_secret*.

Una vez establecida una sesión cifrada, el emisor puede enviar mensajes al receptor aunque este se encuentre desconectado. Cuando el receptor recibe el mensaje, calcula su correspondiente *master_secret* usando sus propias claves privadas y públicas. El receptor elimina One-Time Pre Key usada por el emisor.

Intercambio de mensajes

Los usuarios intercambian mensajes que están protegidos con la clave Message Key usando AES256 para el cifrado y SHA256 para la autenticación. Un nuevo acuerdo ECDH se realiza con cada mensaje, que se envia y se recibe, para crear una nueva clave Chain Key. Esto proporciona "forward secrecy".

Archivos adjuntos

Los archivos adjuntos también están cifrados extremo a extremo. El emisor genera una clave efímera AES256 de 32 byte y una clave efímera HMAC-SHA256 de 32 byte. El emisor cifra el archivo adjunto con la clave AES256 en modo CBC (Cipher-Block chaining) con un IV aleatorio, y luego, anexa una MAC del texto cifrado usando HMAC-SHA256. El receptor descifra el mensaje, verifica el hash SHA256, el MAC y descifra el texto sin formato. Todo tipo de mensaje WhatsApp (incluyendo conversaciones de grupos, imágenes, vídeos, mensajes de voz y documentos) y llamadas WhatsApp están protegidas por el cifrado extremo a extremo.Los servidores de WhatsApp no tienen acceso a las claves privadas de los usuarios.

Nota:. El CBC es un modo de operación de una unidad de cifrado por bloques. En este modo a cada bloque de texto plano se le aplica la operación XOR con el bloque cifrado anterior antes de ser cifrado. De esta forma, cada bloque de texto cifrado depende de todo el texto en claro procesado hasta este punto. Para hacer cada mensaje único se utiliza asimismo un vector de inicialización IV.

4.2. Otras aplicaciones

Facebook Messenger

Facebook Messenger utiliza el mismo protocolo de cifrado que WhatsApp en sus 'conversaciones secretas' dentro de la aplicación.

Figura 4.2: Sistemas que implementan el protocolo Signal

La principal diferencia de WhatsApp frente a Facebook Messenger es que en la primera la función de cifrado extremo a extremo está activada por defecto mientras que los usuarios de Facebook Messenger tienen que decidir en qué conversaciones quieren activar el cifrado, y hacerlo de forma manual.
Con el cifrado activo algunas de las funciones de Facebook Messenger no están operativas. Por ejemplo, los usuarios no tienen acceso a los 'bots' capaces de sugerir información contextual a la conversación. Tampoco se puede compartir vídeos y GIFs animados.
Además, la conversación secreta de Messenger ofrece la opción de autodestrucción, es decir, se pueden configurar los mensajes para que desaparezcan automáticamente a los pocos segundos de haber sido leídos por la otra persona.

Nota:. Un bot es un programa informático, imitando el comportamiento de un humano.

Google Allo y Duo

Allo, servicio de mensajería de Google, está diseñado para permitir un nuevo estilo de comunicación, en el que un bot escucha nuestras conversaciones y da sugerencias y servicios. Allo ofrece un modo *incógnito* que usa el sistema de cifrado extremo a extremo Signal.
En el modo incógnito, los mensajes tienen la capacidad de autodestruirse al pasar un tiempo específico y las conversaciones se encuentran cifradas por defecto.
Duo, aplicación de video llamadas de Google, utiliza también el cifrado extremo a extremo.

Signal

La aplicación Signal es la plataforma más segura, gestionada por Open Whisper System, fundada en 2013 por Moxie Marlinspike. Fue originada a partir de dos aplicaciones móviles distintas, TextSecure (mensajería instantánea cifrada) y RedPhone (llamadas cifradas).
Debido a su fuerte protocolo de cifrado y disponibilidad de su código fuente bajo licencia de código abierto, Signal se ha convertido en una herramienta importante para usuarios que se preocupan por su seguridad.
Para registrarse al servicio basta con tener un número de teléfono. No requiere ni nombre de usuario ni una cuenta de correo electrónico asociada.

Otra de las aplicaciones que utilizan este protocolo son Secure Chat, de G Data Software y Skype. Ésta última cuenta con un chat de Conversaciones Privadas y se empezó a usar en enero de 2018. Signal también tiene influencia sobre tecnologías como Viber y Wire.

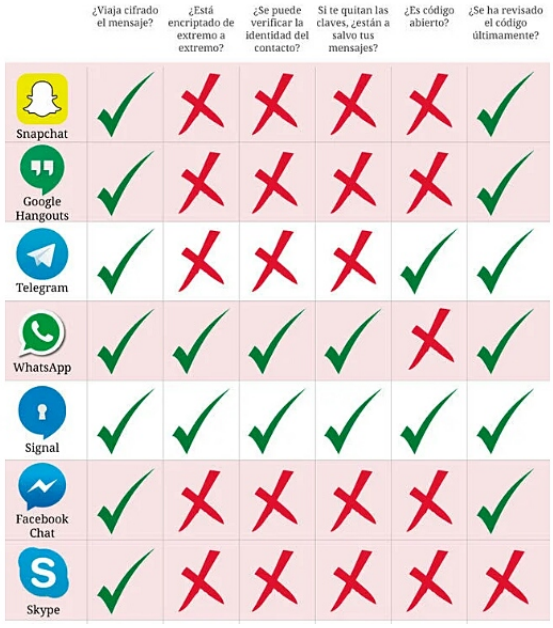


Figura 4.3: Seguridad de algunas aplicaciones de mensajería instantánea. Fuente: Electronic Frontier Foundation(2016)

En la imagen 4.3 se observa que la aplicación Signal cumple todos los requisitos para ser aplicación de mensajería segura.

5

Seguridad del protocolo Signal

Más de la mitad de la población mundial utiliza aplicaciones de mensajería instantánea. Si sus opciones de seguridad no son sólidas, nuestra privacidad y, por ende, nuestra seguridad se pueden ver vulneradas. La mayoría de las personas se preguntarán para qué preocuparse de tanta seguridad si no están haciendo algo ilegal, sin embargo, es una cuestión de proteger nuestra identidad, los contactos o nuestras conversaciones de terceros.
El protocolo Signal se ha desarrollado con el fin de proporcionarnos la seguridad que necesitamos. En este capítulo se estudiará la seguridad de este protocolo y las mejoras que puede tener.

La mensajería segura se divide en tres áreas de problemas: *trust establishment problem* (problema de establecimiento de confianza), *conversation security problem* (problema de seguridad de conversación) y *transport privacy problem* (problema de privacidad de transporte).
La primera asegura la distribución de claves criptográficas de larga duración. La seguridad de conversación asegura la protección de mensajes intercambiados. Esto engloba cómo los mensajes son encriptados, qué información es adjunta a ellos y qué protocolos criptográficos se realizan. Las características de privacidad y seguridad, ya mencionadas en el capítulo 4, son confidencialidad, integridad, autenticación, confidencialidad directa, confidencialidad futura y repudio. Y el último, privacidad de transporte, define cómo los mensajes son intercambiados con el objetivo de ocultar metadatos de mensajes como el emisor, receptor y la conversación a la cual el mensaje pertenece. Algunas de las propiedades de usabilidad de este área son:

- El sistema proporciona un mecanismo para hallar información de contactos.
- El usuario no necesita realizar ninguna tarea significativa antes de empezar la conversación.

- No hay retrasos en los mensajes.

Para que el protocolo sea seguro se deben cumplir las características de privacidad y seguridad descritas anteriormente. Sin embargo, hay una que no se cumple completamente, el repudio.
Cuando una parte envía un mensaje, lo hace por un servidor. El mensaje es encriptado, pero para garantizar la entrega correcta, las identidades del emisor y del receptor deben ser transmitidas al servidor. Esta solicitud de envío es autenticada con el número de móvil y contraseña del emisor.
El servidor no es capaz de leer el contenido de ningún mensaje y solo realiza la entrega. Por tanto, nadie puede demostrar qué contenido se envió. Hasta ahora se cumple la propiedad. Sin embargo, no se puede negar de tener una conversación con alguna parte o estar involucrados en alguna conversación.
Para mejorar esta propiedad de repudio del protocolo Signal, se modifica el intercambio inicial de clave DH, sustituyendo el intercambio de claves DH por tres intercambios DH (Triple Diffie-Hellman).

Ataques genéricos

Vamos a distinguir dos tipos de ataques:

- Ataques sobre la configuración de la primera sesión.
  Este ataque solo puede ser detectado por verificación manual, es decir, escaneando los códigos QR. Por ejemplo, Bob quiere inicializar una sesión segura con Alice, y Bob recibe la clave de identidad del atacante en vez de la clave de identidad de Alice. La aplicación almacenará como identidad de Alice al atacante.
- Ataques en sesiones establecidas.
  Bob ha establecido previamente una sesión segura con Alice y ha almacenado la clave de identidad de Alice correctamente. El atacante podría forzar a las dos partes a renegociar una nueva sesión de comunicación.

Debido a que Signal es un protocolo reciente, no existen análisis de seguridad . El siguiente análisis se realizó sobre la versión antigua de Signal, TextSecure v3 [8, 13, 18].

5.1. Análisis de Frosch

Frosch et al. [13] identificaron un ataque UKS (unknown key-share) contra TextSecure, porque el material criptográfico no estaba ligado a las identidades.
En un ataque UKS, un adversario, Eve, obliga a las partes honestas Alice y Bob en establecer una clave secreta donde al menos uno de los dos desconoce que la clave secreta está compartida con la otra parte.

Por ejemplo, Eve puede obligar a Bob en creer que él comparte la clave con Eve, mientras que él realmente comparte la clave con Alice. Es decir, Bob desconoce que comparte clave con Alice.

El ataque UKS sobre TextSecure v3

El atacante (P_b) realiza los siguientes pasos:
(1-2) P_b solicita $g^{x_{e,z_0}}, \cdots, g^{x_{e,z_i}}$ al servidor de TextSecure (TS) usando el teléfono $_e$.

(3-4) P_b encomenda $g^{x_{e,z_0}}, \cdots, g^{x_{e,z_i}}$ a TS como sus propias claves y g^e como su propia clave pública a largo plazo.

(5) P_b deja a P_a verificar la huella de su nueva clave pública g^e. (Este paso utiliza código QR).

(6-7)Cuando P_a quiere enviar un mensaje a P_b,P_a solicita una clave para P_b usando el teléfono $_b$. TS devuelve $g^{x_{b,z_j}} = g^{x_{e,z_j}}$ y la clave a largo plazo $g^b = g^e$.

(8-9) P_a calcula su nueva clave efímera sec_{int} usando $g^{x_{b,z_j}}$ y g^b de donde $(k_{E_{nc},AB}, k_{MAC,AB})$ van a ser derivados. Para calcular esa clave, P_a usa la clave de identidad de P_e creyendo usar la de P_b. Entonces, cifra el mensaje m, calcula la etiqueta MAC y lo envía a P_b.

(10-11) P_b no es capaz de verificar la etiqueta o descifrar el mensaje. Entonces, envía el texto cifrado y etiquta de mensaje a P_e.

(12) P_e procesa el mensaje recibido. Calcula sec_{int} igual que la de P_a, porque $g^{x_{b,z_j}} = g^{x_{e,z_j}}$ y $g^b = g^e$. El sec_{int} es usado para calcular $(k_{E_{nc},AE} = k_{E_{nc},AB}, k_{MAC,AE} = k_{MAC,AB})$, y por tanto P_e es capaz de leer y verificar el mensaje.

Prevención de ataques UKS

Incluyendo las identidades de las dos partes en la función de derivación de claves previene el ataque. Sin embargo, este ataque UKS es evitado por el núcleo de Signal, porque las claves derivadas no dependen de las identidades de las dos partes.

6

Implementación beta de curvas elípticas

En este capítulo se explica el programa realizado en lenguaje C, usando el entorno de desarrollo Dev-C++, para implementar las curvas elípticas. El programa se divide en 4 fases: calular los puntos de una curva dada, sumar dos puntos de los calculados anteriormente, codificar un mensaje usando los puntos de la curva y por último, decodificar el mensaje.

Las curvas elípticas son utilizadas para implementar criptosistemas basados en el logaritmo discreto, con la ventaja de utilizar claves más pequeñas que repercute directamente en la utilización de menos memoria y hardware más pequeño. La curva elíptica usada en este programa es del tipo $y^2 = x^3+ax+b$ en un cuerpo finito $\mathbb{Z}_p$.
Para la primera fase, el usuario teclea los valores de a, b y el número primo p. Entonces, el programa calcula todos los puntos pertenecientes a esa curva.

```
87              // the additions
88              //2P
89          if ( x1 == x2 && y1 == y2 )
90          {
91
92              s = (3*(x1*x1) + a)/ (2*y1);
93              s=s%prime;
94              while(s < 0)
95                  s += prime;
96              printf("Lambda modulo %d: %d\n", prime, s);
97
98              xR = ((s*s) - 2*x1);
99              xR=xR%prime;
100             while(xR < 0)
101                 xR += prime;
102
103             yR = (-y1 + s*(x1 - xR));
104             yR = yR%prime;
105             while(yR < 0)
106                 yR += prime;
107
108             printf("2P = (%d,%d)\n", xR, yR); //when the two points are equal, P=Q
109             printf("\n");
110         }
```

Figura 6.1: Código del programa para calcular puntos de una curva

En la segunda fase del programa, el usuario inserta dos puntos P y Q de los calculados anteriormente en la fase 1. Dependiendo de si son iguales o distintos, el programa realiza la suma de ellos, i.e, P+Q=R.
En la criptografía de curvas elípticas, el punto R se utiliza como clave pública que es la suma de los puntos P y Q, los cuales corresponden a la clave de tipo privada.
De este modo, al conocer un punto de una curva, en este caso R, la información cifrada con este valor no se compromete, ya que la curva elíptica tiene infinidad de puntos y para poder descifrar un mensaje se necesitan conocer los parámetros P y Q para obtener su suma y así poder descifrar el mensaje.

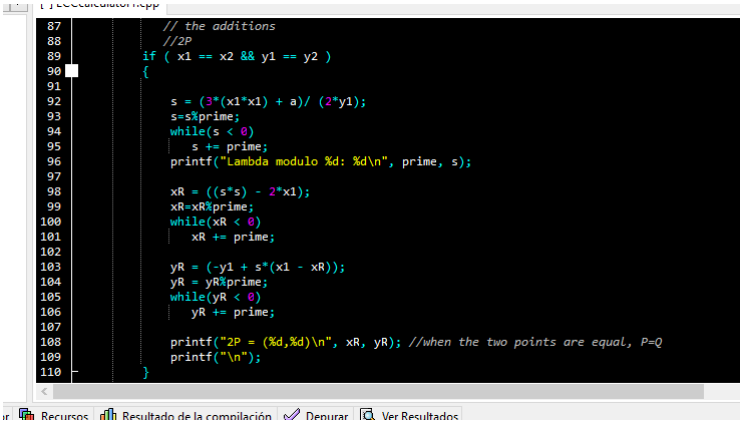


Figura 6.2: Código del programa para la suma de dos puntos iguales.

A continuación, en la siguiente fase, el usuario inserta un mensaje y el programa codifica cada uno de los caracteres del mensaje introducido a puntos de la curva dada.

La imagen 6.3 muestra un ejemplo de esta fase realizado con la curva elíptica $y^2 = x^3 + x + 6$ y en $\mathbb{Z}_{101}$.

El proceso inverso, es decir, el paso de punto de una curva a un número, que en este caso será $j = \{0, ..., 35\}$, es bastante sencillo:
$x = jh + m \rightarrow \frac{x}{h} = j + \frac{m}{h} \rightarrow \lfloor \frac{x}{h} \rfloor = \lfloor j + \frac{m}{h} \rfloor = j$, pues $\frac{m}{h} < 1$.
De este modo, para recuperar el número a partir de las coordenadas del punto $P_j = (x, y)$ basta con calcular $\lfloor \frac{x}{h} \rfloor$.

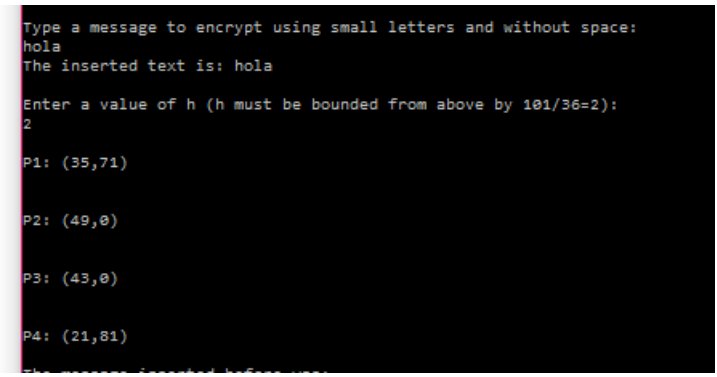


Figura 6.3: Ejemplo de codificar mensajes.

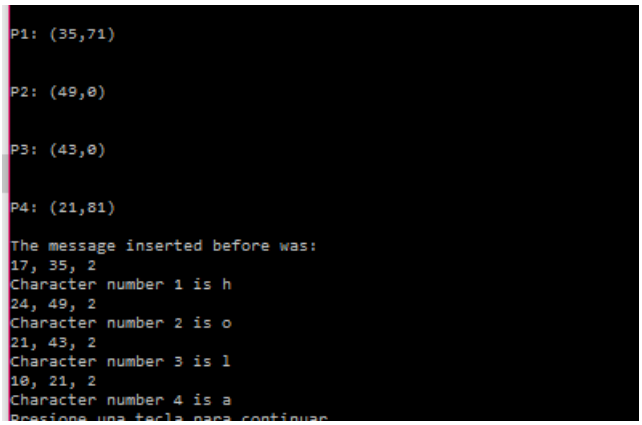


Figura 6.4: Ejemplo decodificar mensaje

7

Conclusión

Del estudio y análisis del protocolo Signal, así como de los algoritmos involucrados en el proceso de encriptación y desencriptación, extraemos diversas conclusiones, que se exponen a lo largo de este escrito.

El protocolo Signal utiliza un conjunto de primitivas criptográficas. La criptografía de clave pública se lleva a cabo con Elliptic curve Diffie-Hellman usando la función Curve25519. AES es usado para la encriptación simétrica y para la autenticación de mensajes se utiliza HMAC-SHA256. El algoritmo Double Ratchet proporciona cifrado extremo a extremo basado en una clave secreta compartida derivada de 3DH y permite el descifrado de mensajes desordenados con una reducción mínima de *forward secrecy*.
Se ve que el protocolo cumple las propiedades de seguridad estándar, i.e, confidencialidad, autenticación e integridad del mensaje y que no tiene problemas que supongan un ataque a la firmeza del protocolo. Además, se han podido estudiar las tres exigentes características de las que Signal provee: *forward secrecy*, *future secrecy* y *deniability* (repudio).

La implantación del cifrado *end-to-end* en las aplicaciones de mensajería instantáneas ha dado visibilidad a cuestiones relacionadas con la privacidad y seguridad en la vida cotidiana, despertando el interés de los medios de comunicación por aspectos poco conocidos como la criptogría.
De las aplicaciones estudiadas anteriormente que implementan este protocolo, analizemos ahora cuál aporta mayor seguridad al usuario.
Facebook comparte datos de los usuarios para la publicidad y como WhatsApp pertenece a su compañía, entonces esta última también lo realiza. La aplicación Signal es de código libre y disponible en GitHub, mientras que en WhatsApp solo el protocolo Signal que se usa para el cifrado es de código libre. Estas dos aplicaciones utilizan el mismo protocolo, sin embargo se dis-

tinguen en como almacenan información personal del usuario y metadatos. WhatsApp almacena datos de sesión, información del dispositivo, información de contacto, cookies, última sesión y la localización (Privacy Policy de WhatsApp). Por el contrario, la única información que almacena la aplicación Signal es el número de teléfono con el que se registra y la última vez que se entró en el servidor. En Google Allo y Duo si se quiere usar la característica de cifrado *end-to-end*, el usuario debe abrir la conversación en modo 'incognito'. De no ser así, Google almacenará todas las conversaciones y mensajes enviados.
Se hace especial énfasis en la aplicación WhatsApp por su gran afluencia de clientes, viendo que ésta, a pesar de su actual orientación hacia la búsqueda de privacidad, tiene un vacío en lo referente a la protección de datos privados.

De lo analizado se puede extraer que, para aquellos usuarios especialmente concienciados con su seguridad, es aconsejable la app Signal Private Messenger. Esta aplicación es la única que usa protocolos criptográficos de fuente abierta para mantener los mensajes seguros ([26]).
Como conclusión general, tenemos que el protocolo cumple las expectativas, así como proporciona el nivel de seguridad más exigente del mercado. Su código libre permite el acceso a interesados y desarrolladores, contribuyendo a una constante mejora y siendo un portal que impulsa la privacidad.

Bibliografía

[1] Baumslag, G., Fine, B., Kreuzer, M. y Rosenberger, G. *A Course in Mathematical Cryptography*. De Gruyter, 2015.

[2] Bellare, H., Canetti, R. y Krawczyk, H. *HMAC: Keyed-Hashing for Message Authentication*. RFC2104, 1997.

[3] Bernstein, D. *Curve25519:new diffie-Hellman speed records*, in PKC, 2006, URL: http://cr.yp.to/ecdh/curve25519-20060209.pdf

[4] Bonneau et al. *SoK: Secure Messaging*. IEEE Symposium on, 2015, pp. 232-249.

[5] Boyd, C. y Mathuria, A. *Protocols for Authentication and Key Establishment*. Springer, 2003.

[6] Caballero Gil, P. *Introducción a la Criptografía*. RaMa, 2002.

[7] Caballero Gil, P. *Seguridad en Sistemas Informáticos*. Curso: Tercero. ETSII.

[8] Cohn-Gordon, K., Cremers, C., Dowling, B., Garratt, L. y Stebila, D. *A Formal Security Analysis of the Signal Messaging Protocol*, 2016.

[9] Delfs, H. y Knebl, H. *Introduction to cryptography. Principles and Application*. Springer, 2002.

[10] Descriptions of SHA-256, SHA-384 y SHA-512. URL: www.iwar.org.uk/comsec/resources/cipher/sha256-384-512.pdf

[11] Diffie, W. y Hellman, M. *New Directions in Cryptography*, IEEE Transactions on information theory, 22(6), 1976.

[12] Ferguson, N. y Schneier, B. *Practical Cryptography*. Wiley, 2003.

[13] Frosch, T. et al. *How Secure is TextSecure?*

[14] Haraty, R., ElKassar, A. y Fanous, S. *Hardening the ElGamal Cryptosystem in the Setting of the Second Group of Units.*, 2014.

[15] Hellman, M. *An Overview of Public Key Criptography*, IEEE Communications, 16(6), 1978.
[16] Hoffstein, J., Pipher, J. y Silverman, J. *An Introduction to Mathematical Cryptography*. Springer, 2008.
[17] Huber, M. et al. *Privacy and Data Protection in Smartphone Messengers*, 2015.
[18] Huber, M., Rottermanner, C., Shroder, S. y Wind, D. *When SIGNAL hits the Fan: On the Usability and Security of State-of-the-Art Secure Mobile Messaging*, July 2016.
[19] Katz, J. y Lindell, Y. *Introduction to Modern Cryptography*. Chapman and Hall, CRC Press, 2007.
[20] Koblitz, N. *Elliptic Curve Cryptosystems*, 48(177), Jan 1987, pp. 203-209.
[21] Math Stack Exchange. URL: https://math.stackexchange.com/questions/2198139/elliptic-curve-formulas-for-point-addition.
[22] National Institute of Standards and Technology (NIST). *Advanced Encryption Standard*. FIPS PUB 197, U.S., Nov 2001.
[23] Nitaj, A. *A new attack on the KMOV cryptosystem*. Laboratoire de Mathematiques Nicolas Oresme Universite de Caen, France.
[24] Open Whisper Systems, Double Ratchet. URL: https://whispersystems.org/docs/specifications/doubleratchet/
[25] Open Whisper Systems, Google Allo. URL: https://whispersystems.org/blog/allo/
[26] Signal Private Messenger, URL: https://play.google.om/store/apps/details?id=org.thoughtcrime.securesms
[27] Silverman, J. *The Arithmetic of Elliptic Curves*. Springer, 2009.
[28] Skype partnership, URL: https://signal.org/blog/skype-partnership/
[29] WhatsApp Encryption Overview, URL: https://www.whatsapp.com¿security
[30] X3DH, URL: https://signal.org/docs/specifications/x3dh/

Study of the Cryptographic Tool Signal

Ashneet Khandpur Singh
Facultad de Ciencias · Sección de Matemáticas
Universidad de La Laguna
alu0100775389@ull.edu.es

Abstract

We study the new cryptographic security protocol, Signal, that provides end-to-end encryption for instant messaging. The protocol is in use in many popular apps like WhatsApp, Facebook Messenger and Google Allo. We focus in understanding Signal and how it functions. For that purpose, a study has been done on the algorithms and ciphers that are part of the protocol: Triple Diffie-Hellman handshake, Double Ratchet, Curve25519, AES-256 and HMAC-SHA256. Finally, a brief analysis is performed on the security of Signal and also, an implementation of elliptic curves using C code.

1. Introduction

There are two popular kinds of cryptographic protocols: asymmetric-key or public key protocols and symmetric-key or private key protocols. Among the first one we can mention Diffie-Hellman key exchange algorithm, RSA, ElGamal cryptosystem and elliptic curves. We will study these in chapter 2. The symmetric-key protocols includes DES, AES and HMAC, that we will study in chapter 3. The Signal protocol combines these algorithms and ciphers to provide end-to-end encryption and it has been implemented in messaging applications such as WhatsApp, Facebook Messenger and Google Allo. The chapter 4 explains how this works. To know if Signal is secure, we show a security analysis done by Frosch et al. in chapter 5.

2. Key management

The purpose of this chapter will be the study of public key cryptosystems like Diffie-Hellman, Double Ratchet Algorithm and Elliptic Curves Cryptography. The security of these cryptosystems is based on the difficulty of the Discrete Logarithm Problem (DLP).
(DLP) Let G be a finite field. The discrete logarithm problem is a problem of solving a equation of the type

$$a^x = b \tag{1}$$

with $x \in N$ y $a, b \in G$.
Also, we will work on finite fields for the cryptosystems to be secure.
Elliptic curves can provide versions of public-key methods that, in some cases, are faster and use smaller keys, while providing an equivalent level of security. Elliptic curves have numerous applications in cryptography.

We define an elliptic curve over a finite field F_p to the equation

$$E : Y^2 = X^3 + AX + B$$

with $A, B \in F_p$ satisfying $A^3 + 27B^2 \neq 0$ and we denote by $E(F_p) = (x, y) : x, y \in F_p$

verify $y^2 = x^3 + Ax + B \cup \mathcal{O}$

to the points in E with coordinates in F_p.

3. Integrity and confidentiality protection

The other two cryptographic algorithms that are part of the Signal protocol are HMAC-SHA256 and AES-256. The first one is used for integrity and the second one for encyption.
HMAC-SHA256 is a type of hash keyed algorithm that is created from the hash function SHA-256 and is used as HMAC (hash-based message authentication code).

Advanced Encryption Standard (AES) is a symmetric key algorithm, i.e the same key is used to encrypt and decrypt the data.

4. Technologies that implement Signal

Signal is a cryptographic protocol developed by Open Whisper Systems in 2013. The main goal of the protocol is to send a message in a secure way, despite of the receiver being 'offline'. The protocol provides characteristics of security such as confidentiality, integrity, authentication, forward secrecy, future secrecy, deniabilty and end-to-end encryption. Some of the technologies that use the cryptographic protocol Signal are WhatsApp, Facebook Messenger, Google Allo, Signal Private Messenger and Skype. The first two have more than one billion users.

5. Signal's Security

More than 50 per cent of the world's population uses applications that have Signal protocol to protect communications. The people who use these apps rely on the security that is provided to them by this protocol. To know whether to trust or not on this protocol, Frosch et al. did an security analysis on Signal and found that there are no major flaws in it and it is both secure and resistant to attack.

6. Beta implementation of elliptic curves

Elliptic curve cryptography is implemented using C language. We divided the code into four phases: calculate points of the given curve, add two of the calculated points and encrypt and decrypt a message using the points of the curve.

References

[1] Bellare, H., Canetti, R. y Krawczyk, H. *HMAC: Keyed-Hashing for Message Authentication*. RFC2104, 1997.

[2] Caballero Gil, P. *Introducción a la Criptografía*. RaMa, 2002.

[3] Ferguson, N. y Schneier, B. *Practical Cryptography*. Wiley, 2003.

[4] Frosch, T. et al. *How Secure is TextSecure?*

[5] Hoffstein, J., Pipher, J. y Silverman, J. *An Introduction to Mathematical Cryptography*. Springer, 2008.

[6] Koblitz, N. *Elliptic Curve Cryptosystems*, 48(177), enero 1987, 203-209.

[7] National Institute of Standards and Technology (NIST). *Advanced Encryption Standard*. FIPS PUB 197, U.S., 2001.

[8] WhatsApp Encryption Overview, URL: https: //www.whatsapp.com>security