Universitat de les Illes Balears

escola politècnica superior

Treball Final de Grau

GRAU DE MATEMÀTIQUES

Revisión de las librerías criptográficas de curva elíptica actuales

MANUEL TRUJILLO VANRELL

Tutor
Macià Mut

Escola Politècnica Superior
Universitat de les Illes Balears
Palma, 4 de septiembre de 2016

PREFACIO

Cuando pensé en realizar el Trabajo de Fin de Grado, tenía la duda de como se enfocaba un trabajo de este tipo. El trabajo se lo solicité al profesor Macià Mut, sobre criptografía asimétrica. Me propuso realizarlo sobre criptografía de curva elíptica y de una forma que me pareció muy interesante: un estudio sobre el rendimiento de los algoritmos criptográficos que usan dicha propiedad. Aunque mis conocimientos sobre el tema se limitaban a las lecciones de *Codificación y Criptografía*, impartida por el profesor L. Huguet, y algunos experimentos sencillos, me pareció que la forma de realizar el trabajo implicaba adquirir un conocimiento asequible y una posibilidad de asomarse a la actualidad y realidad de la materia.

El trabajo me resultó inicialmente desesperante por el vasto panorama teórico de las curvas elípticas. Una vez admitido, que no superado, fue laborioso y errático el repasar y probar, cuando era posible, las múltiples aplicaciones que las utilizan. Después llegó el mareo de protocolos y especificaciones. Continuó con el suplicio de redactarlo de forma más o menos coherente, y al final, supongo que gracias a todo lo anterior, se hizo la luz: Empecé a familiarizarme con el formato de las publicaciones científico-técnicas y hacerme una idea, aunque puede que falsa, del estado actual de la criptografía de curva elíptica.

Además, y gracias al profesor Mut, como el desarrollo del trabajo coincidió en parte con la fecha de presentación de artículos para la **XIV REUNIÓN ESPAÑOLA SOBRE CRIPTOLOGÍA Y SEGURIDAD DE LA INFORMACIÓN (RECSI 2016)**, organizada por la UIB, en cuanto se pudo plasmar parte de la comparativa y revisarla con la colaboración de otros profesores, se envió al Comité del Programa. Nos resultó grato que fuera admitida, interpretando que algo de interés puede tener una comparativa de esta naturaleza.

Finalmente, deseo agradecer la ayuda y lo aprendido, con este trabajo y las lecciones recibidas, a los profesores Macià Mut y Llorenç Huguet.

ÍNDICE GENERAL

Acrónimos

ANSI American National Standards Institute

ARM ARM Holdings plc.

ASIC Application Specific Integrated Circuit

DH Diffie-Hellman

DSA Digital Signature Algorithm

DSS Digital Signature Standard

ECC Criptografía de Curva Elíptica (del inglés: Elliptic curve cryptography)

ECDH Elliptic curve Diffie Hellman

ECDLP Del inglés, Elliptic curve discrete logarithm problem

ECDSA Elliptic Curve Digital Signature Algorithm

ECMQV Elliptic Curve Menezes Qu Vanstone

EFF Electronic Frontier Foundation

ElGamal ElGamal encryption system

IEEE Institute of Electrical and Electronics Engineers

IETF Internet Engineering Task Force

IoT Internet of Things

ISO the International Organization for Standardization

IoT Internet of Things

MIT Massachusetts Institute of Technology

MQV Menezes Qu Vanstone

NIST National Institute of Standards and Technology

NUMS Nothing Up My Sleeve

NSA National Security Agency

PHP Hypertext Preprocessor

PLD Problema del Logaritmo Discreto

RFID Radio Frequency IDentification

RSA Rivest, Shamir y Adleman

RTP Real time Transport Protocol

S/MIME Secure / Multipurpose Internet Mail Extensions

SECG Standards for Efficient Cryptography Group

SSH Secure SHell

SSL Secure Sockets Layer

SRTP Secure Real time Transport Protocol

TLS Transport Layer Security

VoIP Voice over IP

ZRTP Zimmermann Real time Transport Protocol

RESUMEN

EN la actualidad el RSA (Rivest, Shamir y Adleman) suele considerarse el criptosistema predominante en la criptografía asimétrica para funciones de cifrado y firma digital en aplicaciones y protocolos de comunicación. Actualmemente, la ECC (Criptografía de Curva Elíptica, del inglés: Elliptic curve cryptography) desafía este dominio.

Este artículo presenta una introducción a la ECC y revisa algunas librerías criptográficas actuales que implementan la ECC en varios campos de la informática, desde TLS (Transport Layer Security) al IoT (Internet of Things), pasando por Android. Se han realizado pruebas comparativas directas sobre los paquetes OpenSSL, GnuPG, MSRECClib y la plataforma Arduino.

De la revisión realizada y los resultados obtenidos en las pruebas, se desprende que la fortaleza de ECC es considerada similar o superior a la criptografía asimétrica tradicional pero usa claves más cortas (en número de bits necesarios para almacenar y operar la clave), estando disponible para virtualmente cualquier aplicación, protocolo y plataforma, por lo que puede afirmarse que está preparada para reemplazar al estándar actual de la criptografía asimétrica. Es de esperar que la ECC pase a ser una tecnología imprescindible en las comunicaciones y transacciones de todo tipo, especialmente en comercio electrónico y el ámbito de la administración digital.

CAPÍTULO 1

LA CRIPTOGRAFÍA ASIMÉTRICA

La criptografía asimétrica o de clave pública utiliza dos claves para cada usuario, una pública y otra privada. La ventaja de este esquema es que soluciona el problema de distribución de claves, problema que tiene la criptografía simétrica de una sola clave. Si el remitente cifra un mensaje con la clave del destinatario, solo este último podrá descifrar el mensaje con su clave privada. También es posible firmar el mensaje con la clave privada del remitente y solamente la clave pública del remitente certificará el origen del mensaje al destinatario.

Claro está que sabiendo la clave pública no se tiene que tener capacidad para saber la privada. Además la fortaleza del sistema tiene que ser comparable a la seguridad de la criptografía simétrica o de una sola clave privada. Con este esquema y algunos algoritmos auxiliares se consiguen unos protocolos para una transmisión del mensaje de forma privada, certificada e íntegro, sin el problema de distribución de claves por canales inseguros.

Tradicionalmente se ha utilizado para estos esquemas los algoritmos Rivest, Shamir y Adleman (RSA), Diffie-Hellman (DH) y Digital Signature Algorithm (DSA), que trabajan usando números primos grandes (problema de factorización de enteros) o logaritmos discretos en cuerpos finitos, esto es, Problema del Logaritmo Discreto (PLD). Su utilidad y seguridad está fuera de toda duda, tanto por los estudios teóricos que lo soportan, como por el uso que se está haciendo de ellos desde hace tiempo sin encontrar debilidades esenciales [28].

Con todo, a medida que aumenta la capacidad de cálculo de los ordenadores, se necesitan claves de mayor longitud para mantener un nivel de seguridad comparable a la criptografía simétrica [4]. El tiempo de cálculo para las operaciones necesarias aumenta de forma muy acusada con el tamaño de la clave, haciendo que para determinado nivel de seguridad tarden más de lo deseado o simplemente sea inviable.

Una solución a este problema es la Criptografía de Curva Elíptica (del inglés: Elliptic curve cryptography) (ECC). Consiste en un uso similar de los algoritmos de criptografía asimétrica, pero en lugar de operar sobre números primos o un cuerpo finito, se hace sobre los puntos de una curva elíptica definida en un cuerpo finito. Esto hace que las claves resultantes, esto es, la representación binaria de los puntos elegidos sobre la curva, sean de un tamaño menor que la representación de las claves en la criptografía asimétrica tradicional.

Las bases mateméticas de la ECC son sólidas y los algoritmos están bien estudiados. Como muestra inicial de la eficiencia y ancho de banda de la ECC, el siguiente cuadro 1.1 compara diversos criptosistemas mostrando tamaños de clave comparables en términos de esfuerzo computacional para el criptoanálisis [5]. Puede apreciarse como la ECC puede utilizar un tamaño de clave considerablemente más pequeña en comparación con RSA. Por otra parte, el esfuerzo computacional necesario para el uso de ECC y de RSA para claves con la misma longitud es similar [16]. Gráficamente se aprecia mejor dicho crecimiento en la figura 1.1.

Cuadro 1.1: Equivalencia en tamaños de clave

Criptosistema Simetrico (tamaño clave en bits)	**Criptosistema basado en ECC** (representación de de $P \epsilon E(\mathbb{Z}_p)$) en bits)	**Criptosistema RSA/DSA** (tamaño del módulo en bits)
56	112	512
80	160	1024
112	224	2048
128	256	3072
192	384	7680
256	512	15360

Equivalencia de tamaños de clave para en términos de esfuerzo computacional para su criptoanálisis. $P \epsilon E(\mathbb{Z}_p)$ hace referencia a un punto de la curva elíptica y el módulo al producto de los números primos utilizados en el criptosistema.

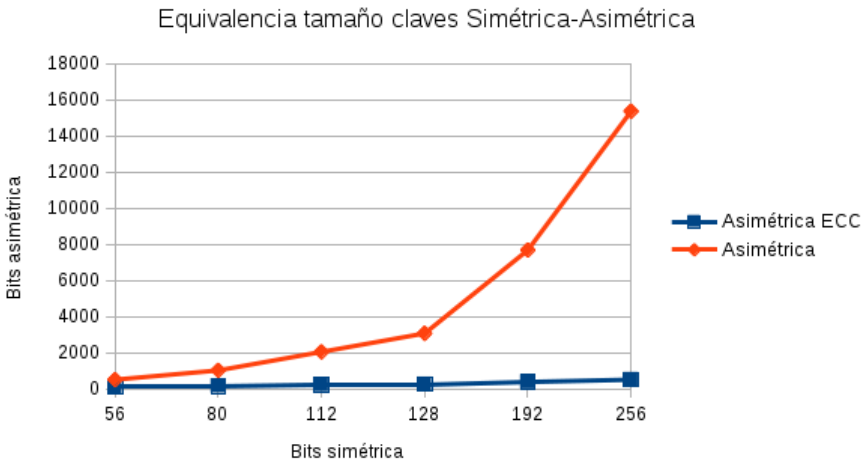


Figura 1.1: Equivalencia en tamaños de clave

En definitiva, el uso de la ECC con una longitud de clave más corta que en un criptosistema RSA puede ser comparativamente igual o más seguro, y de menor carga computacional. Aunque hemos comentado que la base teórica de la ECC esta establecida desde hace algún tiempo, los productos y librerías criptográficas que implementan sus cálculos aún no están suficientemente estudiados ni a nivel criptoanalítico ni de rendimiento [10]. En consecuencia, el nivel de confianza sobre criptosistemas como el RSA aún sigue siendo mayor que en la ECC provocando una desconfianza para el uso de este tipo de criptografía.

En el estudio que se presenta en este trabajo se expondrán las bases de la ECC y sus protocolos de cifrado, comparándolos con RSA, DH y DSA, realiza una evaluación comparativa del rendimiento y valoración de algunas librerías criptográficas para diferentes aplicaciones y plataformas que implementan cálculos con curva elíptica, aportando, por tanto, una introducción y difusión para el uso de las mismas. Un estudio más detallado y sistemático, en la línea del proyecto eBacs [22], específico para ECC, sería deseable.

CAPÍTULO 2

PROTOCOLOS DE LA CRIPTOGRAFÍA ASIMÉTRICA MÁS UTILIZADOS

La noción de criptografía de clave pública fue introducida por Whitfield Diffie y Martin Hellman en 1976 [11]; desde entonces un concepto importante en criptografía es la aparente iresolubilidad del problema del logaritmo discreto. Taher ElGamal describió cómo este problema se puede utilizar en el cifrado de clave pública y en esquemas de firma digital [12]. El criptosistema ElGamal encryption system (ElGamal) se han refinado y está incorporado en diversos protocolos para satisfacer una variedad de aplicaciones. Una de sus extensiones, DSA, constituye la base para el algoritmo de firma digital del Gobierno de los Estados Unidos, a propuesta del National Institute of Standards and Technology (NIST) para el estándar de firma digital Digital Signature Standard (DSS).

No obstante, la respuesta de más éxito al desafío lanzado por Diffie-Hellman fue desarrollado en 1977 por Ron Rivest, Adi Shamir y Len Adleman en el Massachusetts Institute of Technology (MIT) y publicado por primera vez en 1978 [20]. Muchos de los productos y estándares que utilizan la criptografía de clave pública para el cifrado y para las firmas digitales utilizan RSA. Así, protocolos como Secure SHell (SSH), Secure / Multipurpose Internet Mail Extensions (S/MIME), y Secure Sockets Layer (SSL)/Transport Layer Security (TLS) se basan en RSA para funciones de cifrado y firma digital.

2.1 Protocolo RSA

Desarrollado en 1977 por por Ron Rivest, Adi Shamir y Leonard Adleman en el MIT, fue uno de los primeros algoritmos prácticos de cifrado con clave pública. Es quizás el más utilizado actualmente y desde el año 2000 está libre de pantente.

Su fortaleza se basa en la dificultad de factorizar enteros grandes, problema que se considera intratable a partir de ciertos tamaños. En relación al tamaño de la clave usada en RSA, medida en número de bits para la representación del módulo utilizado en el algoritmo, tenemos por ejemplo los siguientes grados de seguridad: Para 256 bits

se puede hacer con un ordenador doméstico en pocas horas, para 512 bits con una red de ordenadores, como se demostró en 1999 [6]. En 2010 se logró romper para 768 bits y los autores desaconsejaban el uso de 1024 bits para los siguientes 3 o 4 años [7]. Para 1024 bits sin embargo se sigue considerando seguro por los laboratorios RSA [8], pero no parece tener el consenso de todo el mundo.

El tamaño de las claves usadas en la práctica está entre 1024 y 4096 bits, siendo este último tamaño de clave tan grande que hace demasiado costoso el algoritmo y crea incompatibilidad con muchos dispositivos ya existentes.

El algoritmo de cifrado y descifrado de RSA es costoso computacionalmente, por lo que se suele utilizar RSA para la firma electrónica, junto con el intercambio de claves de cifrado simétrico y resultados de funciones Hash. Otra limitación es la generación aleatoria de las claves simétricas, un aspecto crítico del protocolo en su conjunto [1].

2.1.1 Esquema del RSA

Veamos el esquema básico del algoritmo RSA (resumido en la figura 2.1), pues el protocolo RSA completo que se utiliza en la práctica tiene detalles técnicos adicionales, como funciones de relleno y elección segura de los parámetros que no procede comentar para ver como funciona. Se procede de la siguiente manera:

Alice quiere mandar a Bob un mensaje cifrado. Para ello acuerdan un sistema de cifrado simétrico y ciertos parámetros, esto es, dos primos grandes de tamaño parecido, p y q. Su tamaño será tal para que la clave resultante tenga un tamaño en bits como el valor pq. Se procede en tres pasos:

Generación de claves

Alicia elige un e_a entero menor que $\phi(n) = (p-1)(q-1)$ y coprimo con este valor. Calcula d_a tal que $ed = 1 \bmod phi(n)$. Bob procede de la misma manera, y obtendrá e_b y d_b. Alice tiene como clave privada (d_a,n) y pública (e_a,n).
Bob tiene como clave privada (d_b,n) y pública (e_b,n).

Cifrado

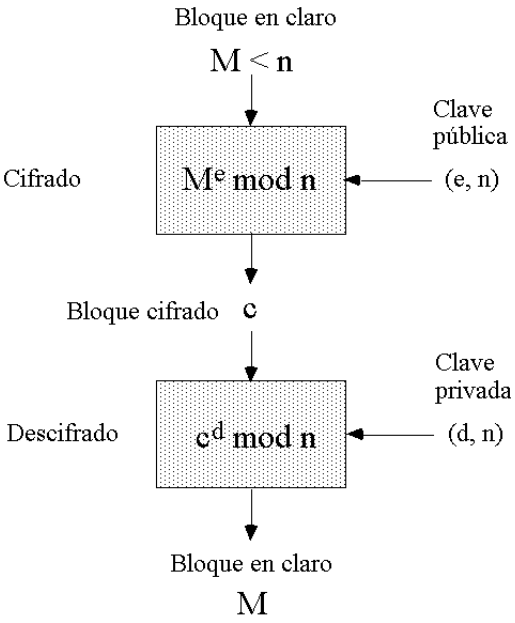
Alice envía a Bob el mensaje cifrado con la clave pública de Bob, C=M^{e_b} $(mod\ n)$.

Descifrado

Bob calcula con su clave privada $M = C^{d_b}$ $(mod\ n)$.

Se puede demostrar que $M = (C^{e_b})^{d_b}$ $(mod\ n)$ [23].

Figura 2.1: Esquema del RSA



De una forma similar se puede enviar mediante una función Hash una verificación y firma del mensaje. Suele hacerse esto con distintos pares de claves. Queda pues de manifiesto que el cálculo del inverso de *e*, no tiene que ser factible para un tercero que conozca solo *d*, y tampoco la factorización de *n*.

Para más detalles matemáticos del algoritmo se sugiere consultar el documento de Burt Kaliski, del propio laboratorio RSA [23].

2.1.2 Ejemplo de RSA

Veamos un ejempo con números suficientemente pequeños para manejarlos mentalmente, aunque sean en la práctica a todas luces insuficientes en seguridad:

Generación de claves

Se elige p=3 y q=11.

Por tanto $n = p * q = 33$ y $\phi(n) = (p-1) * (q-1) = 2 * 10 = 20$

Alice elige e_a tal que $1 < e_a < \phi(n)$ y coprimo con n, Sea $e_a = 7$.

Se calcula d_a tal que $(d * e)\ (mod\ \phi(n)) = 1$. Tenemos $d = 3$, pues $(3*7)\ (mod\ 20) = 1$.

La clave pública de Alice es $(e_a, n) = (7, 33)$ y la privada $(d_a, n) = (3, 33)$.

Cifrado

Si Bob quiere mandar el mensaje M=2 a Alice, lo cifra de la forma:

$$C = M^{e_a} \ (mod\ n) = 2^7 \ (mod\ 33) = 29.$$

Descifrado

Alice recibe $C = 29$, calcula $M = (C^{e_a})^{d_a} \ (mod\ n) = 29^3 \ (mod\ 33) = 2$.

2.2 Protocolo DH

Uno de los primeros algoritmos de cifrado asimétrico, concebido por Ralph Merkle y desarrollado por Whitfield Diffie y Martin Hellman, es el llamado Diffie-Hellman, abreviado, DH. Este se basa en la dificultad de calcular un logaritmo en un cuerpo finito (logaritmo discreto). Permite a un número arbitrario de usuarios acordar una clave simétrica en un canal inseguro sin contacto previo y de manera anónima. Junto con otros protocolos puede permitir la autentificación y verificación del mensaje. Las implementaciones prácticas además incorporan controles adicionales como tiempos y verificación previa para hacerlo seguro frente a terceros. El tamaño sugerido para el orden del cuerpo a utilizar, es que sea, de al menos, 1024 bits de tamaño, siendo recomendable 2048 [2].

2.2.1 Esquema del DH

Veamos como Alice y Bob pueden comunicarse (resumido en la figura 2.2):

Acuerdo de los parámetros

Se acuerda un primo p y un generaror g en $Z^*{}_p$, no importa si esto lo conoce un tercero.

Alice elige $a \in Z_{p-1}$, calcula $A = g^a \ (mod\ p)$ y se lo manda a Bob.

Bob elige $b \in Z_{p-1}$, calcula $B = g^b \ (mod\ p)$ y se lo manda a Alice.

Cálculo de la clave

Entonces ambos pueden calcular la clave $K = g^{a*b} \ (mod\ p)$, ya que:

Alice hace $B^a \ (mod\ p) = (g^b)^a = g^{ab} \ (mod\ p) = K$.

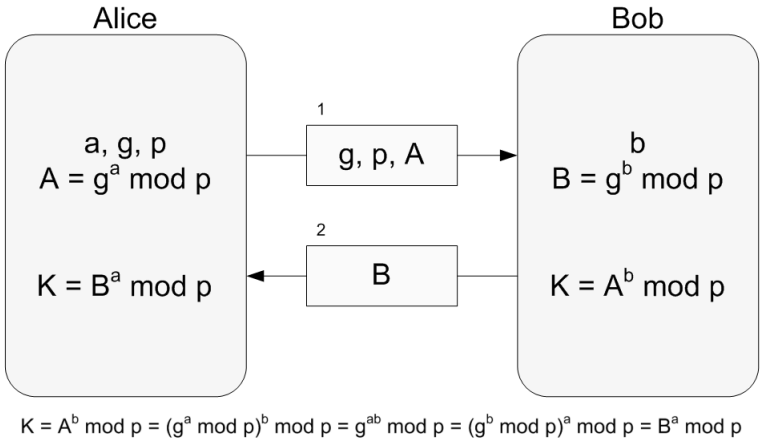
Bob hace $A^b \ (mod\ p) = (g^a)^b = g^{ba} \ (mod\ p) = K$.

Ya tienen una clave simétrica compartida de forma segura: K

Para un tercero que conozca p, g, A y B, y desconozca a y/o b, calcular K, es intratable computacionalmente bajo ciertas condiciones, entre ellas, que p tenga un tamaño de varios centenares de bits. Esto es lo que se conoce como PLD, $a = logdisc_p(A)$, $b = logdisc_p(B)$. Notar nuevamente que el razonamiento sirve no solo para dos usuarios, se puede extender a un número indetermidado con el mismo principio.

Para más detalles matemáticos del algoritmo y estándares de internet, se sugiere consultar el documento de The Internet Society de 1999 [24].

Figura 2.2: Esquema del DH



2.2.2 Ejemplo del DH

Veamos un ejemplo con números pequeños para ilustrarlo, insuficientes en la práctica, claro está:

Acuerdo de los parámetros

Alice y Bob acuerdan usar el número primo p=23 y la base g=5.

Alice elige su número secreto a=6, luego envía a Bob $g^a \ (mod\ p) = 5^6 \ (mod\ 23) = 8$.

Bob elige su número secreto $b = 15$, luego envía a Alice $g^b \ (mod\ p) = 5^{15} \ (mod\ 23) = 19$.

Cálculo de la clave

Alice calcula $(g^b)^a \ (mod\ p) = 5^{15*6} \ (mod\ 23) = 2$.

Bob calcula $(g^a)^b \ (mod\ p) = 5^{6*15} \ (mod\ 23) = 2$.

Ya tienen una clave simétrica compartida de forma segura:

Ambos conocen, y solo ellos, el valor 2, que utilizarán como clave simétrica.

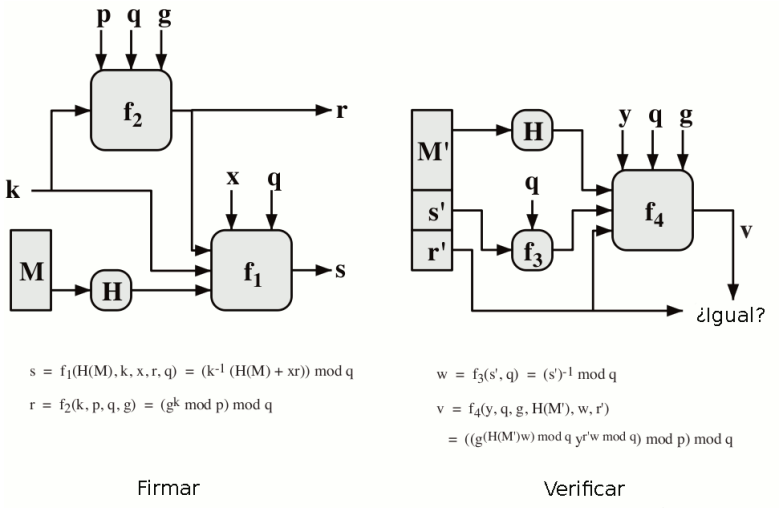
2.3 Protocolo DSA

EL DSA es un estándar del Gobierno de Estados Unidos para firmas digitales. Es más costoso computacionalmente que el RSA. Es una variante de ElGamal. Su fortaleza radica en el cálculo del logaritmo discreto. El tamaño de la clave recomendado está entre 1024 y 2048 bits [3].

2.3.1 Esquema del DSA

Veamos el esquema de funcionamiento (resumido en la figura 2.3):

Figura 2.3: Esquema del DSA



Generación de claves

Se elige un primo p de L bits, con $512 < L < 1024$ (tamaño de clave) y L múltiplo de 64.
Se calcula un primo q de 160 bits, tal que $p-1 = qz$, con $z \in$ N.
Se elige h con $1 < h < p-1$ y se calcula $g = h^z \ mod \ p > 1$.
Se elige x entre 1 y q-1.
Se calcula $y = g^x$ mod p.

(p, q, g, y), pueden ser públicos, la clave privada es x.

Firma

Aleatoriamente se elige $0 < k < q$.
Se calcula $0 \neq r = (g^k \ mod \ p) \ mod \ q$.
Se calcula $0 \neq s = k^{-1}(H(m) + r * x) \ mod \ q$, donde H(m) es una función Hash del mensaje m.

La firma es (r,s).

Verificación

Se tiene que cumplir que $0 < r, s < q$.
Calcular $w = s^{-1}\ (mod\ q)$.
Calcular $u_1 = H(m) * w\ (mod\ q)$.
Calcular $u_2 = r * w\ (mod\ q)$.
Calcular $v = (g^{u_1} * y^{u_2}\ (mod\ p)\ (mod\ q)$.
La firma es válida si $v = r$.

Para más detalles matemáticos del algoritmo y elección de parámetros se sugiere consultar el documento publicado por el NIST para DSS [25].

2.3.2 Ejemplo del DSA

Veamos un ejempo con números pequeños para ilustrarlo, notar que el tamaño de los números utilizados en la práctica es del orden de 1024 bits:

Generamos las claves

Sea L 64 bits, y q de 8 bits. Por ejemplo p=389 y q=97.
Sea h=2, calculamos $g = 2^{(389-1)/97}\ (mod\ 389) = 16$.
Sea x=25, calculamos $y = 16^2 5\ (mod\ 389) = 142$.
La clave pública es (389,97,16,142).
La clave privada es 25.

Firma

Sea H(m) = 75 por ejemplo.
Se genera un número aleatorio $k = 66$.
Se calcula $r = (16^{66}\ (mod\ 389))\ (mod\ 97) = 76$.
Se calcula $s = (66^{-1} * (75 + 25 * 76))\ (mod\ 97) = 2$.
La firma será (76, 2).

Verificación

Vemos que $0 < 2,76 < 97$.
Se calcula $w = 2^{-1}\ (mod\ q) = 49$.
Se calcula $u_1 = (75 * 49)\ (mod\ 97) = 86$.
Se calcula $u_2 = (76 * 49)\ (mod\ 97) = 38$.
Se calcula $v = ((16^8 6 * 142^{38})\ (mod\ 389)\ (mod\ 97) = 76$.
Como $r = v$ se acepta la firma.

CAPÍTULO 3

CURVAS ELÍPTICAS

3.1 Introducción

Las curvas elípticas han sido de gran importancia en las matemáticas desde hace varios siglos por sus propiedades singulares. Aunque su uso en criptografía es bastante reciente, pudiendo situarse su origen en unas publicaciones de V.Miller y N.Koblitz, en torno a 1985. Se buscaba el implementar el PLD, en lugar de sobre el grupo multiplicativo de un cuerpo, sobre los puntos de una curva elíptica en un cuerpo discreto. Su motivación era evitar ciertos ataques criptoanalíticos con unas claves más pequeñas.

Sin embargo, la idea quedó en el ámbito académico hasta hace relativamente poco, hasta el punto de que en 1997 el propio R.Rivest (RSA) dudaba de su sólida estructura. Su desarrollo se debe en gran parte a la compañía Certicom (creada en 1985 por S.A. Vanstone y R.Mullin) y al grupo investigador de la Universidad de Waterloo (A.J. Menezes y S.A.Vanstone entre otros investigadores) [28].

3.2 Ecuación de la curva elíptica

Una curva elíptica se define mediante una ecuación cúbica en un cuerpo $\mathbb{K}$ arbitrario. Los puntos de la forma *(x,y)*, con x e y que pertenecen a la clausura algebraica del cuerpo y cumplen dicha ecuación se les considera pertenecientes a la curva. Si x e y ambos pertenecen al cuerpo, entonces *(x,y)* es llamado punto K-racional [29].

3.2.1 Definición formal

La curva elíptica se define como una curva plana no singular de grado 3 junto a un punto racional prefijado, que se denomina punto base. Cualquier curva elíptica sobre $\mathbb{P}^2$ (y característica del cuerpo diferente de 2 y 3) puede ser escrita de la forma:

$$y^2 = x^3 + Ax + B$$

Podemos ver dos ejemplos sobre $\mathbb{R}$ en el plano afín, en las figuras 3.1 y 3.2.

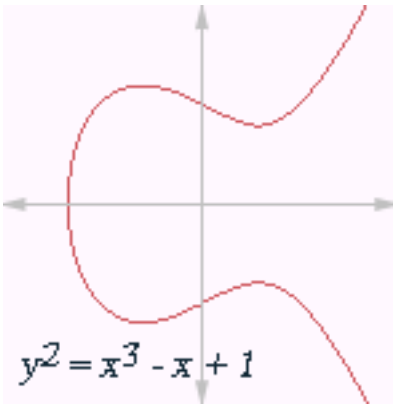


Figura 3.1: Curva en un trazo
Fuente: Wikipedia

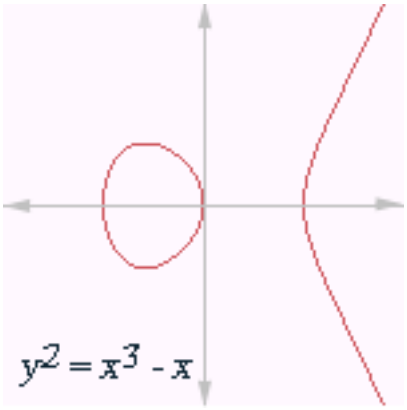


Figura 3.2: Curva en dos trazos
Fuente: Wikipedia

Sin embargo, si estamos sobre un cuerpo finito, por ejemplo $\mathbb{F}_p$ con *p=19,97,127 y 486*, la curva $y^2 = x^3 - 7x + 10$ módulo p, su representación gráfica no nos parecen más que un conjunto aleatorio de puntos sobre el plano, como vemos en la figura 3.3. Si estamos sobre $\mathbb{F}_q$ con q muy grande, del orden de cientos de bits, sencillamente será una nube semejante a un ruido más o menos aleatorio.

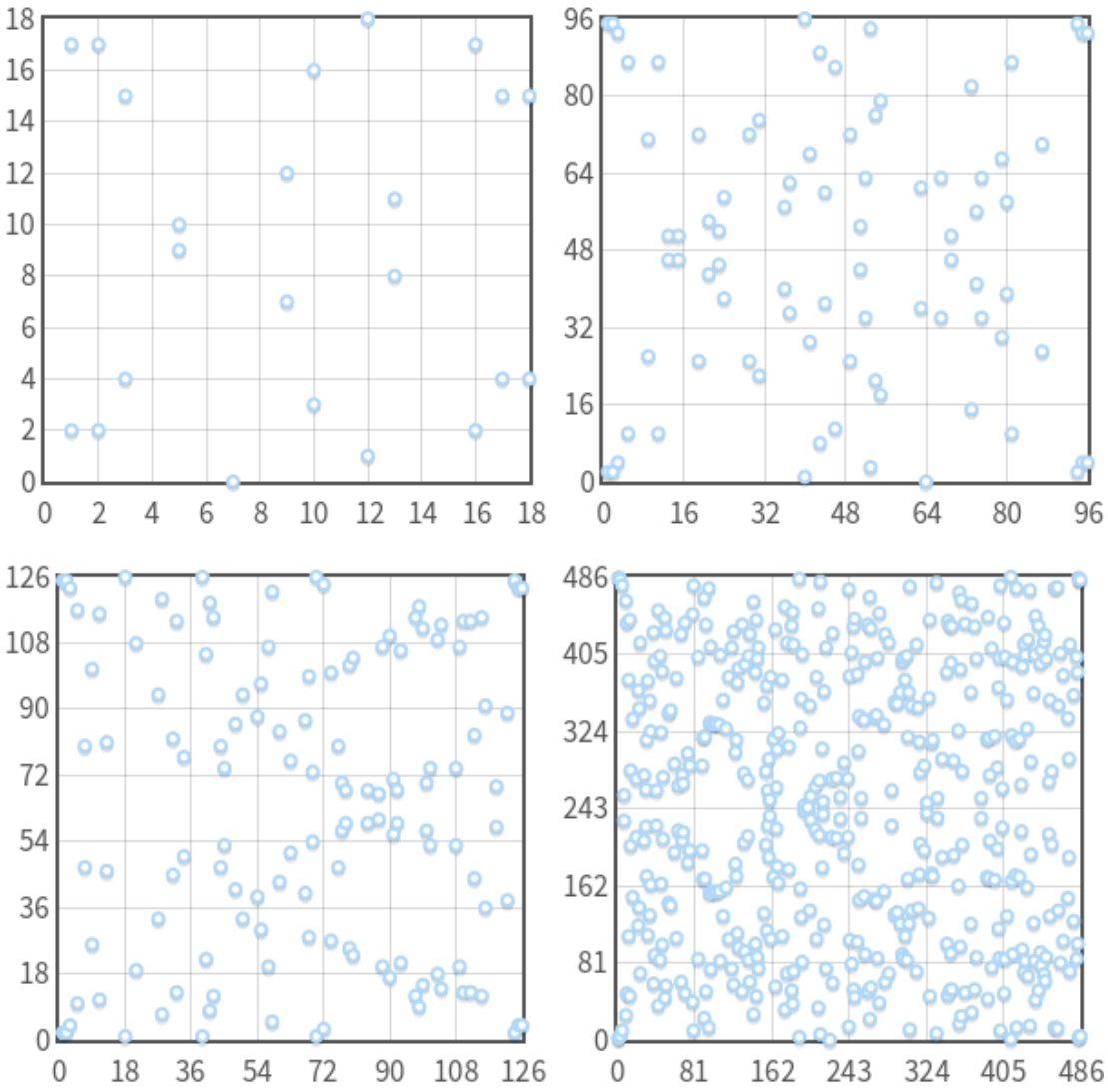


Figura 3.3: Curva $y^2 = x^3 - 7x + 10$ módulo p sobre $\mathbb{F}_{19}$, $\mathbb{F}_{97}$, $\mathbb{F}_{127}$ y $\mathbb{F}_{486}$
Fuente: andrea.corbellini.name

3.3 Aritmética de curva elíptica y ley de grupo

Si tenemos una curva elíptica C en el plano proyectivo $\mathbb{P}^2$ (esto es, el conjunto de puntos del plano junto con los puntos del infinito) y dada una recta L también de dicho plano, entonces la intersección de la curva y la recta son exactamente 3 puntos (la curva tiene grado 3, los puntos pueden no ser diferentes).

Definición:

Con esta propiedad en mente, se define la operación suma para dos puntos de la curva elíptica; es decir, dados P,$Q \in C$, la recta que pasa por P y Q nos definirá un nuevo punto R. Haciendo el simétrico de R con respecto al eje x tenemos R', que entendemos por P+Q. Dicho de otra manera: P+Q+R=0. Veamos los cuatro casos posibles de dicha operación:

Caso 1:

Cuando P y Q son distintos y R es un punto cualquier de la curva. Figura 3.4.

En esta situación es cuando mejor se ilustra la suma de dos puntos.

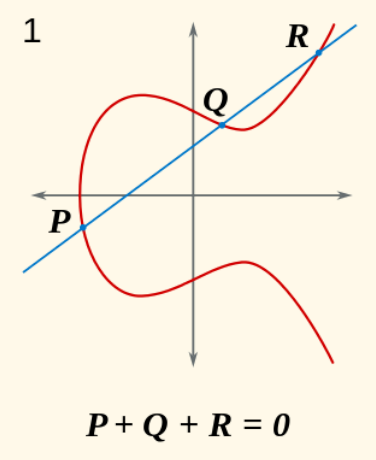


Figura 3.4: Caso 1
Fuente: Wikipedia

Caso 2:

P y Q son distintos y la recta es tangente en Q. Figura 3.5.

Cuando la recta es tangente en un punto, este mismo punto es el resultado de la suma.

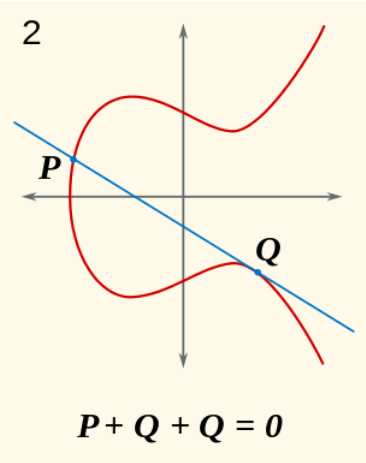


Figura 3.5: Caso 2
Fuente: Wikipedia

Caso 3:

P y Q son distintos y el tercer punto es el 0. Figura 3.6.

La recta pasa por P y Q, no es tangente en estos puntos y no corta a la recta. El punto resultante es el 0.

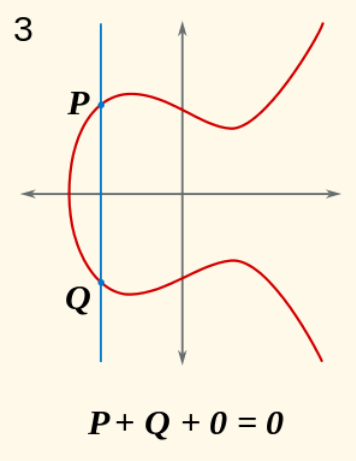


Figura 3.6: Caso 3
Fuente: Wikipedia

Caso 4:

Finalmente donde P y Q son iguales (recta tangente en P) y el tercer punto es el 0. Figura 3.7.

El último caso degenerado, donde solo hay un punto a considerar y la recta es tangente a él. El resultado es el punto 0.

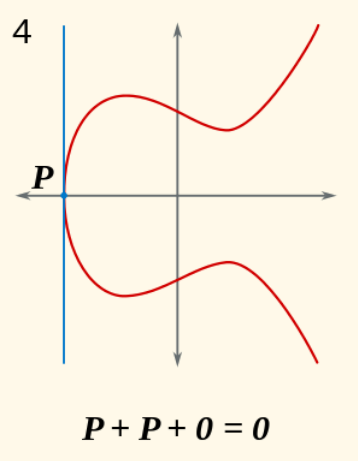


Figura 3.7: Caso 4
Fuente: Wikipedia

Definición:

En el caso general $char(K) \neq 2,3$ se define aritmeticamente para una curva C con ecuación $y^2 = x^3 + Ax + B$ y punto base (en la recta del infinito) $0 = (0:1:0)$ y siendo el simétrico de *P=(x,y)* con respecto al eje y, *-P=(x,-y)*, la suma $P + Q = (x_1, y_1) + (x_2, y_2) = (x_3, y_3)$ si los puntos son distintos con la ecuación 3.1, y si son iguales con la ecuación 3.2:

$$P \neq Q \begin{cases} x_3 = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)^2 - x_1 - x_2 \\ y_3 = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)(x_1 - x_3) - y_1 \end{cases} \tag{3.1}$$

$$P = Q \begin{cases} x_3 = \left(\frac{3x_1^2 + A}{2y_1}\right)^2 - x_1 - x_2 \\ y_3 = \left(\frac{3x_1^2 + A}{2y_1}\right)(x_1 - x_3) - y_1 \end{cases} \tag{3.2}$$

Queda definida de esta manera una operación suma sobre la curva, pudiendose demostrar su asociatividad. Por tanto, dada una curva C, un punto del infinito, un opuesto o simétrico, y una demostrable asociatividad, todo sobre un cuerpo finito, tenemos definida la ley de grupo en C.

Es posible además, que dada una curva sobre un cuerpo $K = \mathbb{F}^q$ con $q = p^m$, p primo, mediante el Teorema de Hasse, acotar el número de puntos racionales N de la curva mediante la ecuación $|N - (1 + q)| \leq 2\sqrt{q}$ (ver [29]).

3.4 El problema del logaritmo elíptico sobre curva elíptica: ECDLP

Dada entonces una curva sobre un cuerpo finito, y una operación tal que los puntos de la curva formen un grupo, nos podemos plantear el problema de, dado un punto base P, y un punto Q, encontrar en caso de existir, un $s \in \mathbb{Z}$, tal que Q=sP.

Este problema es el llamado Del inglés, Elliptic curve discrete logarithm problem (ECDLP) y es donde radica la fortaleza de los algoritmos que sustentan la criptografía con curva elíptica. Su complejidad algorítmica permite que, con claves relativamente pequeñas, esto es, el tamaño en bits de la representación del punto elegido, el cálculo para romperlas es elevado.

Como referencia del coste computacional necesario en función del tamaño , podemos recuperar el cuadro de la introducción 1.1. Por ejemplo, para un punto de una curva sobre un cuerpo, cuya representación nos ocupa 160 bits, equivale a una clave RSA de 1024 bits.

3.4.1 Solución del ECDLP

Existen algunos métodos conocidos para solucionar el problema del logaritmo elíptico. Uno de ellos, Silver-Pohlig-Hellman, dado un cuerpo $\mathbb{F}$, la solución tiene una complejidad del orden de $\sqrt{N_1}$, con $p-1 = N_1 N_2 ... N_r$ la factorización de (p-1) en r primos y N_1 siendo el mayor de ellos.

El mejor método conocido hasta el momento es el llamado ρ de Pollard, que necesita $\frac{\sqrt{\pi n}}{2}$ pasos (siendo n el orden del punto elegido) y se puede paralelizar dividiendo el número de pasos por el número de procesadores, siendo igualmente un problema intratable en general [29].

Una buena referencia para el estado actual de los métodos para el problema ECDLP, algunos de los cuales son muy recientes, se puede encontrar en la página de ***elliptic-news*** [9] (si bien el título incita a pensar en una página de referencia oficial, parece que es mantenida a título personal por el autor).

CAPÍTULO 4

CRIPTOGRAFÍA DE CURVA ELÍPTICA

Un sistema que puede desafíar y hacer la competencia al RSA y DSA, como hemos comentado, es la criptografía de curva elíptica. Actualmente se han hecho esfuerzos para la normalización de la criptografía de curva elíptica y aparece descrita en el estándar del Institute of Electrical and Electronics Engineers (IEEE) *P1363 Standard for Public-Key Cryptography* [19]. Esta especificación incluye protocolos de *key agreement*, de firma digital y esquemas de cifrado utilizando varios enfoques matemáticos: factorización de enteros, logaritmo discreto y, también, el logaritmo discreto sobre curva elíptica.

En los protocolos donde se utiliza la ECC, la seguridad se basa en la implementación del problema del logaritmo discreto, donde el cálculo de un elemento de curva elíptica con respecto a un punto de base conocido, no es factible. Este es el problema ya comentado en el apartado anterior: el problema del logaritmo discreto sobre curva elíptica o ECDLP.

Por tanto, la seguridad de estos esquemas criptográficos depende de la capacidad para calcular una multiplicación escalar de un punto y la incapacidad para calcular su inversa. El tamaño de la curva elíptica determina la dificultad del problema. Existen numerosos textos, por ejemplo [21] donde se explica con detalle los fundamentos de la ECC, no obstante una descripción matemática completa de la ECC queda fuera del objetivo de este artículo.

4.1 Las curvas

Un factor a tener en cuenta antes de nada, es la elección de la propia curva. El diseño de esta tiene un gran impacto en el rendimiento, y cada curva se diseña con un propósito determinado. Si bien es tecnicamente posible elegir aleatoriamente una curva por el desarrollador de un protocolo e implementar sobre ella los algoritmos, hay una serie de recomendaciones y estándares que intentan garantizar la eficiencia y seguridad de las curvas.

En cuanto al número de operaciones que se necesitan realizar para operar sobre una curva frente a otra, es relativamente sencillo el calcularlo; se trata de contar el número de pasos para sumar dos puntos, o, uno dado, duplicarlo o triplicarlo. Se puede ver una lista extensa de dicho cálculo para muchas familias de curvas [36]. También es interesante la comparativa de dos familias distintas de curvas en su desempeño en TLS, realizado por ARM Holdings plc. (ARM) [41].

Sin embargo, para verificar la seguridad de una curva, es bastante más complicado. Gran parte de la seguridad proviene de la elección realmente aleatoria o muy restringida de los parámetros que la definen, y no tiene que extrañar que para una definición dada, se pueda tener serias dudas sobre la existencia de puertas traseras. Citar por ejemplo el caso **Dual EC DRBG (Dual Elliptic Curve Deterministic Random Bit Generator)**, un algoritmo estandarizado por la NIST en 2007 para la generación aleatoria de los parámetros de las curvas, y que rapidamente fue sospechoso de incorporar una puerta trasera controlada por la National Security Agency (NSA), y supuestamente confirmado con los documentos del caso Snowden [37].

Veamos las curvas más usadas actualmente en la ECC:

4.1.1 Curve25519

Esta es una de las curvas más utilizadas y eficientes, considerada muy segura, es un estándar de facto. Fue propuesta por Daniel J. Bernstein en torno al 2005 y se popularizó a partir del 2013 por las sospechas sobre la NSA en su famila de curvas utilizadas hasta el momento. Pertenece a la familia llamada curvas de Montgomery, y tiene por ecuación $y^2 = x^3 + 486662x^2 + x$, sobre el cuerpo finito definido por el primo $2^{255} - 19$, y toma como punto base $x = 9$ [38]. La lista, seguramente parcial, de protocolos y software que la utilizan, es kilométrica [39].

4.1.2 Curvas NIST

El NIST (American National Standards Institute) publicó en 2009 el estándar FIPS PUB 186-3 [40] para definir el DSS (Digital Signature Standard), que proponía el uso de ECDSA sobre un conjunto de 15 curvas tal y como se define en el estándar X9.62 del año 1999. Hay tres tipos de curvas definidas:

Curvas NIST sobre cuerpos primos

Son curvas pseudoaleatorias de la forma $y^2 = x^3 - 3x + b$ sobre un cuerpo primo, y tenemos la *P-192, P-224, P-256, P-384* y *P-521*. La *P* se refiere al ser del cuerpo primo y el número que acompaña, al tamaño de la representación binaria del número primo usado.

Curvas NIST sobre cuerpos binarios

Son curvas pseudoaleatorias de la forma $y^2 + xy = x^3 + x^2 + b$, teniendo la *B-163, B-233, B-283, B-409*, y *B-571*.

Curvas NIST tipo Koblitz

Son de la forma forma $y^2 + xy = x^3 + ax^2 + 1$ con a igual *a* 0 o 1. Tienen la característica estar definidas en un subcuerpo finito pequeño elegido de un cuerpo finito mayor (una extensión) [69], permitiendo mediante endomorfimo un cálculo más rápido, siendo además la construcción determinista. Son las llamadas *K-163, K-233, K-283, K-409* y *K-571*.

4.1.3 Curvas Brainpool

Patrocinado por un grupo de empresas e instituciones alemanas, y amparado por la Internet Engineering Task Force (IETF) en 2005 en el documento *RFC 5639* [43], surgen como respuesta a la imposibilidad de verificar las establecidas por los otros estándares, la búsqueda de una implementación eficaz y la prevención a algunos ataques ocurridos. Tienen la forma $y^2 = x^3 + A * x + B \ mod \ p$ con $4A^3 + 27B^2 \ mod \ p$ diferente de 0, sobre el cuerpo *GF(p)* y *p* primo. Se tiene para primos con representación de 160, 192, 224, 256, 320, 384 y 512 bits.

4.1.4 Curvas CECG

Recomendadas por Standards for Efficient Cryptography Group (SECG), un consorcio de empresas, entre ellas Certicom, en su documento *SEC 2* [44] en torno al año 2000, con la idea de facilitar la estandarización y la interoperabilidad en un gran rango de plataformas. Son curvas definidas sobre un cuerpo $\mathbb{F}_p$ con *p* primo de un tamaño en bits de 192, 224, 256, 384 y 521, para permitir implementaciones eficientes.

La lista es *secp112r1, secp112r2, secp128r1, secp128r2, secp160k1, secp160r1, secp160r2, secp192k1, secp224k1, secp224r1, secp256k1, secp384r1* y *secp521r1*. La *r* hace referencia a la elección pseudoaleatoria del número primo, y la *k* a los parámetros asociados a las curvas Koblitz.

4.1.5 Curvas NUMS, ningún As bajo la manga

Nothing Up My Sleeve (NUMS), o en castellano, *Ningún As bajo la Manga*. Es un conjunto de curvas diseñadas con propósito de prescindir de las curvas NIST y buscar la seguridad y obtimización para la ECC. Construidas de forma sencilla, tienen poca

capacidad para modificar sus parámetros y por tanto una posible inclusión de una debilidad premeditada. Fueron publicadas bajo el respaldo de Microsoft y avaladas por la IETF [67].

Tres de las curvas se pueden expresar en la forma reducida de Weistrass $y^2 = x^3 - 3x + b$ sobre el cuerpo $\mathbb{F}_p$ con p primo. Tenemos la *numsp256d1* para $p = 2^{256} - 18$, *numsp384d1* para $p = 2^{384} - 317$ y *numsp512d2* si $p = 2^{512} - 569$. Las otras tres se representan en la forma Edwards $x^2 + y^2 = 1 + dx^2y^2$ sobre el cuerpo $\mathbb{F}_p$, teniendo *numsp256t1* si $p = 2^{256} - 189$, *numsp384t1* si $p = 2^{384} - 317$ y *numsp512t1* si $p = 2^{512} - 569$.

4.2 Los protocolos

Veamos los protocolos más utilizados en la ECC, que son el Elliptic curve Diffie Hellman (ECDH) y el Elliptic Curve Digital Signature Algorithm (ECDSA). Si bien existen otros protocolos sobre curva elíptica, por ejemplo una variante del RSA bajo el esquema llamado KMOV, no parece que ningún software revisado lo utilice.

4.2.1 ECDH

Este protocolo para el intercambio de una clave secreta consiste en DH adaptado al caso de curvas elípticas. Tal y como se persigue con esta adaptación, es la reducción del tamaño de claves gracias al problema del logaritmo elíptico.

Esquema de ECDH

Veamos su funcionamiento, hay que tener en cuenta que la implementación efectiva lleva otras medidas de seguridad como control de tiempos y demás:

Acuerdo de los parámetros:

Dada una curva sobre un cuerpo $\mathbb{Z}$ y un punto P de la curva acordado que tenga orden primo grande. Tanto Alice como Bob eligen un número en $\mathbb{Z}$ que será secreto y se comunican el valor de multiplicar ese punto P por el número elegido.

Cálculo de la clave:

Alicie calcula $n_A P$ y se lo manda a Bob.
Bob calcula $n_B P$ y se lo manda a Alice.
No importa si Mallory intercepta la comunicación.

Ya tienen una clave simétrica compartida de forma segura:

La clave secreta que han podido establecer es simplemente $K = (n_A n_B)P$, que solo ellos pueden calcular.

Ejemplo de ECDH

Ejemplo con valores manejables, sin fortaleza alguna.

Acuerdo de los parámetros:

Sea por ejemplo la curva $E : y^2 = x^3 + 5x + 7$ sobre $\mathbb{Z}_{113}$.

Esta curva sobre este cuerpo tiene 127 puntos.

Sea *P=(16,51)* cuyo orden es 127.

Cálculo de la clave:

Alice elige 98, manda a Bob 98(16,51)=(24,74).

Bob elige 101, manda a Alice 101(16,51)=(3,7).

Alice calcula 98(3,7)=(5,48).

Bob calcula 101(24,74)=(5,48).

Ya tienen una clave simétrica compartida de forma segura:

Su clave secreta será (5,48).
Ambos ya pueden compartir un secreto que se traduce normalmente en una clave simétrica.

Igualmente que su versión no elíptica, este protocolo se puede extender a un número indifinido de usuarios.

4.2.2 ECDSA

Como en el caso de DSA pero adaptado a ECC para reducir el tamaño de la clave, el ECDSA adoptado en el estandar DSS.

Esquema de ECDSA

Veamos el esquema básico del protocolo:

Generación de claves:

Dada una curva elíptica y un cuerpo $\mathbb{Z}$, se toma un punto de la curva P con orden primo n. Tenemos un mensaje m para firmarlo.

Tanto Alice como Bob eligen un número entre 1 y n-1, sean n_A y n_B respectivamente sus claves privadas.

Firma:

Alice elige un número k entre 1 y n-1.
Calcula un Hash del mensaje m, esto es, $H(m)$.
Calcula $kP = (x_1, y_1)$ y $r = x_1 \ (mod\ n) \neq 0$.
Calcula $k^{-1} \ (mod\ n)$.
Calcula $s = k^{-1}(H(m) + n_A r) \ (mod\ n) \neq 0$.
La firma será *(r,s)*.

Verificación:

Ahora Bob recibe el mensaje m y la firma *(r,s)*. Procede de la siguiente manera:
Obtiene la clave pública de Alice, $P_A = n_A P$.
Verifica que $1 < r, s < n-1$.
Calcula $w = s^{-1}\ (mod\ n)$ y *H(m)*.
Calcula $u_1 = H(m)\,w\ (mod\ n)$ y $u_2 = r\,w\ (mod\ n)$.
Calcula $(x_0, y_0) = u_1 P + u_2 P_A$ y $v = x_0\ (mod\ n)$.
La firma es válida solamente si$v$=r.

4.3 Estándares

Podemos citar un serie de estándares definidos por los principales organismos de estandarización, en orden cronológico:

Los primeros estándares en usar la ECC fueron publicadas por el American National Standards Institute (ANSI), el llamado *X9.62* que adoptaba la ECDSA, sobre el año 1999. Posteriormente el estándar *X9.63* en el año 2000 ya adopta el ECDH y algunos otros protocolos de ECC.

La the International Organization for Standardization (ISO) publica una descripción general de la ECC, *ISO/IEC 14888-:1998* y posteriormente introduce el ECDSA y el ECDH entre otros algoritmos.

Los laboratorios RSA publican los documentos *PKCS11* y *PKCS13* relativos a la ECC.

El NIST publica su estándar *FIPS 186-2* para la DSS en el año 2000.

El IEEE en el año 2000 introdujo la ECC en su documento *P1363* y en el borrador posterior *P1363A*.

La NSA define en su documento *Suite B* [17] del año 2005 los algoritmos criptográficos recomendados. En concreto, y por lo que hace referencia a la criptografía de curva elíptica, se especifican los siguientes sistemas: ECDH para el intercambio de claves y ECDSA para la firma digital. La NSA aconseja el uso de la ECC para proteger la información clasificada hasta el nivel alto secreto (*top secret*) con claves de 384 bits. Sin embargo, en agosto de 2015, la NSA anunció que tiene prevista la sustitución del *Suite B* por un nuevo *suite* de cifrado motivado por ataques a la *ECC* que se han descrito utilizado computación cuántica [18].

El SECG en el 2009 publica el *SEC 1*, a lo que le seguiran el *SEC 2, SEC 3* y *SEC 4*.

Actualmente se siguen publicando nuevos estándares y pone de manifiesto el vigor de esta tecnología. Para una descripción más detallada se recomienda el texto [29]

CAPÍTULO 5

LA CRIPTOGRAFÍA DE CURVA ELÍPTICA ACTUAL

En los siguientes apartados se revisan y comparan distintas aplicaciones que usan ECC, desde lo más general, como el uso en TLS o correo electrónico hasta aplicaciones particulares para plataformas concretas tipo Arduino. Como referencia, la comparativa ha sido realizada sobre un ordenador personal de gama media, un AMD de 6xPhenom 1035T con sistema Linux 3.16.0-4-amd64. También se disponía de un *Arduino Uno*, pero su utilidad resultó muy limitada. También se intentó realizar pruebas con un Smartphone con Android, pero resultó inútil.

Un estudio más detallado y sistemático del desempeño de las librerías de ECC, en la línea del proyecto eBacs (ECRYPT Benchmarking of Cryptographic Systems) [22], sería deseable. Sin embargo la complejidad y metodología de dicho proyecto hace complicado enmarcarlo con este trabajo, o al menos para el autor.

5.1 TLS. OpenSSL

En SSL/TLS es donde probablemente más operaciones se hagan con criptografía asimétrica en general y ECC en particular. Al ser un protocolo de transporte, todas las aplicaciones sobre las que se aplica quedan beneficiadas de la seguridad criptográfica.

Tradicionalmente este conjunto de protocolos (SSH) usaba criptografía asimétrica no ECC para la autentificación e intercambio de claves simétricas, esto es, RSA, DH, DSA y Menezes Qu Vanstone (MQV). A partir de la especificación TLS 1.1 ya se tiene la capacidad de usar ECC: ECDH, normalmente en modo efímero, ECDH(E), ECDSA y en menor medida Elliptic Curve Menezes Qu Vanstone (ECMQV). El ECDH es particularmente poco costoso en térmios de cálculo [42].

Se probó el paquete OpenSSL, que permite implementar el protocolo TLS, trabajando sobre numerosas curvas estandarizadas (*openssl ecparam -list_curves*). Dispone de unas rutinas propias para rendimientos (*openssl speed*) [45]. La instalación y ejecución de esta aplicación en un entorno Linux es sencilla y robusta.

En la tabla A.1 del apéndice se pueden ver los tiempos de ejecución para el RSA, en la tabla A.2 para el ECDSA y en la tabla A.3 para el protocolo ECDH, todos sobre diferentes tamaños de claves y curvas.

En la figura 5.1 podemos apreciar los tiempos medios agrupados por niveles equivalentes de seguridad (1024 bits de clave en RSA corresponden a 160 bits en ECC, por ejemplo. Ver tabla 1.1) de una operación completa (cifrado-descifrado o firmar-verificar). Notar que para niveles de seguridad superiores a los 4096 bits para RSA y DSA ya no se dispone de este tamaño de clave en esta implementación, mientras que para ECDSA y ECDH disponemos de curvas que superan ese nivel de seguridad.

La elección de la curva es también decisiva para el rendimiento: Mientras que las de la familia nistp son rápidas, la familia nistk es muy irregular y la familia nistb las más lentas. En las gráficas 5.2 y 5.3 podemos ver el tiempo de ECDH y ECDSA para las diferentes curvas. En la tabla A.2 y la tabla A.3 se pueden consultar con detalles los tiempos de las distintas familias. La aplicación permite configurar la curva que se usará.

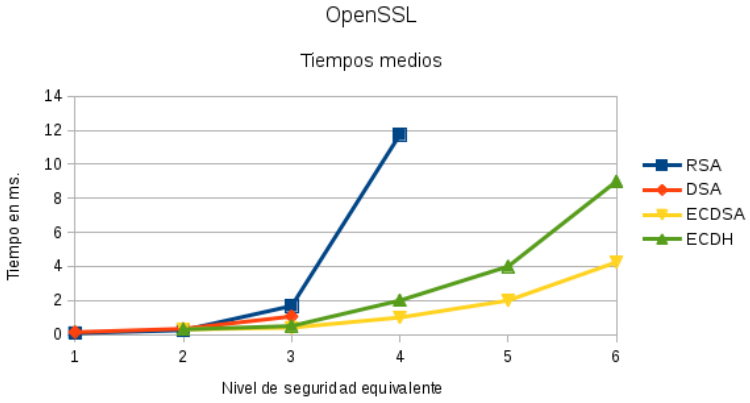


Figura 5.1: Tiempos medios RSA, DSA, ECDSA y ECDH en OpenSSL

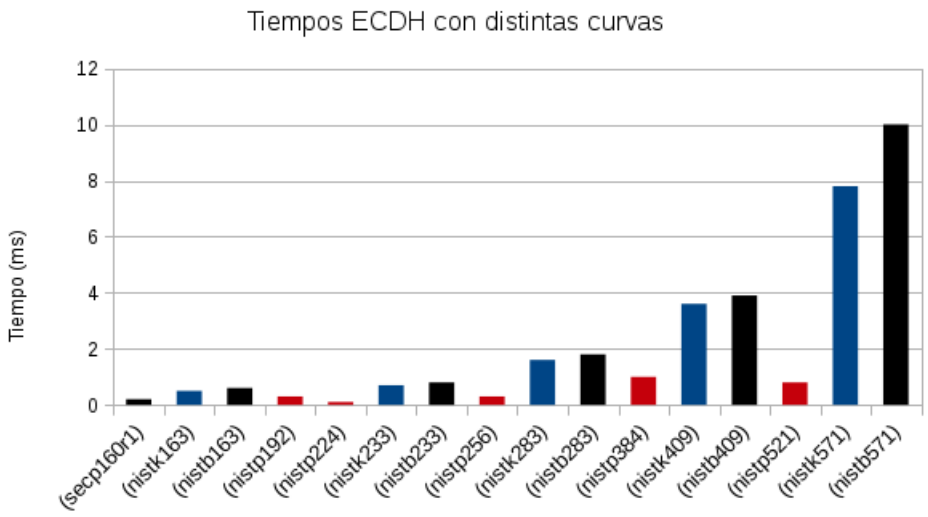


Figura 5.2: Tiempos para diferentes curvas con ECDH en OpenSSL

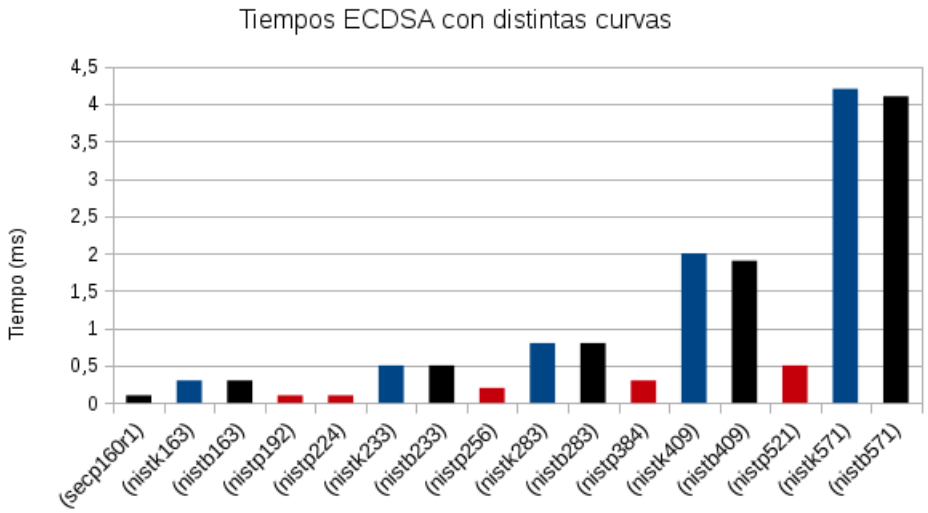


Figura 5.3: Tiempos para diferentes curvas con ECDSA en OpenSSL

A la vista de los resultados podemos ver la necesidad de configurar TLS con ECC para ciertos niveles razonables de seguridad. La familia de curvas *nistp* parece la opción más práctica por su velocidad para uso de propósito general. Con todo, si se desea un grado de confianza mayor, es necesario el uso de las *nistpb* o *nistpk*.

5.2 ZRTP, ECC en tiempo real

Las comunicaciones en tiempo real es otro campo donde la ECC se está introduciendo. Si bien con los apartados anteriores ya se incluye de alguna manera el cifrado de paquetes en redes, por ejemplo TLS, hay aplicaciones que necesitan directamente el cifrado a otro nivel o con un ancho de banda mucho menor.

El cifrado en tiempo real se emplea cuando se transmiten comunicaciones en directo, como voz y vídeo en una videoconferencia mediante Voice over IP (VoIP) o en comunicaciones de radio UHF/VHF. En general hay protocolos que se encargan de esta tarea, como el Real time Transport Protocol (RTP), y cuenta con su versión que utiliza cifrado, Secure Real time Transport Protocol (SRTP), que utiliza cifrado asimétrico tradicional para el intercambio de claves y firma, y simétrico para cifrar el flujo de datos. En nuestro caso, nos centraremos en uno similar que utiliza ECC, el Zimmermann Real time Transport Protocol (ZRTP).

El ZRTP, desarrollado principalmente por Zimmermann (creador de *PGP*) y especificado por la IETF en su documento *RFC6189[81]* en 2011, utiliza DH para el cálculo de la clave compartida, pero lo puede hacer de cuatro maneras, *DH-3072* y*DH-2048*, y otras dos formas con ECC, ECDH *P-256* y ECDH *P-384*, que como hemos visto, es DH sobre las curvas NIST. Para la firma utiliza ECDSA sobre las mismas curvas.

En la web Zfone [82], que mantienen los desarrolladores del protocolo, se pueden encontrar plugins para difentes clientes de softVoIP, así como una plataforma para su implementación en cualquier aplicación. Por ejemplo, en una rápida búsqueda en Google, vemos a un desarrollador que implementa ZRTP para un hardphone [83] mediante una *Raspberry Pi*, o más en general, podemos ver que *FreeSWITCH* [84], un software libre que permite la integración de casi cualquier tipo de mensajería de datos (Radio HAM, móbiles, IoT, etc.), implementa ZRTP [85].

5.3 PHP, ECC en la web

Otra aplicación interesante es *PHPECC*, o sus versiones más modernas PurePHP [73], una implementación de la ECC para Hypertext Preprocessor (PHP). Esto permite que, a pesar de no usarse TLS, se pueda establecer una comunicación segura en la www. La librería está muy bien documentada e implementa DH y DSA, tanto en las curvas SECG como las NIST.

El desarrollador de la librería ha publicado una comparativa usando *GMP* y *bcmath* [74](librerias de cálculo para PHP), la primera librería utiliza enteros de longitud arbitraria usando enteros cuando es posible, y la segunda opera también con enteros de longitud arbitraria pero siempre como strings. Con estas librerías y teniendo en cuenta que PHP es interpretado, no es de extrañar los tiempos que muestra sobre un ordenador de 2,4 GHz ejecutando DH: 5,8 segundos con *GMP* y 1791 segundos para *bcmath*.

5.4 MSR ECCLib, ningún As bajo la manga

Una librería actual que aporta ECC para un conjunto nuevo de curvas consideradas altamente seguras es *MSR_ECClib* [65], que cuenta con el respaldo de Microsoft. La librería cuenta con su rutina de prueba (./crypto_tests), que funcionó perfectamente. Los tiempos obtenidos quedan reflejados en la gráfica 5.4 y detallados en la tabla 5.1. Son tiempos muy similares a los vistos en TLS con OpenSSL (que usa otras curvas, *nist*), contando con la ventaja de una construcción de curva más sólida.

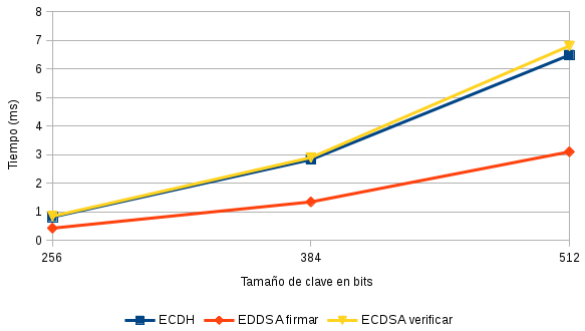


Figura 5.4: Gráfica MSR ECC Lib

Cuadro 5.1: Benchmark MSR_ECClib

Sistema	bits	curva	c/f (ms)	d/v (ms)
ECDH(E)	256	numsp256d1	0.881	0.881
ECDSA	256	numsp256d1	0.445	0.901
ECDH(E)	256	numsp256t1	0.756	0.756
ECDSA	256	numsp256t1	0.405	0.786
ECDH(E)	384	numsp384d1	3.052	3.052
ECDSA	384	numsp384d1	1.398	3.054
ECDH(E)	384	numsp384t1	2.600	2.600
ECDSA	384	numsp384t1	1.292	2.743
ECDH(E)	512	numsp512d1	6.974	6.974
ECDSA	512	numsp512d1	3.189	7.207
ECDH(E)	512	numsp512t1	5.998	5.998
ECDSA	512	numsp512t1	3.011	6.408

5.5 GnuPG

Otro uso importante de la ECC lo encontramos en los protocolos para correo electrónico y documentos en general. En este área se ha realizado un test con *GnuPG*, una alternativa de libre distribución del famoso *PGP* que sigue el estándar *OpenPGP*, donde en su última versión 2.1.11 (del 2016) ya utiliza ECC [46].

Su instalación en un entorno Linux es sencilla, al menos hasta la versión 1.4, pero la última versión que incorpora ECC es algo más complicada en la versión estable de Debian, requiriendo algo de paciencia. Con todo, el trabajo vale la pena cuando al final de resolver todas las dependencias de la última versión, al generar las claves en modo experto, tenemos, aparte de las opciones tradicionales RSA, ElGamal, DSA, y las opciones ECC, esto es, ECDH y ECDSA. También nos permite elegir entre diferentes curvas. Notar que la aplicación usa un par de parejas de claves para firmar y cifrar, RSA-RSA y ECDH-ECDSA.

Los tiempos obtenidos en estas pruebas, los tiempos son relativos al archivo a cifrar, en este caso de 1MB de tamaño. El número de pruebas realizadas es grande pero insuficiente para establecer conclusiones. Después de realizar bastantes pruebas creando parejas de claves y usándolas con diferentes sistemas, tamaños de claves y curvas, no se obtienen datos de RSA coherentes (más bits, menos tiempo), ver tabla A.4) para apreciar la incoherencia y tiempos totales. Sin embargo, para la ECC parece consistente. Queda resumido en la gráfica 5.5. A pesar de las limitaciones de la comparativa, el tiempo de ECC para un nivel de seguiridad muy por encima de RSA es muy moderado.

Lo más destacado a nivel de usuario, es la enorme cantidad de tiempo real, decenas de minutos, que no de ciclos de reloj, necesarios para generar las claves RSA grandes, y la rapidez en tiempo real, pocos segudos, para la generación de claves de ECC (imaginamos que por motivos de la implementación para la obtención de valores aleatorios). Con todo, esto es algo sin mucha importacia, pues la creación de claves en esta aplicación es una operación normalmente *off-line* y poco frecuente.

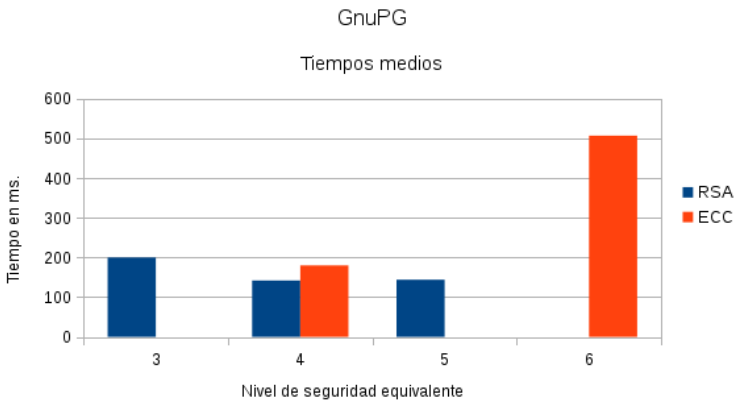


Figura 5.5: Tiempos medios de RSA y ECC

5.6 Bitcoin

Bitcoin se basa en el uso de la ECC. Toda transacción de Bitcoin actualmente se basa en una firma digital realizada con ECDSA sobre la curva *secp256k1* [56]. Esta curva pertenece a la familia de curvas de la SECG en su recomendación SEC 2. La *secp256k1* es del tipo Koblitz. Sin embargo, parece que no todo el mundo piensa bien sobre esta curva [70].

La clave pública de las firmas realizadas con ECDSA de BitCoin, que en este contexto se pueden entender como direcciones, mediante una función Hash sobre los 256 bits de la firma, se consiguen direcciones de 160 bits, tamaño que permite incluirlas por ejemplo en códigos QR. Ver figura 5.6.

Como último apunte decir que este junio de 2016 encontré una noticia del cambio de esquema criptográfico de Bitcoin, dejando la ECC y pasando a usar el esquema Schnorr, apoyándose dicho esquema en el problema del logaritmo discreto pero sin curva elíptica [71]. Este esquema fue publicado en 2014 como borrador por la IETF [72].

Figura 5.6: QR Bitcoin

5.7 Android

Otra plataforma que cada vez tiene más importancia y puede sacar mucho partido de la ECC, es Android. Por supuesto al usarse TLS el uso de la ECC está implícito. Hay otras librerías conocidas que también incorporan ECC a esta plataforma, como las de Java [47] o Bouncy Castle [48].

Esta última, tiene un reempaquetado específico para Android, llamado Spongy Castle [49]. Sin embargo, y por los comentarios de desarrolladores, parece ser que la implementación es muy simple para ser eficiente [51]. Hemos podido consultar una comparativa de ECDH con Spongy Castle sobre Android [50], en un emulador de un modelo de teléfono común, realizado en el 2011: los tiempos que resultan parecen altos, del orden de varios segundos por operación de cifrado/firmado y descifrado/verificación.

En Java, nos podemos fijar en el *toolkit FlexiProvider* [75] (módulo ECProvider), de la cual hemos encontrado una comparativa sobre varios móviles y una tablet de ECC y criptografía simétrica [76]. Hay que notar al comparar los tiempos que muchos de estos dispositivos móbiles ya incorporan un hardware específico para criptografía simétrica. Los resultados parecen tiempos relativamente altos, del orden de medio segundo para la operación.

También se pueden encontrar, por supuesto, muchos paquetes pequeños y/o experimentales específicos para Android, como pueden ser *Keyczar* y *AeroGear* [52], que incorporan ECC.

Una de las implementaciones que se suponen más eficientes para Android en particular y para diversos sistemas en general, es *NaCl* (llamada "Sal") [53], una librería mixta de cifrado, con uso de ECC, multiplataforma (la incorpora por ejemplo el iPhone y OpenDNS), libre y abierta. Su propósito según citan los autores es corregir fallos comunes de implementación de otras librerías [54], optimizando al máximo la aritmética modular empleada. Está adaptada para Android con la variante llamada *libsodium* [77]. Hemos visto desarrolladores que recomiendan esta opción por su calidad y seguridad [78]. Por ver algunos tiempos de referencia, podemos consultar la comparativa realizada por un desarrollador [79] sobre distintos navegadores con *js-nacl* [80] (otra variante de *NaCL* en Javascript) en la curva 25519, siendo un rendimiento muy comparable a OpenSSL.

El principal problema con Android, que limita una comparativa, es el no poder ejecutar sobre el propio teléfono las aplicaciones que uno desea: Dado que Android no otorga por defecto permisos totales del sistema, y los fabricantes dificultan el cambio de permisos, no se puede instalar y ejecutar programas que no estén en el catálogo del fabricante o GooglePlay. El proceso para tener permisos totales, o *rootearlo* es engorroso. La Electronic Frontier Foundation (EFF) se hace eco de la trampa [55].

5.8 Internet of Things

Las plataformas donde más partido se puede sacar de la ECC son las plataformas pequeñas o con limitaciones propias del medio, que existen y conviven actualmente multitud de ellas. En lo que se denomina Internet of Things (IoT). Abarca un número enorme y creciente de dispositivos, sensores y microordenadores, por ejemplo, con muy poca capacidad de cómputo, energía y transferencia. El paso de RSA hacia la ECC es patente y rápido [57]. Ya hay trabajos encaminados a optimizar la ECC en este conjunto heterogéneo [58]. Veamos algunas plataformas particulares.

5.8.1 RFID y Tarjetas inteligentes

Un ejemplo importante de la ECC en estas plataformas, son las etiquetas Radio Frequency IDentification (RFID) y las tarjetas inteligentes o Smart Cards. Con su propio hardware integrado se utiliza ECDSA con un tiempo de cálculo de orden de unos pocos segundos. En 2001 se publicó un informe, *EMV40*, para el uso de curva elíptica en las tarjetas [30]. Uno de los estudios más completos que hemos encontrado para la ECC en estos circuitos específicos es la tesis del autor Daniel Hein, de la universidad politécnica federal de Züric [59].

5.8.2 Microcontroladores

Como disponiamos de un Arduino Uno (procesador *AVR*)[32], se probó sobre él diferentes paquetes experimentales para 8 bits. Este tipo de dispositivo se caracteriza por su sencillez, facilidad de programación y popularidad.

NaCl para 8 bits

Lo más completo y actualizado para estas plataformas parece ser una adaptación de *NaCl* para procesadores AVR de 8 bits, muy bien documentada y estudiada [63]. Tiene la capacidad de trabajar sobre la curva *Curve25519*, que supone 128 bits de tamaño de clave. El paper publicado por los desarrolladores tiene un estudio matemático muy correcto del funcionamiento del procesador, y ofrecen una comparativa en términos de ciclos de reloj, que permite, dependiendo de la velocidad soportada, deducir tiempos reales.

ArduinoLibs

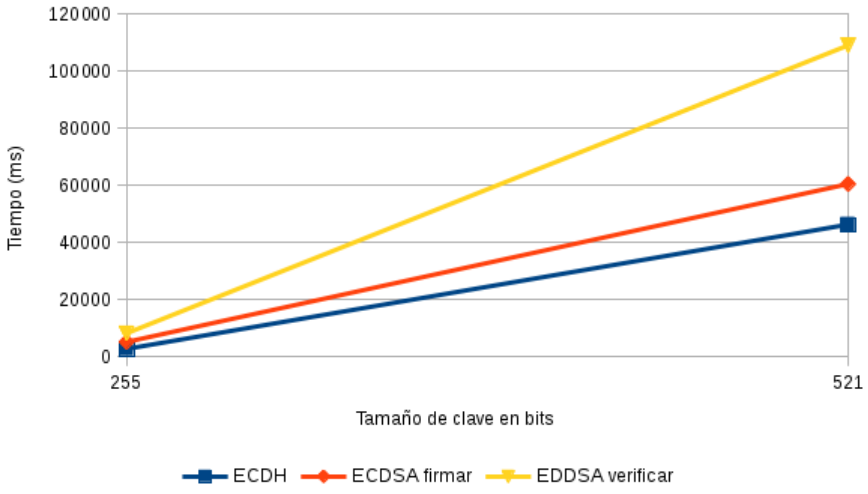
También fijarnos en el paquete criptográfico de *ArduinoLibs* [62], una alternativa muy bien desarrollada. En la propia web de *ArduinoLibs* hay un benchmarh para procesasdores AVR, en particular Arduino, con ECDH y ECDSA que operan sobre las curvas Curve25519, Ed25519 y P521. Intenté repetir los resultados con el Arduino Uno disponible, y aparecieron problemas tanto de compilación como de ejecución. Dada la imposibilidad, me puse en contacto con el desarrollador, Rhys Weatherley, muy conocido en el entorno *GNU* [33], y amablemente me indicó las limitaciones de memoria del Arduino Uno para ciertos test y la necesidad de compilar partes concretas por separado,

por lo que mi experimentación quedó paralizada. Reproduzco su comparativa. Ver tabla 5.2 y figura 5.7.

Operación clave pública	Tiempo(ms)	Comentario
Curve25519::eval()	2716	Evaluar curva
Curve25519::dh1()	2718	Primera mitad DH
Curve25519::dh2()	2717	Segunda mitad DH
Ed25519::sign()	5148	Generar firma DSA
Ed25519::verify()	8196	Verificar firma DSA
Ed25519::derivePublicKey()	5102	Generar clave pública de privada
P521::eval()	46290	Evaluar curva
P521::dh1()	46293	Primera mitad DH
P521::dh2()	46304	Segunda mitad DH
P521::sign()	60514	Generar firma DSA
P521::verify()	109078	Verificar firma DSA
P521::derivePublicKey()	46290	Generar clave pública de privada

Cuadro 5.2: Benchmark de Arduino Uno con ArduinoLibs

Figura 5.7: Gráfica Arduino con ECC



Vistos estos tiempos en los pequeños procesadores AVR, se comprende la importancia de la ECC para niveles aceptables de seguridad. Para este tipo de plataformas, ya existen circuitos integrados con un precio inferior al euro. Atmel por ejemplo fabrica un chip que ejecuta ECDH y ECDSA sobre la curva NIST *P256*. Su velocidad es de órdenes de tiempo menor que el mismo càlculo realizado por software [64].

Nano-ECC

Esta librería [60], que pude parcialmente compilar y ejecutar, no sin muchas dificultades, es una adaptación reducida de otra más general llamada *Micro-ECC* [61], para procesadores de 8,32 y 64 bits. Por desgracia parece algo parado el proyecto. Dado los

problemas de compilación algo aleatorios, me puse en contacto con el programador que la desarrolla y muy amablemente me indicó ciertos pasos para compilar, pero a pesar de todo, no pude hacer más que generar un solo tipo de clave, dando unos tiempos de generación de clave pública para ECDH sobre la curva *secp128r1* de algo menos de 4 segundos (3930ms de media).

5.8.3 ITS, Sistemas inteligentes de transporte

Uno de los campos que más promete desarrollarse en un futuro inmediato, es la automatización del transporte. Todo el conjunto de telecomunicaciones en este ámbito, lleva como condición necesaria la correcta y segura identificación del vehículo. Y es que además se realizan muchas y se necesitan rápido. Si pensamos por ejemplo en el intercambio de mensajes entre coche-coche o coche-base, se pueden realizar hasta unos 1.000 mensajes por minuto. Es por ello que se introduce la ECC en este campo. En particular, se está manejando actualmente el protocolo ECDSA sobre la curva NIST *P-256*, implementado en circuitos dedicados, Application Specific Integrated Circuit (ASIC), permitiendo unas 27.000 verificaciones por segundo, con muy poco consumo y latencia [66].

CAPÍTULO 6

CONCLUSIÓN

El estudio presentado, con todas sus limitaciones, creo que supone una muestra representativa del panorama actual de la ECC. Queda de manifiesto la aceptación y la necesidad de su uso. Después de las investigaciones en este campo y su desarrollo, todo indica que su fortaleza es comparable a los métodos tradicionales pero con claves mucho más pequeñas y manejables que permiten la ejecución de los cálculos criptográficos con una eficiencia mayor que la criptografía asimétrica convencional. La investigación de las curvas y los estudios constantes sobre los fundamentos matemáticos, hacen que la ECC esté en la vanguardia de la criptografía asimétrica. El número de publicaciones, estándares de curvas y protocolos que la implementan, confirma el interés. Actualmente el campo de batalla de este modelo criptográfico, es la elección de la curva, en aras de la seguridad y eficiencia de cálculo, y el diseño esmerado de las librerías para sacar el máximo partido del hardware.

Las librerías criptográficas más extendidas ya se decantan por la ECC. Cualquier desarrollador que quiera implementar en cualquier aplicación la criptografía, tiene a su disposición todo un arsenal de librerías de curva elíptica. Si bien es cierto que no se dispone de suficientes comparativas y referencias, y la matemática que la sustenta requiere un buen esfuerzo para aproximarse. Resulta imprescindible en pequeños dispositivos o en la IoT, donde algún o todos los factores críticos se presentan (capacidad de cálculo, memoria, ancho de banda y velocidad requerida para una transacción). Se ha podido ver que dependiendo de la implementación, bajo una misma plataforma, las diferencias de eficiencia son grandes, y que, usada bajo diferentes plataformas, abisales.

APÉNDICE A

TABLAS

A.1 OpenSSL

Cuadro A.1: Tiempos de RSA y DSA en OpenSSL

Sistema	bits	c/f (ms)	d/v (ms)
rsa	512	0.073	0.006
rsa	1024	0.235	0.015
rsa	2048	1.629	0.050
rsa	4096	11.557	0.190
dsa	512	0.069	0.069
dsa	1024	0.157	0.180
dsa	2048	0.483	0.578

Cuadro A.2: Tiempos de ECDSA en OpenSSL

Sistema	bits	curva	c/f (ms)	d/v (ms)
ecdsa	160	secp160r1	0.1	0.1
ecdsa	192	nistp192	0.1	0.1
ecdsa	224	nistp224	0.1	0.1
ecdsa	256	nistp256	0.2	0.2
ecdsa	384	nistp384	0.3	0.3
ecdsa	521	nistp521	0.5	0.5
ecdsa	163	nistk163	0.3	0.3
ecdsa	233	nistk233	0.5	0.5
ecdsa	283	nistk283	0.8	0.8
ecdsa	409	nistk409	2.0	2.0
ecdsa	571	nistk571	4.2	4.2
ecdsa	163	nistb163	0.3	0.3
ecdsa	233	nistb233	0.5	0.5
ecdsa	283	nistb283	0.8	0.8
ecdsa	409	nistb409	1.9	1.9
ecdsa	571	nistb571	4.1	4.1

Cuadro A.3: Tiempos de ECDH en OpenSSL

Sistema	bits	curva	c/f (ms)	d/v (ms)
ecdh	160	secp160r1	0.2	0.2
ecdh	192	nistp192	0.3	0.3
ecdh	224	nistp224	0.1	0.1
ecdh	256	nistp256	0.3	0.3
ecdh	384	nistp384	1.0	1.0
ecdh	521	nistp521	0.8	0.8
ecdh	163	nistk163	0.5	0.5
ecdh	233	nistk233	0.7	0.7
ecdh	283	nistk283	1.6	1.6
ecdh	409	nistk409	3.6	3.6
ecdh	571	nistk571	7.8	7.8
ecdh	163	nistb163	0.6	0.6
ecdh	233	nistb233	0.8	0.8
ecdh	283	nistb283	1.8	1.8
ecdh	409	nistb409	3.9	3.9
ecdh	571	nistb571	10.001	10.001

A.2 GnuPG

Cuadro A.4: Tiempos de RSA y ECC (ECDH + ECDSA)

Sistema	bits	curva	Crear	c/f (ms)	d/v (ms)
RSA	2048		48	80	72
RSA	3072		28	62	52
RSA	4096		32	60	52
DSA-Elgamal	2048		68	92	78
ECDH-ECDSA	255	ed25519	44	120	120
ECDH-ECDSA	256	NIST P-256	36	104	92
ECDH-ECDSA	256	Brainpool P-256r1	40	106	100
ECDH-ECDSA	256	secp256k1	38	104	100
ECDH-ECDSA	512	Brainpool P512r1	70	206	192
ECDH-ECDSA	512	NIST P-521	92	232	223

BIBLIOGRAFÍA

[1] Colaboradores de Wikipedia, RSA,"Wikipedia, La enciclopedia libre, disponible en la URL https://es.wikipedia.org/wiki/RSA 2.1

[2] Colaboradores de Wikipedia, "Diffie-Hellman,"Wikipedia, La enciclopedia libre, disponible en la URL https://es.wikipedia.org/wiki/Diffie-Hellman 2.2

[3] Colaboradores de Wikipedia, "Digital Signature Algorithm,"Wikipedia, La enciclopedia libre, disponible en la URL https://en.wikipedia.org/wiki/Digital_Signature_Algorithm 2.3

[4] Fortaleza de la criptografía, recomendaciones del NIST, disponible en la URL http://no1bc.com/es/support/articles/cryptographic-strength 1

[5] Certicom, ECC; disponible en la URL https://www.certicom.com/ecc 1

[6] Ataques a DES y módulos factorizados de RSA; disponible en la URL http://www.revistasic.com/revista40/pdf_40/SIC_40_agora.PDF 2.1

[7] Factorization of a 768-bit RSA modulus; disponible en la URL http://eprint.iacr.org/2010/006.pdf 2.1

[8] RSA Laboratories; disponible en la URL http://www.emc.com/emc-plus/rsa-labs/standards-initiatives/key-size.htm 2.1

[9] ellipticnews, ecdlp problem; disponible en la URL https://ellipticnews.wordpress.com/2016/04/07/ecdlp-in-less-than-square-root-time/ 3.4.1

[10] Elliptic Curve Cryptography in Practice; disponible en la URL http://eprint.iacr.org/2013/734.pdf 1

[11] W. Diffie, and M. Hellman, “New directions in cryptography,” en *IEEE Transactions on Information Theory*, vol. 22: 644-654, 1976. 2

[12] T. ElGamal, “A public key cryptosystem and a signature scheme based on discrete logarithms,” en *IEEE Transactions on Information Theory*, vol. 31: 469-472, 1985. 2

[13] V.S. Miller, “Use of elliptic curves in cryptography,” en *Advances in Cryptology CRYPTO 85*, vol. 218 of Lecture Notes in Computer Science, pages 417-426, Berlin, 1986. Springer-Verlag.

[14] N. Koblitz, "Elliptic curve cryptosystems," en *Mathematics of Computation*, vol. 48: 203-209, 1987.

[15] J.M. Pollard, "A monte carlo method for factorization," en *BIT Numerical Mathematics*, vol. 15, num. 3: 331-334, Berlin, 1975. Springer-Verlag.

[16] A. Jurisic, and A. Menezes, "Elliptic Curves and Cryptography," en *Dr. Dobb's Journal*, April 1997. 1

[17] "Suite B Cryptographic Module FIPS 140-2 Security Policy," disponible en la URL http://csrc.nist.gov/groups/STM/cmvp/documents/140-1/140sp/140sp2212.pdf. 4.3

[18] "Cryptography Today. National Security Agency (NSA)," disponible en la URL https://www.nsa.gov/ia/programs/suiteb_cryptography/index.shtml. 4.3

[19] "IEEE P1343 - Standard Specifications For Public-Key Cryptography," disponible en la URL http://grouper.ieee.org/groups/1363/. 4

[20] R. Rivest, A. Shamir, and L. Adleman, "A Method for Obtaining Digital Signatures and Public Key Cryptosystems," en *Communications of the ACM*, February 1978. 2

[21] , W. Stallings, "Cryptography and Network Security: Principles and Practice," Prentice Hall Press, 7th edition, Upper Saddle River, NJ, USA. 4

[22] eBACS: ECRYPT Benchmarking of Cryptographic Systems, disponible en la URL http://bench.cr.yp.to/primitives-encrypt.html 1, 5

[23] The Mathematics of the RSA Public-Key Cryptosystem, Burt Kaliski.RSA Laboratories disponible en la URL http://www.ams.org/samplings/math-awareness-month/06-Kaliski.pdf 2.1.1, 2.1.1

[24] Diffie-Hellman Key Agreement Method, disponible en la URL https://tools.ietf.org/html/rfc2631 2.2.1

[25] Digital Signature Standard (DSS), disponible en la URL http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf 2.3.1

[26] Criptografía asimétrica desde cero, disponible en la URL http://foro.elhacker.net/criptografia/manual_criptografia_asimetrica_desde_cero-t309762.0.html

[27] Use of Elliptic Curve Cryptography (ECC) Algorithms in Cryptographic Message Syntax (CMS), disponible en la URL https://www.ietf.org/rfc/rfc3278.txt

[28] 25 Años de Criptografía con curvas elípticas, Juan G. Tena, IMUVA, Universidad de Valladolid, disponible en la URL http://web.ua.es/es/recsi2014/documentos/papers/25-anos-de-criptografia-con-curvas-elipticas.pdf 1, 3.1

[29] Criptografía con curvas elípticas, Llorenç Huguet Rotger, Josep Rifà Coma, Juan Gabriel Tena Ayuso, disponible en la URL https://www.exabyteinformatica.com/uoc/Informatica/Criptografia_avanzada/Criptografia_avanzada_(Modulo_4).pdf 3.2, 3.3, 3.4.1, 4.3

[30] On the Joint Security of Encryption and Signature in EMV, disponible en la URL https://eprint.iacr.org/2011/615.pdf 5.8.1

[31] TinyOS is an open source, BSD-licensed operating system designed for low-power wireless device, disponible en la URL http://www.tinyos.net/

[32] AVR, Wikipedia, disponible en la URL https://es.wikipedia.org/wiki/AVR 5.8.2

[33] Rhys Weatherley is a member of the DotGNU Steering Committee and the primary author of Portable.NET, disponible en la URL http://rhysweatherley.sys-con.com/ 5.8.2

[34] Curva elíptica, colaboradores de Wikipedia, disponible en la URL https://es.wikipedia.org/wiki/Curva_elíptica

[35] Elliptic curve, Wikipedia contributors, disponible en la URL https://en.wikipedia.org/wiki/Elliptic_curve

[36] Explicit Formulas Database, disponible en la URL http://hyperelliptic.org/EFD/index.html 4.1

[37] DualECDRBG, Wikipedia, disponible en la URL https://en.wikipedia.org/wiki/Dual_EC_DRBG 4.1

[38] Curve25519, Wikipedia, disponible en la URL https://en.wikipedia.org/wiki/Curve25519 4.1.1

[39] Things that use Curve25519, disponible en la URL https://ianix.com/pub/curve25519-deployment.html 4.1.1

[40] FIPS PUB 186-3, disponible en la URL http://csrc.nist.gov/publications/fips/fips186-3/P.pdf 4.1.2

[41] ARM, Elliptic Curve performance: NIST vs Brainpool disponible en la URL https://tls.mbed.org/kb/cryptography/elliptic-curve-performance-nist-vs-brainpool 4.1

[42] Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS), disponible en la URL https://tools.ietf.org/pdf/rfc4492.pdf 5.1

[43] RFC 5639 ECC Brainpool Standard Curves and Curve Generation, disponible en la URL https://tools.ietf.org/html/rfc5639 4.1.3

[44] SEC 2: Recommended Elliptic Curve Domain Parameters, disponible en la URL http://www.secg.org/sec2-v2.pdf 4.1.4

[45] OpenSSL speed man page, disponible en la URL https://www.openssl.org/docs/manmaster/apps/speed.html 5.1

[46] Gniibe: Creating newer ECC keys for GnuPG, disponible en la URL http://www.gniibe.org/memo/software/gpg/keygen-25519.html 5.5

[47] Java Developer Android: java.security.spec, disponible en la URL http://developer.android.com/reference/java/security/spec/package-summary.html 5.7

[48] Bouncy Castle: The Legion of the Bouncy Castle, disponible en la URL https://www.bouncycastle.org/ 5.7

[49] Spongy Castle: repackage of Bouncy Castle for Android, disponible en la URL http://rtyley.github.io/spongycastle/ 5.7

[50] Elliptic Curve Diffie-Hellman 571 on Google Android, disponible en la URL https://es.scribd.com/doc/111518367/Elliptic-Curve-Diffie-Hellman-571-on-Google-Android 5.7

[51] Cryptography Stack Exchange, ECDSA vs RSA: Performance on Android platform and surprising results, disponible en la URL http://crypto.stackexchange.com/questions/16152/ecdsa-vs-rsa-performance-on-android-platform-and-surprising-results 5.7

[52] Developer Economics: Android cryptography tools for beginners, disponible en la URL http://www.developereconomics.com/android-cryptography-tools-for-beginners/ 5.7

[53] NaCl: Networking and Cryptography library, disponible en la URL https://nacl.cr.yp.to/index.html 5.7

[54] Research Plaza: High-security and high-speed protection for computer networks, disponible en la URL https://nacl.cr.yp.to/securing-communication.pdf 5.7

[55] Jailbreaking Is Not A Crime, And EFF Is Fighting To Keep It That Way, disponible en la URL https://www.eff.org/deeplinks/2014/11/jailbreaking-not-crime-and-eff-fighting-keep-it-way 5.7

[56] Bitcoin Wiki: secp256k1, disponible en la URL https://en.bitcoin.it/wiki/Secp256k1 5.6

[57] Atmel, RSA vs ECC Comparison for Embedded Systems, White Paper, disponible en la URL http://www.atmel.com/images/atmel-8951-cryptoauth-rsa-ecc-comparison-embedded-systems-whitepaper.pdf 5.8

[58] Marin L. , Pawlowski MP. , Jara A. : Optimized ECC Implementation for Secure Communication between Heterogeneous IoT Devices, disponible en la URL http://www.ncbi.nlm.nih.gov/pubmed/26343677 5.8

[59] Daniel Hein, Elliptic Curve Cryptography ASIC for Radio Frequency Authentication, disponible en la URL https://online.tugraz.at/tug_online/voe_main2.getvolltext?pCurrPk=43036 5.8.1

[60] iSECPartners nano-ecc: A very small ECC implementation for 8-bit microcontrollers, disponible en la URL https://github.com/iSECPartners/nano-ecc 5.8.2

[61] Micro-ecc, a small and fast ECDH and ECDSA implementation for 8-bit, 32-bit, and 64-bit processors, disponible en la URL https://github.com/kmackay/micro-ecc 5.8.2

[62] ArduinoLibs: Cryptographic Library, disponible en la URL https://rweather.github.io/arduinolibs/crypto.html 5.8.2

[63] NaCl on 8-bit AVR Microcontrollers, disponible en la URL https://eprint.iacr.org/2013/375.pdf 5.8.2

[64] Atmel devices: ATECC508A, disponible en la URL http://www.atmel.com/devices/ATECC508A.aspx 5.8.2

[65] Microsoft Research: MSR Elliptic Curve Cryptography Library, disponible en la URL http://research.microsoft.com/en-us/projects/nums/ 5.4

[66] Low-Latency ECDSA Signature Verification. A Road Towards Safer Traffic, disponible en la URL https://eprint.iacr.org/2014/862.pdf 5.8.3

[67] Deterministic Generation of Elliptic Curves (a.k.a. "NUMSÇurves), disponible en la URL https://www.ietf.org/proceedings/90/slides/slides-90-cfrg-5.pdf 4.1.5

[68] Standards for Efficient Cryptography Group, disponible en la URL http://www.secg.org/

[69] Koblitz curve cryptosystems, disponible en la URL http://www.sciencedirect.com/science/article/pii/S1071579704000395 4.1.2

[70] Controversy Around Bitcoin Elliptic Curve, disponible en la URL http://blog.bettercrypto.com/?p=1004 5.6

[71] Bitcoin: ¿Nuevo esquema criptográfico de firmas?, disponible en la URL http://infocoin.net/2016/06/18/bitcoin-nuevo-esquema-criptografico-de-firmas/ 5.6

[72] Schnorr NIZK Proof: Non-interactive Zero Knowledge Proof for Discrete Logarithm, disponible en la URL https://tools.ietf.org/html/draft-hao-schnorr-01 5.6

[73] Pure PHP Elliptic Curve Cryptography Library, disponible en la URL https://github.com/phpecc/phpecc 5.3

[74] Elliptic Curve PHP-OOP DSA and Diffie-Hellman, disponible en la URL http://www.matyasdanter.com/2010/12/elliptic-curve-php-oop-dsa-and-diffie-hellman/ 5.3

[75] FlexiProvider, a powerful toolkit for the Java Cryptography Architecture (JCA/JCE) disponible en la URL https://www.flexiprovider.de 5.7

[76] Comparison of cryptographic methods based on the arithmetic of elliptic curves (ECC) with symmetric cryptography methods on the Android platform, disponible en la URL http://www.inase.org/library/2014/books/bypaper/MCSI/MCSI-43.pdf 5.7

[77] The Sodium crypto library (libsodium) disponible en la URL https://download.libsodium.org/doc/ 5.7

[78] How to Safely Implement Cryptography Features in Any Application, disponible en la URL https://paragonie.com/blog/2015/09/how-to-safely-implement-cryptography-in-any-application 5.7

[79] New benchmarks of NaCl and scrypt in the browser, disponible en la URL http://eighty-twenty.org/2013/08/15/benchmarking-nacl-and-scrypt-in-the-browser 5.7

[80] js-nacl: Pure-Javascript High-level API to Emscripten-compiled libsodium routines, disponible en la URL https://github.com/tonyg/js-nacl#readme 5.7

[81] ZRTP: Media Path Key Agreement for Unicast Secure RTP, disponible en la URL https://tools.ietf.org/html/rfc6189 5.2

[82] The Zfone Project, disponible en la URL http://zfone.com/ 5.2

[83] A ZRTP Hardphone for secure voice communications, disponible en la URL https://hackaday.io/project/1400-zrtp-hardphone 5.2

[84] The world's first cross-platform scalable free multi-protocol soft switch, disponible en la URL https://freeswitch.org/ 5.2

[85] The Zfone Project, disponible en la URL https://wiki.freeswitch.org/wiki/ZRTP 5.2