Análisis del cifrado ElGamal de un módulo con curvas elípticas propuesto para el GnuPG

Sergi Blanch i Torné[1] y Ramiro Moreno Chiral[2]

[1] Escola Politècnica Superior, Universitat de Lleida. Spain.
d4372211@alumnes.eps.udl.es
[2] Departament de Matemàtica. Universitat de Lleida. Spain.
ramiro@matematica.udl.es

Resumen En la RECSI VIII presentamos un módulo de cifrado y firma digital sobre curvas elípticas para incorporar al GnuPG. El esquema de criptosistema tipo ElGamal utilizado en el desarrollo de ese módulo presenta una debilidad relacionada con la factorización de enteros. Se proponen esquemas alternativos y se analiza la seguridad de los mismos.

1. Introducción.

Hace un tiempo (ver [BMTFC,BM04]) nos propusimos la implementación y desarrollo de un módulo de cifrado y firma digitales con curvas elípticas para incorporarlo al GnuPG. Los algoritmos fundamentales del módulo consistían en un cifrado tipo ElGamal y en el conocido algoritmo ECDSA. Usamos curvas elípticas definidas sobre cuerpos finitos primos, dadas por la forma reducida de Weierstraß

$$E/\mathbb{F}_p:\ y^2 = x^3 + ax + b, \tag{1}$$

donde $a, b \in \mathbb{F}_p$ y el discriminante $-16(4a^3 + 27b^2) \neq 0$.

La base normativa del módulo implementado ([P1363,NIST186-2]) nos daba una garantía de seguridad inicial. Pero ya teníamos claro que la fase de desarrollo debía ir avanzando en paralelo con la de análisis. Y gracias a que así ha sido y gracias también a la potencia de reflexión y comunicación que ofrece liberarlo como software libre, hemos visto una debilidad importante en el cifrado ElGamal implementado en el módulo. Una vez más, esa debilidad no está causada por el algoritmo en sí, sino por el contexto en que se aplica.

Veremos como el cifrado ElGamal, tanto sobre el grupo multiplicativo de los cuerpos finitos como en algunas versiones sobre curvas elípticas, tiene asociada una operación producto que esconde el mensaje a cifrar como un factor de ese producto. En los sistemas *híbridos*, como el GnuPG, en los que se usa un sistema simétrico para cifrar los mensajes largos y un sistema de clave pública o asimétrico, que cifra la clave del simétrico, es normal que ese factor sea precisamente la clave del sistema simétrico. Normalmente, esas claves son enteros de no más de 256 bits. Entonces es planteable una factorización *selectiva* de ese producto ElGamal, que es público, que revele la clave en cuestión.

Así ocurre en nuestro caso, y esa debilidad permite atacar el cifrado de la clave simétrica con bastantes garantías de éxito. Hemos analizado todo ello e intentado solucionar la debilidad.

2. Cifrado tipo ElGamal.

El algoritmo de cifrado ElGamal está ampliamente extendido en aplicaciones criptográficas y multitud de desarrollos, entre otros en el Gnu Privacy Guard (GnuPG). En un sistema de clave pública, este algoritmo implementa de forma muy eficiente un intercambio de claves Diffie-Hellman (DH) para compartir un secreto entre dos comunicandos, con el que, al mismo tiempo, cifra un mensaje. A pesar de que también existe un esquema de firma ElGamal, no goza de gran difusión y es ampliamente superado por el esquema DSA. De hecho, el esquema de firma ElGamal se considera comprometido desde finales de 2000 (ver [BJN]).

Ahora nos proponemos poner a prueba el esquema usado en el cifrado, no sólo cuando el esquema o algoritmo ElGamal se usa sobre el grupo de puntos de las curvas elípticas, sino también cuando es utilizado sobre el grupo multiplicativo de los cuerpos finitos, $\mathbb{F}_q^*$.

2.1. Esquema clásico ElGamal.

El siguiente algoritmo 1 es el esquema básico de cifrado tipo ElGamal en el grupo $\mathbb{F}_p^*$. Con $pkey_M$ se denota un *registro* que es la clave pública del usuario M. Se supone que tal registro está formado por los siguentes campos

$pkey_M.p$	orden del cuerpo finito en el que se plantea el DLP
$pkey_M.g$	generador de ese grupo
$pkey_M.y$	clave pública ($y = g^x$, siendo x la clave privada)

Algoritmo 1 (Cifrado ElGamal)

INPUT: Clave pública $pkey_M$ y entero a cifrar z.
OUTPUT: Cifrado formado por un par de enteros, (a, c).

1: Generar aleatoriamente una clave de sesión $k \in_R [1, (pkey_M.p) - 2]$;
2: $a = (pkey_M.g)^k \mod p$;
3: $b = (pkey_M.y)^k \mod p$; /* $y = g^x \mod p$; $b = (g^x)^k \mod p$ */
4: $c = z \cdot b$;
5: Return (a, c);

Tal algoritmo basa su robustez en los siguientes hechos,

- La aleatorización de la clave de sesión (paso 1) lo hace semánticamente seguro.
- El uso del esquema Diffie-Hellman (DH) en los pasos 2 y 3 (el valor b es la clave compartida que no viajará por el canal).

- La dificultad de resolver el IFP (*Integer Factorization Problem*) en el paso 4.

Si un atacante resuelve el logaritmo discreto calculando k a partir de a, consigue el secreto compartido b que se usa para camuflar la información a cifrar en el producto $c = z \cdot b$, que es la operación ElGamal propiamente dicha y, por lo tanto, el mensaje original z. Pero también puede ser atacado directamente el esquema de ElGamal resolviendo el problema de factorización de enteros (IFP) y factorizar c. Esta factorización debería ser lo suficientemente difícil, pero dependiendo de cómo sean los factores z y b, puede no serlo.

Estudio del esquema.

Las claves usadas comunmente en criptografía están definidas sobre cuerpos finitos, y las opciones de criptosistema más frecuentes son RSA o ElGamal/DSA. De todos modos, las longitudes de claves sobre cuerpos finitos, como es habitual, siguen creciendo: lo que hace unos años se consideraba una clave segura, hoy no tiene una longitud suficiente; así, las claves de 1024 bits son ya cosa del pasado: las recomendables han aumentado hasta longitudes de 2048 bits. Esto que parecería una mejora, puede constituir un problema.

Fijémonos en la operación de ElGamal. Es un simple producto, cuyos factores son de muy distintos tamaños. Efectivamente, el factor que proviene del protocolo de intercambio Diffie-Hellman, tiene una longitud considerable y podríamos decir que sólo rompiendo el protocolo lo obtendríamos. Con las longitudes actuales comentadas, será frecuentemente un valor de 2048 bits.

El principal problema está en el valor que queremos cifrar, z. ¿Qué tamaño tiene? Al usar sistemas híbridos de cifrado, como el GnuPG, se genera una clave de sesión para cifrar con un sistema simétrico el grueso del mensaje y es esta clave la información que ciframos de forma asimétrica. Por lo tanto, el tamaño de ese factor estará entre 128 y 256 bits. Aunque eso constituye una ventaja, ya que ciframos individualmente para cada destinatario un dato relativamente pequeño, evitando la expansión de bits propia de los sistemas asimétricos en el cifrado de todo el mensaje, también puede ser una debilidad por la gran diferencia de tamaño con el otro factor del cifrado. Efectivamente, se puede pensar en una factorización de c que busque factores relativamente pequeños como son esos valores z: tal factorización, aunque difícil, es viable.

Esa clave de sesión es, pues, nuestro problema. Su origen le da unas características muy peculiares, que debilitan el cifrado tipo ElGamal.

2.2. Esquema Elíptico ElGamal: ECElGamal.

Nuestro trabajo consistió en el desarrollo de un módulo criptográfico con curvas elípticas definidas sobre cuerpos finitos primos $\mathbb{F}_p$ y su inclusión dentro del GnuPG: uno de nuestros propósitos era hacer llegar el uso generalizado de los criptosistemas sobre curvas elípticas al público no especializado por medio del software libre. Veamos, pues, cómo funciona un cifrado de este tipo. En el

algoritmo 2 podemos ver la descripción del esquema con curvas elípticas ECElGamal, que es una "traducción" directa del algoritmo 1 sobre cuerpos finitos $\mathbb{F}_p$. En este caso los campos de la clave pública $pkey_M$ son n, el orden del grupo de puntos generado por G, $n = |\langle G \rangle|$, el propio punto generador G, y el punto P, clave pública, tal que $P = d \cdot G$, siendo d un entero que se usa como clave privada.

Algoritmo 2 (Cifrado ECElGamal)

INPUT: Clave pública $pkey_M$ y entero a cifrar z.
OUTPUT: Cifrado formado por un par de puntos, (R, C).

1: Generar aleatoriamente una clave de sesión $k \in_R [1, (pkey_M.n) - 1]$;
2: $R = k \cdot pkey_M.G$;
3: $Q = k \cdot pkey_M.P$; /* $P = d \cdot G$; $Q = k \cdot (d \cdot G)$ */
4: Convertir el mensaje a punto de la curva elíptica $z \rightarrow Z$;
5: $C = Z + Q$;
6: Return (R, C);

La "traducción" a curvas elípticas es sencilla: esencialmente consiste en sustituir el grupo multiplicativo cíclico $(\mathbb{F}_p^*, \cdot)$ por un subgrupo cíclico $(\langle G \rangle, +)$ del grupo de puntos de la curva elíptica, $E(\mathbb{F}_p)$, con lo que

- la notación cambia de multiplicativa a aditiva (productos a sumas, y potencias a productos de puntos por enteros);
- el orden del grupo $\langle G \rangle$, ha de ser lo más próximo posible al del grupo total $E(\mathbb{F}_p)$;
- el mensaje z se ha de convertir a punto.

Estudio del esquema.

Este esquema con curvas elípticas, tiene una clara ventaja: el producto que se usa como operación básica del cifrado ElGamal sobre $\mathbb{F}_p^*$ y que constituye la debilidad ya indicada, es ahora una suma de puntos $C = Z + Q$ que no se puede "deshacer" sin conocer alguno de los sumandos, y eso sólo es posible si resolvemos alguno de los ECDLP subyacentes en los pasos 2 y 3 del algoritmo 2.

Pero la conversión del mensaje z a cifrar a punto Z de la curva elíptica **no** es trivial. No siempre resulta z una abscisa de un punto de $E(\mathbb{F}_p)$, ya que, obviamente, no siempre $z^3 + az + b$ es un cuadrado en el cuerpo base $\mathbb{F}_p$. Aunque se han usado diferentes soluciones para este problema, todas adolecen de un defecto u otro. Y, en general, en todas ellas es pensable que se podría estar dando información adicional sobre el valor de z. Por ello debemos descartar este esquema de cifrado.

3. Esquema ECDH+ElGamal.

Entendiendo el esquema inicial del algoritmo 1 como dos partes separadas, por un lado el intercambio DH de claves y por otro la operación producto, asociada propiamente al cifrado ElGamal, podemos buscar una solución al anterior problema. Si realizamos un intercambio de claves del tipo Diffie-Hellman sobre curvas elípticas obtendremos un secreto compartido con el cual realizar la operación ElGamal sobre enteros, resultando el algoritmo 3 descrito a continuación.

Algoritmo 3 (Cifrado ECDH+ElGamal)

INPUT: Clave pública $pkey_M$ y texto en claro numérico z.
OUTPUT: Punto resultante R, cifra c.

1: Generar aleatoriamente una clave de sesión $k \in_R [1, (pkey_M.n) - 1]$;
2: $R = k \cdot pkey_M.G$;
3: $Q = k \cdot pkey_M.P$; /* $P = d \cdot G$; $Q = k \cdot (d \cdot G)$ */
4: $c = z \cdot Q_x$; /* Aquí Q_x es la abscisa del punto Q, es decir, $Q_x = x(Q)$ */
5: Return (R, c)

Conseguimos evitar la molesta conversión de la información a cifrar en punto de la curva elíptica, resultando, en ese sentido, un esquema más simple que el del algoritmo 2, que estaba implementado usando sólo operaciones sobre los puntos de la curva. Pero volvemos a encontrarnos con el problema inicialmente plantado: la debilidad del producto ElGamal y de su posible ataque mediante una factorización selectiva.

Estudio del esquema.

Por lo tanto, con este algoritmo hemos vuelto al principio: recaemos en la debilidad IFP que sufre la operación ElGamal. Con algunas diferencias sobre la situación descrita en el estudio del algoritmo 1.

Suponiendo que el intercambio DH elíptico ha de tener una seguridad equivalente a ElGamal 2048 sobre cuerpos finitos (a su vez equivalente a un RSA con módulo de 2048 bits), el orden del grupo de puntos $E(\mathbb{F}_p)$ habría de ser de unos 256 bits, es decir, el punto Q, obtenido en el paso 3, vendrá dado por dos coordenadas de ese tamaño (igualmente el punto R y el punto–clave pública P). La clave de sesión que proviene del cifrado simétrico tendrá como mínimo 128 bits. Así pues, el producto planteado en el paso 4 del algoritmo 3 tendrá factores de esos rangos: 128 y 256 bits.

Es decir, tenemos en este caso tamaños de los factores más próximos entre sí que los del algoritmo original: si en el estudio del primer esquema 1 sobre cuerpos finitos, teníamos una gran distancia entre los operandos del producto, ahora tenemos valores de una longitud similar. Pero estamos en lo mismo, los operandos tienen características que les hacen demasiado peculiares.

Al factorizar el valor c del producto, podemos encontrarnos con dos situaciones extremas. Una en la que ambos factores fuesen primos: se trataría de factorizar un entero de 384 bits, producto de dos primos de tamaño similar. No parece una tarea inabordable, sino todo lo contrario: por un lado, estamos realizando un intercambio de claves bastante robusto, mientras que por otro confíamos en una factorización de un entero mucho menor de lo que aceptaríamos para un RSA. En el otro extremo podemos encontrar unos operandos altamente compuestos, con lo que obtendríamos una amplia lista de factores rápidamente (mucho más que en el otro caso). Y la labor mas pesada ahora sería separar esa lista en dos grupos, uno para obtener un entero de 128 bits y el otro para uno de 256.

4. Esquema alternativo ECDH + AES256.

Llegamos a un punto en que hemos de aceptar que el esquema ElGamal no nos sirve para nuestros propósitos. Si lo usamos con sólo operaciones en puntos de la curva elíptica, como en el algoritmo 2, nos encontramos con una solución poco pulcra y con la desconfianza de que el truco que usemos para convertir el mensaje en punto esté revelando información a un criptoanalista. Y si jugamos a medias con el intercambio de claves en curvas elípticas y la operación producto de enteros clásica de ElGamal, resulta fatal para el propósito de seguridad cripotográfica.

Sólo nos queda buscar una alternativa completa a la operación ElGamal. Tendremos que seguir pensando en una operación con enteros, como en el algoritmo 3, pues recuperar algo del algoritmo 2 nos haría volver a caer en el problema de la conversión del mensaje a punto de la curva elíptica.

Parece que lo más sensato es pensar en una operación que sustituya al producto, pero de robustez garantizada. Planteamos como solución una operación basada en un cifrado simétrico. El secreto compartido que nos genera la parte ECDH, cumple con el ideal de una clave de sesión: nunca viaja por el canal y es conocido por ambos participantes lícitos de la comunicación. Podemos ver este nuevo esquema en el algoritmo 4.

Algoritmo 4 (Cifrado ECDH+AES)

INPUT: Clave pública $pkey_M$ y texto en claro numérico z.
OUTPUT: Par punto resultante R y cifra c.

1: Generar aleatoriamente una clave de sesión $k \in_R [1, (pkey_M.n) - 1]$;
2: $R = k \cdot pkey_M.G$;
3: $Q = k \cdot pkey_M.P$; /* $P = d \cdot G$; $Q = k \cdot (d \cdot G)$ */
4: $c = \text{aes256}(z, \text{sha256}(Q_x))$;
5: Return (R, c);

Este esquema nos fué propuesto inicialmente por M. Mylnikov el 7 de Noviembre de 2004. No solo se trata de sustituir la operación que ya hemos descartado por un cifrado simétrico, sino de evitar futuros errores. Usando la operación basada en el AES garantizamos la seguridad de nuestro esquema mediante la

del AES, intensa y regularmente analizado en la actualidad. Ante un hipotético caso en el que el criptosistema AES se vea comprometido, la solución pasa por fortalecerlo o incluso llegar a usar otro esquema más robusto.

Como medida adicional se requiere introducir una función de resumen, sha256, para así obtener siempre una clave de cifrado simétrico de la misma longitud, que sea máxima y que también podrá ser computable al descifrar, según puede advertirse en el esquema de descifrado propuesto en el algoritmo 5.

ALGORITMO 5 (DESCIFRADO ECDH+AES)

INPUT: Clave secreta $skey_M$ y el par punto resultante R y cifra c.
OUTPUT: Texto en claro numérico z.

1: $Q = skey_M.d \cdot R$;
2: $z = \text{aes256}^{-1}(c, \text{sha256}(Q_x))$;
3: Return (z);

5. Conclusión

Después de haber encontrado una situación en la que el esquema ElGamal se ve comprometido, no por su diseño sino por el caso particular en el que lo aplicamos, hemos encontrado una alternativa que parece resolver el compromiso. Se ha puesto en duda un esquema ampliamente utilizado, y esta duda afecta al más importante de sus usos: los algoritmos híbridos, donde el "texto" a cifrar es una clave se sesión con la que se ha cifrado simétricamente el grueso de la información.

La solución aportada presenta una gran ventaja y es que es un esquema constantemente puesto a prueba ya que forma parte de otras soluciones. Actualmente el algoritmo AES esta bajo presión ([OHT05,BERN05]) debido a que durante este último año han aparecido nuevos puntos de partida para el criptoanalisis. Se trata de ataques eminentemente informáticos que aprovechando pequeños detalles del computador, consiguen hacerse con información útil del cifrado. Por el momento, tiene un impacto comparable al producido por el riesgo de almacenar una frase de paso en la memoria de intercambio de disco.

Debemos estar muy pendientes de la evolución de estos ataques y en especial de las contramedidas ([OST05]) planteadas contra ellos. Por el momento, no se han propuesto soluciones factibles a nivel de la aplicación. Tan solo existe alguna propuesta, mas o menos aplicable, para ofrecer una protección desde el sistema operativo o incluso desde la arquitectura del computador.

Referencias

[AES99] J. Deamen and V. Rijmen, *AES Proposal: Rijndael, version 2*, AES submission, 1999.

[BERN05] D. Bernstein, *Cache-timing attacks on AES*, 2005.

[BJN] D. Boneh, A. Joux and P.Q. Nguyen, *Why textbook ElGamal and RSA encryption are insecure*, Proceedings of Asiacrypt'00, Lecture Notes in Computer Science, no. 1976 (2000), Springer, pp. 30-43.

[BM04] S. Blanch and R. Moreno, *Implementación GnuPG con curvas elípticas*, Avances en Criptología y Seguridad de la Información. Proceedings RECSI VIII, 2004, pp. 515–526.

[BMTFC] S. Blanch and R. Moreno, *GnuPG Implementation with Elliptic Curves*, Trabajo final de carrera, Marzo 2004.

[BRUM03] D. Brumley and D. Boneh, *Remote timing attacks are practical*, 2003.

[DHAES] M. Abdalla, M. Bellare and P. Rogeway, *DHAES: An encryption scheme based on the Diffie-Hellman Problem*, 1999.

[KSWH98] J. Kelsey, B. Scheneier and D. Warner, Chris Hall, *Side channel cryptanalysis of product ciphers*, 5th European Symposium on Research in Computer Security, LNCS 1485, 97-110, Sptinger-Verlag, 1998.

[NFS] A. Shamir and E. Tromer, *Special-Purpose hardware for factoring: the NFS sieving step*, 2005.

[NIST186-2] National Institute of Standards and Technology, *Digital Signature Standard (DSS) (FIPS PUB 186-2)*, 2000 January 27.

[NIST180-2] National Institute of Standards and Technology, *Secure Hash Standard (SHS) (FIPS PUB 180-2)*,2002.

[NIST197] National Institute of Standards and Technology, *Advanced Encryption Standard (AES) (FIPS PUB 197)*, 2001.

[OHT05] M. O'Hanlon and A. Tonge, *Investigation of cache-timing attacks on AES*, 2005.

[OST05] D. Arne Osvik, A. Shamir and E. Tromer, *Cache attacks and countermeasures: the case of AES*, 2005.

[P1363] IEEE P1363 Standard Specifications for Public key Cryptography. 2000 January 30.