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, UdL, sblanch@alumnes.udl.cat
[2] Departament de Matemàtica, UdL, ramiro@matematica.udl.cat

Resumen

Desde marzo de 2007, el paquete de software libre Gnu Privacy Guard (GnuPG) incorpora en la versión de desarrollo soporte para criptografía con curvas elípticas. El código allí incluido fué realizado como proyecto final de carrera del primero de los autores. Sin embargo, y aunque el esquema de firma ECDSA se mantiene en esa versión de desarrollo, el esquema de cifrado *ElGamal* sobre curvas elípticas presenta algunos problemas que se analizan y se plantea una solución alternativa como propuesta a la comunidad del GnuPG.

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 el conocido algoritmo ECDSA. Se usaron curvas elípticas definidas sobre cuerpos finitos primos, dadas en la forma reducida de Weierstraß

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

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]) da una garantía de seguridad inicial. Pero ya estaba 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, se ha visto una debilidad importante en el cifrado ElGamal implementado en el módulo. Como es frecuente, esa debilidad no está causada por el algoritmo en sí, sino por el contexto en que se aplica.

El esquema de cifrado ElGamal (cf. por ejemplo [HAC, §8.4]) sobre un grupo cíclico $\mathcal{G}$ cualquiera, con notación multiplicativa y generador g, consiste en usar el intercambio de claves Diffie–Hellman (DH) como una parte de la cifra. Así, se calcula $\gamma = g^k$, donde k es una clave de sesión, menor que el orden de $\mathcal{G}$. El mensaje m, se convierte en un elemento de $\mathcal{G}$, i.e., $m \in \mathcal{G}$, y se "esconde" operándolo en $\mathcal{G}$ con la clave común DH, $\delta = m.(g^a)^k$, siendo g^a la clave pública del receptor. El mensaje cifrado es el par (γ, δ). El receptor puede recuperar el mensaje ya que $m = \delta(\gamma^a)^{-1}$ y a es su clave privada. Visto el esquema sobre el grupo multiplicativo de un cuerpo finito primo, $\mathbb{F}_p^*$, m será un entero $1 \leq m \leq p-1$, y la segunda parte del cifrado se calcula mediante productos y potencias módulo p,

$$\delta = m.(g^a)^k \mod p.$$

En las curvas elípticas la operación es la suma y la "suma abreviada" $[n]P$ (que se denota simplemente $k \cdot P$) en el grupo de puntos $E(\mathbb{F}_p)$,

$$\delta = m + k \cdot (a \cdot g) \in E(\mathbb{F}_p),$$

siendo ahora el mensaje y el generador puntos de $E(\mathbb{F}_p)$, es decir $m, g \in E(\mathbb{F}_p)$. La robusted del criptosistema se basa en la del problema Diffie–Hellman: obtener g^{ak} a partir de g^a y g^k en el grupo en que se haya definido.

Pero, al usar curvas elípticas, aparece una nueva dificultad: representar el mensaje en cla-

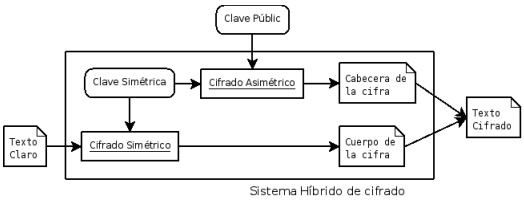

Figura 1: Esquema de funcionamiento de un sistema híbrido de cifrado.

ro m como punto de $E(\mathbb{F}_p)$. Como se verá no existen soluciones buenas para ello, por lo que se opta por tratarlo como entero $1 \leq m \leq p-1$ y esconderlo modularmente,

$$\delta = m.x(k \cdot (a \cdot g)) \mod p,$$

donde $x(\cdot)$ es la abscisa del punto correspondiente. Pero el tamaño de p en los criptosistemas con curvas elípticas es mucho menor que el correspondiente a ElGamal original sobre $\mathbb{F}_p$ (160 bits frente a 1024) y puede pasar que $m > p$ con mucha probabilidad, de modo que al descifrar no se obtenga m sino $m \mod p$. Concretamente, 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 m sea precisamente la clave del sistema simétrico. Normalmente, esas claves son *strings* de no más de 256 bits que, al tratarlos como mensajes del criptosistema asimétrico, se han de convertir a enteros. Esos tamaños resultan ser mayores que el módulo p en un criptosistema con curvas elípticas. Aumentar el tamaño de p es renunciar a una de las mayores ventajas de estos criptosistemas: el menor tamaño de las claves con la misma seguridad de ElGamal sobre cuerpos finitos o del RSA. La solución podría ser no hacer modular ese producto, pero entonces, como es público, su factorización es plausible, revelando así la clave en cuestión.

Es lo que ocurre en nuestra implementación para el `GnuPG`, y esa debilidad permite atacar el cifrado asimétrico que se realiza a la clave simétrica, con bastantes garantías de éxito. Hacer este producto modular significa, pues, rechazar toda pretensión de cifrar una información cuyo tamaño sea mayor que el de la clave. Para el caso de un esquema sobre cuerpos finitos eso no produce ningún contratiempo, pues las claves asimétricas son mucho mayores que las simétricas. Pero cuando sobre el grupo cíclico sobre el que se quiere hacer criptografía es un subgrupo del $E(\mathbb{F}_p)$, esta diferencia entre simétricos y asimétricos conduce peligrosamente a la posibilidad de la factorización. Se ha analizado todo ello e ingeniado una solución de esta debilidad que no deje ninguna limitación, ni en ésta ni en posteriores mejoras.

2. Cifrado tipo ElGamal.

El algoritmo de cifrado ElGamal está ampliamente extendido en aplicaciones criptográficas y multitud de desarrollos, entre otros en el `GnuPG`. En un sistema de clave pública, este algoritmo implementa de forma muy eficiente un intercambio de claves Diffie-Hellman 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 está superado por el esquema DSA. De hecho, el esquema de firma ElGamal se considera comprometido desde finales de 2000 (ver [BJN]).

Nuestra propuesta es 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, $\mathbb{F}_p^*$. Como veremos, resulta altamente dependiente del tamaño de p, pues es el segundo operando el que plantea el problema del tamaño del espacio de mensajes o, en su caso, el de factorización de enteros (`IFP`).

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
$pkey_M.g$	generador: $\mathbb{F}_p^* = \langle g \rangle$
$pkey_M.y$	clave pública $y = g^x$, siendo x la privada

El algoritmo 1 recibe, como una componente de la entrada, el mensaje a cifrar que damos en llamar z. Hay que destacar que, a pesar de que se trata como un elemento de $\mathbb{F}_p^*$, en nuestra implementación representará una clave de cifrado simétrico, que originalmente son *strings* de no más de 256 bits.

ALGORITMO 1 (Cifrado ElGamal)

INPUT: Clave pública $pkey_M$ y entero a cifrar $z \in \mathbb{F}_p^*$.
OUTPUT: Cifrado formado por un par de enteros, $(a, c) \in \mathbb{F}_p^* \times \mathbb{F}_p^*$.

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: $c = z \cdot b \mod p$;
5: Return (a, c);

Estudio del esquema.

Tal algoritmo basa su robustez en los siguientes hechos,

- La aleatorización de la clave de sesión (paso 1) lo hace semánticamente seguro y resistente a los clásicos ataques por texto cifrado escogido.
- El uso del esquema Diffie-Hellman en los pasos 2 y 3 (el valor b es la clave compartida que no viajará por el canal).
- Sólo resolviendo el DLP (*Discrete Logarithm Problem*) un atacante puede calcular k a partir de a. Así conseguirá el secreto compartido b que se usa para camuflar la información a cifrar en el producto $c = z \cdot b \mod p$, que es la operación ElGamal propiamente dicha y, por lo tanto, el mensaje original z

Sin embargo, como todo esquema básico ElGamal sobre un grupo genérico $\mathcal{G}$, también éste adolece de una deficiencia: la limitación del espacio de mensajes, ya que $z \in \mathbb{F}_p^*$ (cf. por ejemplo [DHAES]). Esta limitación es muy importante si el cardinal del grupo es "pequeño".

2.2. Esquema Elíptico ElGamal: ECElGamal.

El trabajo original 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 se puede 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 en punto de la curva elíptica $z \to Z$;
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 $(\mathcal{G}, +) = (\langle G \rangle, +)$ 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 $\mathcal{G}$, ha de ser lo más próximo posible al del grupo total $E(\mathbb{F}_p)$;
- el mensaje z se ha de convertir en 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 se resuelven algunos de los ECDLP, *Elliptic Curve Discrete Logarithm Problem*, subyacentes en los pasos 2 y 3 del algoritmo 2.

Pero la conversión del mensaje z a cifrar en punto $Z \in E(\mathbb{F}_p)$ **no** es trivial. Recordemos que el mensaje a cifrar z es originalmente un *string* de bits que constituye la clave de un sistema de cifrado simétrico, por lo que no se puede hacer ninguna conjetura sobre su estructura. Una vez transformado a un entero $\mod p$, i.e., $z \in \mathbb{F}_p$, se puede considerar candidato a abscisa de un punto de $E(\mathbb{F}_p)$. Pero no siempre resulta z una abscisa de un punto racional de la curva elíptica $E/\mathbb{F}_p$: efectivamente, según la ecuación (1), el valor $z^3 + az + b$ ha de ser un cuadrado en el cuerpo base $\mathbb{F}_p$. Si no lo es, hay que hacer alguna transformación de z, $z \mapsto z_1$, tal que el símbolo de Legendre de $z_1^3 + az_1 + b$ sobre p sea 1. Entonces $Z_1 = (z_1, y_1) \in E(\mathbb{F}_p)$, donde y_1 es una de las dos raíces cuadradas de $z_1^3 + az_1 + b$ en $\mathbb{F}_p$. Pero, para poder descifrar, hay que enviar, junto con el par de puntos $(R, Z_1 + Q)$, del paso 6 del algoritmo 2, información adicional que permita invertir la transformación $z \mapsto z_1$.

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 se debe 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 de claves DH y por otro la operación producto, asociada propiamente al cifrado ElGamal, se puede buscar una solución al anterior problema. Realizando un intercambio de claves del tipo Diffie-Hellman sobre curvas elípticas, se obtiene un secreto compartido con el cual llevar a cabo 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: $c = z \cdot Q_x \mod p$;
/* Q_x es la abscisa de Q, $Q_x = x(Q)$ */
5: Return (R, c)

Estudio del esquema.

Con este algoritmo se evita 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 otra vez aparece el problema ya comentado de la limitación del espacio de mensajes: para recuperar z ha de ser $z \in \mathbb{F}_p^*$, es decir, $1 \leq z \leq p - 1$.

Una primera aproximación es hacer el producto del paso 4 no modular,

$$4.\ c = z \cdot Q_x.$$

Pero la robusted del esquema depende ahora de la dificultad de resolver el IFP (*Integer Factorization Problem*) en ese paso 4. Y en nuestra implementación esa dificultad es menor de lo habitual. Efectivamente, 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, ese producto no modular, dado como alternativo al del paso 4 del algoritmo 3, tendrá factores de esos rangos, 128 y 256 bits, y por tanto, un tamaño final de 384 bits: ¡nada difícil de factorizar para algoritmos modernos como el de la GNFS, *General Number Field Sieve* ([BBFK05])!.

Manteniendo el citado paso 4 como producto modular aparecerán con frecuencia problemas de los tamaños relativos de z y p. Así ocurrirá cuando se use una clave z para el AES256 y un cuerpo base de la curva elíptica para el esquema ECElGamal de 192 bits (el menor de los permitidos en la implementación): será $z > p$ y, al descifrar, no se recuperará z sino $z \mod p$.

Una situación como la descrita haría que la ejecución del `GnuPG` reúsase siempre el cifrado. Cabría pensar que si se "troceara" la clave de sesión z que se ha usado en el cifrado simétrico y se procediera a dos cifrados asimétricos separados, se resolvería el problema de los tamaños relativos. Pero esta grata solución teórica no resulta posible en la prática de las implementaciones del `OpenPGP` [RFC2440]: en las más usadas, y una de ellas es el `GnuPG`, el módulo de cifrado asimétrico recibe como entrada un solo *string* de bits y sólo puede devolver como salida una cifra de estructura estática; por lo tanto, no es posible decidir cuándo no hace falta trocear, y cuándo sí (en el futuro, incluso podría no ser suficiente con dos partes). Esta solución, pues, también se queda corta: hay que encontrar algo que resulte más genérico y no obligue a dar soluciones *ad hoc* en cada paso y momento.

4. Esquema alternativo ECDH + AES256.

Llegados a este punto, se ha de aceptar que el esquema ElGamal no sirve para los propósitos expuestos. Implementado con sólo operaciones en el grupo de puntos $E(\mathbb{F}_p)$, como en el algoritmo 2, aparecen soluciones poco recomendables y no hay seguridad de que el truco usado para convertir el mensaje en punto esté revelando información a un criptoanalista. La operación producto ElGamal módulo p hará el mensaje irrecuperable, dado el pequeño tamaño de p. Y, el punto medio consistente en el intercambio de claves en curvas elípticas y la operación producto no modular, resulta fatal para el propósito de seguridad criptográfica.

Sólo queda buscar una alternativa completa a la operación ElGamal. Hay que seguir pensando en una operación con enteros, como en el algoritmo 3, pues recuperar algo del algoritmo 2 nos llevaría de nuevo al problema de la conversión del mensaje en punto de la curva elíptica. Parece que lo más sensato es pensar en una operación que sustituya al producto, pero de robustez garantizada. Se da como solución alternativa una operación basada en un cifrado simétrico. El secreto compartido que 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. Se puede 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);

Conviene aclarar la notación usada en el paso 4 anterior: con $c = \text{aes256}(z, \{\text{sha256}(Q_x)\})$

se quiere significar que c es el cifrado obtenido mediante un AES256 del valor z, al usar como clave de ese cifrado simétrico $\{\text{sha256}(Q_x)\}$.

Este esquema fué propuesto inicialmente en [MMR]. No sólo se trata de sustituir la operación descartada por un cifrado simétrico, sino de evitar futuros errores. Al usar el AES se garantiza la seguridad de este 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. En este mismo sentido se propuso a la P1363a del IEEE, como alternativa a los cifrados tipo ElGamal, un esquema más general que el aquí desarrollado, por parte de Abdalla, Bellare y Rogeway, [DHAES]. Como ya se ha señalado, en esa propuesta se subrayan algunas debilidades de los cifrados ElGamal: la limitación del espacio de mensajes es la que aparece como definitiva en los cifrados con curvas elípticas en sistemas híbridos.

Más allá de esta concreción, es de desear un algoritmo versátil, eficiente y que proporcione un espacio de mensajes sin demasiadas restricciones. Como medida adicional hay 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 calculable al descifrar, según se advierte 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);

En este caso $skey_M$ representa la estructura de clave secreta que contiene los campos de la clave pública además del campo $skey_M.d$ que es el entero que se usa como clave privada. Y, al igual que al cifrar, la expresión $z = \text{aes256}^{-1}(c, \{\text{sha256}(Q_x)\})$ indica la recuperación de z al descifrar c mediante el AES256 con la misma clave $\{\text{sha256}(Q_x)\}$.

Estudio del esquema.

Los algoritmos que se han expuesto como solución presentan la seguridad basada en el ECDLP y en la del AES y garantizan diferentes niveles de seguridad, que se especifican a continuación.

- La aleatoriedad de la clave de sesión k, permite seguridad semántica y resistencia a los ataques por texto cifrado elegido.
- No son posibles los ataques de [BJN] ya que el cifrado ElGamal propiamente dicho ha desaparecido: no existe la clásica operación producto de este tipo de cifrados.
- Y, lo que parece más importante en los sistemas híbridos, el espacio de mensajes no tiene limitaciones de tamaño: las claves simétricas z se cifran con AES.

5. Conclusión

Se ha visto como el esquema ElGamal se ve comprometido, no por su diseño sino por el caso particular en el que se aplica y se ha podido encontrar una alternativa que parece resolver el compromiso. Un esquema ampliamente utilizado ha quedado puesto en duda, y esta duda afecta al más importante de sus usos: los algoritmos híbridos, donde el texto a cifrar es una clave de sesión con la que se ha cifrado simétricamente el grueso de la información.

La solución aportada presenta una gran ventaja, ya que el AES es un criptosistema muy generalizado y constantemente puesto a prueba. Actualmente el algoritmo AES esta bajo presión ([OHT05, BERN05]) debido a que durante este pasado año han aparecido nuevos puntos de partida para el criptoanalisis. Se trata de ataques eminentemente informáticos que aprovechando pequeños detalles del ordenador, 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.

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

Referencias

[DHAES] M. Abdalla, M. Bellare and P. Rogeway, *DHAES: An encryption scheme based on the Diffie-Hellman Problem*, Submission to IEEEE P1363a, 1999.

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

[BBFK05] F. Bahr, M. Boehm, J. Franke, and T. Kleinjung, *RSA640 factored by GNFS*, `NMBRTHRY @LISTSERV. NODAK. EDU`, November 2005.

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

[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, EPS, Universid de Lleida, Marzo 2004.

[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.

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

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

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

[RFC2440] J. Callas, L. Donnerhacke, H. Finney, R. Thayer, *Request for Comments: 2440; OpenPGP Message Format* November 1998.

[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.

[HAC] A. J. Menezes, P. C. van Oorschot and S. A. Vanstone, *Handbook of Applied Cryptography*, Fifth Printing (August 2001).

[MMR] M. Mylnikov, Comunicación personal, Noviembre de 2004.

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

[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.

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

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