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

Sergi Blanch i Torné
y
Ramiro Moreno Chiral

Criptografía y Grafos
Departamento de Matemáticas
Universidad de Lleida

11 de septiembre de 2007

(Versión 2.1)

Outline

Introducción

Cifrado tipo ElGamal

Esquema ECDH+ElGamal

Esquema alternativo ECDH+AES256

Conclusión

Algunas características

Algunas características

- Al decir *GnuPG* uno piensa en *software libre*.

Algunas características

- Al decir *GnuPG* uno piensa en *software libre*.
- Software criptográfico de proposito general: cumple con estandares como el rfc2440.

Algunas características

- Al decir *GnuPG* uno piensa en *software libre*.
- Software criptográfico de proposito general: cumple con estandares como el rfc2440.
- Proyecto eccGnuPG:

Algunas características

- Al decir *GnuPG* uno piensa en *software libre*.
- Software criptográfico de proposito general: cumple con estandares como el rfc2440.
- Proyecto eccGnuPG:
  - GnuPG v1.4: Soporta de cifrado y firma con curva elíptica.

Algunas características

- Al decir *GnuPG* uno piensa en *software libre*.
- Software criptográfico de proposito general: cumple con estandares como el rfc2440.
- Proyecto eccGnuPG:
  - GnuPG v1.4: Soporta de cifrado y firma con curva elíptica.
  - GnuPG v2: Soporta firma y tiene proyectado el cifrado.

Algunas características

- Al decir *GnuPG* uno piensa en *software libre*.
- Software criptográfico de proposito general:
  cumple con estandares como el rfc2440.
- Proyecto eccGnuPG:
  - GnuPG v1.4: Soporta de cifrado y firma con curva elíptica.
  - GnuPG v2: Soporta firma y tiene proyectado el cifrado.
    Filosofía modular de *Unix*:
    Cosas pequeñas que hacen muy bien tareas simples y que al unirlas hacen bien tareas complejas.

Algunas características

- Al decir *GnuPG* uno piensa en *software libre*.
- Software criptográfico de proposito general: cumple con estandares como el rfc2440.
- Proyecto eccGnuPG:
  - GnuPG v1.4: Soporta de cifrado y firma con curva elíptica.
  - GnuPG v2: Soporta firma y tiene proyectado el cifrado. Filosofía modular de *Unix*: Cosas pequeñas que hacen muy bien tareas simples y que al unirlas hacen bien tareas complejas.
  - LibPth + LibGpg-error + *LibGcrypt* + LibAssuan + LibKsba + Gnupg-2.0

Algunas características

- Al decir *GnuPG* uno piensa en *software libre*.
- Software criptográfico de proposito general: cumple con estandares como el rfc2440.
- Proyecto eccGnuPG:
  - GnuPG v1.4: Soporta de cifrado y firma con curva elíptica.
  - GnuPG v2: Soporta firma y tiene proyectado el cifrado. Filosofía modular de *Unix*: Cosas pequeñas que hacen muy bien tareas simples y que al unirlas hacen bien tareas complejas.
    - LibPth + LibGpg-error + *LibGcrypt* + LibAssuan + LibKsba + Gnupg-2.0
- Y, sobre todo, es un *sistema híbrido*.

Sistemas híbridos

Sistemas híbridos

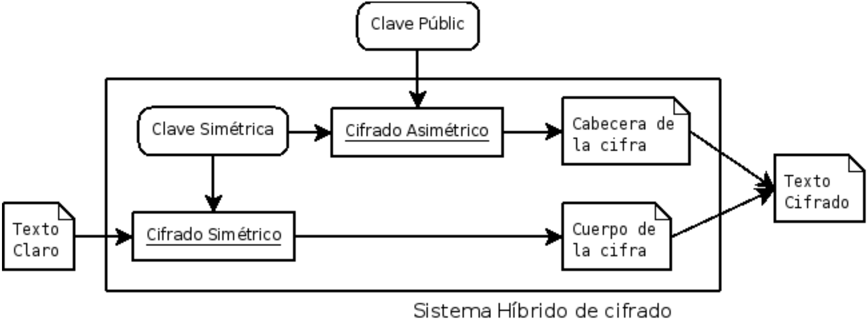

Figura: Esquema de funcionamiento de un sistema híbrido de cifrado.

Nociones sobre curvas elípticas

Nociones sobre curvas elípticas

Una curva elíptica definida sobre un cuerpo finito está determinada por una ecuación de Weierstraß

$$E/\mathbb{F}_p: \; y^2 = x^3 + ax + b, \text{ donde } a, b \in \mathbb{F}_p \text{ y } 4a^3 + 27b^2 \neq 0. \quad (1)$$

Nociones sobre curvas elípticas

Una curva elíptica definida sobre un cuerpo finito está determinada por una ecuación de Weierstraß

$$E/\mathbb{F}_p: \; y^2 = x^3 + ax + b, \text{ donde } a, b \in \mathbb{F}_p \text{ y } 4a^3 + 27b^2 \neq 0. \quad (1)$$

Grupo de puntos:

$$E(\mathbb{F}_p) = \{(x, y) \in \mathbb{F}_p \times \mathbb{F}_p : \; y^2 = x^3 + ax + b\} \cup \{\mathcal{O}\}$$

Nociones sobre curvas elípticas

Una curva elíptica definida sobre un cuerpo finito está determinada por una ecuación de Weierstraß

$$E/\mathbb{F}_p: \ y^2 = x^3 + ax + b, \text{ donde } a, b \in \mathbb{F}_p \text{ y } 4a^3 + 27b^2 \neq 0. \quad (1)$$

Grupo de puntos:

$$E(\mathbb{F}_p) = \{(x, y) \in \mathbb{F}_p \times \mathbb{F}_p : \ y^2 = x^3 + ax + b\} \cup \{\mathcal{O}\}$$

Si k es un entero y $P \in E(\mathbb{F}_p)$, escribiremos $\overbrace{P + \cdots + P}^{(k} = k \cdot P$.

Cifrado ElGamal en $\mathcal{G} = \langle g \rangle$

Cifrado ElGamal en $\mathcal{G} = \langle g \rangle$

Dado un grupo cíclico cualquiera $\mathcal{G} = \langle g \rangle$, con notación multiplicativa, el cifrado ElGamal consiste en usar el intercambio de claves Diffie–Hellman (DH) como una parte de la cifra.

Cifrado ElGamal en $\mathcal{G} = \langle g \rangle$

Dado un grupo cíclico cualquiera $\mathcal{G} = \langle g \rangle$, con notación multiplicativa, el cifrado ElGamal consiste en usar el intercambio de claves Diffie–Hellman (DH) como una parte de la cifra.

1. Se calcula $\alpha = g^r$, donde $1 \leq r < |\mathcal{G}|$ es una clave de sesión aleatoria.

Cifrado ElGamal en $\mathcal{G} = \langle g \rangle$

Dado un grupo cíclico cualquiera $\mathcal{G} = \langle g \rangle$, con notación multiplicativa, el cifrado ElGamal consiste en usar el intercambio de claves Diffie–Hellman (DH) como una parte de la cifra.

1. Se calcula $\alpha = g^r$, donde $1 \leq r < |\mathcal{G}|$ es una clave de sesión aleatoria.
2. Se convierte el mensaje m en un elemento del grupo, $m \in \mathcal{G}$.

Cifrado ElGamal en $\mathcal{G} = \langle g \rangle$

Dado un grupo cíclico cualquiera $\mathcal{G} = \langle g \rangle$, con notación multiplicativa, el cifrado ElGamal consiste en usar el intercambio de claves Diffie–Hellman (DH) como una parte de la cifra.

1. Se calcula $\alpha = g^r$, donde $1 \leq r < |\mathcal{G}|$ es una clave de sesión aleatoria.
2. Se convierte el mensaje m en un elemento del grupo, $m \in \mathcal{G}$.
3. En $\mathcal{G}$ se calcula $\beta = m.(g^a)^r$, siendo $k = g^a$ la clave pública del receptor. La clave común DH es $k_{DH} = g^{ar} = \alpha^a$.

Cifrado ElGamal en $\mathcal{G} = \langle g \rangle$

Dado un grupo cíclico cualquiera $\mathcal{G} = \langle g \rangle$, con notación multiplicativa, el cifrado ElGamal consiste en usar el intercambio de claves Diffie–Hellman (DH) como una parte de la cifra.

1. Se calcula $\alpha = g^r$, donde $1 \leq r < |\mathcal{G}|$ es una clave de sesión aleatoria.
2. Se convierte el mensaje m en un elemento del grupo, $m \in \mathcal{G}$.
3. En $\mathcal{G}$ se calcula $\beta = m.(g^a)^r$, siendo $k = g^a$ la clave pública del receptor. La clave común DH es $k_{DH} = g^{ar} = \alpha^a$.
4. El mensaje cifrado es el par (α, β).

Cifrado ElGamal en $\mathcal{G} = \langle g \rangle$

Dado un grupo cíclico cualquiera $\mathcal{G} = \langle g \rangle$, con notación multiplicativa, el cifrado ElGamal consiste en usar el intercambio de claves Diffie–Hellman (DH) como una parte de la cifra.

1. Se calcula $\alpha = g^r$, donde $1 \leq r < |\mathcal{G}|$ es una clave de sesión aleatoria.
2. Se convierte el mensaje m en un elemento del grupo, $m \in \mathcal{G}$.
3. En $\mathcal{G}$ se calcula $\beta = m.(g^a)^r$, siendo $k = g^a$ la clave pública del receptor. La clave común DH es $k_{DH} = g^{ar} = \alpha^a$.
4. El mensaje cifrado es el par (α, β).

El receptor puede recuperar el mensaje ya que $m = \beta(\alpha^a)^{-1}$ y a es su clave privada.

ElGamal en el grupo $\mathbb{F}_p^*$

ElGamal en el grupo $\mathbb{F}_p^*$

Algoritmo (Cifrado ElGamal)

INPUT: Clave pública $pkey_U$ y mensaje a cifrar $m \in \mathbb{F}_p^*$.
OUTPUT: Cifrado formado por un par de enteros, $(\alpha, \beta) \in \mathbb{F}_p^* \times \mathbb{F}_p^*$.

1: Generar una clave de sesión $r \in_{\mathcal{R}} [1, (pkey_U.p) - 1]$;
2: $\alpha = (pkey_U.g)^r \mod p$;
3: $k_{DH} = (pkey_U.k)^r \mod p$;
$/* \ k = g^a \mod p; \ k_{DH} = g^{ar} \mod p \ */$
4: $\beta = mk_{DH} \mod p$;
5: Return (α, β);

ElGamal en el grupo $\mathbb{F}_p^*$: comentarios

ElGamal en el grupo $\mathbb{F}_p^*$: comentarios

- Pasar los mensajes al grupo es fácil: basta convertirlos en enteros en el rango $[1..p-1]$.

ElGamal en el grupo $\mathbb{F}_p^*$: comentarios

- Pasar los mensajes al grupo es fácil: basta convertirlos en enteros en el rango $[1..p-1]$.
- Las operaciones en el grupo son productos y potencias mod p, pocos pero p grande.

ElGamal en el grupo $\mathbb{F}_p^*$: comentarios

- Pasar los mensajes al grupo es fácil: basta convertirlos en enteros en el rango $[1..p-1]$.
- Las operaciones en el grupo son productos y potencias mod p, pocos pero p grande.
- La seguridad es *aproximadamente* la de un RSA cuyo módulo tenga el mismo número de bits que p.

Esquema Elíptico ElGamal: ECElGamal

Esquema Elíptico ElGamal: ECElGamal

Algoritmo (Cifrado ECElGamal)

INPUT: Clave pública $pkey_U$ y texto en claro numérico m.
OUTPUT: Cifrado formado por un par de puntos, (A, B).

1: Generar una clave de sesión $r \in_{\mathcal{R}} [1, (pkey_U.n) - 1]$;
2: $A = r \cdot (pkey_U.G)$;
3: $K_{DH} = r \cdot (pkey_U.K)$; /* $K = a \cdot G$; $K_{DH} = (ra) \cdot G$ */
4: Convertir el mensaje en punto de la curva elíptica $m \rightarrow M$;
5: $B = M + A$;
6: Return (A, B);

Esquema Elíptico ElGamal: ECElGamal

Algoritmo (Cifrado ECElGamal)

INPUT: Clave pública $pkey_U$ y texto en claro numérico m.
OUTPUT: Cifrado formado por un par de puntos, (A, B).

1: Generar una clave de sesión $r \in_{\mathcal{R}} [1, (pkey_U.n) - 1]$;
2: $A = r \cdot (pkey_U.G)$;
3: $K_{DH} = r \cdot (pkey_U.K)$; /* $K = a \cdot G$; $K_{DH} = (ra) \cdot G$ */
4: Convertir el mensaje en punto de la curva elíptica $m \rightarrow M$;
5: $B = M + A$;
6: Return (A, B);

Se recupera m calculando $M = B + a \cdot (-A)$ y, finalmente, se pasa $M \rightarrow m$.

ECElGamal: comentarios

ECElGamal: comentarios

- El criptosistema se plantea en $\langle (pkey_U.G) \rangle$, subgrupo cíclico de $E(\mathbb{F}_p)$ generado por el punto G.

ECElGamal: comentarios

- El criptosistema se plantea en $\langle (pkey_U.G) \rangle$, subgrupo cíclico de $E(\mathbb{F}_p)$ generado por el punto G.
- La suma de puntos consiste en varios productos $\mod p$.

ECElGamal: comentarios

- El criptosistema se plantea en $\langle (pkey_U.G) \rangle$, subgrupo cíclico de $E(\mathbb{F}_p)$ generado por el punto G.
- La suma de puntos consiste en varios productos $\mod p$.
- La seguridad equivalente a un RSA con módulo de 1024 bits se consigue en curvas elípticas sobre un cuerpo finito para valores de p de $\sim$160 bits.

ECElGamal: comentarios

- El criptosistema se plantea en $\langle (pkey_U.G) \rangle$, subgrupo cíclico de $E(\mathbb{F}_p)$ generado por el punto G.
- La suma de puntos consiste en varios productos $\mod p$.
- La seguridad equivalente a un RSA con módulo de 1024 bits se consigue en curvas elípticas sobre un cuerpo finito para valores de p de $\sim$160 bits.

Pero los pasos de mensaje a punto de $E(\mathbb{F}_p)$, $m \rightarrow M$ y su inverso, son problemáticos en su implementación.

Esquema ECDH+ElGamal

Esquema ECDH+ElGamal

Algoritmo (Cifrado ECDH+ElGamal)

INPUT: Clave pública $pkey_U$ y texto en claro numérico m.
OUTPUT: Punto resultante A, cifra β.

1: Generar una clave de sesión $r \in_{\mathcal{R}} [1, (pkey_U.p) - 1]$;
2: $A = r \cdot (pkey_U.G)$;
3: $K_{DH} = r \cdot (pkey_U.K)$; /* $K = a \cdot G$; $K_{DH} = (ra) \cdot G$ */
4: $\beta = mx(K_{DH}) \mod p$; /* $x(K_{DH})$ es la abscisa de K_{DH} */
5: Return (A, β)

Esquema ECDH+ElGamal

Algoritmo (Cifrado ECDH+ElGamal)

INPUT: Clave pública $pkey_U$ y texto en claro numérico m.
OUTPUT: Punto resultante A, cifra β.

1: Generar una clave de sesión $r \in_{\mathcal{R}} [1, (pkey_U.p) - 1]$;
2: $A = r \cdot (pkey_U.G)$;
3: $K_{DH} = r \cdot (pkey_U.K)$; /* $K = a \cdot G$; $K_{DH} = (ra) \cdot G$ */
4: $\beta = mx(K_{DH}) \mod p$; /* $x(K_{DH})$ es la abscisa de K_{DH} */
5: Return (A, β)

¡Recuperaremos $m \mod p$ y no m! En un sistema híbrido como el GnuPG, se puede perder la clave del cifrado simétrico.

Esquema ECDH+ElGamal

Algoritmo (Cifrado ECDH+ElGamal)

INPUT: Clave pública $pkey_U$ y texto en claro numérico m.
OUTPUT: Punto resultante A, cifra β.

1: Generar una clave de sesión $r \in_{\mathcal{R}} [1, (pkey_U.p) - 1]$;
2: $A = r \cdot (pkey_U.G)$;
3: $K_{DH} = r \cdot (pkey_U.K)$; /* $K = a \cdot G$; $K_{DH} = (ra) \cdot G$ */
4: $\beta = mx(K_{DH}) \cancel{\bmod p}$;
/* $x(K_{DH})$ es la abscisa de K_{DH} */
5: Return (A, β)

Esquema ECDH+ElGamal

Algoritmo (Cifrado ECDH+ElGamal)

INPUT: Clave pública $pkey_U$ y texto en claro numérico m.
OUTPUT: Punto resultante A, cifra β.

1: Generar una clave de sesión $r \in_{\mathcal{R}} [1, (pkey_U.p) - 1]$;
2: $A = r \cdot (pkey_U.G)$;
3: $K_{DH} = r \cdot (pkey_U.K)$; /* $K = a \cdot G$; $K_{DH} = (ra) \cdot G$ */
4: $\beta = mx(K_{DH}) \cancel{\bmod p}$;
/* $x(K_{DH})$ es la abscisa de K_{DH} */
5: Return (A, β)

Pero ahora $mx(K_{DH})$ puede ser un número demasiado pequeño, susceptible de un ataque por factorización.

Esquema alternativo ECDH+AES256

Esquema alternativo ECDH+AES256

Algoritmo (Cifrado ECDH+AES)

INPUT: Clave pública $pkey_U$ y texto en claro numérico m.
OUTPUT: Punto resultante A, cifra β.

1: Generar una clave de sesión $r \in_{\mathcal{R}} [1, (pkey_U.n) - 1]$;
2: $A = r \cdot (pkey_U.G)$;
3: $K_{DH} = r \cdot (pkey_U.K)$;
/* $K = a \cdot G$; $K_{DH} = (ra) \cdot G$ */
4: $\beta = \mathrm{aes256}(m, \{\mathrm{sha256}(x(K_{DH}))\})$;
/* $x(K_{DH})$ es la abscisa de K_{DH} */
5: Return (A, β)

ECDH + AES256: comentarios

ECDH + AES256: comentarios

- ¿Qué significa $\beta = \mathrm{aes256}(m, \{\mathrm{sha256}(x(K_{DH}))\})$?

ECDH + AES256: comentarios

- ¿Qué significa $\beta = \mathrm{aes256}(m, \{\mathrm{sha256}(x(K_{DH}))\})$?
  - Queremos cifrar m con la clave $x(K_{DH})$ con un *aes256*.

ECDH + AES256: comentarios

- ¿Qué significa $\beta = \mathrm{aes256}(m, \{\mathrm{sha256}(x(K_{DH}))\})$?
  - Queremos cifrar m con la clave $x(K_{DH})$ con un *aes256*.
  - La clave $x(K_{DH})$ ha de ser de tamaño máximo y recuperable por el receptor: $\mathrm{sha256}(x(K_{DH}))$.

ECDH + AES256: comentarios

- ¿Qué significa $\beta = \mathrm{aes256}(m, \{\mathrm{sha256}(x(K_{DH}))\})$?
  - Queremos cifrar m con la clave $x(K_{DH})$ con un *aes256*.
  - La clave $x(K_{DH})$ ha de ser de tamaño máximo y recuperable por el receptor: $\mathrm{sha256}(x(K_{DH}))$.
- Podríamos resumirlo como una operación:

$$\beta = m \otimes (x(K_{DH})) \tag{2}$$

ECDH + AES256: comentarios

- ¿Qué significa $\beta = \mathrm{aes256}(m, \{\mathrm{sha256}(x(K_{DH}))\})$?
  - Queremos cifrar m con la clave $x(K_{DH})$ con un *aes256*.
  - La clave $x(K_{DH})$ ha de ser de tamaño máximo y recuperable por el receptor: $\mathrm{sha256}(x(K_{DH}))$.
- Podríamos resumirlo como una operación:

$$\beta = m \otimes (x(K_{DH})) \tag{2}$$

- La resistencia del valor β queda garantizada por la del $\mathrm{aes256}$.

Conclusión

Conclusión

- Las debilidades aparecen según los contextos: el *ECDH+ElGamal* $\mod p$, es un algoritmo teórico admitido y difundido, pero no se puede usar en el contexto híbrido del GnuPG, en general, cuando $m > p$.

Conclusión

- Las debilidades aparecen según los contextos: el *ECDH+ElGamal* $\mod p$, es un algoritmo teórico admitido y difundido, pero no se puede usar en el contexto híbrido del GnuPG, en general, cuando $m > p$.
- Potencia del *Open Source*: Mikael Mylnikov.

Conclusión

- Las debilidades aparecen según los contextos: el *ECDH+ElGamal* $\mod p$, es un algoritmo teórico admitido y difundido, pero no se puede usar en el contexto híbrido del GnuPG, en general, cuando $m > p$.
- Potencia del *Open Source*: Mikael Mylnikov.
- GnuPG v1.4 ya soporta el esquema ECDH + AES256 de forma experimental.

Conclusión

- Las debilidades aparecen según los contextos: el *ECDH+ElGamal* mod p, es un algoritmo teórico admitido y difundido, pero no se puede usar en el contexto híbrido del GnuPG, en general, cuando $m > p$.
- Potencia del *Open Source*: Mikael Mylnikov.
- GnuPG v1.4 ya soporta el esquema ECDH + AES256 de forma experimental.
- GnuPG v2 va a implementar, nativamente desde la librería *LibGcrypt*, este esquema.