

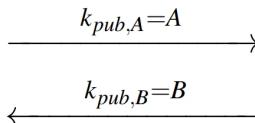
## 8.5. Esquema Elgamal

Rec. (Intercambio de llaves Diffie-Hellman DHKE)

### Diffie-Hellman Key Exchange

#### Alice

choose  $a = k_{pr,A} \in \{2, \dots, p-2\}$   
compute  $A = k_{pub,A} \equiv \alpha^a \pmod{p}$



$$k_{AB} = k_{pub,B}^{k_{pr,A}} \equiv B^a \pmod{p}$$

#### Bob

choose  $b = k_{pr,B} \in \{2, \dots, p-2\}$   
compute  $B = k_{pub,B} \equiv \alpha^b \pmod{p}$

$$k_{AB} = k_{pub,A}^{k_{pr,B}} \equiv A^b \pmod{p}$$

Elgamal se sigue casi inmediatamente de Diffie-Hellman:

Supongamos que tenemos generados  $p$  primo  
 $\alpha$  elemento primitivo

Idea: Alice usa  $k_{AB} = k_M$  como "máscara" multiplicativa en  $x$ :  
 $y = (x \cdot k_M) \pmod{p}$

### Principle of Elgamal Encryption

#### Alice

- (c) choose  $i = k_{pr,A} \in \{2, \dots, p-2\}$   
(d) compute  $k_E = k_{pub,A} \equiv \alpha^i \pmod{p}$

$\xleftarrow{\beta}$

#### Bob

- (a) choose  $d = k_{pr,B} \in \{2, \dots, p-2\}$   
(b) compute  $\beta = k_{pub,B} \equiv \alpha^d \pmod{p}$

$\xrightarrow{k_E}$

- (e) compute  $k_M \equiv \beta^i \pmod{p}$   
(g) encrypt message  $x \in \mathbb{Z}_p^*$   
 $y \equiv x \cdot k_M \pmod{p}$

$\xrightarrow{y}$

- (f) compute  $k_M \equiv k_E^d \pmod{p}$

- (h) decrypt  $x \equiv y \cdot k_M^{-1} \pmod{p}$

(a) - (f) — DHKE

(g) — Cifrado del mensaje

$$y \equiv x \cdot k_u \pmod{p}$$

(h) — Descifrado del mensaje

$$x \equiv y \cdot k_u^{-1} \pmod{p}$$

$k_E \rightarrow$  llave "efímera"       $k_u \rightarrow$  llave "máscara"

Propiedad útil de los grupos cíclicos: dada cualquier llave  $k_u \in \mathbb{Z}_p^*$ , todo mensaje  $x$  se manda a otro texto cifrado  $y$  si ambos valores se multiplican.

Más aún, si  $k_u$  se toma al azar de  $\mathbb{Z}_p^*$ , todo texto cifrado  $y \in \{1, \dots, p-1\}$  es equiprobable.

Taher Elgamal

Ingeniero Eléctrico



A PUBLIC KEY CRYPTOSYSTEM AND A SIGNATURE SCHEME BASED ON DISCRETE LOGARITHMS

TaherElGamal\*

Hewlett-Packard Labs  
1501 Page Mill Rd  
Palo Alto CA 94301

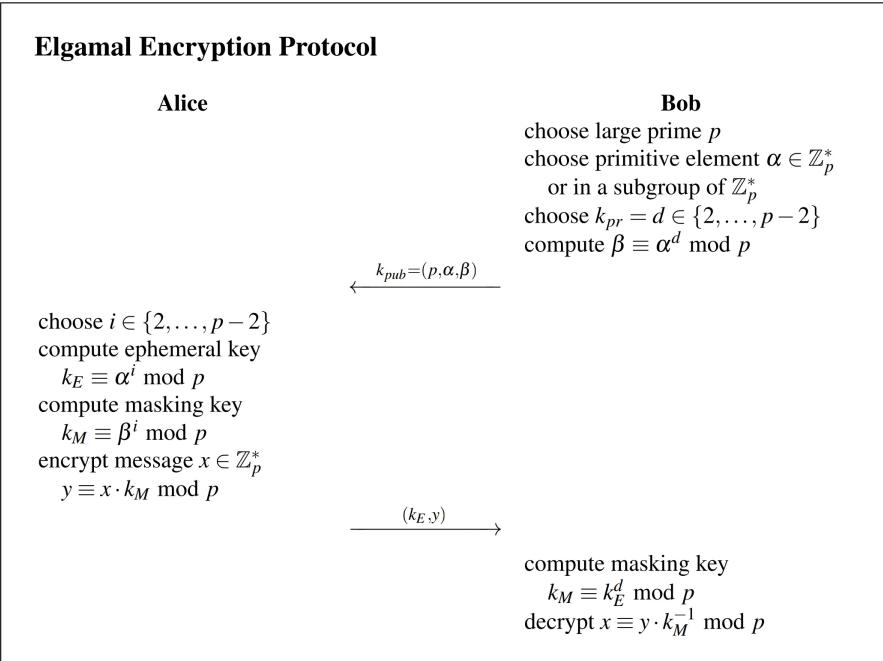
Publicado en 1985

Distinguimos tres fases:

- ① Fase de preparación → se ejecuta una vez por Bob  
↓  
emite la clave pública y recibirá el mensaje
  - ② Fase de cifrado
  - ③ Fase de descifrado
- se ejecutan cada vez que se manda un mensaje

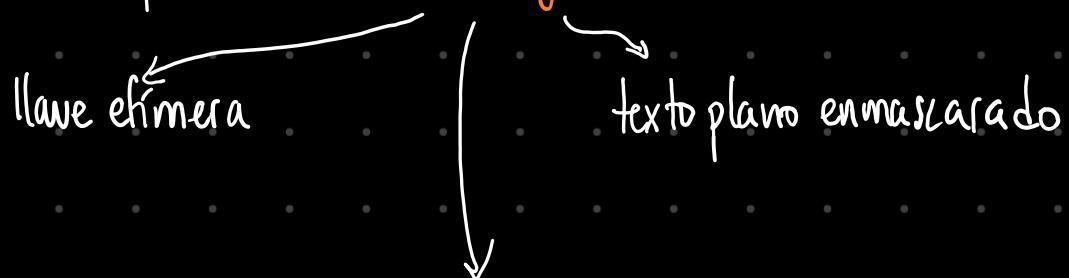
Diferencias con DH KE:

- No se necesita un tercero confiable para escoger  $p \in \mathbb{Z}$  primo y  $a \in \mathbb{Z}_p^*$ .  
↳ Bob los genera y los hace públicos



Elgamal reacomoda las operaciones del DH "ingenuo" que vimos  
↳ Alice ahora tiene que mandar un único mensaje.

Texto cifrado  $\rightarrow$  2 partes:  $(k_E, y)$



todos los parámetros tienen longitud  $\lceil \log_2 p \rceil$

el factor de expansión del mensaje de Elgamal = 2

Probemos la corrección de Elgamal:

P.D.  $d_{k_{pr}}(k_E, y)$  nos da el mensaje original  $x$ :

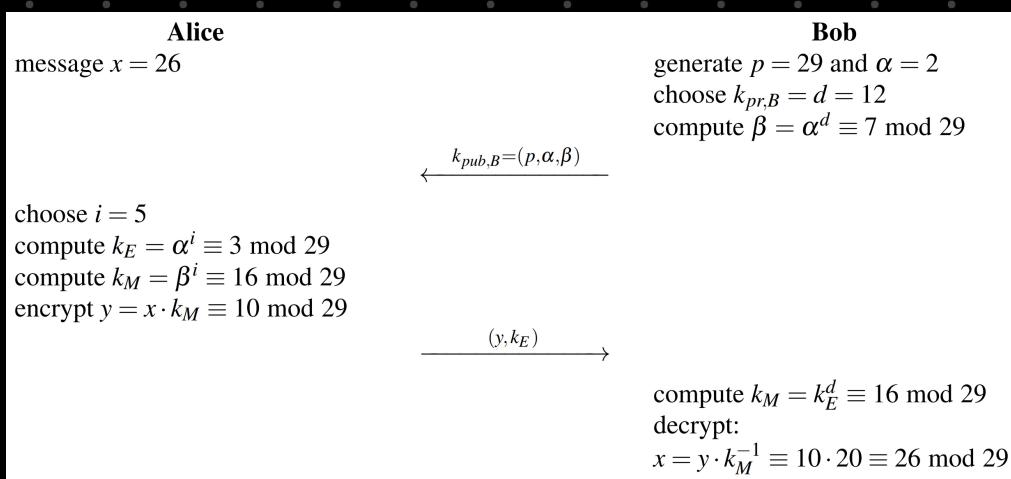
$$\begin{aligned} d_{k_{pr}}(k_E, y) &\equiv y \cdot k_u^{-1} \pmod{p} \\ &\equiv (x \cdot k_u) \cdot (k_E^d)^{-1} \pmod{p} \\ &\equiv (x \cdot (\alpha^d)^i) \cdot ((\alpha^i)^d)^{-1} \pmod{p} \\ &= x \cdot \alpha^{di - di} \pmod{p} \\ &= x \pmod{p} \end{aligned}$$

Obs. Elgamal es un cifrado probabilista: cifrar dos mensajes  $x_1 = x_2$  con la misma llave pública resulta en dos textos cifrados  $y_1 \neq y_2$  con alta probabilidad.

Se debe a que  $i$  se escoge aleatoriamente de  $\{2, 3, \dots, p-2\}$  para cada cifrado  $\Rightarrow k_M = \beta^i$  es aleatorio en  $\mathbb{Z}_p^*$

$\Rightarrow$  Fuerza bruta no funciona para  $k_M$  ni  $x$  para  $p$  suf. grande

Ejemplo: Bob genera llaves de Elgamal y Alice cifra  $x=26$ .



## Aspectos computacionales

- Generación de llaves. Se debe generar un primo  $p$  y con éste las llaves pública y privada deben generarse.

$\Rightarrow$  La seguridad de Elgamal también depende del DLP

$\Rightarrow$   $p$  debe cumplir dichas propiedades. En particular, debe tener al menos 2048 bits de longitud

- Se deben usar los algoritmos para encontrar primos

- Llave privada  $\rightarrow$  generada por algún TRNG.
- Llave pública  $\rightarrow$  necesita exponenciación
  - $\hookrightarrow$  exponenciación eficiente
  - $\hookrightarrow$  square-and-multiply alg.

• **Cifrado** Requiere dos exponenciaciones modulares y una multiplicación modular

$$k_E, k_M \downarrow$$

$$y \equiv_p x \cdot k_M$$

independientes del texto plano.

$\rightarrow$  Para mejor eficiencia, se pueden calcular previamente, almacenarse y usarse.

• **Descifrado** Esencialmente consta de una exponenciación y una inversión

Atajo: Pequeño Teorema de Fermat  $(a^p \equiv a \pmod p)$

En este caso,  $k_E^{p-1} \equiv 1 \pmod p \quad \forall k_E \in \mathbb{Z}_p^*$

$$\begin{aligned} \Rightarrow k_M^{-1} &\equiv (k_E^d)^{-1} \pmod p \\ &\equiv (k_E^d)^{-1} \cdot k_E^{p-1} \pmod p \\ &\equiv k_E^{p-d-1} \pmod p \end{aligned}$$

Esto nos permite usar sólo una exponenciación (exp.  $p-d-1$ ) para calcular la inversa y una multiplicación para recuperar  $x \equiv y \cdot k_M^{-1} \pmod p$

## Seguridad

→ Ataques pasivos

En este caso, ataque pasivo = recuperar  $x$  de  $p, \alpha, \beta = \alpha^d, k_E = \alpha^i, y = x \cdot \beta^i$  obtenidos de espionaje

Seguridad → recae en la seguridad de DHP

↳ no se conocen métodos para resolver DHP salvo calcular logaritmos discretos

Sup. Oscar tiene superpoderes y puede calcular DLPs, ent. hay dos formas de atacar Elgamal.

① Recuperar  $x$  encontrando la llave privada de Bob  $d$ :

$$d \equiv \log_{\alpha} \beta \pmod{p}$$

Esto resuelve DLP (computacionalmente imposible si se escogen bien los parámetros)

Sin embargo, si Oscar lo logra, puede desifrar el texto plano:

$$x \equiv y \cdot (k_E^d)^{-1} \pmod{p}$$

② En lugar de calcular el exponente secreto de Bob, d, Oscar puede intentar recuperar el exponente aleatorio de Alice i:

$$i \equiv \log_{\alpha} k \pmod{p}$$

$$\Rightarrow x \equiv y \cdot (\beta^i)^{-1} \pmod{p} \quad \text{Pero, otra vez, esto es resolver DLP.}$$

Para asegurar la seguridad de Elgamal sobre  $\mathbb{Z}_p^*$ , p debe tener longitud al menos 2048 bits.

Rec. Ataque de subgrupo pequeño. Para eso se usan elementos primivos, que generan subgrupos de orden primo  
↳ todo elemento es primitivo

→ Ataques activos

Ataque MitM → certificados (aseguran autenticación)

Hay otro ataque si se reusa el exponente secreto i.

Supongamos que Alice reusa i para calcular  $x_1$  y  $x_2 \Rightarrow k_{M_1} = k_{M_2} = \beta^i$   
 $\Rightarrow k_{E_1} = k_{E_2} = k_E$

Si ella manda  $(y_1, k_E)$  y  $(y_2, k_E)$  y Oscar conoce el primer mensaje, puede calcular  $k_M = y_1 x_1^{-1} \pmod{p}$ .

$$\Rightarrow x_2 \equiv y_2 k_M^{-1} \pmod{p} \quad \text{y cualquier mensaje cifrado usando } i$$

Obs. Oscar puede detectar el reuso de los exponentes secretos porque se tiene la misma llave efímera.

Otro ataque: maleabilidad

Si Oscar observa el texto cifrado  $(k_E, y)$ , lo puede reemplazar con  $(k_E, s \cdot y)$ ,  $s \in \mathbb{Z}$

$$\begin{aligned} \Rightarrow d_{k_{pr}}(d_E, sy) &\equiv s \cdot y \cdot k_{\mu}^{-1} \pmod{p} \\ &\equiv s(x k_{\mu}) k_{\mu}^{-1} \pmod{p} \\ &\equiv s \cdot x \pmod{p} \end{aligned}$$

Y esto no lleva a la misma situación de maleabilidad de RSA (Oscar no puede descifrar, pero sí manipular de forma específica)