Project Report

Jesús Javier Chi Domínguez
Oliver Fernando Cuate González

April 22, 2014

Abstract

This document contains a brief description of the Elgamal cryptosystem, a proof about its security under certain assumptions, and the study of a proposal cryptosystem based on this.

1 Elgamal

The Elgamal is an asymmetric cryptosystem, which is based on the Diffie–Hellman key exchange as follows.

1.1 Description

Let $G = \langle g \rangle$ be a cycle group, let $\mathbb{M}, \mathbb{K}, \mathbb{C} = G$ be the message space, the key space, and the ciphertext space, respectively.

Alice		*Bob*
GEN(G)		
$a \xleftarrow{\$} \mathbb{Z}_{\|G\|}$		
$p_k \leftarrow g^a$		
$s_k \leftarrow a$	$\xrightarrow{p_k}$	
ENC(p_k, m)		
		$m \leftarrow G$
		$b \xleftarrow{\$} \mathbb{Z}_{\|G\|}$
		$c_1 \leftarrow m \cdot p_k^b$
		$c_2 \leftarrow g^b$
	$\xleftarrow{c}$	$c \leftarrow (c_1, c_2)$
DEC(s_k, c)		
$g' \leftarrow c_2^{s_k}$		
$m \leftarrow c_1 \cdot (g')^{-1}$		

1.2 Security Analysis

For the Elgamal Security, we need the following assumptions: Let $\pi = (\mathrm{GEN}, \mathrm{ENC}, \mathrm{DEC})$ be a public key encryption scheme.

CPA-EXP$_\pi^{\mathcal{A}}(n)$

$(p_k, s_k) \leftarrow \mathrm{GEN}(n)$
Give p_k to $\mathcal{A}$
$\mathcal{A}$ returns two messages (m_0, m_1) s.t. $|m_1| = |m_0|$
$b \xleftarrow{\$} \{0, 1\}$
$c \leftarrow \mathrm{ENC}(p_k, m_b)$.
Give c to $\mathcal{A}$. Ultimately $\mathcal{A}$ outputs a bit b'
if $b = b'$ **then**
  outputs 1
else
  outputs 0
end if

We say that π is CPA secure if for all efficient adversary $\mathcal{A}$.

$$|\Pr[\text{CPA–EXP}_{\pi}^{\mathcal{A}}(n) \Rightarrow 1] - \frac{1}{2}| \leq neg(n)$$

Let D be an adversary that takes as input triples of group elements, and outputs a bit. We define the DDH advantage of D on G to be

$$\mathbf{Adv}_{G}^{DDH}(D) \quad = \quad |\Pr[x, y \xleftarrow{\$} \mathbb{Z}_{|G|} : D(g^x, g^y, g^{xy}) \Rightarrow 1] - \Pr[x, y, z \xleftarrow{\$} \mathbb{Z}_{|G|} : D(g^x, g^y, g^z) \Rightarrow 1]|$$

The DDH assumption for G is the assumption that for any adversary D, $\mathbf{Adv}_{G}^{DDH}(D)$ is negligible.

Claim 1.1 *Elgamal is CPA secure under the Desicional Diffie-Hellman (DDH) assumption.*

Proof. Let $\mathcal{A}$ be an arbitrary adversary. Denoting Elgamal by π, define the following games

---

GAME$_0$

$x \xleftarrow{\$} \mathbb{Z}_{|G|}$
$\alpha \leftarrow g^x$
Give α to $\mathcal{A}$
$\mathcal{A}$ returns two messages (m_0, m_1) s.t. $|m_1| = |m_0|$
$b \xleftarrow{\$} \{0, 1\}$
$y \xleftarrow{\$} \mathbb{Z}_{|G|}$
$\beta \leftarrow g^y$
$\delta \leftarrow \alpha^y$
$c \leftarrow \delta \cdot m_b$
Give (c, β) to $\mathcal{A}$. Ultimately $\mathcal{A}$ outputs a bit b'
if $b = b'$ **then**
  outputs 1
else
  outputs 0
end if

GAME$_1$

$x \xleftarrow{\$} \mathbb{Z}_{|G|}$
$\alpha \leftarrow g^x$
Give α to $\mathcal{A}$
$\mathcal{A}$ returns two messages (m_0, m_1) s.t. $|m_1| = |m_0|$
$b \xleftarrow{\$} \{0, 1\}$
$y \xleftarrow{\$} \mathbb{Z}_{|G|}$
$\beta \leftarrow g^y$
$\boxed{z \xleftarrow{\$} \mathbb{Z}_{|G|},\ \delta \leftarrow \alpha^z}$
$c \leftarrow \delta \cdot m_b$
Give (c, β) to $\mathcal{A}$. Ultimately $\mathcal{A}$ outputs a bit b'
if $b = b'$ **then**
  outputs 1
else
  outputs 0
end if

Clearly: $\Pr[\text{CPA–EXP}_{\pi}^{\mathcal{A}}(|G|) \Rightarrow 1] = \Pr[\text{GAME}_0 \Rightarrow 1]$. For other hand, if we consider that the GAME$_1$ is actually a random algorithm then we have:

$$\Pr[\text{GAME}_1 \Rightarrow 1] = \frac{1}{2}$$

Claim 1.2 $|\Pr[\text{GAME}_0 \Rightarrow 1] - \Pr[\text{GAME}_1 \Rightarrow 1]| = \mathbf{Adv}_G^{DDH}(D)$, *for some efficient adversary* D.

Hence:

$$\mathbf{Adv}_\pi^{CPA}(\mathcal{A}) \quad = \quad |\Pr[\text{CPA–EXP}_\pi^{\mathcal{A}}(n) \Rightarrow 1] - \frac{1}{2}| = \mathbf{Adv}_G^{DDH}(D)$$

□

Proof. Considere the following adevrsary D.

$\mathbf{D}(\alpha, \beta, \delta)$
Give α to $\mathcal{A}$
$\mathcal{A}$ returns two messages (m_0, m_1) s.t. $\lvert m_1\rvert = \lvert m_0\rvert$
$b \xleftarrow{\$} \{0, 1\}$
$c \leftarrow \delta \cdot m_b$
Give (c, β) to $\mathcal{A}$. Ultimately $\mathcal{A}$ outputs a bit b'
if $b = b'$ **then**
outputs 1
else
outputs 0
end if

We should note that

$$\Pr[x, y \xleftarrow{\$} \mathbb{Z}_{|G|} : \mathbf{D}(g^x, g^y, g^{xy}) \Rightarrow 1] = \Pr[\text{GAME}_0 \Rightarrow 1]$$
$$\Pr[x, y, z \xleftarrow{\$} \mathbb{Z}_{|G|} : \mathbf{D}(g^x, g^y, g^z) \Rightarrow 1] = \Pr[\text{GAME}_1 \Rightarrow 1]$$

Thus, $\mathbf{Adv}_\pi^{CPA}(\mathcal{A}) = |\Pr[\text{GAME}_0 \Rightarrow 1] - \Pr[\text{GAME}_1 \Rightarrow 1]| = \mathbf{Adv}_G^{DDH}(D)$.

2 Proposed Cryptosystem

Basing on the Elgamal, we construct the following public key encryption scheme:

2.1 Description

Let $G = \langle g \rangle$ be a cycle group, let $\mathbb{M}, \mathbb{K}, \mathbb{C} = G$ be the message space, the key space, and the ciphertext space, respectively. Let $H\colon \mathbb{X} \to G$ be a hash function where $G \subseteq \mathbb{X}$. Now, consider the following scheme.

Scheme$_1$		
Alice		*Bob*
$\text{GEN}_{\text{Sch}}(G)$		
$a \xleftarrow{\$} \mathbb{Z}_{|G|}$		
$p_k \leftarrow g^a$		
$s_k \leftarrow a$	$\xrightarrow{p_k}$	
$\text{ENC}_{\text{Sch}}(p_k, m)$		
		$m \leftarrow G$
		$b \xleftarrow{\$} \mathbb{Z}_{|G|}$
		$m' \leftarrow m \oplus H((p_k)^b)$
		$c' \leftarrow m' \cdot (p_k)^b$
		$c'' \leftarrow g^b$
	$\xleftarrow{c}$	$c \leftarrow (c', c'')$
$\text{DEC}_{\text{Sch}}(s_k, c)$		
$g' \leftarrow (c'')^{s_k}$		
$m' \leftarrow c' \cdot (g')^{-1}$		
$m \leftarrow m' \oplus H(g')$		

For compute $H((p_k)^b)$, the value $(p_k)^b$ is viewed as an element of $\mathbb{X}$ and not as in G.

Now, the scheme that we propose is the following.

Alice		*Bob*
$\text{GEN}(G)$		
$(p_k^0, s_k^0) \leftarrow \text{GEN}_{\text{Sch}}(G)$ $(p_k^1, s_k^1) \leftarrow \text{GEN}_{\text{Sch}}(G)$ $p_k \leftarrow (p_k^0, p_k^1)$ $s_k \leftarrow (s_k^0, s_k^1)$	$\xrightarrow{p_k}$	
$\text{ENC}(p_k, m)$		
	$\xleftarrow{c}$	$m \leftarrow G$ $m_0 \| m_1 \leftarrow m$ $c_0 \leftarrow \text{ENC}_{\text{Sch}}(p_k^0, m_0)$ $c_1 \leftarrow \text{ENC}_{\text{Sch}}(p_k^1, m_1)$ $c \leftarrow (c_0, c_1)$
$\text{DEC}(s_k, c)$		
$m_0 \leftarrow \text{DEC}_{\text{Sch}}(s_k^0, c_0)$ $m_1 \leftarrow \text{DEC}_{\text{Sch}}(s_k^1, c_1)$ $m \leftarrow m_0 \| m_1$		

2.2 Security Analysis

Claim 2.1 *If the hash function H is **preimage resistant**, then scheme* **Scheme**$_1$ *is CPA secure under the DDH assumption.*

Remembering, given a hash function H and $y \in G$, **Preimage** is to find $x \in G' >> G$ s.t. $y = H(x)$. If there exist no efficient algorithm to solve Preimage then we say that H is *preimage resistant*. In other words:

$$\mathbf{Adv}_H^{PR}(\mathcal{A}) = \Pr[\mathcal{A} \text{ solves Primage}] \leq neg(n) = \epsilon_{PR}$$

Proof. Let $\mathcal{B}$ be an arbitrary adversary for **Scheme**$_1$. Consider the following adversary $\mathcal{A}$ for Elgamal. $\mathcal{A}$ will act as the challenger for $\mathcal{B}$.

$\mathcal{A}(\alpha, \beta, \delta)$

- Give α to $\mathcal{B}$
- $\mathcal{B}$ returns two messages (m_0, m_1) s.t. $|m_1| = |m_0|$
- $b \xleftarrow{\$} \{0, 1\}$
- $c \leftarrow \delta \cdot (m_b \oplus H(\delta))$
- $d \leftarrow \delta \cdot m_b$
- Give (d, β) to $\mathcal{B}$. Ultimately $\mathcal{B}$ outputs a bit b'
- **if** $b = b'$ **then**
  - outputs 0
- **else**
  - outputs 1
- **end if**

Now consider the following game:

GAME$_2$

$x \xleftarrow{\$} \mathbb{Z}_{|G|}$
$\alpha \leftarrow g^x$
Give α to $\mathcal{A}$
$\mathcal{A}$ returns two messages (m_0, m_1) s.t. $|m_1| = |m_0|$
$b \xleftarrow{\$} \{0, 1\}$
$y \xleftarrow{\$} \mathbb{Z}_{|G|}$
$\beta \leftarrow g^y$
$\delta \leftarrow \alpha^y$
$c \leftarrow \delta \cdot (m_b \oplus H(\delta))$
Give (c, β) to $\mathcal{A}$. Ultimately $\mathcal{A}$ outputs a bit b'
if $b = b'$ **then**
  outputs 0
else
  outputs 1
end if

As the above game uses an Elgamal Adversary $\mathcal{A}$, then the output bit b' will be the same **Scheme**$_1$ if and only if the value of the Hash function $H(\delta)$ is equal to zero, it is true beacuse the bit b' is such that $(m_{b'} \oplus H(\delta)) \cdot \delta = c$ (the ciphertext), then b' will be equal to the bit b if and only and if only if $H(\delta)$. Assuming that a reasonable adversary for our scheme will consider this fact and denoting **Scheme**$_1$ by π_σ, then we have:

$$\Pr[\text{CPA–EXP}^{\mathcal{B}}_{\pi_\sigma}(|G|) \Rightarrow 1] = \Pr[\text{GAME}_2 \Rightarrow 0]$$

but because

$$\Pr[\text{GAME}_2 \Rightarrow 0] = 1 - \Pr[\text{GAME}_2 \Rightarrow 1]$$

We obtain that

$$\Pr[\text{GAME}_2 \Rightarrow 1] = 1 - \Pr[\text{CPA–EXP}^{\mathcal{B}}_{\pi_\sigma}(|G|) \Rightarrow 1]$$

Also of GAME$_2$ we need to define the next game to calculate $\mathbf{Adv}^{CPA}_{\pi_\sigma}(\mathcal{B})$.

GAME$_3$

$x \xleftarrow{\$} \mathbb{Z}_{|G|}$
$\alpha \leftarrow g^x$
Give α to $\mathcal{A}$
$\mathcal{B}$ returns two messages (m_0, m_1) s.t. $|m_1| = |m_0|$
$b \xleftarrow{\$} \{0, 1\}$
$y \xleftarrow{\$} \mathbb{Z}_{|G|}$, $z \xleftarrow{\$} \mathbb{Z}_{|G|}$
$\beta \leftarrow g^y$
$\delta \leftarrow \alpha^y$, $\delta' \leftarrow x^z$
$c \leftarrow \delta \cdot (m_b \oplus H(\delta))$
$c' \leftarrow \delta' \cdot (m_b \oplus H(\delta))$
Give (c', β) to $\mathcal{A}$. Ultimately $\mathcal{A}$ outputs a bit b'
if $b = b'$ **then**
  outputs 0
else
  outputs 1
end if

We should note that

$$\Pr[x, y \xleftarrow{\$} \mathbb{Z}_{|G|} : \mathcal{B}(g^x, g^y, g^{xy}) \Rightarrow 1] = \Pr[\text{GAME}_2 \Rightarrow 1]$$

$$\Pr[x, y, z \xleftarrow{\$} \mathbb{Z}_{|G|} : \mathcal{B}(g^x, g^y, g^z) \Rightarrow 1] = \Pr[\text{GAME}_3 \Rightarrow 1]$$

Anew we have that GAME_3 is a random game, but in this case $\Pr[b' = 1] = \dfrac{1}{2}$ (that is the same that $\text{GAME}_0 \Rightarrow 1$ for the Elgamal adversary $\mathcal{A}$), must be multiply to ϵ_{pr} to obtain $\Pr[b = 1]$, it is considered the Hash function H and the previous explanation. Then:

$$\Pr[\text{GAME}_3 \Rightarrow 1] = \epsilon_{pr}\Pr[\text{GAME}_1 \Rightarrow 1] = \frac{1}{2}\epsilon_{pr}$$

With a similar analysis for GAME_2, then we have:

$$\begin{aligned}
\mathbf{Adv}_{\pi_\sigma}^{CPA}(\mathcal{B}) &= |\Pr[\text{GAME}_2 \Rightarrow 1] - \Pr[\text{GAME}_3 \Rightarrow 1]| \\
&= |\epsilon_{pr}\Pr[\text{GAME}_0 \Rightarrow 1] - \epsilon_{pr}\Pr[\text{GAME}_1 \Rightarrow 1]| \\
&= \epsilon_{pr}|(\Pr[\text{GAME}_0 \Rightarrow 1] - \Pr[\text{GAME}_1 \Rightarrow 1]| \\
\Rightarrow \mathbf{Adv}_{\pi_\sigma}^{CPA}(\mathcal{B}) &= \epsilon_{pr}\mathbf{Adv}_{\pi}^{CPA}(\mathcal{A})
\end{aligned}$$

This analysis must be done for each of two parts of a message, the encryption of these parts can be consider as independent, then finally for our scheme, denoting by σ, we have:

$$\mathbf{Adv}_{\sigma}^{CPA}(\mathcal{B}) = \frac{1}{2}\mathbf{Adv}_{\pi_\sigma}^{CPA}(\mathcal{B})$$

□

2.3 Implementation

Preliminaries

We will make the implementation of our scheme in a subgroup of the quadratic residual of $\mathbb{Z}_p$ for some large prime p. Thus, let $p = 2 \cdot k \cdot q + 1$ be a prime such that q is a "large" prime and k is a positive integer. Let $\langle g \rangle$ be a subgroup of $\mathbb{Z}_p^*$, this implies that $\text{ord}(g) = 2 \cdot k \cdot q$.

Let $\mathbf{QR}$ be the subgroup of quadratic residual of $\mathbb{Z}_p$, then $|\mathbf{QR}| = k \cdot q$, and because $g' = g^2$ is such that g' is a quatratic residual, and $\text{ord}(g') = k \cdot q$, and every subgroup of a cyclic goup is also a cyclic group, g' is a generator of $\mathbf{QR}$.

Let $g'' = h^k$, and $G'' = \langle g'' \rangle$ be a subgroup of $\mathbf{QR}$. We will work in this subgroup G''. Now, if we find a genereator g of $\mathbb{Z}_p^*$, then we can construct a generator $g'' = g^{2 \cdot k}$ of G''.

To find this generator g, we do the following algorithm

Test Generator(g, P)

r $\leftarrow P - 1$
$\prod_{i=1}^{l}(p_i)^{e_i} \leftarrow$ r
for $i \leftarrow 1$ to l **do**
  $r_i \leftarrow$ (r / p_i)
  if $g^{r_i} \equiv 1 \bmod P$ **then**
    return **Failure**
  end if
end for
return **Successful**

Numerical results

All the implementation was done in *C*-language using the *gmp* library of big numbers. So, it is needed to install this library to running the code. The adjunct file *scheme.c* needs two file texts, one file in which the message is (we are supposing that the message is an element of $\mathbb{Z}_p$) and other in which the prime p and q and the factorizarion of k are. This last is required because we need to compute a generator of G''.

The prime used in the example are given in [JG85]. To run the code, you should do the following in the terminal (console):

```
[tsugumi@localhost código]$ gcc scheme.c -o scheme -lgmp
[tsugumi@localhost código]$ ./scheme
Enter the name of file where the possible prime is:
primes.txt

The primes.txt file contains the public prime q:
7645817649953398726194923102564833517
And 2*k*q + 1 is equal to:
7918324333004779287780879909121159911537551977796076554305607309994905870203

Enter the name of file where the message is:
message.txt

*********EXAMPLE OF ELGAMAL*********
value of g:
1726993547190075963798373354827674627436448746281027425618008826361562074495
value of mod:
7918324333004779287780879909121159911537551977796076554305607309994905870203
value of pk:
5505090336430564427388920363806970741945647053894480487188757870174301720849
value of sk:
2558659747784382241965366288334503830087074210813859191761606497144564205604

value of msg0:
352015466596088420260883280075658662319625787846437566
value of c0:
6716121961554031368891947069052319260261465679141620712573926541270118984559
value of c1:
5321297254104382293744813292137135628374804638327042185461085805699329293868
value of m0:
352015466596088420260883280075658662319625787846437566

value of msg1:
47773109869245232364730066609837018108561065242031153677
value of c0:
3681640246460377932210743121764805544069444552770777339538297169447287524760
value of c1:
4202226701769224115020768327908212218962912078475863171620295839257573102139
value of m1:
47773109869245232364730066609837018108561065242031153677

*********EXAMPLE OF OUR CRYPTOSYSTEM*********
value of msg0:
352015466596088420260883280075658662319625787846437566
value of c0:
939173188911047500625938452141966068207987587314569753300341924307700755514
value of c1:
5976489338807791056430901469234924942440524293974334298966215667539266302630
value of m0:
352015466596088420260883280075658662319625787846437566

value of msg1:
47773109869245232364730066609837018108561065242031153677
value of c0:
7349637287685330765618319669264622242216083617689448798033821399439984658756
value of c1:
3079188805011434152741699300800319341375794344780060337168720670438411705563
value of m0:
47773109869245232364730066609837018108561065242031153677
```

References

[BB09] Timo Bartkewitz and Ruhr-University Bochum. Building hash functions from block ciphers, their security and implementation properties. 2009.

[htt24] GMP 6.0.0 https://gmplib.org/. The gnu multiple precision arithmetic library, 2014-03-24.

[JG85] Cybermation Ltd John Gordon. Strong primes arc easy to find. *Advances in Cryptology - EUROCRYPT '84, LNCS 209, pp. 216-223, 1985. Springer-Verlag Berlin Heidelberg*, 1985.

[KL07] Jonathan Katz and Yehuda Lindell. *Introduction to modern cryptography.* Boca Raton : Chapman and Hall/CRC., 2007.

[Sho06] Victor Shoup. Sequences of games: A tool for taming complexity in security proofs. *Computer Science Dept. NYU.*, 2006.

[Wan97] Thomas Wang. Integer hash function. 1997.