Diffie-Hellman Problem and Elgamal Encryption Scheme

Yu Zhang

HIT/CST/NIS

Cryptography, Spring, 2012

Outline

1. Cyclic Groups and Discrete Logrithms
2. Diffie-Hellman Assumptions and Applications
3. The Elgamal Encryption Scheme

Content

Cyclic Groups and Generators

$\mathbb{G}$ is a finite group and $g \in \mathbb{G}$, the set

$$\langle g \rangle \stackrel{\text{def}}{=} \{g^0, g^1, \ldots, \} = \{g^0, g^1, \ldots, g^{i-1}\}.$$

- The **order** of g is the smallest positive integer i with $g^i = 1$.
- $\mathbb{G}$ is a **cyclic group** if $\exists\ g$ has order $m = |\mathbb{G}|$. $\langle g \rangle = \mathbb{G}$, g is a **generator** of $\mathbb{G}$.
- $\langle g \rangle$ is a subgroup of $\mathbb{G}$, and $|\langle g \rangle| \mid |\mathbb{G}|$.
- If $\mathbb{G}$ is a group of prime order p, then $\mathbb{G}$ is cyclic. All elements of $\mathbb{G}$ except the identity are generators of $\mathbb{G}$.

Theorem 1

If p is prime, then $\mathbb{Z}_p^*$ is cyclic.

Examples of Cyclic Groups

- $\mathbb{Z}_{15}$ with '$+$' is cyclic and the '1' is a generator.
- $\mathbb{Z}_7$ with '$+$' is cyclic and all non-zero elements are generators.
- For $\mathbb{Z}_7^*$, 2 is not a generator, but 3 is.

$\mathbb{G}$ is a cyclic group of order n, and g is a generator of $\mathbb{G}$. Then the mapping $f : \mathbb{Z}_n \to \mathbb{G}$ given by $f(a) = g^a$ is an isomorphism. For $a, a' \in \mathbb{Z}_n$,

$$f(a + a') = g^{[a+a' \bmod n]} = g^{a+a'} = g^a \cdot g^{a'} = f(a) \cdot f(a').$$

All cyclic groups of the same order are "the same" in an algebraic sense, but this is not true in a computational sense.

Discrete Logarithm

If $\mathbb{G}$ is a cyclic group of order q, then $\exists$ a generator $g \in \mathbb{G}$ such that $\{g^0, g^1, \ldots, g^{q-1}\} = \mathbb{G}$.

- $\forall h \in \mathbb{G}$, $\exists$ a unique $x \in \mathbb{Z}_q$ such that $g^x = h$.
- $x = \log_g h$ is the **discrete logarithm of h with respect to g**.
- If $g^{x'} = h$, then $\log_g h = [x' \bmod q]$.
- $\log_g 1 = 0$ and $\log_g(h_1 \cdot h_2) = [(\log_g h_1 + \log_g h_2) \bmod q]$.

The Discrete Logarithm Assumption

The discrete logarithm experiment $\mathsf{DLog}_{\mathcal{A},\mathcal{G}}(n)$:

1. Run a group-generating algorithm $\mathcal{G}(1^n)$ to obtain $(\mathbb{G}, q, g)$, where $\mathbb{G}$ is a cyclic group of order q (with $\|q\| = n$), and g is a generator of $\mathbb{G}$.
2. Choose $h \leftarrow \mathbb{G}$. ($x' \leftarrow \mathbb{Z}_q$ and $h := g^{x'}$)
3. $\mathcal{A}$ is given $\mathbb{G}, q, g, h$, and outputs $x \in \mathbb{Z}_q$.
4. $\mathsf{DLog}_{\mathcal{A},\mathcal{G}}(n) = 1$ if $g^x = h$, and 0 otherwise.

Definition 2

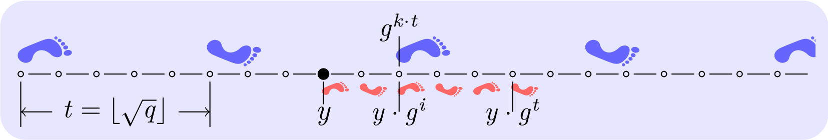
The discrete logarithm problem is hard relative to $\mathcal{G}$ if $\forall$ PPT algorithm $\mathcal{A}$, $\exists$ negl such that

$$\Pr[\mathsf{DLog}_{\mathcal{A},\mathcal{G}}(n) = 1] \leq \mathsf{negl}(n).$$

Overview of Discrete Logarithm Algorithms

- Given a generator $g \in \mathbb{G}$ and $y \in \langle g \rangle$, find x such that $g^x = y$.
- **Brute force**: $\mathcal{O}(q)$, $q = \text{ord}(g)$ is the order of $\langle g \rangle$.
- **Baby-step/giant-step** method [Shanks]: $\mathcal{O}(\sqrt{q} \cdot \text{polylog}(q))$.
- **Pohlig-Hellman** algorithm: when q has small factors.
- **Index calculus** method: $\mathcal{O}(\exp(\sqrt{n \cdot \log n}))$.
- The best-known algorithm is the **general number field sieve** with time $\mathcal{O}(\exp(n^{1/3} \cdot (\log n)^{2/3}))$.
- Elliptic curve groups vs. $\mathbb{Z}_p^*$: more efficient for the honest parties, but that are equally hard for an adversary to break. (Both 1024-bit $\mathbb{Z}_p^*$ and 132-bit elliptic curve need 2^{66} steps.)

The Baby-Step/Giant-Step Algorithm



Algorithm 1: The baby-step/giant-step algorithm

input : $g \in \mathbb{G}$ and $y \in \langle g \rangle$; $q = \mathrm{ord}(g)$ ($t := \lfloor \sqrt{q} \rfloor$)

output: $\log_g y$

```
for i = 0 to ⌊q/t⌋ do compute g_i := g^(i·t)   /* giant steps */
sort the pairs (i, g_i) by g_i
for i = 0 to t do
    compute y_i := y · g^i                      /* baby steps */
    if y_i = g_k for some k then return [kt − i mod q]
```

The time complexity is $\mathcal{O}(\sqrt{q} \cdot \mathrm{polylog}(q))$.

Example of Baby-Step/Giant-Step Algorithm

In $\mathbb{Z}_{29}^*$, $q = 28$, $g = 2$, $y = 17$.

$t = 5$, compute the giant steps:

$$2^0 = 1,\ 2^5 = 3,\ 2^{10} = 9,\ 2^{15} = 27,\ 2^{20} = 23,\ 2^{25} = 11.$$

compute the baby steps:

$$17 \cdot 2^0 = 17,\ 17 \cdot 2^1 = 5,\ 17 \cdot 2^2 = 10,$$

$$17 \cdot 2^3 = 20,\ 17 \cdot 2^4 = 11,\ 17 \cdot 2^5 = 22.$$

$2^{25} = 11 = 17 \cdot 2^4$. So $\log_2 17 = 25 - 4 = 21$.

The Pohlig-Hellman Algorithm

Idea: when q is known and has small factors, reduces the discrete logarithm instance to multiple instances in groups of smaller order.

According to CRT: If $q = \prod_{i=1}^{k} q_i$ and $\forall i \neq j, \gcd(q_i, q_j) = 1$, then

$$\mathbb{Z}_q \simeq \mathbb{Z}_{q_1} \times \cdots \times \mathbb{Z}_{q_k} \text{ and } \mathbb{Z}_q^* \simeq \mathbb{Z}_{q_1}^* \times \cdots \times \mathbb{Z}_{q_k}^*$$

$$(g_i)^x \stackrel{\text{def}}{=} \left(g^{q/q_i}\right)^x = (g^x)^{q/q_i} = y^{q/q_i} \text{ for } i = 1, \ldots, k.$$

We have k instances in k smaller groups, $\mathrm{ord}(g_i) = q_i$. [1]
Use any other algorithm to solve $\log_{g_i}(y^{q/q_i})$.
Answers are $\{x_i\}_{i=1}^{k}$ for which $g_i^{x_i} \equiv y^{q/q_i} \equiv g_i^x$.
$\forall i,\ x \equiv x_i \pmod{q_i}$. $x \bmod q$ is uniquely determined (CRT).
The time complexity is $\mathcal{O}(\max_i\{\sqrt{q_i}\} \cdot \mathrm{polylog}(q))$.

[1] If $p \mid q$, then $\mathrm{ord}(g^p) = q/p$.

Example of Pohlig-Hellman Algorithm

In $\mathbb{Z}_{31}^*$, $q = 30 = 5 \cdot 3 \cdot 2$, $g = 3$, $y = 26 = g^x$.

$$
\begin{aligned}
(g^{30/5})^x = y^{30/5} &\Longrightarrow (3^6)^x = 26^6 \Longrightarrow 16^x \equiv 1 \\
(g^{30/3})^x = y^{30/3} &\Longrightarrow (3^{10})^x = 26^{10} \Longrightarrow 25^x \equiv 5 \\
(g^{30/2})^x = y^{30/2} &\Longrightarrow (3^{15})^x = 26^{15} \Longrightarrow 30^x \equiv 30
\end{aligned}
$$

$$x \equiv 0 \pmod 5,\ x \equiv 2 \pmod 3, x \equiv 1 \pmod 2,$$

so $x \equiv 5 \pmod{30}$.

The Index Calculus Method

Idea: find a relatively small factor base and build a system of ℓ linear equations related to g; find a linear equation related to y; solve $\ell + 1$ linear equations to give $\log_g y$.

1. for $\mathbb{Z}_p^*$, choose a base $B = \{p_1, \ldots, p_k\}$ of prime numbers.
2. find $\ell \geq k$ distinct $x_1, \ldots, x_\ell$ for which $[g^{x_i} \bmod p]$ decompose into the elements of B: $g^{x_i} \equiv \prod_{j=1}^{k} p_j^{e_j} \pmod{p}$.
3. ℓ equations: $x_i = \sum_{j=1}^{k} e_{i,j} \cdot \log_g(p_j) \pmod{p-1}$.
4. find x^* for which $[g^{x^*} \cdot y \bmod p]$ can be factored.
5. new equation: $x^* + \log_g y = \sum_{j=1}^{k} e_j^* \cdot \log_g(p_j) \pmod{p-1}$.
6. Use linear algebra to solve equations and give $\log_g y$.

The time complexity is identical to that of the quadratic sieve.

Example of Index Calculus Method

$p = 101$, $g = 3$ and $y = 87$. $B = \{2, 5, 13\}$.

$3^{10} \equiv 65 \pmod{101}$ and $65 = 5 \cdot 13$. Similarly, $3^{12} \equiv 80 = 2^4 \cdot 5 \pmod{101}$ and $3^{14} \equiv 13 \pmod{101}$. The linear equations:

$$\begin{aligned} x_1 &= 10 \equiv \log_3 5 + \log_3 13 \pmod{100} \\ x_2 &= 12 \equiv 4 \cdot \log_3 2 + \log_3 5 \pmod{100} \\ x_3 &= 14 \equiv \log_3 13 \pmod{100}. \end{aligned}$$

We also have $x^* = 5$, $3^5 \cdot 87 \equiv 32 \equiv 2^5 \pmod{101}$, or

$$5 + \log_3 87 \equiv 5 \cdot \log_3 2 \quad \pmod{100}.$$

Adding the 2nd and 3rd equations and subtracting the 1st, we derive $4 \cdot \log_3 2 \equiv 16 \pmod{100}$. So $\log_3 2$ is 4, 29, 54, or 79. Trying all shows that $\log_3 2 = 29$. The last equation gives $\log_3 87 = 40$.

Content

Diffie-Hellman Assumptions

- **Computational Diffie-Hellman (CDH)** problem:

$$\mathsf{DH}_g(h_1, h_2) \stackrel{\mathsf{def}}{=} g^{\log_g h_1 \cdot \log_g h_2}$$

- **Decisional Diffie-Hellman (DDH)** problem:
  Distinguish $\mathsf{DH}_g(h_1, h_2)$ from a random group element h'.

Definition 3

DDH problem is hard relative to $\mathcal{G}$ if $\forall$ PPT $\mathcal{A}$, $\exists$ negl such that

$$|\Pr[\mathcal{A}(\mathbb{G}, q, g, g^x, g^y, g^z) = 1] - \Pr[\mathcal{A}(\mathbb{G}, q, g, g^x, g^y, g^{xy}) = 1]|$$

$$\leq \mathsf{negl}(n).$$

Intractability of DL, CDH and DDH

DDH is easier than CDH and DL.

Why Prime-Order Groups

- The discrete logarithm problem is hardest in such groups.
- Finding a generator in such groups is trivial.
- Any non-zero exponent will be invertible modulo the order.
- A necessary condition for the DDH problem to be hard is that $\mathsf{DH}_g(h_1, h_2)$ by itself should be indistinguishable from a random group element. This is (almost) true for such groups:

$$\Pr[\mathsf{DH}_g(g^{x_1}, g^{x_2}) = 1] = 1 - \left(1 - \frac{1}{q}\right)^2 = \frac{2}{q} - \frac{1}{q^2},$$

and $\forall y \neq 1:$

$$\Pr[\mathsf{DH}_g(g^{x_1}, g^{x_2}) = y] = \frac{1}{q} \cdot \left(1 - \frac{1}{q}\right) = \frac{1}{q} - \frac{1}{q^2}.$$

Working in (Subgroups of) $\mathbb{Z}_p^*$

- $y \in \mathbb{Z}_p^*$ is a **quadratic residue modulo** p if $\exists x \in \mathbb{Z}_p^*$ such that $x^2 \equiv y \pmod{p}$.
- The set of quadratic residues modulo p is a subgroup with $(p-1)/2 = q$ elements, since $x^2 \equiv (p-x)^2 \pmod{p}$.
- p is a **strong prime** if $p = 2q + 1$ with q prime.

Algorithm 2: A group generation algorithm $\mathcal{G}$

---

input : Security parameter 1^n

output: Cyclic group $\mathbb{G}$, its order q, and a generator g

generate a random $(n+1)$-bit strong prime p
$q := (p-1)/2$
choose an arbitrary $x \in \mathbb{Z}_p^*$ with $x \neq \pm 1 \bmod p$
$g := x^2 \bmod p$
return p, q, g

Recall Secure Key-Exchange Experiment

The key-exchange experiment $\mathsf{KE}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n)$:

1. Two parties holding 1^n execute protocol Π. Π results in a **transcript** trans containing all the messages sent by the parties, and a **key** k that is output by each of the parties.
2. A random bit $b \leftarrow \{0,1\}$ is chosen. If $b = 0$ then choose $\hat{k} \leftarrow \{0,1\}^n$ *u.a.r*, and if $b = 1$ then set $\hat{k} := k$.
3. $\mathcal{A}$ is given trans and $\hat{k}$, and outputs a bit b'.
4. $\mathsf{KE}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n) = 1$ if $b' = b$, and 0 otherwise.

Definition 4

A key-exchange protocol Π is secure in the presence of an eavesdropper if $\forall$ PPT $\mathcal{A}$, $\exists$ negl such that

$$\Pr[\mathsf{KE}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n) = 1] < \frac{1}{2} + \mathsf{negl}(n).$$

Diffie-Hellman Key-Exchange Protocol

$(\mathbb{G}, q, g) \leftarrow \mathcal{G}$

$x \leftarrow \mathbb{Z}_q$

$h_1 := g^x$ $\xrightarrow{\mathbb{G}, q, g, h_1}$

$y \leftarrow \mathbb{Z}_q$

$\xleftarrow{h_2}$ $h_2 := g^y$

$k_A := h_2^x$ $\qquad$ $k_B := h_1^y$

$$k_A = k_B = k = g^{xy}.$$

$\widehat{\mathsf{KE}}^{\mathsf{eav}}_{\mathcal{A},\Pi}$ denote an experiment where if $b = 0$ the adversary is given $\hat{k} \leftarrow \mathbb{G}$.

Theorem 5

If DDH problem is hard relative to $\mathcal{G}$, then DH key-exchange protocol Π is secure in the presence of an eavesdropper (with respect to the modified experiment $\widehat{\mathsf{KE}}^{\mathsf{eav}}_{\mathcal{A},\Pi}$).

Security

Insecurity against active adversaries (Man-In-The-Middle).

Proof of Security in DH Key-Exchange Protocol

Proof.

$$\Pr\left[\widehat{\mathsf{KE}}^{\mathsf{eav}}_{\mathcal{A},\Pi} = 1\right]$$

$$= \frac{1}{2} \cdot \Pr\left[\widehat{\mathsf{KE}}^{\mathsf{eav}}_{\mathcal{A},\Pi} = 1 | b = 1\right] + \frac{1}{2} \cdot \Pr\left[\widehat{\mathsf{KE}}^{\mathsf{eav}}_{\mathcal{A},\Pi} = 1 | b = 0\right]$$

If $b = 1$, then give true key; otherwise give random g^z.

$$\begin{aligned}
&= \frac{1}{2} \cdot \Pr\left[\mathcal{A}(g^x, g^y, g^{xy}) = 1\right] + \frac{1}{2} \cdot \Pr\left[\mathcal{A}(g^x, g^y, g^z) = 0\right] \\
&= \frac{1}{2} \cdot \Pr\left[\mathcal{A}(g^x, g^y, g^{xy}) = 1\right] + \frac{1}{2} \cdot \left(1 - \Pr\left[\mathcal{A}(g^x, g^y, g^z) = 1\right]\right) \\
&= \frac{1}{2} + \frac{1}{2} \cdot \left(\Pr\left[\mathcal{A}(g^x, g^y, g^{xy}) = 1\right] - \Pr\left[\mathcal{A}(g^x, g^y, g^z) = 1\right]\right) \\
&\leq \frac{1}{2} + \frac{1}{2} \cdot \left|\Pr\left[\mathcal{A}(g^x, g^y, g^{xy}) = 1\right] - \Pr\left[\mathcal{A}(g^x, g^y, g^z) = 1\right]\right|
\end{aligned}$$

$\square$

Constructing Collision-Resistant Hash Functions

Construction 6

Define a fixed-length hash function (Gen, H)*:*

- Gen*: on input* 1^n*, run* $\mathcal{G}(1^n)$ *to obtain* $(\mathbb{G}, q, g)$ *and then select* $h \leftarrow \mathbb{G}$*. Output* $s := \langle \mathbb{G}, q, g, h \rangle$ *as the key.*
- H*: given a key* $s = \langle \mathbb{G}, q, g, h \rangle$ *and input* $(x_1, x_2) \in \mathbb{Z}_q \times \mathbb{Z}_q$*, output* $H^s(x_1, x_2) := g^{x_1} h^{x_2}$*.*

Theorem 7

If the discrete logarithm problem is hard relative to $\mathcal{G}$*, then Construction is a fixed-length CRHF.*

Proof of Security of Construction

Proof.

$\mathcal{A}'$ uses $\mathcal{A}$ to solve the discrete logarithm problem:

1. $\mathcal{A}'$ is given input $s = \langle \mathbb{G}, q, g, h \rangle$.
2. Run $\mathcal{A}(s)$ and obtain output x, x'.
3. If $x \neq x' \wedge H^s(x) = H^s(x')$ then:
   - If $h = 1$ return 0;
   - Otherwise, parse x as (x_1, x_2) and x' as (x_1', x_2'). Return $[(x_1 - x_1') \cdot (x_2' - x_2)^{-1} \bmod q]$.

$$H^s(x_1, x_2) = H^s(x_1', x_2') \implies g^{x_1} h^{x_2} = g^{x_1'} h^{x_2'}$$

$$\implies g^{x_1 - x_1'} = h^{x_2' - x_2}$$

$$\implies \log_g h = [(x_1 - x_1') \cdot (x_2' - x_2)^{-1} \bmod q].$$

□

Content

Lemma on Perfectly-secret Private-key Encryption

Lemma 8

$\mathbb{G}$ is a finite group and $m \in \mathbb{G}$ is an arbitrary element. Then choosing random $g \leftarrow \mathbb{G}$ and setting $g' := m \cdot g$ gives the same distribution for g' as choosing random $g' \leftarrow \mathbb{G}$. I.e, $\forall \hat{g} \in \mathbb{G}$:

$$\Pr[m \cdot g = \hat{g}] = 1/|\mathbb{G}|.$$

Proof.

Let $\hat{g} \in \mathbb{G}$ be arbitrary, then

$$\Pr[m \cdot g = \hat{g}] = \Pr[g = m^{-1} \cdot \hat{g}].$$

Since g is chosen *u.a.r*, the probability that g is equal to the fixed element $m^{-1} \cdot \hat{g}$ is exactly $1/|\mathbb{G}|$. □

The Elgamal Encryption Scheme

An algorithm $\mathcal{G}$, on input 1^n, outputs a description of a cyclic group $\mathbb{G}$, its order q (with $\|q\| = n$), and a generator g.

Construction 9

- Gen*: on input* 1^n *run* $\mathcal{G}(1^n)$ *to obtain* $(\mathbb{G}, q, g)$*. Choose a random* $x \leftarrow \mathbb{Z}_q$ *and compute* $h := g^x$*.* $pk = \langle \mathbb{G}, q, g, h \rangle$ *and* $sk = \langle \mathbb{G}, q, g, x \rangle$*.*
- Enc*: on input* pk *and* $m \in \mathbb{G}$*, choose a random* $y \leftarrow \mathbb{Z}_q$ *and output* $\langle c_1, c_2 \rangle = \langle g^y, h^y \cdot m \rangle$*.*
- Dec*: on input* sk *and* $\langle c_1, c_2 \rangle$*, output* $m := c_2 / c_1^x$*.*

Theorem 10

If the DDH problem is hard relative to $\mathcal{G}$*, then the Elgamal encryption scheme is CPA-secure.*

Example of Elgamal Encryption

$q = 83$, $p = 2q + 1 = 167$, $g = 2^2 = 4 \pmod{167}$, $\hat{m} = 011101$

The receiver chooses secrete key $37 \in \mathbb{Z}_{83}$.
The public key is $pk = \langle 167, 83, 4, [4^{37} \bmod 167] = 76 \rangle$.
$\hat{m} = 011101 = 29$, $m = [(29 + 1)^2 \bmod 167] = 65$.
Choose $y = 71$, the ciphertext is
$\langle [4^{71} \bmod 167], [76^{71} \cdot 65 \bmod 167] \rangle = \langle 132, 44 \rangle$.

Decryption: $m = [44 \cdot (132^{37})^{-1}] \equiv [44 \cdot 66] \equiv 65 \pmod{167}$.
65 has the two square roots 30 and 137, and $30 < q$, so $\hat{m} = 29$.

Proof of Security of Elgamal Encryption Scheme

Proof.

Idea: Prove that Π is secure in the presence of an eavesdropper by reducing an algorithm D for DDH problem to the eavesdropper $\mathcal{A}$.

Modify Π to $\tilde{\Pi}$: the encryption is done by choosing random $y \leftarrow \mathbb{Z}_q$ and $z \leftarrow \mathbb{Z}_q$ and outputting the ciphertext:

$$\langle g^y, g^z \cdot m \rangle.$$

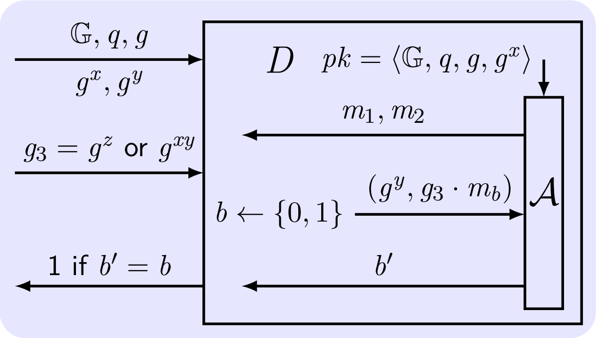
- $\tilde{\Pi}$ is not an encryption scheme.
- g^y is independent of m.
- $g^z \cdot m$ is a random element independent of m (Lemma 8).

$$\Pr\left[\mathsf{PubK}^{\mathsf{eav}}_{\mathcal{A},\tilde{\Pi}}(n) = 1\right] = \frac{1}{2}.$$

$\square$

Proof (Cont.)

D receives $(\mathbb{G}, q, g, g^x, g^y, g_3)$ where g_3 equals either g^{xy} or g^z, for random x, y, z:



Proof (Cont.)

Case I: $g_3 = g^z$, ciphertext is $\langle g^y, g^z \cdot m_b \rangle$.

$$\Pr[D(g^x, g^y, g^z) = 1] = \Pr\left[\mathsf{PubK}^{\mathsf{eav}}_{\mathcal{A},\tilde{\Pi}}(n) = 1\right] = \frac{1}{2}.$$

Case II: $g_3 = g^{xy}$, ciphertext is $\langle g^y, g^{xy} \cdot m_b \rangle$.

$$\Pr[D(g^x, g^y, g^{xy}) = 1] = \Pr\left[\mathsf{PubK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n) = 1\right] = \varepsilon(n).$$

Since the DDH problem is hard,

$$\mathsf{negl}(n) \geq |\Pr[D(g^x, g^y, g^z) = 1] - \Pr[D(g^x, g^y, g^{xy}) = 1]|$$

$$= |\frac{1}{2} - \varepsilon(n)|.$$

CCA in Elgamal Encryption

Constructing the ciphertext of the message $m \cdot m'$.

Given $pk = \langle g, h \rangle$, $c = \langle c_1, c_2 \rangle$, $c_1 = g^y$, $c_2 = h^y \cdot m$,
Method I: compute $c_2' := c_2 \cdot m'$, and $c' = \langle c_1, c_2' \rangle$.

$$\frac{c_2'}{c_1^x} = \frac{h^y \cdot m \cdot m'}{g^{xy}} = \frac{g^{xy} \cdot m \cdot m'}{g^{xy}} = m \cdot m'.$$

Method II: compute $c_1'' := c_1 \cdot g^{y''}$, and $c_2'' := c_2 \cdot h^{y''} \cdot m'$.

$$c_1'' = g^y \cdot g^{y''} = g^{y+y''} \text{ and } c_2'' = h^y m \cdot h^{y''} m' = h^{y+y''} m m'$$

so $c'' = \langle c_1'', c_2'' \rangle$ is an encryption of $m \cdot m'$.

Elgamal Implementation Issues

- **Encoding binary strings**: depends on the particular type of group.
  - the subgroup of quadratic residues modulo a strong prime $p = (2q + 1)$.
  - a string $\hat{m} \in \{0, 1\}^{n-1}$, $n = \|q\|$.
  - map $\hat{m}$ to the plaintext $m = [(\hat{m} + 1)^2 \bmod p]$.
  - The mapping is one-to-one and efficiently invertible.
- **Sharing public parameters**: $\mathcal{G}$ generates parameters $\mathbb{G}$, q, g.
  - generated "once-and-for-all".
  - used by multiple receivers.
  - each receiver must choose their own secrete values x and publish their own public key containing $h = g^x$.

Parameter sharing

In the case of Elgamal, the public parameters can be shared, but in the case of RSA, parameters cannot be shared.

Summary

- cyclic group, discrete log., baby-step/giant-step
- CDH, DDH, DHKE protocol.
- Elgamal encryption, sharing public parameters.