

EECS 475: Introduction to Cryptography Notes

Noah Peters

April 5, 2023

Abstract

Lecture notes for EECS 475 at the University of Michigan. L^AT_EXtemplate by Pingbang Hu.

Contents

4	Number Theory and Public Key Cryptography	2
4.1	Modular Arithmetic and Euclid's Algorithm	2
4.2	Group Theoretic View of Numbers	3
4.3	Fast Exponentiation	5
4.4	Diffie-Hellman	5
4.5	Modeling Public Key Encryption	6
4.6	El Gamal Cryptosystem	8
A	Additional Proofs	11
A.1	Proof of ??	11

Chapter 4

Number Theory and Public Key Cryptography

Lecture 19: Number Theory

4.1 Modular Arithmetic and Euclid's Algorithm

20 Mar. 10:30

We define the set of integers, $\mathbb{Z} = \{\dots, -2, -1, 0, 1, 2, \dots\}$, and natural numbers, $\mathbb{N} = \mathbb{Z}^+ = \{1, 2, 3, \dots\}$.

Theorem 4.1.1 (Product of primes). Every integer $N > 1$ can be written *uniquely* as a product of (power of) primes.

Lemma 4.1.1 (Division with remainder). Let $a \in \mathbb{Z}, b \in \mathbb{Z}^+$. \exists unique integers q, r such that $a = q.b + r$ where $0 \leq r < b$, and they can be efficiently computed in *polynomial time* relative to the *bit length*: i.e. $\log_2 a + \log_2 b + O(1)$

With the ability to perform division in polynomial time, we are able to find the **greatest common divisor** of two integers a, b :

Definition 4.1.1 (Greatest common divisor). Let $a, b \in \mathbb{Z}^+$. Then, there exists $x, y \in \mathbb{Z}$ such that $\gcd(a, b) = a.x + b.y$. Further, $\gcd(a, b)$ is the *smallest* positive integer that can be written this way.

We claim there is an efficient algorithm, the **Extended Euclidean Algorithm**, that computes not only $\gcd(a, b)$, but also the x, y coefficients above:

Algorithm 4.1: Extended Euclidean Algorithm

```
Data:  $a, b \in \mathbb{Z}$  where  $a \geq b > 0$ 
1 if  $b|a$  then
2   return  $x = 0, y = 1$ 
3 else
4   write  $a = q.b + r$  where  $0 < r < b$ 
5    $x', y' \leftarrow \text{ExtEuclid}(b, r)$ 
6 return  $x = y', y = x' - y'.q$ 
```

Verification of Extended Euclidean Algorithm. Assuming the recursive call is successful (by induction, we can), we will get back x', y' that are the gcd of b and r :

$$\begin{aligned} b.x' + r.y' &= \gcd(b, r) = \gcd(a, b) \\ b.x' + (a - q.b).y' &= a.y' + b(x' - q.y') \end{aligned}$$

We note that $y' = x$ and $x' - q.y' = y$ and so we are done. ■

We claim that the **Extended Euclidean Algorithm** makes $O(n)$ recursive calls.

4.2 Group Theoretic View of Numbers

Definition 4.2.1. Let $\mathbb{Z}_N = \{0, 1, 2, \dots, N-1\}$, indicating the set of all possible remainders of division by N , further: $\mathbb{Z}_N^* = \{x \in \mathbb{Z}_N : \gcd(x, N) = 1\}$.

Remark. For example, $\mathbb{Z}_6^* = \{1, 5\}$. Alternatively, $\mathbb{Z}_7^* = \{1, 2, 3, 4, 5, 6\}$, showing how $\mathbb{Z}_p^* = \mathbb{Z}_p \setminus \{0\}$, where p is any prime number.

Lecture 20: Group Theory

We continue our review of modular arithmetic. Modular addition, subtraction, and multiplication are all closed in modular arithmetic. We have the property $a \equiv b \pmod{N} \Leftrightarrow N \mid a - b$ and further, if $a \equiv a' \pmod{N}$ and $b \equiv b' \pmod{N}$, we have:

22 Mar. 10:30

- $a + b \equiv a' + b' \pmod{N}$
- $a - b \equiv a' - b' \pmod{N}$
- $a \cdot b \equiv a' \cdot b' \pmod{N}$

However, division is not always possible:

$$3 \cdot 2 \equiv 15 \cdot 2 \pmod{24} \not\equiv 3 \equiv 15 \pmod{24}$$

Definition 4.2.2 (Invertability). An integer b is **invertible** if $\exists c$ where $b \cdot c \equiv 1 \pmod{N}$.

Lemma 4.2.1. If $b \geq 1, N > 1$, b is invertible mod $N \Leftrightarrow \gcd(b, N) = 1$.

Proof. Let $b \cdot c \equiv 1 \pmod{N}$.

$$\begin{aligned} &\Rightarrow b \cdot c - 1 = N \cdot q \\ &\Rightarrow b \cdot c - N \cdot q = 1. \end{aligned}$$

Thus, $\gcd(b, N) = 1$ since \gcd is the smallest positive integer that is expressible in this way. ■

Using the [Extended Euclidean Algorithm](#), we can compute inverse mod N efficiently.

Definition 4.2.3 (Group). (G, \circ) , where $\circ : G \times G \rightarrow G$ and $\circ(g, h)$ is denoted as $g \circ h$, is a **group** if it satisfies the properties of:

- identity: $\exists e \in G$ such that $\forall g \in G: e \circ g = g \circ e = g$
- invertability: $\forall g \in G, \exists g^{-1}$ (or $-g$) such that $g \circ g^{-1} = e$
- associativity: $\forall g_1, g_2, g_3 \in G: (g_1 \circ g_2) \circ g_3 = g_1 \circ (g_2 \circ g_3)$
- (abelian groups): $\forall g, h \in G, g \circ h = h \circ g$ (i.e commutativity)

Example. For $(\mathbb{Z}_N, + \pmod{N})$ we have:

- identity: $a + 0 \equiv 0 + a \equiv a \pmod{N}$
- invertability: $a + (-a) \equiv 0 \pmod{N}$
- associativity: $a + (b + c) \equiv (a + b) + c \pmod{N}$

Example. For $(\mathbb{Z}_N^*, \cdot \bmod N)$ we have:

- identity: $a \cdot 1 \equiv 1 \cdot a \equiv a \bmod N$
- invertability: $a \cdot (a^{-1}) \equiv 1 \bmod N$
- associativity: $a \cdot (b \cdot c) = (a \cdot b) \cdot c \bmod N$

The size of a group, called the **group order**, is denoted as $|G|$. For example, $(\mathbb{Z}_N, +)$ has order N , and \mathbb{Z}_p^*, \cdot has order $p - 1$

Theorem 4.2.1. For a group, G , where $m = |G|$, we have that $\forall g \in G: g^m = 1$ (where $g^m = ((g \circ g) \circ g) \circ g \dots$).

Proof. For simplicity, assume G is abelian and suppose $G = \{g_1, g_2, \dots, g_m\}$ and let $g \in G$ be arbitrary. Note that

$$g \circ g_i = g \circ g_j \Rightarrow g_i = g_j$$

and so the set $\{g \circ g_i : i \in \{1, m\}\}$ covers all elements of G exactly once. Therefore:

$$g_1 \circ g_m = g^m(g_1 \circ \dots \circ g_m) \Rightarrow 1 = g^m.$$

■

Corollary 4.2.1 (Fermat's Little Theorem). \forall prime p , $\gcd(a, p) = 1 \Rightarrow a^{p-1} \equiv 1 \bmod p$

Theorem 4.2.2 (Euler's Theorem). For $\Phi(N) = |\{a | 1 \leq a \leq N, \gcd(a, N) = 1\}|$, $|\mathbb{Z}_N^*| = \Phi(N)$ we have:

$$\text{if } \gcd(g, N) = 1 \Rightarrow g^{\Phi(N)} \equiv 1 \bmod N$$

Corollary 4.2.2. For $m = |G| > 1, e \in \mathbb{Z}, \gcd(e, m) = 1$, define $d = e^{-1} \bmod m$ and function $f_e: G \rightarrow G$ as $f_e(g) = g^e$. Then, f_e is a bijection whose inverse is f_d .

Definition 4.2.4 (Cyclic Groups). A group, G , is considered **cyclic** if $\exists g \in G$ such that $G = \{1 = g^0, g^1, g^2, \dots, g^{m-1}\}$. If this is the case, we say that g generates G .

Example. For $\mathbb{Z}_7^* = \{1, \dots, 6\}$:

$$\text{powers of 3: } \{3^0 \equiv 1, 3^1 \equiv 3, 3^2 \equiv 2, 3^3 \equiv 6, 3^4 \equiv 4, 3^5 \equiv 5\}$$

$$\text{powers of 2: } \{2^0 \equiv 1, 2^1 \equiv 2, 2^2 \equiv 4, 2^3 \equiv 1, 2^4 \equiv 2, \dots\}$$

and so 3 is a generator of \mathbb{Z}_7^* , but 2 is not.

If there is no $g \in G$ that generates G , then G is not cyclic. Furthermore, when g does not generate G , it generates a **subgroup**:

Definition 4.2.5. $G' \subseteq G$, is a subgroup if (G', \circ) is a group.

Theorem 4.2.3 (Lagrange's Theorem). If $G' \subseteq G$ is a subgroup, then $|G'|$ divides $|G|$.

4.3 Fast Exponentiation

Suppose we have an element g and we want to compute g^M . Instead of naively multiplying g M times, we can observe that if $M = 2^m$:

$$g^M = g^{2^m} = g^{2^{m-1}} \cdot g^{2^{m-1}} = (g^{2^{m-1}})^2.$$

So for any M we can rewrite it as

$$M = \sum_{i=0}^l m_i \cdot 2^i,$$

allowing us to calculate g^m :

$$g^m = \prod_{i=0}^l g^{2^i},$$

where each g^{2^i} can be calculated using the above trick. This results in $O(l^2)$ multiplications altogether if $|M| = l$.

Corollary 4.3.1. Fast exponentiation allows us to compute inverses efficiently because $g^{-1} = g^{|G|-1}$ since $g^{|G|} = 1$.

We've shown we can compute g^m efficiently given g and m . But, can we efficiently compute m from g^m ? It's unknown, but conjectured to be extremely difficult to do so.

Lecture 21: Diffie-Hellman Key Exchange, DDH Assumption, Public Key Encryption and CPA Security

4.4 Diffie-Hellman

27 Mar. 10:30

The key exchange problem occurs when two individuals seek to communicate over an insecure channel. The canonical story has *Alice* and *Bob* attempting to communicate over a channel being *passively monitored* by an eavesdropper, *Eve*.

Definition 4.4.1 (Diffie-Hellman Protocol).

1. let Alice fix a large **cyclic group** G of known order q . ($|q| \approx$ security parameter).
2. Alice discovers a generator g for $G = \mathbf{Z}_{q+1}^*$ for $q+1$ prime. In other words, Alice finds a number g that enumerates all elements of the group G when raised to the powers $\{0, 1, \dots, q-1\}$.
3. Alice chooses random $a \leftarrow \mathbb{Z}_q$, let $\mathcal{A} = g^a \in G$.
4. Alice sends $\mathcal{A} = g^a, g$ to Bob over the insecure channel. Eve has access to this information.
5. Bob receives the message and chooses a random $b \leftarrow \mathbb{Z}_q$ and sends $\mathcal{B} = g^b \in G$ back to Alice.
6. Alice and Bob both calculate $\mathcal{K} = (g^a)^b = (g^b)^a = g^{ab}$ as their shared secret key.

An eavesdropper will have to take *discrete log* to break this scheme (to find a or b). Formally, the security of this scheme is based on **DDH: Decisional Diffie-Hellman Assumption**.

Definition 4.4.2 (Decisional Diffie-Hellman Assumption (DDH)). DDH holds for a group $G = \langle g \rangle$ (i.e. G is generated by g) if

$$(g, g^a, g^b, g^{ab}) \in G^4$$

is indistinguishable (for $a, b \leftarrow \mathbb{Z}_q, q = |G|$) from

$$(g, g^a, g^b, g^c), \text{ where } c \leftarrow \mathbf{Z}_q$$

Remark. Key derivation can simply be an algorithm for turning a group element into a random bit-string.

4.5 Modeling Public Key Encryption

We want to have a protocol (Gen, Enc, Dec) that can *directly* handle public key encryption. We can model this analogously to EAV/CPA security, just in a public key setting:

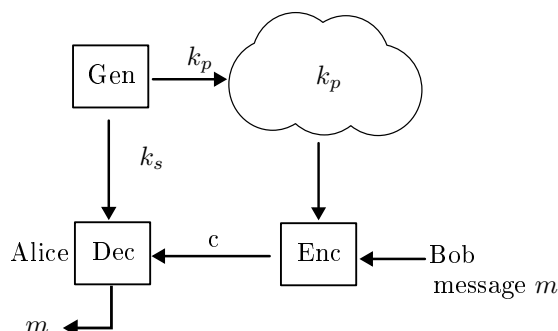


Figure 4.1: Analog of EAV/CPA security in the context of public keys.

Definition 4.5.1 (Public Key CPA (EAV) Secure Scheme). For our **CPA secure public key scheme** we have:

- $\text{Gen}(1^n)$: outputs (k_p, k_s)
- $\text{Enc}(k_p, m \in M)$: outputs ciphertext c
- $\text{Dec}(k_s, c)$: outputs $m \in M$ (or fail " \perp ")

We analyze its correctness. For $(k_p, k_s) \leftarrow \text{Gen}(1^n)$, we always have $\forall m \in M$:

$$\text{Dec}(k_s, \text{Enc}(k_p, m)) = m$$

Remark. Note that for the first time, our Gen function outputs two *related* random keys, not just one private random string.

Just like previous instances of security, we can define public key CPA security in the context of a game. Here we have adversary \mathcal{A} against $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$. Note that unlike the original CPA game, in this scenario the adversary only prompts the oracle once since the *adversary can encrypt messages of their own*.

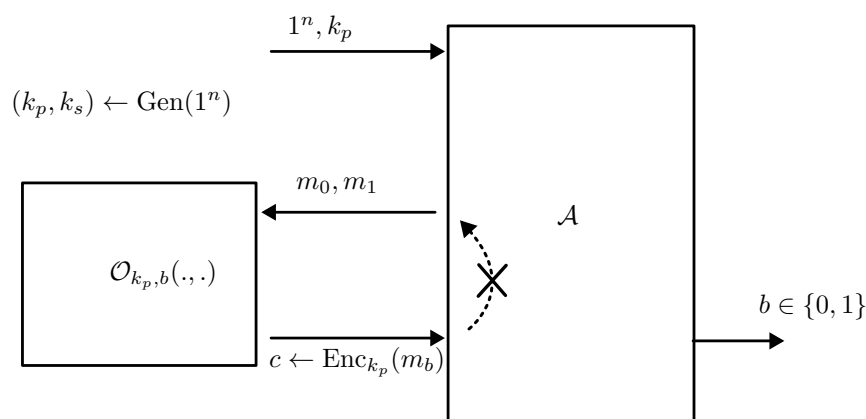


Figure 4.2: The public key CPA game.

Definition 4.5.2 (Public CPA Security). A public key encryption scheme Π as defined above is secure if \forall p.p.t. \mathcal{A} :

$$\text{Adv}_{\Pi}(\mathcal{A}) := \left| \Pr_{(k_p, k_s) \leftarrow \text{Gen}(1^n)} (\mathcal{A}^{\mathcal{O}_{k_p, 0}(\cdot, \cdot)}(k_p) = 1) - \Pr_{(k_p, k_s) \leftarrow \text{Gen}(1^n)} (\mathcal{A}^{\mathcal{O}_{k_p, 1}(\cdot, \cdot)}(k_p) = 1) \right| = \text{negl}(n).$$

Remark. The number of queries to the LR oracle, \mathcal{O} doesn't matter. I.e. if we have EAV-security allowing \mathcal{A} to query the oracle once, then we have security even if multiple queries are allowed.

Proof. To show that *one-query public CPA security* \Rightarrow *many query public CPA security* we are essentially showing that

$$\text{public EAV security} \Rightarrow \text{public CPA security}.$$

We prove this by proposing a new adversary, \mathcal{A} , that is capable of breaking a many-query CPA scheme, and then using this adversary to break the single-query EAV scheme.

Imagine a many-query attacker, \mathcal{A} that makes up to $q : \text{poly}(n)$ queries. Consider the following worlds:

- (i) *Hybrid 0*: All queries answered by $c \leftarrow \text{Enc}_{k_p}(m_0)$.
- (ii) *Hybrid 1*: First query (m_0, m_1) answered by $c \leftarrow \text{Enc}_{k_p}(m_1)$, then $\text{Enc}_{k_p}(m_0)$ thereafter.
- (iii) *Hybrid 2*: First **two** queries are answered by $c \leftarrow \text{Enc}_{k_p}(m_1)$, then $\text{Enc}_{k_p}(m_0)$ thereafter.
- (iv) *Hybrid q* : All queries answered by $c \leftarrow \text{Enc}_{k_p}(m_1)$

These hybrid worlds reflect the *left* and *right* worlds as follows:

- (i) **Left world**: equivalent to *Hyb₀*: all queries to the LR oracle are answered by $c \leftarrow \text{Enc}_{k_p}(m_0)$.
- (ii) **Right world**: equivalent to *Hyb_q*: all queries answered by $c \leftarrow \text{Enc}_{k_p}(m_1)$.

We can use the idea that if an adversary, \mathcal{A} can distinguish between the left and right worlds, they must be able to distinguish between the i th hybrid world and the $i + 1$ th hybrid world where the i th hybrid world is defined as

$$\text{HybridWorld}_i = \text{First } i \text{ queries of } (m_0, m_1) \text{ answered by } c \leftarrow \text{Enc}_{k_p}(m_1), \\ \text{then } c \leftarrow \text{Enc}_{k_p}(m_0) \text{ thereafter.}$$

We first note that the difference between *Hyb _{$i-1$}* and *Hyb _{i}* is only in how the i th query is answered. We can build a "simulator" $\mathcal{S}_i^{\mathcal{O}}(k_p)$ that gets *one query* and simulates either *Hyb _{$i-1$}* or *Hyb _{i}* depending on b .

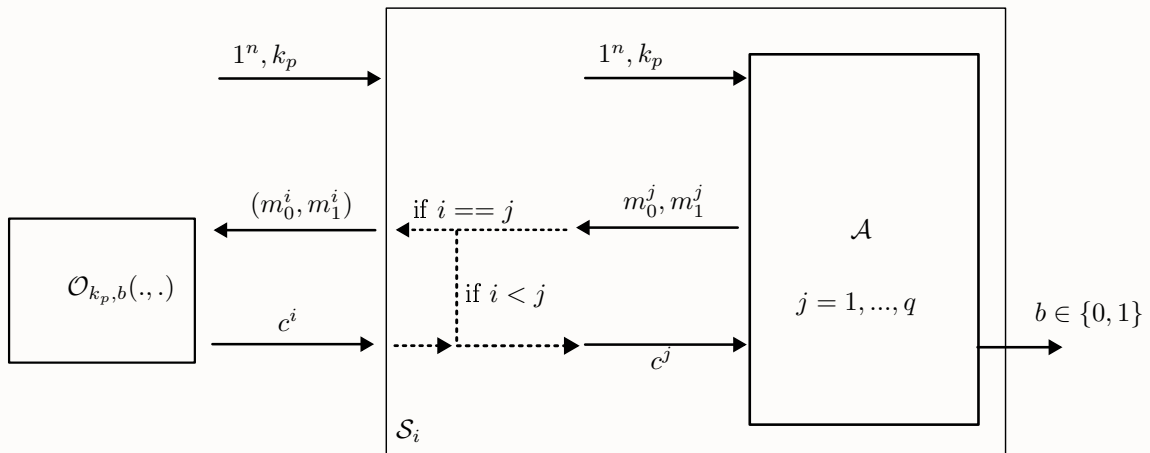


Figure 4.3: A simulator that wins pEAV game using an adversary that wins the pCPA game.

Upon the j^{th} query of \mathcal{A} (m_0^j, m_1^j):

- if $j < i$, S_i runs $c^j \leftarrow \text{Enc}_{k_p}(m_1^j)$.
- if $j > i$, S_i runs $c^j \leftarrow \text{Enc}_{k_p}(m_0^j)$.
- if $j = i$, S_i queries its LR oracle and gives the result to \mathcal{A} .

Consider the two cases where S_i is in the worlds:

- Left world* ($b = 0$): then we are simulating Hyb_{i-1} .
- Right world* ($b = 1$): then we are simulating Hyb_i .

By the **triangle inequality**,

$$\begin{aligned} \text{Adv}_{\Pi}^{\text{CPA}} &= |\Pr(\mathcal{A} = 1 \text{ in } \text{Hyb}_0) - \Pr(\mathcal{A} = 1 \text{ in } \text{Hyb}_q)| \\ &= |\Pr(\mathcal{A} = 1 \text{ in } \text{Hyb}_0) - \Pr(\mathcal{A} = 1 \text{ in } \text{Hyb}_1) + \\ &\quad \Pr(\mathcal{A} = 1 \text{ in } \text{Hyb}_1) - \Pr(\mathcal{A} = 1 \text{ in } \text{Hyb}_2) + \\ &\quad \dots - \Pr(\mathcal{A} = 1 \text{ in } \text{Hyb}_q)| \\ &\leq \sum_{i=1}^q \text{Adv}_{\Pi}^{\text{Single CPA}}(\mathcal{S}_i) = q \cdot \text{negl}(n) = \text{poly}(n) \cdot \text{negl}(n) = \text{negl}(n). \end{aligned}$$

Thus, by assuming we had single-query (pEAV) security we have concluded we also have multi-query (pCPA) security. If we didn't have multi-query security, you could use a multi-query attacker to make a single-query attacker (contradicting the assumption that we are single-query secure). ■

Lecture 22: CPA Model, El Gamal Cryptosystem

The above implies we can encrypt long messages bit-by-bit (or block-by-block) or broken up any reasonable way. One call to Enc on "long" messages translates to many calls on "short" messages, which is fine by the theorem above. 29 Mar. 10:30

Theorem 4.5.1. Any public key encryption scheme with a **deterministic** $\text{Enc}_{k_p}(\cdot)$ algorithm can't be CPA secure *even for one query!*

Proof. Query $c \leftarrow \text{LR}_{k_p, b}(m_0, m_1)$ for any $m_0 \neq m_1$. Then run $c' = \text{Enc}_{k_p}(m_0)$. If $c = c'$ output 0, else 1. This strategy has perfect advantage. ■

4.6 El Gamal Cryptosystem

We can construct a CPA secure PKE scheme using Diffie-Hellman. We can represent a message as some $m \in G$. The "one-time-pad effect" would involve *multiplying* m with something random (\mathcal{K}).

Definition 4.6.1 (El Gamal). We have a scheme:

- $\text{Gen}(1^n)$: choose random $a \leftarrow \mathbb{Z}_q$, output $(k_p = \mathcal{A} = g^a \in G, k_s = a)$
- $\text{Enc}(k_p = \mathcal{A}, m \in G)$: choose random $b \leftarrow \mathbb{Z}_q$, output ciphertext $(\mathcal{B} = g^b \in G, \mathcal{C} = m \cdot \mathcal{A}^b \in G)$
- $\text{Dec}(k_s = a, (\mathcal{B}, \mathcal{C}))$: compute $\mathcal{K} = \mathcal{B}^a$, output $\mathcal{C} \cdot \mathcal{K}^{-1} \in G$

And it is correct, as $\forall m \in G, k_p = g^a, k_s = a$ and:

$$\begin{aligned} \text{Enc}(k_p, \mathcal{A}) &= (\mathcal{B} = g^b, \mathcal{C} = m \cdot (g^a)^b) \\ \text{Dec}(\mathcal{B}, \mathcal{C}) &= \mathcal{C} \cdot (\mathcal{B}^a)^{-1} = m \cdot g^{ab} \cdot (g^{ab})^{-1} = m. \end{aligned}$$

Theorem 4.6.1. If the **DDH assumption** holds, then El Gamal is CPA-secure.

Proof. Assume there is a p.p.t. adversary \mathcal{A} that can break El Gamal, we can use \mathcal{A} to build a distinguisher against DDH.

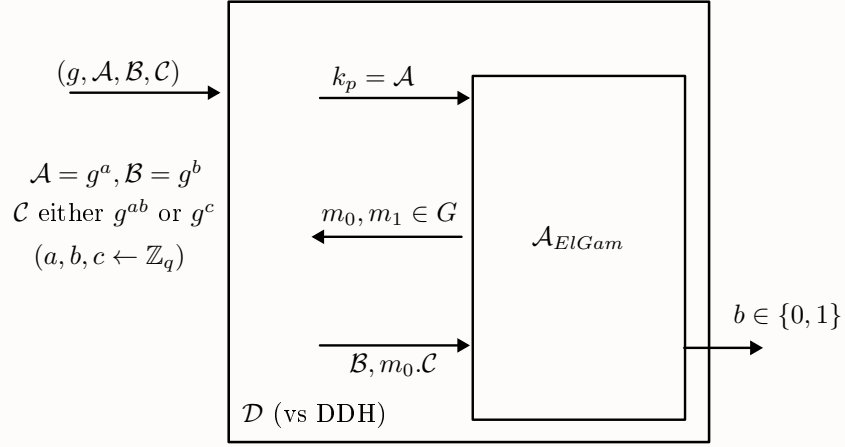


Figure 4.4: A distinguisher using an adversary that can break El Gamal can break DDH.

In the case that $(g, \mathcal{A}, \mathcal{B}, \mathcal{C})$ is a DH tuple ("**real**" world), \mathcal{D} perfectly simulates the left CPA world because $\mathcal{C} = g^{ab}$. In the "**ideal**" world, $(g, \mathcal{A}, \mathcal{B}, \mathcal{C})$ is random, so \mathcal{D} perfectly simulates a "hybrid" CPA world where the ciphertext is two independent random group elements.

Similarly to \mathcal{D} above, we can construct a \mathcal{D}' where instead \mathcal{A}_{ElGam} receives $(\mathcal{B}, m_1.\mathcal{C})$. Thus, by the triangle inequality we have

$$\text{Adv}^{CPA}(\mathcal{A}) \leq \text{Adv}^{DDH}(\mathcal{D}) + \text{Adv}^{DDH}(\mathcal{D}') = \text{negl}(n) + \text{negl}(n) = \text{negl}(n).$$

■

Appendix

Appendix A

Additional Proofs

A.1 Proof of ??

We can now prove ??.

Proof of ??. See [here](#).

