EECS 475: Introduction to Cryptography Notes

Noah Peters

April 11, 2023

Abstract

Lecture notes for EECS 475 at the University of Michigan. LaTeXtemplate by Pingbang Hu.

Contents

Chapter 4

Public Key Cryptography

Lecture 19: Number Theory

20 Mar. 10:30

4.1 Modular Arithmetic and Euclid's Algorithm

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 $0 \le r < b$, 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 ∈ ℤ where a ≥ b > 0
if b|a then
    return x = 0, y = 1
else
    write a = q.b + r where 0 < r < b
x', y' ← ExtEuclid(b, r)
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:

$$b.x' + r.y' = \gcd(b, r) = \gcd(a, b)$$
$$b.x' + (a - q.b).y' = a.y' + b(x' - q.y')$$

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

We claim that the Extended Euclidean Algorithm makes $\mathrm{O}(n)$ recursive calls.

4.2 Group Theoretic View of Numbers

Definition 4.2.1. Let $\mathbb{Z}_N = \{0, 1, 2, \ldots, N-1\}$, 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 $a = b \mod N \Leftrightarrow N|a-b$ and further, if $a = a' \mod N$ and $b = b' \mod N$, we have: 22 Mar. 10:30

- $a + b = a' + b' \mod N$
- $a - b = a' - b' \mod N$
- $a.b = a'.b' \mod N$

However, division is not always possible:

$$3 \cdot 2 = 15 \cdot 2 \mod 24 \not\Rightarrow 3 = 15 \mod 24$$

Definition 4.2.2 (Invertability). An integer b is **invertible** if $\exists c$ where $b \cdot c = 1 \mod N$.

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

Proof. Let $b \cdot c = 1 \mod N$.

$$\Rightarrow b.c - 1 = N.q$$
$$\Rightarrow b.c - N.q = 1.$$

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 $\circ : G \times G \to G$ 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 $(\mathbb{Z}_N, + \mod N)$ we have:

- identity: $a + 0 \equiv 0 + a \equiv a \mod N$
- invertability: $a + (-a) \equiv 0 \mod N$
- associativity: $a + (b + c) = (a + b) + c \mod N$

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

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

The size of a group, called the **group order**, is denoted as $|G|$. For example, $(\mathbb{Z}_N, +)$ has order N, and $\mathbb{Z}_p^*, .$ 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 $g^m = (((g \circ g) \circ g) \circ g) \ldots)$.

Proof. For simplicity, assume G is abelian and suppose $G = \{g_1, g_2, \ldots, g_m$ 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:

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

■

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

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

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

Corollary 4.2.2. For $m = |G| > 1, e \in \mathbb{Z}, \gcd(e, m) = 1$, define $d = e^{-1} \mod m$ and function $f_e \colon G \to G$ 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 $G = \{1 = g^0, g^1, g^2, \ldots, g^{m-1}\}$. If this is the case, we say that g generates G.

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

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

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

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

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

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 exchance 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 $\{0, 1, \ldots, q-1\}$.
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:

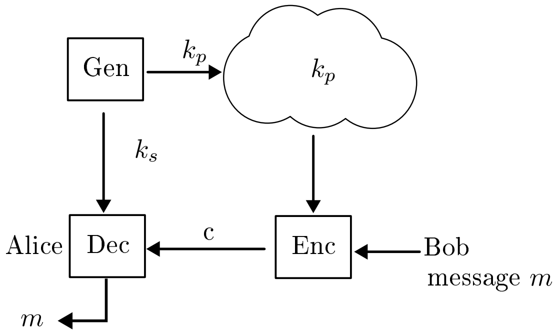

Figure 4.1: Analog of EAV/CPA security in the context of public keys.

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 it's 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.*

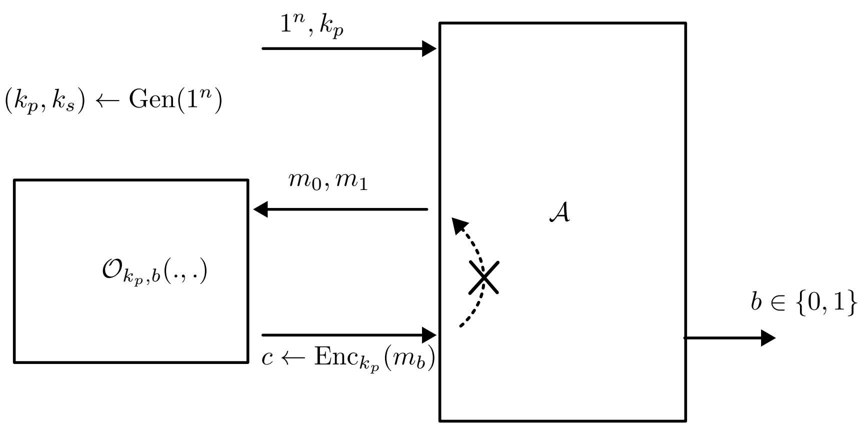

Figure 4.2: The public key CPA game.

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

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

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

public EAV security $\Rightarrow$ *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 : $\mathsf{poly}(n)$ queries. Consider the following worlds:

(i) *Hybrid 0*: All queries answered by $c \leftarrow Enc_{k_p}(m_0)$.

(ii) *Hybrid 1*: First query (m_0, m_1) answered by $c \leftarrow Enc_{k_p}(m_1)$, then $Enc_{k_p}(m_0)$ thereafter.

(iii) *Hybrid 2*: First **two** queries are answered by $c \leftarrow Enc_{k_p}(m_1)$, then $Enc_{k_p}(m_0)$ thereafter.

(iv) *Hybrid q*: All queries answered by $c \leftarrow Enc_{k_p}(m_1)$

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

(i) **Left world**: equivalent to Hyb_0: all queries to the LR oracle are answered by $c \leftarrow Enc_{k_p}(m_0)$.

(ii) **Right world**: equivalent to Hyb_q: all queries answered by $c \leftarrow Enc_{k_p}(m_1)$.

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 ith hybrid world and the $i+1$th hybrid world where the ith hybrid world is defined as

$$HybridWorld_i = \text{First } i \text{ queries of } (m_0, m_1) \text{ answered by } c \leftarrow Enc_{k_p}(m_1),$$
$$\text{then } c \leftarrow Enc_{k_p}(m_0) \text{ thereafter.}$$

We first note that the difference between Hyb_{i-1} and Hyb_i is only in how the ith 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.

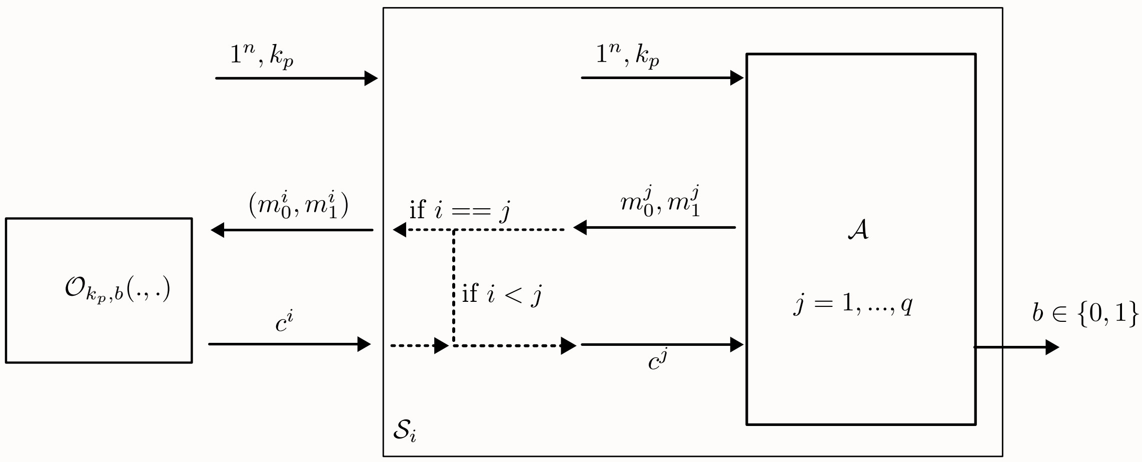

Figure 4.3: A simulator that wins pEAV game using an adversary that wins the pCPA game.

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

- if $j < i$, S_i runs $c^j \leftarrow Enc_{k_p}(m_1^j)$.
- if $j > i$, S_i runs $c^j \leftarrow Enc_{k_p}(m_0^j)$.
- 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:

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

(ii) *Right world* $(b = 1)$: then we are simulating Hyb_i.

By the **triangle inequality**,

$$\begin{aligned}\mathbf{Adv}_{\Pi}^{CPA} &= |\Pr(\mathcal{A} = 1 \text{ in } Hyb_0) - \Pr(\mathcal{A} = 1 \text{ in } Hyb_q)| \\ &= |\Pr(\mathcal{A} = 1 \text{ in } Hyb_0) - \Pr(\mathcal{A} = 1 \text{ in } Hyb_1) + \\ &\quad \Pr(\mathcal{A} = 1 \text{ in } Hyb_1) - \Pr(\mathcal{A} = 1 \text{ in } Hyb_2) + \\ &\quad \ldots - \Pr(\mathcal{A} = 1 \text{ in } Hyb_q)| \\ &\le \sum_{i=1}^{q} \mathbf{Adv}_{\Pi}^{\text{Single CPA}}(\mathcal{S}_i) = q \cdot \operatorname{negl}(n) = \operatorname{poly}(n) \cdot \operatorname{negl}(n) = \operatorname{negl}(n).\end{aligned}$$

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** $Enc_{k_p}(.)$ algorithm can't be CPA secure *even for one query*!

Proof. Query $c \leftarrow LR_{k_p,b}(m_0, m_1)$ for any $m_0 \neq m_1$. Then run $c' = Enc_{k_p}(m_0)$. 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:

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

$$Enc(k_p, \mathcal{A}) = (\mathcal{B} = g^b, \mathcal{C} = m \cdot (g^a)^b)$$

$$Dec(\mathcal{B}, \mathcal{C}) = \mathcal{C} \cdot (\mathcal{B}^a)^{-1} = m \cdot g^{ab} \cdot (g^{ab})^{-1} = m.$$

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.

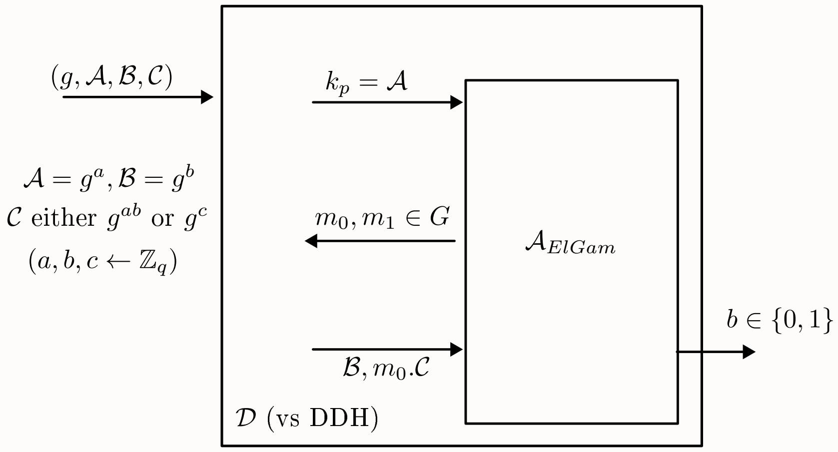

Figure 4.4: A distinguisher using an adversary that can break El Gamal can break DDH.

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

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

■

Lecture 23: RSA Cryptosystem

4.7 RSA Cryptosystem

3 Apr. 10:30

Whereas Diffie-Hellman relies on the hardness of the discrete log problem in a group of *known* order, RSA relies on the hardness of **factoring and finding roots** in a group of *unknown* order.

4.7.1 RSA Math Foundation

First, recall Euler's totient function.

Definition 4.7.1 (Euler's Totient Function)**.** The totient, $\Phi(n)$, of a positive integer $n > 1$ is defined as hee number of positive integers less than n that are coprime to n. The following table shows some function values.

n	$\Phi(n)$	numbers coprime to n
1	1	1
2	1	1
3	2	1, 2
4	2	1, 3
5	4	1, 2, 3, 4
6	2	1, 5

Let $N = p \cdot q$ be the product of two (huge) distinct primes. Then

$$\mathbb{Z}_N^* = \{a \in \mathbb{Z}_N = 0, \dots, N-1 : \gcd(a, N) = 1\}.$$

We start with $\mathbb{Z}_N$ and remove all multiples of p (i.e. $0, p, 2p, \ldots, (q-1)p$) and all multtiples of q (i.e $0, q, 2q, \ldots, (p-1)q$) double counted 0. This means that

$$|\mathbb{Z}_N^*| = \Phi(N) = p.q - q - p + 1 = (p-1)(q-1).$$

Definition 4.7.2 (Euler's Theorem)**.** In any group G, $\forall a \in G$, $a^{|G|} = 1 \in G$. Say $G = \mathbb{Z}_N^*$, therefore we have

$$\forall a \in \mathbb{Z}_N^*, a^{\Phi(N)} = a^{(p-1)(q-1)} = 1 \mod N.$$

By Euclid's Theorem, we can compute $A, B \in \mathbb{Z}$ such that

$$A.e + B.\Phi(N) = \gcd(e, \Phi(N)) = 1.$$

$$\Rightarrow A.e = 1 - B.\Phi(N) = 1 \mod \Phi(N)$$

We can define $d = A \mod \Phi(N)$ as the multiplicative inverse of $e \mod \Phi(N)$:

$$d = e^{-1} \mod \Phi(N)$$
$$d.e = 1 \mod \Phi(N).$$

4.7.2 RSA Function

The choice of N, e, d gives us the RSA function and its inverse.

Definition 4.7.3 (RSA Function)**.** For $N = p.q$ where p, q are large disttinctt primes, and $e \in \mathbb{Z}_{\Phi(N)}^*$ with $d = e^{-1} \mod \Phi(N)$ the RSA function

$$\mathrm{RSA}_{N,e} : \mathbb{Z}_N^* \to \mathbb{Z}_N^*; \ \mathrm{RSA}_{N,e}(x) := x^e \mod N$$

is a bijection. The inverse is $\mathrm{RSA}_{N,d}(y) = \mathrm{RSA}_{N,e}^{-1}(y) = y^d \mod N$.

Proof. $\mathrm{RSA}_{N,e}$ maps $\mathbb{Z}_N^* \to \mathbb{Z}_N^*$. Need to show $\mathrm{RSA}_{N,d} = \mathrm{RSA}_{N,e}^{-1}$. Why? Let $y = \mathrm{RSA}_{N,e}(x) = x^e \mod N$. Then we have

$$y^d = (x^e)^d = x^{e.d} = x^{e.d + k.\Phi(N)} = x^1 \mod N,$$

where k is some number that will make $e.d + k.\Phi(N) = 1$. RSA is an example of a **trapdoor function**. We can efficienly evaluate $\mathrm{RSA}_{N,e}$ in the *forward* direction, and given **trapdoor information**, d, we can efficiently invert.

Remark. What about without d?

■

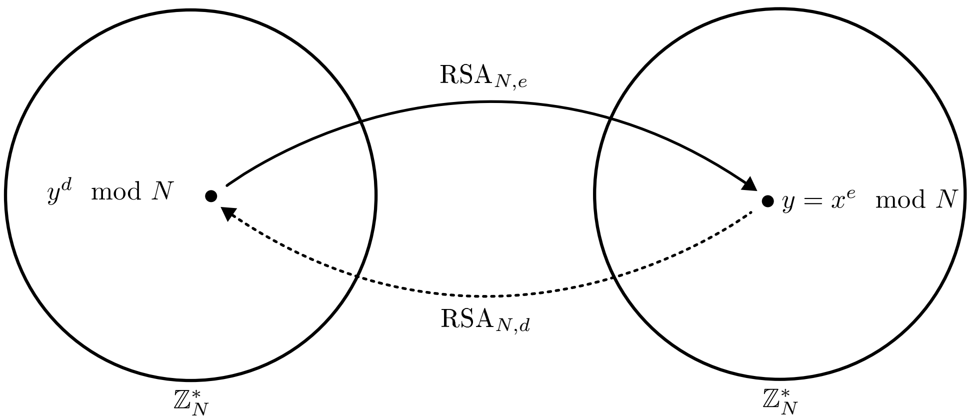

Figure 4.5: RSA function is a trapdoor function: it is easy to go left to right, but not the other way around.

4.7.3 RSA Key Generation

We define the RSA key generation process for $\text{GenRSA}(1^n)$ as follows:

Algorithm 4.2: RSA Key Generation

Data: 1^n

let p, q be large primes having bit lengths approximately related to n
let $N = p.q$
compute $\Phi(N) = (p-1)(q-1)$
choose some $e > 1$ such that $\gcd(e, \Phi(N)) = 1$; Euclid also gives us $d = e^{-1} \mod \Phi(N)$
return $pk = (N, e)$ and $sk = (N, d)$.

Remark. Common choices for e:

- random value
- $e = 3$ if $3t(p-1), 3t(q-1)$
- $e = 2^{16} + 1$ (prime) (in binary: $e = 1000\ldots0001$, so exponentiation is faster)

Definition 4.7.4 (RSA Hardness Assumption)**.** Given a public key (N, e) and a *random* $y \leftarrow \mathbb{Z}_N^*$, it is hard to find the pre-image $x = y^d = y^{e^{-1}} \mod N$. So, the assumption is that $\forall$ p.p.t $\mathcal{A}$:

$$\text{Adv}^{\text{RSA}}(\mathcal{A}) = \Pr_{(pk=(N,e),sk)\leftarrow\text{GenRSA}(1^n),y\leftarrow\mathbb{Z}_N^*}\left[\mathcal{A}(1^n, (N, e), y) \text{ outputs } x = \text{RSA}_{N,e}^{-1}(y)\right] = \text{negl}(n).$$

How plausible is the **RSA hardness assumption**? Well, we can note the following:

(i) RSA $\leq$ Factoring; i.e. if there is an efficient algorithm for factoring integers into their prime factors, then there is one for solving RSA.

(ii) Factoring $\equiv$ Finding $\Phi(N)$ from N.

(iii) Factoring $\equiv$ Finding d from (N, e).

Remark. It is noteworthy, however, that despite RSA $\leq$ Factoring, it is *unknown* whether RSA $\equiv$ Factoring. In other words, we know Factoring is at least as hard as RSA, but we don't know if it is *just* as hard.

4.7.4 RSA Encryption

Say our encryption scheme is the "textbook" RSA encryption scheme with a key generator $\text{Gen}(1^n)$: $(pk = (N, e), sk = (N, d))$, defined as:

- $\text{Enc}(pk = (N, e), m \in \mathbb{Z}_N^*)$: output $c = \text{RSA}_{N,e}(m) = m^e \mod N$
- $\text{Dec}(sk = (N, d), c \in \mathbb{Z}_N^*)$: output $m = \text{RSA}_{N,d}(c) = c^d \mod N$

This scheme is correct, but since *Enc is deterministic*, it is **not CPA secure**.

Appendix