

Introduction

788.11 Cryptography and Security

Fall 2009

Steve Lai

Outline

- Basics of encryption
- Homomorphic encryption

Basics of Encryption

For more information, see my
CSE 651 or 794Q notes

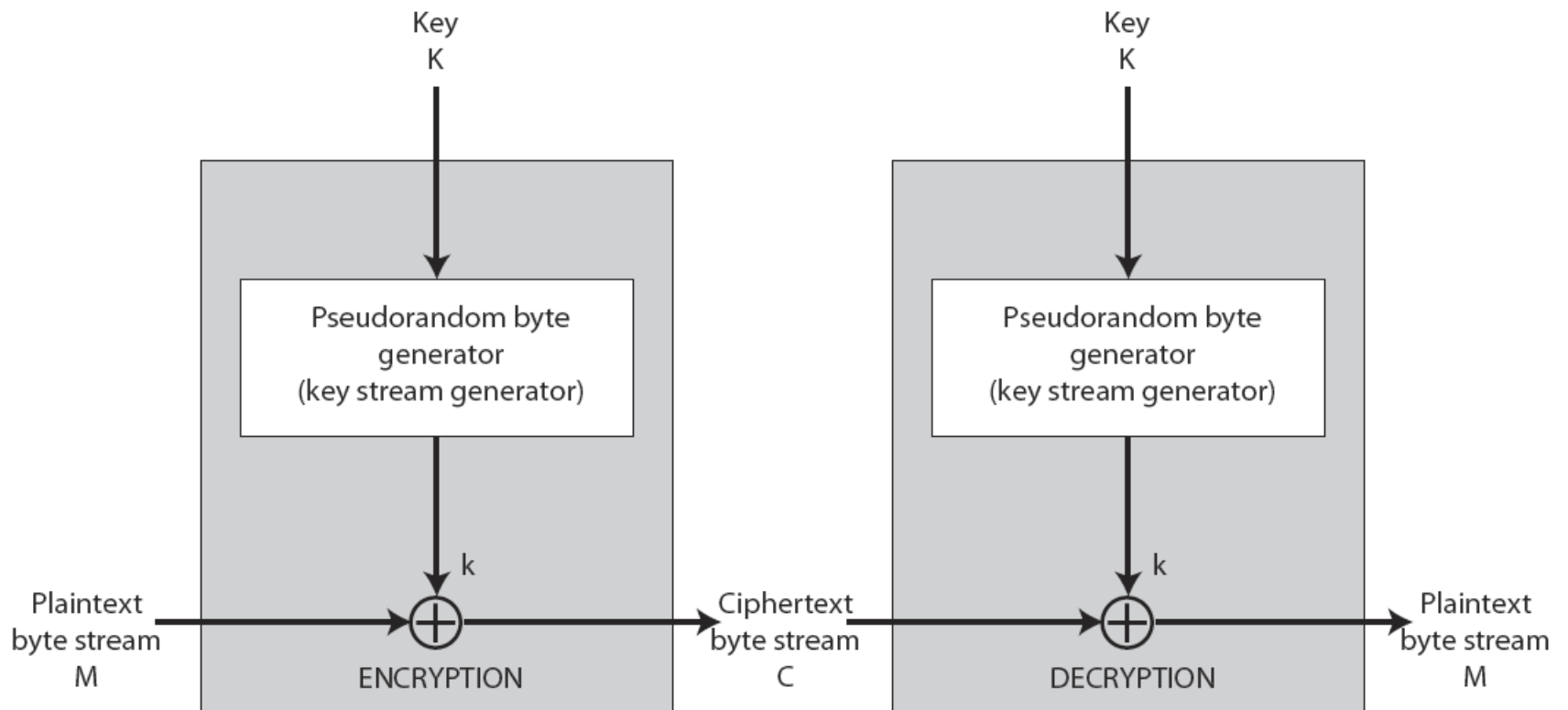
Summary

- Symmetric encryption
 - Stream cipher (e.g., RC4)
 - Block cipher (e.g., DES, AES)
- Asymmetric encryption
 - RSA
 - ElGamal (based on Diffie-Hellman)
- Performance issues
- Security issues

Symmetric-Key Encryption

- Stream cipher
(e.g., Vernan's one-time pad, RC4)
- Block cipher (e.g., DES, AES)

Stream ciphers



Stream ciphers

- Stream ciphers typically process the plaintext byte by byte.
- So, the plaintext is a stream of bytes: P_1, P_2, P_3, \dots
- Use a key K as the seed to generate a sequence of pseudorandom bytes (key-stream): K_1, K_2, K_3, \dots
- The ciphertext is $C_1, C_2, C_3, C_4, \dots$, where
$$C_i = P_i \oplus K_i$$
- Various stream ciphers differ in their key-stream generators.
- Stream ciphers require that a new key be used for each plaintext (or it will not be secure).

- In practice, Alice and Bob wish to share a permanent key K and use it to encrypt many messages. One possible strategy:
 - Suppose Bob and Alice share a secret key K .
 - Each time Bob (or Alice) wants to send a message, he randomly generates a string IV and use $K || IV$ as the key (seed) to the pseudorandom generator.
 - Send IV along with the ciphertext.
- Unfortunately, the resulting scheme is not necessarily secure.

Example: WEP's use of RC4

- WEP is a protocol using RC4 to encrypt packets for transmission over IEEE 802.11 wireless LAN.
- Each packet is encrypted with a separate key equal to the concatenation of a 24-bit IV (initialization vector) and a 40 or 104-bit permanent key.
- Not secure. See “**Breaking 104 bit WEP in less than 60 seconds.**”

RC4 key:

IV (24)	Permanent key (40 or 104 bits)
---------	--------------------------------

Block Ciphers

Block ciphers are encryption schemes that use pseudorandom functions or pseudorandom permutations.

Traditional view of block ciphers

- A block cipher is a symmetric-key **encryption scheme** that maps a block of n bits to a block of n bits.
 - $M = C = \{0,1\}^n$ and $K = \{0,1\}^r$.
 - Block length: n .
 - Key length: r .
 - For a fixed key $k \in K$, $E_k : \{0,1\}^n \rightarrow \{0,1\}^n$ is a permutation.

Practical Block Ciphers: DES and AES

DES: Data Encryption Standard

AES: Advanced Encryption Standard

Public-Key Cryptography and RSA

Public-Key Cryptography

- Also known as **asymmetric-key** cryptography.
- Each user has a pair of keys: a public key and a private key.
- The public key is used for encryption.
 - The key is known to the public.
- The private key is used for decryption.
 - The key is only known to the owner.

Public-Key Cryptosystem (PKC)

- Each user u has a pair of keys (PK_u, SK_u) .
 - PK_u is the public key, available in a public directory.
 - SK_u the private key, known to u only.
- **Key-generation algorithm:** to generate keys.
- **Encryption algorithm E :** to send message M to user u , compute $C = E(PK_u, M)$.
- **Decryption algorithm D :** Upon receiving C , user u computes $D(SK_u, C)$.
- Requirement: $D(SK_u, E(PK_u, M)) = M$.

Why Public-Key Cryptography?

- Developed to address two main issues:
 - key distribution
 - digital signatures
- Invented by Diffie & Hellman in 1976.

One-way function with trapdoor

Easy: $x \xrightarrow{f} y$

Hard: $x \xleftarrow{f^{-1}} y$

Easy: $x \xleftarrow[\text{trapdoor}]{f^{-1}} y$

Use **trapdoor** as the private key.

- Most (believed) one-way functions come from number theory.

The RSA Cryptosystem

- RSA Encryption
- RSA Digital signature

The RSA Cryptosystem

- By **R**ivest, **S**hamir & **A**dleman of MIT in 1977.
- Best known and most widely used public-key scheme.
- Based on the **assumed** one-way property of modular powering:

$$f : x \rightarrow x^e \pmod n \quad (\text{easy})$$

$$f^{-1} : x^e \rightarrow x \pmod n \quad (\text{hard})$$

Idea behind RSA

It works in group Z_n^* .

Encryption (easy): $x \xrightarrow{\text{RSA}} x^e$

Decryption (hard): $x \xleftarrow{\text{RSA}^{-1}} x^e$

Looking for a trapdoor: $(x^e)^d = x$.

If d is a number such that $ed \equiv 1 \pmod{\varphi(n)}$, then

$ed = k\varphi(n) + 1$ for some k , and

$$(x^e)^d = x^{ed} = x^{\varphi(n)k+1} = \left(x^{\varphi(n)}\right)^k \cdot x = 1 \cdot x = x.$$

RSA Cryptosystem

- Key generation:
 - (a) Choose large primes p and q , and let $n := pq$.
 - (b) Choose e ($1 < e < \varphi(n)$) coprime to $\varphi(n)$, and compute $d := e^{-1} \bmod \varphi(n)$. ($ed \equiv 1 \bmod \varphi(n)$.)
 - (c) Public key: $pk = (n, e)$. Secret key: $sk = (n, d)$.
 - Encryption: $E_{pk}(x) := x^e \bmod n$, where $x \in Z_n^*$.
 - Decryption: $D_{sk}(y) := y^d \bmod n$, where $y \in Z_n^*$.
- (E_{pk} and D_{sk} work for $x, y \in Z_n \setminus Z_n^*$, but not secure.)

Mathematical Attacks

- Factor n into pq . Then $\varphi(n) = (p - 1)(q - 1)$ and $d = e^{-1} \bmod \varphi(n)$ can be calculated easily.
- Determine $\varphi(n)$ directly. Equivalent to factoring n . Knowing $\varphi(n)$ will enable us to factor n by solving
$$\begin{cases} n = pq \\ \varphi(n) = (p - 1)(q - 1) \end{cases}$$
- Determine d directly. The best known algorithms are not faster than those for factoring n . Also, if d is known, n can be factored with high probability.

Remarks

- In light of current factorization technologies, RSA recommends that n be of 1024-2048 bits.
- If a message $m \in \mathbb{Z}_n \setminus \mathbb{Z}_n^*$,
 - RSA works, but
 - Since $\gcd(m, n) > 1$, the sender can factor n .
 - Also, since $\gcd(m^e, n) > 1$, the adversary can factor n , too.
- Question: how likely is $m \in \mathbb{Z}_n \setminus \mathbb{Z}_n^*$?

Security of RSA

- We have seen many attacks on RSA.
- Also, RSA is deterministic and, therefore, not CPA-secure (i.e., not ciphertext-indistinguishable against CPA).
- We wish to make RSA secure against CPA and aforementioned attacks.
- RSA primitive: the RSA we have described.
 - also called plain RSA or textbook RSA

Padded RSA

- Encryption: $E_{pk}(m) = \text{RSA}(r \parallel m) = (r \parallel m)^e \bmod n$,
where r is a random string.
- Thus, $\text{Padded-RSA}(m) = \text{RSA}(r \parallel m)$ for some random r .
- Secure against many of aforementioned attacks.
- Theorem: Padded RSA is CPA-secure if $|m| = O(\log|n|)$.
- Padded RSA is adopted in PKCS #1 v.1.5.

Padded RSA as in PKCS #1 v.1.5

- PKCS: **P**ublic **K**ey **C**ryptography **S**tandard.
- Let (n, e, d) give a pair of RSA keys.
- Say $|n| = k$ bytes (e.g., $k = 216$). First byte $\neq 00$.
- To encrypt a message m :
 - pad m so that $m' = 00 \parallel 02 \parallel r \parallel 00 \parallel m$ (k bytes)
 - where $r = 8$ or more random bytes $\neq 00$.
 - original message m must be $\leq k - 11$ bytes.
 - the ciphertext is $c := \text{RSA}(m') = (m')^e \bmod n$.
- In 1998, Bleichenbacher published a chosen-ciphertext attack, forcing RSA to upgrade its PKCS #1, now using OAEP.

OAEP: basic idea

- Message padding: instead of encrypting m directly, we encrypt $m \oplus r || r$, where r is a random bit string.
- As such, however, there is a 50% overhead.
So, we wish to use a shorter bit string r .
- Besides, r should be protected, too.
- This leads to a scheme called Optimal Asymmetric Encryption Padding (OAEP). It can be applied not only to RSA but to other trapdoor functions.

OAEP

- Choose k, l ($k \ll l$) s.t. $k + l = |n|$. (n , RSA modulus).
- $G : \{0,1\}^k \rightarrow \{0,1\}^l$, a pseudorandom generator.
- $h : \{0,1\}^l \rightarrow \{0,1\}^k$, a hash function.
- **Encryption.** To encrypt a block m of l bits :
 1. choose a random bit string $r \in \{0,1\}^k$.
 2. encode m as $x := (m \oplus G(r) \parallel r \oplus h(m \oplus G(r)))$
(if $x \notin Z_n$, the message space of RSA, return to step 1).
 3. compute the ciphertext $y := E_{pk}(x)$.
- **Decryption:** $x := D_{sk}(y) = a \parallel b$. $m = a \oplus G(b \oplus h(a))$.

Remarks on OAEP

- OAEP is adopted in current RSA PKCS #1 (v. 2.1).
- It is a **padding** scheme, not an encryption scheme.
- Intuitively, with OAEP, the ciphertext should not reveal any information about the plaintext if RSA is one-way and h and G are **truly random (random oracles)**.
- A slightly more complicated version of OAEP, in which
$$x = (m0^{k'} \oplus G(r) || r \oplus h(m0^{k'} \oplus G(r))),$$
has been proved CCA-secure in the **random oracle** model (i.e., if G , h are random oracles.)
- In practice, hash functions such as SHA-1 are used for G , h .

Random Oracle

- A random oracle is a random function $f : \{0,1\}^n \rightarrow \{0,1\}^{l(n)}$.
- Recall: there are $2^{l(n) \cdot 2^n}$ such functions.
- Each random oracle is a black box that implements one of the $2^{l(n) \cdot 2^n}$ random functions, say f_0 .
- The 2^n values of f_0 are totally independent and random.
- The only way to know the value of $f_0(x)$ is to explicitly evaluate f_0 at x (i.e., to ask the oracle).
- No practical/feasible way to implement a random oracle.
 - Infeasible: use a trusted authority.
 - Infeasible: use a $l(n) \cdot 2^n$ -bit disk.

Cryptosystems Based on Discrete Logarithms

Outline

- Discrete Logarithm Problem
- Diffie-Hellman key agreement
- ElGamal encryption

Discrete logarithm problem (DLP)

- A group G is **cyclic** if there is an element $\alpha \in G$ of order $|G|$.
In this case, $G = \{\alpha^0, \alpha^1, \alpha^2, \dots, \alpha^{|G|-1}\}$; α is called a generator.
- If $(G, *)$ be a finite group (not necessarily cyclic) and $\alpha \in G$ an element of order n , then $\langle \alpha \rangle = \{\alpha^0, \alpha^1, \alpha^2, \dots, \alpha^{n-1}\}$ is a cyclic (sub)group of order n .
- For any $y \in \langle \alpha \rangle$, there is a unique $x \in \mathbb{Z}_n$ such that $\alpha^x = y$.
This integer x is called the discrete logarithm (or index) of y with respect to base α . We write $\log_\alpha y = x$.
- The DLP is to compute $\log_\alpha y$ for a given y .

Frequently used settings

- $G = Z_p^*$. $\langle \alpha \rangle = \{ \alpha^0, \alpha^1, \alpha^2, \dots, \alpha^{p-2} \} = G$,

where p is a large prime, and α is a generator of G .

(Z_p^* is cyclic when p is prime.)

- $G = Z_p^*$. $\langle \alpha \rangle = \{ \alpha^0, \alpha^1, \alpha^2, \dots, \alpha^{q-1} \} \subset Z_p^*$,

where $\alpha \in Z_p^*$ is an element of prime order q .

- For these settings, there is no polynomial-time algorithm for DLP.

Example 1

$$G = Z_{19}^* = \{1, 2, \dots, 18\}.$$

2 is a generator. That is, $Z_{19}^* = \langle 2 \rangle$.

$$2^0 = 1, 2^1 = 2, 2^2 = 4, 2^3 = 8, 2^4 = 16, 2^5 = 13,$$

$$2^6 = 7, 2^7 = 14, \dots$$

$$\log_2 7 = 6$$

$$\log_2 14 = 7$$

$$\log_2 12 = ?$$

Example 2

$$G = Z_{11}^* = \{1, 2, \dots, 10\}.$$

$$G' = \langle 3 \rangle = \{1, 3, 9, 5, 4\}$$

3 is a generator of G' , but not a generator of Z_{11}^* .

$$\log_3 5 = 3$$

$$\log_3 10 = \text{not defined}$$

DLP in Z_p^*

- Let α be a generator of Z_p^* (a primitive root of unity modulo p).
- $Z_p^* = \{1, 2, \dots, p-1\} = \{\alpha^0, \alpha^1, \alpha^2, \dots, \alpha^{p-2}\}$.
- $Z_{p-1} = \{0, 1, 2, \dots, p-2\}$.
- Given $y \in Z_p^*$, find the unique $x \in Z_{p-1}$ such that $y = \alpha^x \pmod p$.
- That is, given $\alpha^x \in Z_p^*$, find x .
- There is a subexponential-time algorithm for DLP in Z_p^*
 - Index Calculus, $O\left(2^{O(\sqrt{n \log n})}\right)$, where $n = \log p$.

RSA vs. Discrete Logarithm

- RSA is a one-way **trapdoor** function:

$$x \xrightarrow{\text{RSA}} x^e \quad (\text{easy})$$

$$x \xleftarrow{\text{RSA}^{-1}} x^e \quad (\text{difficult})$$

$$x \xleftarrow{\text{RSA}^{-1}} (x^e)^d \quad (d \text{ is a trapdoor})$$

- Logarithm is the inverse of exponentiation:

$$x \xrightarrow{\text{exp}_\alpha} \alpha^x \quad (\text{easy})$$

$$x \xleftarrow{\text{log}_\alpha} \alpha^x \quad (\text{difficult})$$

- log is hard to compute, so exp is a one-way function, but **without a trapdoor**.
- An encryption scheme based on the difficulty of log will **not** simply encrypt x as α^x .

Diffie-Hellman key agreement

- $Z_p^* = \{\alpha^0, \alpha^1, \alpha^2, \dots, \alpha^{p-2}\}$. $Z_{p-1} = \{0, 1, 2, \dots, p-2\}$.
- Alice and Bob wish to set up a **secret** key.
 1. Alice and Bob agree on a large prime p and a primitive root (generator) $\alpha \in Z_p^*$. (p, α , not secret)
 2. Alice \rightarrow Bob: $\alpha^a \bmod p$, where $a \leftarrow_R Z_{p-1}$.
 3. Alice \leftarrow Bob: $\alpha^b \bmod p$, where $b \leftarrow_R Z_{p-1}$.
 4. They agree on the key: $\alpha^{ab} \bmod p$.
- Diffie-Hellman problem: given $\alpha^a, \alpha^b \in Z_p^*$, compute α^{ab} .
- Diffie-Hellman assumption: the Diffie-Hellman problem is intractable.

Ideas behind ElGamal encryption in Z_p^*

0. Bob is to send a message m to Alice, who has private key x and public key $y := \alpha^x$.
1. Regard m as an element in Z_p^* .
2. Use Diffie-Hellman to set up a temporary key.
 - Bob generates k and computes $y^k (= \alpha^{xk})$.
3. Bob uses this key to encrypt m as $m \cdot y^k$.
4. Bob sends α^k along with $m \cdot y^k$ so that Alice can compute α^{xk} .
That is, $E(m) = (\alpha^k, m \cdot y^k)$

ElGamal encryption in Z_p^*

1. Key generation (e.g. for Alice):

- choose a large prime p and a primitive root $\alpha \in Z_p^*$, where $p - 1$ has a large prime factor.
- randomly choose a number $x \in Z_{p-1}$ and compute $y = \alpha^x$;
- set $sk = (p, \alpha, x)$ and $pk = (p, \alpha, y)$.

2. Encryption: $E_{pk}(m) = (\alpha^k, my^k)$, where $m \in Z_p^*$, $k \leftarrow_{\mathbb{R}} Z_{p-1}$.

3. Decryption: $D_{sk}(a, b) = ba^{-x}$.

4. Remarks: All operations are done in Z_p^* , *i.e.*, modulo p .

The encryption scheme is **non-deterministic**.

Security of ElGamal encryption against CPA

- Based on the Diffie-Hellman assumption.
- Diffie-Hellman problem \leq discrete logarithm problem.
- Open problem: discrete logarithm \leq Diffie-Hellman ?

Theorem: If the Diffie-Hellman assumption is true, then the ElGamal encryption scheme is CPA-secure.

Security of ElGamal encryption against CCA

- A function $f : G \rightarrow G'$ is **homomorphic** if $f(xy) = f(x)f(y)$.
- **ElGamal encryption is homomorphic**, $E(mm') = E(m) \cdot E(m')$, in the following sense:

If $E(m) = (\alpha^k, my^k)$ and $E(m') = (\alpha^{k'}, m'y^{k'})$, then $E(m) \cdot E(m')$
 $= (\alpha^k, my^k) \cdot (\alpha^{k'}, m'y^{k'}) = (\alpha^k \alpha^{k'}, my^k m'y^{k'}) = (\alpha^{k+k'}, mm'y^{k+k'})$
is a valid encryption of mm' .

- As such, ElGamal encryption is **not CCA-secure** (i.e., not indistinguishable against CCA).

Symmetric vs. Asymmetric

- Symmetric encryptions are much faster than asymmetric ones.
 - AES is typically 100 times faster than RSA encryption, and 1000 times faster than RSA decryption.
- Use asymmetric cipher to set up a session key and then use symmetric cipher to encrypt data.

Security Issues

What does it mean that an encryption scheme is **secure** (or **insecure**)?

- Semantic security
- Ciphertext-indistinguishability
- Non-malleability

Different levels of security

- Consider ciphertext-only attacks; i.e., the adversary is an eavesdropper. **How to define security?**
- Several options: An encryption scheme is **secure** if given a ciphertext $c = E_k(m)$, no adversary can
 - (1) find the secret key k
 - (2) find the plaintext m
 - (3) find any character of the plaintext
 - (4) find any meaningful information about the plaintext
 - (5) find any information about the plaintext.
- We will adopt (and formalize) **#5**, which is called **semantic security** and seems to indicate the highest level of security.

Different types of attackers

- Different types of attacks (classified by the amount of information that may be obtained by the attacker):
 - Ciphertext-only attack
 - Known-plaintext attack
 - Chosen-plaintext attack (CPA)
 - Chosen-ciphertext attack (CCA)

Security Parameter

- The security of an encryption scheme typically depends on its key length.
 - Is RSA secure if $|n| = 216, 512, \text{ or } 1024$?
- In general, an encryption scheme is associated with an integer called its **security parameter**. (For now, you may think of it as **key length**.)
- When we say that the **probability $\Pr(n)$ of an encryption scheme being broken** is negligible, it is w.r.t. the encryption scheme's **security parameter**.

Negligible functions

- A nonnegative function $f : N \rightarrow R$ is said to be **negligible** if for every positive polynomial $P(n)$, there is an integer n_0 such that

$$f(n) < \frac{1}{P(n)} \quad \text{for all } n > n_0 \quad (\text{i.e., for sufficiently large } n).$$

- Examples: 2^{-n} , $2^{-\sqrt{n}}$, $n^{-\log n}$ are negligible functions.
- Negligible functions approach zero faster than the reciprocal of **every** polynomial.
- We write **negl**(n) to denote an unspecified negligible function.

Symmetric-key encryption scheme

- Message space: $M \subseteq \{0,1\}^*$.
- Key generation algorithm G : On input 1^n , $G(1^n)$ outputs a key $k \leftarrow \{0,1\}^n$. ($K = \{0,1\}^n$; and n is the security parameter.)
- Encryption algorithm E : On input a key k and a plaintext $m \in M$, E outputs a ciphertext c . We write $c \leftarrow E(k, m)$ or $c \leftarrow E_k(m)$.
- Decryption algorithm D : On input a key k and a ciphertext c , D outputs a message m . We write $m := D(k, c)$ or $m := D_k(c)$.
- Correctness requirement: for each $k \in K$ and $m \in M$,
$$D_k(E_k(m)) = m.$$
- G, E are polynomial probabilistic algorithms. D is deterministic.

Semantic Security

- Informally, an encryption scheme is **semantically secure** if whatever an adversary with $c = E(m)$ can learn about m , one can learn equally well without c .
- A private-key encryption scheme (G, E, D) with security parameter n is **semantically secure** against an eavesdropper if for every probabilistic polynomial-time (PPT) algorithm A there exists a PPT A' such that for all polynomial-time computable functions f and h , there exists a negligible function $negl$ such that:

$$\left| \Pr \left[A(1^n, E_k(m), h(m)) = f(m) : k \leftarrow G(1^n), m \leftarrow \{0,1\}^n \right] - \Pr \left[A'(1^n, h(m)) = f(m) : m \leftarrow \{0,1\}^n \right] \right| \leq negl(n).$$

Ciphertext-Indistinguishability

- Adversary: a **polynomial-time** eavesdropper.
- (G, E, D) : an encryption scheme with security parameter n .
- Imagine a game played by Bob and Eve (adversary):
 - Eve is given input 1^n and outputs a pair of messages m_0, m_1 **of the same length**.
 - Bob chooses a key $k \leftarrow G(1^n)$ and $m \leftarrow_u \{m_0, m_1\}$.
He computes $c \leftarrow E_k(m)$ and gives c to Eve.
 - Eve tries to determine whether c is the encryption of m_0 or m_1 .
- An encryption scheme is **ciphertext-indistinguishable against eavesdroppers** if no adversary can succeed with probability **non-negligibly greater than $1/2$** .

- **Definition:** An encryption scheme is **ciphertext-indistinguishable against eavesdroppers** if for every PPT algorithm A and all $m_0, m_1 \in M$, $|m_0| = |m_1|$, it holds:

$$\Pr \left[A(1^n, m_0, m_1, E_k(m)) = m : m \leftarrow_u \{m_0, m_1\}, k \leftarrow G(1^n) \right] \leq \frac{1}{2} + \text{negl}(n)$$

Equivalence of semantic security and ciphertext-indistinguishability

- **Theorem:** Against an eavesdropper, an encryption scheme is semantically secure iff it is ciphertext-indistinguishable.
- **Theorem:** Under CPA, CCA1 or CCA2, an encryption scheme is semantically secure if and only if it is ciphertext-indistinguishable.

Chosen-plaintext attacks (CPA)

- In CSE 651 we described CPA as follows:
 - Given: $(m_1, c_1), (m_2, c_2), \dots, (m_t, c_t)$, where m_1, m_2, \dots, m_t are chosen by the adversary; and a new ciphertext c .
 - Q: what is the plaintext of c ?
- Adaptively-chosen-plaintext attack: m_1, m_2, \dots, m_t are chosen adaptively.
- Now we describe CPA in terms of oracle.

Chosen-plaintext attacks (CPA)

A CPA on an encryption scheme (G, E, D) is modeled as follows.

1. A key $k \leftarrow G(1^n)$ is generated.
2. The adversary is given input 1^n and oracle access to E_k . She may request the oracle to encrypt plaintexts of her choice.
3. The adversary chooses two messages m_0, m_1 with $|m_0| = |m_1|$; and is given a challenge ciphertext $c \leftarrow E_k(m_b)$, where $b \leftarrow_u \{0, 1\}$.
4. The adversary continues to have oracle access and may request the encryptions of additional plaintexts of her choice, even m_0 and m_1 .
5. The adversary finally answers 0 or 1.

Note: The CPA here actually refers to an **adaptive** CPA.

Ciphertext-indistinguishability against CPA

- An encryption scheme (G, E, D) is IND-CPA if no **polynomial-time** adversary can answer correctly with probability non-negligibly greater than $1/2$.
- Definition: an encryption scheme (G, E, D) is IND-CPA if for every polynomial adversary A it holds that:

$$\left| \Pr \left[A^{E_k} \left(1^n, m_0, m_1, E_k(m) \right) = m : k \leftarrow G(1^n), m \leftarrow_u \{m_0, m_1\}, \right. \right. \\ \left. \left. m_0, m_1 \leftarrow_A M \right] \right| \\ \leq \frac{1}{2} + \text{negl}(n)$$

Chosen-ciphertext attacks (CCA)

- In CSE 651 we also described CCA as follows:
 - Given : $(m_1, c_1), (m_2, c_2), \dots, (m_t, c_t)$, where c_1, c_2, \dots, c_t are chosen by the adversary; and a new ciphertext c .
 - Q: what is the plaintext of c ?
- Adaptively-chosen-ciphertext attack : c_1, c_2, \dots, c_t are chosen adaptively.
- Now we describe CCA in terms of oracle.
- We will allow a CCA adversary to also have CPA capability.
(So, combined CCA+CPA, rather than pure CCA.)

Chosen-ciphertext attacks (CCA)

A CCA on an encryption scheme (G, E, D) is modeled as follows.

1. A key $k \leftarrow G(1^n)$ is generated.
2. The adversary is given input 1^n and oracle access to E_k and D_k .
She may request the oracles to perform encryptions and/or decryptions for her.
3. The adversary chooses two messages m_0, m_1 with $|m_0| = |m_1|$; and is given a challenge ciphertext $c \leftarrow E_k(m_b)$, where $b \leftarrow_u \{0, 1\}$.
4. The adversary continues to have oracle access to E_k and D_k , but is not allowed to request the decryption of c .
5. The adversary finally answers 0 or 1.

CCA1 vs. CCA2

- The CCA described above is also called CCA2.
- If in item #4 the adversary has no access to the decryption oracle, the CCA is called CCA1.

Ciphertext-indistinguishability against CCA

- An encryption scheme (G, E, D) is IND-CCA if no **polynomial-time** adversary can answer correctly with probability non-negligibly greater than $1/2$.
- Definition: an encryption scheme (G, E, D) is IND-CCA if for every polynomial-time adversary A , it holds that:

$$\left| \Pr \left[A^{E_k, D_k} \left(1^n, m_0, m_1, E_k(m) \right) = m : k \leftarrow G(1^n), m \leftarrow_u \{m_0, m_1\}, \right. \right.$$

$$\left. \left. m_0, m_1 \leftarrow_A M \right] \right|$$

$$\leq \frac{1}{2} + \text{negl}(n)$$

Non-malleability

- An encryption scheme (G, E, D) is **non-malleable** if given a ciphertext $c = E(m)$, it is computationally infeasible for an adversary to produce a ciphertext c' such that $m' = D(c')$ has some known relation with m .
- RSA is malleable.
- IND-CCA2 \Rightarrow non-malleable.
- Later we will see that every homomorphic encryption scheme is malleable, and hence cannot be IND-CCA2.
- Highest security level possible: IND-CCA1. (?)

Homomorphic Encryption

Fontaine and Galand, “A survey of homomorphic encryption for nonspecialists,” EURASIP Journal on Information Security, 2007.

RSA is homomorphic

- $\text{RSA}(m_1 \cdot m_2) = \text{RSA}(m_1) \cdot \text{RSA}(m_2)$
where \cdot is the multiplication in Z_n^* (i.e., modulo n).
- Easy to verify:
 - $\text{RSA}(m_1 \cdot m_2) = (m_1 \cdot m_2)^e$
 - $\text{RSA}(m_1) = m_1^e$
 - $\text{RSA}(m_2) = m_2^e$
 - $\text{RSA}(m_1) \cdot \text{RSA}(m_2) = m_1^e \cdot m_2^e = (m_1 \cdot m_2)^e$

Homomorphic encryption

M : message space

C : ciphertext space

\odot_M : some binary operation in M

\odot_C : some binary operation in C

Definition: An encryption scheme is **homomorphic** if for any encryption key k the encryption function E satisfies

$$E(m_1 \odot_M m_2) = E(m_1) \odot_C E(m_2)$$

for all messages $m_1, m_2 \in M$.

Comment: applicable only to deterministic encryption schemes.

ElGamal encryption is homomorphic

- $E(m_1 \cdot m_2) \leftarrow E(m_1) \cdot E(m_2)$, in the following sense:

$E(m_1) \cdot E(m_2)$ is **a** valid encryption of $m_1 m_2$.

- Verification:

If $E(m_1) = (g^{k_1}, m_1 y^{k_1})$ and $E(m_2) = (g^{k_2}, m_2 y^{k_2})$, then

$$\begin{aligned} E(m_1) \cdot E(m_2) &= (g^{k_1}, m_1 y^{k_1}) \cdot (g^{k_2}, m_2 y^{k_2}) \\ &= (g^{k_1+k_2}, m_1 m_2 y^{k_1+k_2}) \end{aligned}$$

is **an** encryption of $m m'$.

Homomorphic encryption redefined

M : message space

C : ciphertext space

\odot_M : some binary operation in M

\odot_C : some binary operation in C

Definition: An encryption scheme is **homomorphic** if for any encryption key k the encryption function E satisfies

$$E(m_1 \odot_M m_2) \leftarrow E(m_1) \odot_C E(m_2)$$

for all messages $m_1, m_2 \in M$.

Comment: \leftarrow means "an encryption can be computed from"

An equivalent definition

Definition: An encryption scheme is **homomorphic** if its encryption E and decryption D satisfy

$$m_1 \odot_M m_2 = D(E(m_1) \odot_C E(m_2))$$

for all messages $m_1, m_2 \in M$ and all encryption/decryption key pairs.

A generalized definition

Definition: An encryption scheme is **homomorphic** w.r.t \odot_M if there is a polynomial time algorithm A such that

$$E(m_1 \odot_M m_2) \leftarrow A(E(m_1), E(m_2))$$

or

$$m_1 \odot_M m_2 = D(A(E(m_1), E(m_2)))$$

for all messages $m_1, m_2 \in M$ and all encryption/decryption key pairs.

Question: How to further generalize it?

Various homomorphic encryptions

- An encryption scheme is
 - **additively** homomorphic if it is homomorphic w.r.t $+_M$
 - **multiplicatively** homomorphic if it is homomorphic w.r.t \cdot_M
 - **algebraically** homomorphic if it is homomorphic w.r.t both $+_M$ and \cdot_M
- RSA and ElGamal are **multiplicatively** homomorphic.
- Padded RSA and OAEP-RSA are **not** homomorphic.
- RSA is not IND-CPA secure; ElGamal is.

Additively homomorphic ElGamal encryption

- ElGamal encryption can be made additively homomorphic.
- Original ElGamal: $E(m) = (g^k, my^k)$.
- Now, encrypt m as $c = E(m) = (g^k, h^m y^k)$, where g, h are generators of Z_p^* .
- Decrypting c takes two steps:

$$m \xleftarrow{\text{DL}} h^m \xleftarrow{\text{ElGamal decryption}} c$$

- $E(m_1 + m_2) \leftarrow E(m_1) \cdot E(m_2)$.

- A simple application

- To vote yes or no, encode a yes-vote as $m = 1$ and a no-vote as $m = -1$.

- Encrypt m as $c = (g^k, h^m \cdot y^k)$.

- Send the encrypted vote c to a trusted party.

- All votes: $\{c_1, c_2, c_3, \dots, c_k\}$

- $\prod_{i=1}^k c_i = (g^{\sum k_i}, h^{\sum m_i} \cdot y^{\sum k_i}) \rightarrow E\left(\sum_{i=1}^k m_i \bmod (p-1)\right)$

- $D\left(\prod_{i=1}^k c_i\right) = \sum_{i=1}^k m_i \bmod (p-1) = \sum_{i=1}^k m_i$ (why?)

Yao's Millionaire Problem

- Two millionnaires, Alice and Bob, want to know who is richer without revealing their actual wealth.
- Alice is worth a millions, and Bob b millions. Q: $a < b$?
- Initially suggested and solved by Andrew Yao in 1982.
- Later latergeneralized to a problem called **Multiparty Computation**.
- Would be trivial if there is a secure encryption scheme that is homomorphic w.r.t. " $<$ ", namely,

$$m_1 < m_2 \leftarrow D\left(A\left(E(m_1), E(m_2)\right)\right)$$

Quadratic Residues

- Let $n \geq 2$ be any number .
- Quadratic residues: elements in Z_n^* which are a square.
- QR_n = the subgroup of quadratic residues in Z_n^* .
- $QNR_n = Z_n^* - QR_n = \{ \text{quadratic non-residues in } Z_n^* \}$.
- Legendre symbol:
$$\left(\frac{x}{p}\right) = \begin{cases} +1 & \text{if } [x] \in QR_p \text{ (} x \text{ is a square)} \\ -1 & \text{if } [x] \in QNR_p \text{ (not a square)} \\ 0 & \text{if } [x] = 0 \end{cases}$$
- Euler's criterion: $\left(\frac{x}{p}\right) = x^{(p-1)/2} \pmod{p}$.
- Jacobi symbol: $\left(\frac{x}{n}\right) = \left(\frac{x}{p}\right)\left(\frac{x}{q}\right)$, assuming $n = pq$.

Quadratic Residues (cont'd)

- Thus, $\left(\frac{x}{n}\right) = 1$ iff $\left(\frac{x}{p}\right) = \left(\frac{x}{q}\right) = \pm 1$.
- x is a quadratic residue in Z_n^* iff $\left(\frac{x}{p}\right) = \left(\frac{x}{q}\right) = 1$.
- $Z_n^* = QR_n \cup QNR_n = QR_n \cup QNR_n^+ \cup QNR_n^-$.
- If $\left(\frac{x}{n}\right) = -1$, then $x \in QNR_n^-$.
- If $\left(\frac{x}{n}\right) = 1$, $x \in QR_n \cup QNR_n^+$.
- **Quadratic residuosity assumption:**
Given $x \in Z_n^*$ with $\left(\frac{x}{n}\right) = 1$, it is intractable to determine whether $x \in QR_n$ or $x \in QNR_n^+$ without knowing $n = pq$.
- Knowing $n = pq$, easy to determine if $x \in QR_n$ or QNR_n^+ .

Goldwasser-Micali encryption scheme (idea)

- First probabilistic encryption scheme.
- Encrypt one bit $b \in \{0,1\}$ at a time.
- Encrypt $b = 0$ as a random number in QR_n .
- Encrypt $b = 1$ as a random number in QNR_n^+ .
- To decrypt $c = E(b)$, simply determine if $c \in \text{QR}_n$
(i.e., $\left(\frac{c}{p}\right) = \left(\frac{c}{q}\right) = 1$?)

Goldwasser-Micali encryption scheme

- System setup: Alice chooses $n = pq$ and $g \in_{\mathbf{R}} \text{QNR}_n^{+1}$.
Public key: (n, g) . Private key: (p, q)
- Encryption: $E(b) = g^b r^2$, where $r \in_{\mathbf{R}} \mathbf{Z}_n^*$.
- Note: $E(b)$ is a quadratic residue iff $b = 0$.
- To decrypt $c = E(b)$, simply determine if $c \in \text{QR}_n$.

Drawback: it takes $|n|=1024$ bits to encrypt a single bit.

This scheme has an expansion of 1024.

Reducing the expansion

- Idea of Goldwasser-Micali:
 - Take a group G and a subgroup H .
 - Partition G into two parts: $M_0 = H$ and $M_1 = G \setminus H$.
 - Randomly select an element in M_b to encrypt b .
- To generalize, choose G and H such that G can be split into more parts.
- Benaloh: $k =$ small prime; $E(m) = g^m r^k$, $m \in \{0, \dots, k-1\}$;
expansion: $|n|/|k|$.
- Okamoto & Uchiyama: reduced the expansion to 3.
- Paillier: reduced the expansion to 2 using group $Z_{n^2}^*$.
- Damgard & Jurik: generalized Paillier's scheme using group $Z_{n^{s+1}}^*$, with expansion $1 + 1/s$.

Paillier's encryption scheme

- One of the most well-known homomorphic encryption.
- $G = Z_{n^2}^*$, where $n = pq$.
 - $|G| = \varphi(n^2) = n \varphi(n)$.
- $H = \{z \in G : z \text{ is an } n\text{th residue mod } n^2\}$.
 - $z = y^n \text{ mod } n^2$ for some $y \in G$.
 - H is a subgroup and $|H| = \varphi(n)$.
- Use H to divide G into n classes.
- Let $g \in G$ be any element with order a multiple of n .

- Define $f : Z_n \times Z_n^* \rightarrow Z_{n^2}^*$

$$(x, y) \rightarrow g^x y^n \bmod n^2$$
- Theorem: f is bijective.
- Each $x \in Z_n$ defines a class in $Z_{n^2}^*$, namely,

$$f(x, Z_n^*) = \{f(x, y) : y \in Z_n^*\}$$

- Encryption:
 - plaintext $m \in Z_n$
 - select a random $r \in Z_n^*$
 - ciphertext $c = g^m r^n \bmod n^2$
 - additively homomorphic

- Decryption: (private key: $n = pq$ or $\lambda(n)$)
 - ciphertext $c \in Z_{n^2}^*$
 - plaintext $m = \left[\frac{L(c^{\lambda(n)} \bmod n^2)}{L(g^{\lambda(n)} \bmod n^2)} \right] \bmod n$
 - where $L(u) = (u - 1) / n$
 - $\lambda(n)$ is the Carmichael function, i.e., the smallest integer such that $a^{\lambda(n)} \equiv 1 \pmod n$ for all $a \in Z_n^*$.
 - For $n = pq$, $\lambda(n) = \text{lcm}(p - 1, q - 1)$.
 - (In RSA, $\lambda(n)$ can be used in place of $\varphi(n)$.)

- Security:
 - Assumption: Without knowing $n = pq$, it is intractable to determine if an element $z \in \mathbb{Z}_{n^2}^*$ is an n th residue modulo n^2 .
 - If this assumption holds, Paillier's encryption scheme is semantically secure under CPA.
 - Let c be the ciphertext of either m_0 or m_1 .
 - So, either $c = g^{m_0} r_0^n \bmod n^2$ **or** $g^{m_1} r_1^n \bmod n^2$.
 - So, $cg^{-m_0} = r_0^n \bmod n^2$ **or** $g^{m_1-m_0} r_1^n \bmod n^2$.
 - c is the ciphertext of m_0 iff cg^{-m_0} is an n th residue modulo n^2 .

- Question: In the above argument, which problem is reduced to which problem?

- Additively homomorphic on Z_n :
 - Recall: $E(m) = g^m r^n \bmod n^2$, $m \in Z_n$, $r \in_{\mathbb{R}} Z_n^*$.
 - $D\left(E(m_1)E(m_2) \bmod n^2\right) = m_1 + m_2 \bmod n$.
 - $D\left(E(m)^k \bmod n^2\right) = m^k \bmod n$.
 - $D\left(E(m_1)^{m_2} \bmod n^2\right) = m_1^{m_2} \bmod n$.

- A simple application

- To vote yes or no, encode a yes vote as $m = 1$ and a no vote as $m = -1$.

- Encrypt m as $c = g^m r^n \bmod n^2$.

- Send the encrypted vote c to a trusted party.

- All votes: $\{c_1, c_2, c_3, \dots, c_k\}$

- $D\left(\prod_{i=1}^k c_i \bmod n^2\right) = \sum_{i=1}^k m_i \bmod n \Rightarrow \sum_{i=1}^k m_i$ (why?)

Fully homomorphic encryption

- At STOC'09, Craig Gentry presented a fully homomorphic encryption scheme.
- A homomorphic public-key encryption scheme S has four algorithms: KeyGen, Encrypt, Decrypt, Evaluate.
- C : a circuit.
- S is homomorphic for C if for any key pair (sk, pk) output by KeyGen, any plaintext π_1, \dots, π_t , and any ciphertext ψ_1, \dots, ψ_t with $\psi_i = \text{Encrypt}(\pi_i)$, it holds that:
$$C(\pi_1, \dots, \pi_t) = \text{Decrypt}\left(\text{Evaluate}\left(C, \psi_1, \dots, \psi_t\right)\right).$$
- S is **fully homomorphic** if it is homomorphic for all circuits.

Applications

- Protection of mobile agents
- Watermarking/fingerprinting protocols
- Electronic auction and lottery protocols
- Multiparty computation
- Oblivious transfer
- Privacy preserving data mining
- Others