04 Public Key Encryption

2021 Spring

Information Security

Teacher: Po-Wen Chi

neokent@gapps.ntnu.edu.tw

February 18, 2021

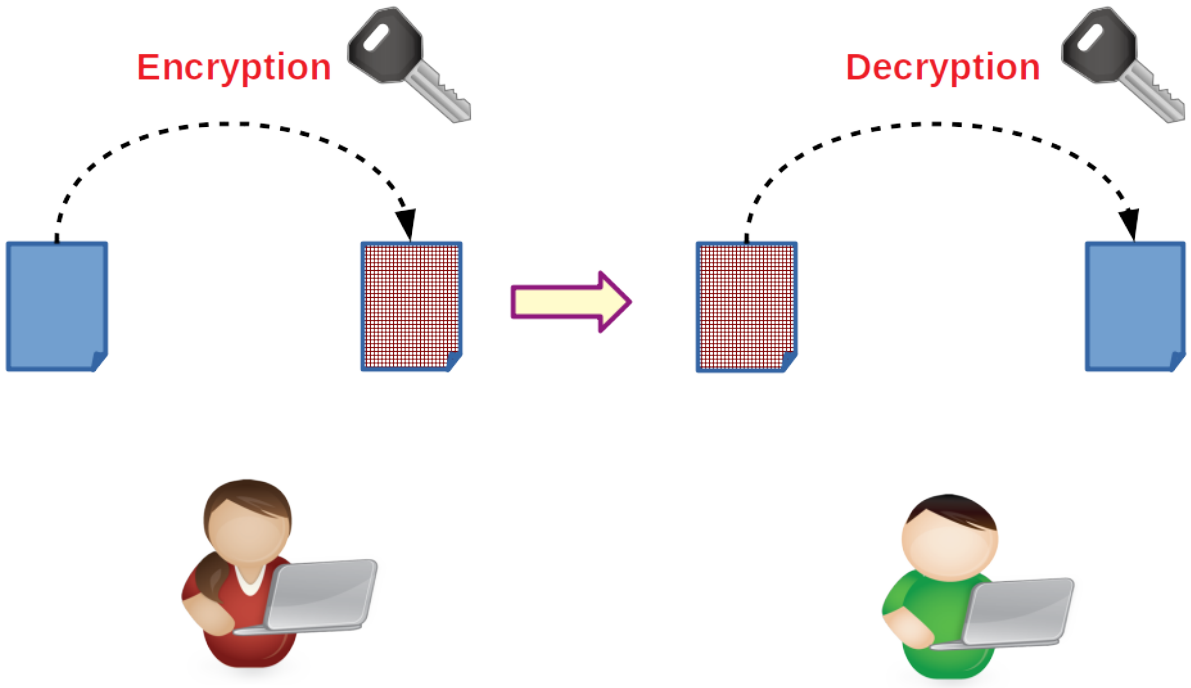
Department of Computer Science and Information Engineering,

National Taiwan Normal University

Public Key Encryption

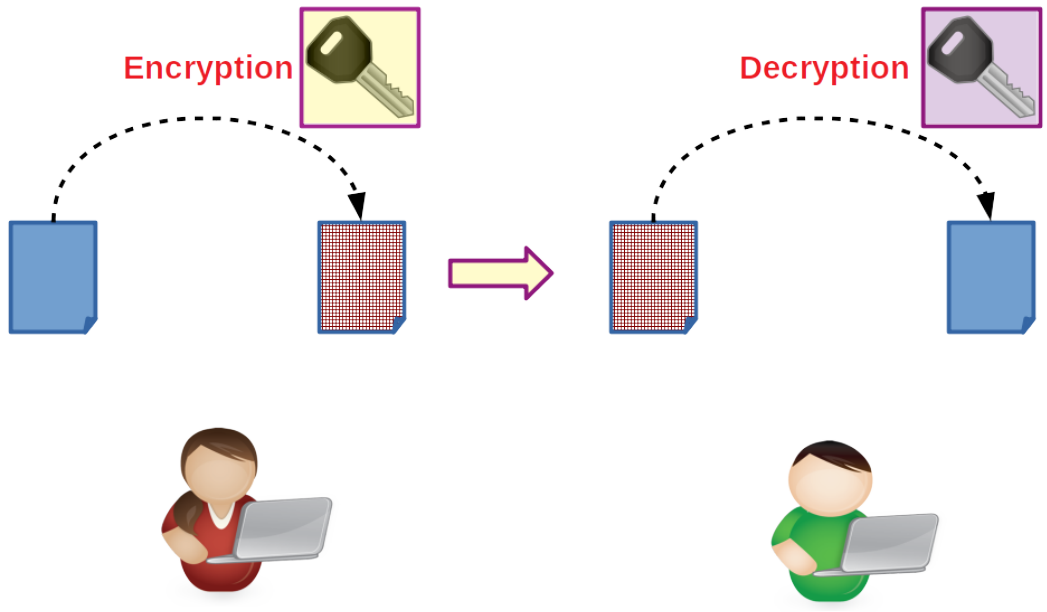
Symmetric Key Encryption

Alice and Bob use the same key for securing data exchange.



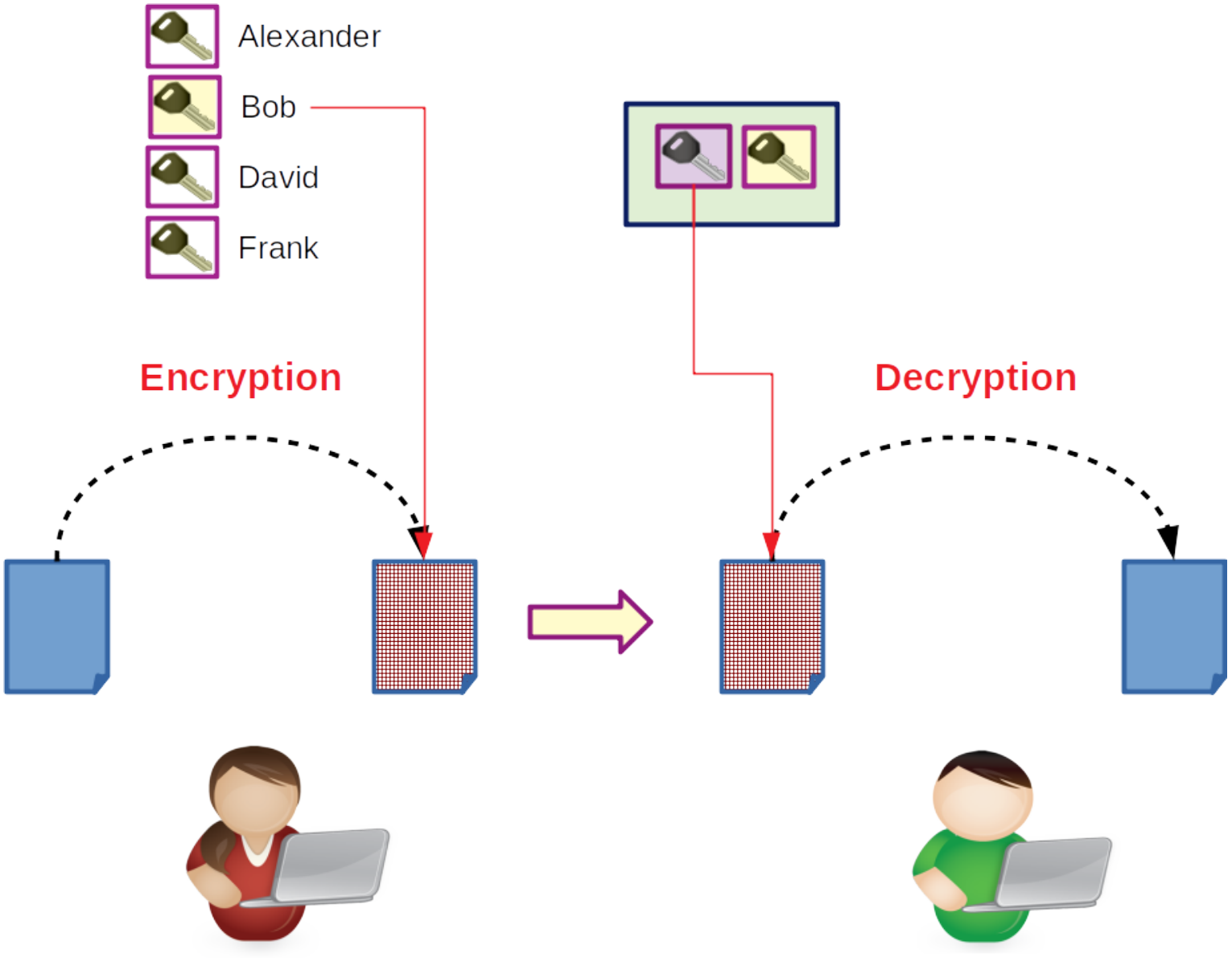
Asymmetric Key Encryption

Alice and Bob use different keys for securing data exchange.

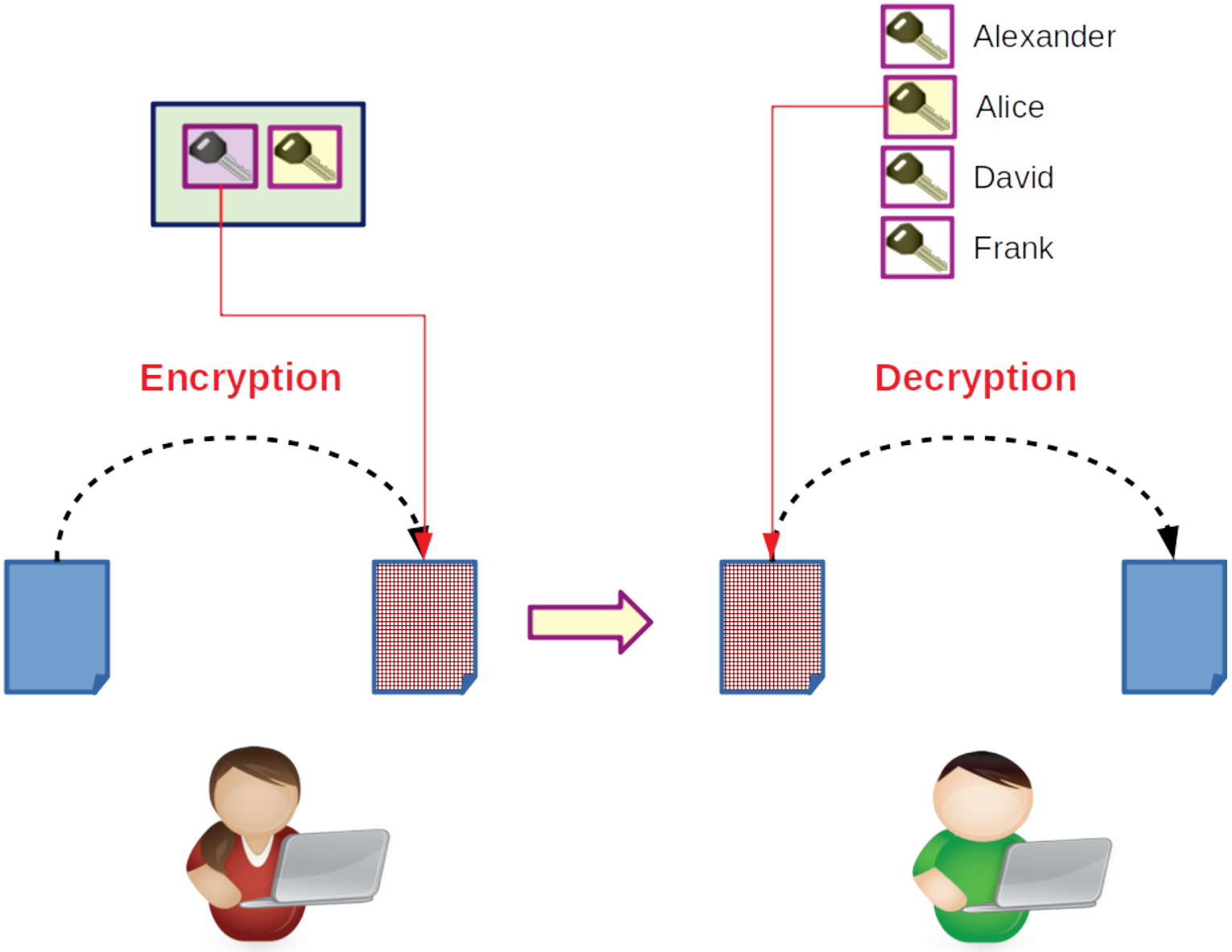


How can we use this system?

Scenario 1



Scenario 2



Symmetric vs Asymmetric

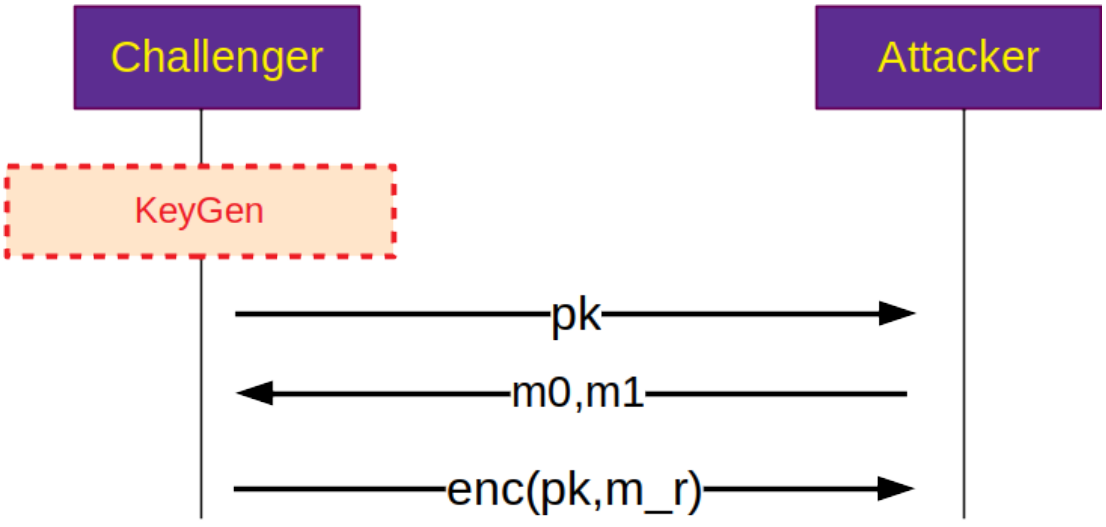
- **Symmetric Key**:
  - One key.
  - One algorithm.
  - The key should be kept secret.
- **Asymmetric Key**:
  - One pair of keys.
  - Two algorithms.
  - One of the two keys must be kept secret.

Public Key Encryption

A public key encryption system is composed of three algorithms:

1. **KeyGen**$(1^\lambda) \to (pk, sk)$.
2. **Enc**$(pk, m) \to c$.
3. **Dec**$(sk, c) \to m$.

Security Model



(KeyGen,Enc,Dec) is semantic secure (IND-CPA-secure) if for all efficient A,

$$\mathrm{Adv}_{\mathrm{SS}}[A, (\mathrm{KeyGen},\ \mathrm{Enc},\ \mathrm{Dec})] = |\Pr[\exp(0) = 1] - \Pr[\exp(1) = 1]|$$

is negligible.

Basic Math Background

Before We Start

- Much of the following should be covered by Discrete Mathematics.
- So I believe that you all have good background.
- Here I just review some important things for you.
- **Do Not Be Afraid!!**

Modulus

- Given a positive integer N which we call the *modulus*.
  - For any integer a, $a = qN + r$.
  - $q = \lfloor \frac{a}{n} \rfloor$.
  - $0 \leq r < N$.
- Examples:
  - $11 \bmod 7 = 4$.
  - $-11 \bmod 7 =?$

Modulus

- Given a positive integer N which we call the *modulus*.
  - For any integer a, $a = qN + r$.
  - $q = \lfloor \frac{a}{n} \rfloor$.
  - $0 \leq r < N$.
- Examples:
  - $11 \bmod 7 = 4$.
  - $-11 \bmod 7 = 3$.

Congruence

Congruence

Two integers a and b are said to be **congruent modulo** N if

$$a \bmod N = b \bmod N.$$

This is written as $a \equiv b (\bmod N)$.

Examples:

- $73 \equiv 4 (\bmod 23)$.
- $21 \equiv -9 (\bmod 10)$.

Why Do We Need Modular?

- Because there is **no infinity** in the real world.
- We want to make a boundary for all operation outputs.
- With a modulus N, all elements are bounded in 0 to $N - 1$.
- So we have a set with static size.

Why Do We Need Modular?

- Because there is **no infinity** in the real world.
- We want to make a boundary for all operation outputs.
- With a modulus N, all elements are bounded in 0 to $N-1$.
- So we have a set with static size.
- For convenience, we define the set

$$\mathbb{Z}/N\mathbb{Z} = \{0, \ldots, N-1\}.$$

as the set of remainder modulo N.

Why Do We Need Modular?

- Because there is **no infinity** in the real world.
- We want to make a boundary for all operation outputs.
- With a modulus N, all elements are bounded in 0 to $N-1$.
- So we have a set with static size.
- For convenience, we define the set

$$\mathbb{Z}/N\mathbb{Z} = \{0, \ldots, N-1\}.$$

  as the set of remainder modulo N.
- Someone prefers $\mathbb{Z}_N$ for $\mathbb{Z}/N\mathbb{Z}$.

Why Do We Need Modular?

- Because there is **no infinity** in the real world.
- We want to make a boundary for all operation outputs.
- With a modulus N, all elements are bounded in 0 to $N-1$.
- So we have a set with static size.
- For convenience, we define the set

$$\mathbb{Z}/N\mathbb{Z} = \{0, \ldots, N-1\}.$$

  as the set of remainder modulo N.
- Someone prefers $\mathbb{Z}_N$ for $\mathbb{Z}/N\mathbb{Z}$.
- For any set S, $\#S$ denotes the number of element in S.
- $\#(\mathbb{Z}_N) =$??

Properties of Congruences

1. $a \equiv b \pmod{N}$ implies $b \equiv a \pmod{N}$.
2. $a \equiv b \pmod{N}$ and $b \equiv c \pmod{N}$ imply $a \equiv c \pmod{N}$.
3. $a \equiv 0 \pmod{N}$ then $N|a$.
4. $a \equiv b \pmod{N}$ if $N|(a - b)$.

Divisibility

The notation $b|a$ means b divides a and we say that b is a **divisor** of a.

Modular Arithmetic

- Can we perform arithmetic operations within the set?

Modular Arithmetic

- Can we perform arithmetic operations within the set? Sure!
- Modular arithmetic:
  1. $[(a \bmod N) + (b \bmod N)] \bmod N = (a + b) \bmod N$.
  2. $[(a \bmod N) - (b \bmod N)] \bmod N = (a - b) \bmod N$.
  3. $[(a \bmod N) \times (b \bmod N)] \bmod N = (a \times b) \bmod N$.
- Actually, it just like clock arithmetic.

Properties of Modular Arithmetic in $\mathbb{Z}_N$

- **Commutative Law**

$$(a + b) \bmod N = (b + a) \bmod N$$

$$(a \times b) \bmod N = (b \times a) \bmod N$$

- **Associative Law**

$$(a + b) + c \bmod N = a + (b + c) \bmod N$$

$$(a \times b) \times c \bmod N = a \times (b \times c) \bmod N$$

- **Distributive Law**

$$c \times (a + b) \bmod N = (c \times a) + (c \times b) \bmod N$$

Properties of Modular Arithmetic in $\mathbb{Z}_N$

- **Identities**

$$(0 + a) \bmod N = a \bmod N$$

$$(1 \times a) \bmod N = a \bmod N$$

- **Additive Inverse**
  - For each $a \in \mathbb{Z}_N$, there exists a b that $a + b \equiv 0 \bmod N$.

Properties of Modular Arithmetic in $\mathbb{Z}_N$

- **Identities**

$$(0 + a) \bmod N = a \bmod N$$

$$(1 \times a) \bmod N = a \bmod N$$

- **Additive Inverse**
  - For each $a \in \mathbb{Z}_N$, there exists a b that $a + b \equiv 0 \bmod N$.
- **Multiplicative Inverse**: ??

Group

Group

A group is a set with an operation on its element which

1. **Is closed**.
2. **Has an identity**.
3. **Is associative**.
4. **Every element has an inverse**.

A group which is commutative is called **abelian group**.

Group

Group

A group is a set with an operation on its element which

1. **Is closed**.
2. **Has an identity**.
3. **Is associative**.
4. **Every element has an inverse**.

A group which is commutative is called **abelian group**.

Quiz:

- Is $(\mathbb{N}, +)$ a group? .

Group

Group

A group is a set with an operation on its element which

1. **Is closed**.
2. **Has an identity**.
3. **Is associative**.
4. **Every element has an inverse**.

A group which is commutative is called **abelian group**.

Quiz:

- Is $(\mathbb{N}, +)$ a group? Yes, identity is 0.
- Is $(\mathbb{Q}, \times)$ a group? .

Group

Group

A group is a set with an operation on its element which

1. **Is closed**.
2. **Has an identity**.
3. **Is associative**.
4. **Every element has an inverse**.

A group which is commutative is called **abelian group**.

Quiz:

- Is $(\mathbb{N}, +)$ a group? Yes, identity is 0.
- Is $(\mathbb{Q}, \times)$ a group? No, but $(\mathbb{Q} \setminus \{0\}, \times)$ is.
- Is $(\mathbb{Z}_{12}, +)$ a group?

Group

Group

A group is a set with an operation on its element which

1. **Is closed**.
2. **Has an identity**.
3. **Is associative**.
4. **Every element has an inverse**.

A group which is commutative is called **abelian group**.

Quiz:

- Is $(\mathbb{N}, +)$ a group? Yes, identity is 0.
- Is $(\mathbb{Q}, \times)$ a group? No, but $(\mathbb{Q} \setminus \{0\}, \times)$ is.
- Is $(\mathbb{Z}_{12}, +)$ a group? Yes, identity is 0.
- Is $(\mathbb{Z}_{12} \setminus \{0\}, \times)$ a group?

Group

Group

A group is a set with an operation on its element which

1. **Is closed**.
2. **Has an identity**.
3. **Is associative**.
4. **Every element has an inverse**.

A group which is commutative is called **abelian group**.

Quiz:

- Is $(\mathbb{N}, +)$ a group? Yes, identity is 0.
- Is $(\mathbb{Q}, \times)$ a group? No, but $(\mathbb{Q} \setminus \{0\}, \times)$ is.
- Is $(\mathbb{Z}_{12}, +)$ a group? Yes, identity is 0.
- Is $(\mathbb{Z}_{12} \setminus \{0\}, \times)$ a group? No.

$(\mathbb{Z}_{12} \setminus \{0\}, \times)$ is Not a Group!

$\mathbb{Z}_{12} \setminus \{0\} = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11\}$.

Suppose the multiplicative identity is **1**.

$$
\begin{array}{ll}
 & 1 \times 1 \equiv 1 \bmod 12, \\
2 \times ?? \equiv 1 \bmod 12, & 3 \times ?? \equiv 1 \bmod 12, \\
4 \times ?? \equiv 1 \bmod 12, & 5 \times 5 \equiv 1 \bmod 12, \\
6 \times ?? \equiv 1 \bmod 12, & 7 \times 7 \equiv 1 \bmod 12, \\
8 \times ?? \equiv 1 \bmod 12, & 9 \times ?? \equiv 1 \bmod 12, \\
10 \times ?? \equiv 1 \bmod 12, & 11 \times 11 \equiv 1 \bmod 12
\end{array}
$$

Modular Multiplicative Inverse

Modular Multiplicative Inverse

A modular multiplicative inverse of an integer a is an integer x such that the product ax is congruent to 1 with respect to the modulus N.

$$a \cdot x \equiv 1 \bmod N.$$

Example: $2^{-1} = 4 \bmod 7$.

Modular Multiplicative Inverse

Modular Multiplicative Inverse

A modular multiplicative inverse of an integer a is an integer x such that the product ax is congruent to 1 with respect to the modulus N.

$$a \cdot x \equiv 1 \bmod N.$$

Example: $2^{-1} = 4 \bmod 7$.

Please do not tell me that $2^{-1} = 0.5$.

Modular Multiplicative Inverse

Modular Multiplicative Inverse

A modular multiplicative inverse of an integer a is an integer x such that the product ax is congruent to 1 with respect to the modulus N.

$$a \cdot x \equiv 1 \bmod N.$$

Example: $2^{-1} = 4 \bmod 7$.

Please do not tell me that $2^{-1} = 0.5$.

The inverse of a only exists when a and N are coprime. We will prove it later.

Cyclic Group

Cyclic Group

A group $(G, \cdot)$ is cyclic if every element in G is g^k where g is a fixed element in G.

- g is called a generator.
- Note g^k means a repeated application of group operator.

Example: The generator of $(\mathbb{Z}, +)$ is 1.

Ring

Ring

A ring is a set with two operations, usually denoted by $+$ and $\cdot$, which satisfies

1. $(R, +)$ **is a abelian group**.
2. $(R, \cdot)$ **is closed**.
3. $(R, \cdot)$ **is associative**.
4. $(R, \cdot)$ **has an identity**.
5. $(R, +, \cdot)$ **satisfies the distributive law**.

We can denote a ring by $(R, +, \cdot)$.

What is the difference between $+$ and $\cdot$ in a ring?

1. $\cdot$ may has no inverse.
2. $\cdot$ may not be commutative.

Ring

- If multiplication is **commutative**, we say the ring is **commutative**.
- Example:
  - $\mathbb{Z}_N$ is a ring.
- **Important Note**:
  - $+, \cdot$ are two operations and are not necessarily addition and multiplication as you know. Though in general we still call them addition and multiplication.

Field

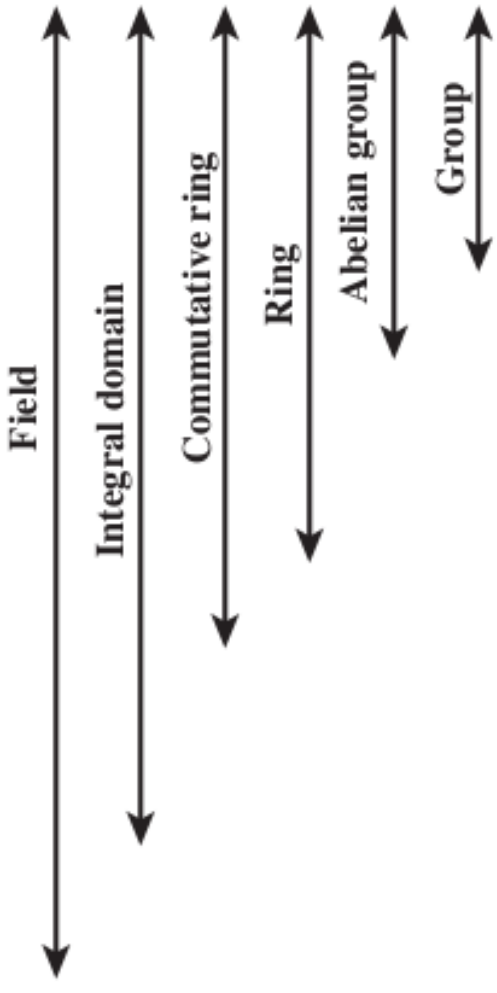
Field

A field is a set with two operations $(G, +, \cdot)$ such that

- $(G, +)$ is an abelian group with identity denoted by 0.
- $(G \setminus \{0\}, \cdot)$ is an abelian group.
- $(G, +, \cdot)$ satisfies the distributive law.

That is, a field is a **commutative ring** where every non-zero element has a multiplicative inverse.

Group, Ring, Field



(A1) Closure under addition:	If a and b belong to S, then $a + b$ is also in S
(A2) Associativity of addition:	$a + (b + c) = (a + b) + c$ for all a, b, c in S
(A3) Additive identity:	There is an element 0 in R such that $a + 0 = 0 + a = a$ for all a in S
(A4) Additive inverse:	For each a in S there is an element $-a$ in S such that $a + (-a) = (-a) + a = 0$
(A5) Commutativity of addition:	$a + b = b + a$ for all a, b in S
(M1) Closure under multiplication:	If a and b belong to S, then ab is also in S
(M2) Associativity of multiplication:	$a(bc) = (ab)c$ for all a, b, c in S
(M3) Distributive laws:	$a(b + c) = ab + ac$ for all a, b, c in S $(a + b)c = ac + bc$ for all a, b, c in S
(M4) Commutativity of multiplication:	$ab = ba$ for all a, b in S
(M5) Multiplicative identity:	There is an element 1 in S such that $a1 = 1a = a$ for all a in S
(M6) No zero divisors:	If a, b in S and $ab = 0$, then either $a = 0$ or $b = 0$
(M7) Multiplicative inverse:	If a belongs to S and $a \neq 0$, there is an element a^{-1} in S such that $aa^{-1} = a^{-1}a = 1$

Problem

Given a and b, when the equation

$$a \cdot x = b \bmod N$$

has a solution?

Example:

1. $7 \cdot x = 3 \bmod 143$, $x =$
2. $11 \cdot x = 3 \bmod 143$, $x =$
3. $11 \cdot x = 22 \bmod 143$, $x =$

Problem

Given a and b, when the equation

$$a \cdot x = b \bmod N$$

has a solution?

Example:

1. $7 \cdot x = 3 \bmod 143$, $x = 123$.
2. $11 \cdot x = 3 \bmod 143$, $x =$
3. $11 \cdot x = 22 \bmod 143$, $x =$

Problem

Given a and b, when the equation

$$a \cdot x = b \bmod N$$

has a solution?

Example:

1. $7 \cdot x = 3 \bmod 143$, $x =$ 123.
2. $11 \cdot x = 3 \bmod 143$, $x =$ No solution.
3. $11 \cdot x = 22 \bmod 143$, $x =$

Problem

Given a and b, when the equation

$$a \cdot x = b \bmod N$$

has a solution?

Example:

1. $7 \cdot x = 3 \bmod 143$, $x =$ 123.
2. $11 \cdot x = 3 \bmod 143$, $x =$ No solution.
3. $11 \cdot x = 22 \bmod 143$, $x =$ 2,15,28,41,54,67,80,93,106,119,132.

It is Easy

$$a \cdot x = b \bmod N$$

- If $gcd(a, N) = 1$, then there is exactly one solution.
- If $g = gcd(a, N) \neq 1$ and g divides b, then there are g solutions.
- Otherwise, there is no solution.

If $gcd(a, N) = 1$, we say a and N are **relatively prime** or **coprime**.

How to calculate gcd??

Euclidean Algorithm

- $gcd(a, b) = gcd(b, a \bmod b)$
- 輾轉相除法
- Extended Euclidean Algorithm:

$$gcd(a, b) = d = ax + by.$$

- Example:

$$gcd(42, 30) = 6 = 42 \times (-2) + 30 \times 3.$$

- Interesting, it means that the algorithm can be done without integer overflow.
- This algorithm implies some trivial facts:
  - If $gcd(a, b) = 1$, a, b can generate every element with linear composition.
  - If $gcd(a, b) = d$, a, b can only generate d's multiples.

Modular Multiplicative Inverse

The modular multiplicative inverse of a modulo N exists if and only if a and N are coprime.

Proof:

- $\Leftarrow$

$$gcd(a, N) = 1 \Rightarrow ax + Ny = 1 \Rightarrow ax \equiv 1 \bmod N.$$

- $\Rightarrow$

$$ax \equiv 1 \bmod N \Rightarrow ax - 1 = kN \Rightarrow ax - Nk = 1 \Rightarrow gcd(a, N) = 1.$$

Euler's Totient Function

Euler's Totient Function

Euler's totient function counts the positive integers up to a given integer n that are relatively prime to n. It is written using the Greek letter $\phi(n)$ and may also be called **Euler's phi function**.

$$\phi(n) = \#\{k|1 \leq k \leq n, gcd(k, n) = 1\}$$

Quiz:

1. $\phi(11) =$
2. $\phi(12) =$

Euler's Totient Function

Euler's Totient Function

Euler's totient function counts the positive integers up to a given integer n that are relatively prime to n. It is written using the Greek letter $\phi(n)$ and may also be called **Euler's phi function**.

$$\phi(n) = \#\{k|1 \leq k \leq n, gcd(k, n) = 1\}$$

Quiz:

1. $\phi(11) =$10.
2. $\phi(12) =$

Euler's Totient Function

Euler's Totient Function

Euler's totient function counts the positive integers up to a given integer n that are relatively prime to n. It is written using the Greek letter $\phi(n)$ and may also be called **Euler's phi function**.

$$\phi(n) = \#\{k|1 \leq k \leq n, gcd(k, n) = 1\}$$

Quiz:

1. $\phi(11) =$10.
2. $\phi(12) =$4, which are 1,5,7,11.

Euler's Totient Function

If N has the prime factorization

$$N = \prod_{i=1}^{n} p_i^{e_i}$$

then

$$\phi(N) = N \prod_{i=1}^{n} (1 - \frac{1}{p_i})$$

Example:

$$\phi(12) = 12 \cdot (1 - \frac{1}{2}) \cdot (1 - \frac{1}{3}) = 4.$$

- We define the set of all invertible elements in $\mathbb{Z}_N$ by

$$\mathbb{Z}_N^* = (\mathbb{Z}/N\mathbb{Z})^* = \{x \in \mathbb{Z}_N : gcd(x, N) = 1\}$$

- Why do we need this?
  - As I talk before, the size of group means all candidates in cryptography.
  - If $gcd(a, N) \neq 1$, this implies group size reduction and will affect security.
- Special case:

$$\mathbb{Z}_p^* = (\mathbb{Z}_p) \setminus \{0\} = \{1, \ldots, p-1\}$$

Fermat's Little Theorem

Fermat's Little Theorem

Suppose p is a prime and $a \in \mathbb{Z}$, then

$$a^p \equiv a (\text{mod } p).$$

- I will not prove this since Fermat's Little Theorem is a special case of Euler's Theorem.
- This theorem implies $a^{p-1} \equiv 1 (\text{mod } p)$.

Euler's Theorem

Euler's Theorem

For every a and n that are relatively prime, then

$$a^{\phi(n)} \equiv 1 (\text{mod } n).$$

Proof

- Consider the set of integers that are less than n and are relatively prime to n:

$$R = \{x_1, x_2, \ldots, x_{\phi(n)}\}.$$

- Now multiply each element by a, modulo n.

$$S = \{(ax_1 \bmod n), (ax_2 \bmod n), \ldots, (ax_{\phi(n)} \bmod n)\}.$$

  - Because a and x_i are all relatively prime to n, all members in S are relatively prime to n.
  - There are no duplicates in S.
  - So S is a permutation of R.

Proof

$$\prod_{i=1}^{\phi(n)} (ax_i \bmod n) = \prod_{i=1}^{\phi(n)} x_i$$

$$\prod_{i=1}^{\phi(n)} ax_i \equiv \prod_{i=1}^{\phi(n)} x_i (\bmod n)$$

$$a^{\phi(n)} \prod_{i=1}^{\phi(n)} x_i \equiv \prod_{i=1}^{\phi(n)} x_i (\bmod n)$$

$$a^{\phi(n)} \equiv 1 (\bmod n)$$

Lagrange's Theorem

Lagrange's Theorem

If $(G, \cdot)$ is a group of order (size) $n = \#G$, then for all $a \in G$, we have $a^n = 1$.

This is an important theorem in group theory. However, I will not prove here since it is a little bit abstract. If you are interested in this topic, please see the following book.

`https://www.shoup.net/ntb/ntb-v2.pdf`

Why Do We Need to Know These Things?

- Encryption and decryption are actually compositions of additions and multiplications.
- All arithmetic operations should be in the same set.
  - Of course, some cryptographic techniques have special cases.
- The set size implies possible messages and ciphertexts.
- In this class, we only focus on integers. However, there are lots of different groups.
  - For example: **elliptic curves**.
  - Some of them have special features.

Why Do We Need to Know These Things?

- We also introduce some properties about modular operations since we will use them to build a cryptosystem.
- When you review these contents, please remember one important thing:
  The property belongs to group/ring/field or belongs to modular operations. If it belongs to group/ring/field, then other group/ring/field must satisfy this property; Otherwise, it only can be applied to integers.

So We are Well-Prepared for Public Key Cryptography

- Of Course Not!
- There are lots of things that you need to learn before doing cryptography researches.
- What you learn here is only enough for RSA trapdoor permutation.
  - RSA is actually a very old cryptography technique.
  - BTW, in the real world, old cryptography technique implies reliable technique.
- I will not cover all of them for several reasons:
  - I do not want to make you sleep.
  - I am not good at mathematics.

RSA Trapdoor Permutation

RSA

From left to right: Ron Rivest, Adi Shamir, Leonard Adleman.

Question

Why called RSA instead of SAR or RAS?

RSA Trapdoor Permutation

- **KeyGen**
  - Pick up two distinct random primes p, q.
  - $N = p \cdot q$. $\phi(N) = (p-1)(q-1)$.
  - Randomly pick e that $gcd(\phi(N), e) = 1$.
  - $d \equiv e^{-1} \mod \phi(N)$.
  - $pk = (e, N), sk = (d, N)$.
- **Permutation**
  - $c = E(pk, m) = m^e \mod N$.
- **Reverse Permutation with the Trapdoor**
  - $D(sk, c) = c^d \mod N = m$.

Correctness

$$
\begin{aligned}
c^d \bmod N &= m^{ed} \bmod N \\
&= m^{k\phi(N)+1} \bmod N \\
&= m^{k\phi(N)} \cdot m \bmod N \\
&= m \bmod N
\end{aligned}
$$

RSA Trapdoor Permutation Example

- **KeyGen**
  - Pick up two distinct random primes $p = 11, q = 13$.
  - $N = p \cdot q = 143$. $\phi(N) = (p-1)(q-1) = 120$.
  - Randomly pick $e = 7$ that $gcd(\phi(N), e) = 1$.
  - $d \equiv e^{-1} \bmod \phi(N) = 103$.
  - $pk = (7, 143), sk = (103, 143)$.
- **Permutation**
  - $c = E(pk, m = 15) = m^e \bmod N = 115$.
- **Reverse Permutation with the Trapdoor**
  - $D(sk, c) = c^d \bmod N = 115^{103} \bmod 143 = 15$.

Assumption: Wrong Version

Factoring

Given a composite number N which is the product of two large primes $N = p \cdot q$, the integer factorization problem is to find p and q. We assume this is a hard problem.

Assumption: Wrong Version

Factoring

Given a composite number N which is the product of two large primes $N = p \cdot q$, the integer factorization problem is to find p and q. We assume this is a hard problem.

- Many people claim that RSA is based on the factoring assumption. That is absolutely **wrong**.

Assumption: Wrong Version

Factoring

Given a composite number N which is the product of two large primes $N = p \cdot q$, the integer factorization problem is to find p and q. We assume this is a hard problem.

- Many people claim that RSA is based on the factoring assumption. That is absolutely **wrong**.
- If you can break the factoring problem, you can break RSA absolutely. However, this means Factoring is harder than RSA!

Assumption: Wrong Version

Factoring

Given a composite number N which is the product of two large primes $N = p \cdot q$, the integer factorization problem is to find p and q. We assume this is a hard problem.

- Many people claim that RSA is based on the factoring assumption. That is absolutely **wrong**.
- If you can break the factoring problem, you can break RSA absolutely. However, this means Factoring is harder than RSA!
- If you want to prove the security of RSA, you need to have a hard problem that is simpler than RSA.

RSA Assumption

RSA Assumption

The attack game runs as follows:

- The challenger computes

$$\text{RSAGen} \to (N = pq, e, d), x \xleftarrow{R} \mathbb{Z}_N, c = x^e \in \mathbb{Z}_N.$$

  Then the challenger sends (N, e, c) to $\mathcal{A}$.
- The adversary outputs $\widehat{x} \in \mathbb{Z}_N$.

$\mathcal{A}$ wins the game if $\widehat{x} = x$. We say that the RSA assumption holds if the probability that $\mathcal{A}$ wins the game is negligible.

Open Questions

1. Is RSA as hard as factoring?
2. Is factoring a NP-complete problem?

So We Have a Public Key Encryption System

- Unfortunately, that is not true.
- RSA is a trapdoor permutation instead of an encryption scheme. Why?
- **Any deterministic encryption scheme cannot be semantic secure.**

The Need for Randomized Encryption

- Suppose $\mathcal{E} = (G, E, D)$ is a public key encryption scheme where E is deterministic. Then the following adversary $\mathcal{A}$ breaks semantic security of $\mathcal{E}$:
  - $\mathcal{A}$ receives a public key pk from its challenger.
  - $\mathcal{A}$ chooses two distinct messages m_0, m_1 and sends them to the challenger. The challenger sends $c = E(pk, m_b)$ back.
  - How to get b?

The Need for Randomized Encryption

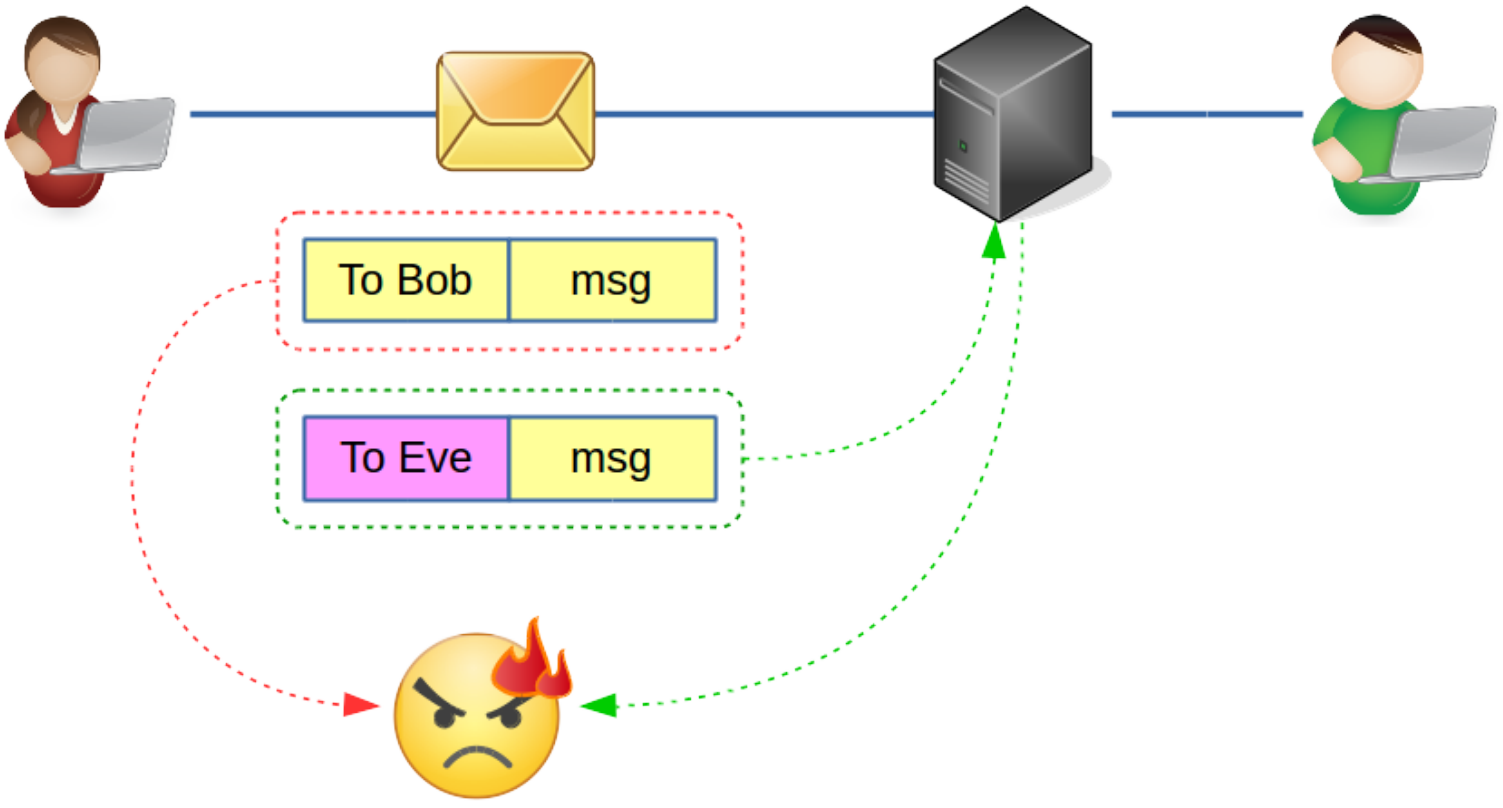
- Suppose $\mathcal{E} = (G, E, D)$ is a public key encryption scheme where E is deterministic. Then the following adversary $\mathcal{A}$ breaks semantic security of $\mathcal{E}$:
  - $\mathcal{A}$ receives a public key pk from its challenger.
  - $\mathcal{A}$ chooses two distinct messages m_0, m_1 and sends them to the challenger. The challenger sends $c = E(pk, m_b)$ back.
  - How to get b?
  - $\mathcal{A}$ simply compute $c' = E(pk, m_0)$ and check if c is equal to c'.

The Need for Randomized Encryption

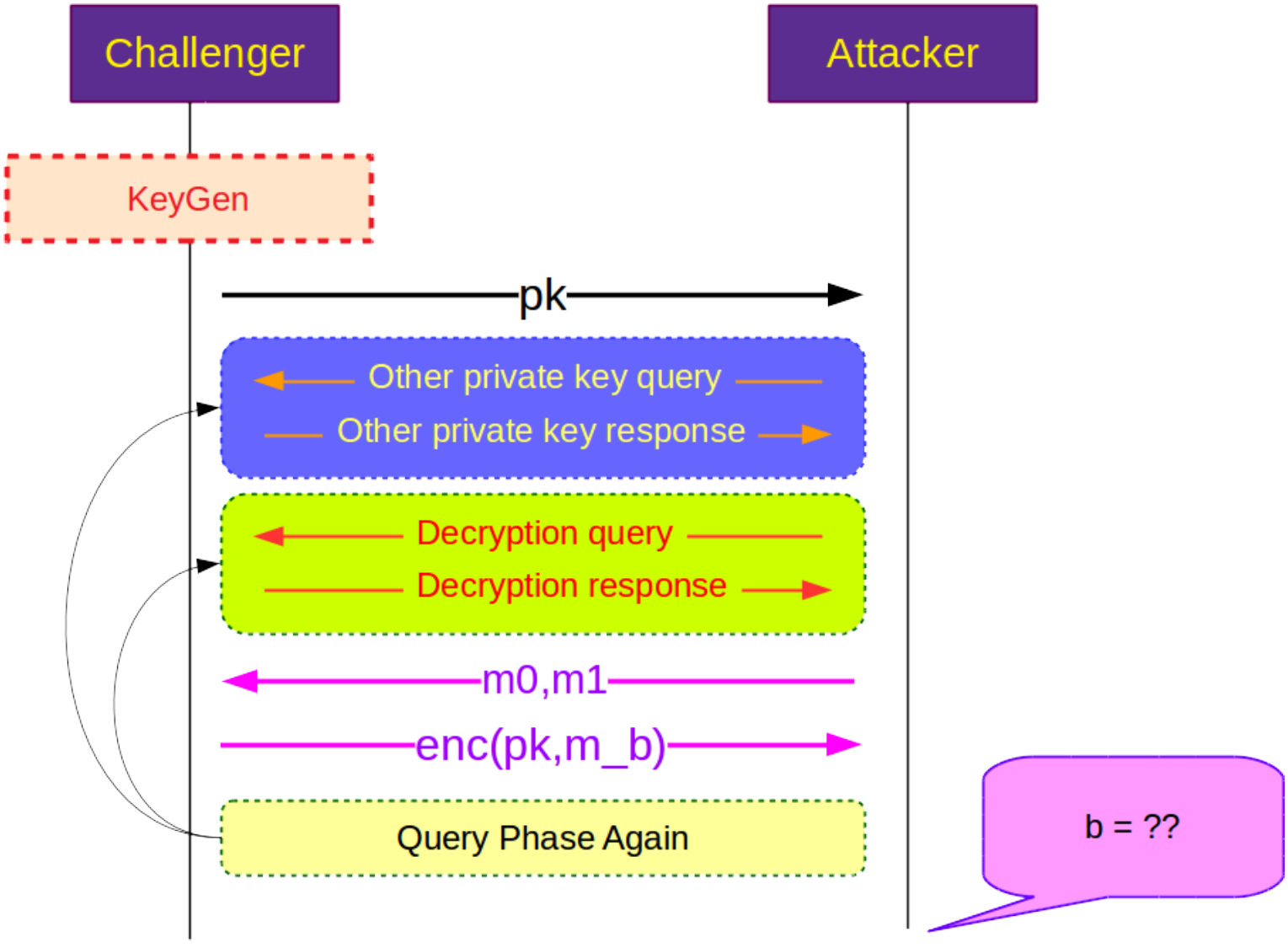
- Suppose $\mathcal{E} = (G, E, D)$ is a public key encryption scheme where E is deterministic. Then the following adversary $\mathcal{A}$ breaks semantic security of $\mathcal{E}$:
  - $\mathcal{A}$ receives a public key pk from its challenger.
  - $\mathcal{A}$ chooses two distinct messages m_0, m_1 and sends them to the challenger. The challenger sends $c = E(pk, m_b)$ back.
  - How to get b?
  - $\mathcal{A}$ simply compute $c' = E(pk, m_0)$ and check if c is equal to c'.
- Unlike the symmetric key encryption scheme, the public key is well-known.

Chosen Ciphertext Attack

Active Attack



Chosen Ciphertext Attack: Game Model



There are some variations about this model.

Quiz

RSA permutation is not CCA-secure.

Why?

Chosen Ciphertext Attack

- Chosen Ciphertext Attack security implies that the attacker is able to modify the ciphertext meaningfully.
- Solution: Let the ciphertext **unchangeable**.

RSA in Practice

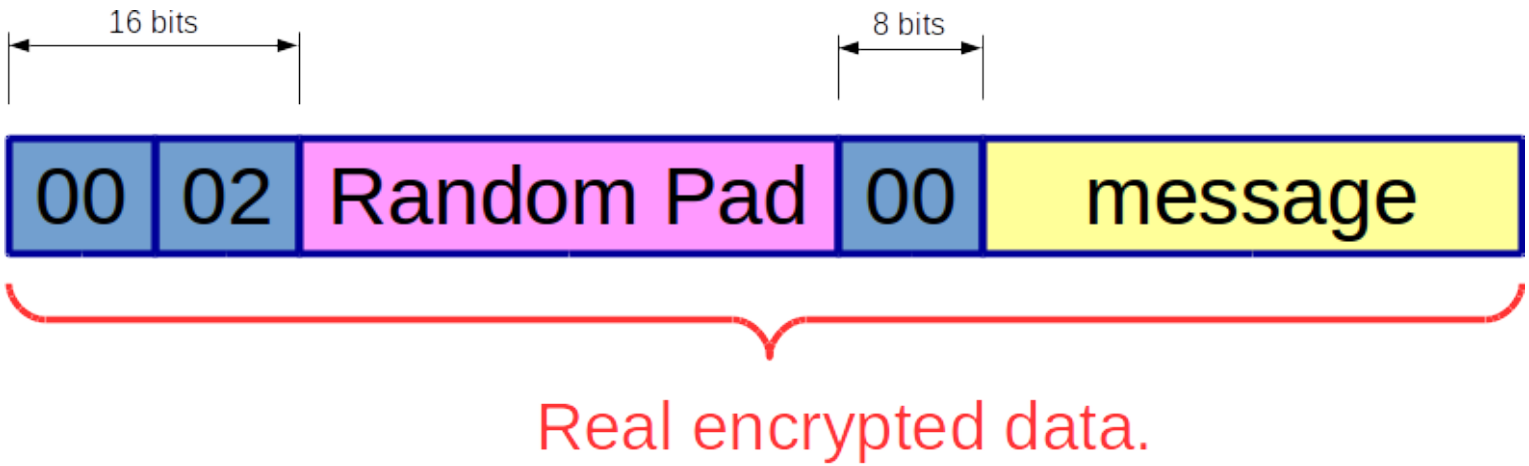
PKCS

- Again, what you learned before is not RSA encryption scheme, we can just call it RSA trapdoor permutation function.
  - I know most textbooks call it RSA encryption.
  - Dan Boneh calls it *textbook RSA*.
- PKCS: Public Key Cryptography Standards.
- These are a group of public-key cryptography standards devised and published by **RSA Security Inc**.
  - PKCS #1: RSA Cryptography Standard.
  - PKCS #3: Diffie-Hellman Key Agreement Standard.

PKCS#1 v1.5

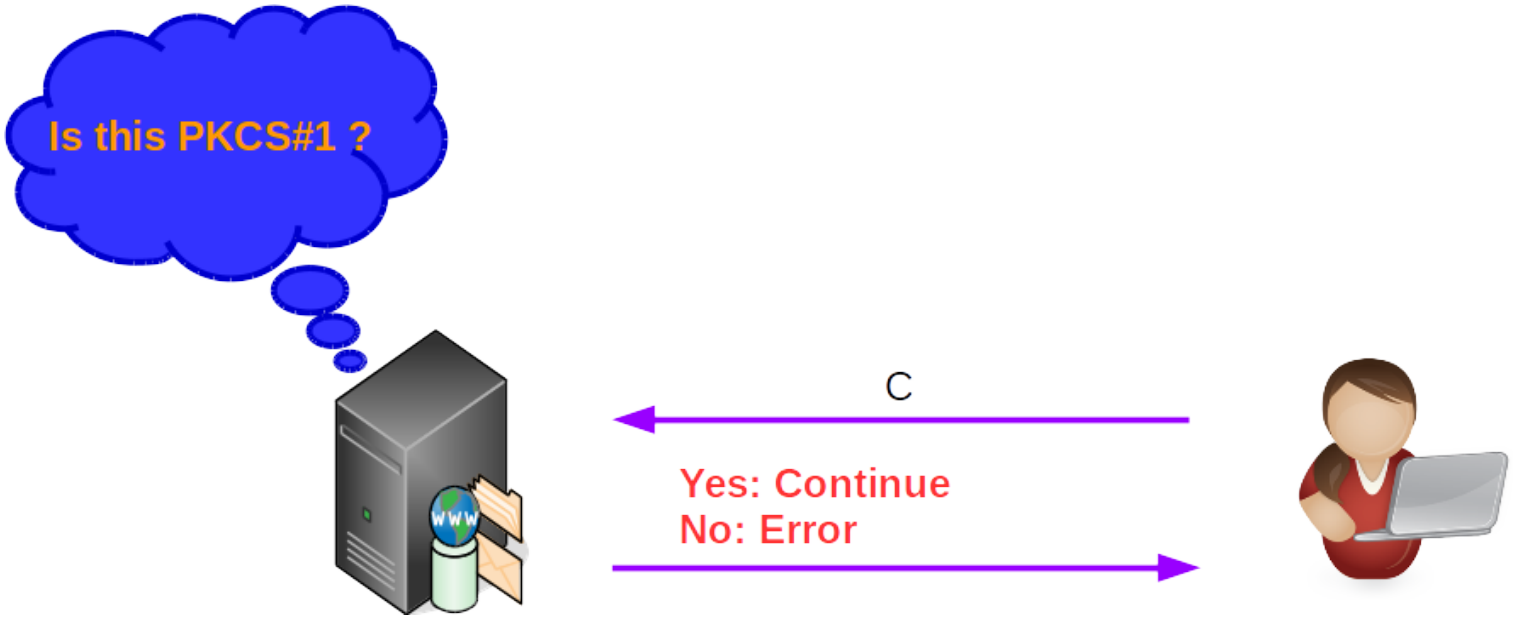
Motivation:

If you want randomness, I will give you randomness.



Widely deployed in web servers.

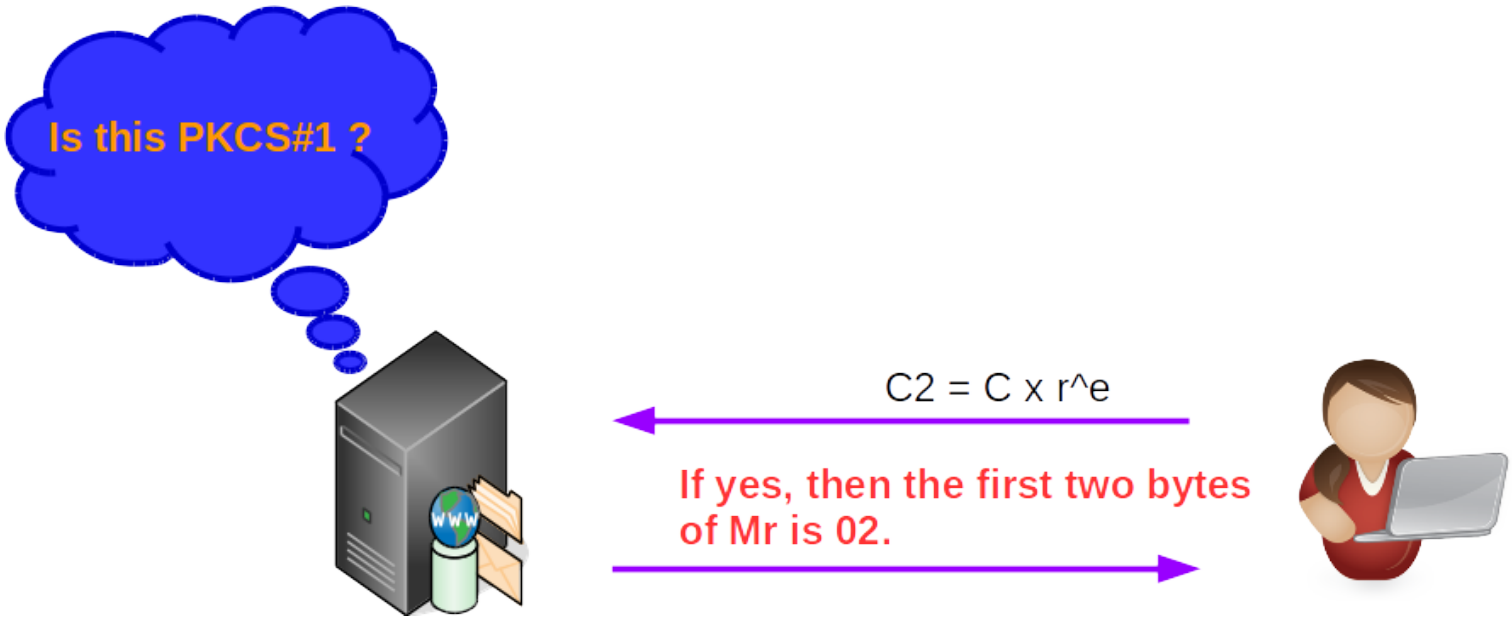
Practical Scenario



Do you think this is SECURE?

How to Attack??

Let's Attack



Sometimes we call this Oracle.

A Defense Approach

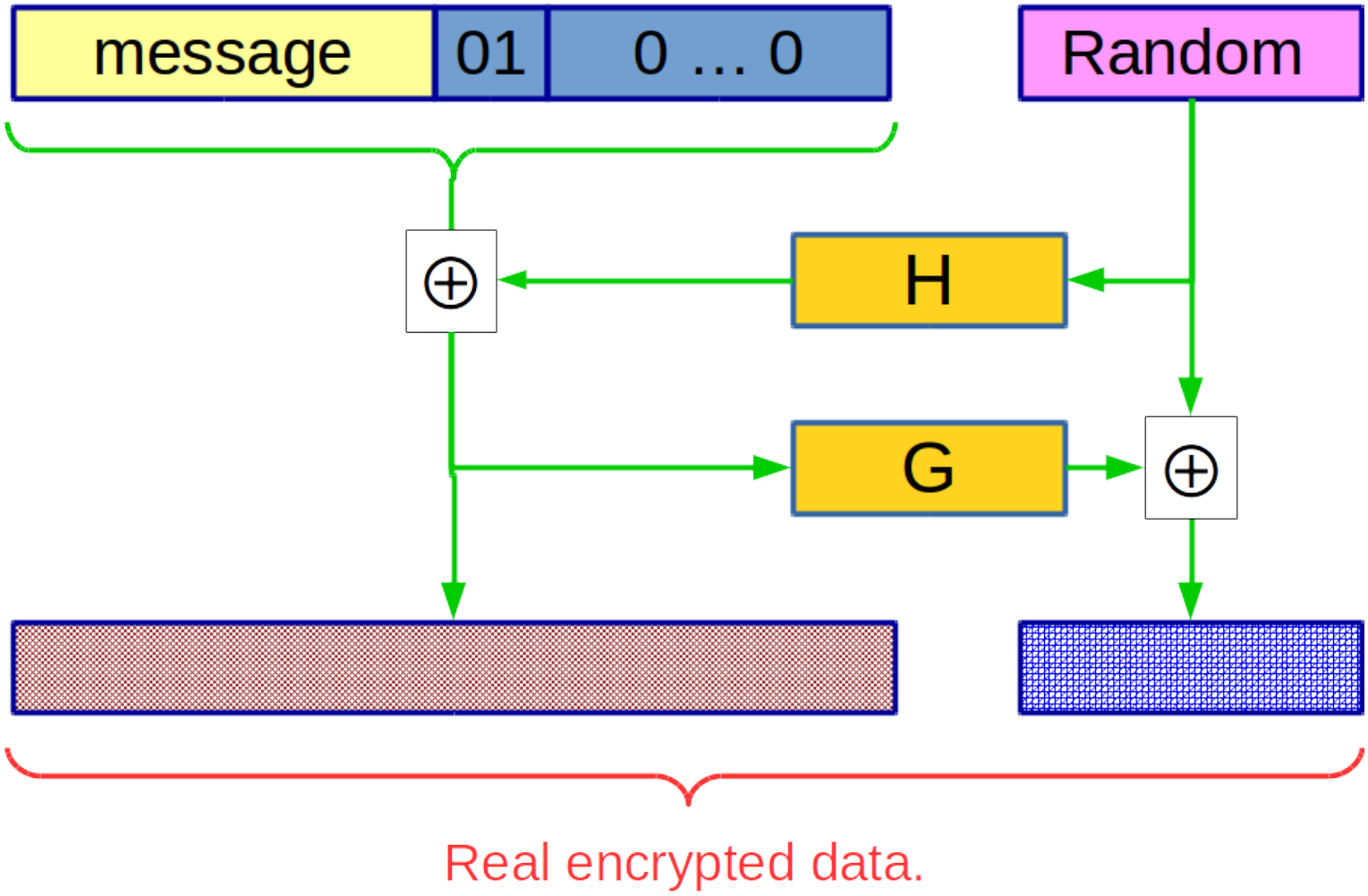
No matter success or not, always send a response:

- YES: Continue.
- No: return a random string.

Some Notes

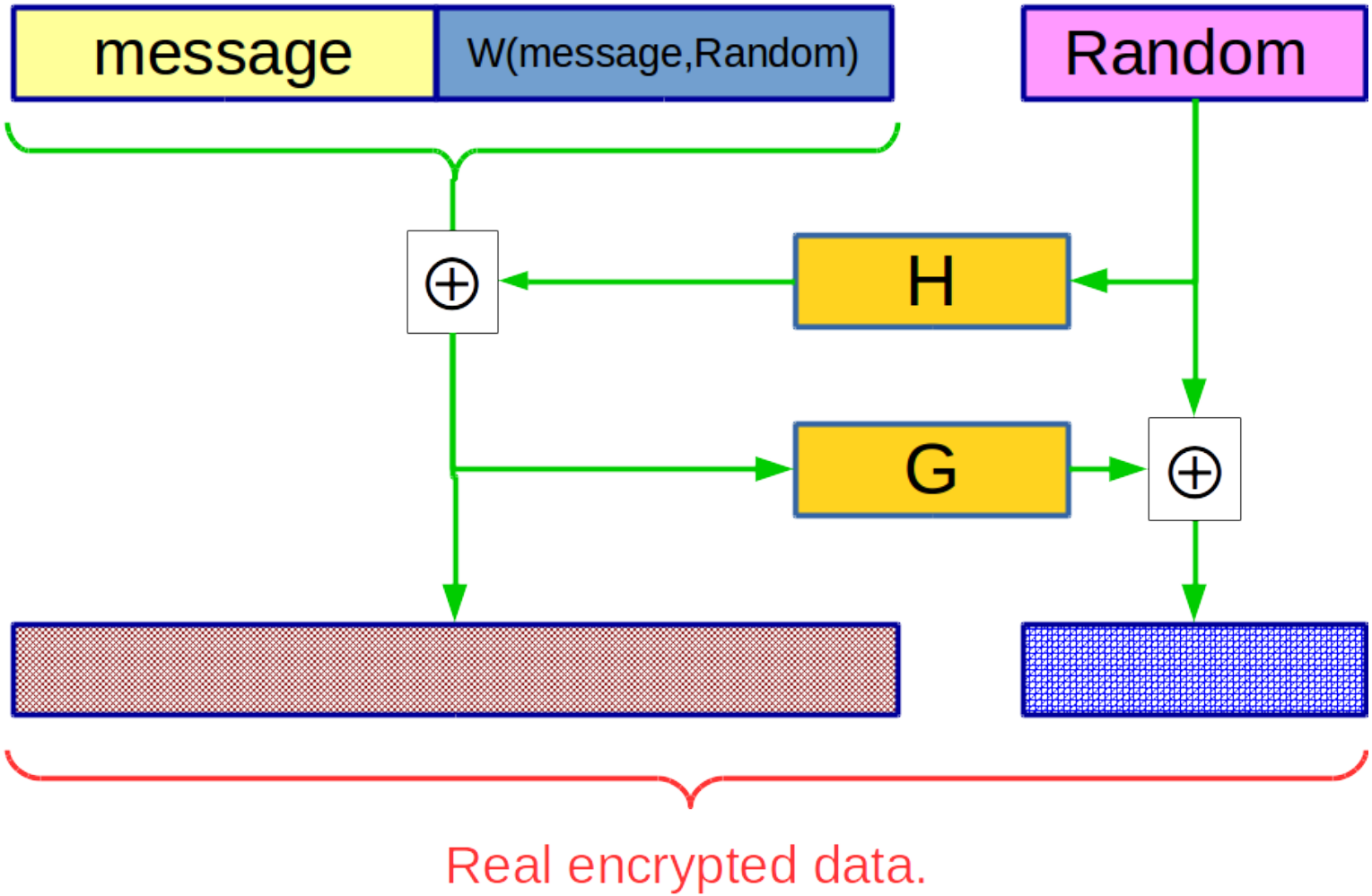
- In many cases, a security system is crashed because of wrong usage instead of crypto being broken.
- Hacker is often a crypto user instead of a math expert.

PKCS#1 v2.0: OAEP

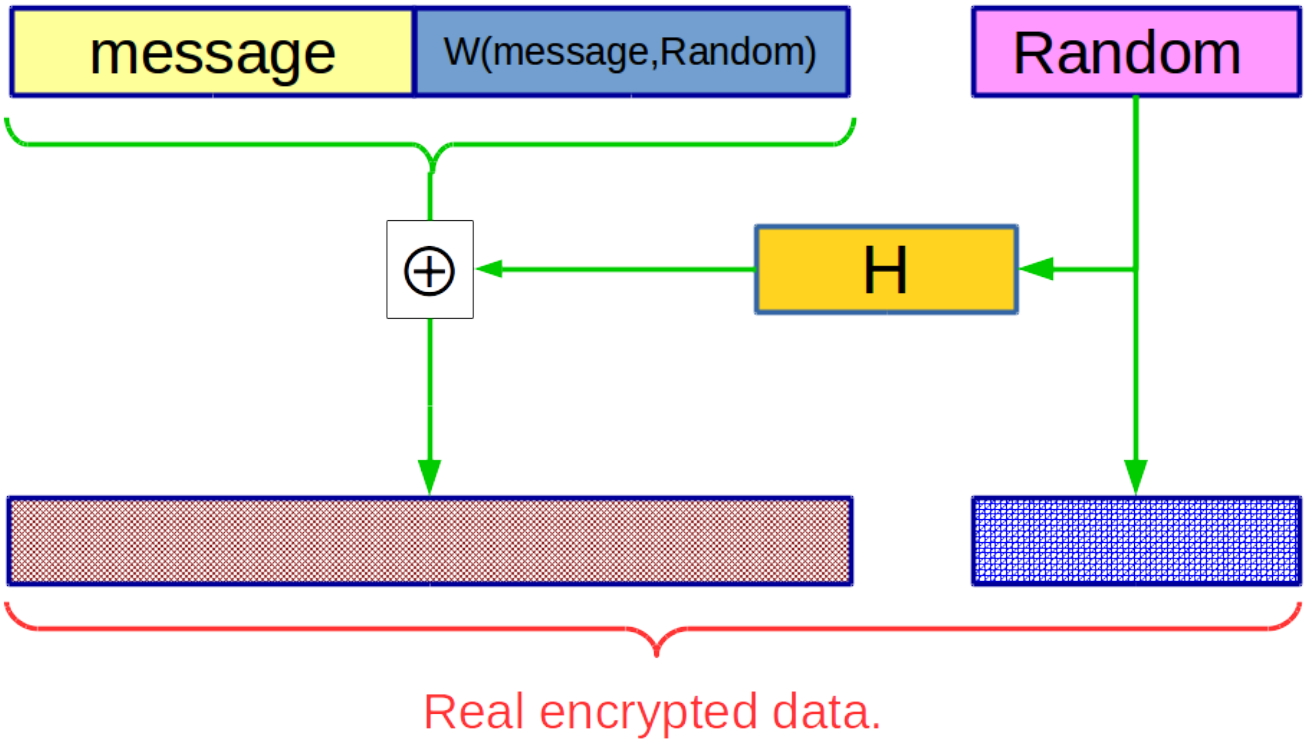


How to decrypt it??

OAEP Improvement



OAEP Improvement 2



OAEP Implementation

```
OAEP_Dec()
{
    ...
    if ( error case 1 )
    {
        goto error;
    }
    ...
    if ( error case 2 )
    {
        goto error;
    }
    ...
}
```

OAEP Implementation

```
OAEP_Dec()
{
    ...
    if ( error case 1 )
    {
        goto error;
    }
    ...
    if ( error case 2 )
    {
        goto error;
    }
    ...
}
```

Timing information leaks types of error and attackers can break the ciphertext.

OAEP Implementation

```
OAEP_Dec()
{
    ...
    if ( error case 1 )
    {
        goto error;
    }
    ...
    if ( error case 2 )
    {
        goto error;
    }
    ...
}
```

Timing information leaks types of error and attackers can break the ciphertext.

Never implement cryptographic tools yourself.

Key Length

- For most security systems, key length implies the security levels.
- It implies with the brute force attack, how many times an attacker need to try.

Key Length

- For most security systems, key length implies the security levels.
- It implies with the brute force attack, how many times an attacker need to try.
- When we say n-bits key, it means you need to try 2^n times instead of n times.

Key Length

Cipher Key Size	RSA Modulus Size
80 bits	1024 bits
128 bits	3076 bits
256 bits	15360 bits

This implies with the same security level, symmetric key encryption is faster than asymmetric key encryption.

Other Attacks

Implementation Attack

- **Timing Attack**:
  - The time it take to compute c^d can expose d.
- **Power Attack**:
  - The power consumption when computing c^d can expose d.
- **Fault Attack**:
  - A computer error when computing c^d can expose d.

Fault Attack

- A common implementation of RSA decryption:

$$\left.\begin{aligned} x_p &\equiv c^d \in \mathbb{Z}_p \\ x_q &\equiv c^d \in \mathbb{Z}_q \end{aligned}\right\} \Rightarrow x \equiv c^d \in \mathbb{Z}_N$$

- Suppose there is an error in x_q, but x_p is correct.

$$\begin{aligned} x' &\equiv c^d \in \mathbb{Z}_p \\ x' &\neq c^d \in \mathbb{Z}_q \end{aligned}$$

Fault Attack

- A common implementation of RSA decryption:

$$\left.\begin{aligned} x_p &\equiv c^d \in \mathbb{Z}_p \\ x_q &\equiv c^d \in \mathbb{Z}_q \end{aligned}\right\} \Rightarrow x \equiv c^d \in \mathbb{Z}_N$$

- Suppose there is an error in x_q, but x_p is correct.

$$\begin{aligned} x' &\equiv c^d \in \mathbb{Z}_p \\ x' &\neq c^d \in \mathbb{Z}_q \end{aligned}$$

- Then we have

$$\gcd((x')^e - c, N) = p.$$

Fault Attack

- A common implementation of RSA decryption:

$$\left.\begin{aligned} x_p &\equiv c^d \in \mathbb{Z}_p \\ x_q &\equiv c^d \in \mathbb{Z}_q \end{aligned}\right\} \Rightarrow x \equiv c^d \in \mathbb{Z}_N$$

- Suppose there is an error in x_q, but x_p is correct.

$$\begin{aligned} x' &\equiv c^d \in \mathbb{Z}_p \\ x' &\neq c^d \in \mathbb{Z}_q \end{aligned}$$

- Then we have

$$\gcd((x')^e - c, N) = p.$$

- Why?

Random is Not Enough

- RSA Setup:
  - p = random_prime_gen()
  - q = random_prime_gen()
- Suppose poor entropy at startup ...
  - p may be the same for multiple devices.

My poor experience.

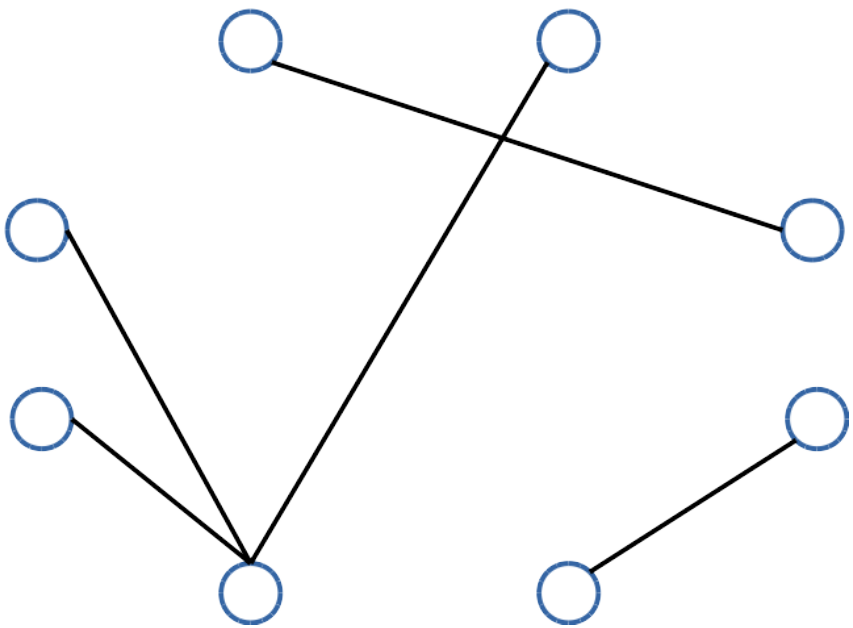
Toy Construction

How to Prove Security?

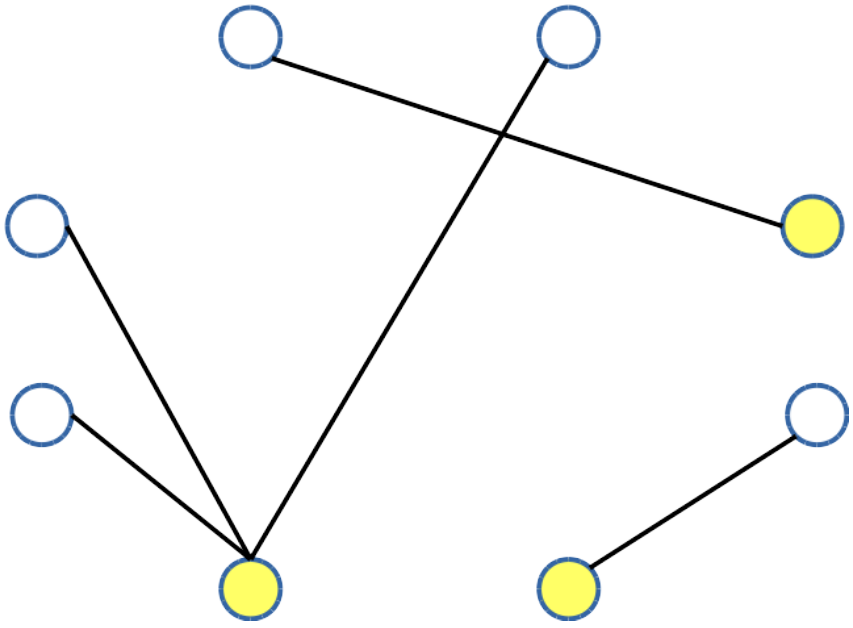
- You may find that I introduce the security game model and some backgrounds to you, but I do not prove any security.
  - For RSA trapdoor permutation, of course! Since the permutation function is definitely our assumption. How can I prove an assumption?
- Do not worry! I will give you an assignment and ask you to prove a crypto system's security.
- But Now ... I will construct a public key encryption system through an NP-complete problem and show you how it works and why it is not secure.

Perfect Codes

- A perfect code in a graph is a subset of vertices such that every vertex is in the neighborhood of one and only one vertex in the subset.
  - Not all graphs have perfect codes.
  - The problem of finding a perfect code in a graph is **NP-hard**.
- The following graph is an example. Can you find a perfect code?

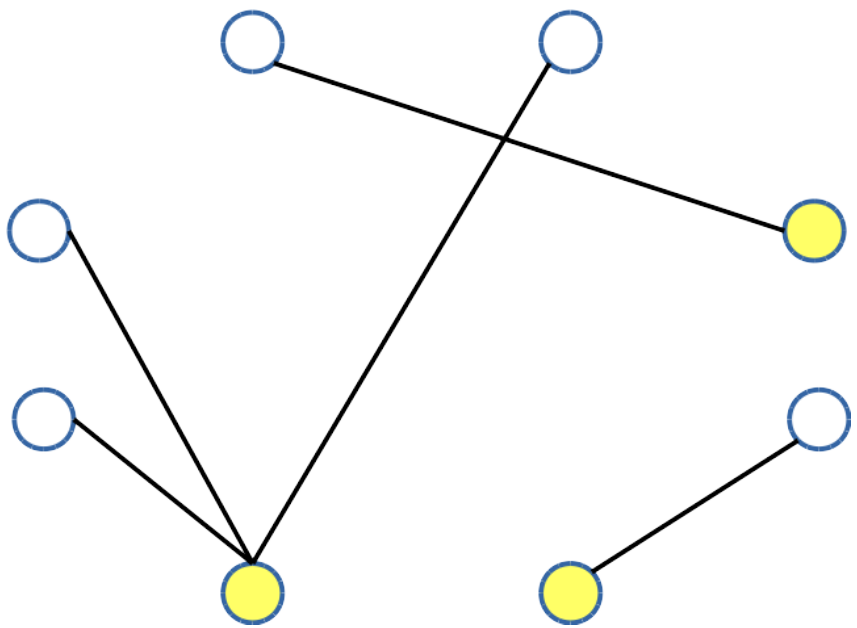


Perfect Code



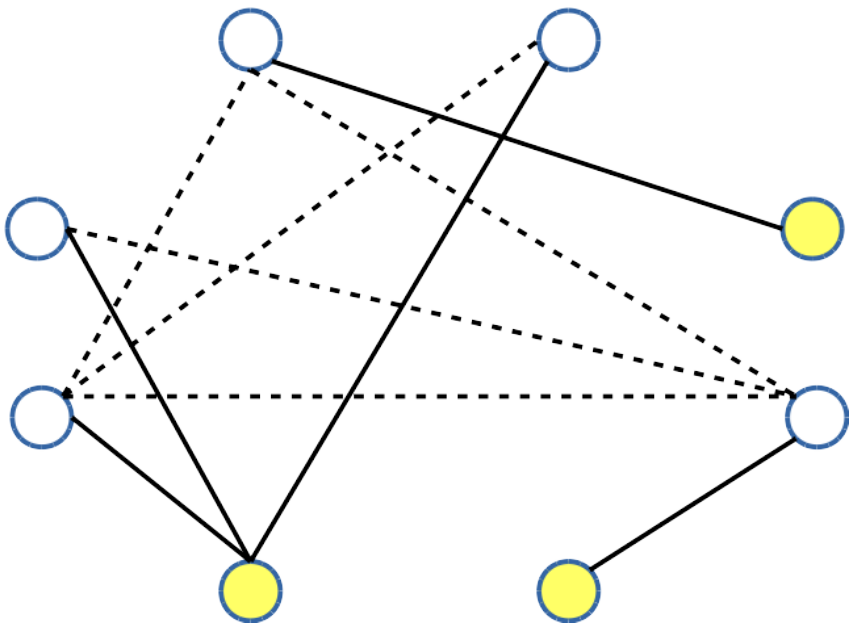
A Toy Public Key Construction

- **Key Generation**
  - Start with a simple graph $\mathcal{G}$ containing a perfect code $\mathbb{C}$.



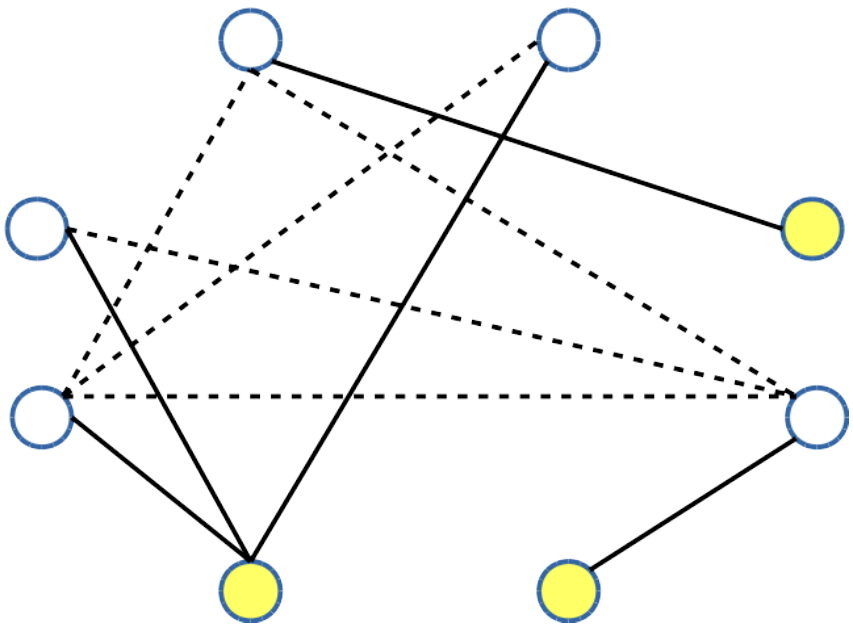
A Toy Public Key Construction

- **Key Generation**
  - Start with a simple graph $\mathcal{G}$ containing a perfect code $\mathbb{C}$.
  - Add to $\mathcal{G}$ new edges, making sure they do not touch vertices in $\mathbb{C}$. The result is $\mathcal{G}'$.



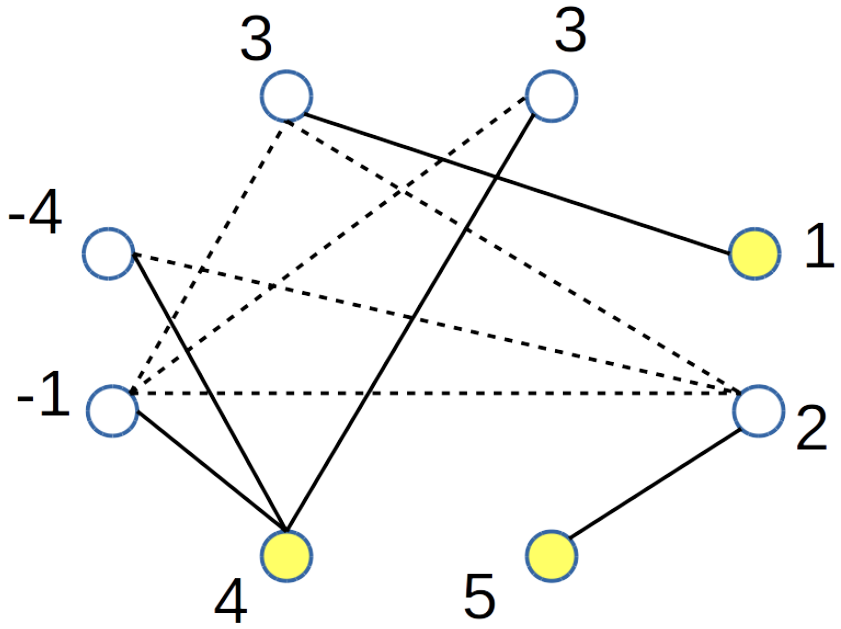
A Toy Public Key Construction

- **Key Generation**
  - Start with a simple graph $\mathcal{G}$ containing a perfect code $\mathbb{C}$.
  - Add to $\mathcal{G}$ new edges, making sure they do not touch vertices in $\mathbb{C}$. The result is $\mathcal{G}'$.
  - Public Key: $\mathcal{G}'$.
  - Private Key: $\mathbb{C}$.



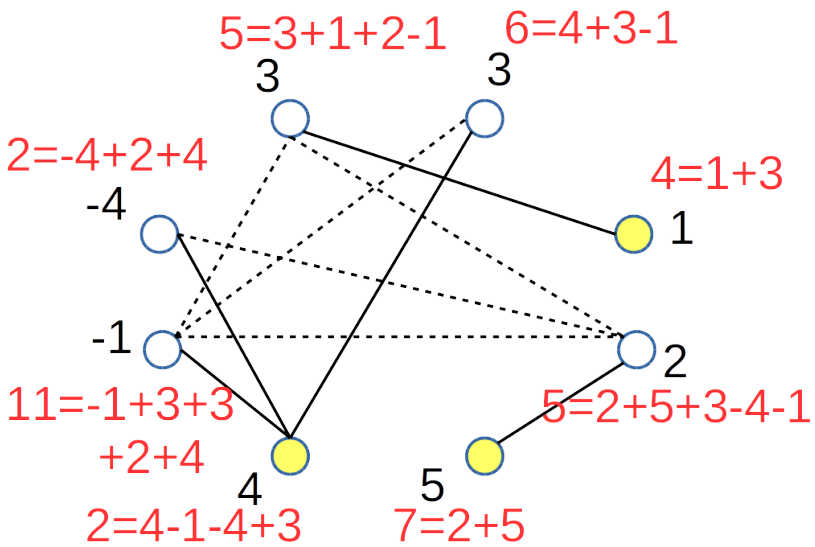
A Toy Public Key Construction

- **Encryption**:
  - To encrypt an integer, first partition it into eight shares.
  - Example: $13 = 1 + 3 + 3 - 4 - 1 + 4 + 5 + 2$.



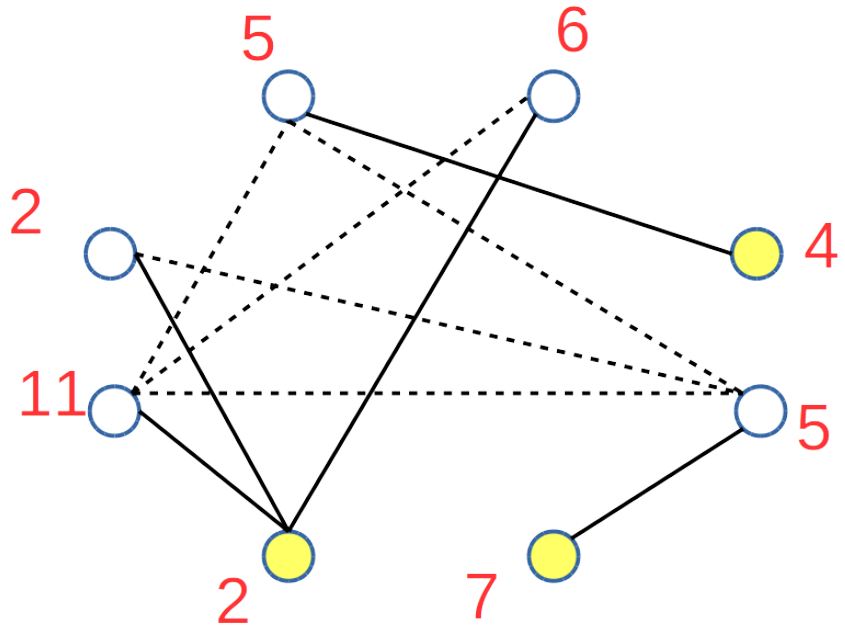
A Toy Public Key Construction

- **Encryption**:
  - To encrypt an integer, first partition it into eight shares.
  - Example: $13 = 1 + 3 + 3 - 4 - 1 + 4 + 5 + 2$.
  - Each node sums up all the values on the nodes in its neighborhood.



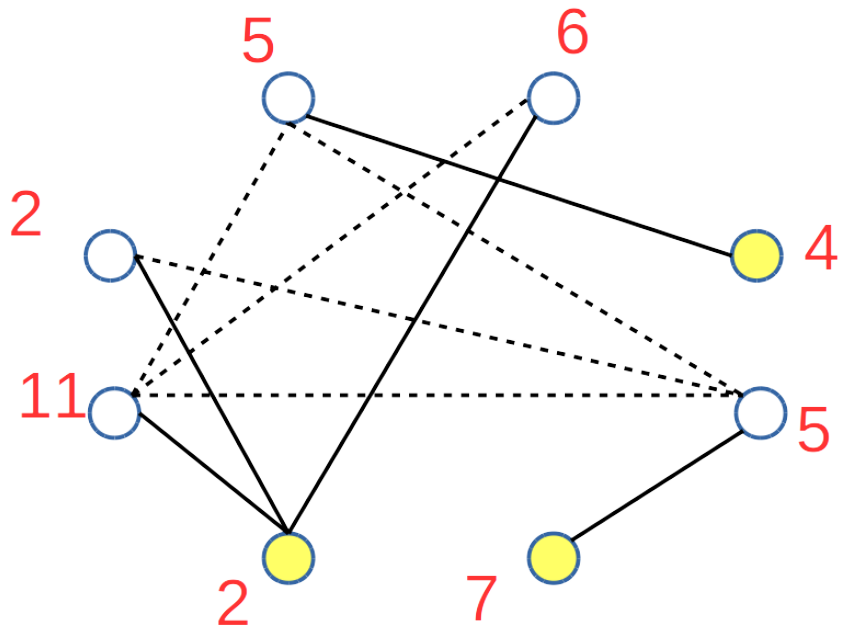
A Toy Public Key Construction

- **Encryption**:
  - To encrypt an integer, first partition it into eight shares.
  - Example: $13 = 1 + 3 + 3 - 4 - 1 + 4 + 5 + 2$.
  - Each node sums up all the values on the nodes in its neighborhood.
  - Ciphertext will be:



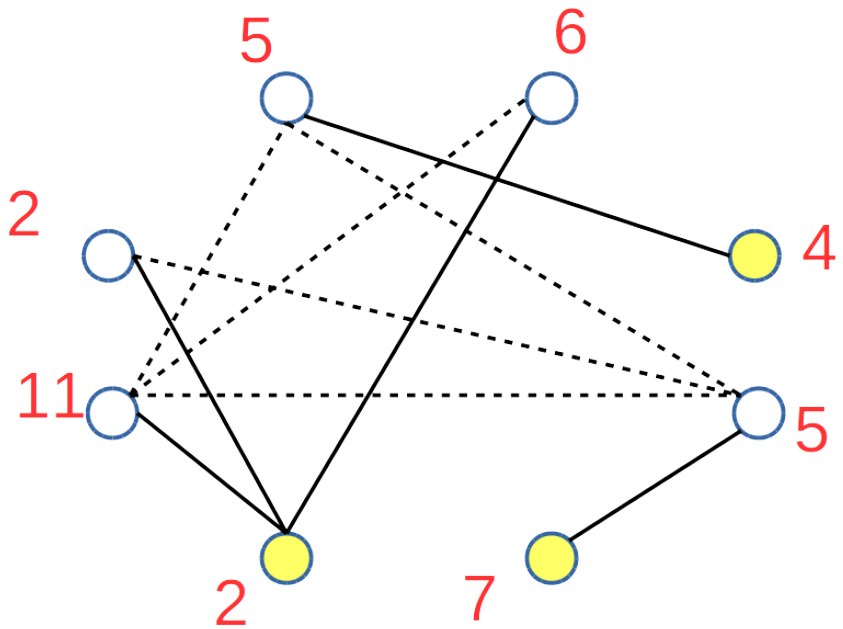
A Toy Public Key Construction

- **Decryption**:
  - Simply sum up all the values on the nodes in $\mathbb{C}$.
  - Why does it work?



A Toy Public Key Construction

- **Decryption**:
  - Simply sum up all the values on the nodes in $\mathbb{C}$.
  - Why does it work?
  - Because each node will send its value to exactly one node in $\mathbb{C}$.

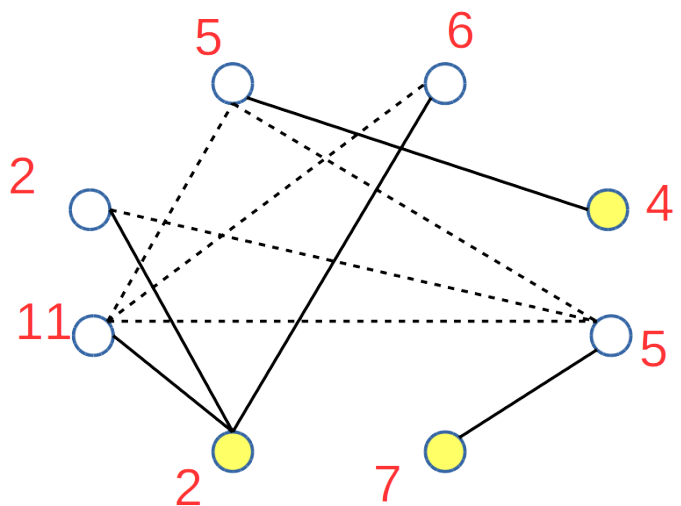


How to Crack?

Let's Break It!

- Write down the adjacency matrix of $\mathcal{G}'$.

$$A = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 \end{bmatrix}$$

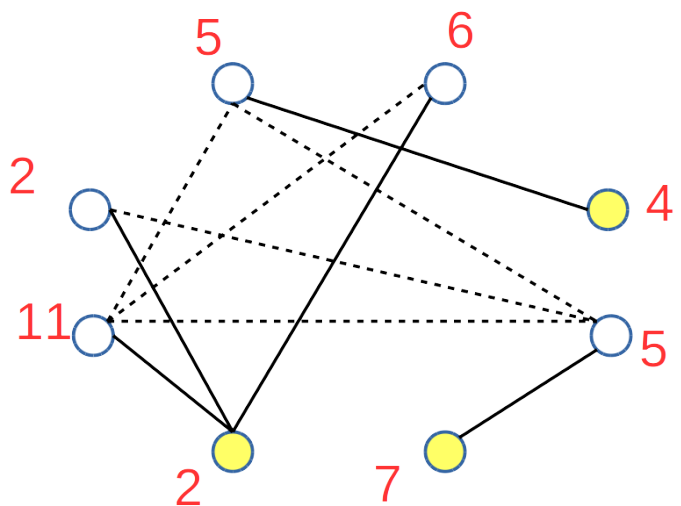


Let's Break It!

- Write down the adjacency matrix of $\mathcal{G}'$.

$$A = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 \end{bmatrix}$$

- $y = (4, 6, 5, 2, 11, 2, 7, 5)$.

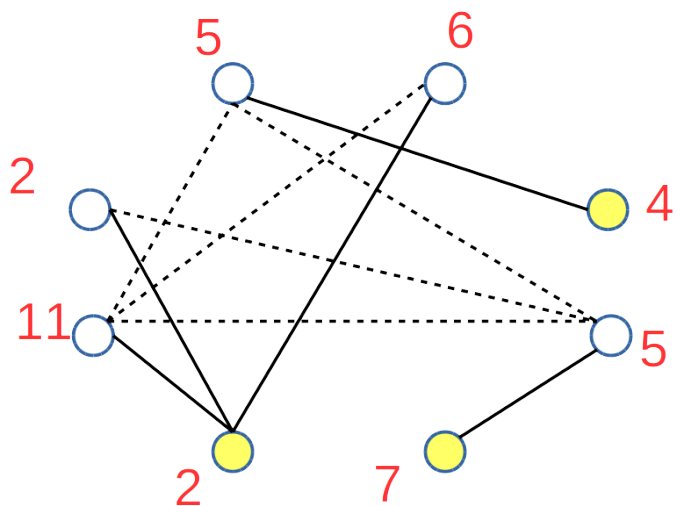


Let's Break It!

- Write down the adjacency matrix of $\mathcal{G}'$.

$$A = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 \end{bmatrix}$$

- $y = (4, 6, 5, 2, 11, 2, 7, 5)$.
- Solve $Ax = y$.

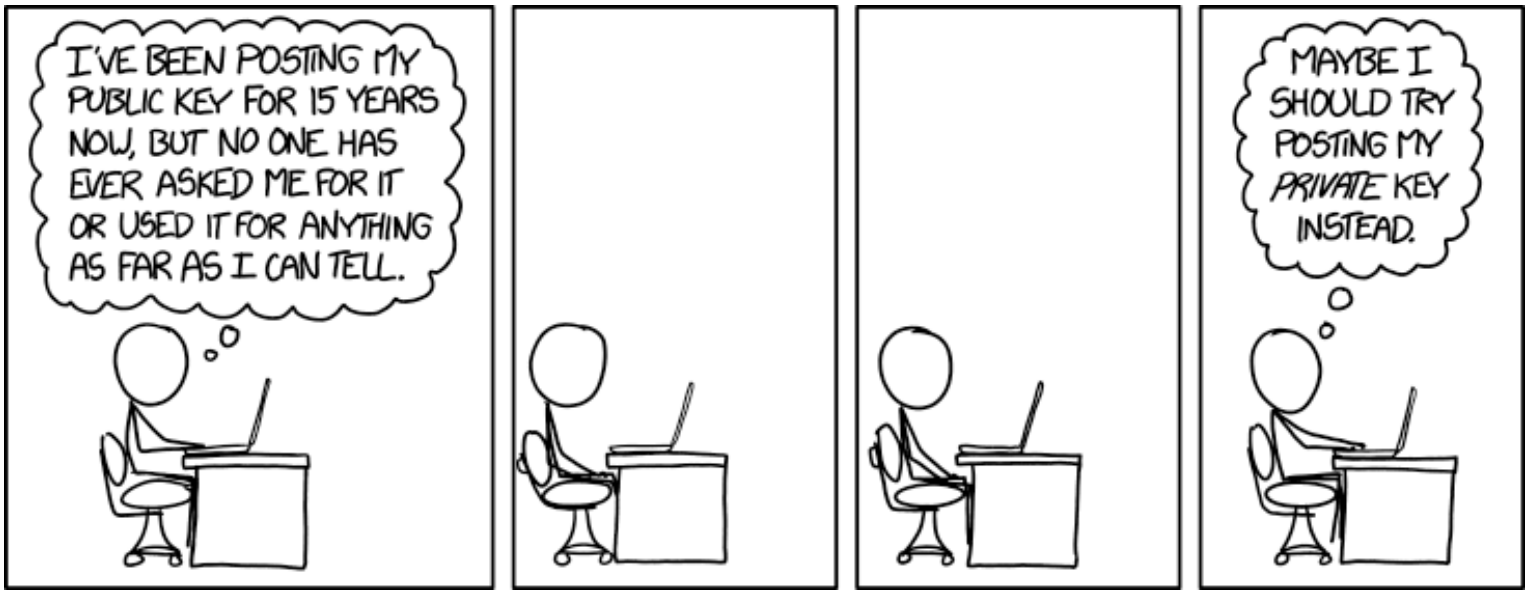


Important!

- So you can crack this encryption system.
- Do you solve the perfect code problem?

Relax

Relax


I'VE BEEN POSTING MY PUBLIC KEY FOR 15 YEARS NOW, BUT NO ONE HAS EVER ASKED ME FOR IT OR USED IT FOR ANYTHING AS FAR AS I CAN TELL.
MAYBE I SHOULD TRY POSTING MY PRIVATE KEY INSTEAD.