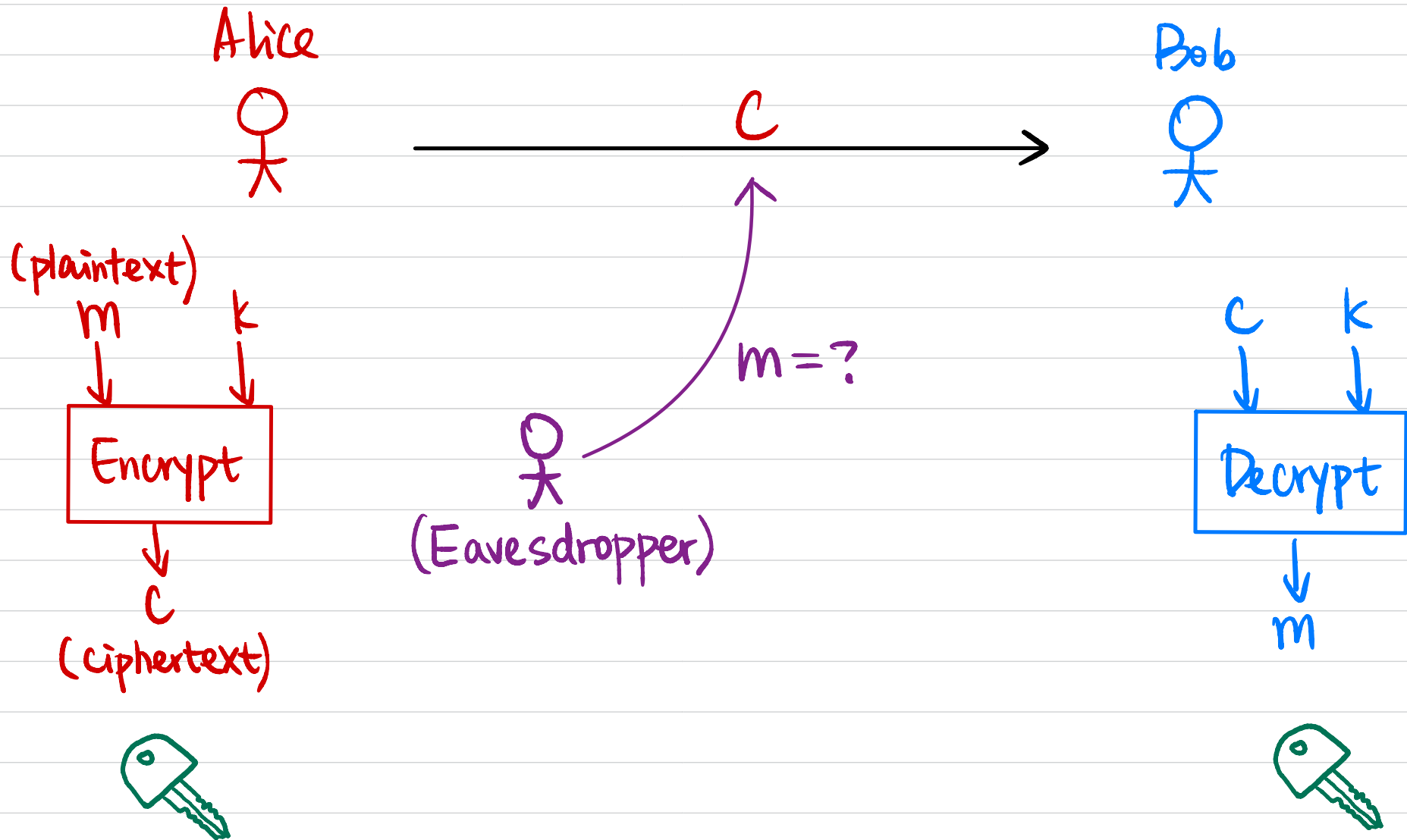


CSCI 1515 Applied Cryptography

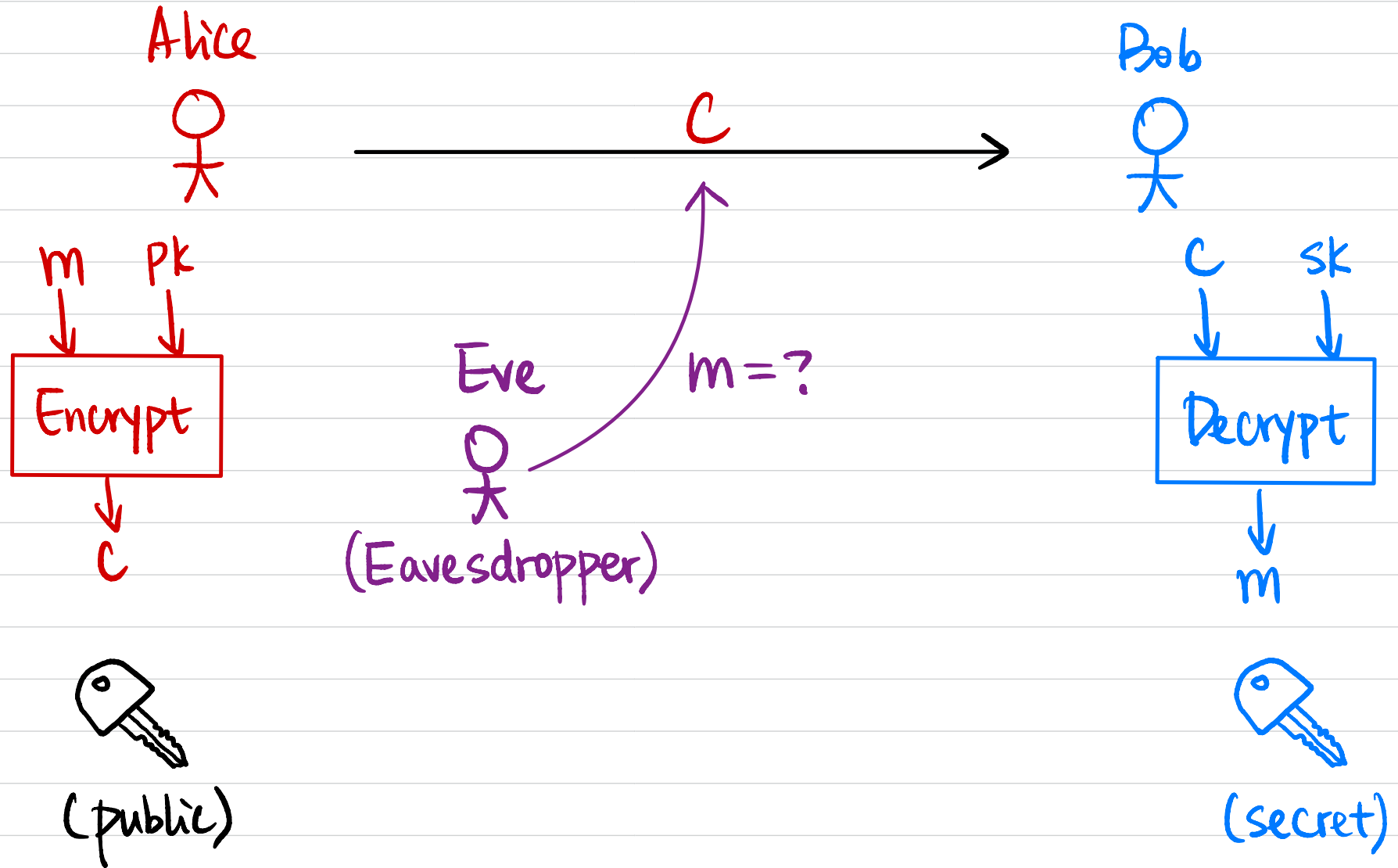
This Lecture:

- Encryption Scheme Basics
- One-Time Pad (OTP)
- Computational Assumptions
- RSA Assumption/Encryption

Message Secrecy: Symmetric-Key Encryption



Message Secrecy: Public-Key Encryption



Syntax

Symmetric-Key Encryption (SKE) Scheme $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$

$k \leftarrow \text{Gen}$

$c \leftarrow \text{Enc}(k, m)$ $\text{Enc}_k(m)$

$m := \text{Dec}(k, c)$ $\text{Dec}_k(c)$

Public-Key Encryption (PKE) Scheme $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$

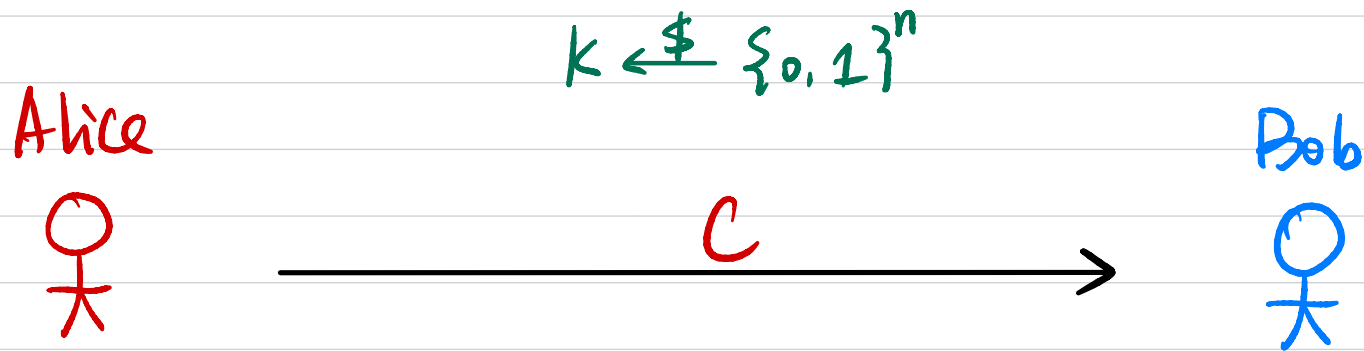
$(pk, sk) \leftarrow \text{Gen}$

$c \leftarrow \text{Enc}(pk, m)$ $\text{Enc}_{pk}(m)$

$m := \text{Dec}(sk, c)$ $\text{Dec}_{sk}(c)$

Why ever using SKE?

One-Time Pad (OTP)



Encrypt:

Secret key $k = 0100101$

$\oplus$ Plaintext $m = 1001001$

Ciphertext $c = 1101100$

Decrypt:

Secret key $k = 0100101$

$\oplus$ Ciphertext $c = 1101100$

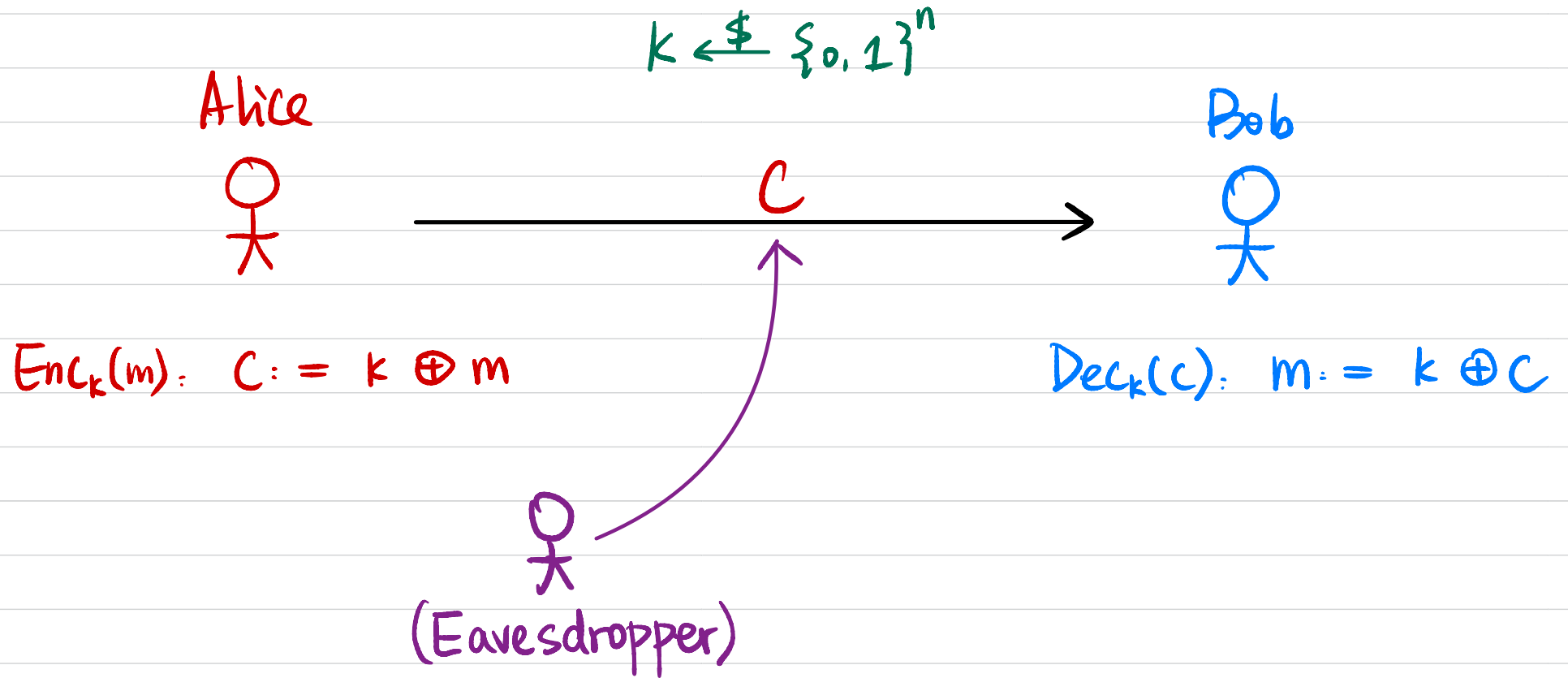
plaintext $m = 1001001$

$\oplus$	0	1
0	0	1
1	1	0

Correctness?

Security?

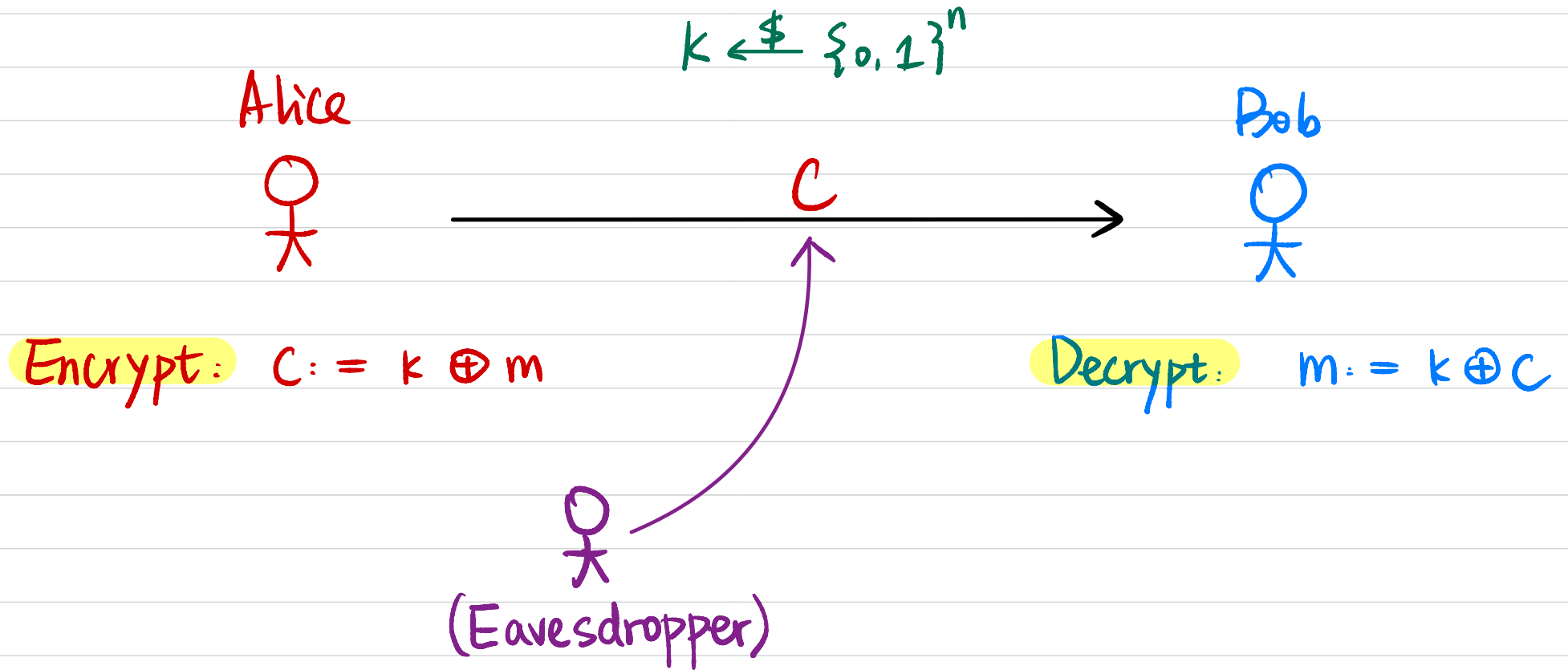
One-Time Pad (OTP)



Distribution of c ?

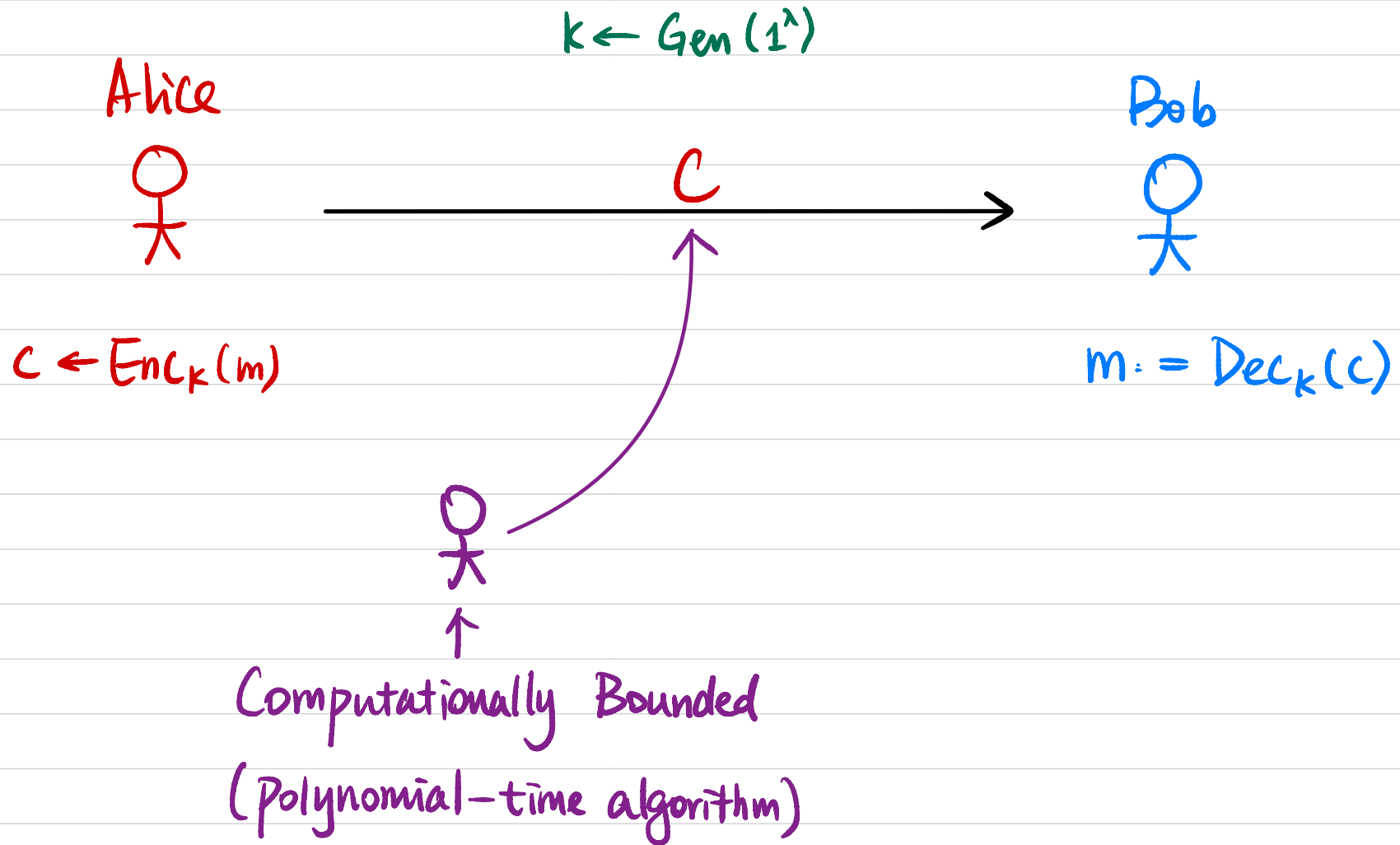
Can we re-use k ?

Shannon's Theorem



(Informal) For perfect (information-theoretic) security, $n \geq |m|$

Computational Security



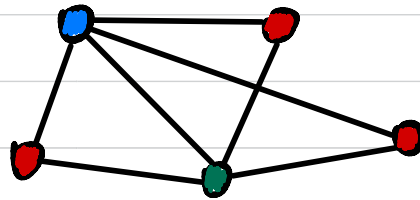
Computational Assumptions

polynomial-time algorithm: $A(x)$

Input x of length n , A 's running time $O(n^c)$ for a constant c .

NP Problem: decision problems whose solution can be verified in poly time.

Ex: Graph 3-Coloring

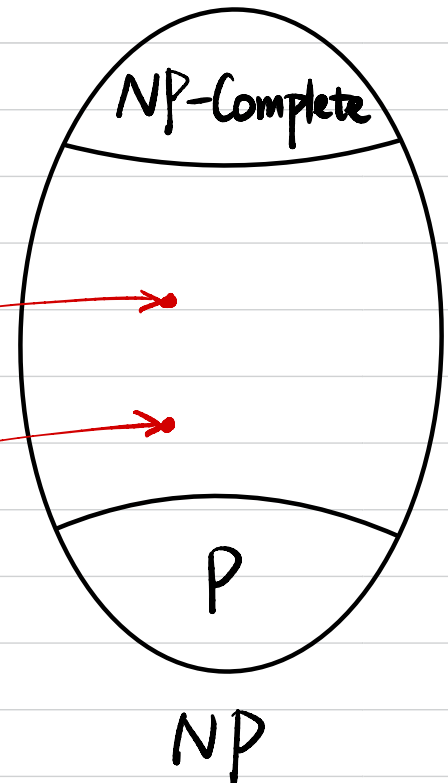


NP-Complete Problems: "hardest" problems in NP.

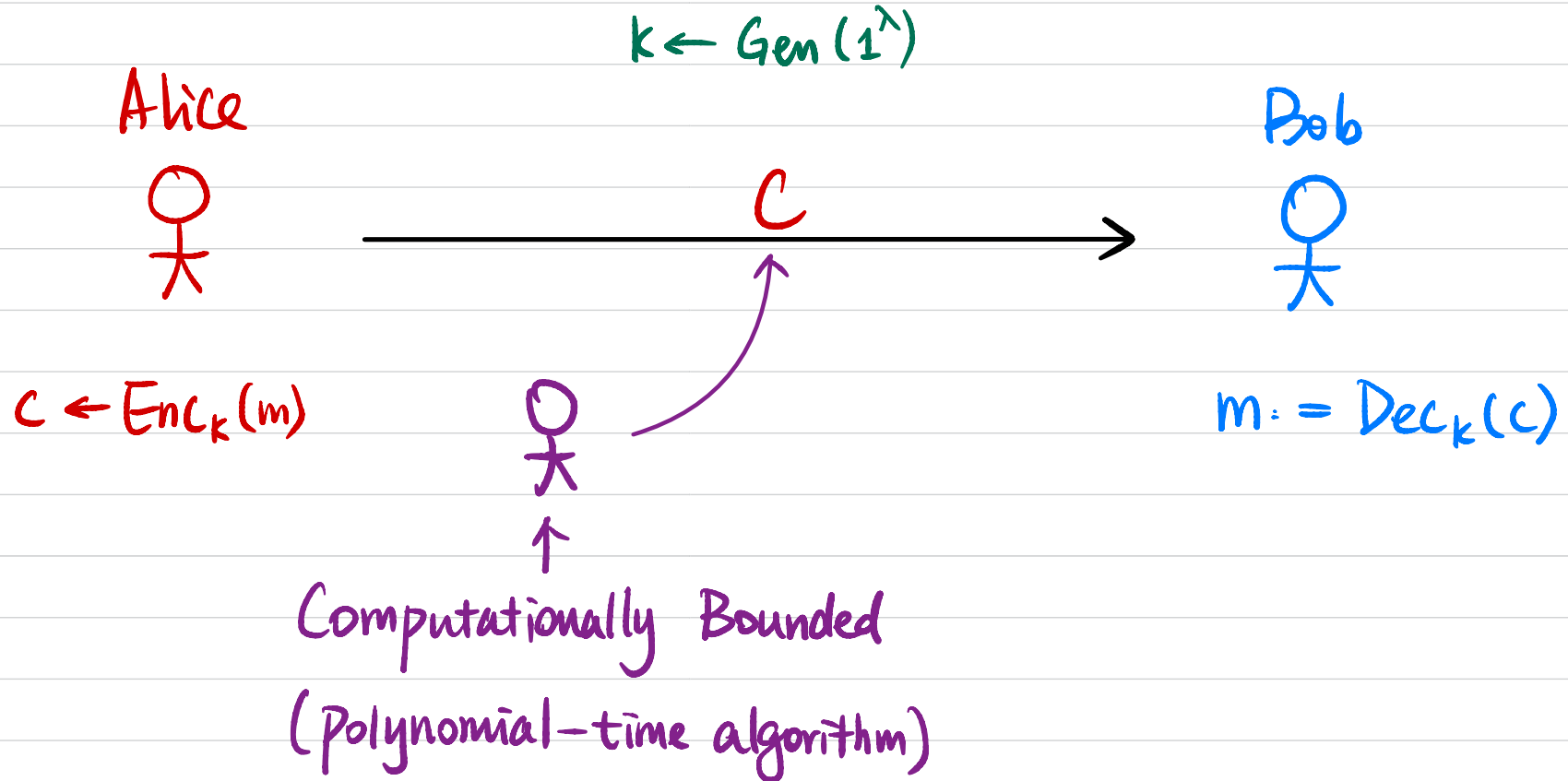
Is $P = NP$?

Factoring

DLOG



Computational Security



$\forall$ probabilistic poly-time (PPT) $\mathcal{A}$, $\text{Enc}_k(m_0) \stackrel{c}{\approx} \text{Enc}_k(m_1)$
"Computationally indistinguishable"

Computational Security

Alice



$$c_0 \leftarrow \text{Enc}_k(m_0)$$

$$c_1 \leftarrow \text{Enc}_k(m_1)$$

$$c_2 \leftarrow \text{Enc}_k(m_2)$$

⋮

m_0, m_1

$$b \leftarrow \{0, 1\}$$

$$c \leftarrow \text{Enc}_k(m_b)$$

$$k \leftarrow \text{Gen}(1^\lambda)$$

Bob



$$m_0 := \text{Dec}_k(c_0)$$

$$m_1 := \text{Dec}_k(c_1)$$

$$m_2 := \text{Dec}_k(c_2)$$

⋮

$c_0, c_1, c_2, \dots$



c



$b = ?$

Computational Security

Chosen-Plaintext Attack (CPA) Security

$k \leftarrow \text{Gen}(1^\lambda)$

Alice



$$c_0 \leftarrow \text{Enc}_k(m_0)$$

$$c_1 \leftarrow \text{Enc}_k(m_1)$$

$$c_2 \leftarrow \text{Enc}_k(m_2)$$

$\vdots$

$$m_0, m_1 \in \{0, 1\}^n$$

$$b \leftarrow \{0, 1\}$$

$$c \leftarrow \text{Enc}_k(m_b)$$

③ choose



(PPT) $\xrightarrow{\text{⑤ Output}} b'$

② see

$c_0, c_1, c_2, \dots$

c

④ see

Bob



$$m_0 := \text{Dec}_k(c_0)$$

$$m_1 := \text{Dec}_k(c_1)$$

$$m_2 := \text{Dec}_k(c_2)$$

$\vdots$

2^{-128}

$$\Pr[b = b'] \leq \frac{1}{2} + \text{negligible}$$

Security Parameter

$$k \leftarrow \text{Gen}(1^\lambda)$$

λ : security parameter

① adversary runs in time $\text{poly}(\lambda)$

② distinguishing advantage negligible(λ) $\leftarrow 2^{-\lambda}$

$$\uparrow$$
$$\text{negligible}(\lambda) \ll \frac{1}{\lambda^c} \quad \forall \text{ constant } c$$

Set parameters in practice:

Computational security parameter $\lambda = 128$

Best algorithm to break the scheme (e.g. find secret key) takes time $\sim 2^\lambda$.

How long does 2^{128} CPU cycles take?

Apple M5 chip: $\sim 4.6 \text{ GHz}$
($4.6 \times 10^9 \text{ CPU cycles/s}$)

Construction for SKE

From pseudorandom function/permutation (PRF/PRP)

Practical construction for PRF/PRP: block cipher

Standardized implementation: AES

Computational Assumption: "The construction is secure" (heuristics)

Best attack is brute-force search (classical/quantum).

Constructions for PKE

RSA Encryption: Factoring / RSA Assumption

El Gamal Encryption: Discrete Logarithm / Diffie-Hellman Assumption

Lattice-Based Encryption Schemes (Post-Quantum Security)

Thm (Informal): It's impossible to construct PKE from SKE in a black-box way.

Basic Number Theory

- $a \mid b$: a divides b ($b = a \cdot c$)
- Primes: an integer $p > 1$ that only has 2 divisors: 1 & p .
- Modular Arithmetic:

$a \bmod N$: remainder of a when divided by N

$$a \cdot b \bmod N = (a \bmod N) \cdot (b \bmod N) \bmod N.$$

$a \equiv b \pmod{N}$: a and b are congruent modulo N

How to compute $a^b \bmod N$ for a, b, N of n bits? Time complexity?

$n = 1000$

Ex: $5^{10} \bmod 7$

$219426^{736459} \bmod 392643$

Basic Number Theory

- $\gcd(a, b)$: greatest common divisor

How to compute $\gcd(a, b)$? Time complexity?

a, b both of n bits

Euclidean Algorithm

$$\gcd(17, 12) = 1$$

$$17 \bmod 12 = 5$$

$$12 \bmod 5 = 2$$

$$5 \bmod 2 = 1$$

$$2 \bmod 1 = 0$$

$$\gcd(18, 12) = 6$$

$$18 \bmod 12 = 6$$

$$12 \bmod 6 = 0$$

- $\gcd(a, N) = 1$: a & N are coprime

$\Rightarrow \exists b$ st. $a \cdot b \equiv 1 \pmod{N}$: a is invertible modulo N ,

b is its inverse, denoted as a^{-1} .

How to compute b ?

Extended Euclidean Alg.

a N

$$\gcd(17, 12) = 1$$

$$17 \bmod 12 = 5$$

$$12 \bmod 5 = 2$$

$$5 \bmod 2 = 1$$

$$2 \bmod 1 = 0$$

$$5 = 17 - 12 \times 1$$

$$2 = 12 - 5 \times 2$$

$$1 = 5 - 2 \times 2$$

$$\gcd(a, N) = 1$$

$\Downarrow$

$$1 = a \cdot x + N \cdot y$$

$\Downarrow \pmod{N}$

$$1 \equiv a \cdot x$$

Basic Number Theory

$$\mathbb{Z}_N^* := \{a \mid a \in [1, N-1], \gcd(a, N) = 1\}$$

$$\text{Euler's phi (totient) function } \phi(N) := |\mathbb{Z}_N^*|$$

Ex: N is prime. $\phi(N) = N-1$.

$$N = p \cdot q \text{ (} p, q \text{ are primes)} \quad \phi(N) = (p-1) \cdot (q-1).$$

Euler's Theorem $\forall a, N$ where $\gcd(a, N) = 1$, $a^{\phi(N)} \equiv 1 \pmod{N}$.

Corollary If $d \equiv e^{-1} \pmod{\phi(N)}$, then $(a^d)^e \equiv a \pmod{N}$.

$\Downarrow$

$$d \cdot e \equiv 1 \pmod{\phi(N)}$$

$\Downarrow$

$$d \cdot e = \phi(N) \cdot c + 1$$

$\Downarrow$

$$a^{de} \equiv a^{\phi(N) \cdot c + 1} \pmod{N}$$

$$\equiv 1^c \cdot a \pmod{N}$$

$$\equiv a \pmod{N}$$

RSA Assumption

- Factoring Assumption:

Generate two n -bit primes p, q ($p \neq q$)

Compute $N = p \cdot q$

Given N , it's computationally hard to find p & q (classically)

- RSA Assumption:

Generate two n -bit primes p, q ($p \neq q$)

Compute $N = p \cdot q$, $\phi(N) = (p-1)(q-1)$

Choose e s.t. $\gcd(e, \phi(N)) = 1$

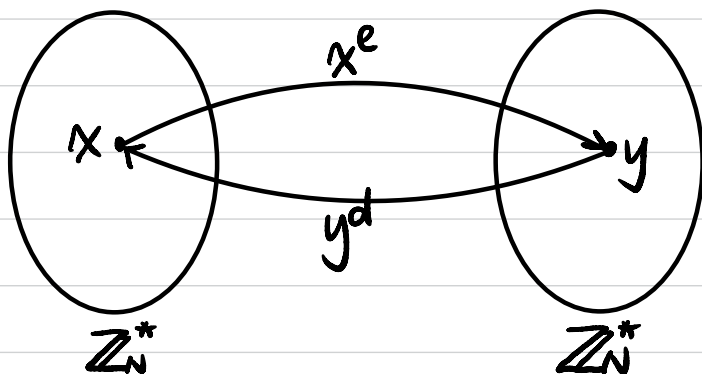
Compute $d = e^{-1} \bmod \phi(N)$

Given N & a random $y \leftarrow \mathbb{Z}_N^*$, it's computationally hard to find x s.t.
 $x^e \equiv y \pmod{N}$

If breaking Factoring



then break RSA



"Plain" RSA Encryption

$\lambda = 128$

$n = 1024$, key length 2048

• Gen(1^λ):

$n = O(\lambda)$

Generate two n -bit p, q ($p \neq q$)

Compute $N = p \cdot q$, $\phi(N) = (p-1)(q-1)$

Choose e s.t. $\gcd(e, \phi(N)) = 1$

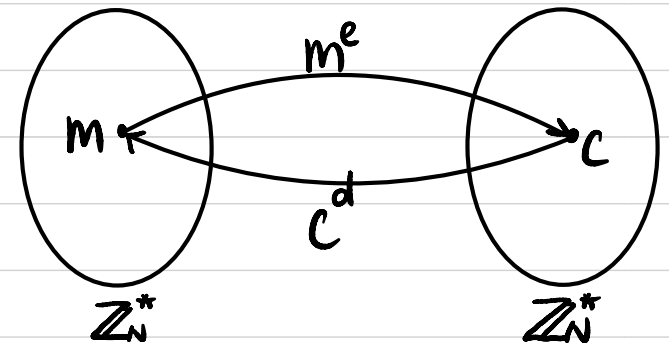
Compute $d = e^{-1} \bmod \phi(N)$.

$pk = (N, e)$

$sk = d$.

• $Enc_{pk}(m)$: $c = m^e \bmod N$

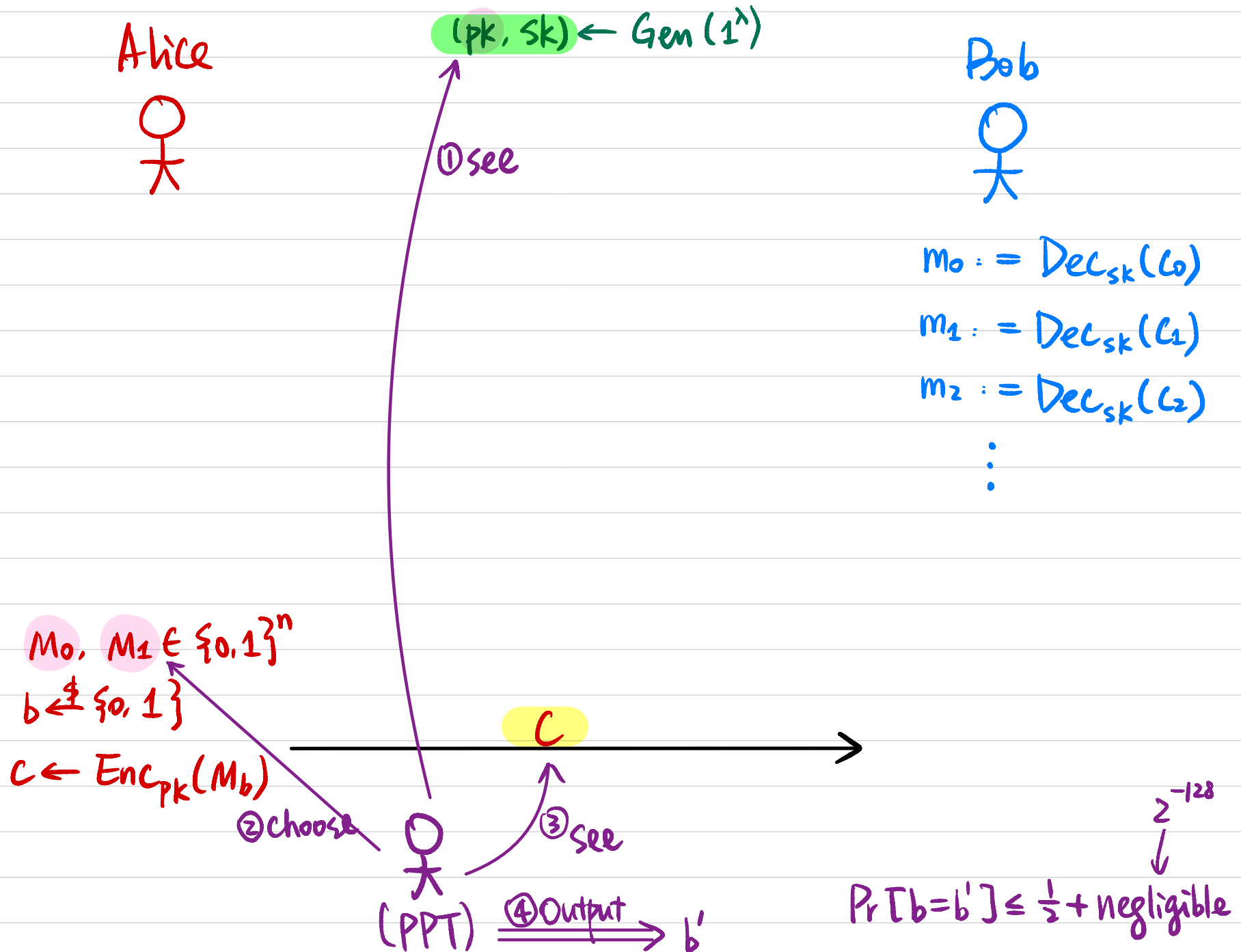
• $Dec_{sk}(c)$: $m = c^d \bmod N$



Any security issues? CPA secure?

Computational Security

Chosen-Plaintext Attack (CPA) Security



Q & A

CSCI 1040 (The Basics of Cryptographic Systems) "Crypto for poets"

MATH 1580 (Cryptography) Why is it correct?

CSCI 1510 (Introduction to Cryptography and Computer Security) Why is it secure?

CSCI 1515 (Applied Cryptography) How to use it?

