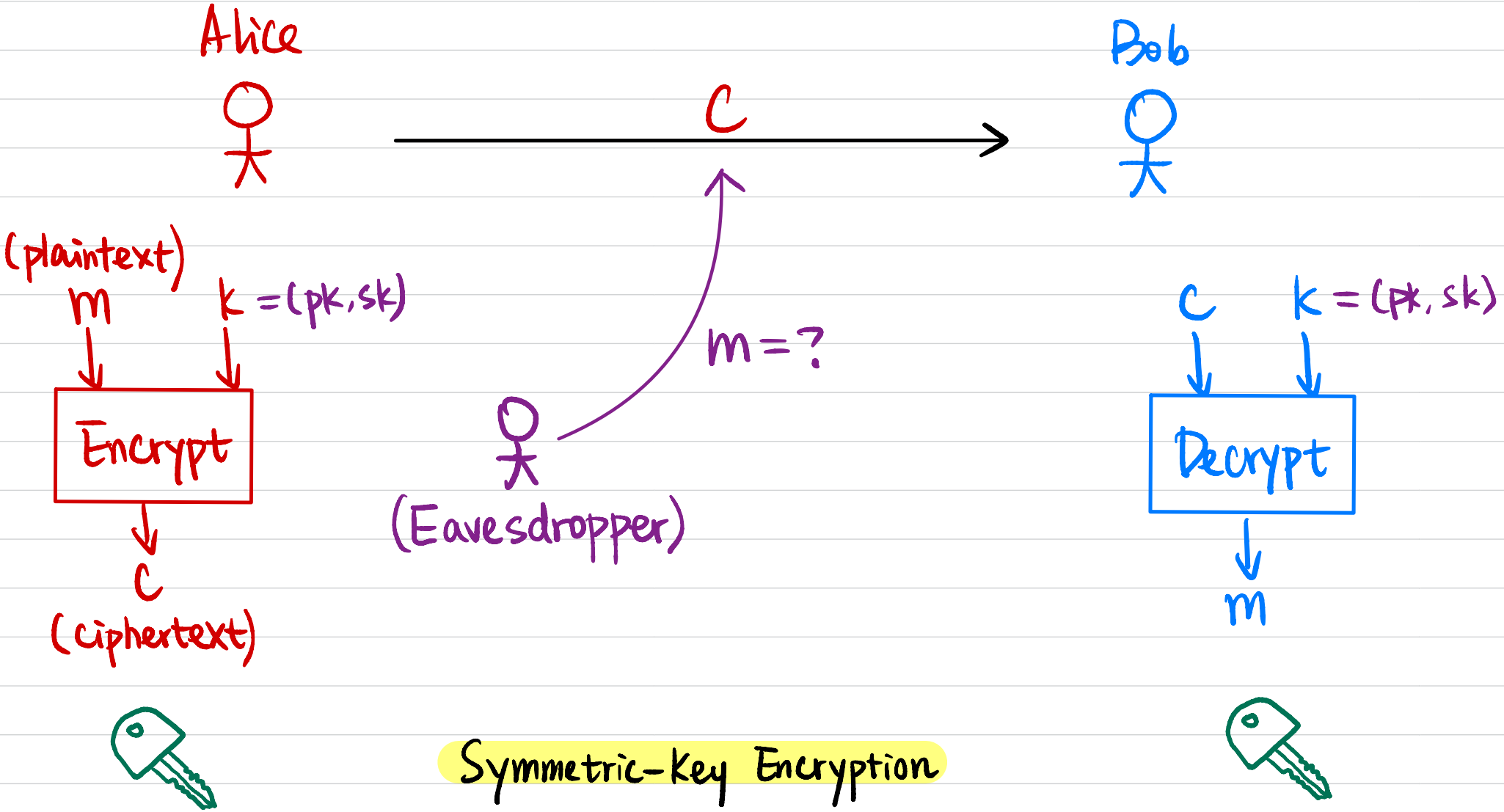
CSCI 1515 Applied Cryptography

This Lecture:

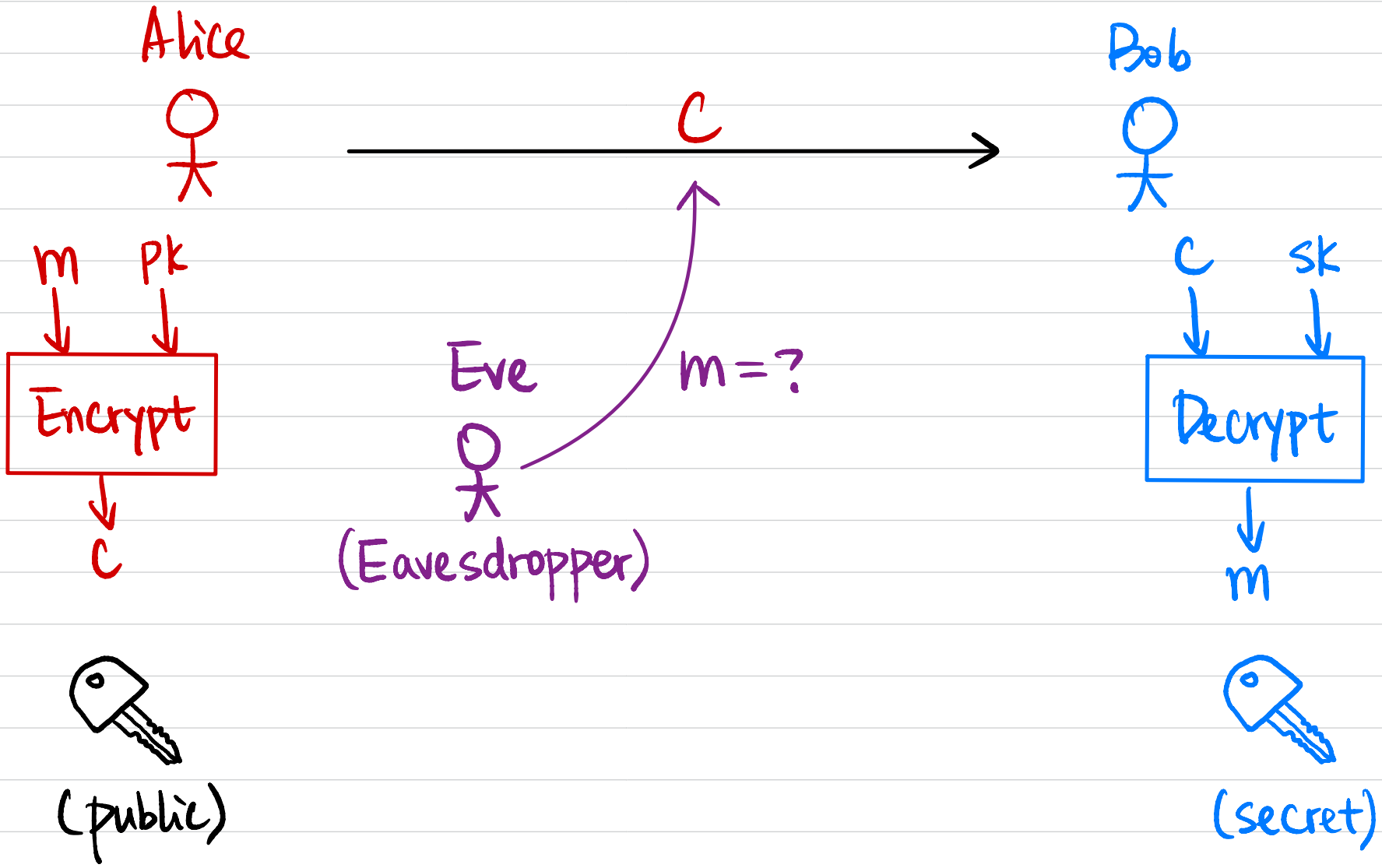
- Encryption Scheme Basics
- Computational Assumptions
- RSA Encryption

Message Secrecy



Symmetric-Key Encryption

Public-Key Encryption



Syntax

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

$k \leftarrow Gen$

$c \leftarrow Enc(k, m)$ $\quad Enc_k(m)$

$m := Dec(k, c)$ $\quad Dec_k(c)$

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

$(pk, sk) \leftarrow Gen$

$c \leftarrow Enc(pk, m)$ $\quad Enc_{pk}(m)$

$m := Dec(sk, c)$ $\quad Dec_{sk}(c)$

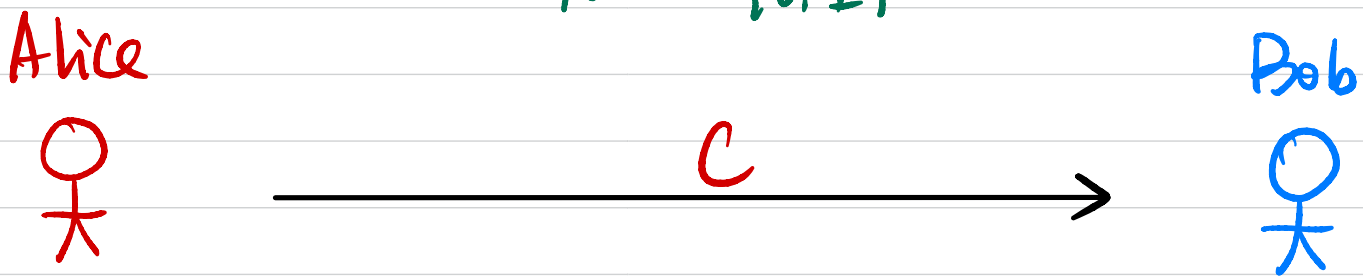
Why ever using SKE?

① Efficiency!

② Stronger Computational Assumptions

One-Time Pad (OTP)

$k \xleftarrow{\$} \{0,1\}^n$



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?

$c = k \oplus m$

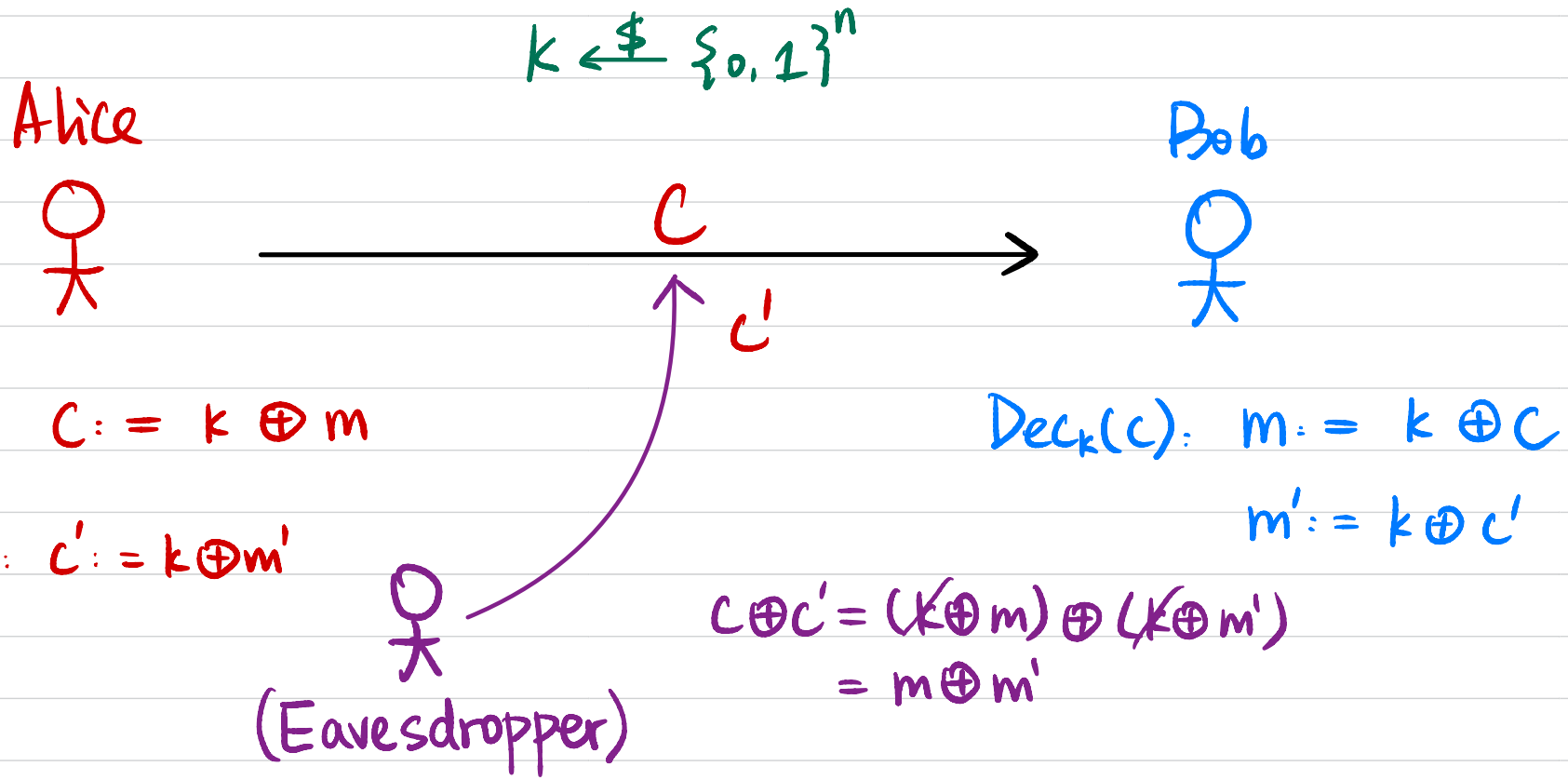
$k \oplus c = k \oplus (k \oplus m) = (k \oplus k) \oplus m$

$= 0 \cdots 0 \oplus m$

$= m$

Security?

One-Time Pad (OTP)

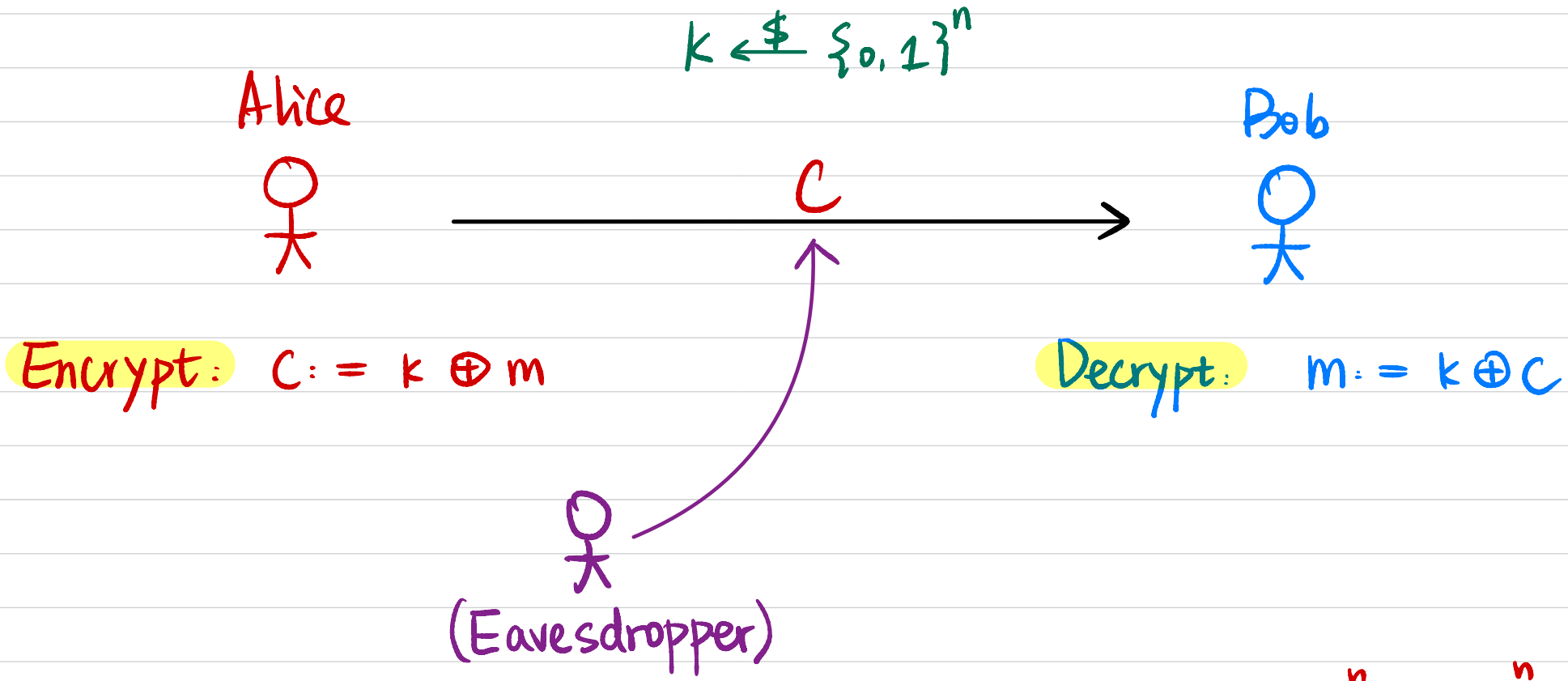


Distribution of c ? $\forall m \in \{0,1\}^n$, $Enc_k(m) \sim$ uniform over $\{0,1\}^n$

$\forall m_0, m_1 \in \{0,1\}^n$, $Enc_k(m_0) \equiv Enc_k(m_1)$

Can we re-use k ?

Shannon's Theorem



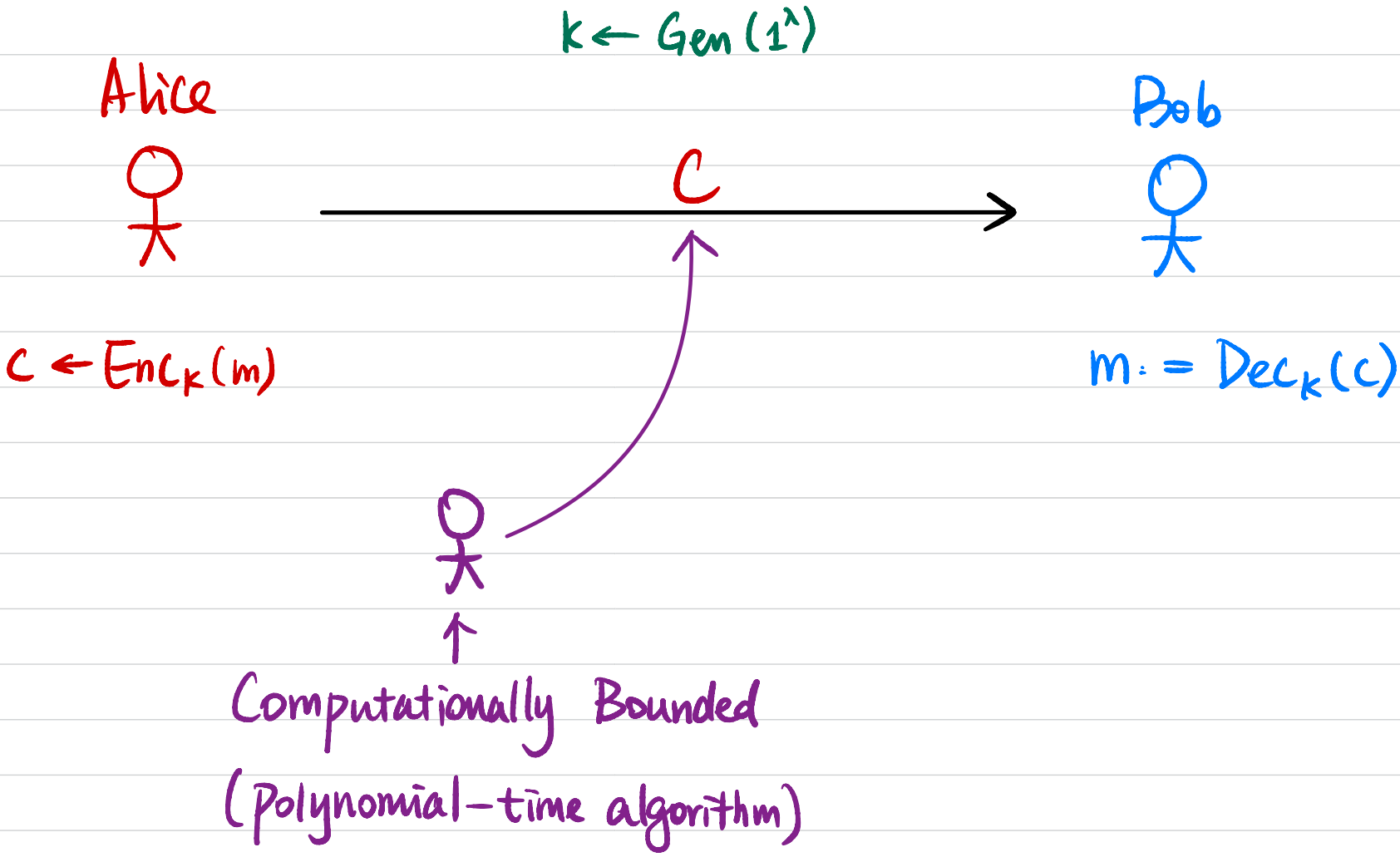
(Informal) For perfect (information-theoretic) security, $|K| \geq |M|$

($|K| \geq 2^n$, $|M| \geq 2^n$)

K: key space

M: message space

Computational Security



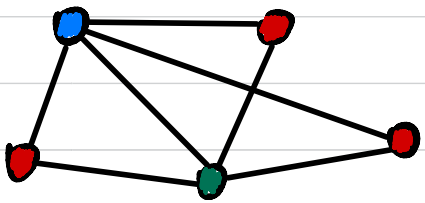
Computational Assumptions

polynomial-time algorithm : $A(x)$

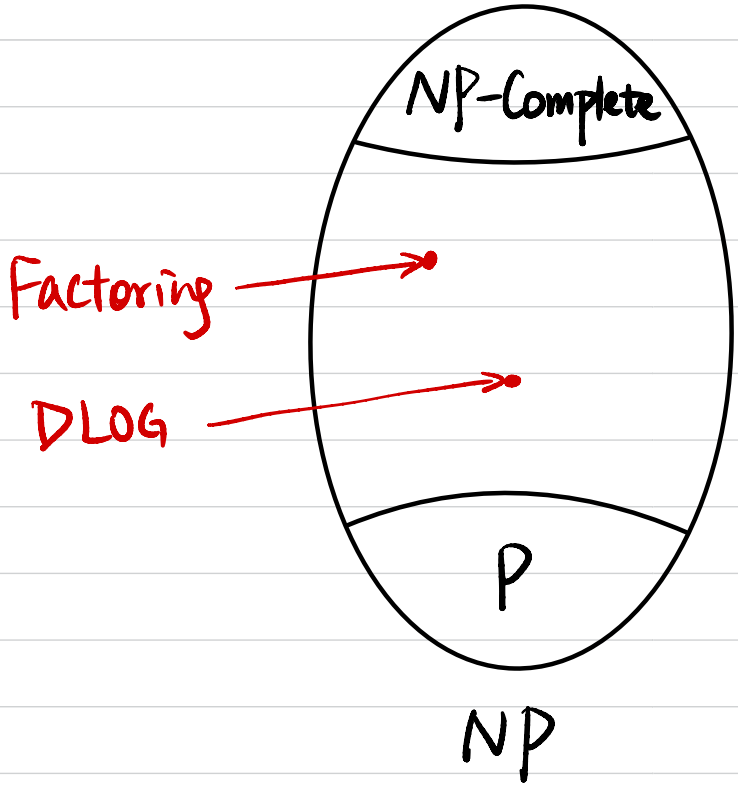
Input x of length n, A's running time $O(n^c)$ for a constant c.

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

Ex: Gragh 3-Coloring



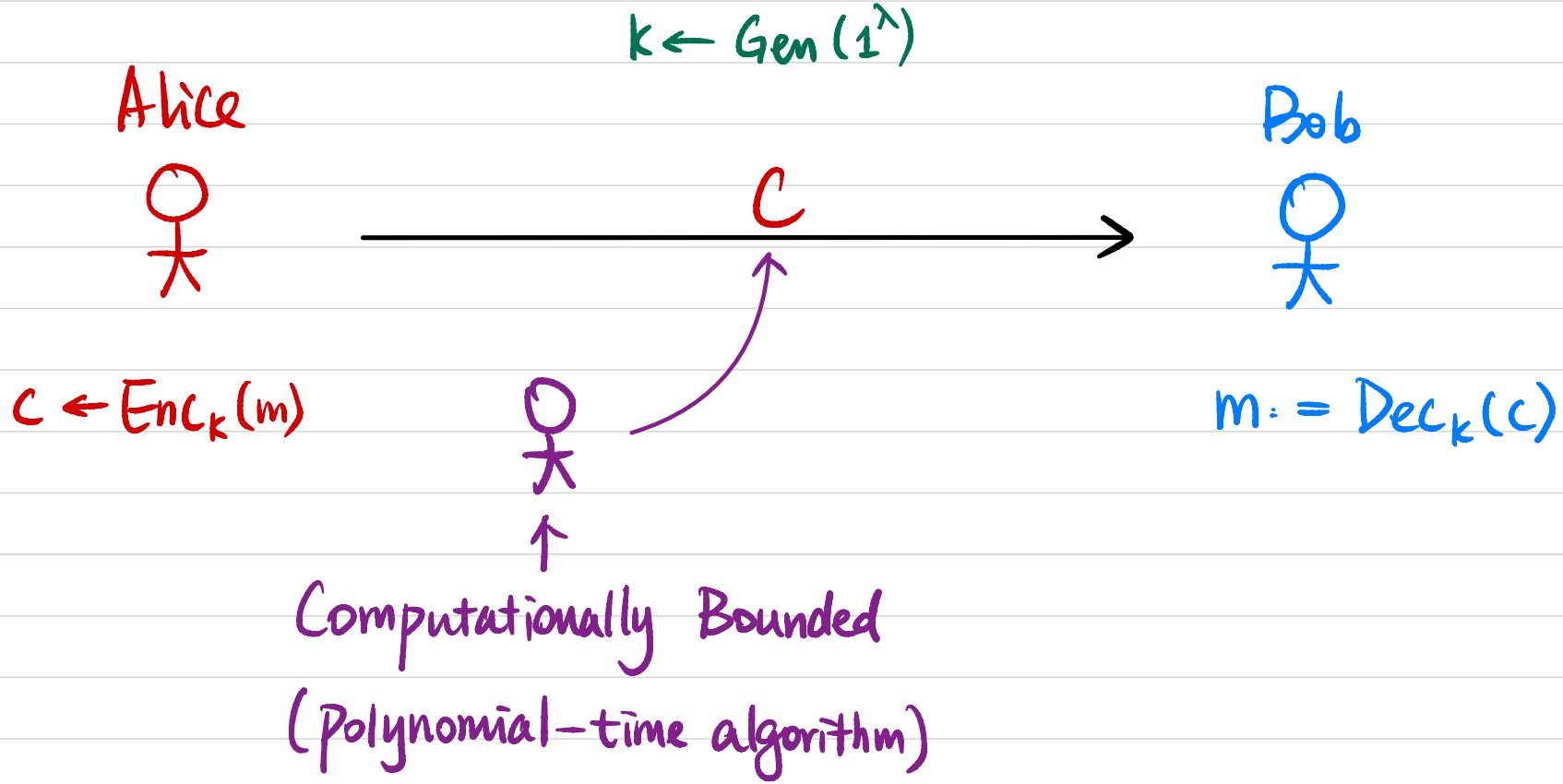
NP-Complete Problems: "hardest" problems in NP.



Is $P = NP$?

Assume $P \neq NP$

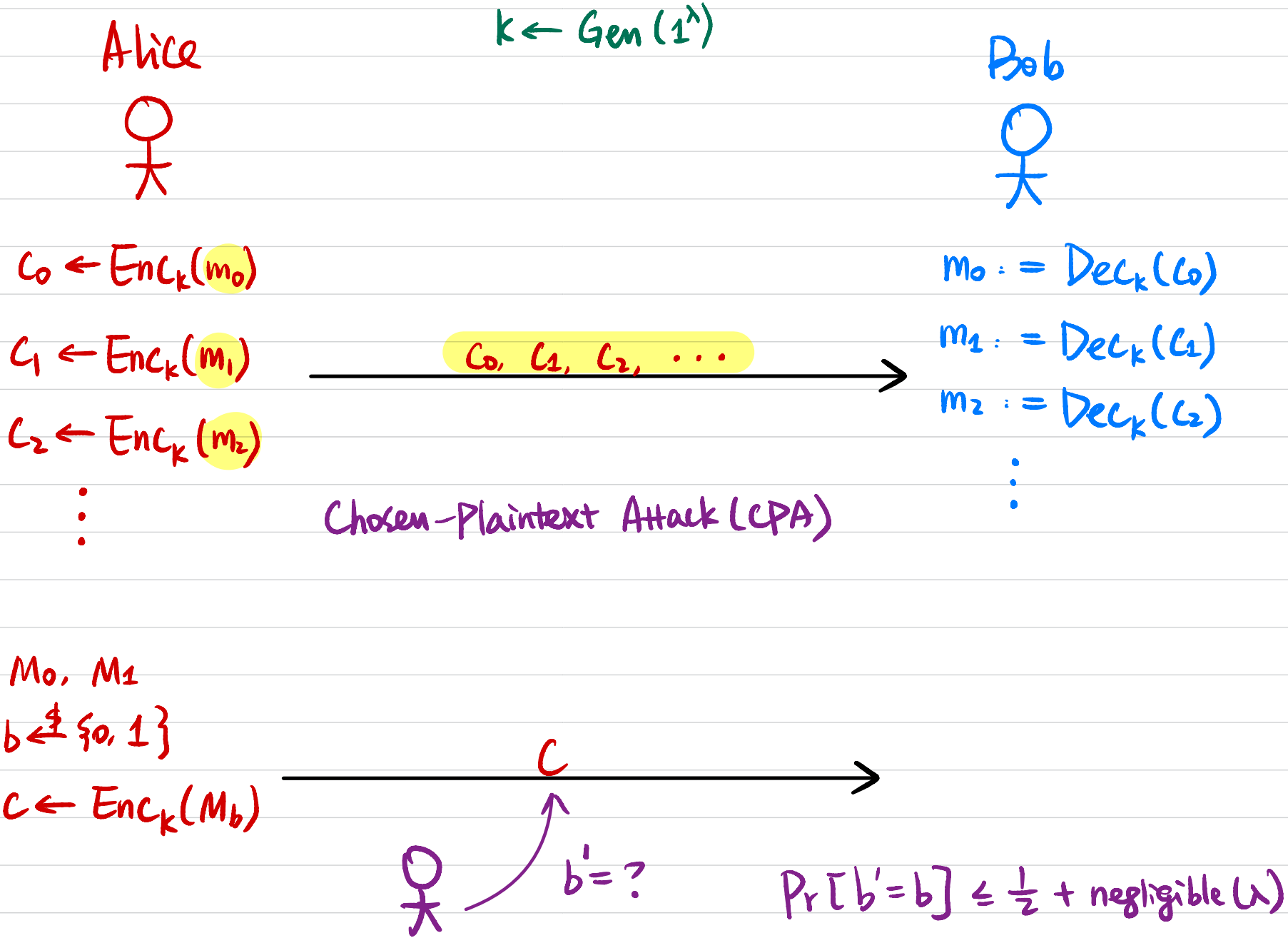
Computational Security



$\forall$ probabilistic poly-time (PPT) $\mathcal{A}$. $Enc_k(m_0) \overset{c}{\approx} Enc_k(m_1)$

"Computationally indistinguishable"

Computational Security



Security Parameter

$k \leftarrow Gen(1^\lambda)$ $\quad$ $Gen(\underbrace{11\cdots1}_{\lambda})$

λ : security parameter

① adversary runs in time poly(λ)

② distinguishing advantage negligible(λ)

$\uparrow$

negligible(λ) $\ll \frac{1}{\lambda^c}$ $\forall$ constant c

Set parameters in practice:

Computational security parameter $\lambda = 128$

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

Ex: Best algorithm is brute-force search $\Rightarrow$ key length = λ $\quad$ $\overbrace{1001\cdots1}^{\lambda}$

Best algorithm for a key length ℓ takes time $\sim \sqrt{2^\ell}$ $\Rightarrow$ key length = $2\cdot\lambda$

$\underbrace{\qquad\qquad}_{\ell}$

$\sqrt{2^\ell} \approx 2^\lambda \Rightarrow \ell = 2\cdot\lambda$

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"

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

Constructions for PKE

RSA Encryption: Factoring / RSA Assumption

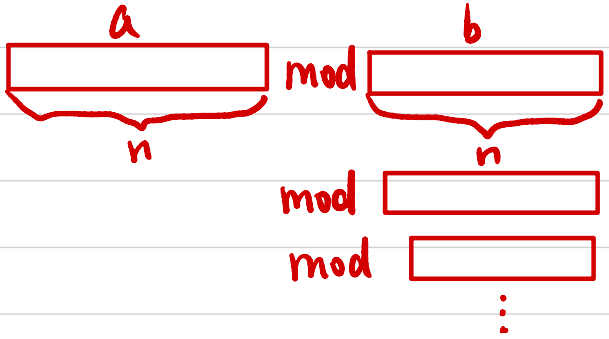
ElGamal Encryption: Discrete Logarithm / Diffie-Hellman Assumption

Lattice-Based Encryption Schemes (Post-Quantum Security)

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

"black-box separation"

Number Theory



- $a \mid b$: a divides b ($b = a \cdot c$)

  gcd(a, b): greastest common divisor

  gcd(a, b) = 1: a & b are coprime

a, b both O(n) bits

How to compute gcd? Time complexity? O(n)

Euclidean Alg. Ex: gcd(17, 12) = 1

17 mod 12 = 5

12 mod 5 = 2

5 mod 2 = 1

2 mod 1 = 0

gcd(18, 12) = 6

18 mod 12 = 6

12 mod 6 = 0

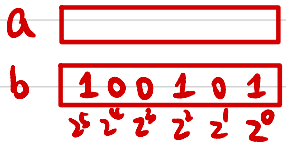
- Modular Arithmetic:

  a mod N: remainder of a when divided by N

  $a \equiv b \pmod N$: a and b are **Congruent modulo** N

How to compute a^b mod N? Time complexity? O(n)

a, b, N all O(n) bits



$a^b \equiv a \cdot a^4 \cdot a^{32} \pmod N$

a mod N

$a^2 \bmod N = (a \bmod N) \cdot (a \bmod N) \bmod N$

$a^4 \bmod N = (a^2 \bmod N) \cdot (a^2 \bmod N) \bmod N$

$a^8 \bmod N$

⋮

N

Number Theory

- If $\gcd(a,N) = 1$, then $\exists\, b$ s.t.

  $a \cdot b \equiv 1 \pmod N$: a is invertible modulo N,

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

$\gcd(a,N) = 1$

$\Downarrow$

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

$\Downarrow \bmod N$

$1 \equiv a \cdot x$

How to Compute b?

Extended Euclidean Alg.

$\gcd(17, 12) = 1$

$17 \bmod 12 = 5$

$12 \bmod 5 = 2$

$5 \bmod 2 = 1$

$2 \bmod 1 = 0$

$5 = 17 - 12 \times 1$

$2 = 12 - 5 \times 2 = 17 \cdot x + 12 \cdot y$

$1 = 5 - 2 \times 2 = 17 \cdot x' + 12 \cdot y'$

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

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

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

RSA Assumption

- Factoring Assumption:

  Generate two n-bit primes p, q (How?)

  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

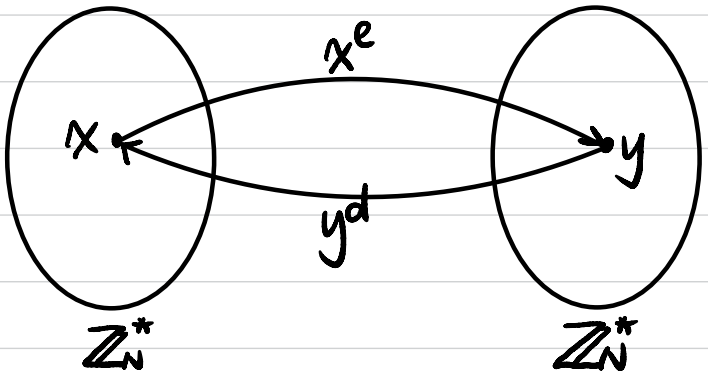
  Compute $N = p \cdot q$, $\phi(N) = (p-1)(q-1)$

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

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

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

$$x^e \equiv y \pmod N$$



$$(x^e)^d = x^{ed} \equiv x \bmod N$$

$$e \cdot d \equiv 1 \bmod \phi(N)$$

$$x^{\phi(N)} \equiv 1 \bmod N$$

RSA Encryption

- $Gen(1^\lambda)$: $n = O(\lambda)$ $n = 1024$, key length 2048

  Generate two n-bit primes p, q ← How?

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

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

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

  $pk = (N, e)$ $\quad sk = d$.

- $Enc_{pk}(m)$: $c = m^e \bmod N$ ← How (efficiently)?

- $Enc_{sk}(c)$: $m = c^d \bmod N$ ← How (efficiently)?

Any security issue?