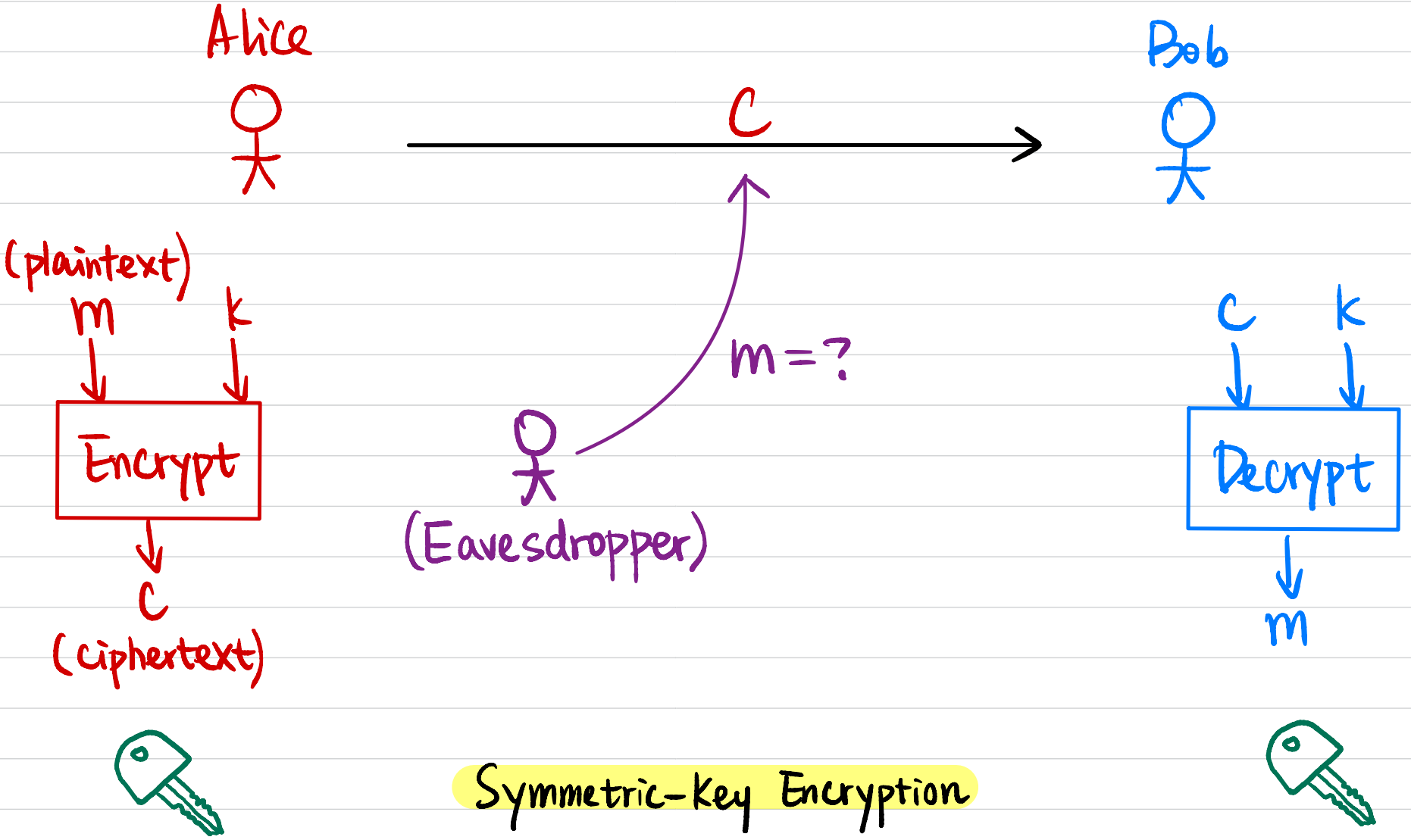
CSCI 1515 Applied Cryptography

This Lecture:

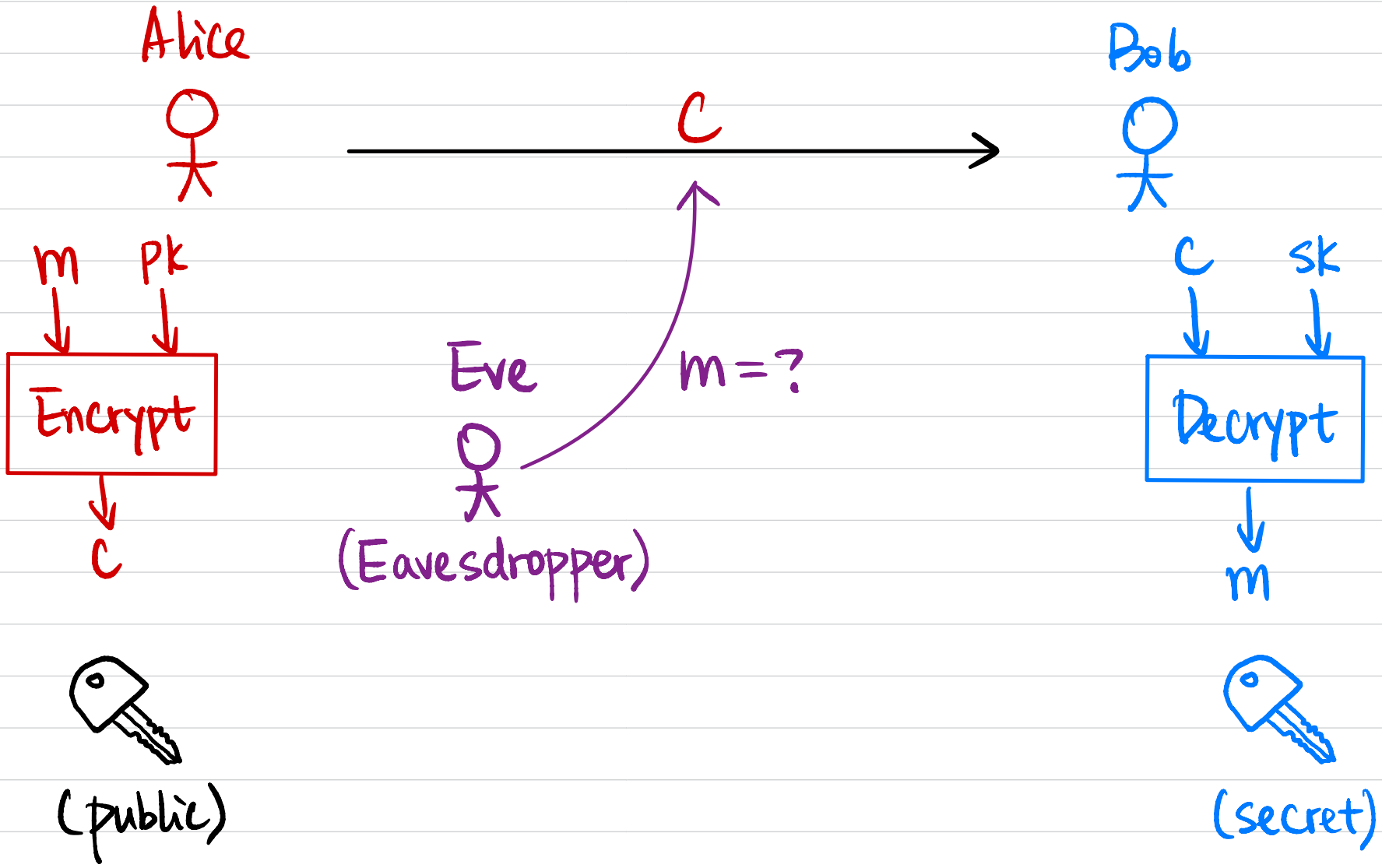
- Encryption Scheme Basics
- Computational Assumptions
- RSA Encryption
- ElGamal Encryption
- Diffie-Hellman Key Exchange

Message Secrecy



Symmetric-Key Encryption

Public-Key Encryption



Syntax

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

$k \leftarrow Gen$

$c \leftarrow Enc(k, m)$

$m := Dec(k, c)$

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

$(pk, sk) \leftarrow Gen$

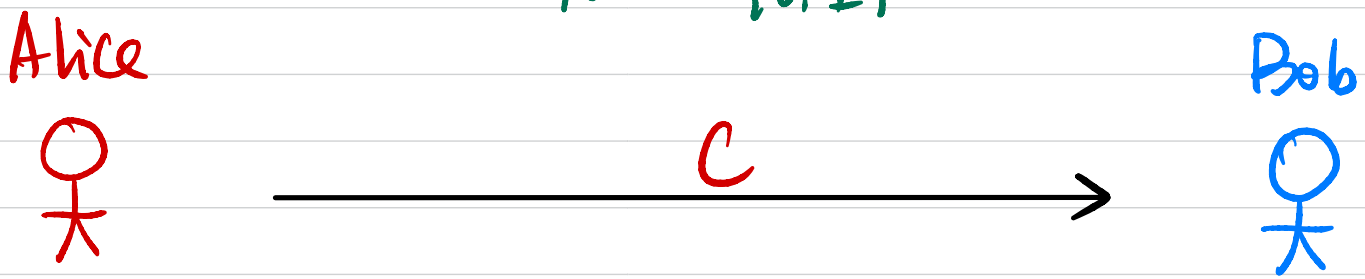
$c \leftarrow Enc(pk, m)$

$m := Dec(sk, c)$

Why ever using SKE?

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 $c = 1101100$

---

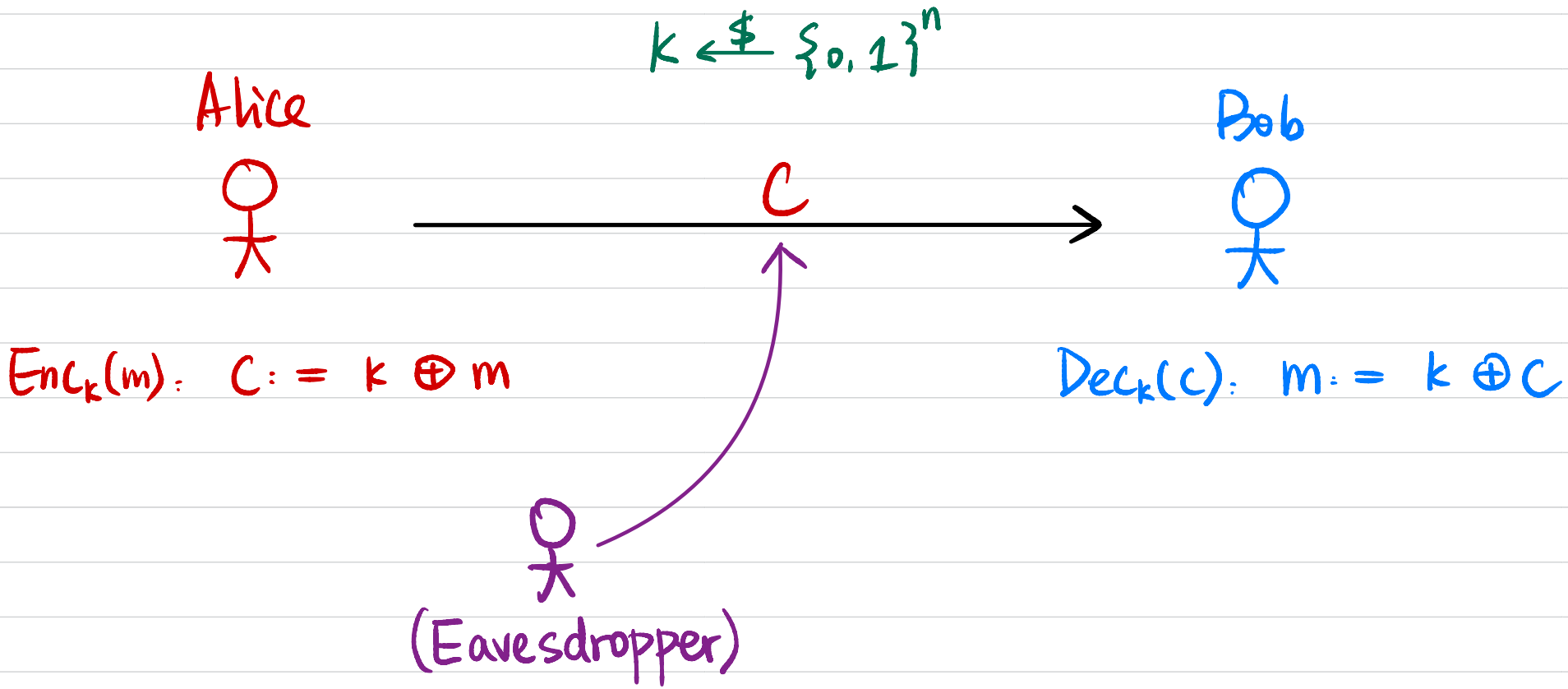
plaintext $m = 1001001$

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

Correctness?

Security?

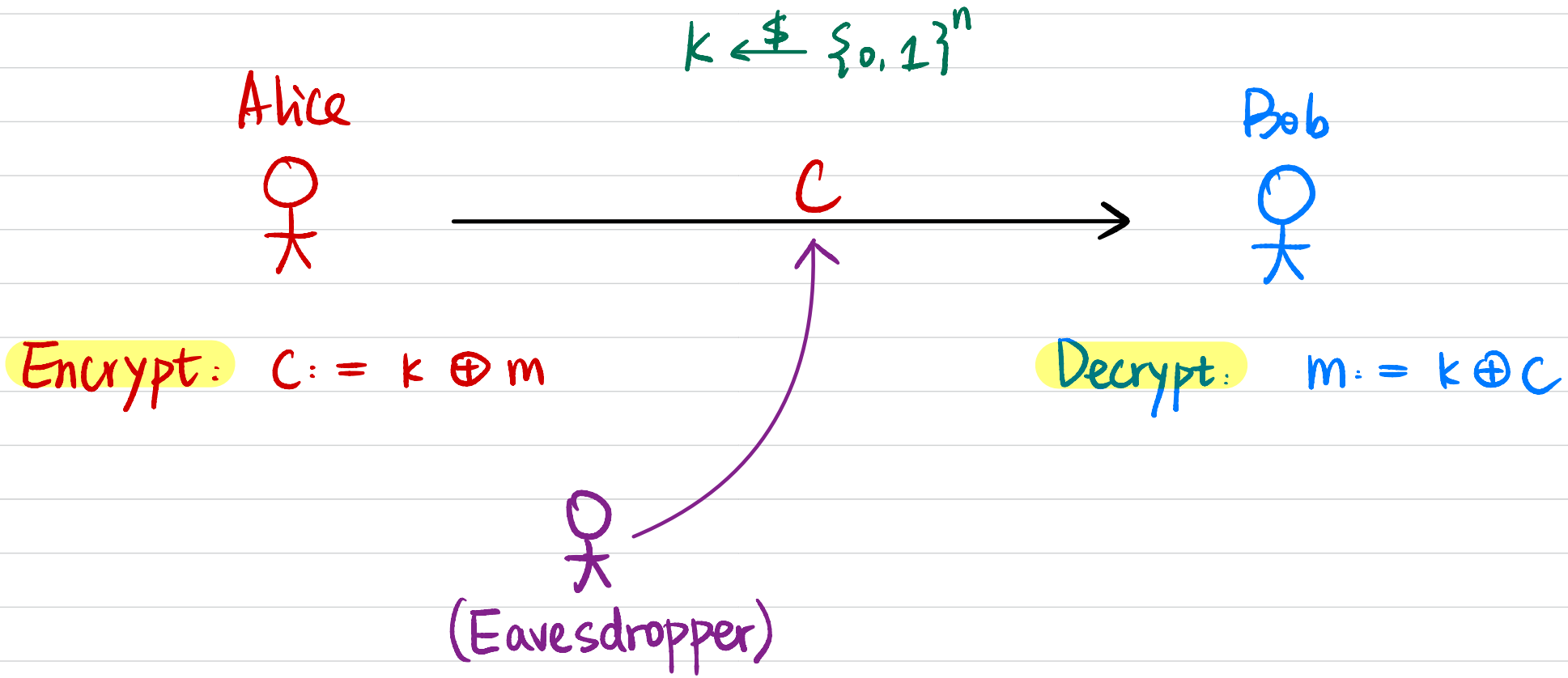
One-Time Pad (OTP)



Distribution of c ?

Can we re-use k ?

Shannon's Theorem

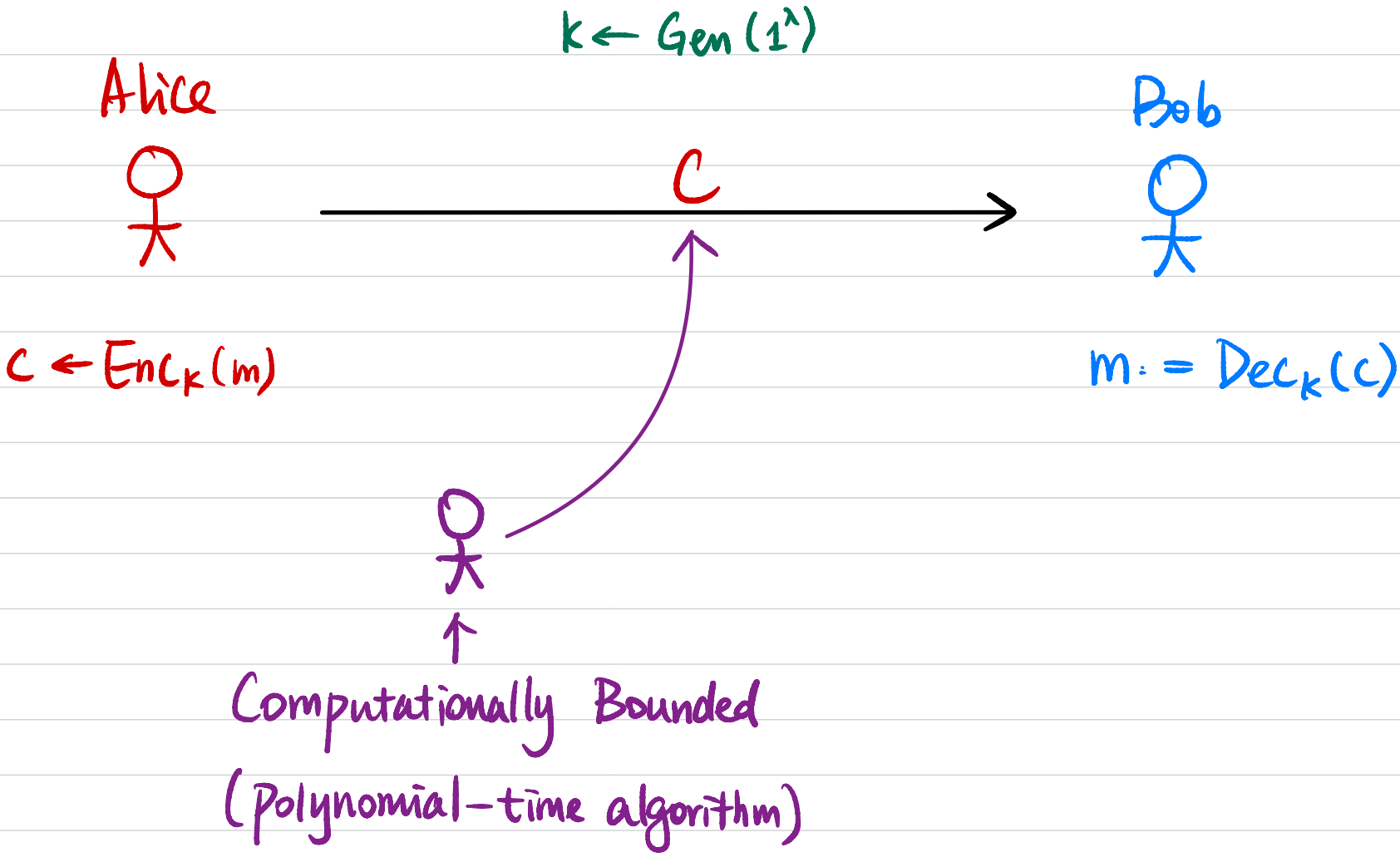


(Informal) For perfect (information-theoretic) security, $|K| \geq |M|$

K: key space

M: message space

Computational Security



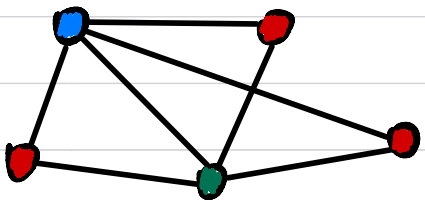
Computational Assumptions

polynomial-time algorithm : $A(x)$

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

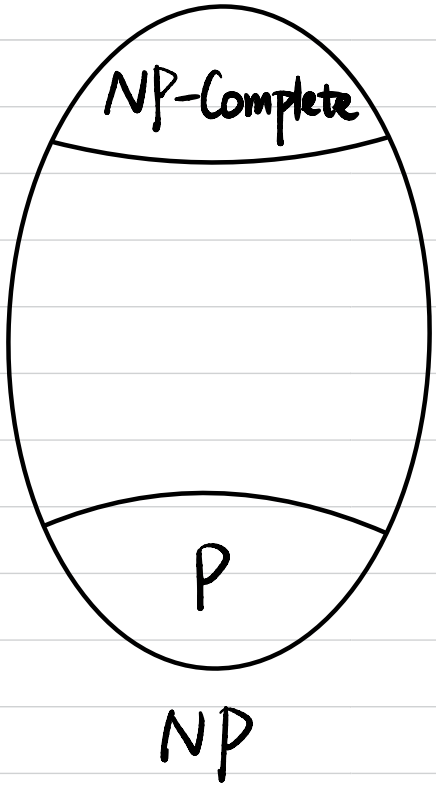
NP Problem : decision problems whose solution can be verified in poly time.

Ex: Gragh 3-Coloring

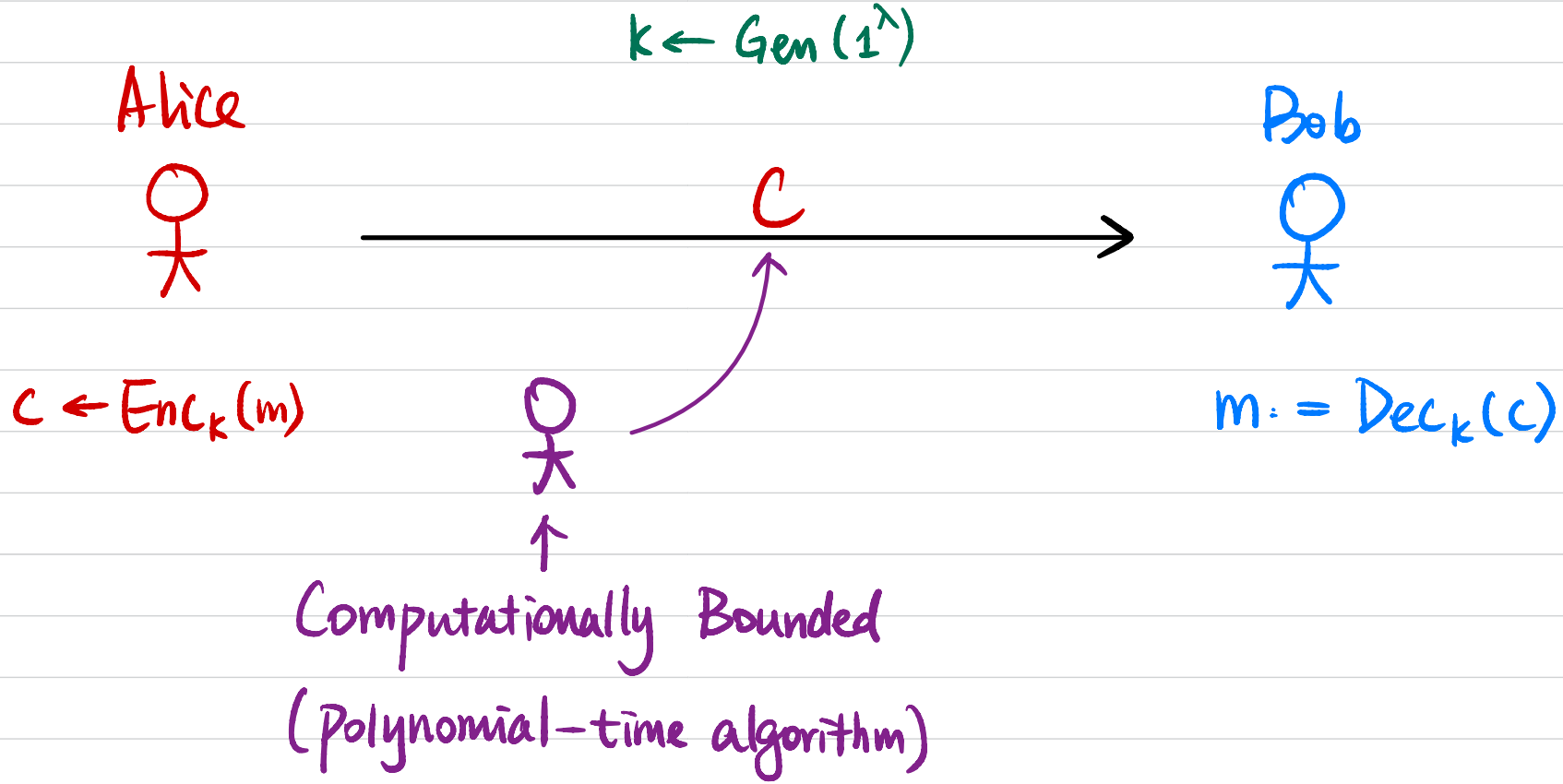


NP-Complete Problems: "hardest" problems in NP.

Is $P = NP$?



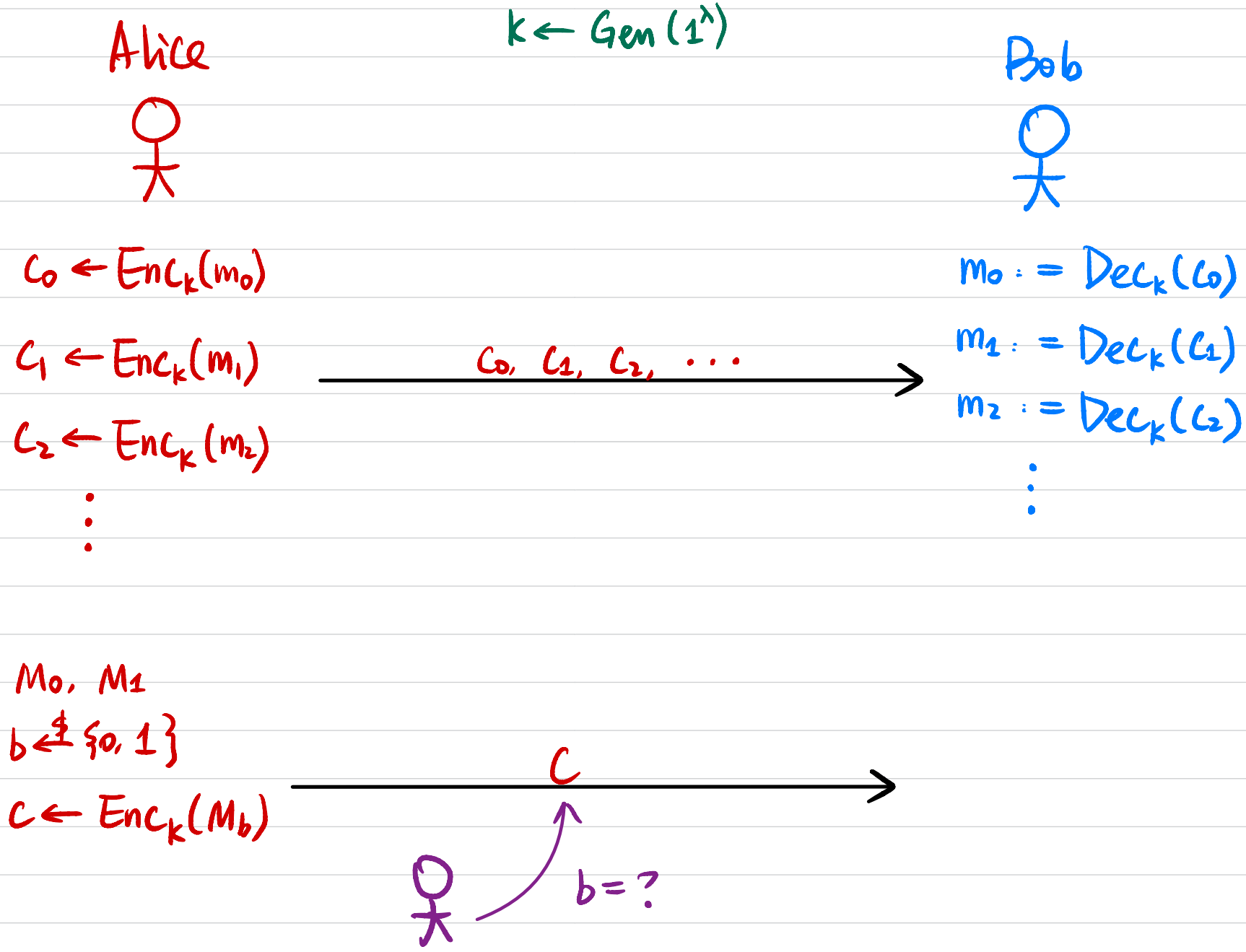
Computational Security



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

"Computationally indistinguishable"

Computational Security



Security Parameter

$k \leftarrow Gen(1^\lambda)$

λ : security parameter

① adversary runs in time $poly(\lambda)$

② distinguishing advantage $negligible(\lambda)$

$\uparrow$

$negligible(\lambda) \ll \frac{1}{\lambda^c}$ $\forall$ 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 = ?

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

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

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.

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

  How to compute gcd? Time complexity?

- 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 $a^b \bmod N$? Time complexity?

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}.

  How to compute b?

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

  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 prime p, q

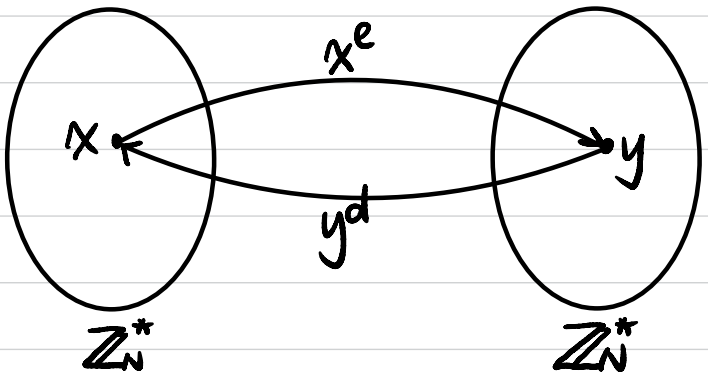
  Compute $N = p \cdot q$, $\Phi(N) = (p-1)(q-1)$

  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.

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



RSA Encryption

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

  Generate two n-bit prime p, q

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

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

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

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

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

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

Any security issue?

Group Theory

Def A group is a set G along with a binary operation $\circ$ with properties:

① Closure: $\forall g, h \in G$, $g \circ h \in G$

② Existence of an identity: $\exists e \in G$ s.t. $\forall g \in G$, $e \circ g = g \circ e = g$.

③ Existence of inverse: $\forall g \in G$, $\exists h \in G$ s.t. $g \circ h = h \circ g = e$

Inverse of g denoted as g^{-1}.

④ Associativity: $\forall g_1, g_2, g_3 \in G$, $(g_1 \circ g_2) \circ g_3 = g_1 \circ (g_2 \circ g_3)$

We say a group is abelian if it satisfies:

⑤ Commutativity: $\forall g, h \in G$, $g \circ h = h \circ g$

For a finite group, we use $|G|$ to denote its order (# of elements)

Group Theory

Ex. $(\mathbb{Z}, +)$ is an abelian group

$(\mathbb{Z}, \cdot)$ is not a group

$(\mathbb{Z}_N^*, \cdot)$ is an abelian group ($\cdot$ denotes multiplication mod N)

Def Let G be a group of order m.

Denote $\langle g \rangle = \{g^0, g^1, g^2, \cdots, g^{m-1}\}$

G is a cyclic group if $\exists g \in G$ s.t. $\langle g \rangle = G$.

g is a generator of G.

Ex. $\mathbb{Z}_p^*$ (for a prime p) is a cyclic group of order $p-1$.

How to find a generator?

Diffie-Hellman Assumptions

Integer group key 2048-bit
Elliptic Curve group key 256-bit

$(G, q, g) \leftarrow G(1^\lambda)$ — $O(\lambda)$-bit integer

cyclic group G of order q, with generator g.

- Discrete Logarithm (DLOG) Assumption:

  $x \xleftarrow{\$} \mathbb{Z}_q$, compute $h = g^x$

  Given (G, q, g, h), it's computationally hard to find x (classically).

- Computational Diffie-Hellman (CDH) Assumption:

  $x, y \xleftarrow{\$} \mathbb{Z}_q$, compute $h_1 = g^x$, $h_2 = g^y$

  Given (G, q, g, h_1, h_2), it's computationally hard to find g^{xy}.

- Decisional Diffie-Hellman (DDH) Assumption:

  $x, y, z \xleftarrow{\$} \mathbb{Z}_q$, compute $h_1 = g^x$, $h_2 = g^y$

  Given (G, q, g, h_1, h_2), it's computationally hard to distinguish between g^{xy} and g^z.

ElGamal Encryption

- $Gen(1^\lambda)$:

  $(G, q, g) \leftarrow G(1^\lambda)$

  $x \xleftarrow{\$} \mathbb{Z}_q$, compute $h = g^x$

  $pk = (G, q, g, h)$ $\quad sk = x$

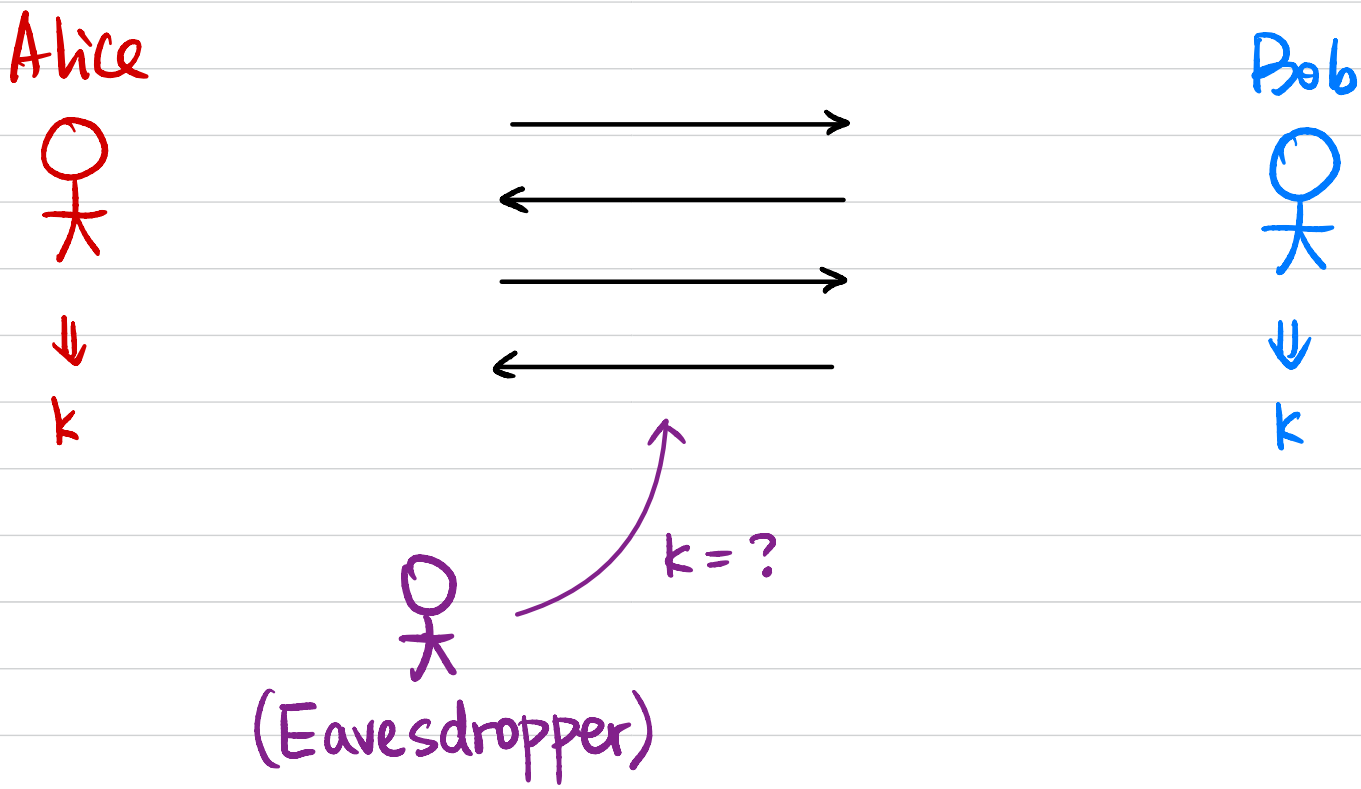
- $Enc_{pk}(m)$: $m \in G$

  $y \xleftarrow{\$} \mathbb{Z}_q$

  $c = \langle g^y, h^y \cdot m \rangle$

- $Enc_{sk}(c)$:

Secure Key Exchange



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

Key Exchange from PKE?

Diffie-Hellman Key Exchange

$(G, q, g) \leftarrow G(1^\lambda)$

$x \xleftarrow{\$} \mathbb{Z}_q$, compute $h_A = g^x$

(G, q, g, h_A) →

$y \xleftarrow{\$} \mathbb{Z}_q$, compute $h_B = g^y$

← h_B

$k = h_B^x$

$k = h_A^y$

$k = ?$

(Eavesdropper)