COS433/Math 473: Cryptography

Mark Zhandry

Princeton University

Spring 2020

Announcements

HW5 Due Today
PR2 Due April 19^{th}

Previously on COS 433…

Integer Factorization

Integer Factorization

Given an integer **N**, find it's prime factors

Studied for centuries, presumed difficult

- Grade school algorithm: $O(N^{1/2})$
- Better algorithms using birthday paradox: $O(N^{1/4})$
- Even better assuming G. Riemann Hyp.: $O(N^{1/5})$
- Still better heuristic algorithms:

$$\exp(\ C\ (\log N)^{1/3}\ (\log \log N)^{2/3}\ )$$

- However, all require super-polynomial time in bit-length of **N**

Factoring Assumption: For any factoring algorithm  running in polynomial time, $\exists$ negligible **ε** such that:

Pr[(p,q)←**(N):**

N=pq

p,q ←random **λ**-bit primes**] ≤ ε(λ)**

Chinese Remainder Theorem

Let $\mathbf{N = pq}$ for distinct prime $\mathbf{p,q}$

Let $\mathbf{x}\in\mathbb{Z}_\mathbf{p}$, $\mathbf{y}\in\mathbb{Z}_\mathbf{q}$

Then there exists a unique integer $\mathbf{z}\in\mathbb{Z}_\mathbf{N}$ such that

- $\mathbf{x = z \bmod p}$, and
- $\mathbf{y = z \bmod q}$

Proof: $\mathbf{z = [py(p^{-1} \bmod q)+qx(q^{-1} \bmod p)] \bmod N}$

Quadratic Residues

Definition: y is a quadratic residue mod N if there exists an x such that $y = x^2 \bmod N$. x is called a "square root" of y

Ex:

- Let p be a prime, and $y \neq 0$ a quadratic residue mod p. How many square roots of y?
- Let $N=pq$ be the product of two primes, y a quadratic residue mod N. Suppose $y \neq 0 \bmod p$ and $y \neq 0 \bmod q$. How many square roots?

QR Assumption: For any algorithm  running in polynomial time, $\exists$ negligible **ε** such that:

$\Pr[y^2 = x^2 \bmod N:$

$y \leftarrow$  (N, x^2)

$N = pq$, $p, q \leftarrow$ random **λ**-bit primes

$x \leftarrow \mathbb{Z}_N \quad] \leq \varepsilon(\lambda)$

This Time

Factoring continued

Public key cryptography

Theorem: If the factoring assumption holds, then the QR assumption holds

Proof

To factor **N**:

- $x \leftarrow \mathbb{Z}_N$
- $y \leftarrow$  (N, x^2)
- Output $GCD(x-y, N)$

Analysis:

- Let $\{a,b,c,d\}$ be the 4 square roots of x^2
-  has no idea which one you chose
- With probability ½, y will not be in $\{+x, -x\}$
- In this case, we know $x = y \bmod p$ but $x = -y \bmod q$

Collision Resistance from Factoring

Let $\mathbf{N=pq}$, $\mathbf{y}$ a QR mod $\mathbf{N}$
Suppose $\mathbf{-1}$ is not a **QR** mod $\mathbf{N}$

Hashing key: $\mathbf{(N,y)}$
Domain: $\{1,\ldots,(N-1)/2\}\times\{0,1\}$
Range: $\{1,\ldots,(N-1)/2\}$

H((N,y), (x,b)): Let $z = y^b x^2 \bmod N$

- If $z \in \{1,\ldots,(N-1)/2\}$, output z
- Else, output $-z \bmod N \in \{1,\ldots,(N-1)/2\}$

Theorem: If the factoring assumption holds, **H** is collision resistant

Proof:

- Collision means $(x_0,b_0)\neq(x_1,b_1)$ s.t.

$$y^{b0}\ x_0^2 = \pm\ y^{b1}\ x_1^2 \bmod N$$

- If $b_0=b_1$, then $x_0\neq x_1$, but $x_0^2=\pm x_1^2 \bmod N$
  - $x_0^2=-x_1^2$ mod N not possible. Why?
  - $x_0\neq -x_1$ since $x_0,x_1\in\{1,\dots,(N-1)/2\}$
- If $b_0\neq b_1$, then $(x_0/x_1)^2 = \pm y^{\pm1} \bmod N$
  - $-y$ case not possible. Why?
  - (x_0/x_1) or (x_1/x_0) is a square root of y

Choosing **N**

How to choose **N** so that **-1** is not a QR?

By CRT, need to choose **p,q** such that -1 is not a QR mod **p** or mod **q**

Fact: if **p = 3 mod 4**, then **-1** is not a QR mod **p**
Fact: if **p = 1 mod 4**, then **-1** is a QR mod **p**

Is Composite **N** Necessary for SQ to be hard?

Let p be a prime, and suppose $p = 3 \bmod 4$

Given a QR x mod p, how to compute square root?

Hint: recall Fermat: $x^{p-1}=1$ mod p for all $x\neq 0$

Hint: what is $x^{(p+1)/2}$ mod p?

Solving Quadratic Equations

In general, solving quadratic equations is:

- Easy over prime moduli
- As hard as factoring over composite moduli

Other Powers?

What about $x \rightarrow x^4 \bmod N$? $x \rightarrow x^6 \bmod N$?

The function $x \rightarrow x^3 \bmod N$ appears quite different

- Suppose 3 is relatively prime to $p-1$ and $q-1$
- Then $x \rightarrow x^3 \bmod p$ is injective for $x \neq 0$
  - Let a be such that $3a = 1 \bmod p-1$
  - $(x^3)^a = x^{1+k(p-1)} = x(x^{p-1})^k = x \bmod p$
- By CRT, $x \rightarrow x^3 \bmod N$ is injective for $x \in \mathbb{Z}_N^*$

x^3 mod N

What does injectivity mean?

Cannot base of factoring:

Adapt alg for square roots?

- Choose a random **z mod N**
- Compute **y = z^3 mod N**
- Run inverter on **y** to get a cube root **x**
- Let **p = GCD(z-x, N), q = N/p**

RSA Problem

Given

- **N = pq,**
- **e** such that **GCD(e,p-1)=GCD(e,q-1)=1**,
- $y=x^e \bmod N$ for a random **x**

Find **x**

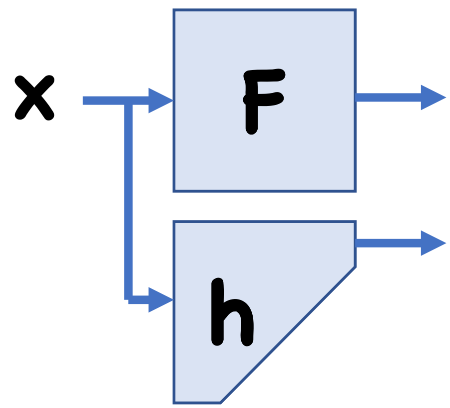
Injectivity means cannot base hardness on factoring, but still conjectured to be hard

RSA Assumption: For any algorithm  running in polynomial time, , ∃ negligible **ε** such that:

Pr[x←(N,x³ mod N)
N=pq and **p,q** random **λ**-bit primes s.t.
GCD(3,p-1)=GCD(3,q-1)=1
x←$\mathbb{Z}_N^*$] ≤ ε(λ)

Application: PRGs

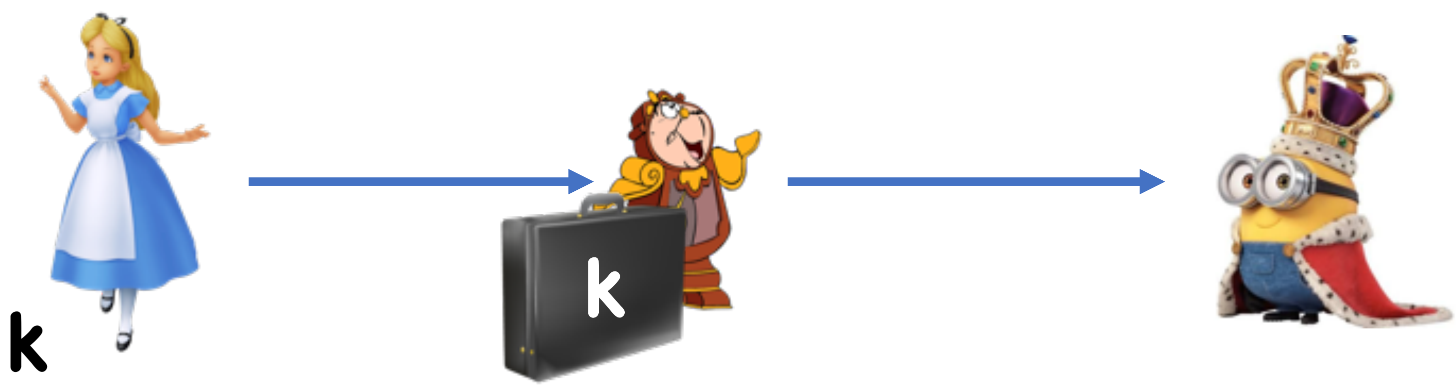
Let **F(x) = x³ mod N, h(x)** = least significant bit



Theorem: If **RSA** Assumption holds, then
G(x) = (F(x), h(x)) is a secure PRG

Public Key Cryptography

How do Alice & Bob get **k**?



Limitations

Time consuming

Not realistic in many situations

- Do you really want to send a courier to every website you want to communicate with

Doesn’t scale well

- Imagine 1M people communicating with 1M people

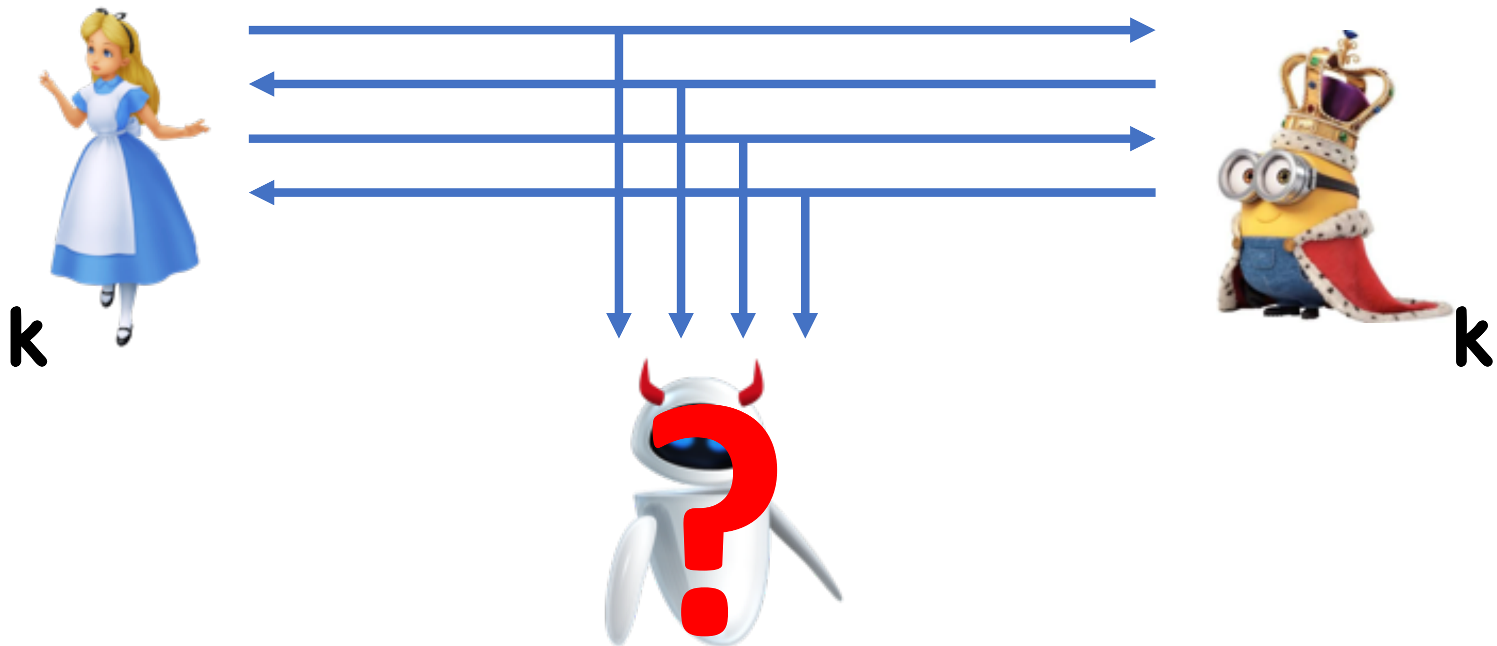
If not meeting in person, need to trust courier

Public Key Distribution

Public Key Distribution

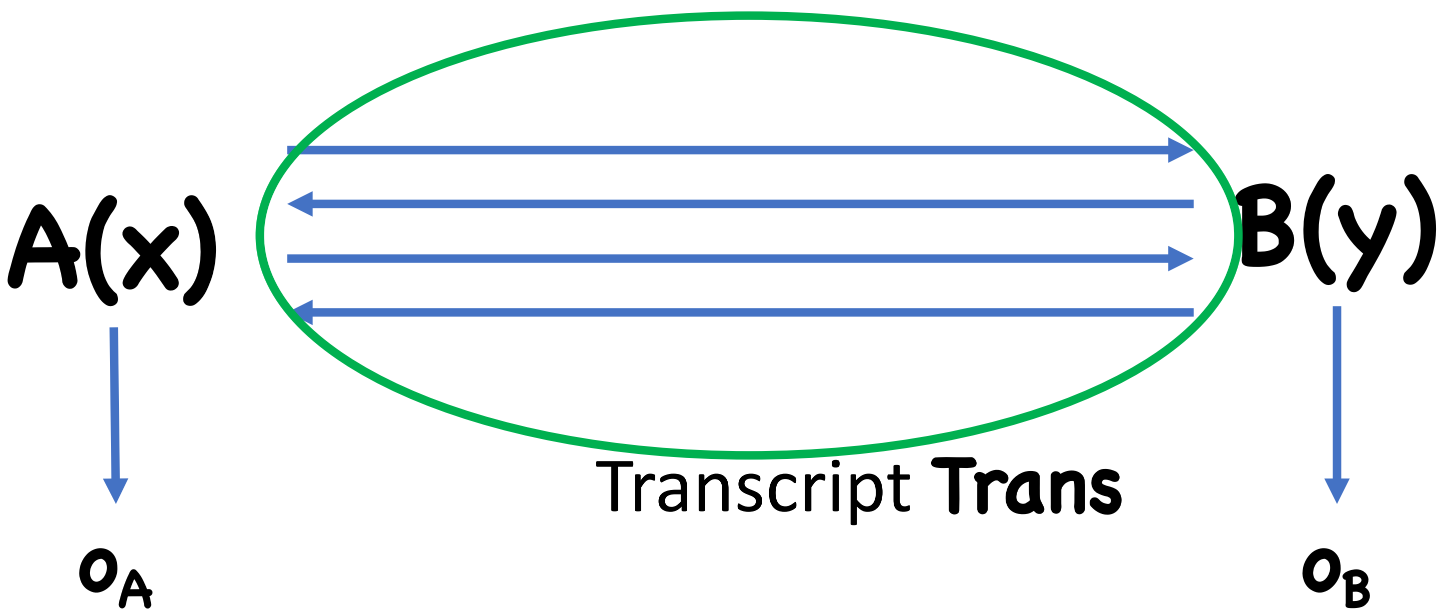
Public Key Distribution

Public Key Distribution



Interactive Protocols

Pair of interactive (randomized) algorithms **A, B**



Write $(\mathbf{Trans}, o_A, o_B) \leftarrow (A,B)(x,y)$

Public Key Distribution

Pair of interactive algorithms **A,B**

Correctness:

$$\Pr[o_A=o_B:\ (\text{Trans},o_A,o_B)\leftarrow(A,B)()] = 1$$

Shared key is $k := o_A=o_B$

- Define $(\text{Trans},k)\leftarrow(A,B)()$

Security: (Trans,k) is computationally indistinguishable from (Trans,k') where $k'\leftarrow K$ independent of k

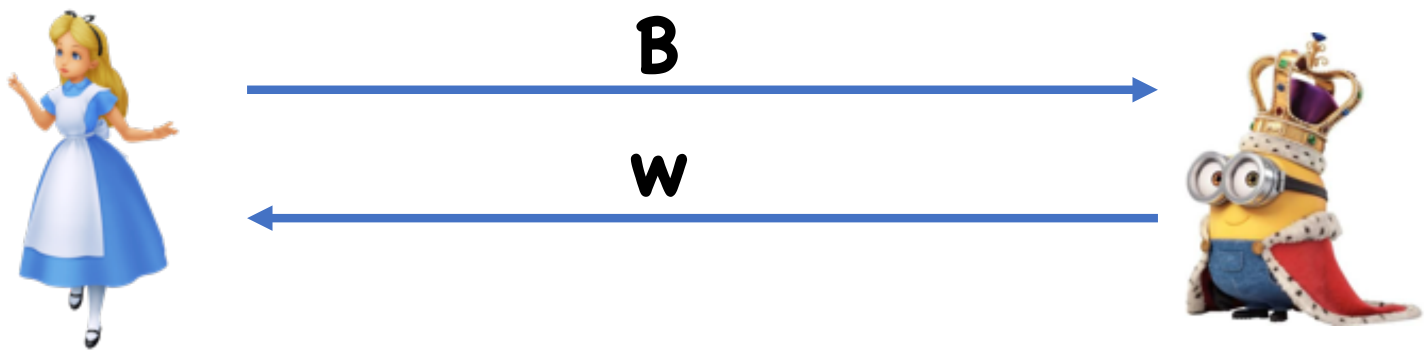
Matrix Multiplication Approach

B →

$B \leftarrow \mathbb{Z}_q^{\lambda\times\lambda}$

$A \leftarrow B^{-1}$

Matrix Multiplication Approach



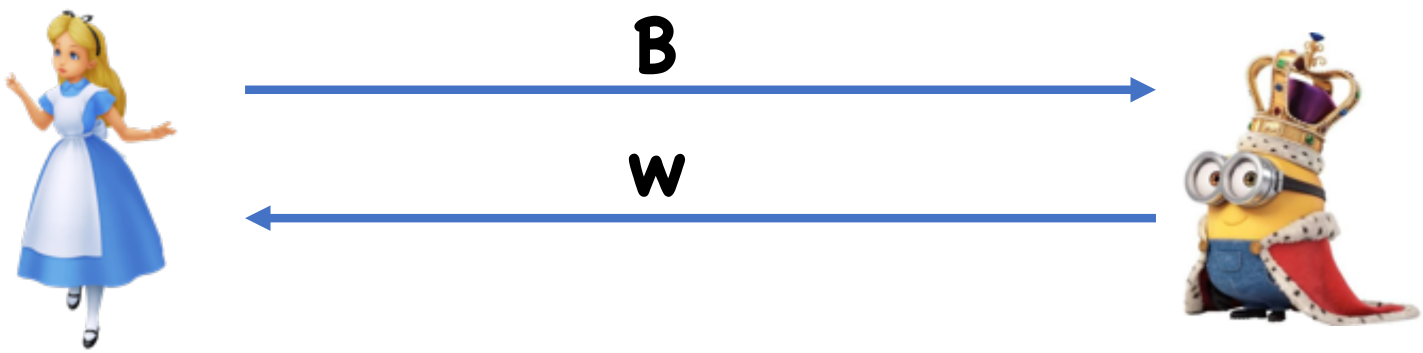
$\mathbf{B} \leftarrow \mathbb{Z}_q^{\lambda\times\lambda}$

$\mathbf{A} \leftarrow \mathbf{B}^{-1}$

$\mathbf{v} \leftarrow \mathbb{Z}_q^{\lambda}$

$\mathbf{w} \leftarrow \mathbf{B}\cdot\mathbf{v}$

Matrix Multiplication Approach

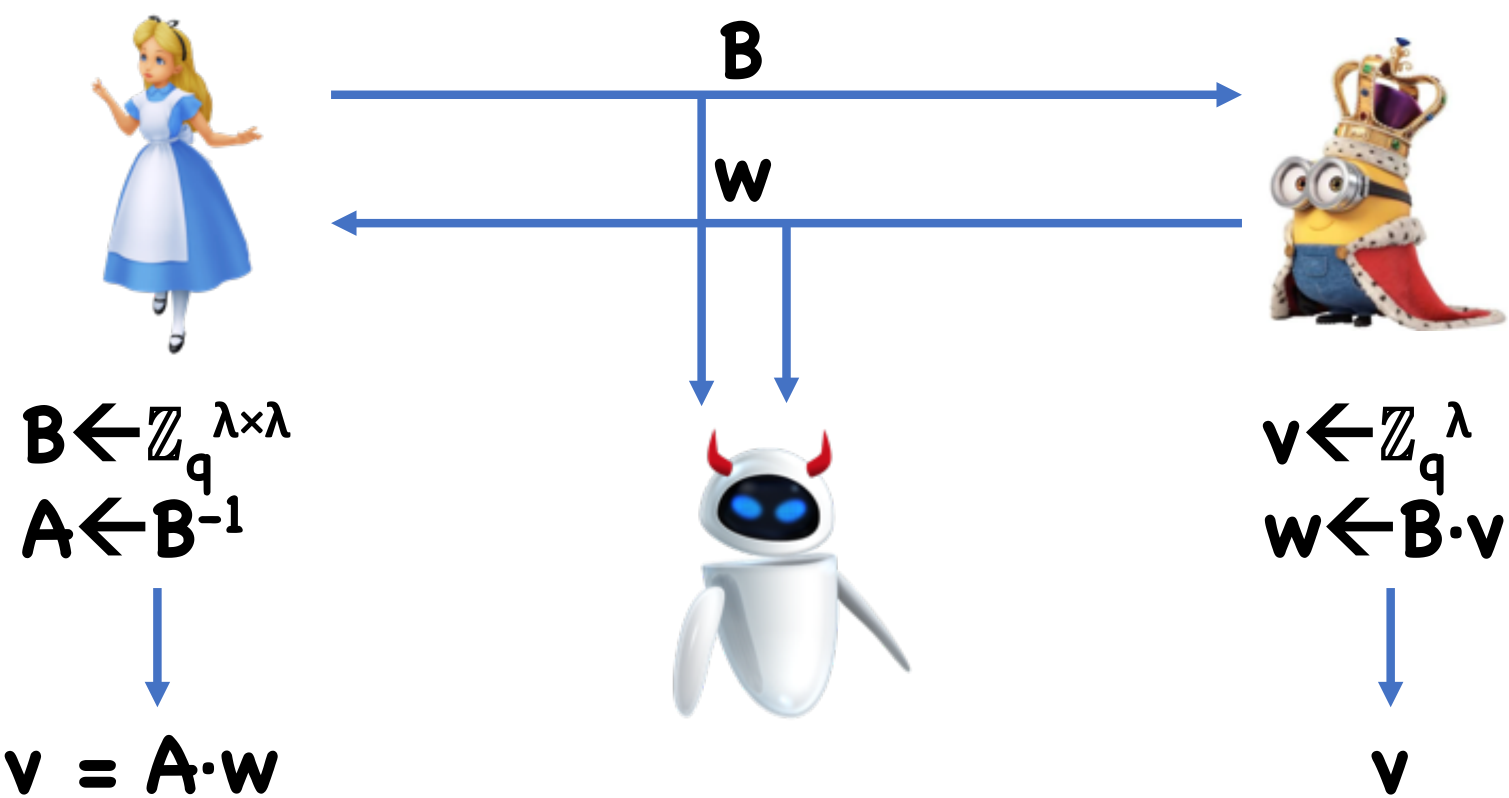


$B \leftarrow \mathbb{Z}_q^{\lambda\times\lambda}$ $v \leftarrow \mathbb{Z}_q^{\lambda}$

$A \leftarrow B^{-1}$ $w \leftarrow B\cdot v$

$v = A\cdot w$ v

Matrix Multiplication Approach



Running Times?

Bob: $\mathbf{O(\lambda^2)}$
Eve: $\mathbf{O(\lambda^3)}$

Running Times?

Bob: $O(\lambda^2)$
Eve: $O(\lambda^\omega)$ where $\omega \leq 2.373$
Alice: $O(\lambda^\omega)$

Different Approach:

- Start with $A = B = I$
- Repeatedly apply random elementary row ops to A, inverse to B
- Output (A,B)

Running Times?

Bob: $\mathbf{O(\lambda^2)}$

Eve: $\mathbf{O(\lambda^\omega)}$ where $\boldsymbol{\omega \leq 2.373}$

Alice: $\mathbf{O(\lambda^\omega)}$

Assuming Matrix Multiplication exponent $\boldsymbol{\omega > 2}$, adversary must work harder than honest users

inverse to **B**

- Output **(A,B)**

Merkle Puzzles

Let **H** be some hash function with domain $[\lambda]=\{1,\ldots,\lambda\}$

$\{A_i\}$ →

← $\{B_i\}$

$a_1,\ldots,a_t \leftarrow [\lambda]$
$A_i \leftarrow H(a_i)$

$b_1,\ldots,b_t \leftarrow [\lambda]$
$b_i \leftarrow H(b_i)$

Merkle Puzzles

Let **H** be some hash function with domain $[\lambda]=\{1,\ldots,\lambda\}$

$\{A_i\}$ →

← $\{B_i\}$

$a_1,\ldots,a_t \leftarrow [\lambda]$

$A_i \leftarrow H(a_i)$

↓

a_i s.t. $A_i \in \{B_i\}$

$b_1,\ldots,b_t \leftarrow [\lambda]$

$b_i \leftarrow H(b_i)$

↓

b_i s.t. $B_i \in \{A_i\}$

Analysis

Protocol succeeds iff:

- $\mathbf{H}$ is injective (why?)
- $\{A_i\}\cap\{B_i\}\neq\varnothing$ (equiv, $\{a_i\}\cap\{b_i\}\neq\varnothing$)

What does t need to be to make $\{A_i\}\cap\{B_i\}\neq\varnothing$?

If adversary can only query $\mathbf{H}$ on various inputs, how many queries needed?

Limitations

Both matrix multiplication and Merkle puzzle approaches have a polynomial gap between honest users and adversaries

To make impossible for extremely powerful adversaries, need at least $\lambda^2 > 2^{80}$

- Special-purpose hardware means λ needs to be even bigger
- Honest users require time at least $\lambda=2^{40}$
- Possible, but expensive

Limitations

Instead, want want a super-polynomial gap between honest users and adversary

- Just like everything else we've seen in the course

Key Distribution from Obfuscation

Software obfuscation:

- Compile programs into unreadable form

  (intentionally)

```
@P=split//,".URRUU\c8R";@d=split//,"\nrekcah xinU / lreP rehtona tsuJ";sub p{
@p{"r$p","u$p"}=(P,P);pipe"r$p","u$p";++$p;($q*=2)+=$f=!fork;map{$P=$P[$f^ord
($p{$_})&6];$p{$_}=/ ^$P/ix?$P:close$_}keys%p}p;p;p;p;p;map{$p{$_}=~/^[P.]/&&
close$_}%p;wait until$?;map{/^r/&&<$_>}%p;$_=$d[$q];sleep rand(2)if/\S/;print
```

Key Distribution from Obfuscation

Let **F,F^{-1}** be a block cipher

P

k←{0,1}$^{\lambda}$
P←Obf(F(k, ·))

Key Distribution from Obfuscation

Let F, F^{-1} be a block cipher

$k \leftarrow \{0,1\}^\lambda$

$P \leftarrow Obf(\ F(k,\ \cdot)\ )$

$r \leftarrow \{0,1\}^\lambda$

$x \leftarrow P(r)$

Key Distribution from Obfuscation

Let **F,F^{-1}** be a block cipher

P →

← x

$k \leftarrow \{0,1\}^\lambda$	$r \leftarrow \{0,1\}^\lambda$
$P \leftarrow Obf(\ F(k,\ \cdot)\ )$	$x \leftarrow P(r)$
↓	↓
$r \leftarrow F^{-1}(k,x)$	r

Key Distribution From Obfuscation

For decades, many attempts at commercial code obfuscators

- Simple operations like variable renaming, removing whitespace, re-ordering operations

Really only a “speed bump” to determined adversaries

- Possible to recover something close to original program (including cryptographic keys)

Don’t use commercially available obfuscators to hide cryptographic keys!

Key Distribution From Obfuscation

Recently (2013), new type of obfuscator has been developed

- Much stronger security guarantees
- Based on mathematical tools
- Many cryptographic applications beyond public key distribution

Downside?

- Extraordinarily impractical (currently)

Practical Key Exchange

Instead of obfuscating a general PRP, we will define a specific abstraction that will enable key agreement

Then, we will show how to implement the abstraction using number theory

Trapdoor Permutations

Domain $\mathbf{X}$

Gen(): outputs **(pk,sk)**
$F(pk, x \in X) = y \in X$
$F^{-1}(sk, y) = x$

Correctness:
$\Pr[\ F^{-1}(sk,\ F(pk,\ x)) = x : (pk,sk) \leftarrow Gen()\ ] = 1$

Correctness implies F, F^{-1} are deterministic, permutations

Trapdoor Permutation Security

pk,y

(sk,pk)←Gen()
x←X
y←F(pk,x)

x'

Adversary wins if **x=x'**

In other words, **F(pk, ·)** is a one-way function

Key Distribution from TDPs

$(pk,sk) \leftarrow Gen()$

pk

$y \leftarrow F(pk,x)$

$x \leftarrow X$

$x \leftarrow F^{-1}(sk,y)$

x

Analysis

Correctness follows from correctness of TDP

Security:

- By TDP security, adversary cannot compute **x**
- However, **x** is distinguishable from a random key

Hardcore Bits

Let **F** be a one-way function with domain **D**, range **R**

Definition: A function **h:D→{0,1}** is a hardcore bit for **F** if, for any polynomial time , ∃ negligible **ε** such that:

| Pr[1←(F(x), h(x)), x←D]
– Pr[1←(F(x), b), x←D,b←{0,1}] | ≤ ε(λ)

In other words, even given **F(x)**, hard to guess **h(x)**

Examples of Hardcore Bits

Define **$lsb(x)$** as the least significant bit of **x**

For $x \in Z_N$, define **$Half(x)$** as **1** iff $0 \leq x < N/2$

Theorem: Let p be a prime, and $F: Z_p^* \rightarrow Z_p^*$ be $F(g,x) = (g, g^x \bmod p)$

Half is a hardcore bit for **F** (assume **F** is one-way)

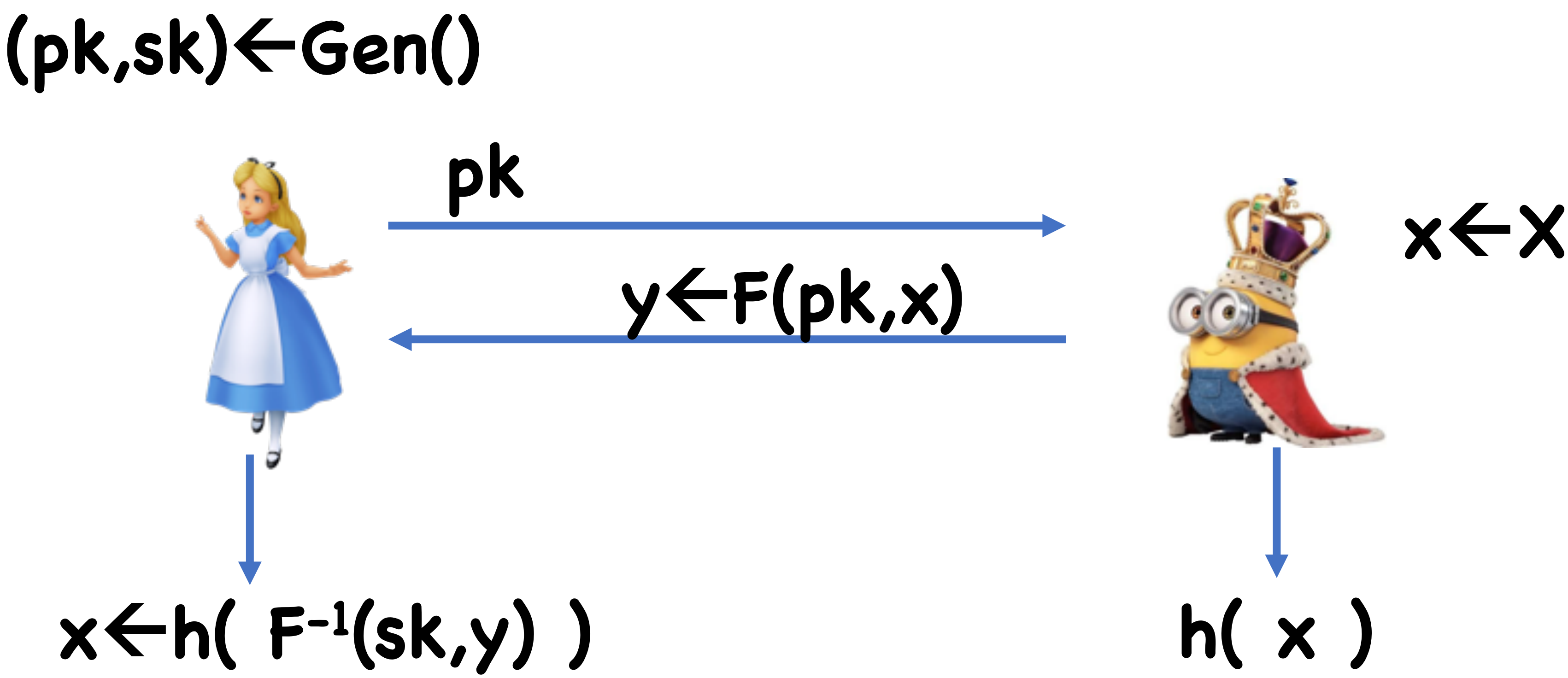
Theorem: Let **N** be a product of two large primes **p,q**, and $F: Z_N^* \rightarrow Z_N^*$ be $F(x) = x^e \bmod N$ for some e relatively prime to $(p-1)(q-1)$

Lsb and Half are hardcore bits for **F** (assuming RSA)

Theorem: Let **N** be a product of two large primes **p,q**, and $F: Z_N^* \rightarrow Z_N^*$ be $F(x) = x^2 \bmod N$

Lsb and Half are hardcore bits for **F** (assuming factoring)

Key Distribution from TDPs



h a hardcore bit for **F(pk, ·)**

Theorem: If **h** is a hardcore bit for **F(pk, ·),** then protocol is secure

Proof:

- **(Trans,k) = ((pk,y), h(x))**
- Hardcore bit means indist. from **((pk,y), b)**

Trapdoor Permutations from RSA

Gen():

- Choose random primes **p,q**
- Let **N=pq**
- Choose **e,d** .s.t **ed=1 mod (p-1)(q-1)**
- Output **pk=(N,e), sk=(N,d)**

F(pk,x): Output $y = x^e \bmod N$

F^{-1}(sk,c): Output $x = y^d \bmod N$

Caveats

RSA is not a true TDP as defined

- Why???
- What's the domain?

Nonetheless, distinction is not crucial to most applications

- In particular, works for key agreement protocol

Other TDPs?

For long time, essentially none known

- Still interesting object:
  - Useful abstraction in protocol design
  - Maybe more will be discovered…

Using obfuscation:

- Let **P** be a PRP
- **sk = k, pk = Obf(P(k, ·))**

Key Distribution from DH

Everyone agrees on group **G** of prime order **p**

$a \leftarrow \mathbb{Z}_p$

$b \leftarrow \mathbb{Z}_p$

Key Distribution from DH

Everyone agrees on group **G** or prime order **p**

$a \leftarrow \mathbb{Z}_p$  $g^a \rightarrow$ $\leftarrow g^b$  $b \leftarrow \mathbb{Z}_p$

Key Distribution from DH

Everyone agrees on group **G** or prime order **p**

g^a →

← g^b

$a \leftarrow \mathbb{Z}_p$

$b \leftarrow \mathbb{Z}_p$

$k = (g^b)^a = g^{ab}$

$k = (g^a)^b = g^{ab}$

Key Distribution from DH

Theorem: If **(t,ε)–**DDH holds on **G**, then the Diffie-Hellman protocol is **(t,ε)–**secure

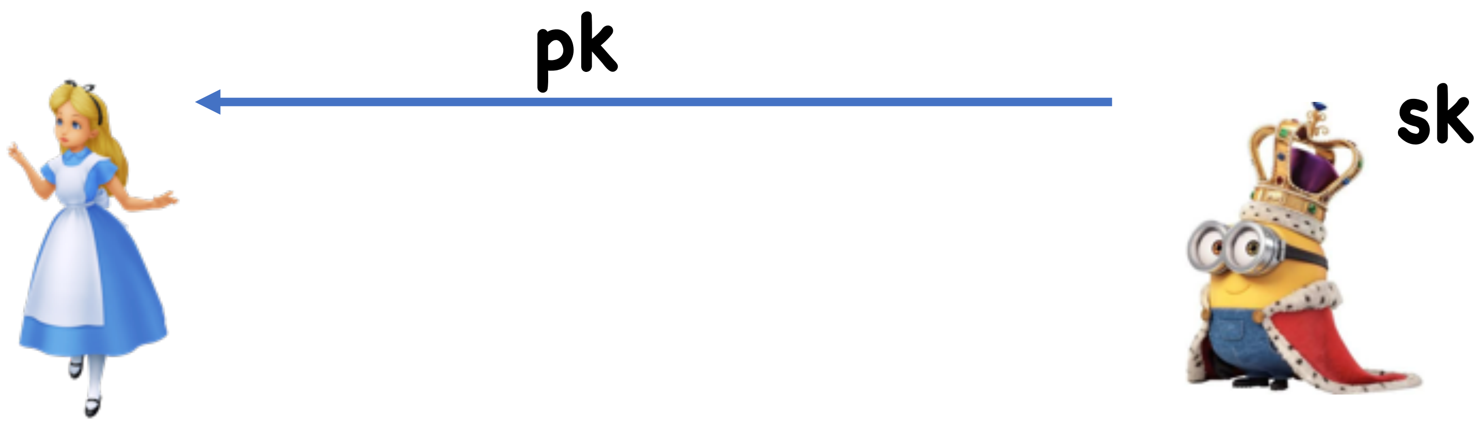
Proof:

- **$(Trans,k) = ((g^a,g^b),\ g^{ab})$**
- DDH means indistinguishable from **$((g^a,g^b),\ g^c)$**

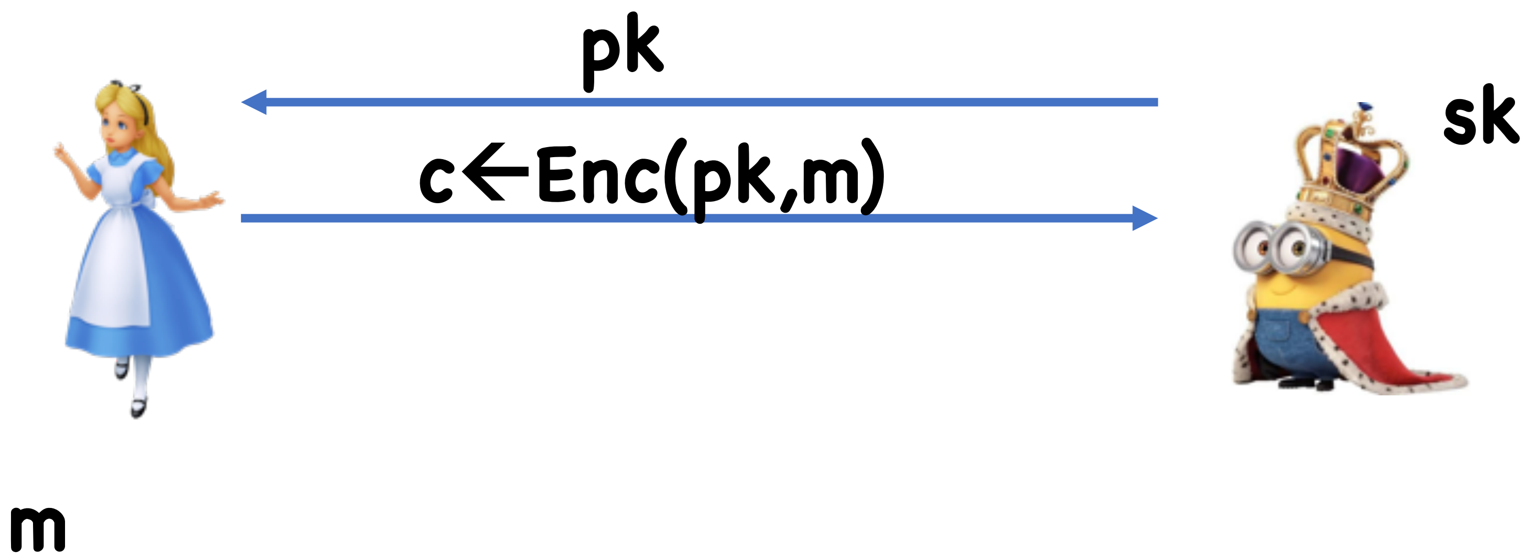
What if only CDH holds, but DDH is easy?

Public Key Encryption

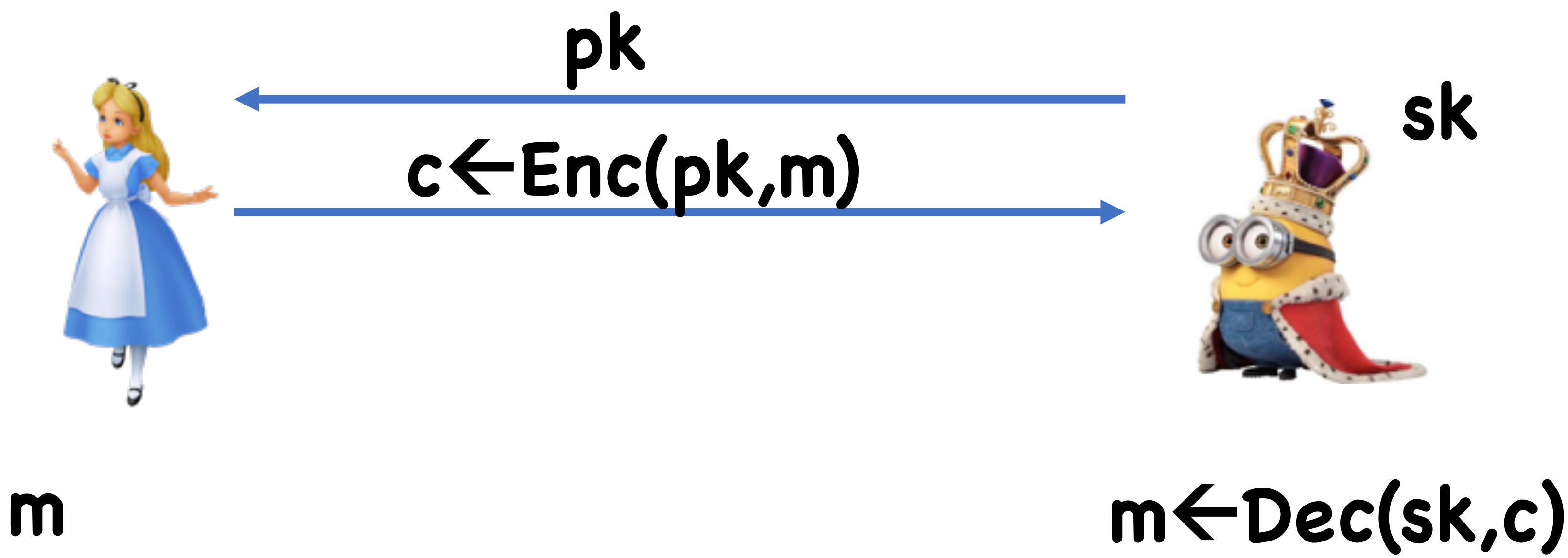
Public Key Encryption



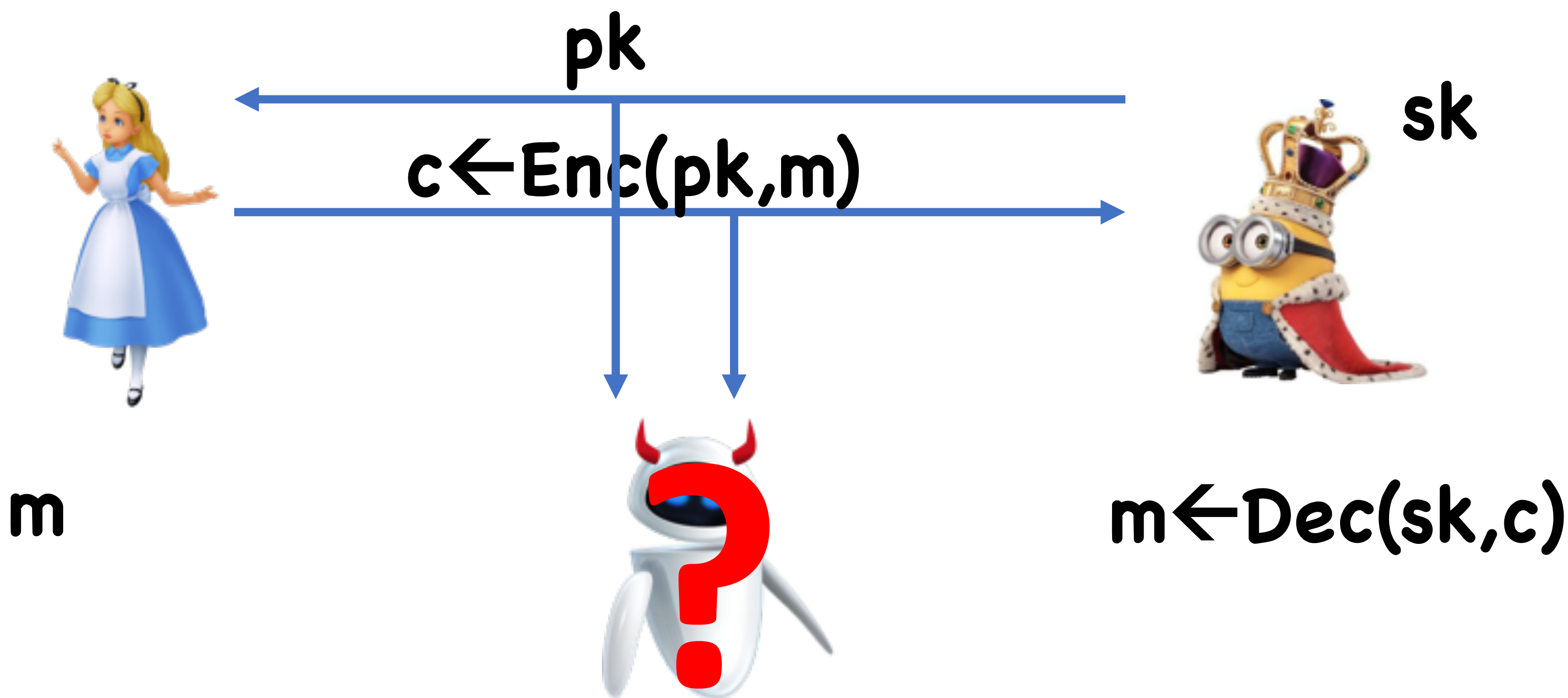
Public Key Encryption



Public Key Encryption



Public Key Encryption

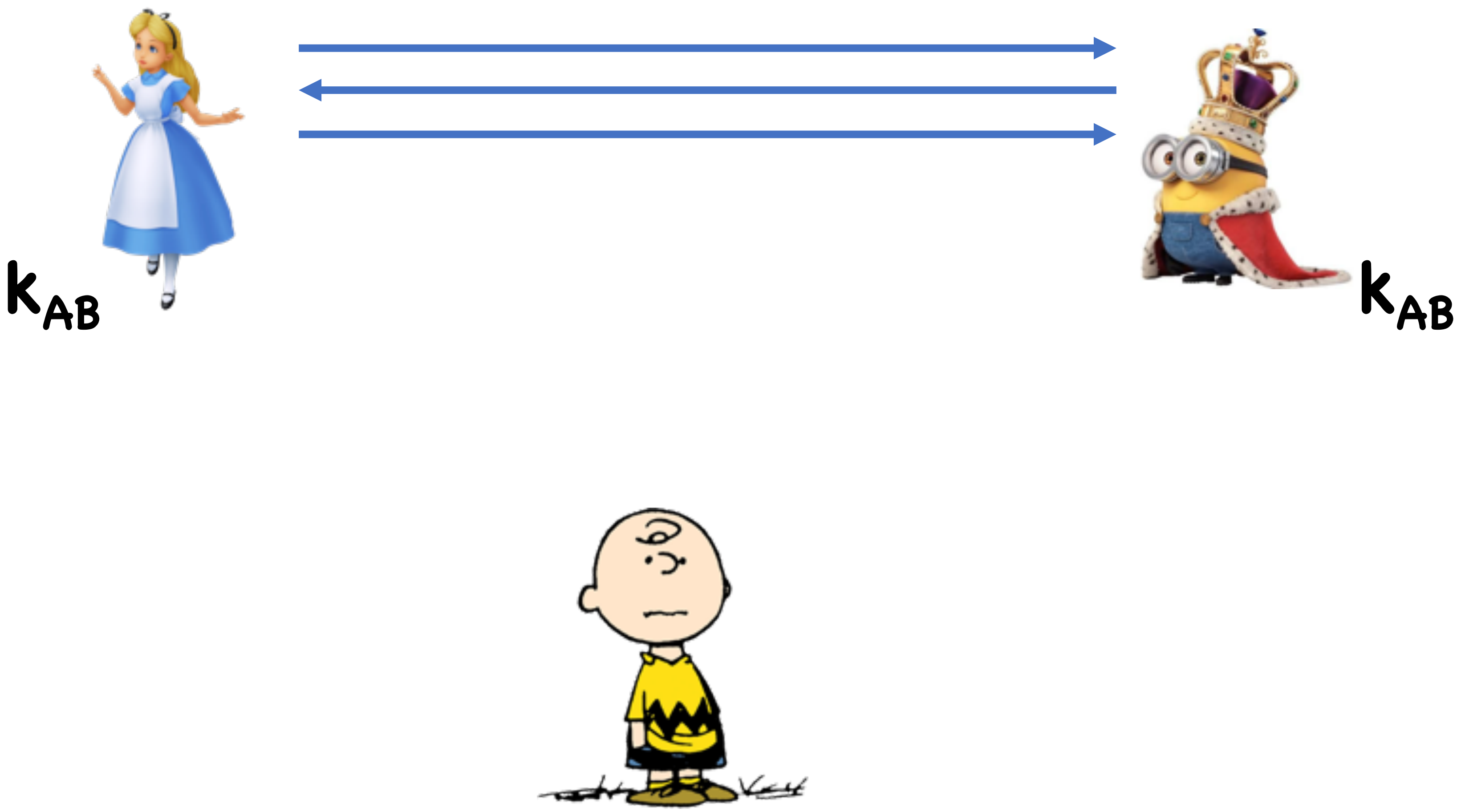


PKE vs Key Agreement

Key agreement:

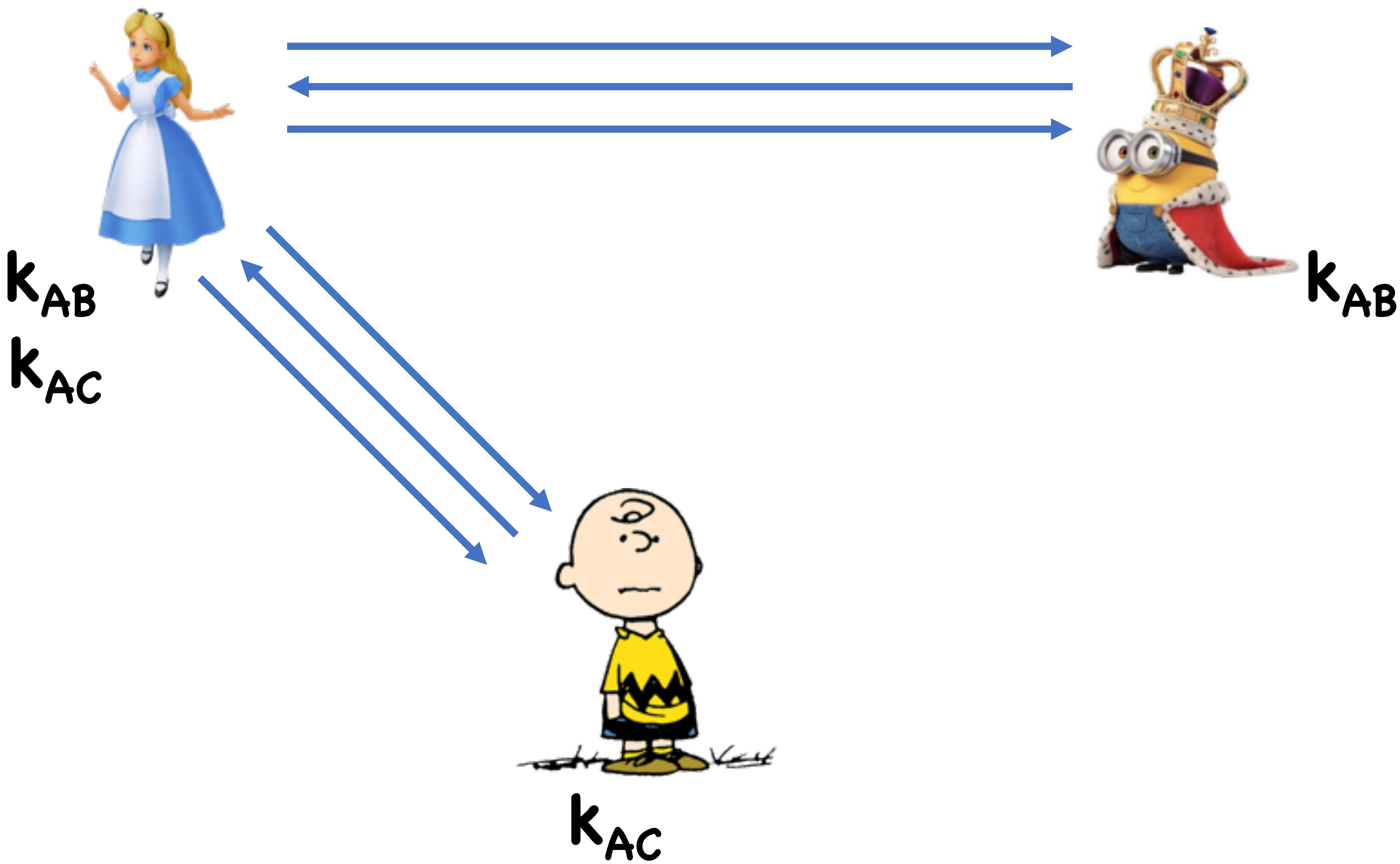
PKE vs Key Agreement

Key agreement:



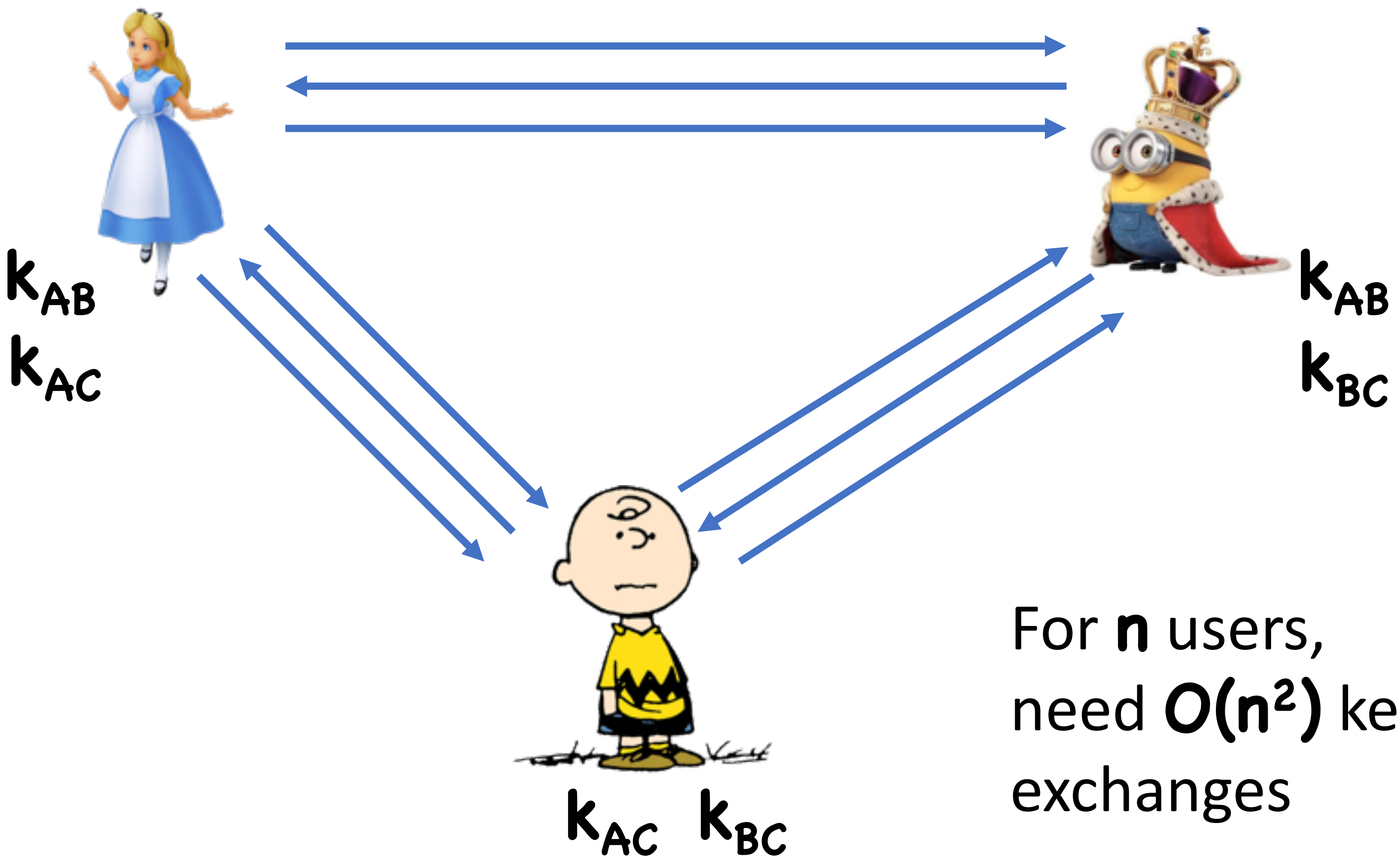
PKE vs Key Agreement

Key agreement:



PKE vs Key Agreement

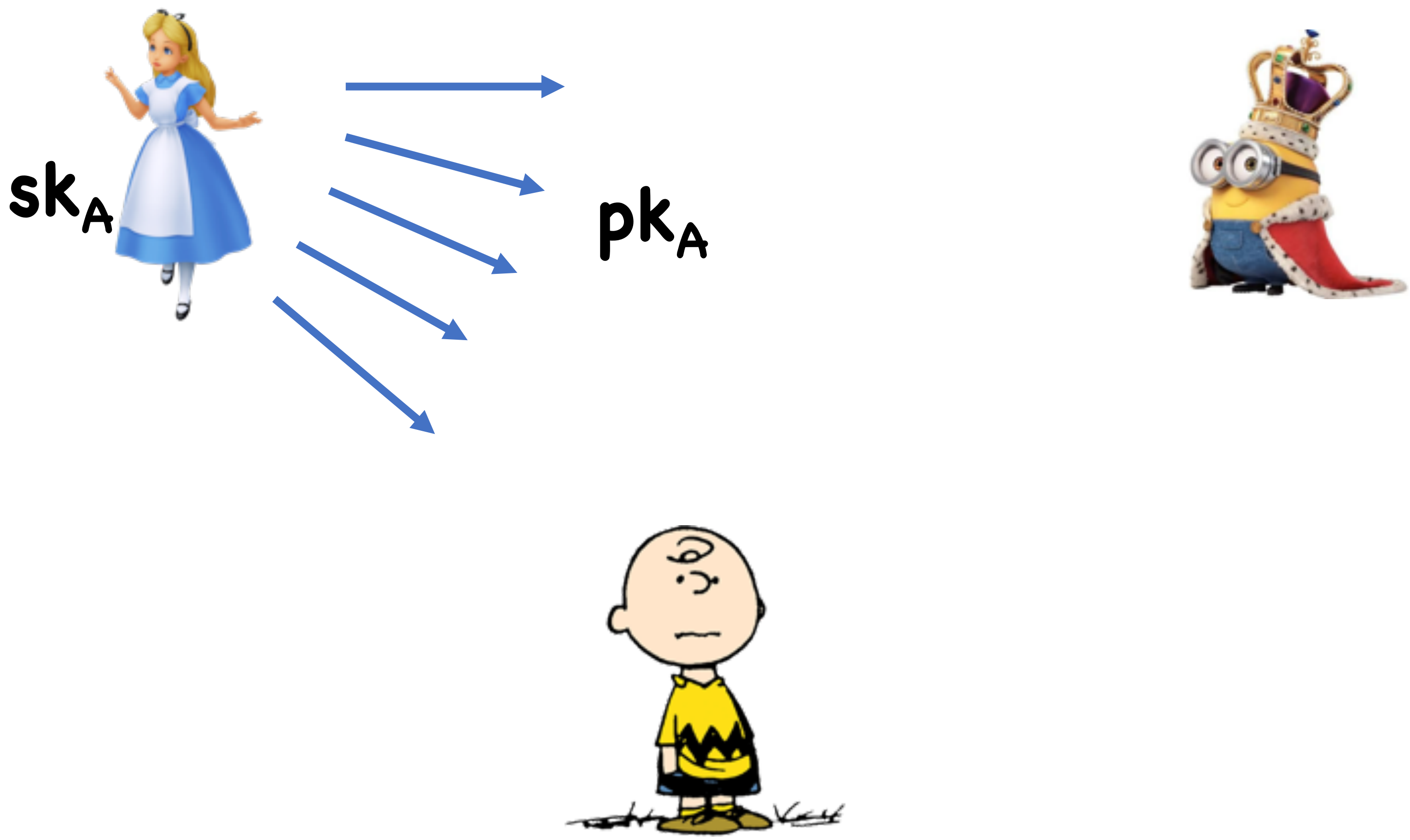
Key agreement:



For **n** users, need **O(n²)** key exchanges

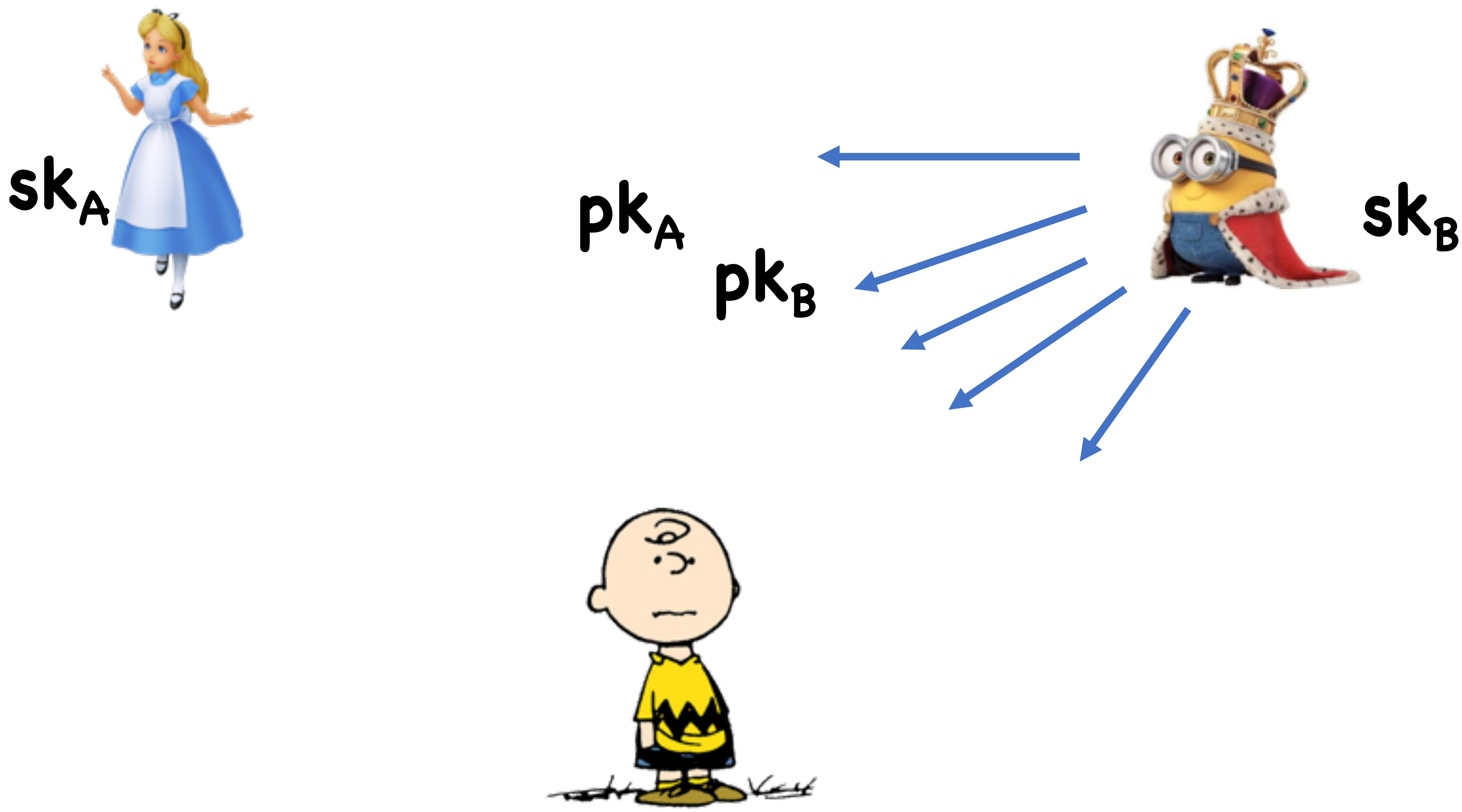
PKE vs Key Agreement

PKE:



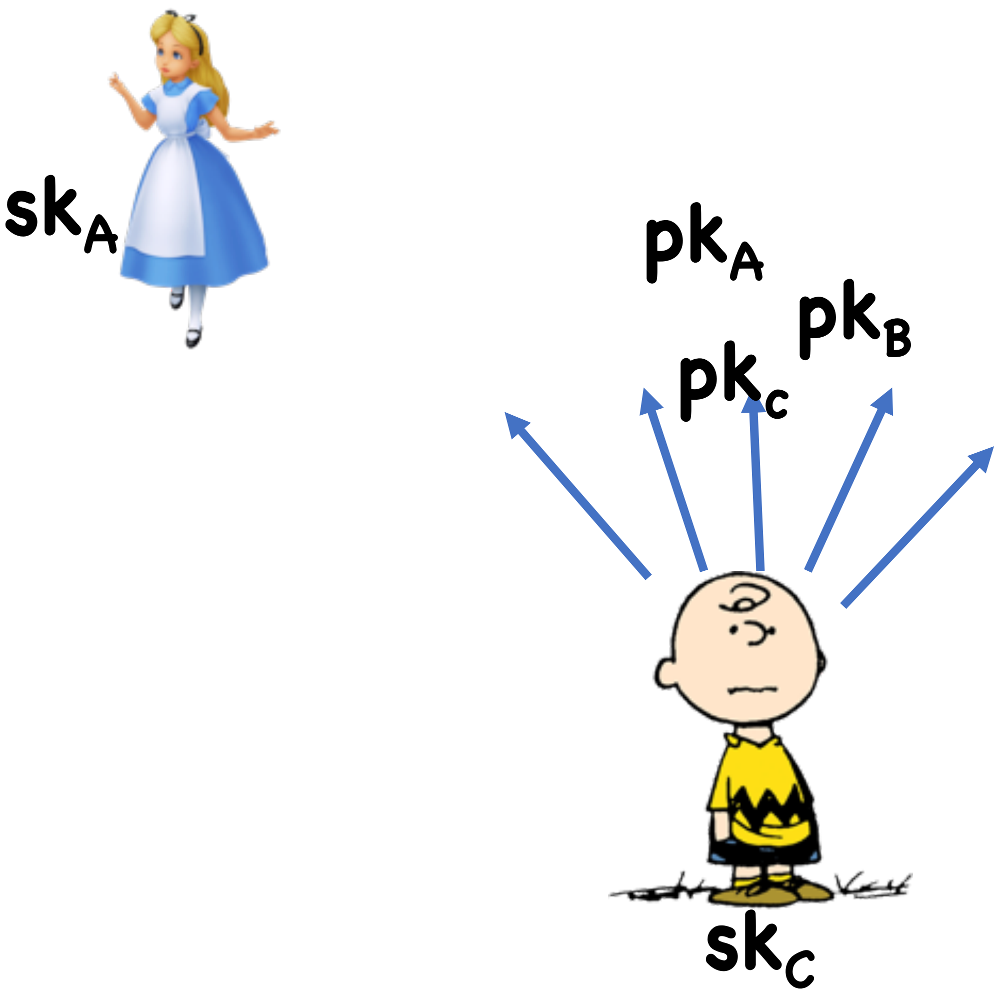
PKE vs Key Agreement

PKE:



PKE vs Key Agreement

PKE:



For **n** users,
need **O(n)**
public keys

PKE Syntax

Message space **M**

Algorithms:

- **(sk,pk)←Gen(λ)**
- **Enc(pk,m)**
- **Dec(sk,m)**

Correctness:

Pr[Dec(sk,Enc(pk,m)) = m: (sk,pk)←Gen(λ)] = 1

Security

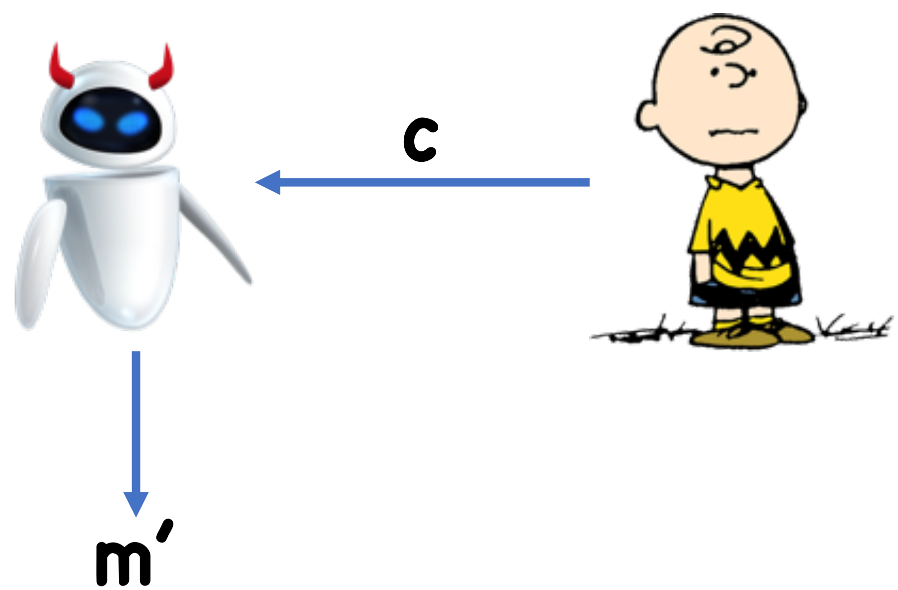
One-way security

Semantic Security

CPA security

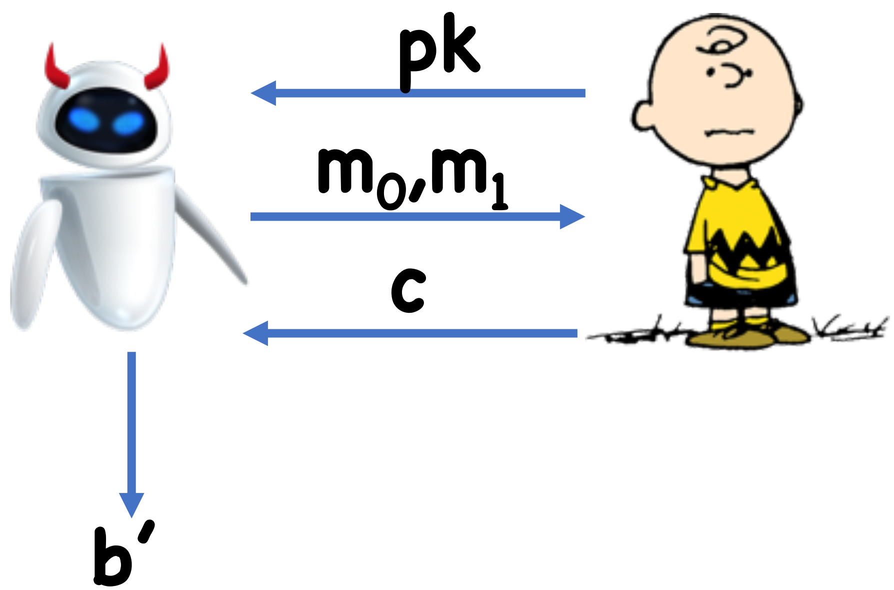
CCA Security

One-way Security



(sk,pk)←Gen()
m←M
c←Enc(pk,m)

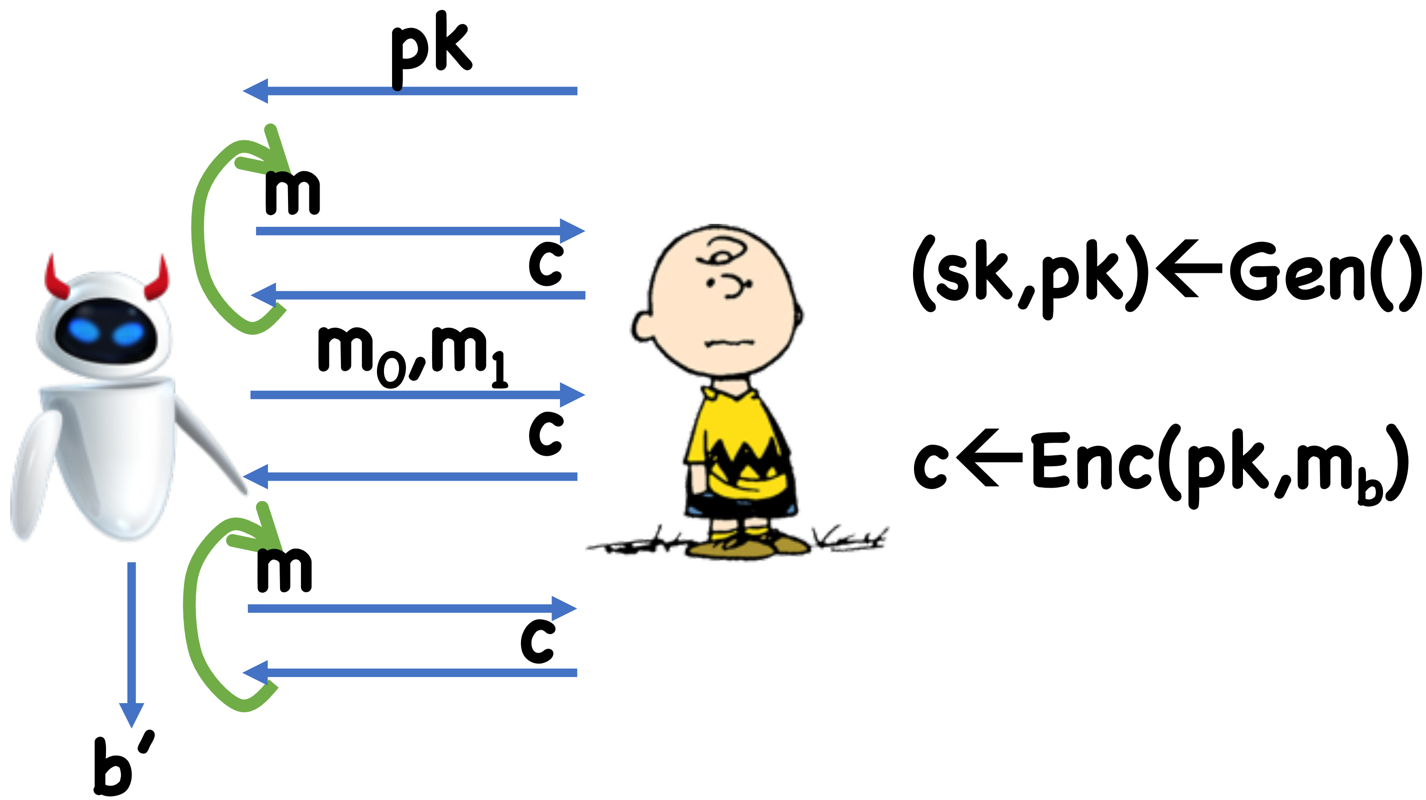
Semantic Security



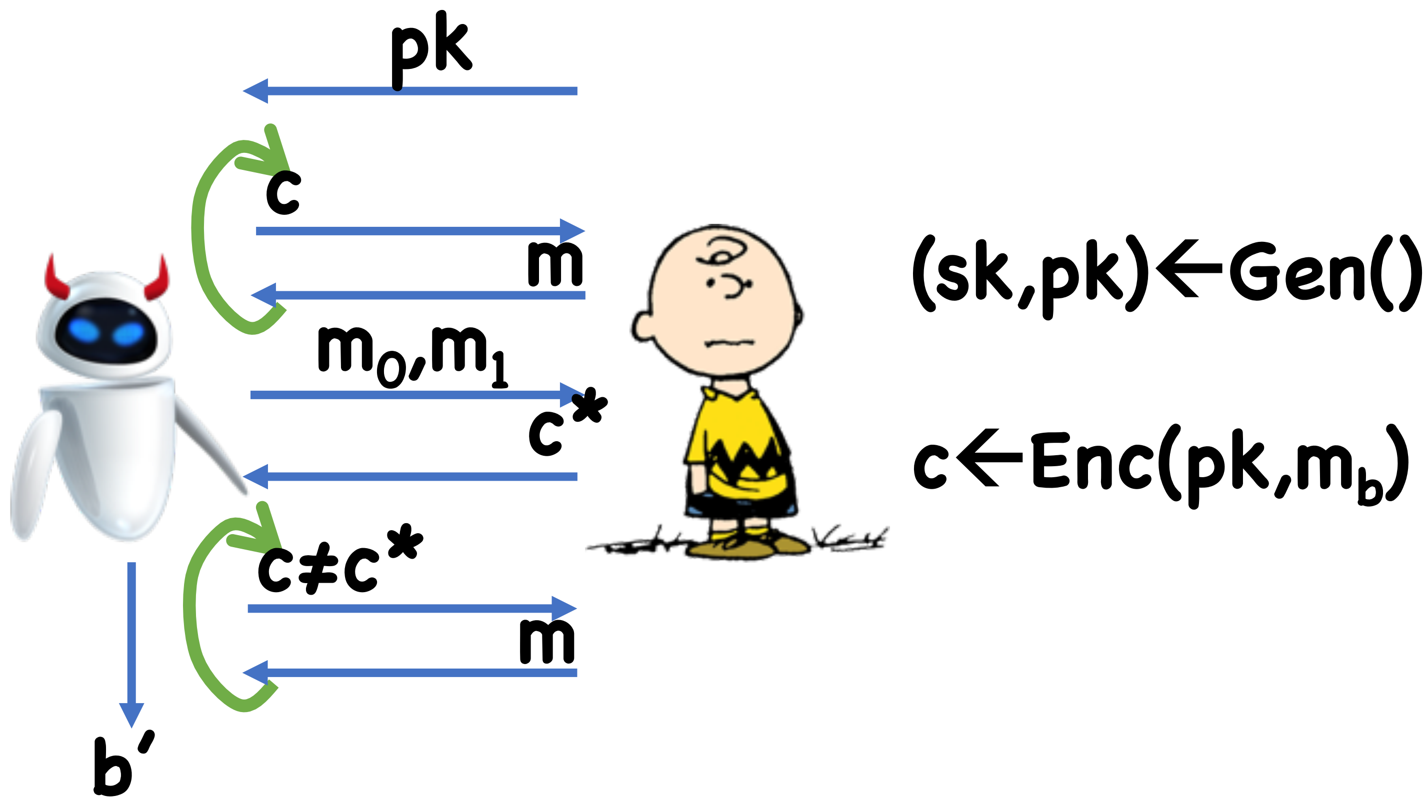
$(sk,pk) \leftarrow Gen()$

$c \leftarrow Enc(pk,m_b)$

CPA Security



CCA Security



Reminders

HW5 Due Today

PR2 Due April 19th