

# COS433/Math 473: Cryptography

Mark Zhandry

Princeton University

Spring 2017

Previously...

# One-way Functions


The minimal assumption for crypto

Syntax:

- Domain  **$\mathcal{D}$**
- Range  **$\mathcal{R}$**
- Function  **$F: \mathcal{D} \rightarrow \mathcal{R}$**

No correctness properties other than deterministic


# Security

**Definition:**  $F$  is  $(t, \epsilon)$ -One-Way if, for all  running in time at most  $t$ ,

$$\Pr[F(x) = F(y) : y \leftarrow \text{pirate}(F(x)), x \leftarrow D] < \epsilon$$

# Hardcore Bits

Let  $\mathbf{F}$  be a one-way function with domain  $\mathbf{D}$ , range  $\mathbf{R}$

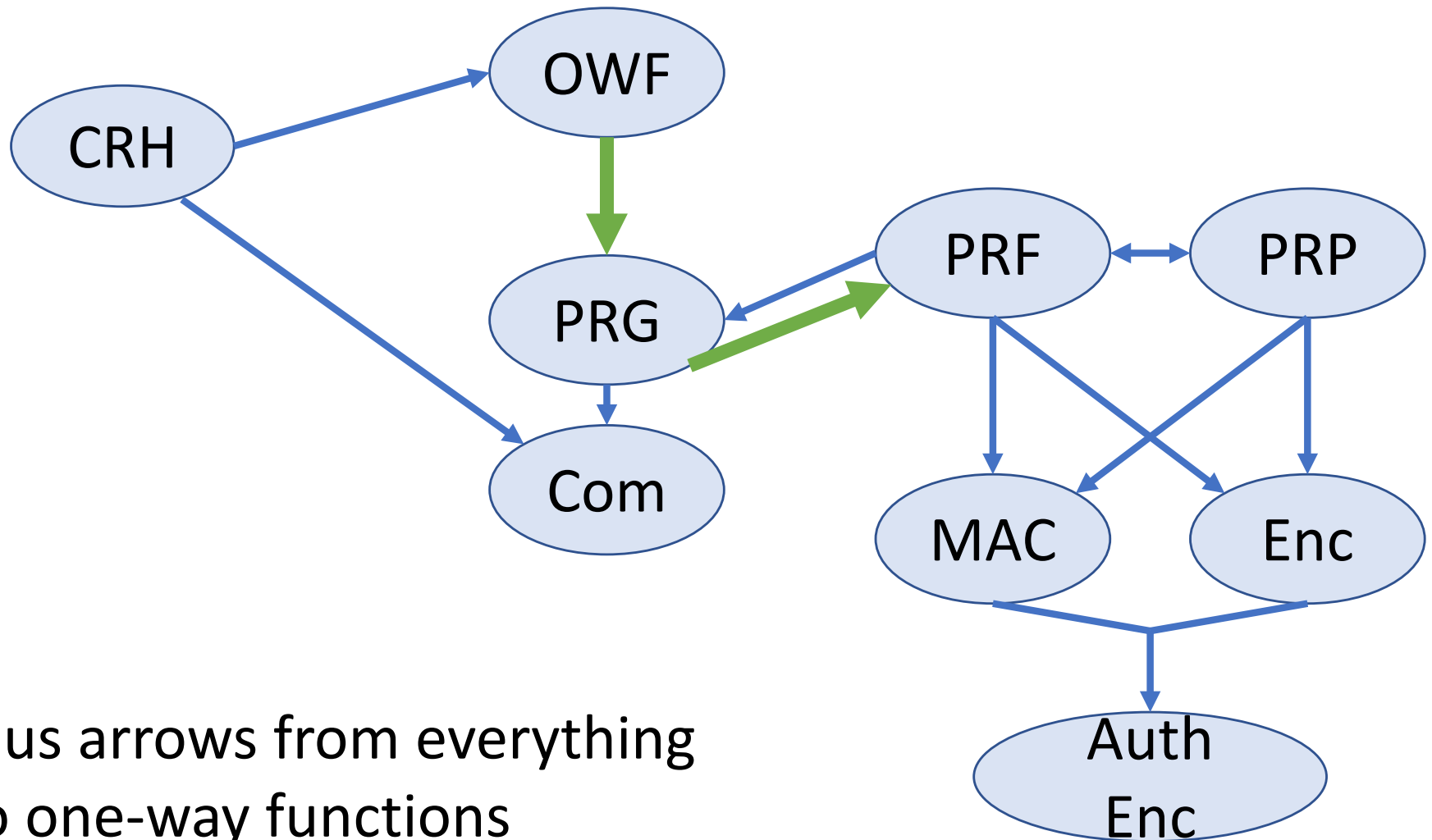
**Definition:** A function  $\mathbf{h}:\mathbf{D}\rightarrow\{0,1\}$  is a  $(t,\epsilon)$ -hardcore bit for  $\mathbf{F}$  if, for any  running in time at most  $t$ ,

$$| \Pr[1 \leftarrow \text{robot}(F(x), h(x)), x \leftarrow D]$$

$$- \Pr[1 \leftarrow \text{robot}(F(x), b), x \leftarrow D, b \leftarrow \{0,1\}] | \leq \epsilon$$

In other words, even given  $\mathbf{F(x)}$ , hard to guess  $\mathbf{h(x)}$

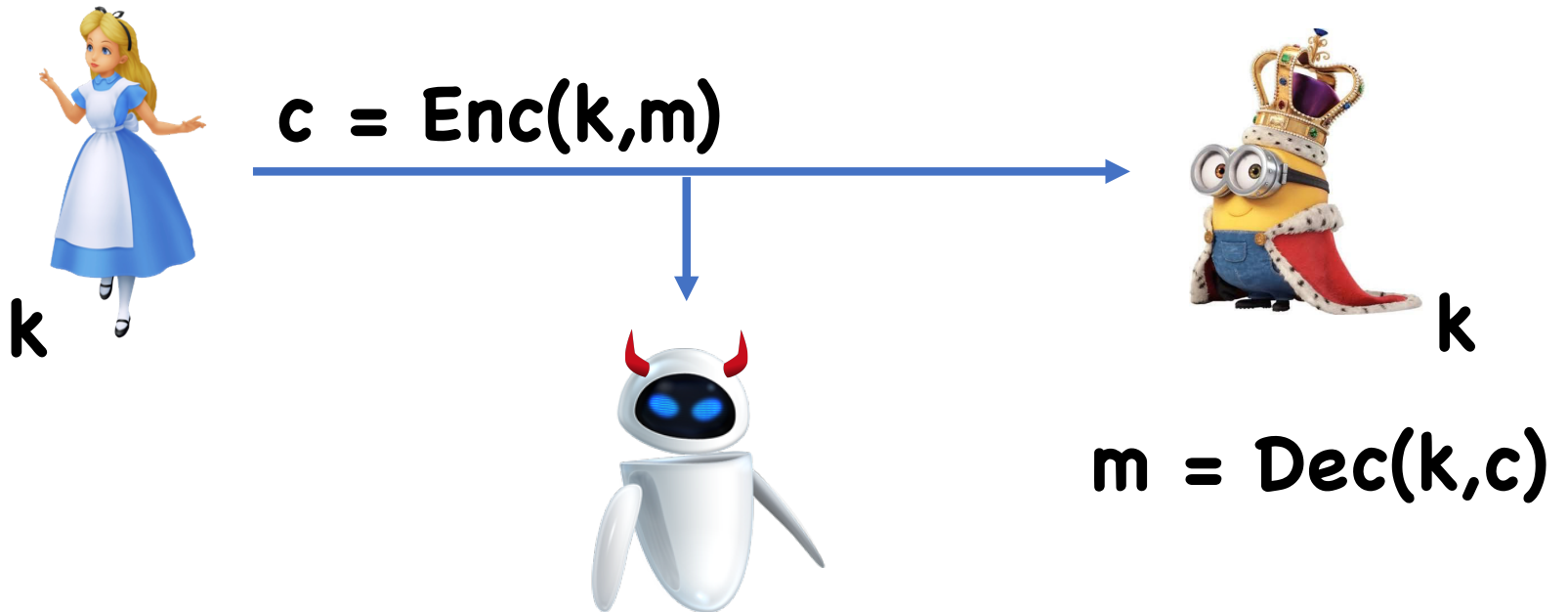
# So Far



# Today

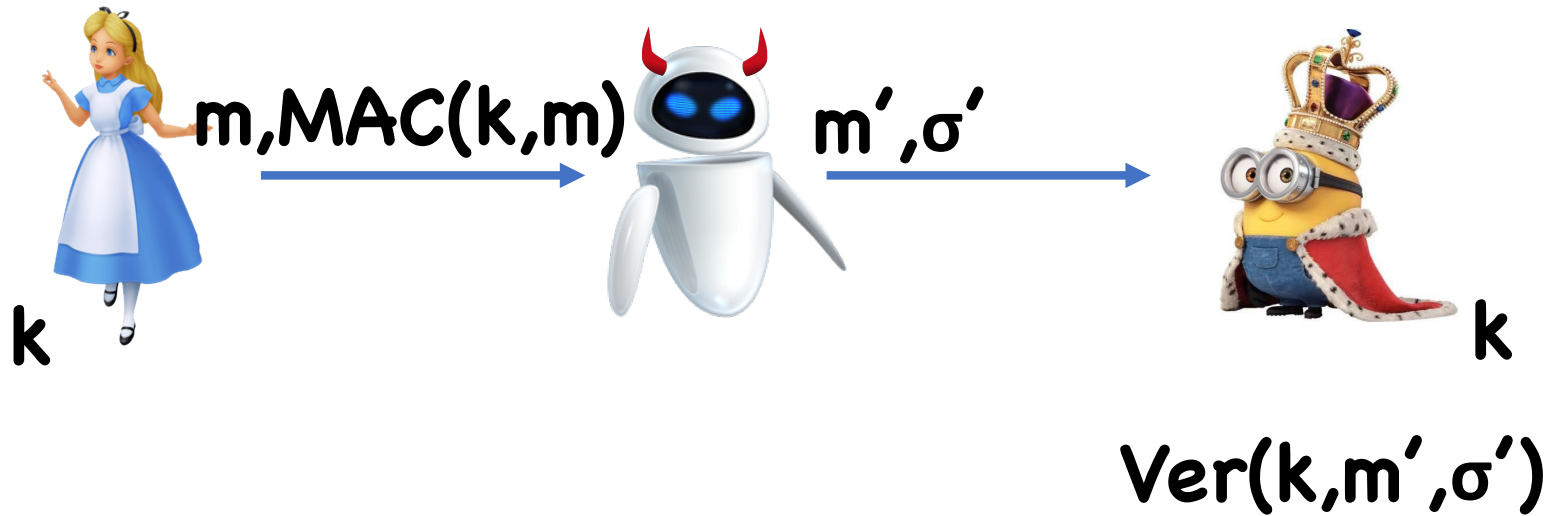
## Exchanging keys

# Previously





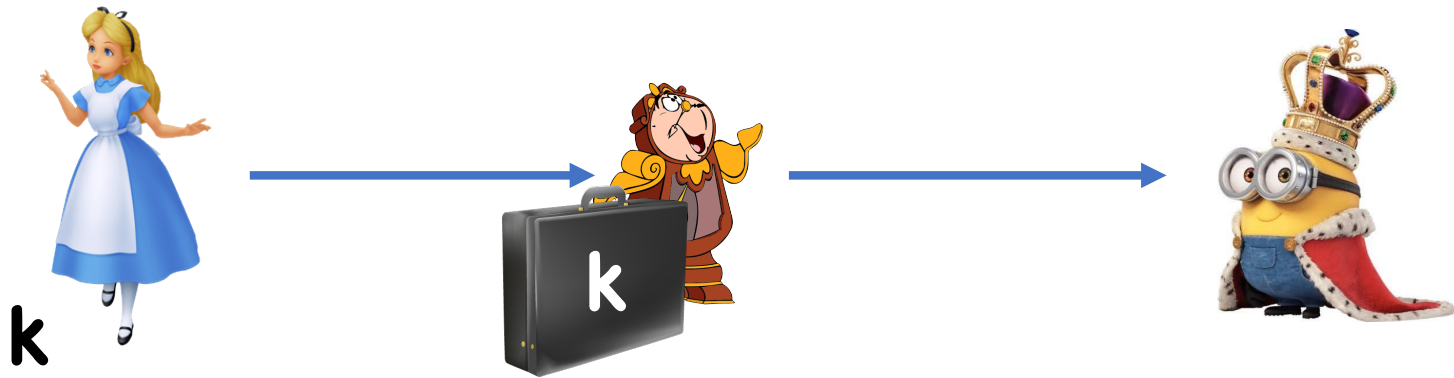
# Previously



# Today

Where do Alice and Bob get their shared key from?

# Traditional Approach



# 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

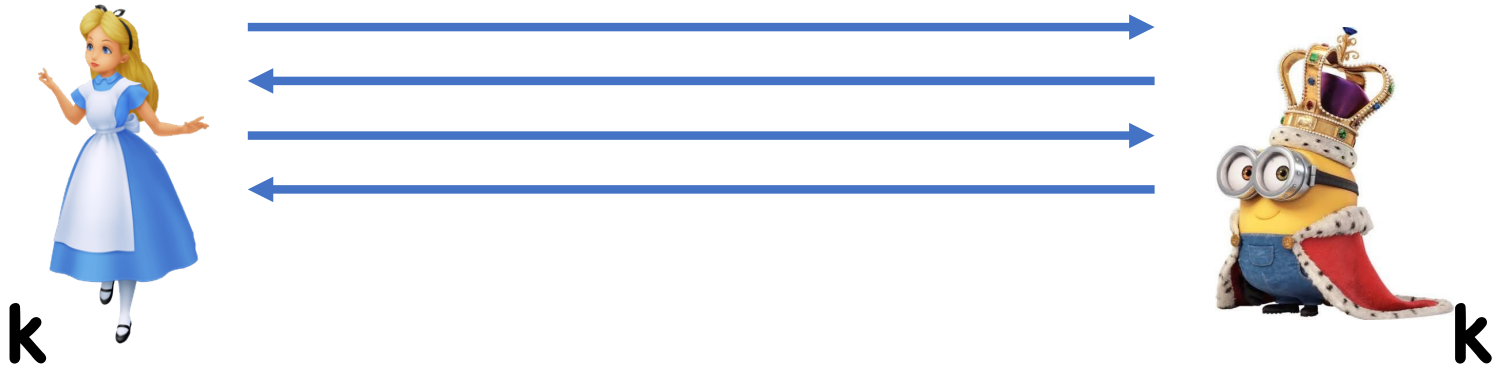
# Public Key Distribution



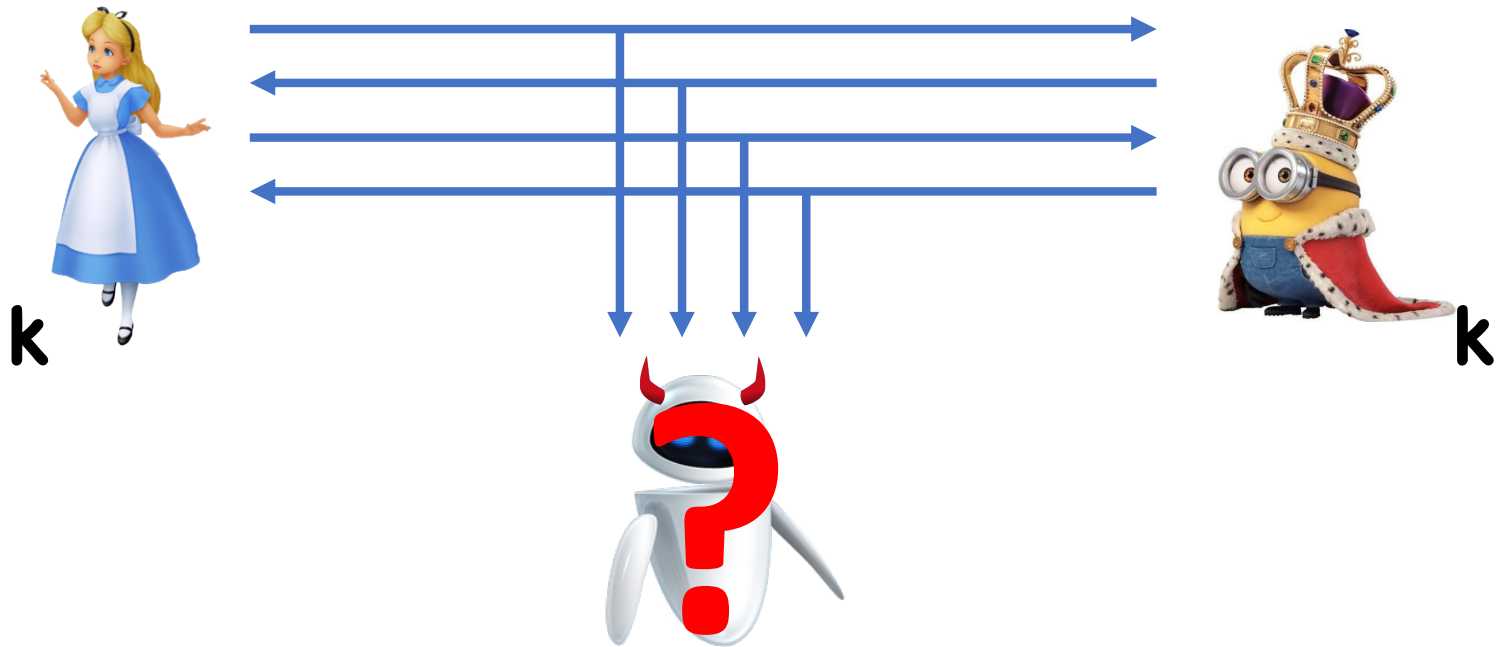
# Public Key Distribution



# Public Key Distribution



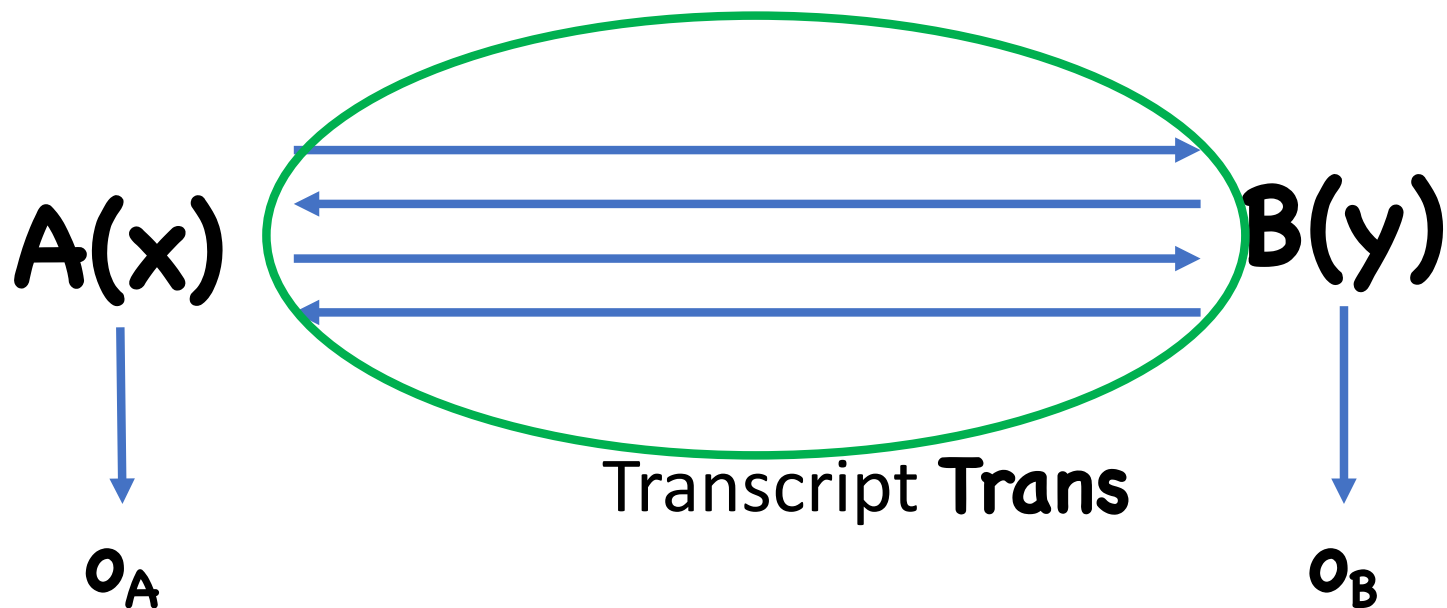
# Public Key Distribution





# Interactive Protocols

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



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

# 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:  **$(\text{Trans}, k)$**  is computationally indistinguishable from  **$(\text{Trans}, k')$**  where  **$k' \leftarrow K$**

# Matrix Multiplication Approach

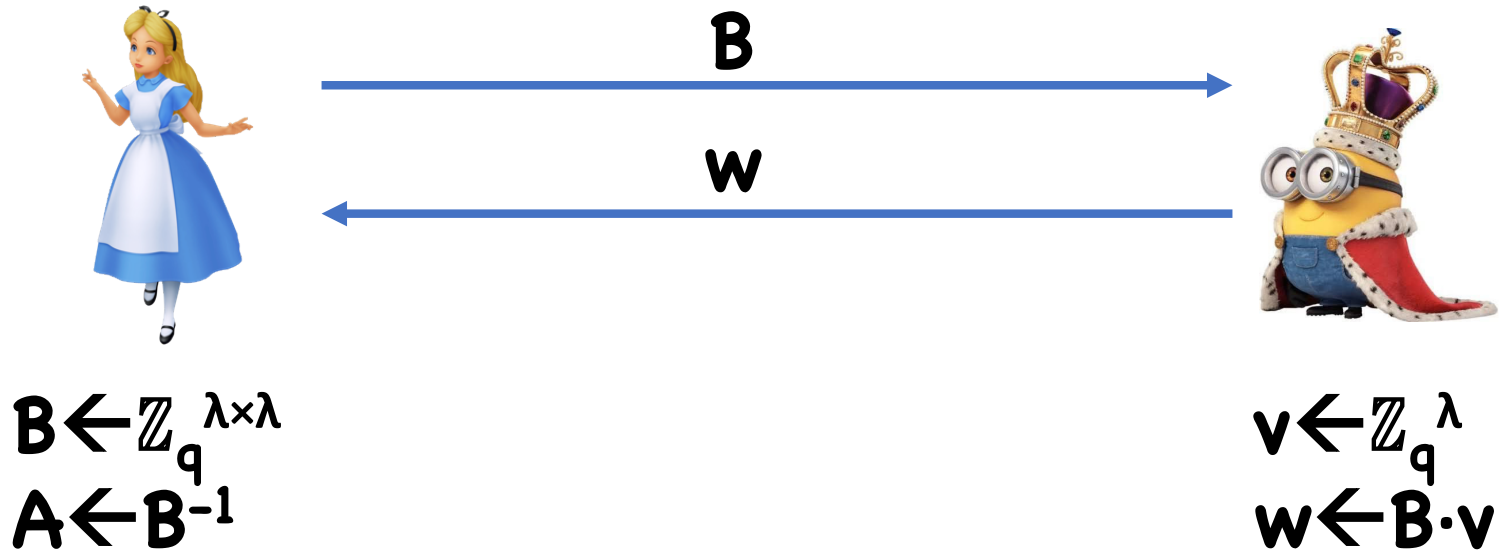


**B**

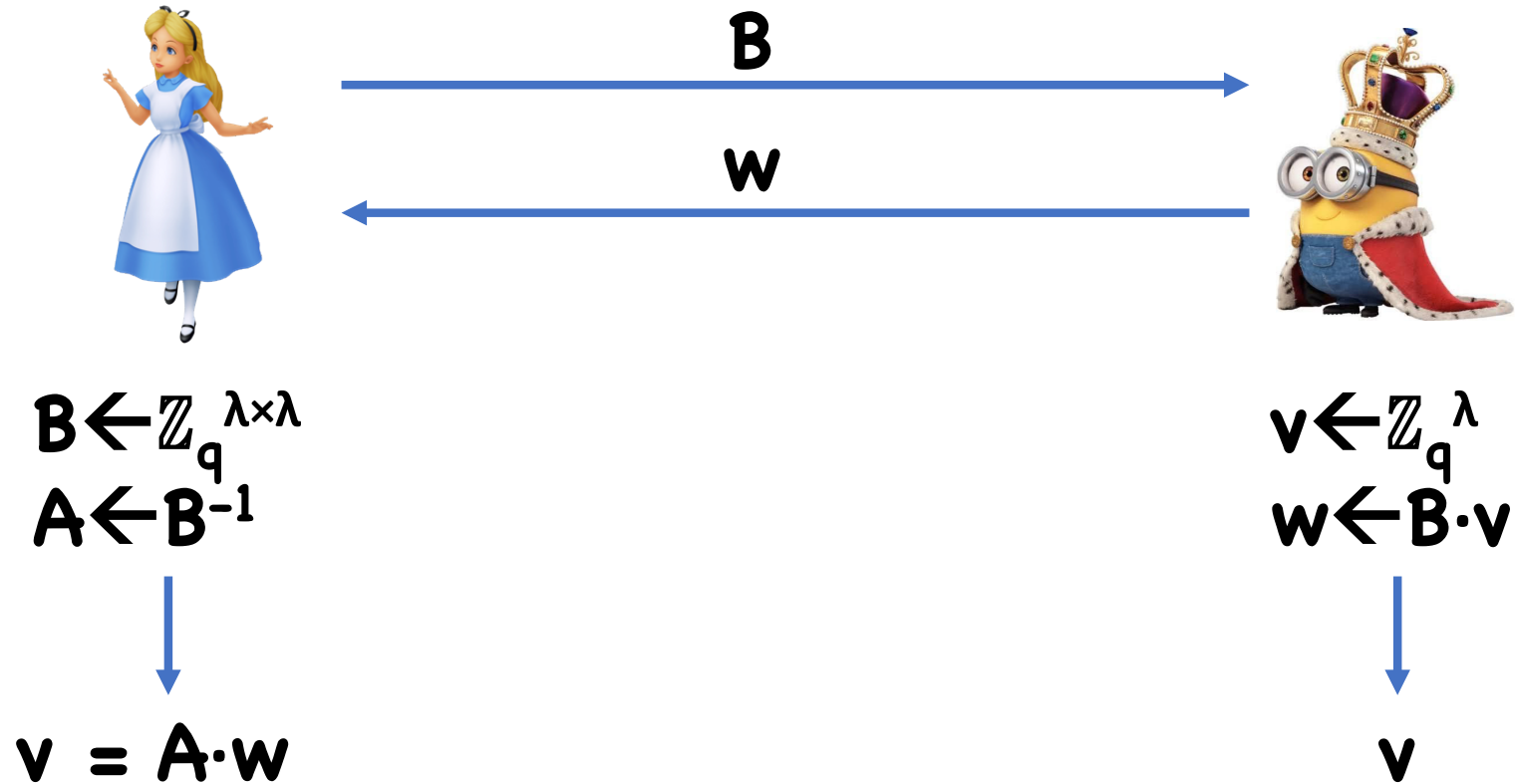


$$\mathbf{B} \leftarrow \mathbb{Z}_q^{\lambda \times \lambda}$$
$$\mathbf{A} \leftarrow \mathbf{B}^{-1}$$

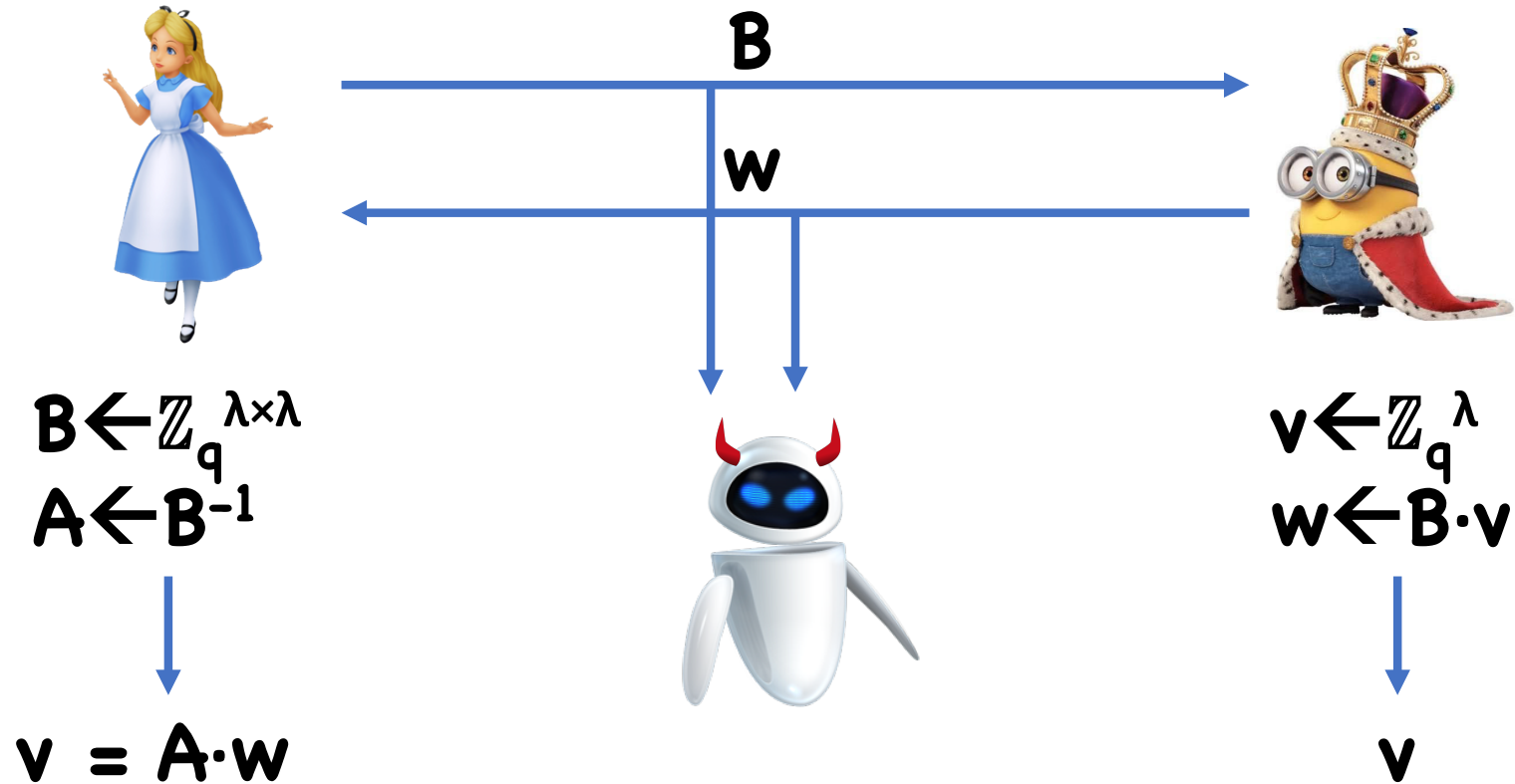
# Matrix Multiplication Approach



# Matrix Multiplication Approach



# Matrix Multiplication Approach



# Running Times?

Bob:  $O(\lambda^2)$

Eve:  $O(\lambda^3)$

# Running Times?

Bob:  $O(\lambda^2)$

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

Alice:  $O(\lambda^\omega)$

Different Approach:

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



# Running Times?

Bob:  $O(\lambda^2)$

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

Alice:  $O(\lambda^\omega)$

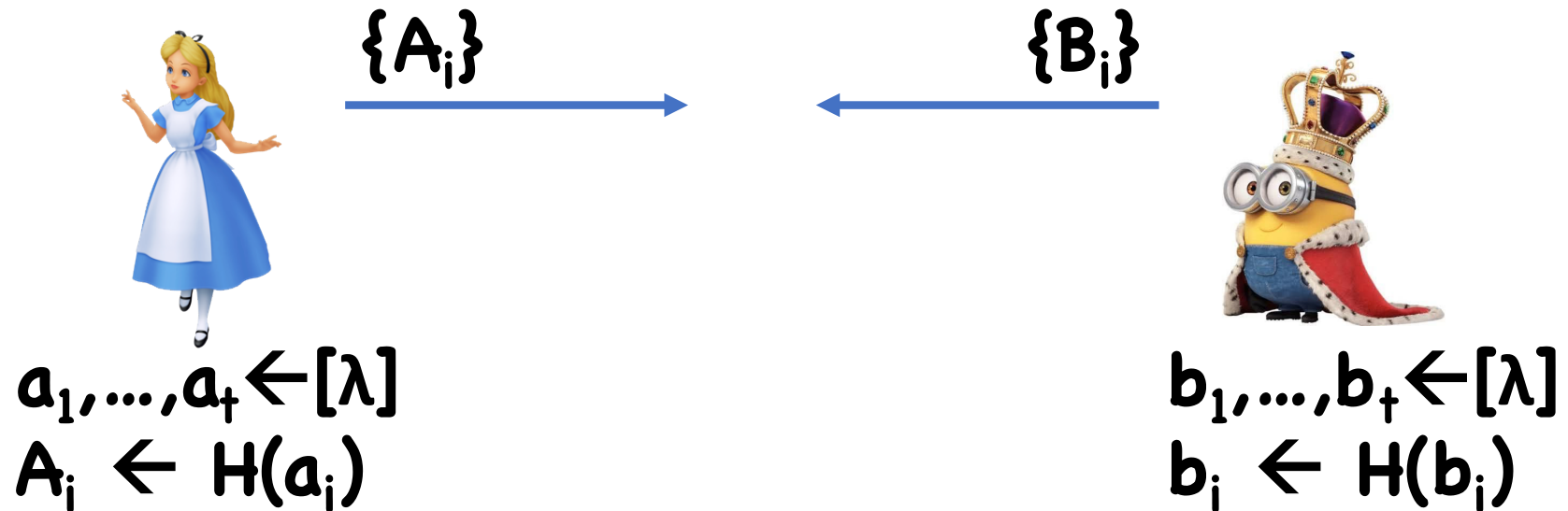
Assuming Matrix Multiplication exponent  $\omega > 2$ ,  
adversary must work harder than honest users

inverse to  $\mathbf{B}$

- Output  $(\mathbf{A}, \mathbf{B})$

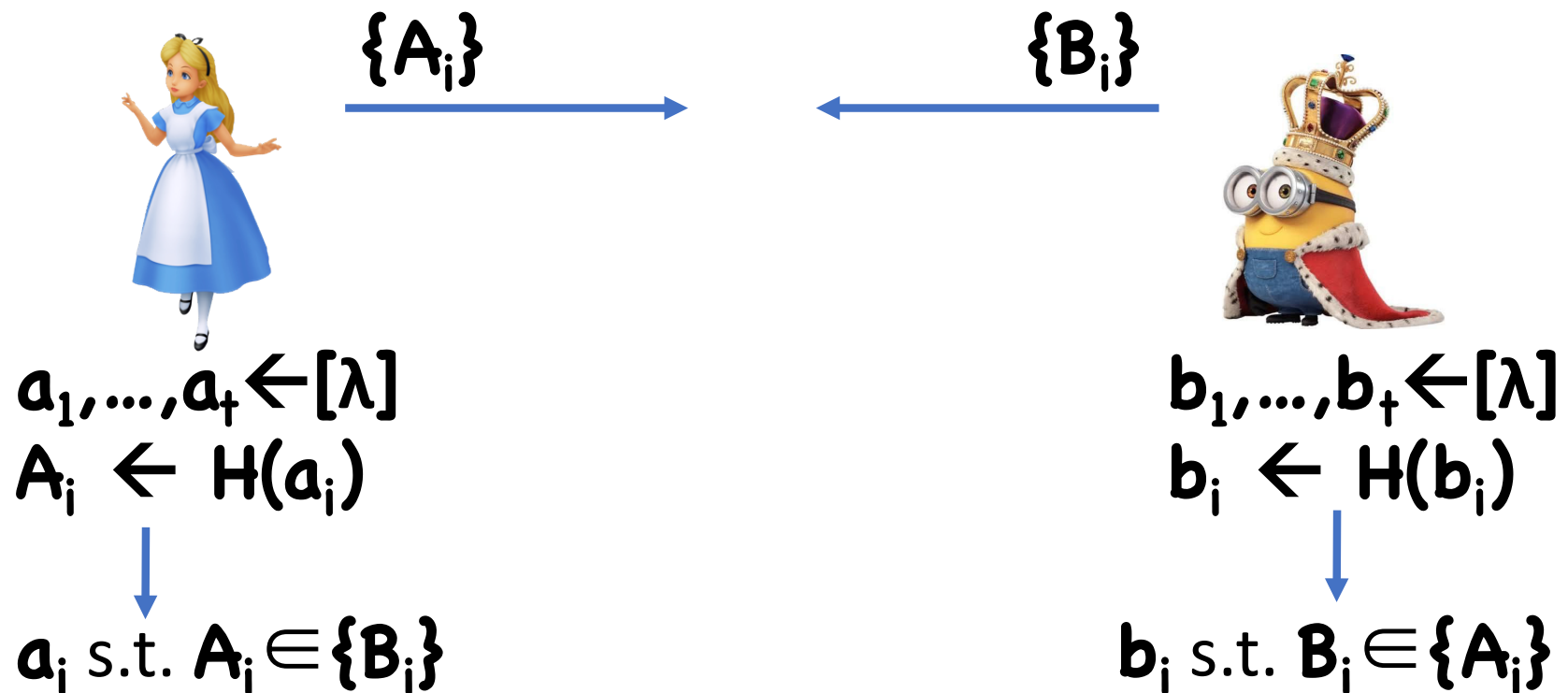
# Merkle Puzzles

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



# Merkle Puzzles

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



# Analysis

Protocol succeeds iff:

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

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

Treating  $H$  as ideal hash function (random oracle),  
how many queries does adversary need?

# 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  $\lambda$  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$_}keys%p}p;p;p;p;p;p;map{$p{$_}=~/^[P.]/&&
close$_}%p;wait until$?;map{/^r/&&<$_>}%p;$_=$d[$q];sleep rand(2)if/\S/;print
```

# Key Distribution from Obfuscation

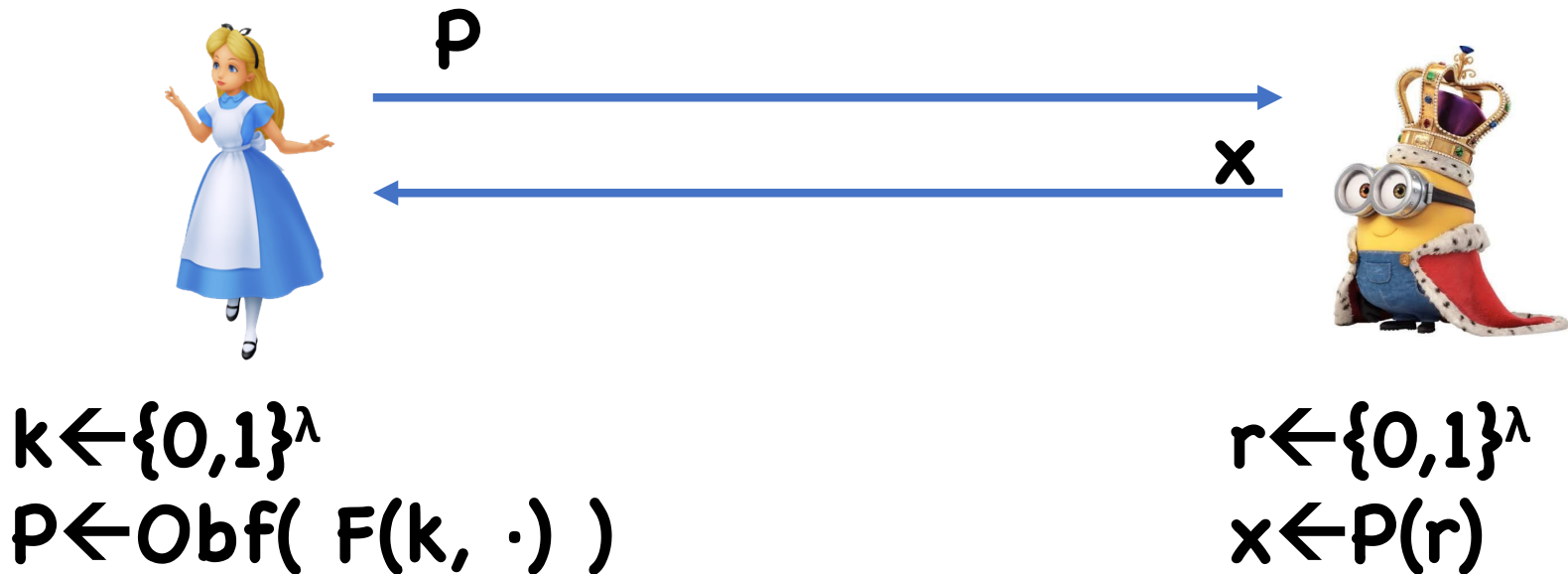
Let  $F, F^{-1}$  be a block cipher


$$k \leftarrow \{0,1\}^\lambda$$
$$P \leftarrow \text{Obf}( F(k, \cdot) )$$



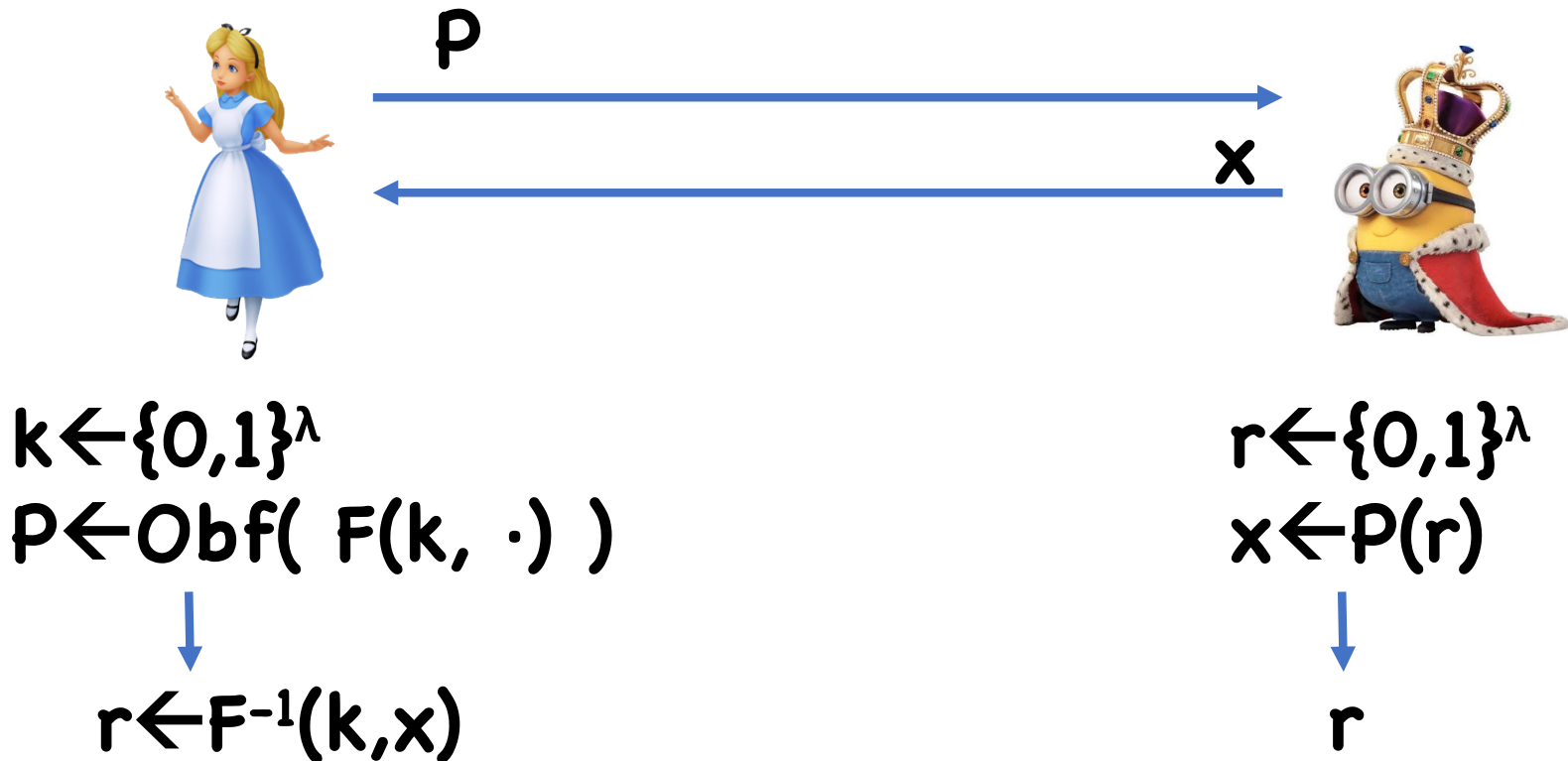
# Key Distribution from Obfuscation

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



# Key Distribution from Obfuscation

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



# 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  $X$

**Gen()**: outputs  $(pk, sk)$

$F(pk, x \in X) = y \in X$

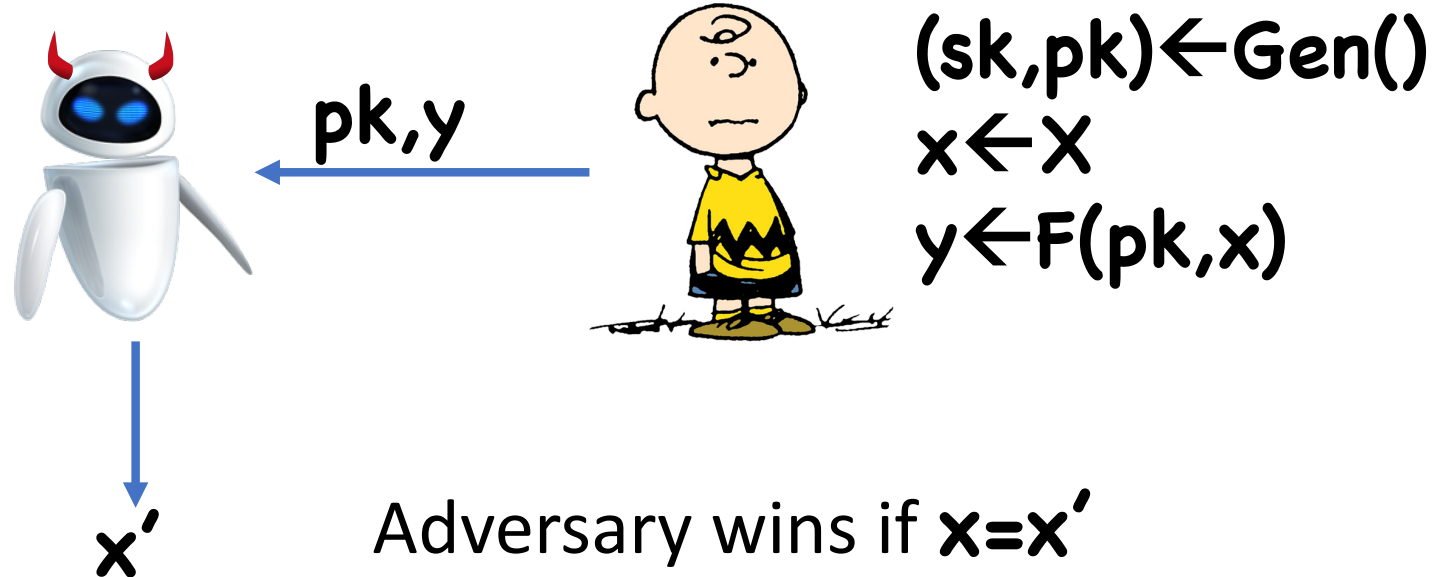
$F^{-1}(sk, y) = x$

Correctness:

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

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

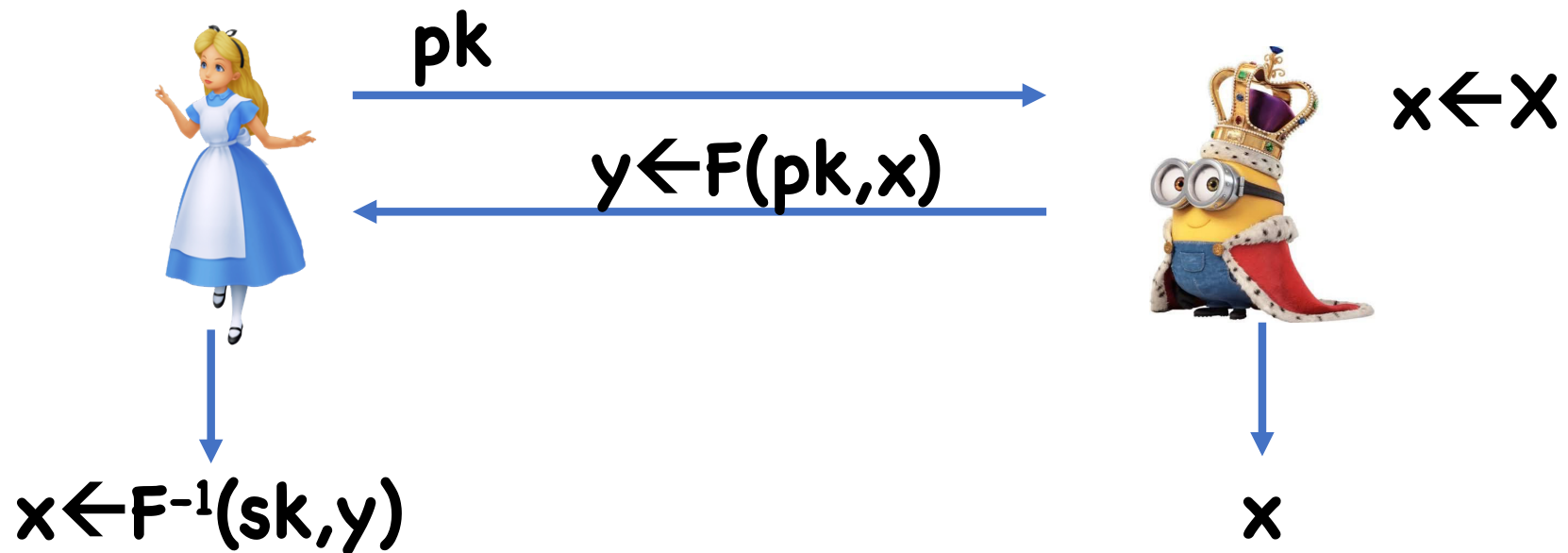
# Trapdoor Permutation Security



In other words,  $F(pk, \cdot)$  is a one-way function

# Key Distribution from TDPs

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





# Analysis

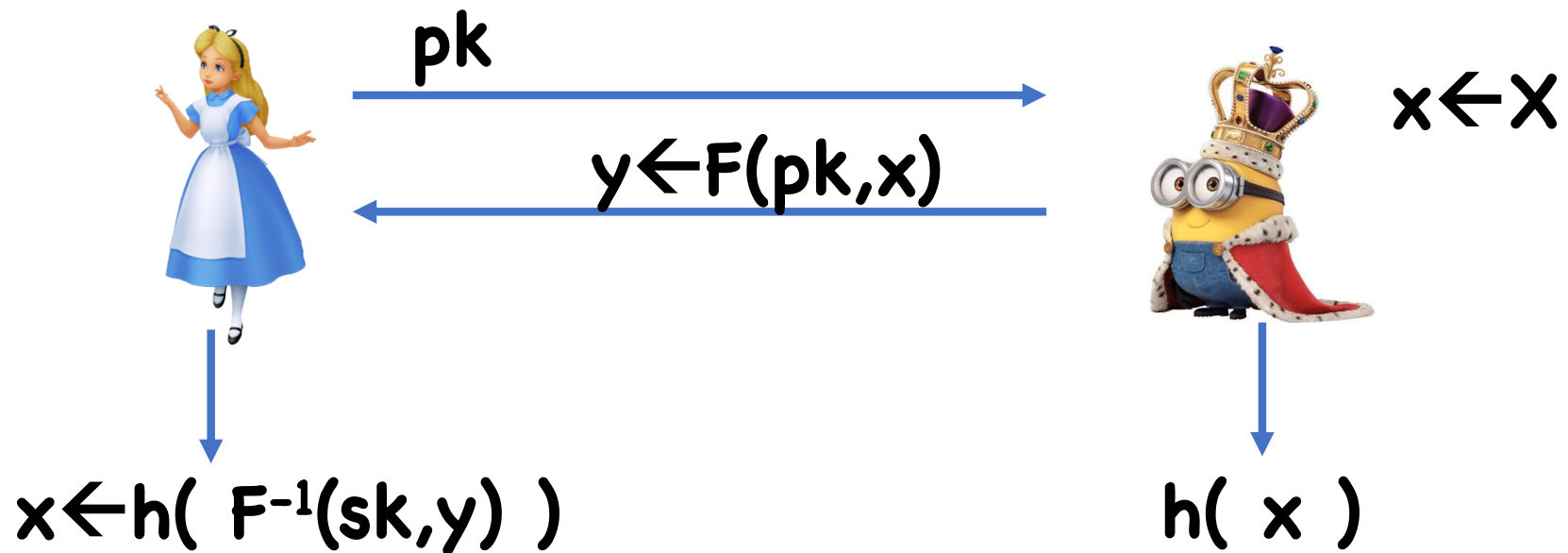
Correctness follows from correctness of TDP

Security:

- By TDP security, adversary cannot compute  $\mathbf{x}$
- However,  $\mathbf{x}$  is distinguishable from a random key

# Key Distribution from TDPs

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



$h$  a hardcore bit for  $F(pk, \cdot)$

**Theorem:** If  $h$  is  $(t, \epsilon)$ -secure hardcore bit for  $F(pk, \cdot)$ , then protocol is  $(t, \epsilon)$ -secure

Proof:

- $(Trans, k) = ((pk, y), h(x))$
- Hardcore bit means indistinguishable 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<sup>e</sup> mod N**

**F<sup>-1</sup>(sk,c):** Output **x = y<sup>d</sup> mod 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, none known

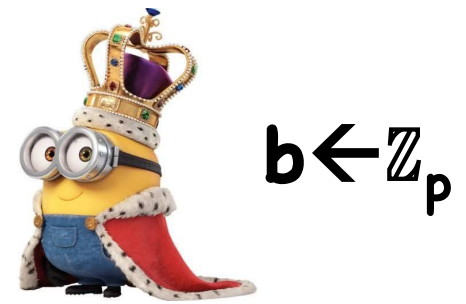
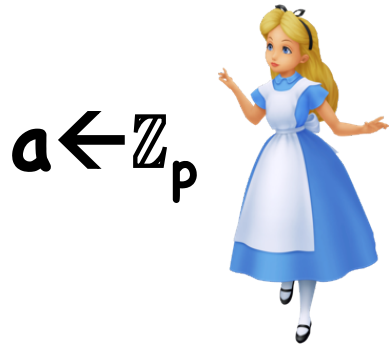
- Still interesting object:
  - Useful abstraction in protocol design
  - Maybe more will be discovered...

Using obfuscation:

- Let  $\mathcal{P}$  be a PRP
- $sk = k, pk = \text{Obf}( \mathcal{P}(k, \cdot) )$

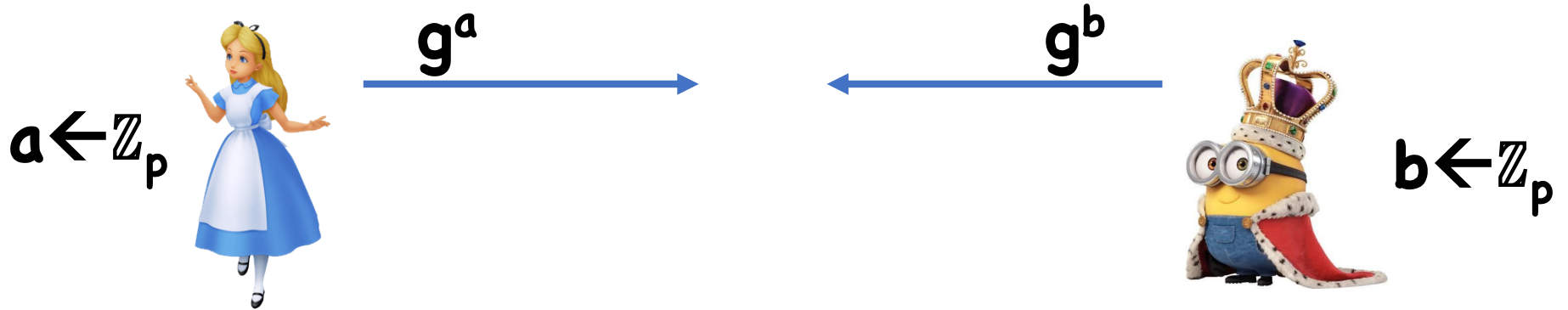
# Key Distribution from DH

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



# Key Distribution from DH

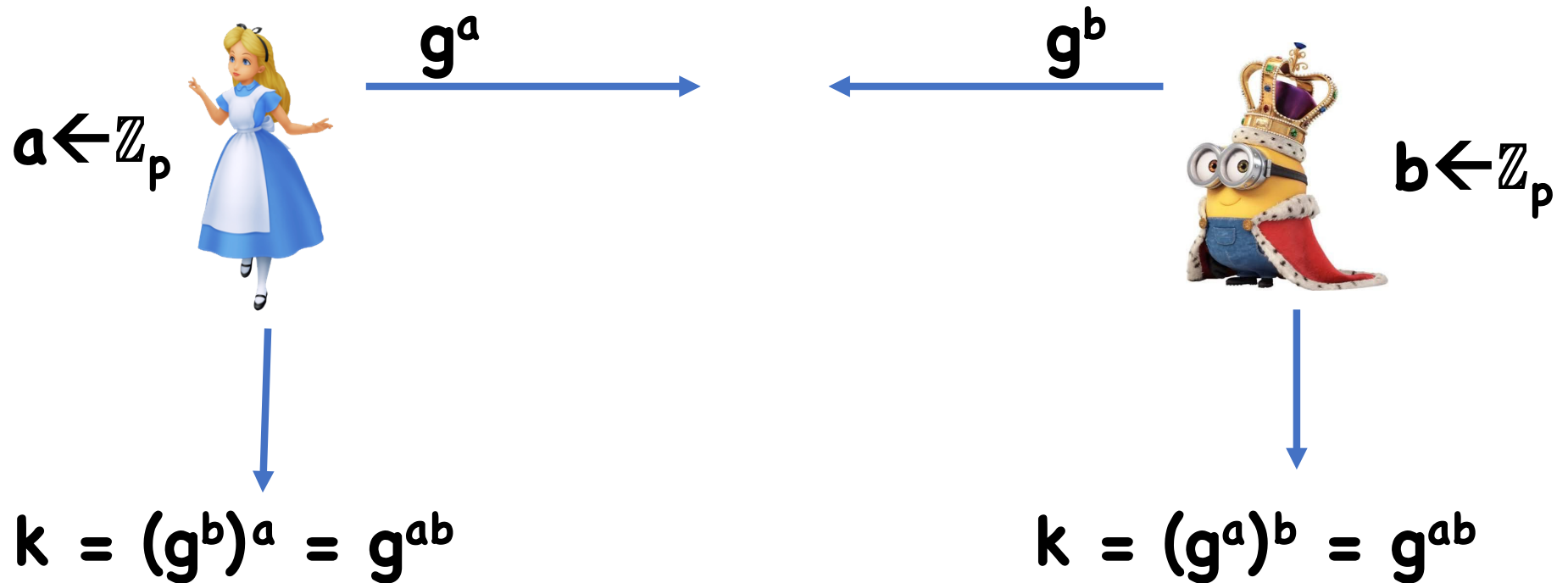
Everyone agrees on group **G** or prime order **p**





# Key Distribution from DH

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



# Key Distribution from DH

**Theorem:** If  $(t, \epsilon)$ -DDH holds on  $\mathbf{G}$ , then the Diffie-Hellman protocol is  $(t, \epsilon)$ -secure

Proof:

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

What if only CDH holds, but DDH is easy?

# Known Constructions of Public Key Distribution

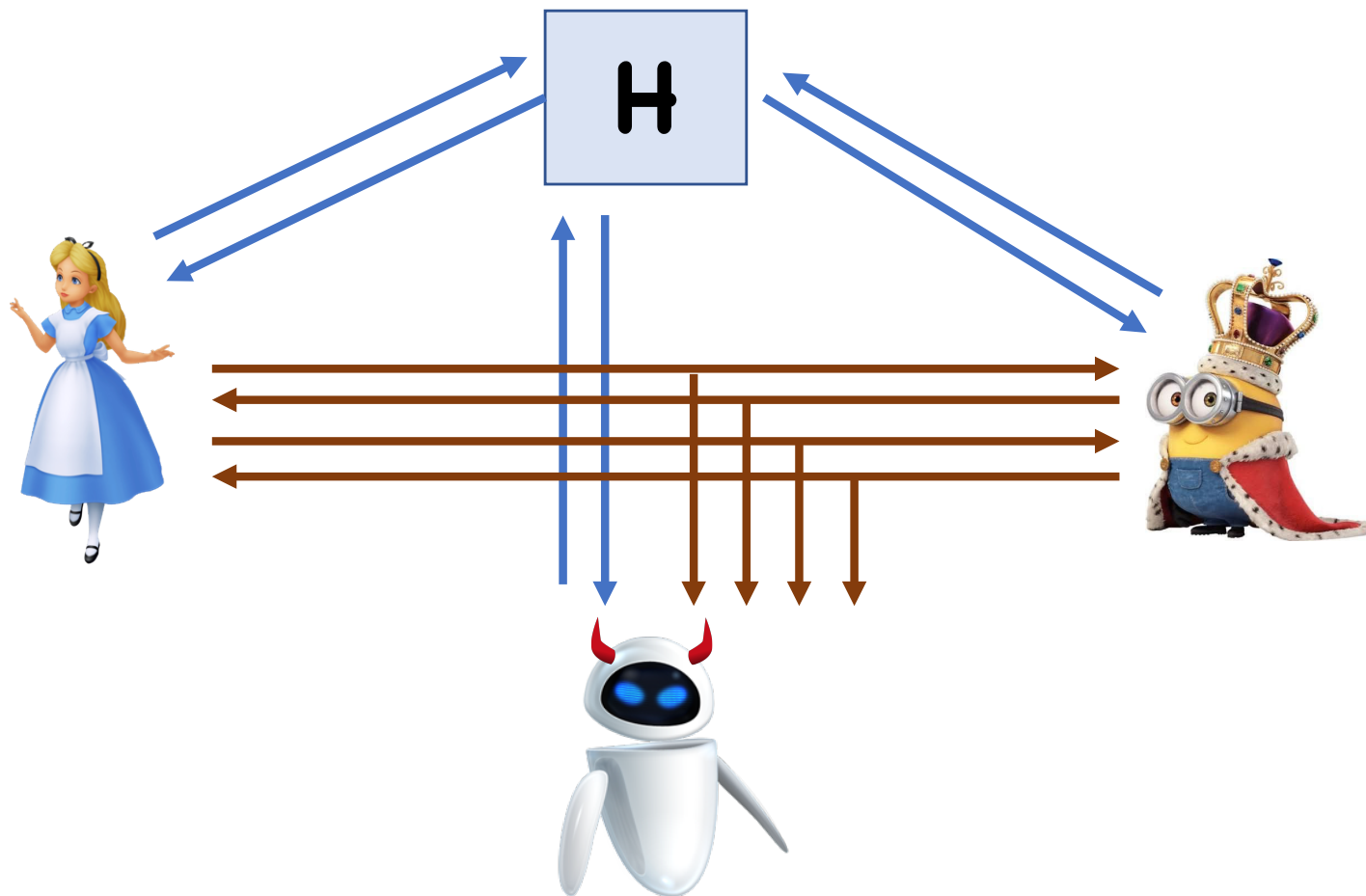
All based on specific number theoretic problems

- RSA, Factoring
- Discrete log, Diffie-Hellman
- ...

No known ways to base (solely) on block ciphers, PRFs, etc.

Is this inherent?

# Black Box Separation



**Theorem:** If  $H$  is a random oracle, then for any key agreement protocol in which Alice and Bob make at most  $n$  queries, there is an (inefficient) adversary that makes at most  $O(n^2)$  queries

Therefore, true public key distribution likely hard to build from one-way functions

If allowing for polynomial hardness gap, then Merkle is likely optimal from one-way functions

# History

1974: Merkle invents his puzzles while an undergrad

1976: Diffie and Hellman publish their scheme

- First public mention of public key crypto

1977: RSA publish their scheme

1997: Revealed that public key crypto was developed at GCHQ even earlier

- James H. Ellis: idea for public key crypto
- Clifford Cocks: develops RSA
- Malcolm Williamson: develops Diffie-Hellman

2002: RSA win Turing Award

2015: Diffie-Hellman win Turing Award

# Next Time

## Public key encryption

- Removes need to key exchange in the first place

# Public Key Encryption

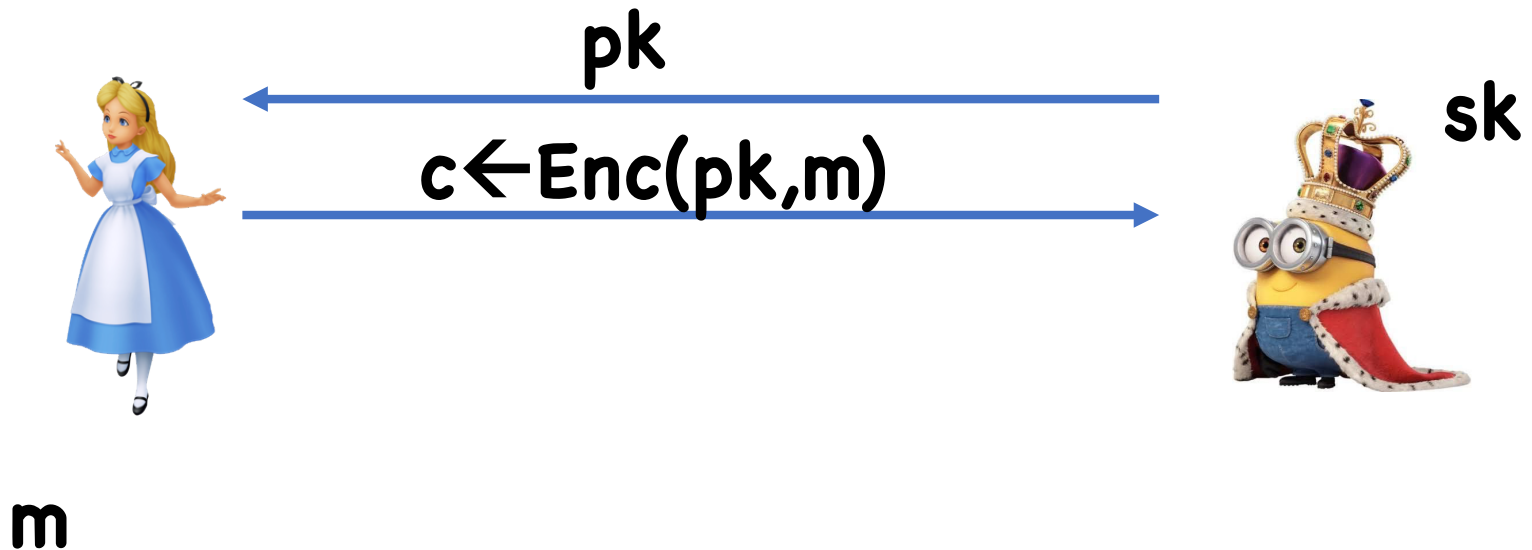




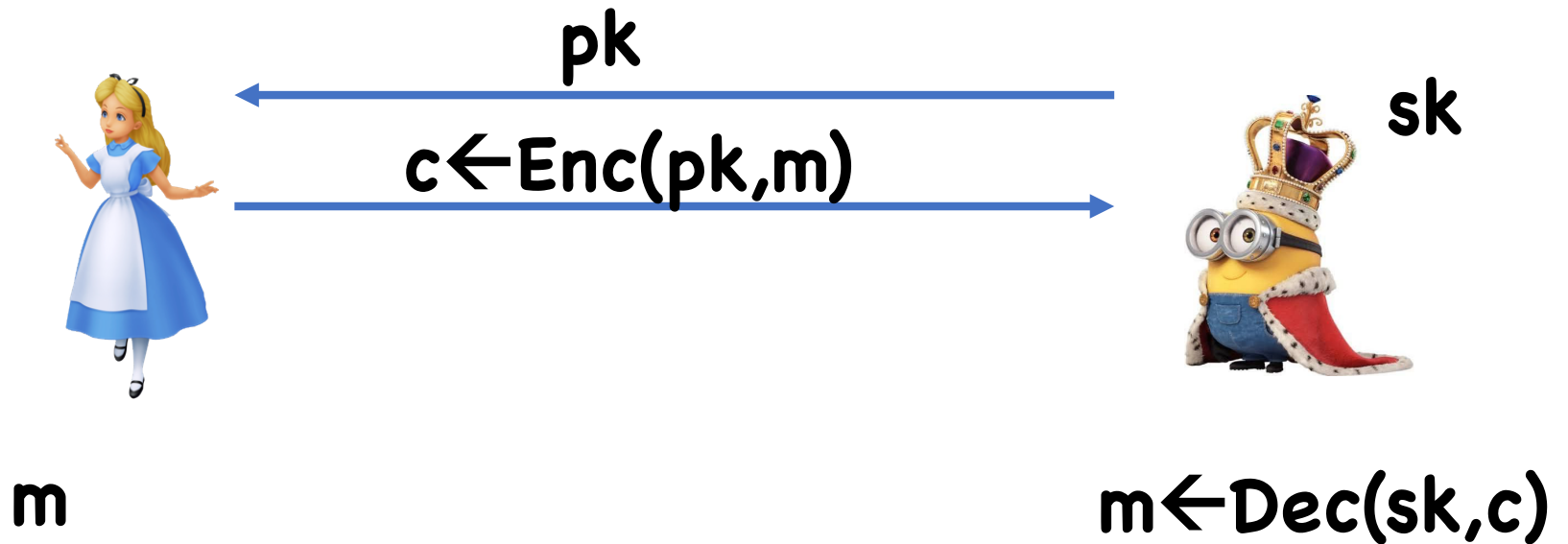
# Public Key Encryption



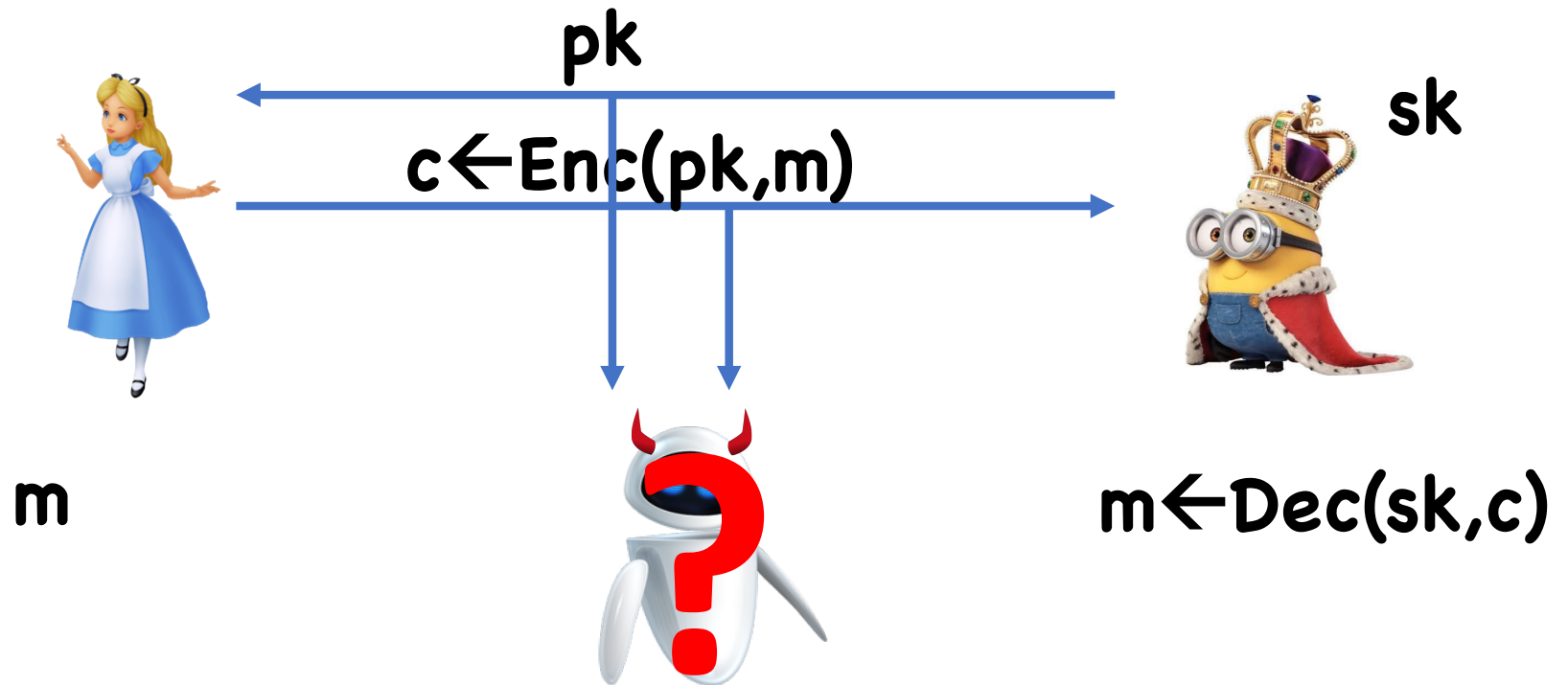
# Public Key Encryption



# Public Key Encryption



# Public Key Encryption



# PKE vs Key Agreement

Key agreement:



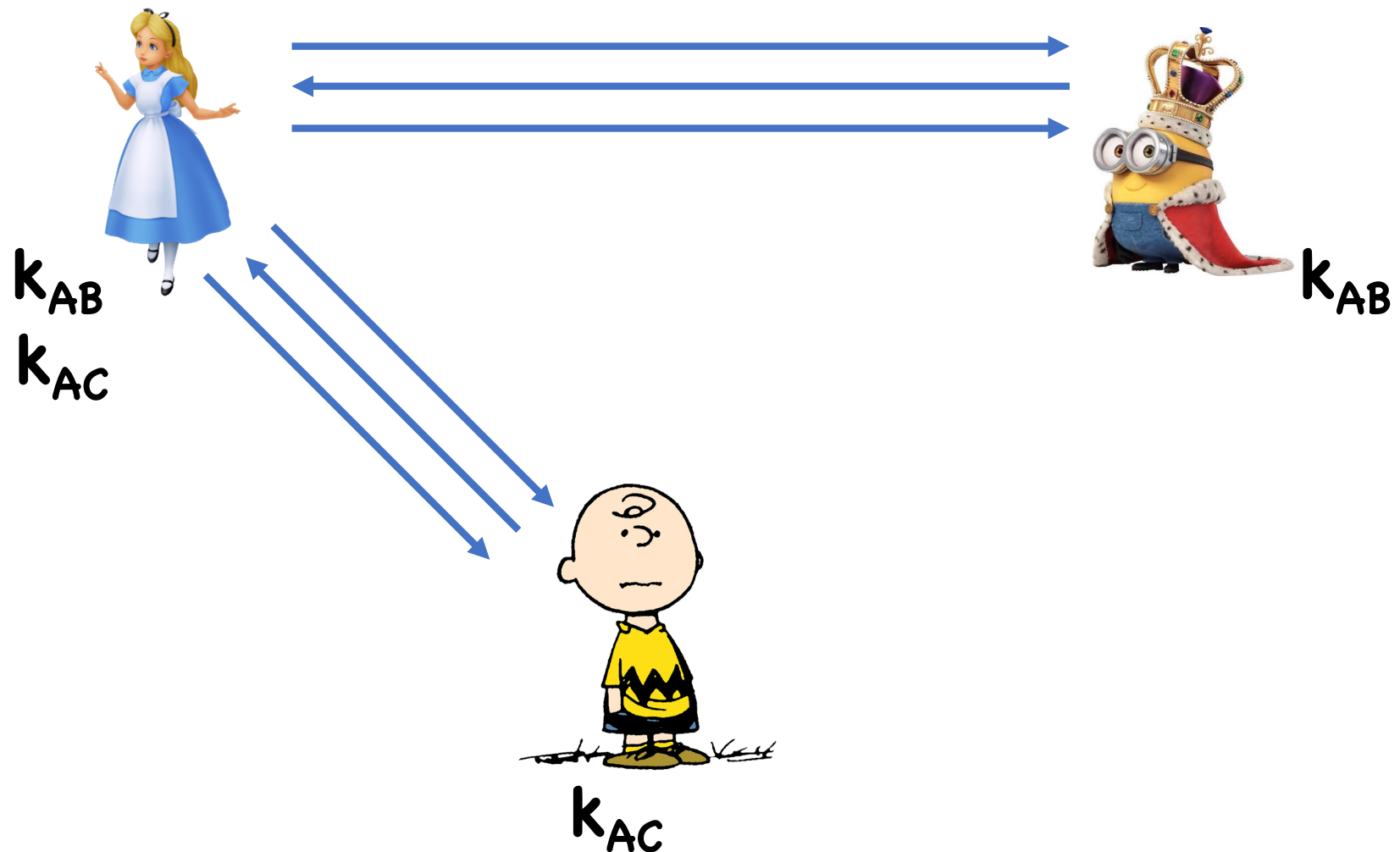
# PKE vs Key Agreement

Key agreement:



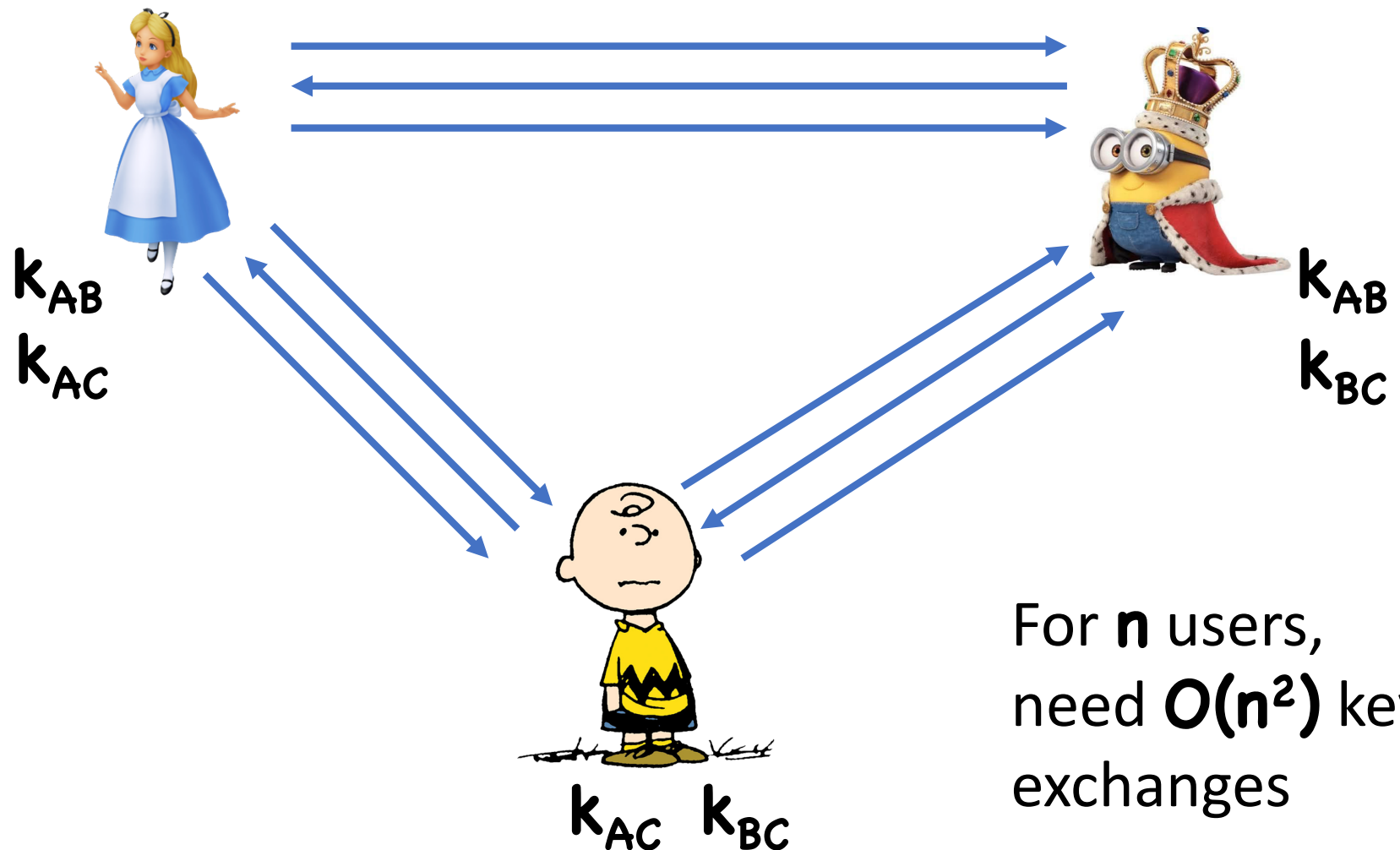
# PKE vs Key Agreement

Key agreement:



# PKE vs Key Agreement

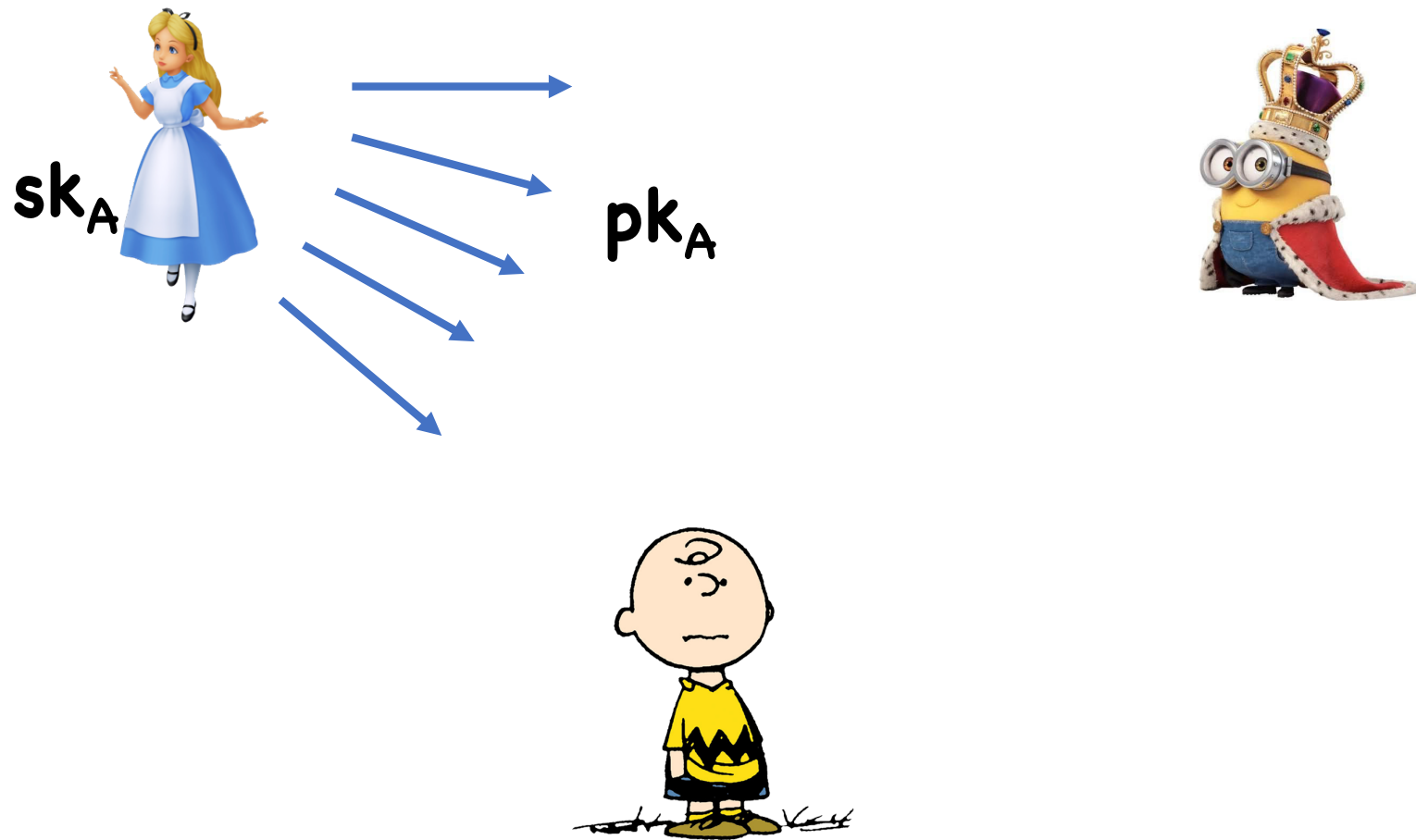
Key agreement:





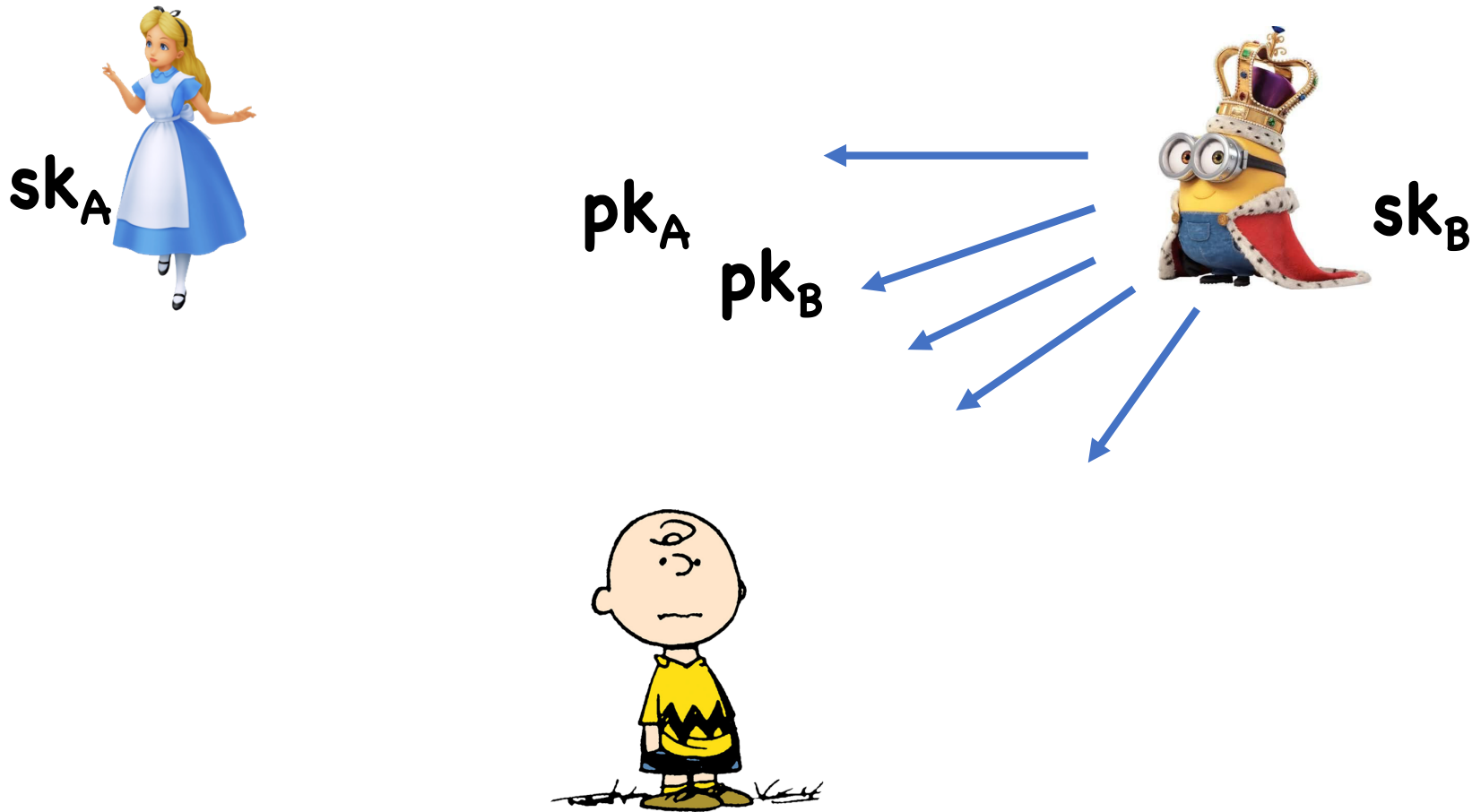
# PKE vs Key Agreement

PKE:



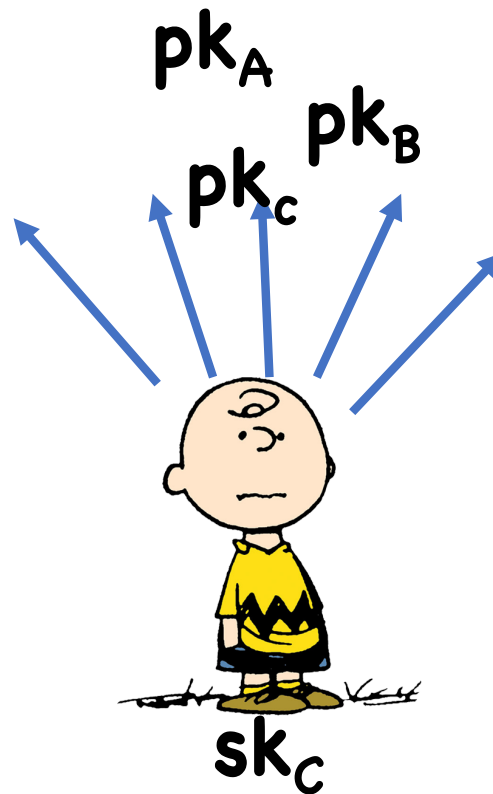
# PKE vs Key Agreement

PKE:



# PKE vs Key Agreement

PKE:



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

# PKE Syntax

Message space  **$\mathcal{M}$**

Algorithms:

- **$(sk, pk) \leftarrow \text{Gen}(\lambda)$**
- **$\text{Enc}(pk, m)$**
- **$\text{Dec}(sk, m)$**

Correctness:

$$\Pr[\text{Dec}(sk, \text{Enc}(pk, m)) = m : (sk, pk) \leftarrow \text{Gen}(\lambda)] = 1$$

# Security

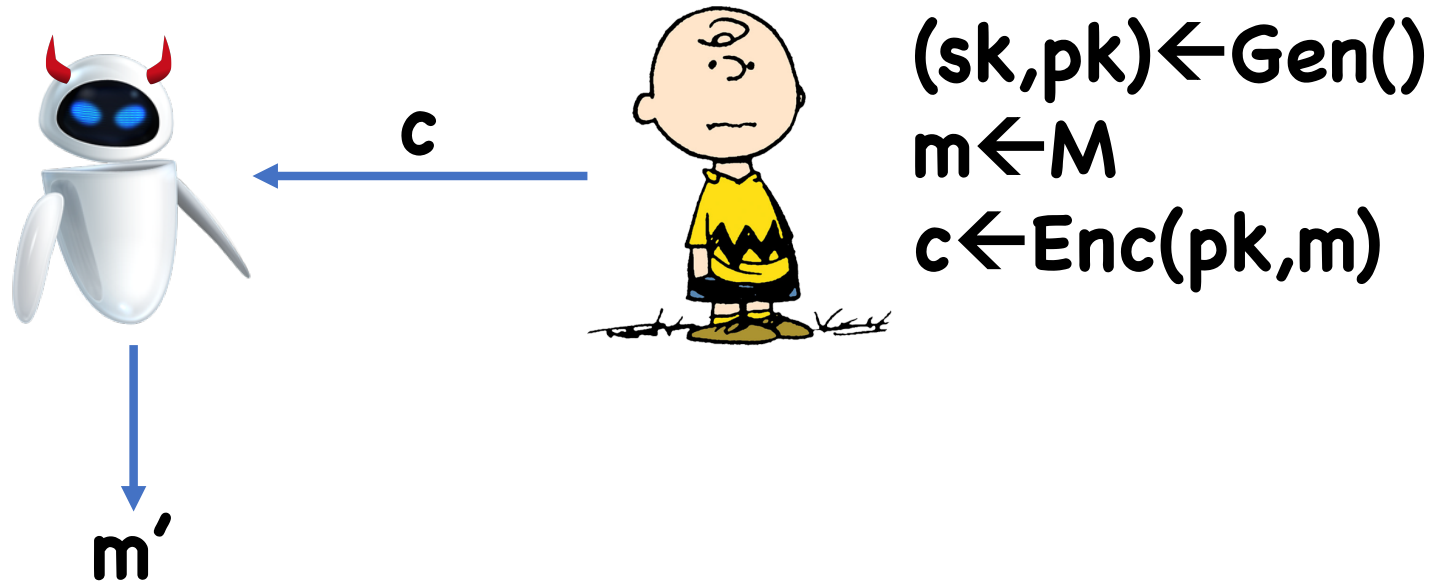
One-way security

Semantic Security

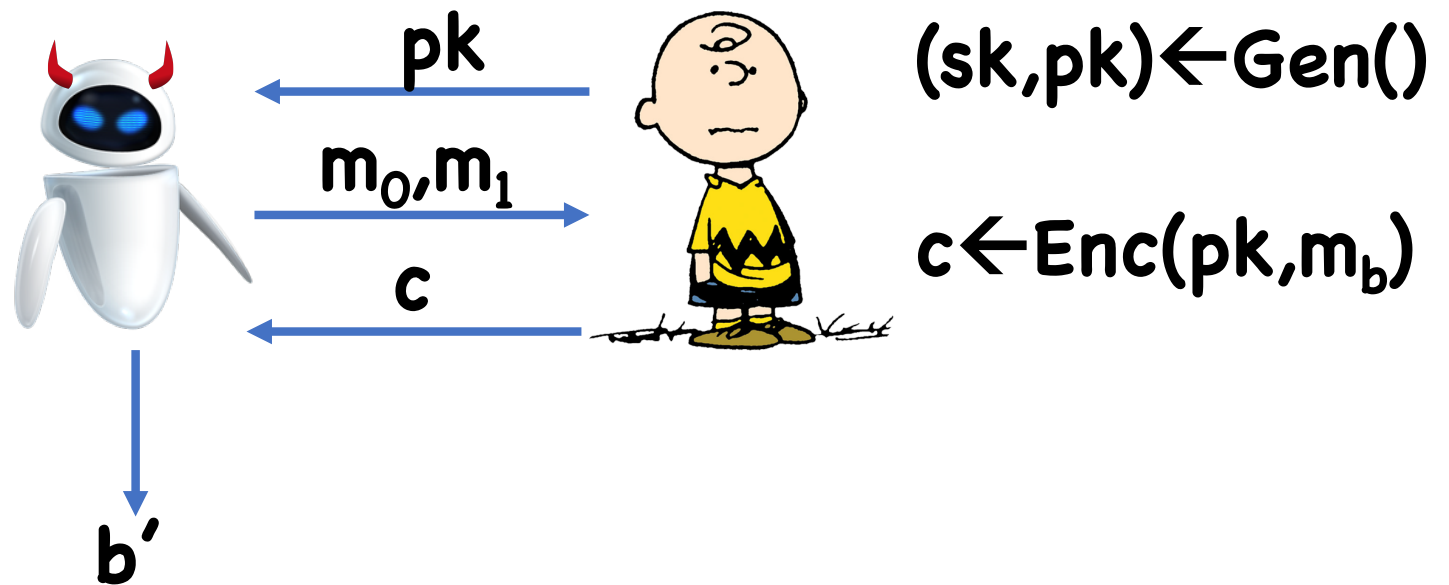
CPA security

CCA Security

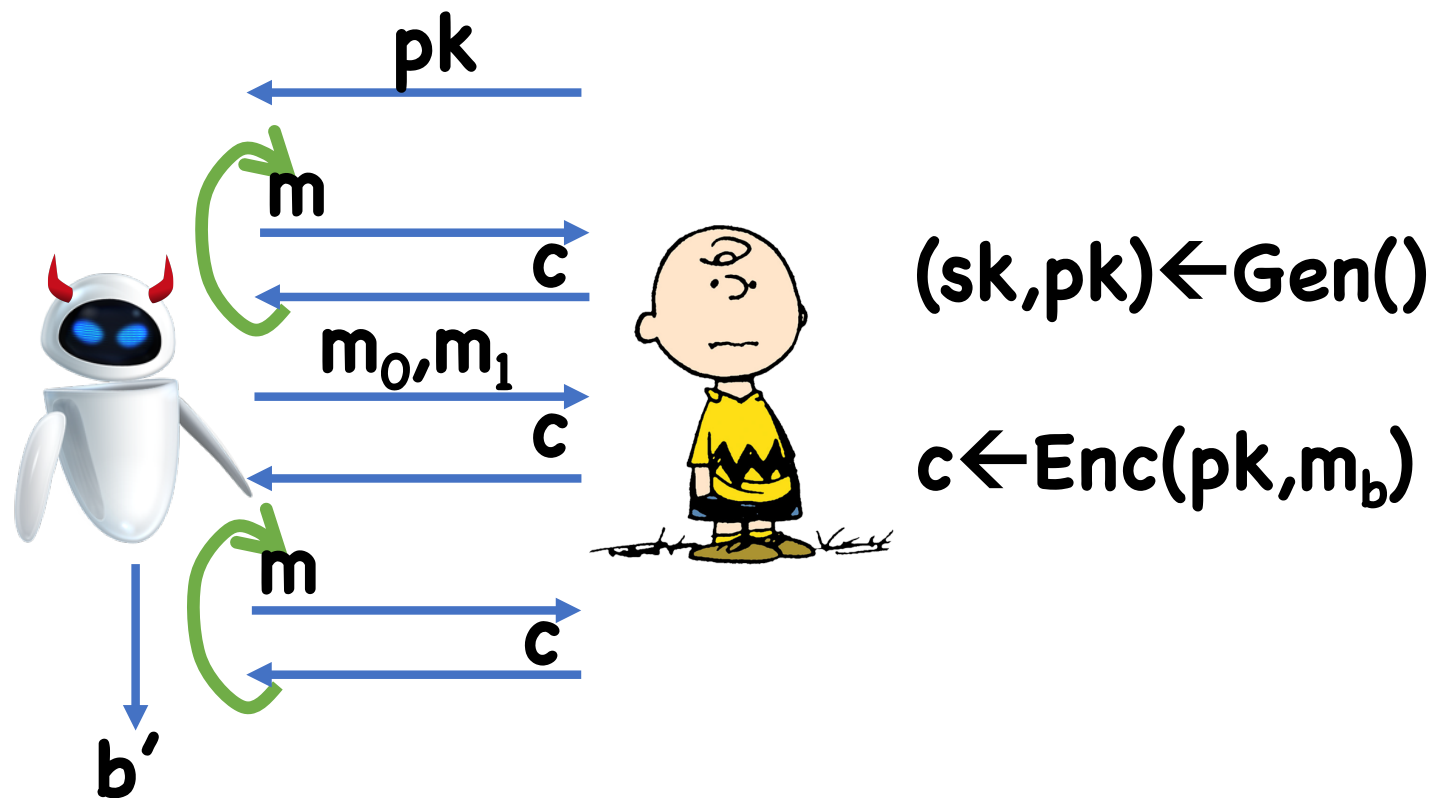
# One-way Security



# Semantic Security



# CPA Security





# CCA Security

