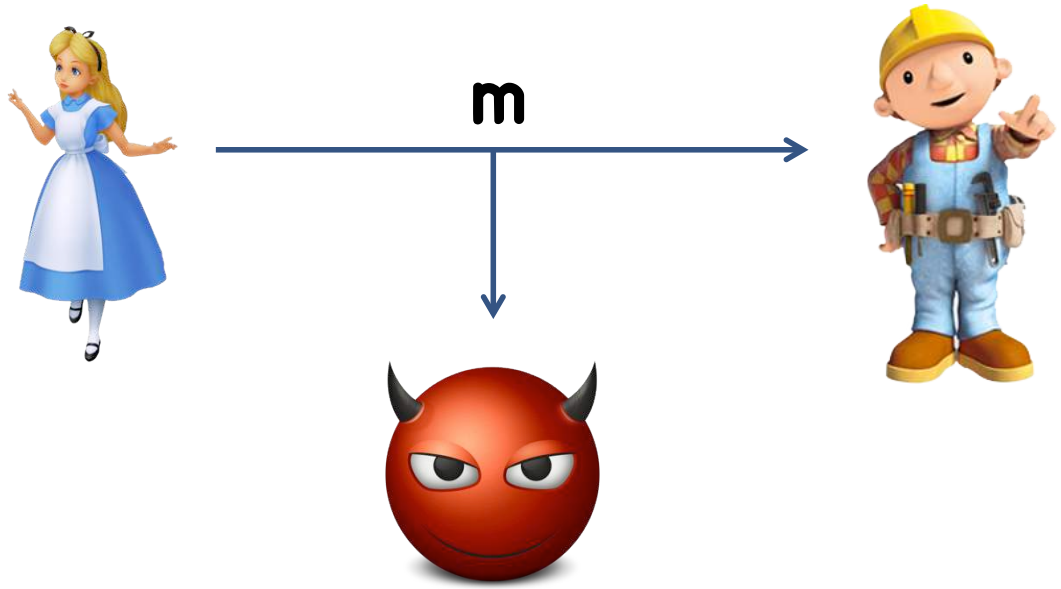
THE SURPRISING POWER OF MODERN CRYPTOGRAPHY

Mark Zhandry – Stanford University

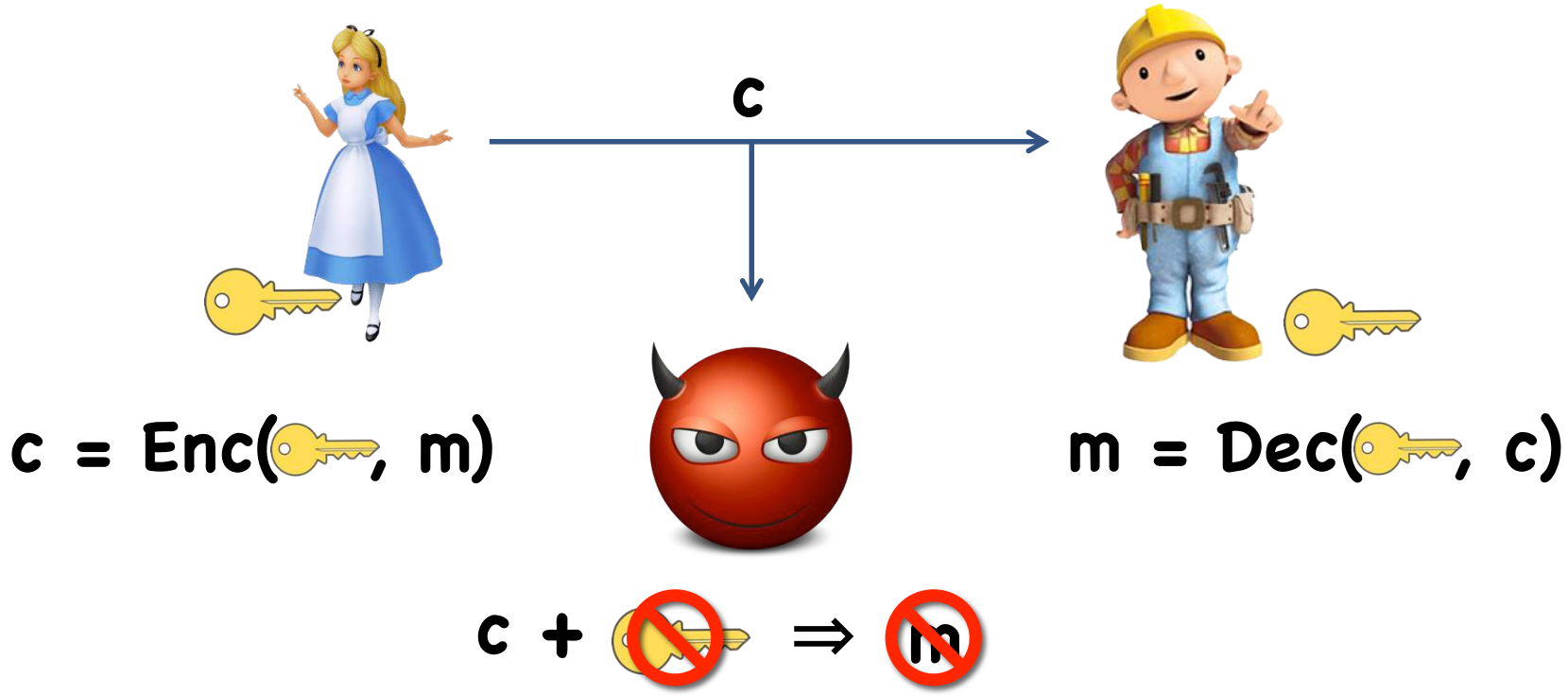
Typical Crypto Application



Goals

Privacy Integrity Authenticity

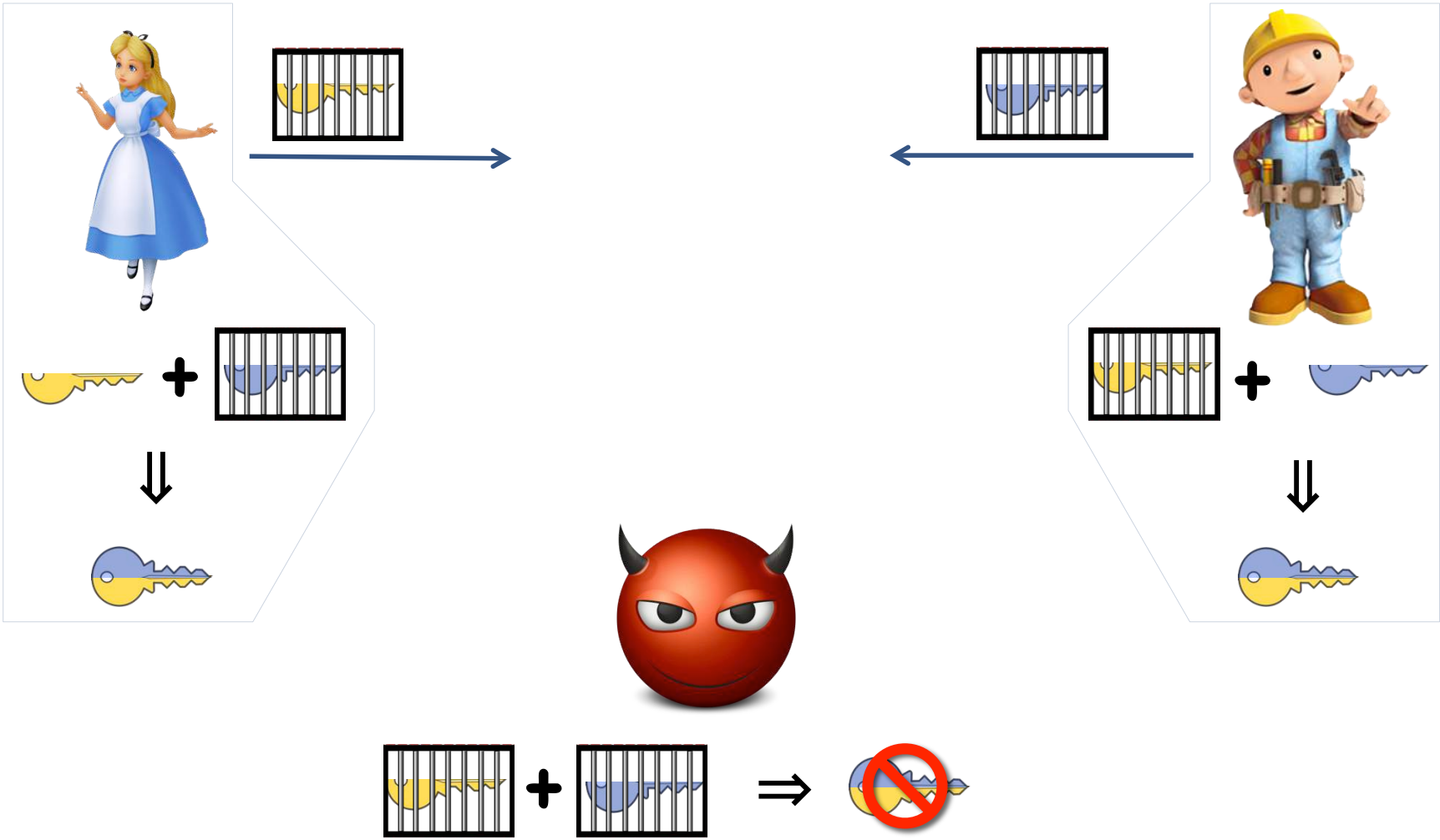
Solution (Pre 1970's)



Major question: how to obtain secret keys?

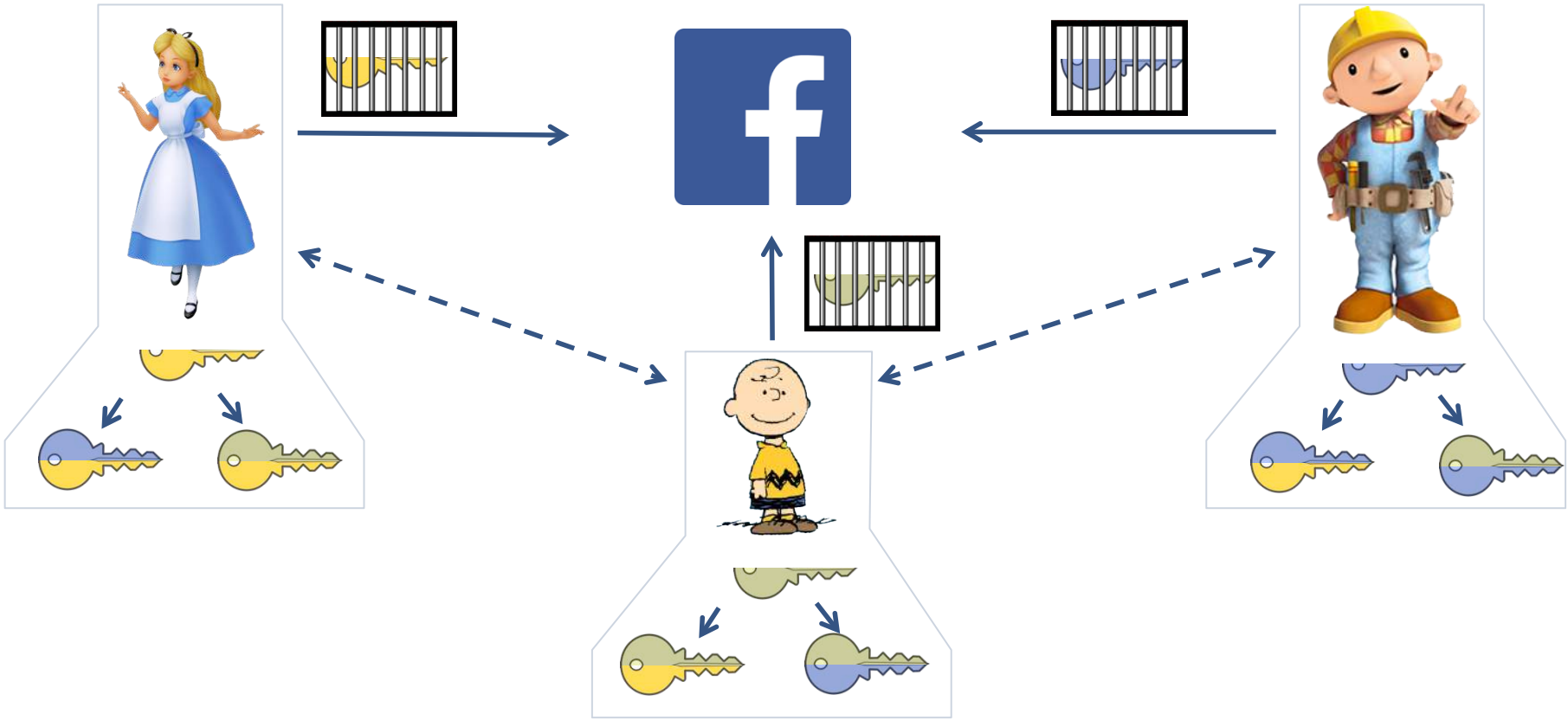
Diffie-Hellman Key Exchange [DH'76]

a.k.a. Non-Interactive Key Exchange (NIKE)



Benefit of *Non-Interactive* Key Exchange

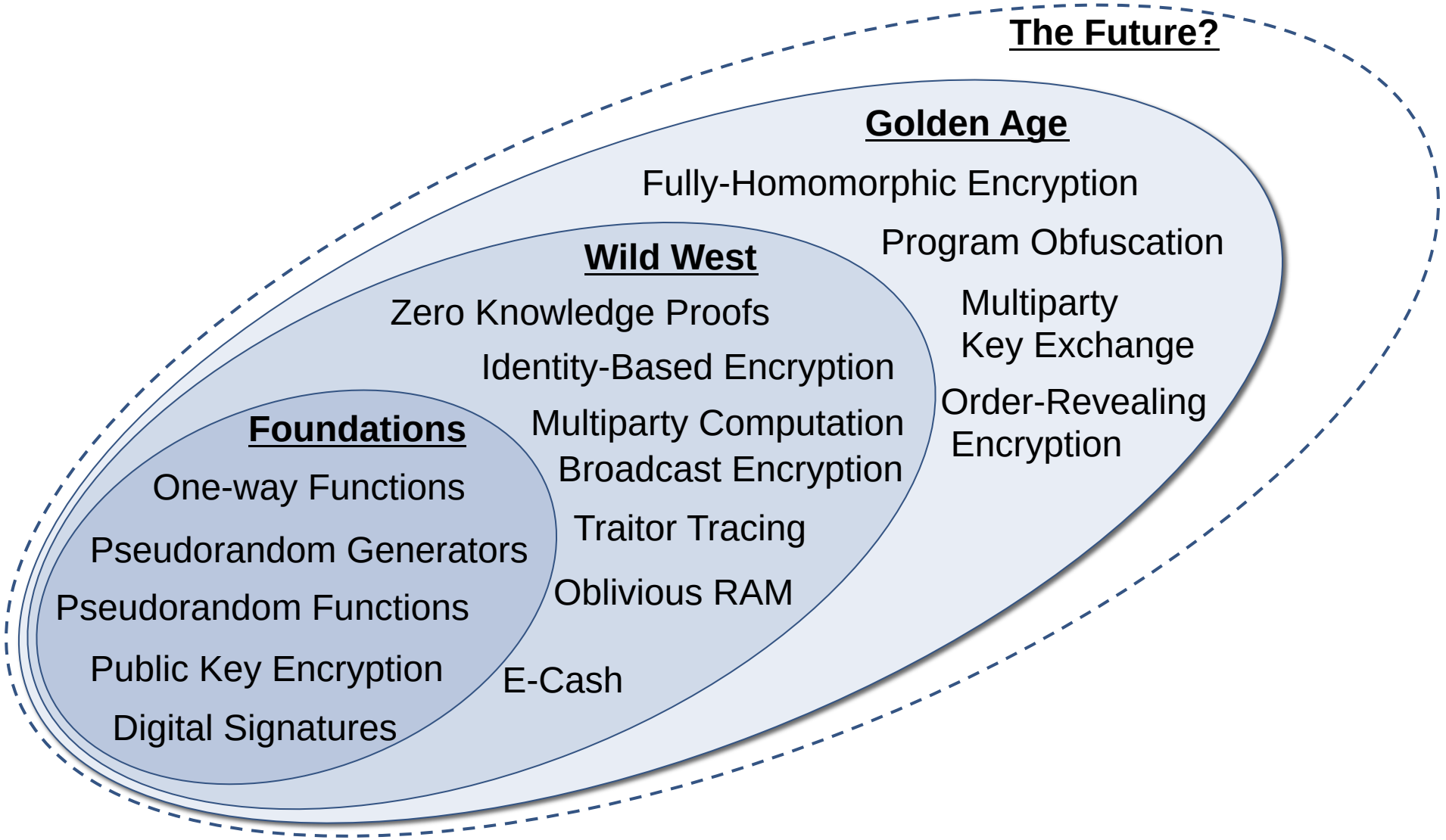
Re-usable:



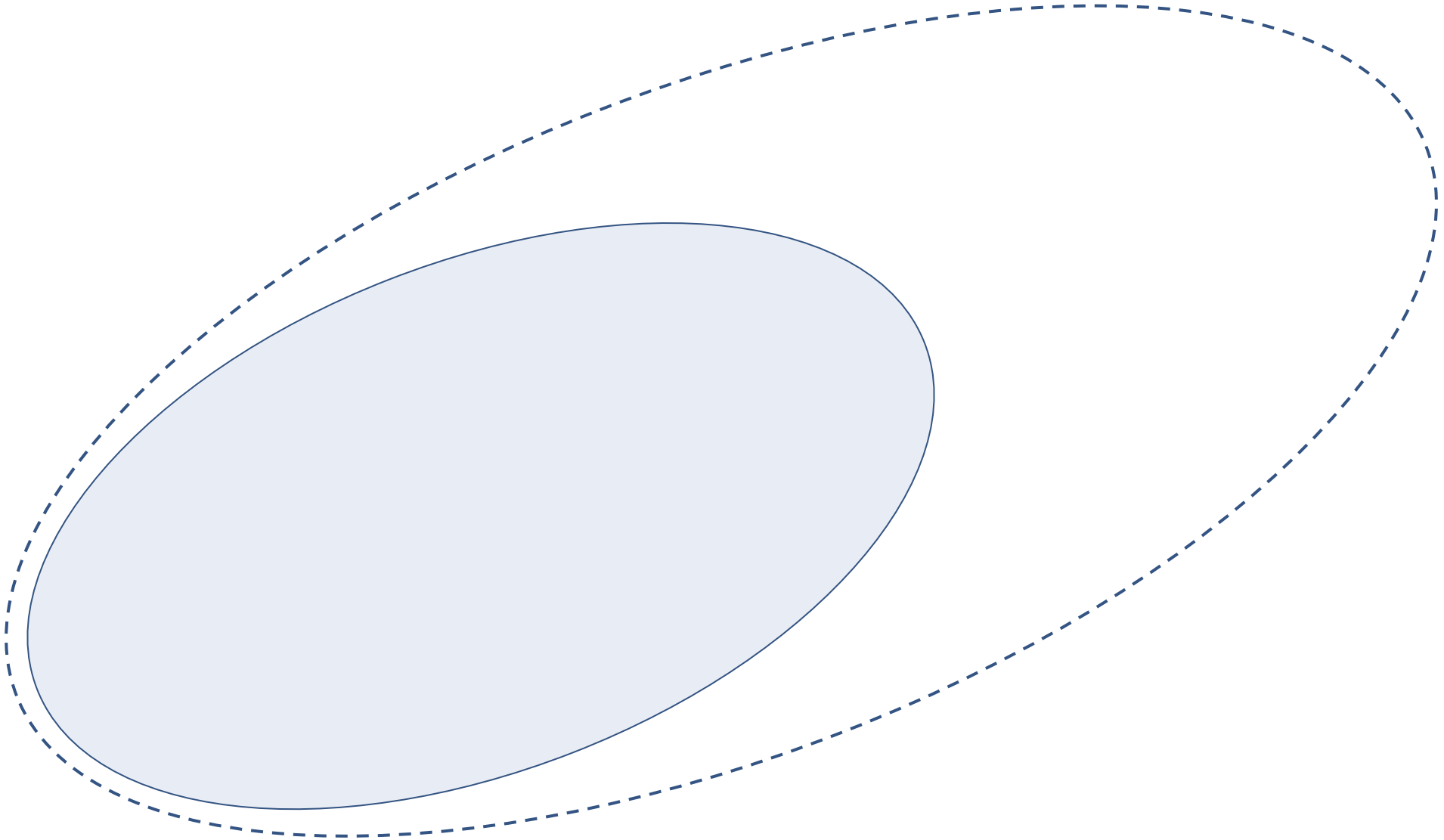
n parties: $O(n^2)$ shared keys

- Non-interactive protocol: $O(n)$ messages
- Interactive protocol: $O(n^2)$ messages

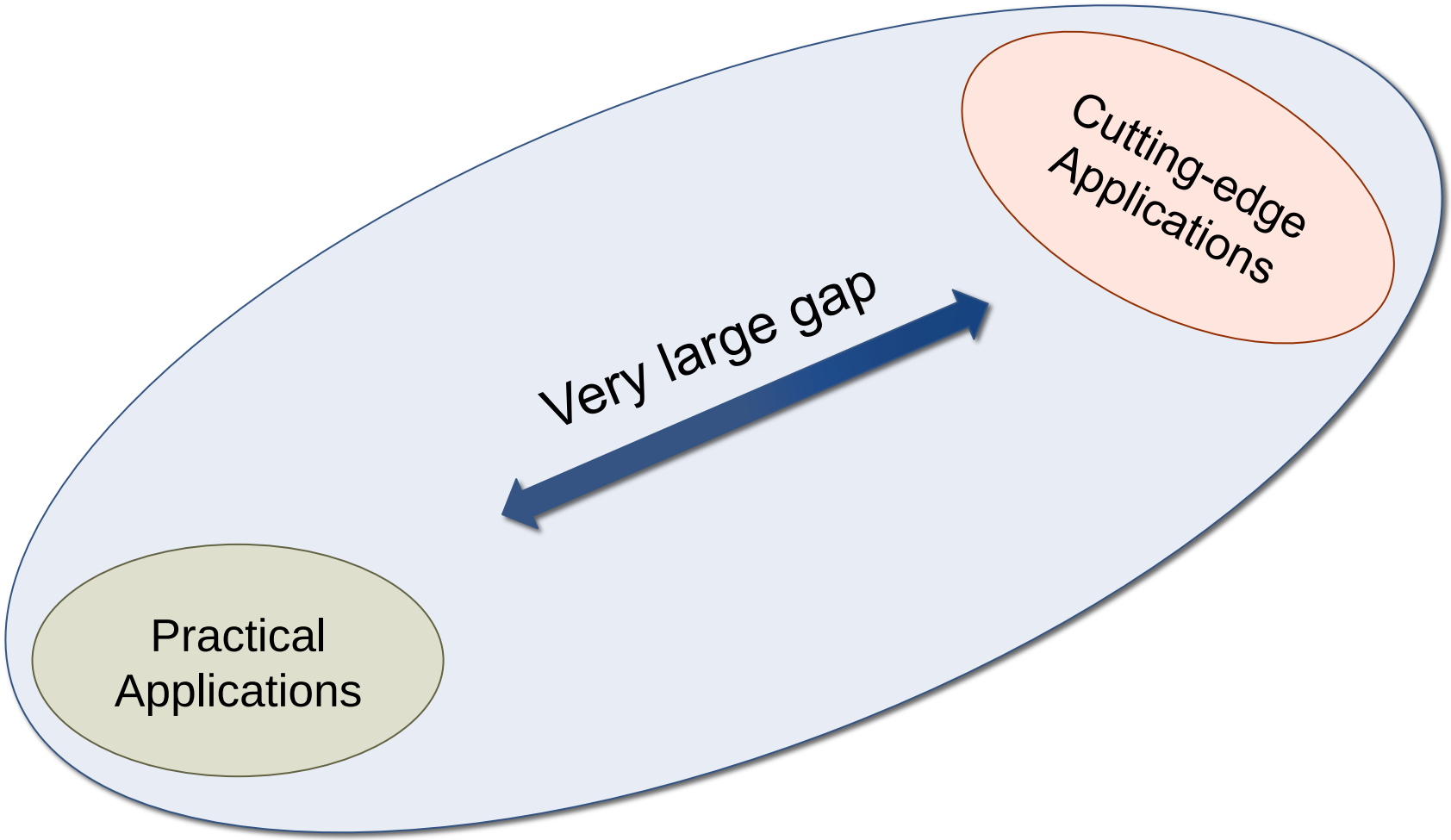
The Modern Cryptographic Landscape



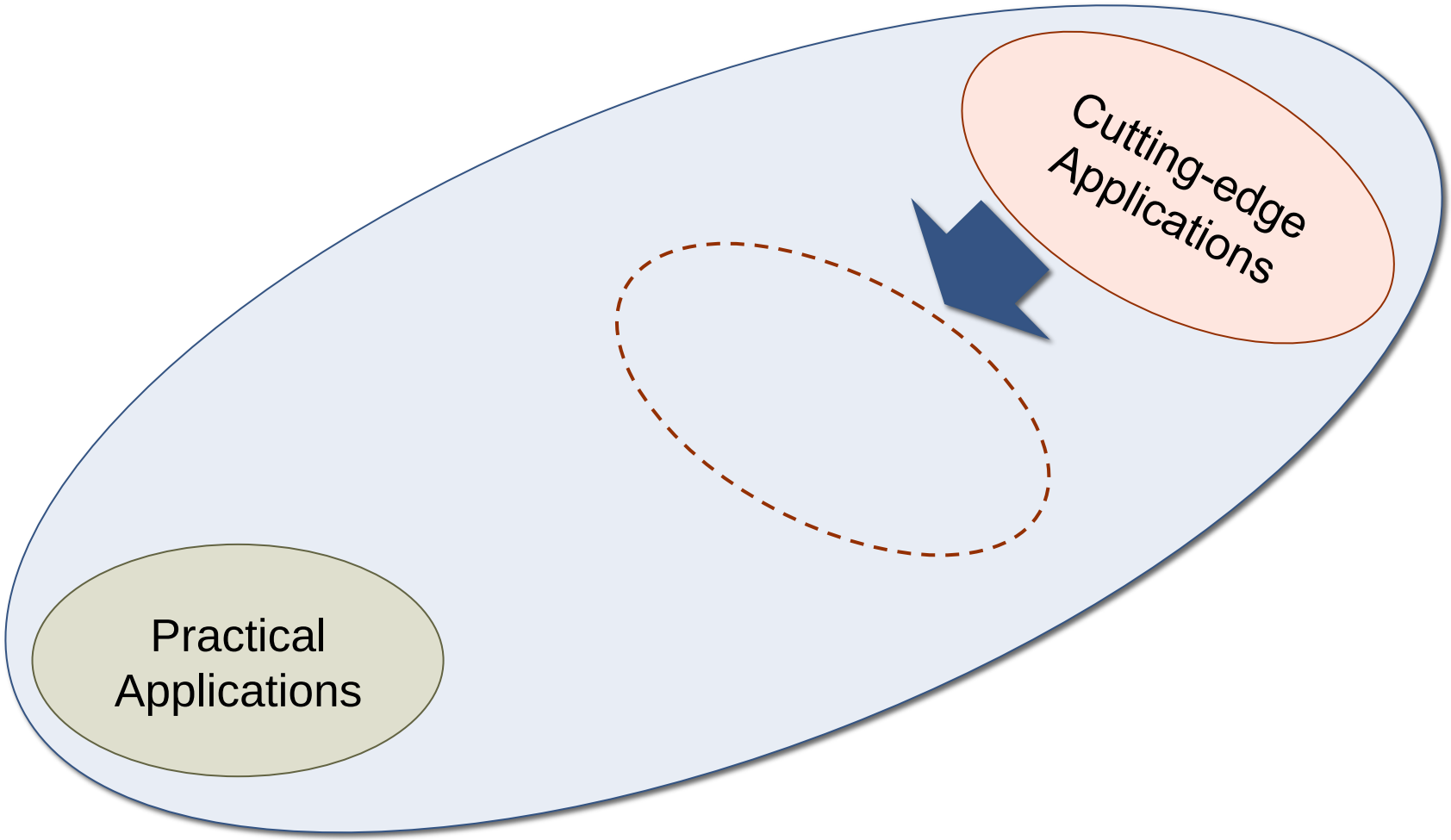
Goal 1: New Cutting-Edge Crypto



Another Look at the Cryptographic Landscape



Goal 2: Bring Cutting-Edge Closer to Practice



Contributions

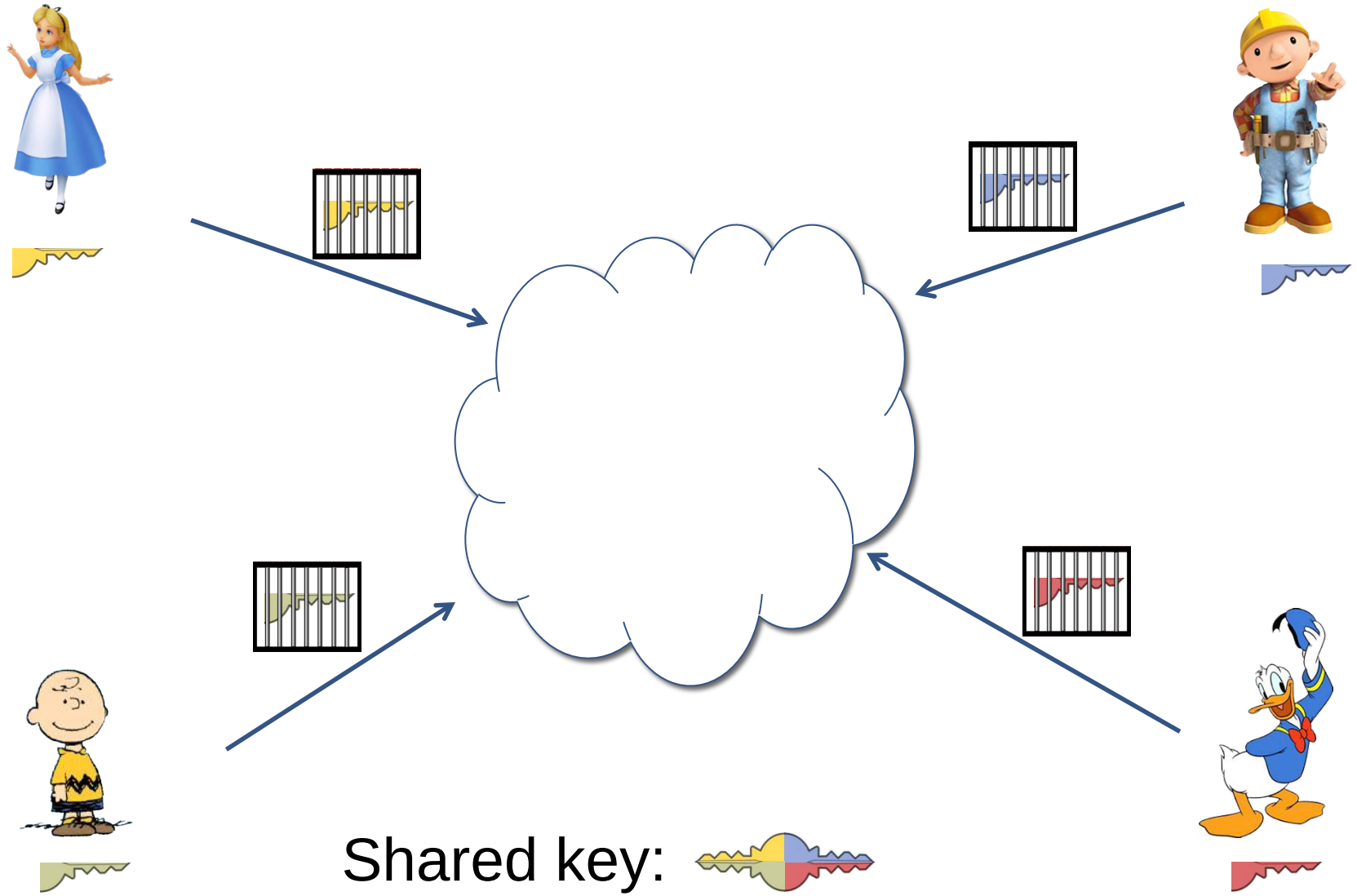
	Initial Constructions	Efficiency Improvements
Multiparty NIKE w/o Setup	[BZ'13, ABGSZ'13]	[Zha'14a]
Order-Revealing Encryption (ORE)	(Folklore)	[BLRSZZ'14]
Obfuscation	([GGHRSW'13])	[BMSZ'15]
Broadcast Encryption	[BZ'13, ABGSZ'13, Zha'14b]	[BWZ'14, Zha'14a, Zha'14b]
Functional Encryption	([GGHRSW'13])	[GGHZ'14]
Universal Samplers	[HJKSWZ'14]	(Hopefully still to come?)

[BZ'15]: ORE ⇒ impossibility for differentially-private learning

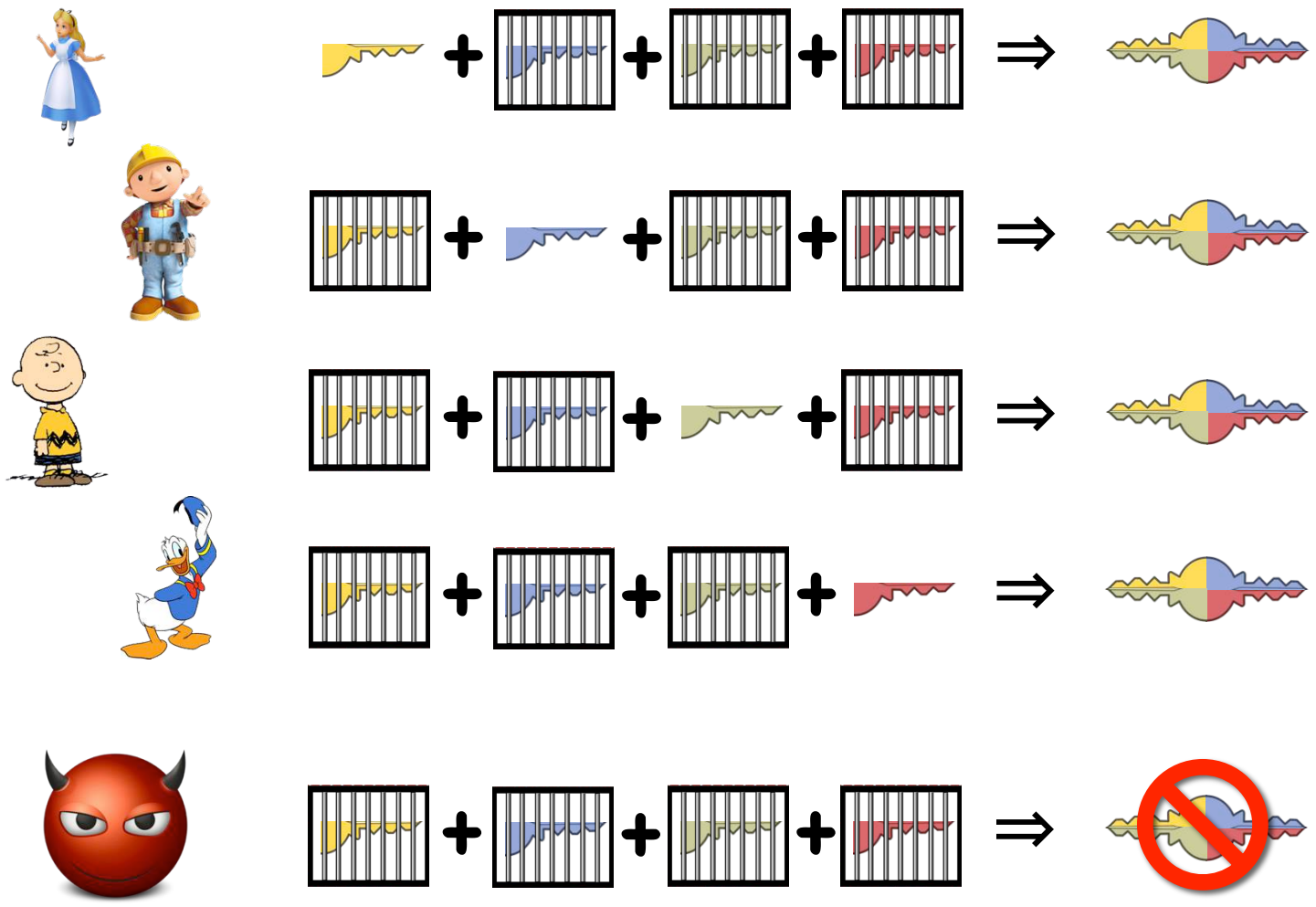
This Talk:
Multiparty Key Exchange

Motivation: Group Communication

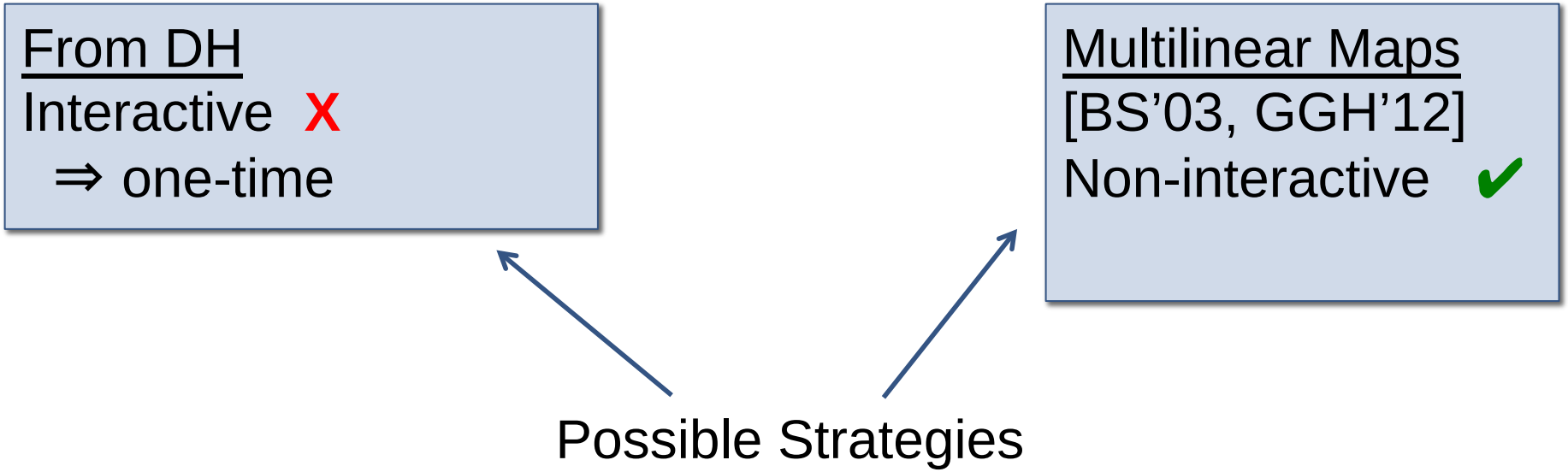
Multiparty NIKE: Natural Generalization



Multiparty NIKE: Natural Generalization



Constructing Multiparty Key Exchange



Tool: Cryptographic Multilinear Maps [BS'03]

"Source Group" $G = \{ 1, g, g^2, g^3, \ldots \}$

"Target Group" $G_T = \{ 1, g_T, g_T^2, g_T^3, \ldots \}$

"Pairing" operation: $e: G^n \rightarrow G_T$ where

$$e(g^a , g^b , \ldots , g^z) = g_T^{ab\ldots z}$$

2000-2001: **n=2** [Joux'00,BF'01]

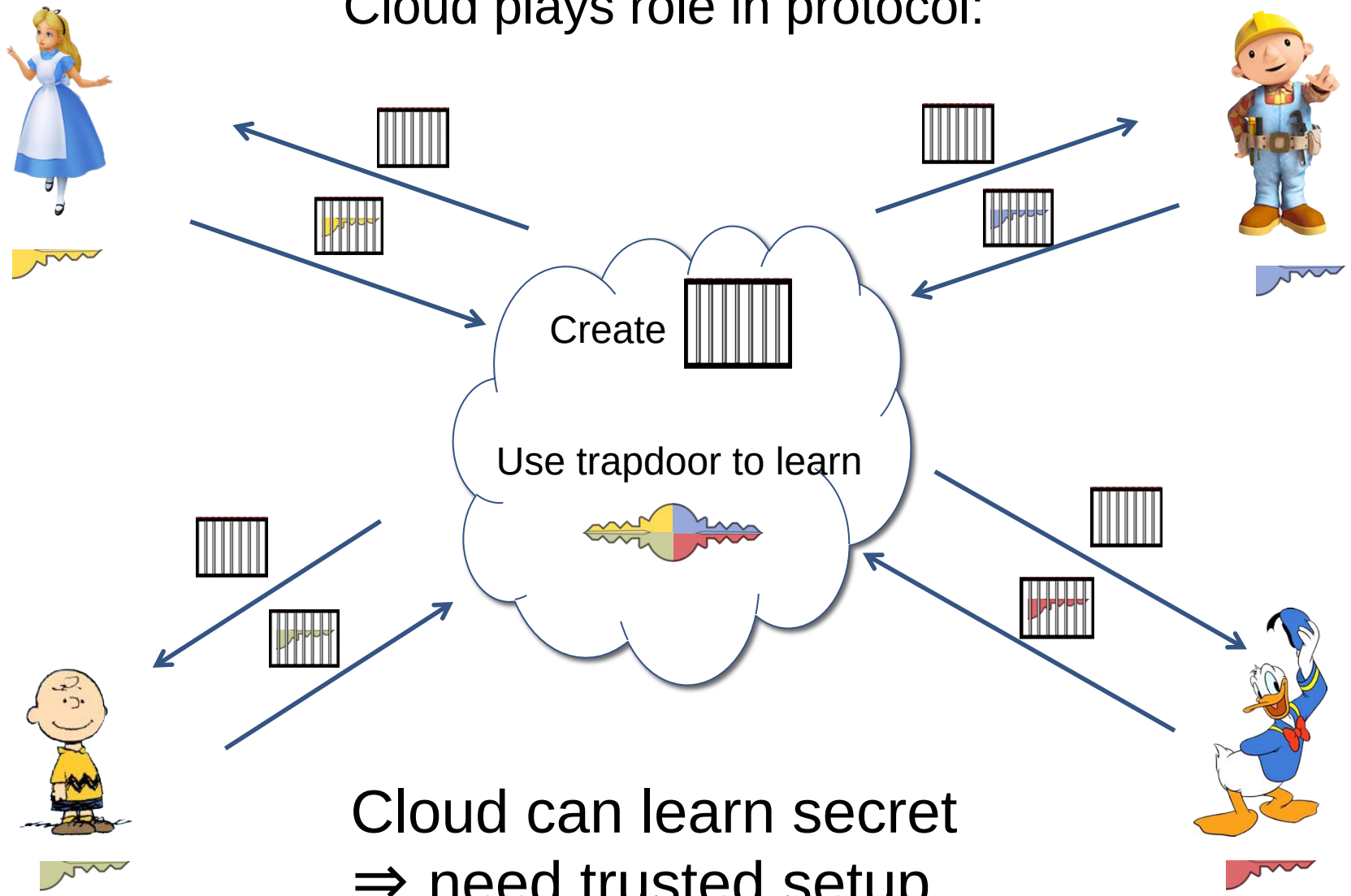
- Instrumental for solving many problems (IBE, 3-party NIKE,…)

2012: First candidate construction for **n>2** [GGH'12]

- **(n+1)**-party NIKE, obfuscation, and more

Limitation of NIKE Using Mmaps

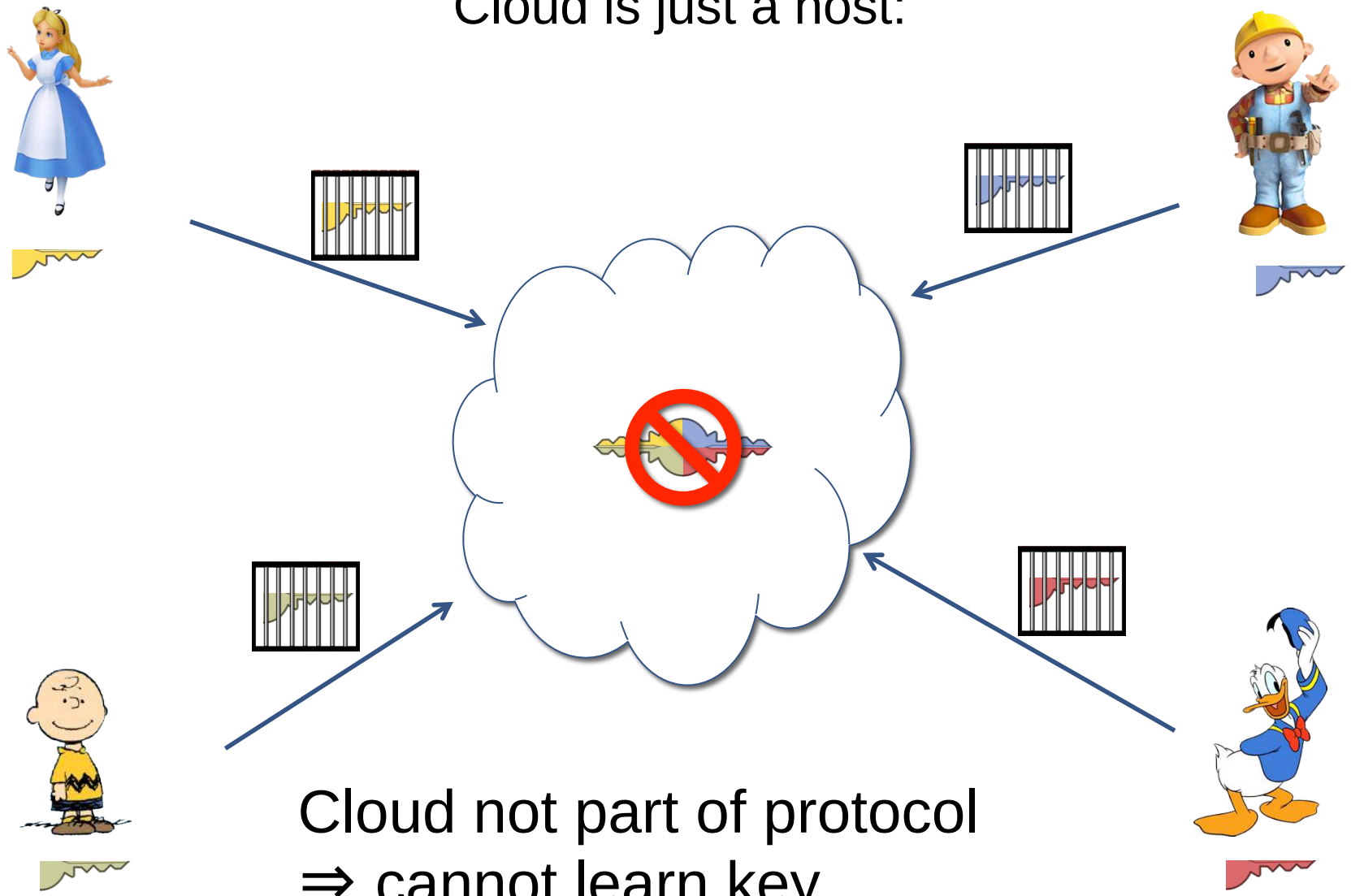
Cloud plays role in protocol:



Cloud can learn secret
⇒ need trusted setup

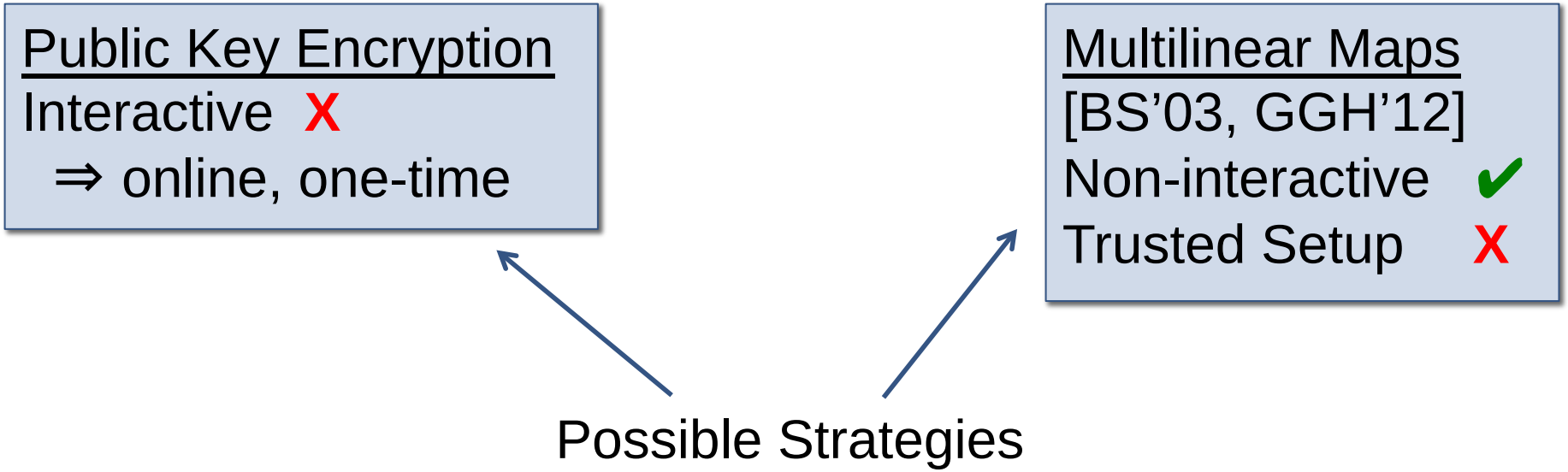
Goal: No Setup

Cloud is just a host:



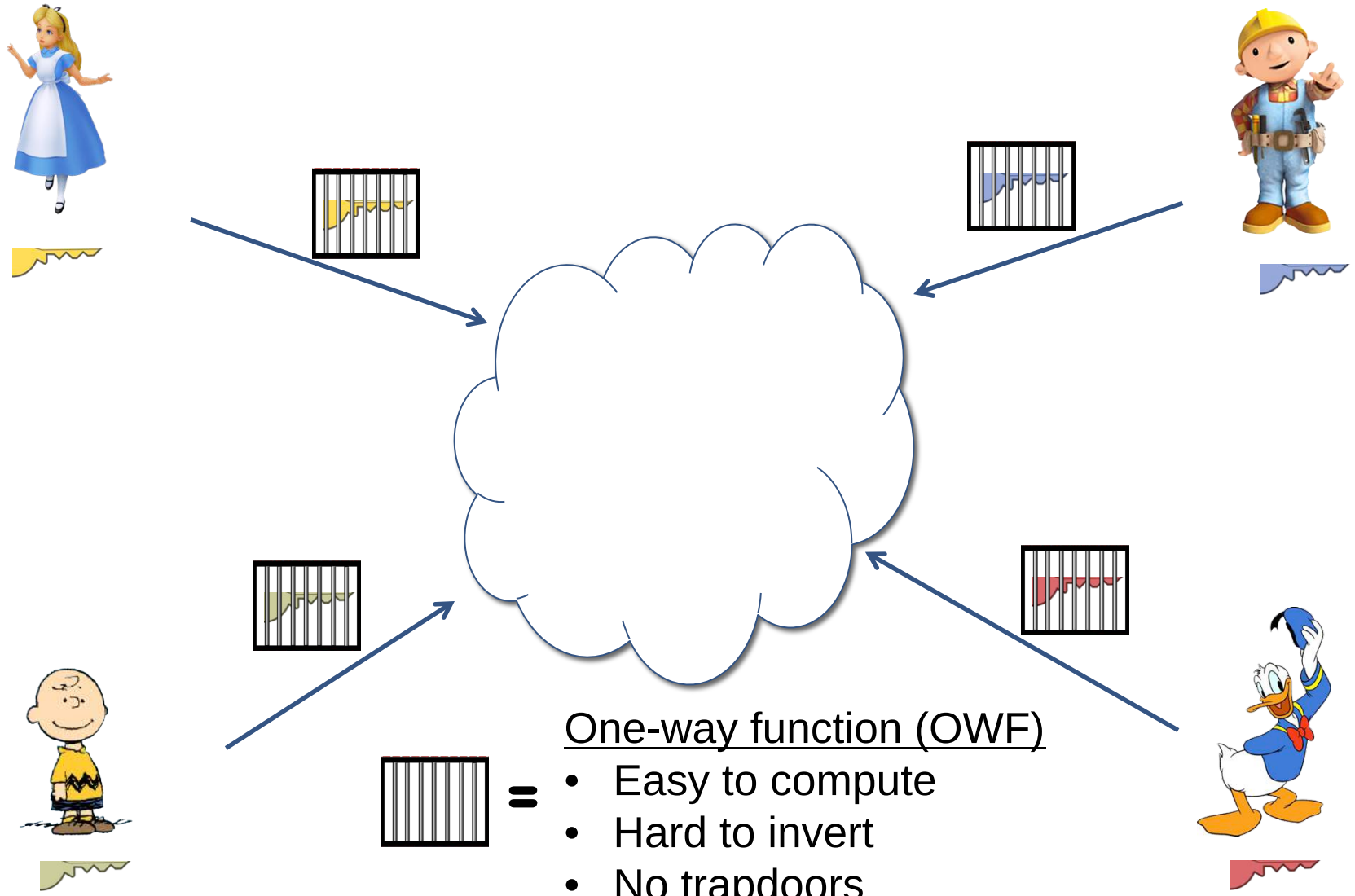
Cloud not part of protocol
⇒ cannot learn key

Constructing Multiparty Key Exchange



Removing Setup Using Obfuscation

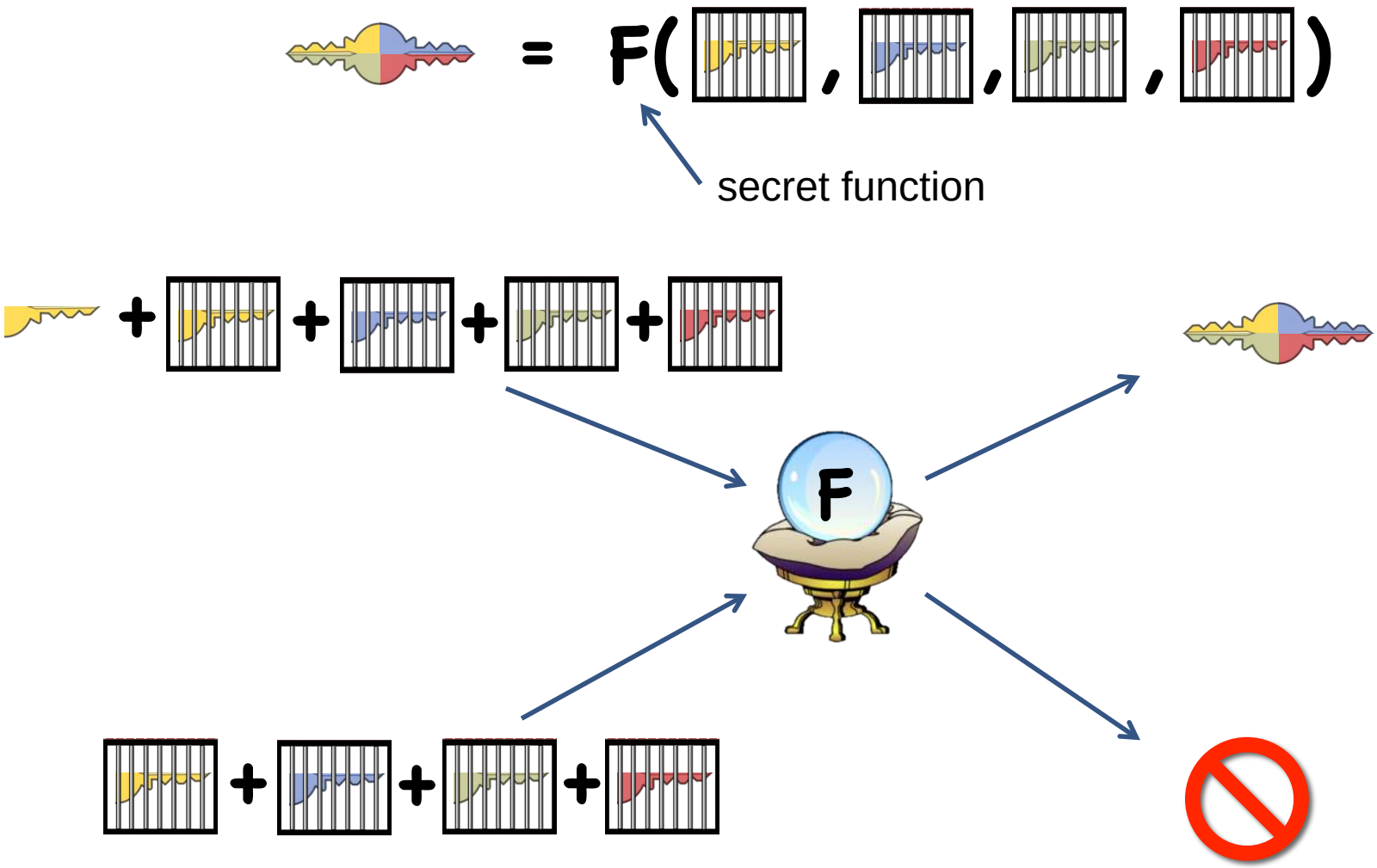
Step 1: Use basic locking mechanism



One-way function (OWF)

- Easy to compute
- Hard to invert
- No trapdoors

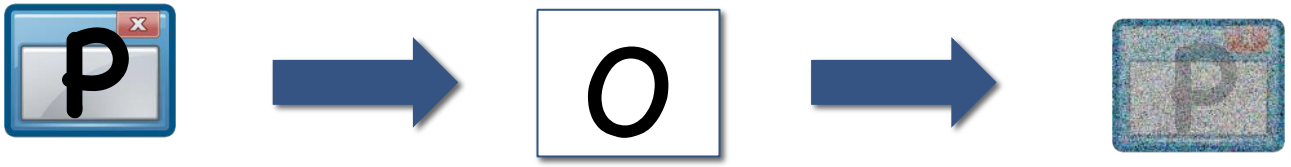
Dream World: Oracle Computes Shared Key



Problem: how to implement oracle without interaction?

General Purpose Program Obfuscation

"Scramble" a program, hiding implementation details

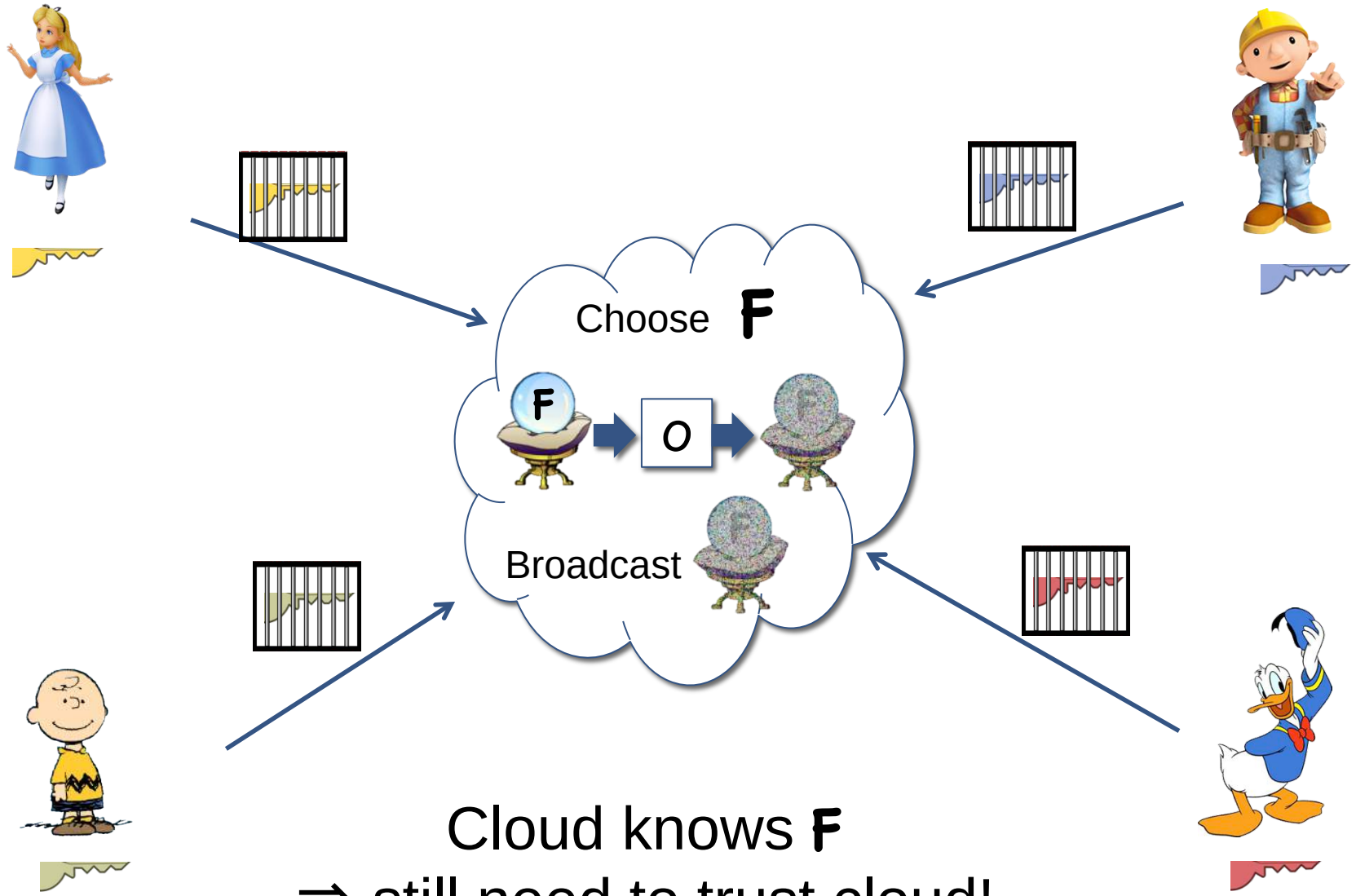


Requirements:

- Functionality:  functionally equivalent to 
- Efficiency: ***O*** "efficient",  "not much slower than" 
- Ideal security: having **code**  "as good as" **oracle** for 
  - Too strong ([BGIRSVY'01]), but provides intuition
  - In reality, use weaker notion: indistinguishability obfuscation (iO)

[GGHRSW'13]: First candidate obfuscator

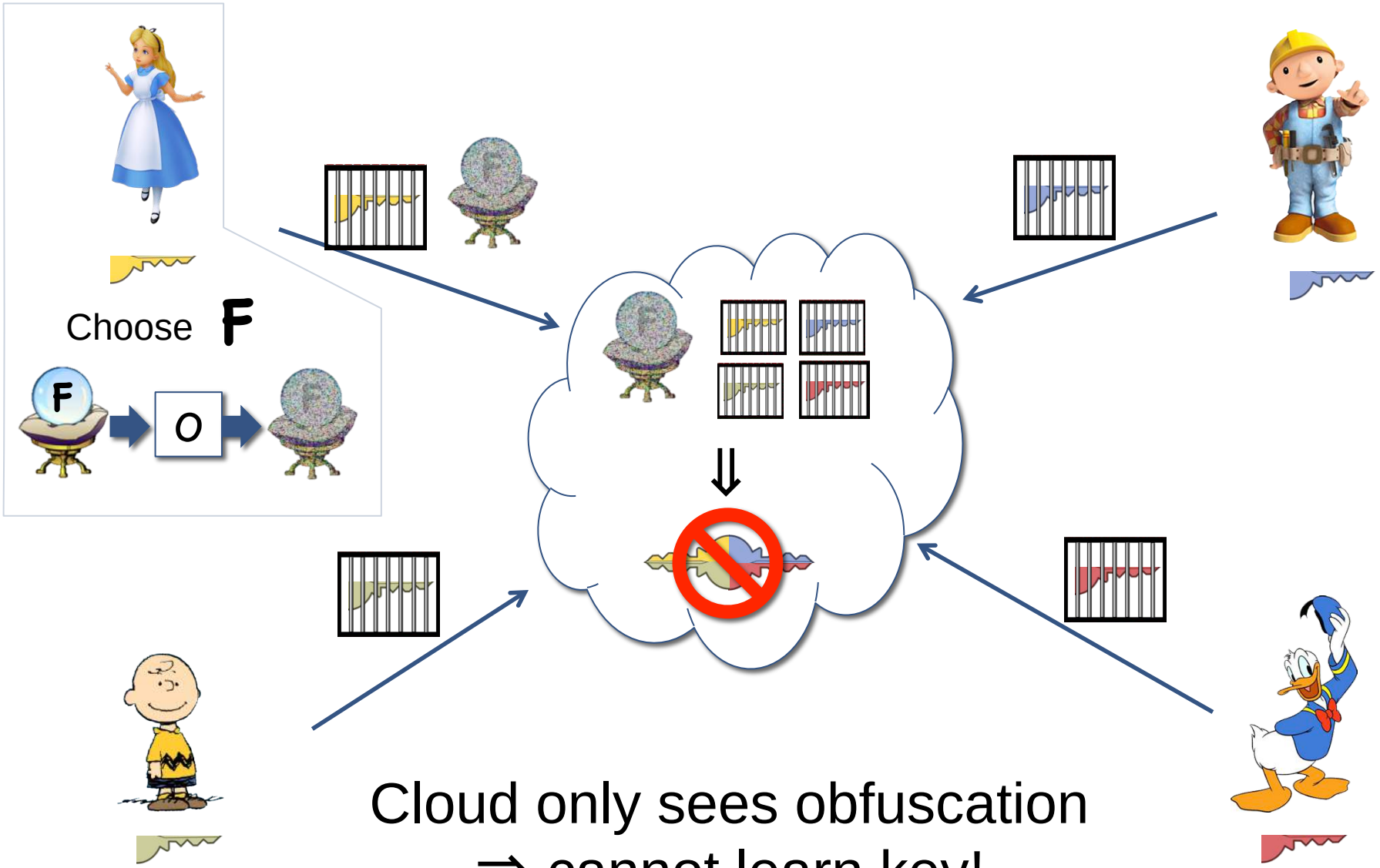
Step 2: Cloud Obfuscates Oracle



Cloud knows F

⇒ still need to trust cloud!

Step 3: “Master” User Obfuscates Oracle



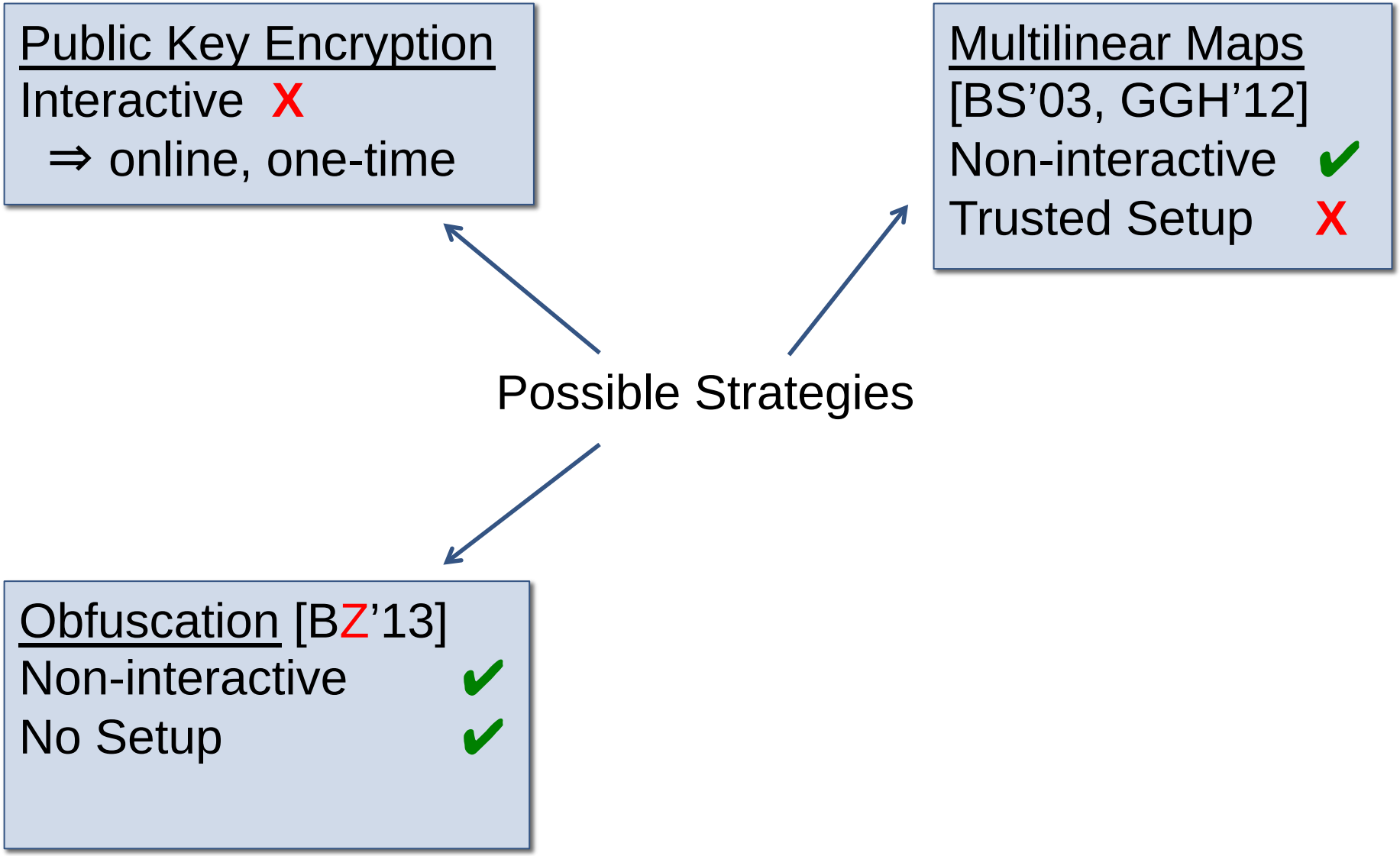
Security of Protocol

Theorem

iO + OWF  Multiparty NIKE

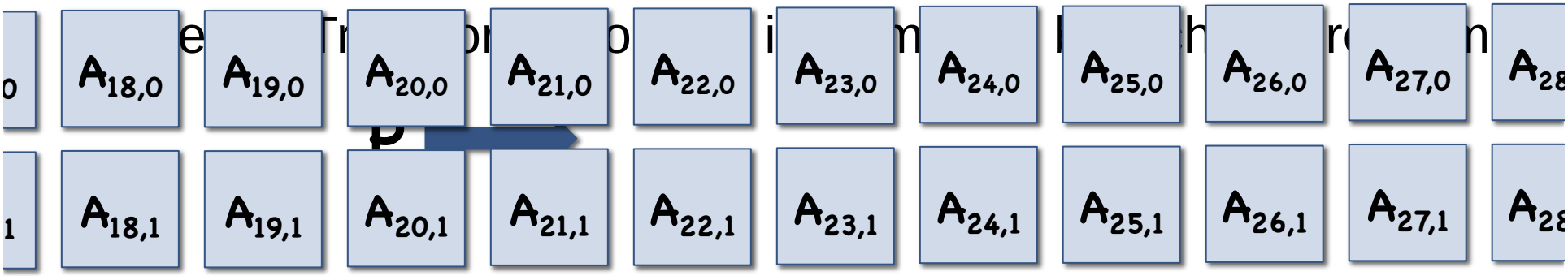
(without setup)

Constructing Multiparty Key Exchange



Implementing Obfuscation

Can build obfuscator using multilinear maps [GGHRSW'13]



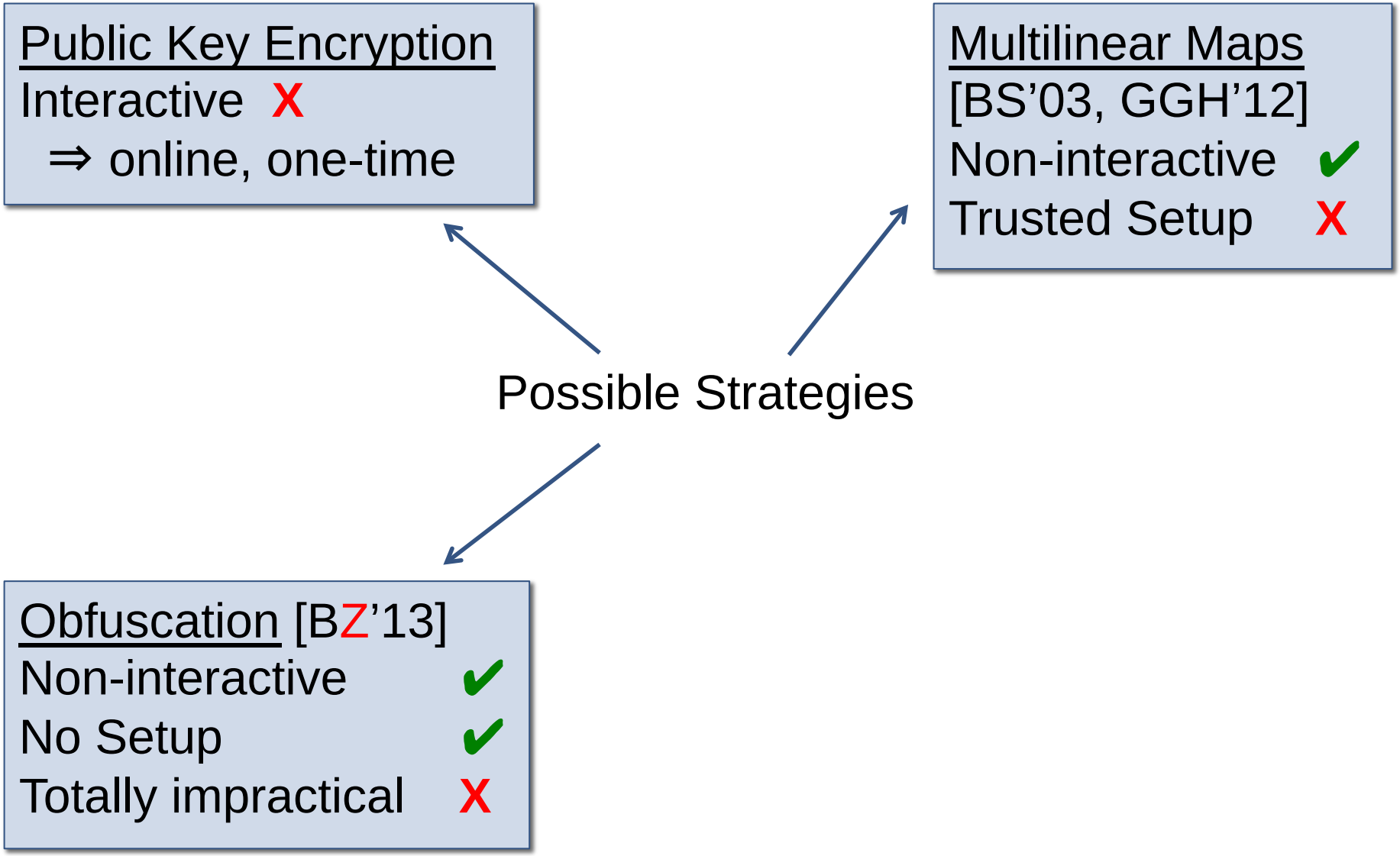
- Step 2: Encode branching program in multilinear map

- Step 3: Additional costly steps
  - FHE and more

[AGIS'14,BMSZ'15]: Efficiency improvements for Step 1

Still extremely impractical

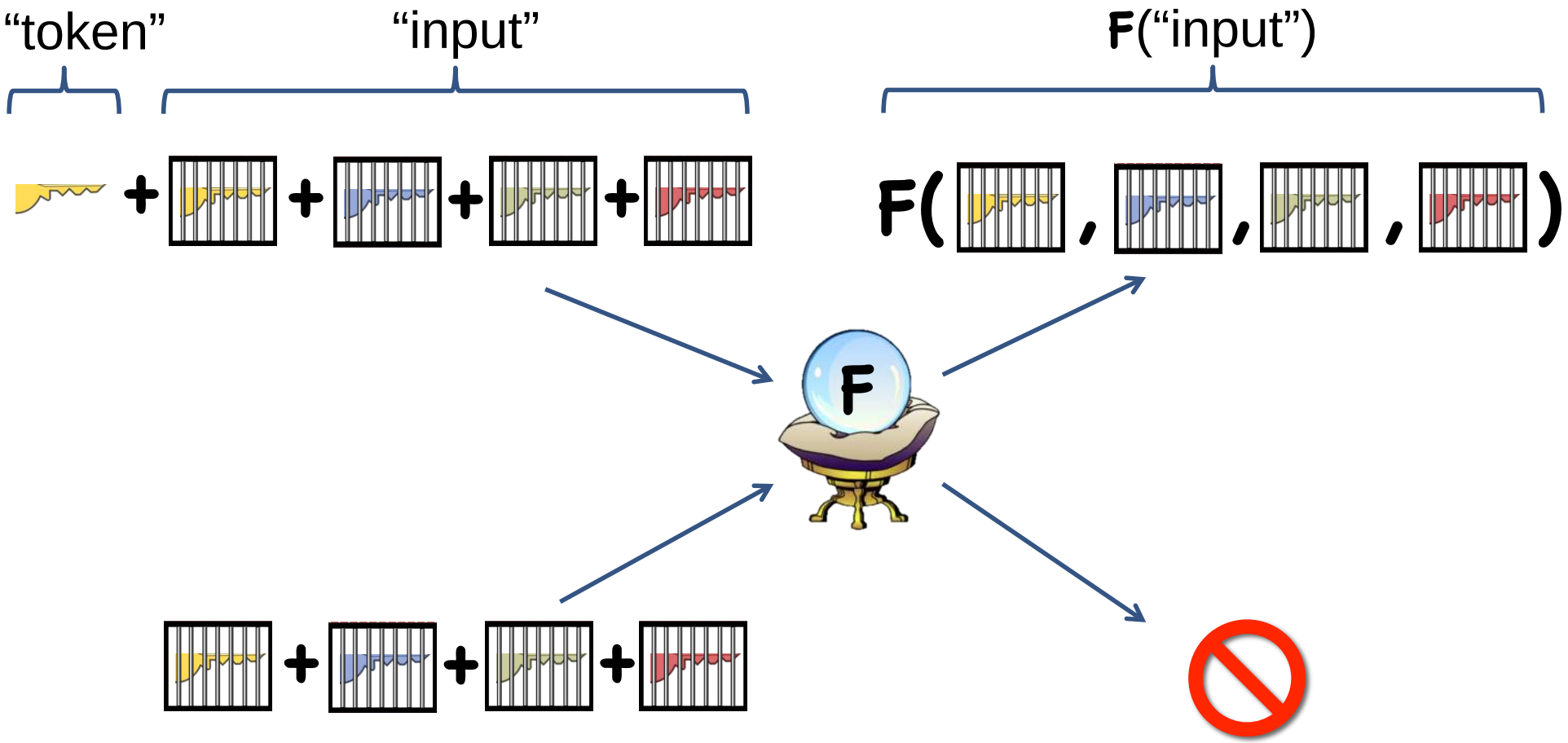
Constructing Multiparty Key Exchange



Removing Obfuscation

Are we working too hard?

Only need to obfuscate “gatekeeper” programs



New Primitive: Witness PRFs [Zha'14]

NP Language **L**

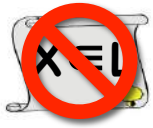
"Generation" procedure

Secret function

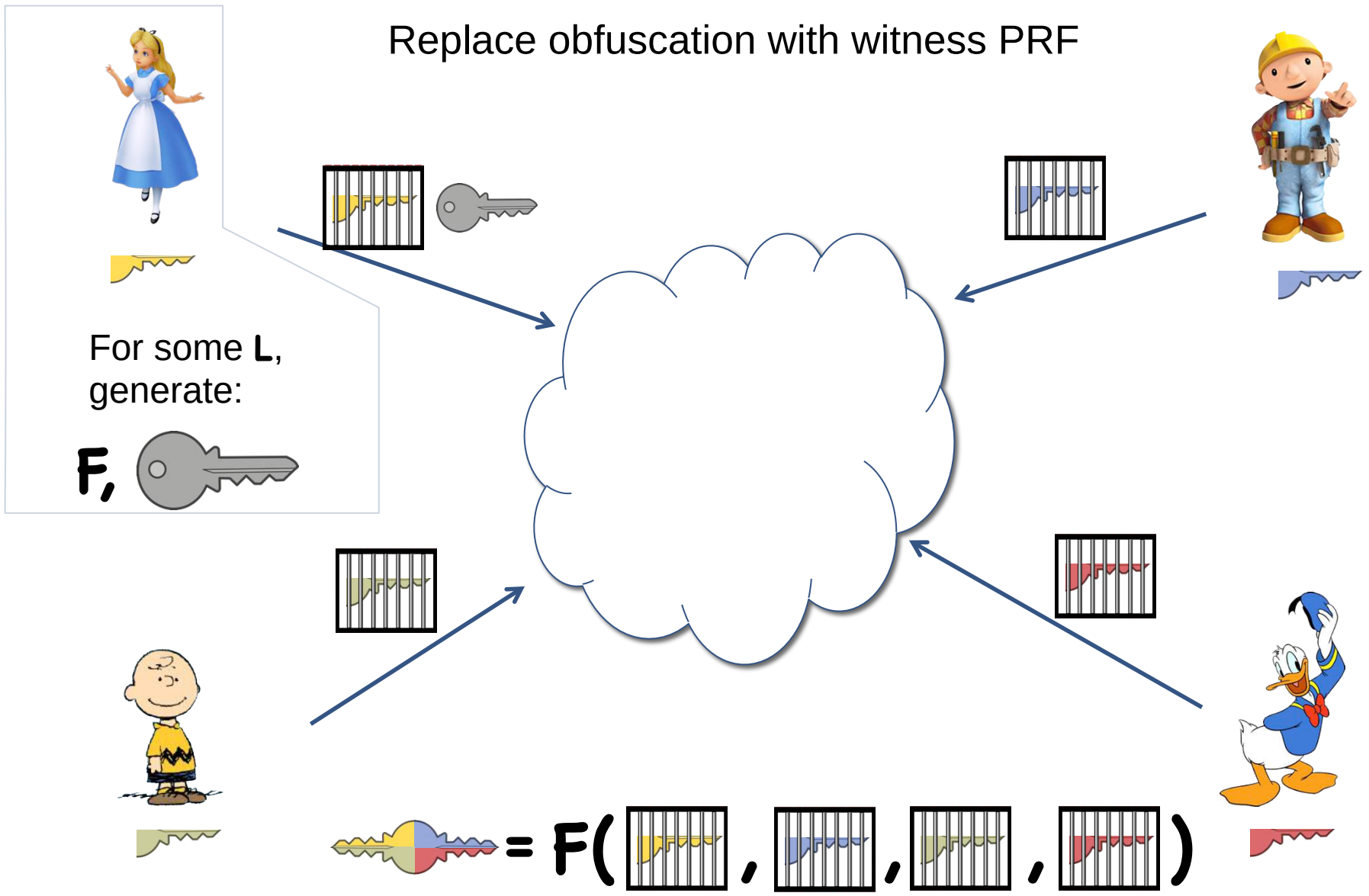
F

Domain = instances

Public evaluation key

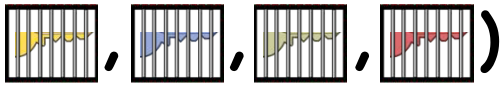
$x \in L$	$x \notin L$
$F \Rightarrow F(x)$	
 +  $\Rightarrow F(x)$	 $\Rightarrow$ 
 +  $\Rightarrow$?	

Key Exchange from Witness PRFs



Key Exchange from Witness PRFs

What language **L** should be used to generate **F**,  ?

Each party must have witness for **z** = ()

- In particular, **z**∈**L**

Need witness hard to compute given **z**

- Obvious witnesses: 

L = (1, 2, 3, 4) where:

∃ i,  s.t. i = 

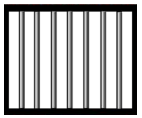
Security of Protocol

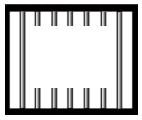
Theorem

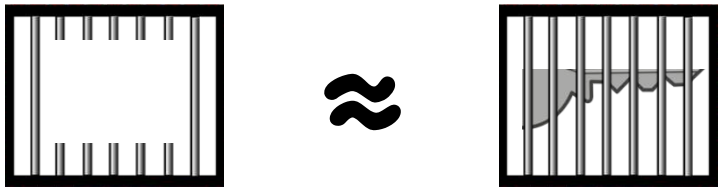
Witness PRFs + OWF ➡ Multiparty NIKE (without setup)

Witness PRFs ➡ OWF

Minor Modification to Protocol

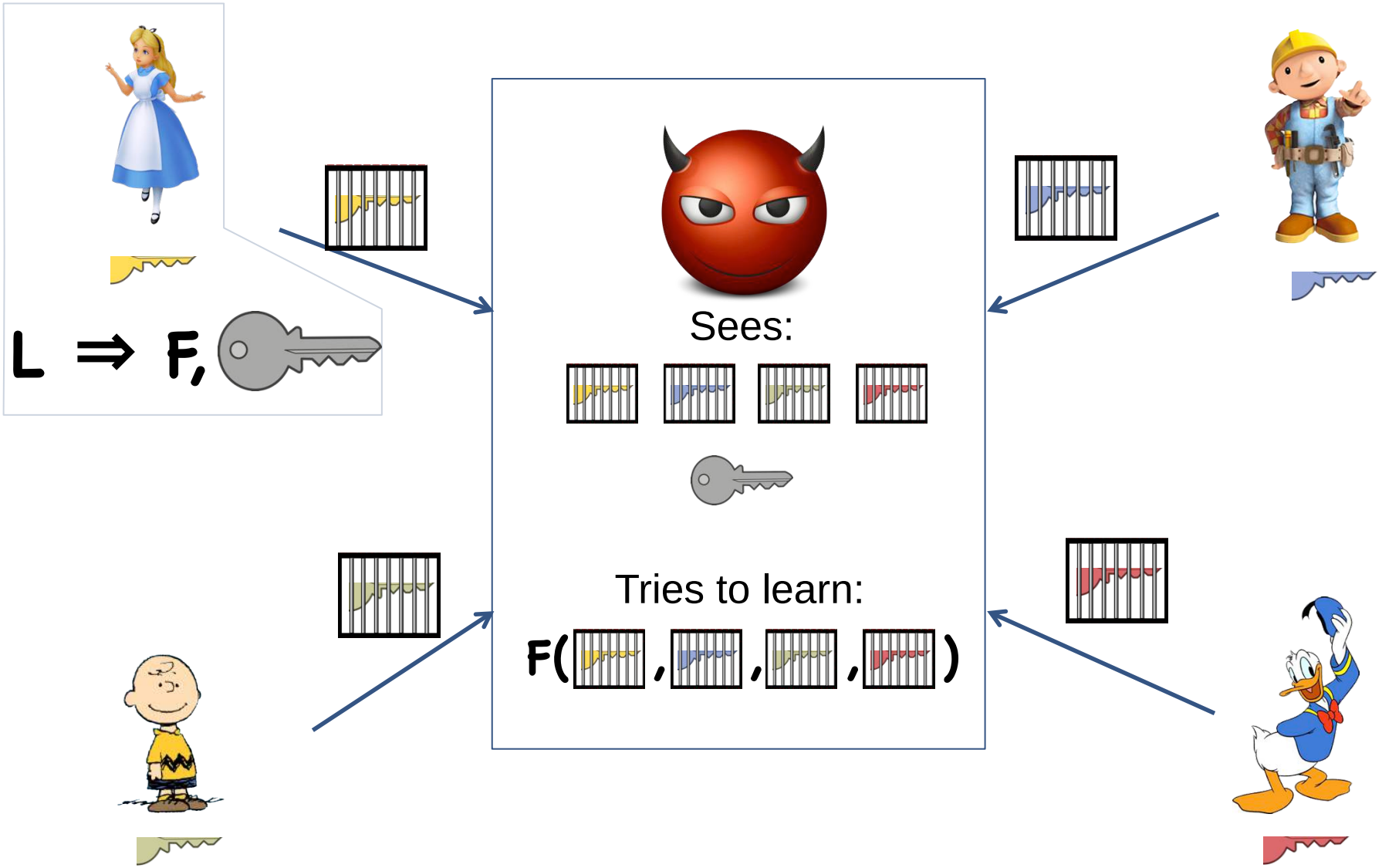
Assume  is a **pseudorandom generator (PRG)**

- Possible to generate “empty” boxes  with no keys
- “Empty” box indistinguishable from “full” box

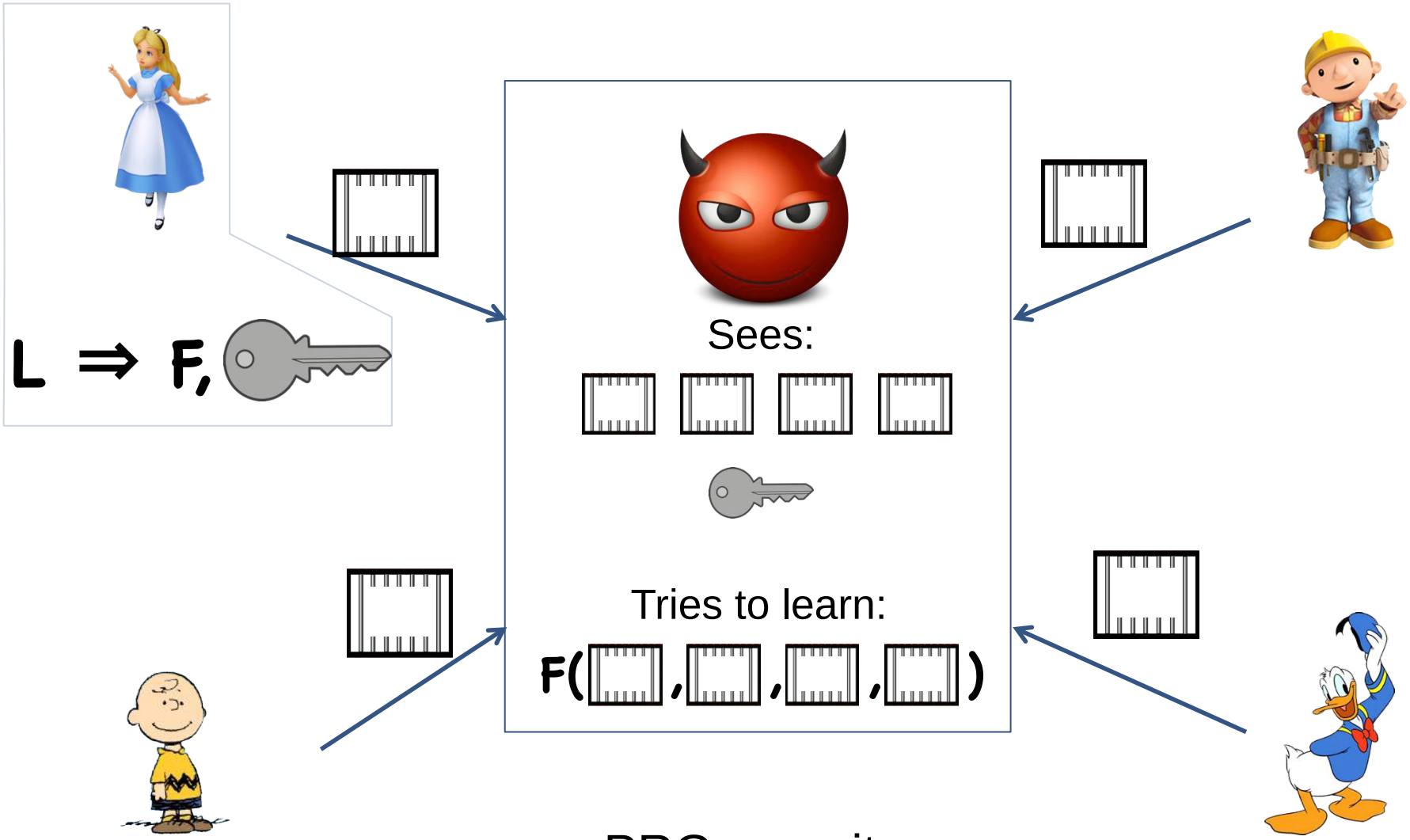


Can build from any one-way function [HILL’99]

Security Proof



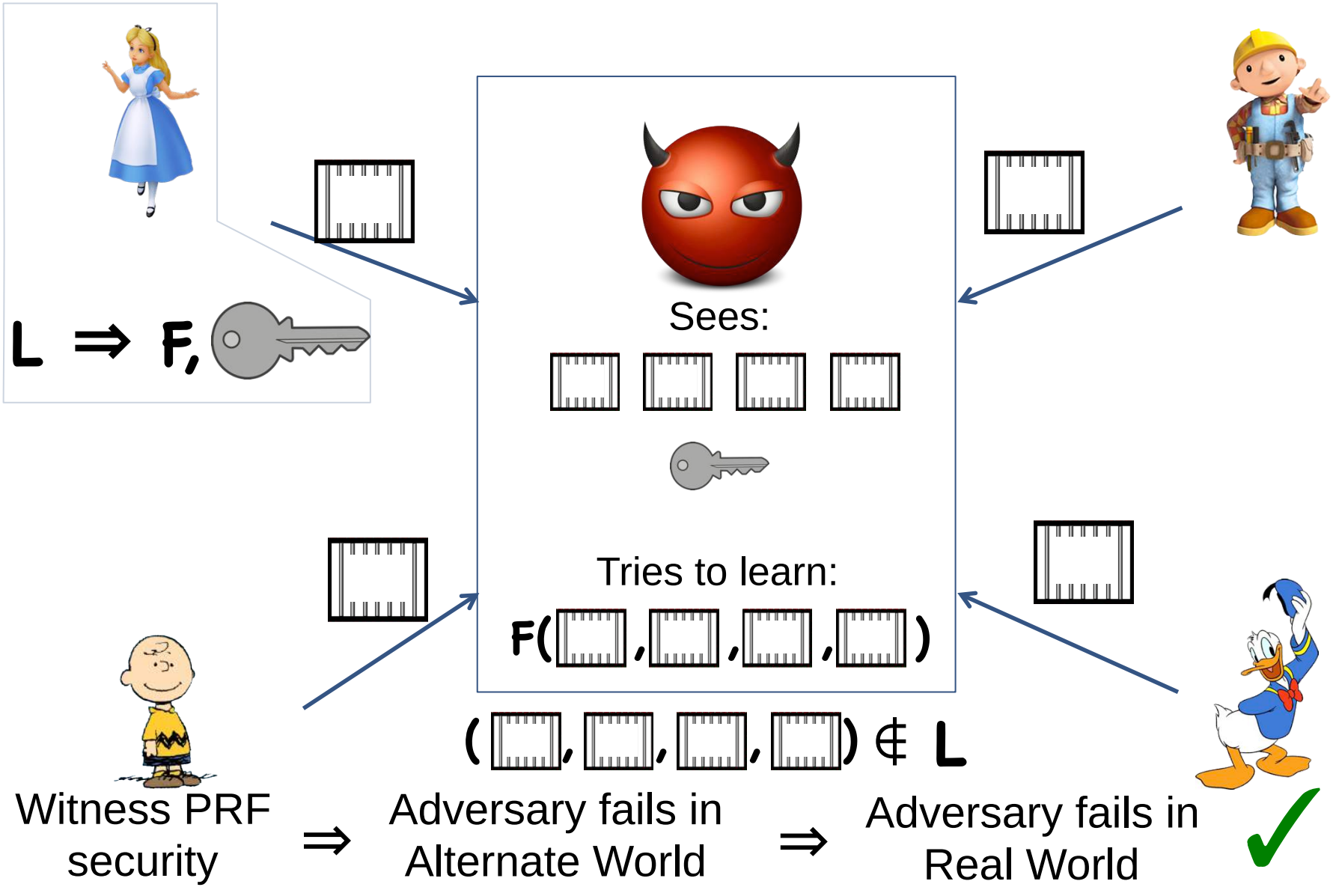
Alternate World: Empty Boxes



PRG security:

Success in Real World ⇔ Success in Alternate World

Invoke Witness PRF Security



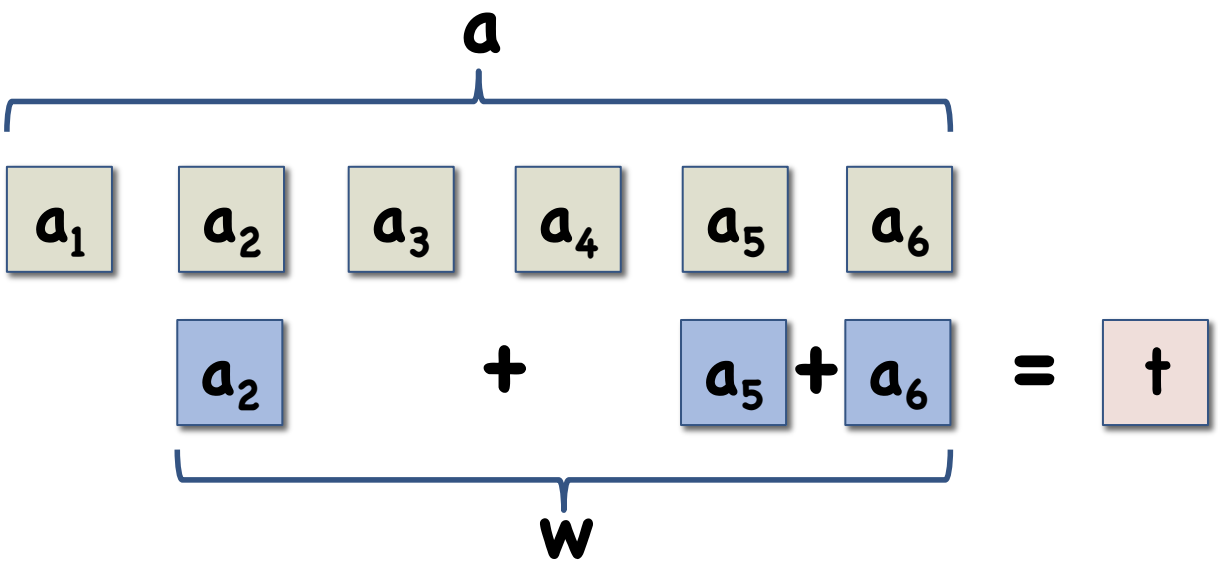
Constructing Witness PRFs

Construction of Witness PRFs

First, construct witness PRF for limited class of languages:

SubSums($a \in \mathbb{Z}^n$) = { $a \cdot w : w \in \{0,1\}^n$ }
(i.e. subset-sums of elements of **a**)

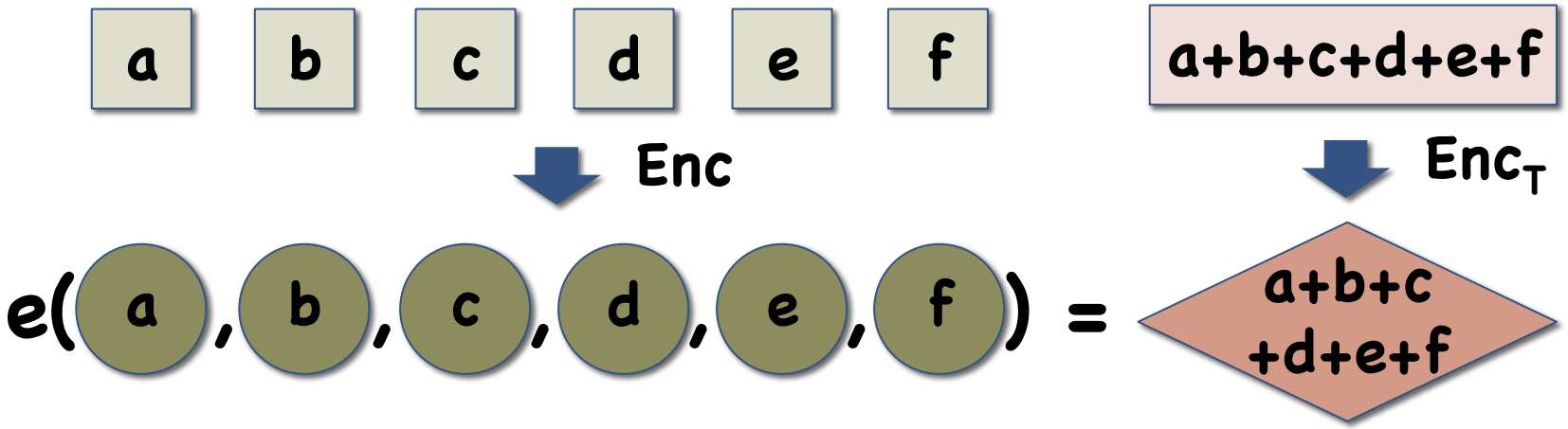
Subset-sum problem: given **($a \in \mathbb{Z}^n$, $t \in \mathbb{Z}$)**, determine if **$t \in$ SubSums(a)**



Witness PRF for **SubSums(a)**:

Given a_1 a_2 a_3 a_4 a_5 a_6,

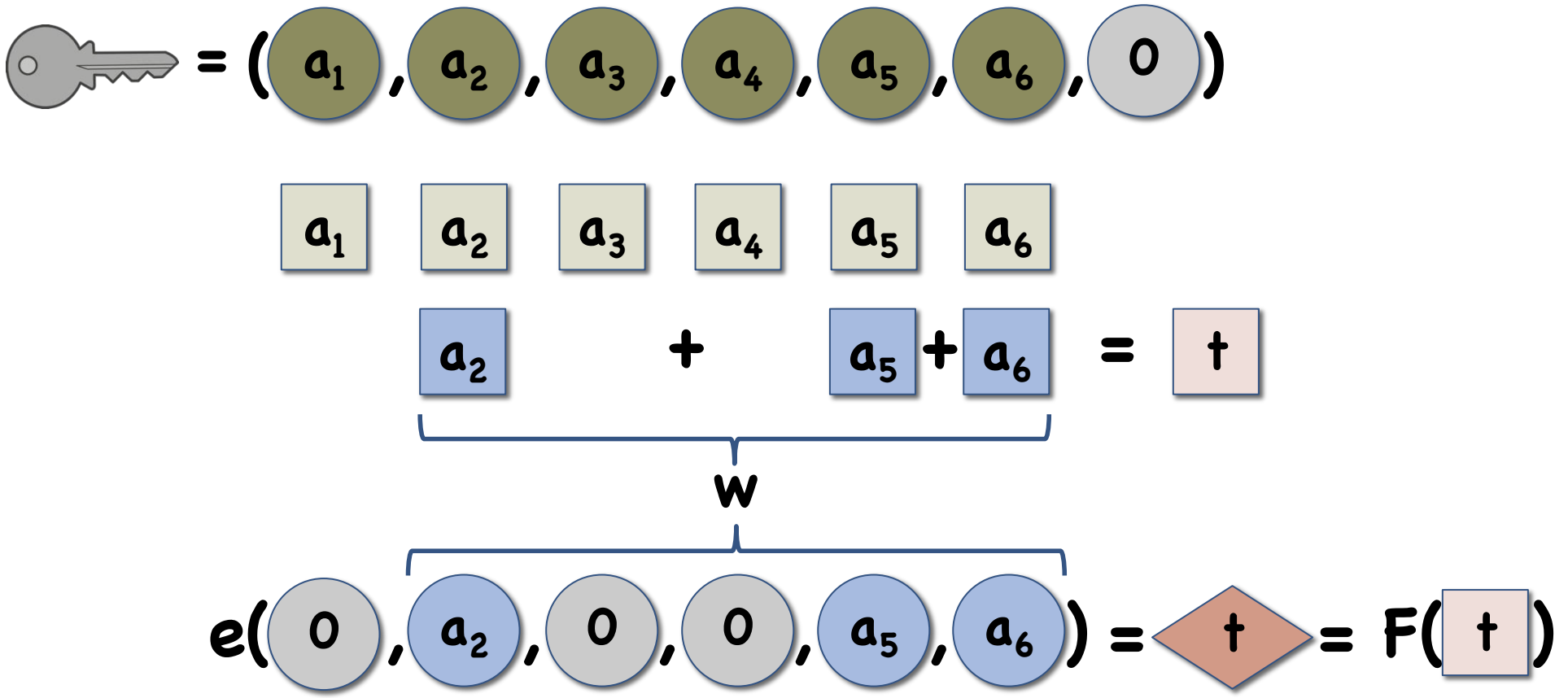
- Construct "encoding functions" **Enc:** $\mathbb{Z} \rightarrow G$, **Enc$_T$:** $\mathbb{Z} \rightarrow G_T$ s.t.



- **F(** t **) = Enc$_T$(** t **) =** t

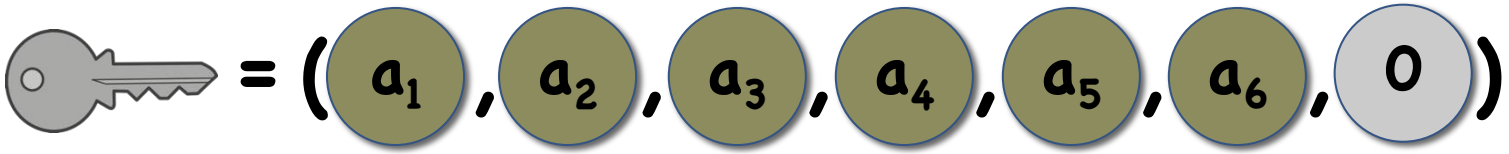
-  = (a_1 , a_2 , a_3 , a_4 , a_5 , a_6 , 0)

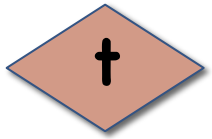
How to Evaluate with Witness?

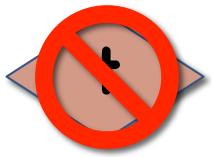


Final Pieces

Security: If $t \notin \mathbf{SubSums}(a)$, no subset* of



pairs together to form 

⇒ Complexity assumption:  given 

Witness PRF for NP: we give new reduction from any language $L \in NP$ to **SubsetSum** where:

- a only depends on language (not instance)
- t depends on language and instance

Witness PRF for **SubSums** ⇒ Witness PRF for **NP**

Comparison for Key Exchange

Use multilinearity **n** as proxy for efficiency

Basic mmap protocol: **$n = \#(users) - 1$**

- With trusted setup

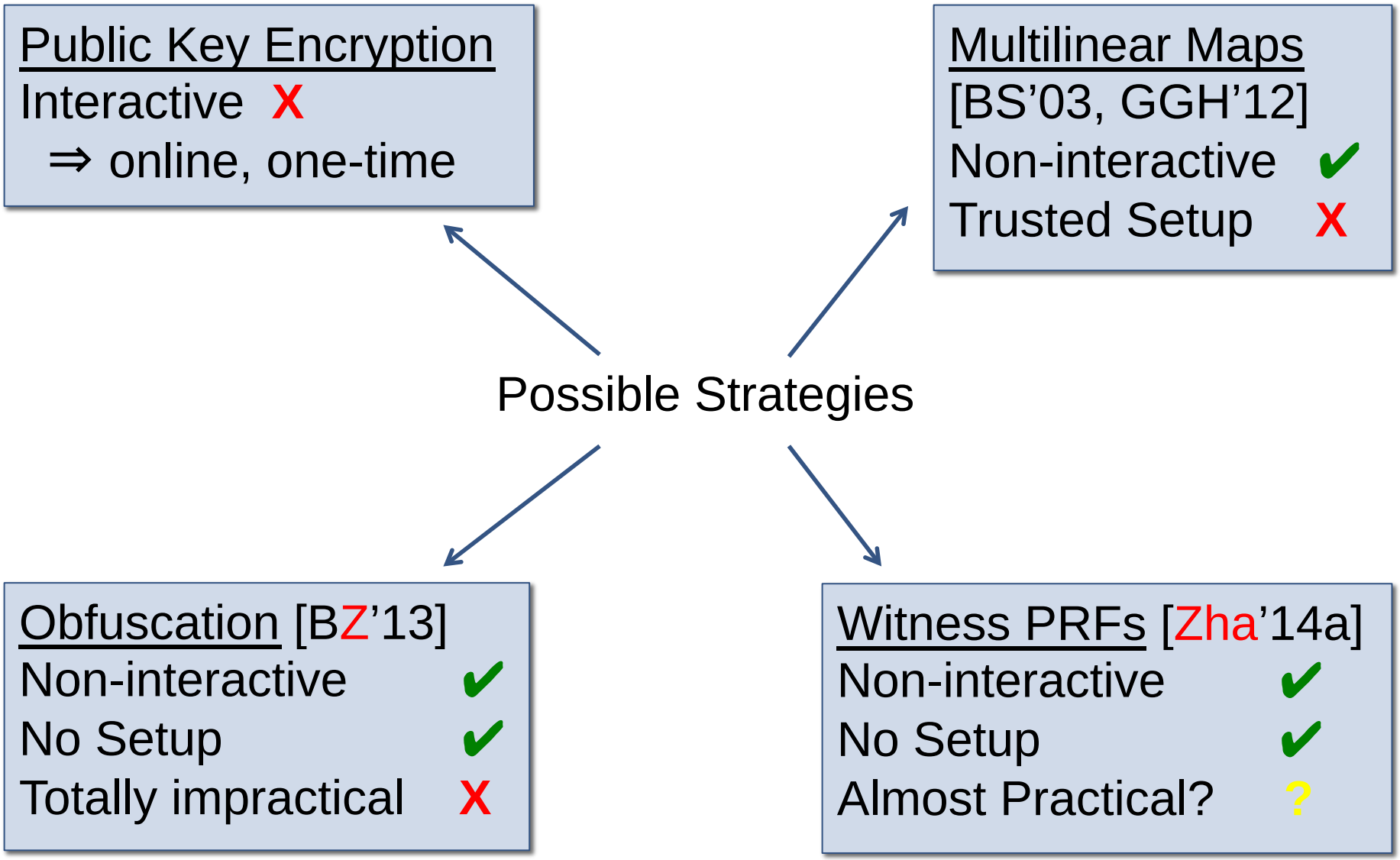
Obfuscation: **$n = (\#(users) \times \lambda)^c$**

- Constant **$c > 5$** (maybe much larger)
- λ = security parameter (can set to 128)

Witness PRF: **$n = \#(users) \times 8\lambda + O(\lambda)$**

- Still room for improvement, but orders of magnitude better

Constructing Multiparty Key Exchange



Still Much Work To Do

Theory: asymptotically polynomial ⇒ “efficient”

Practice: still far from implementation

Goal 1: Bring protocols even closer to practice

- Using LWE?
- By building more efficient multilinear maps?

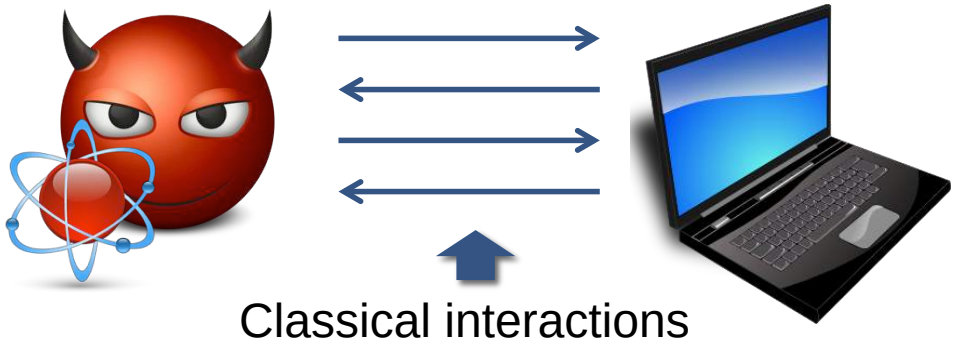
Goal 2: Better security understanding for obfuscation, multilinear maps

Goal 3: Experiment with implementation

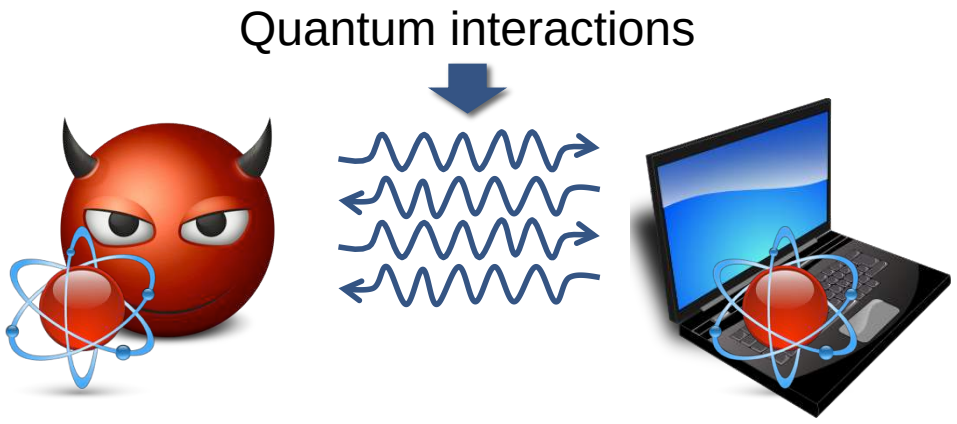
Into the Future: Quantum

Quantum Attacks on Classical Crypto

Post-quantum attacks (aka quantum computing attacks):



New quantum attacks:



Quantum Attacks on Classical Crypto

[BDFLSZ'11, Zha'12a]: Quantum random oracle model

[Zha'12b]: Pseudorandom functions

[BZ'13a]: Message authentication codes

[BZ'13b]: Signatures and encryption

[Zha'13]: Quantum collision resistance of random oracles

Thanks!