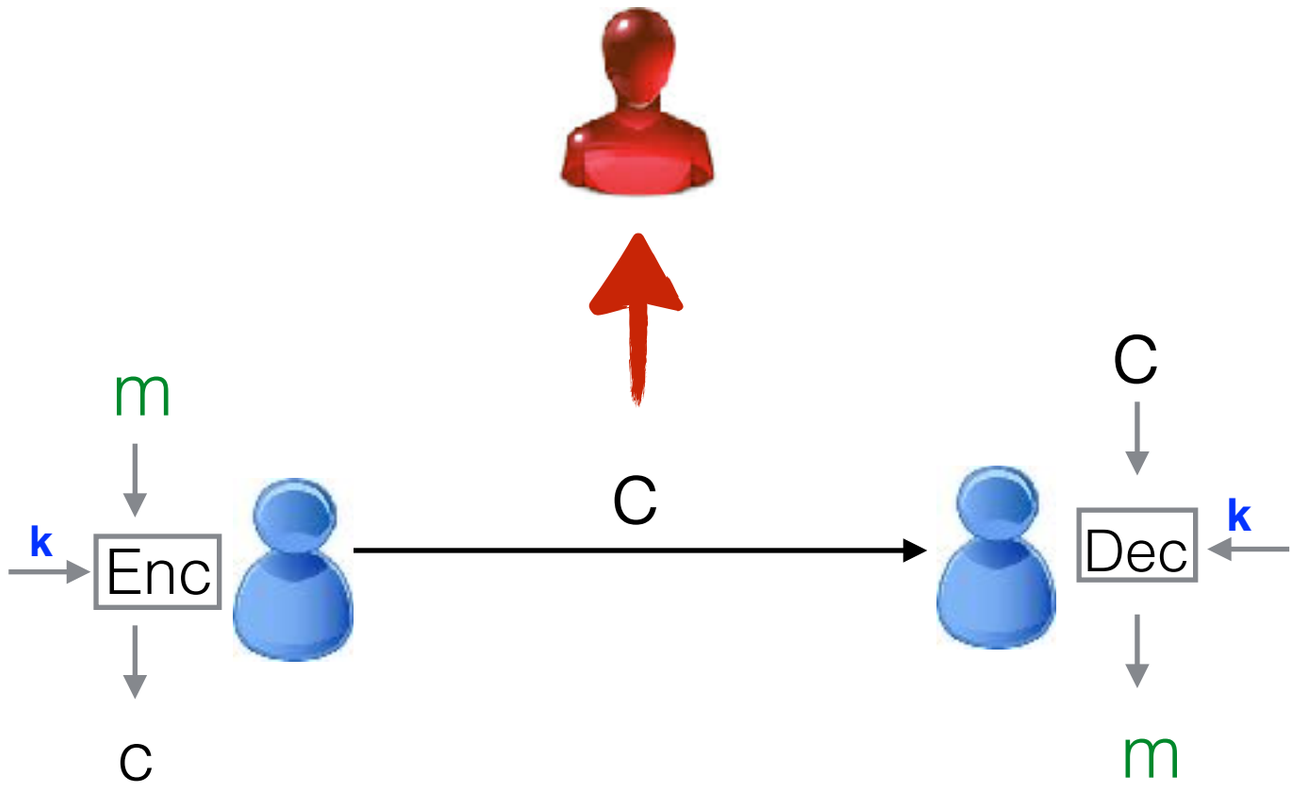
- Tutorial on Proofs
- Homework 1

So far…

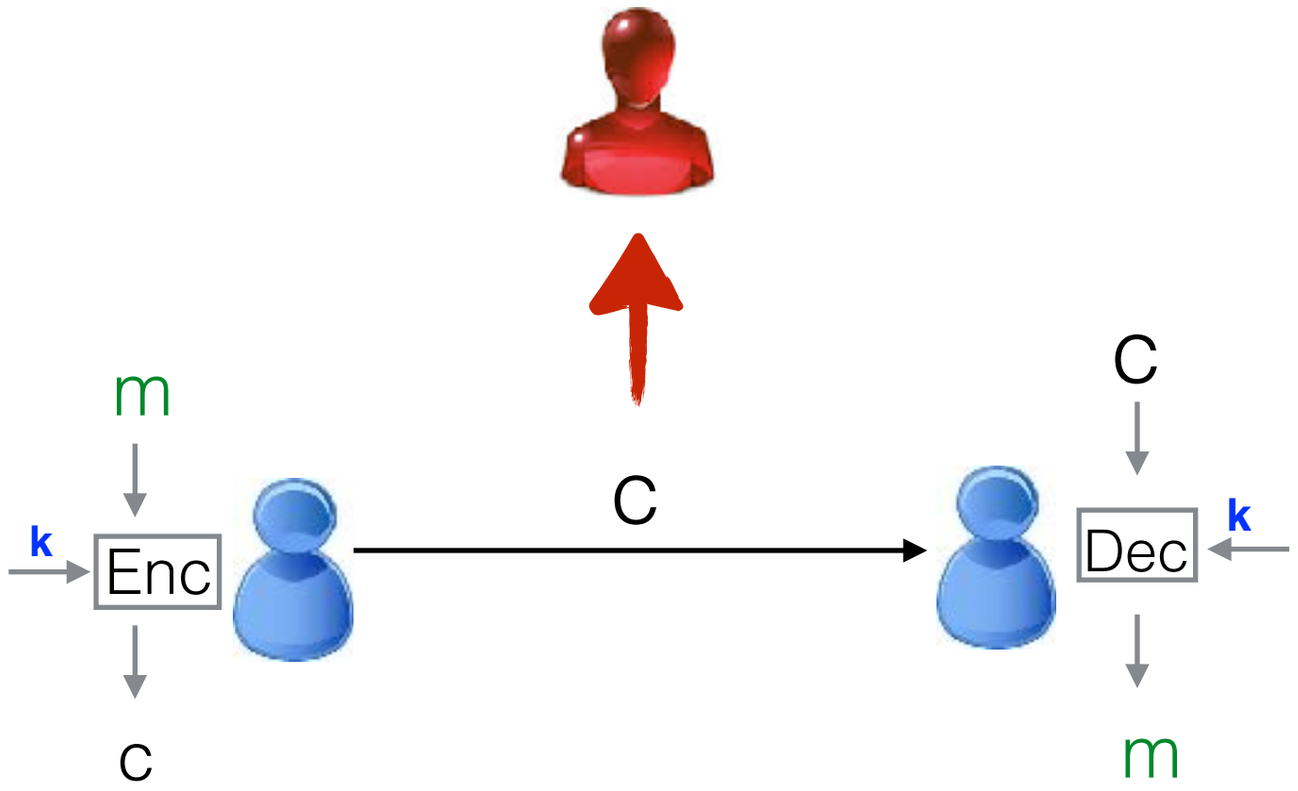


- Definition

- Assumptions

- Scheme/proof

So far…



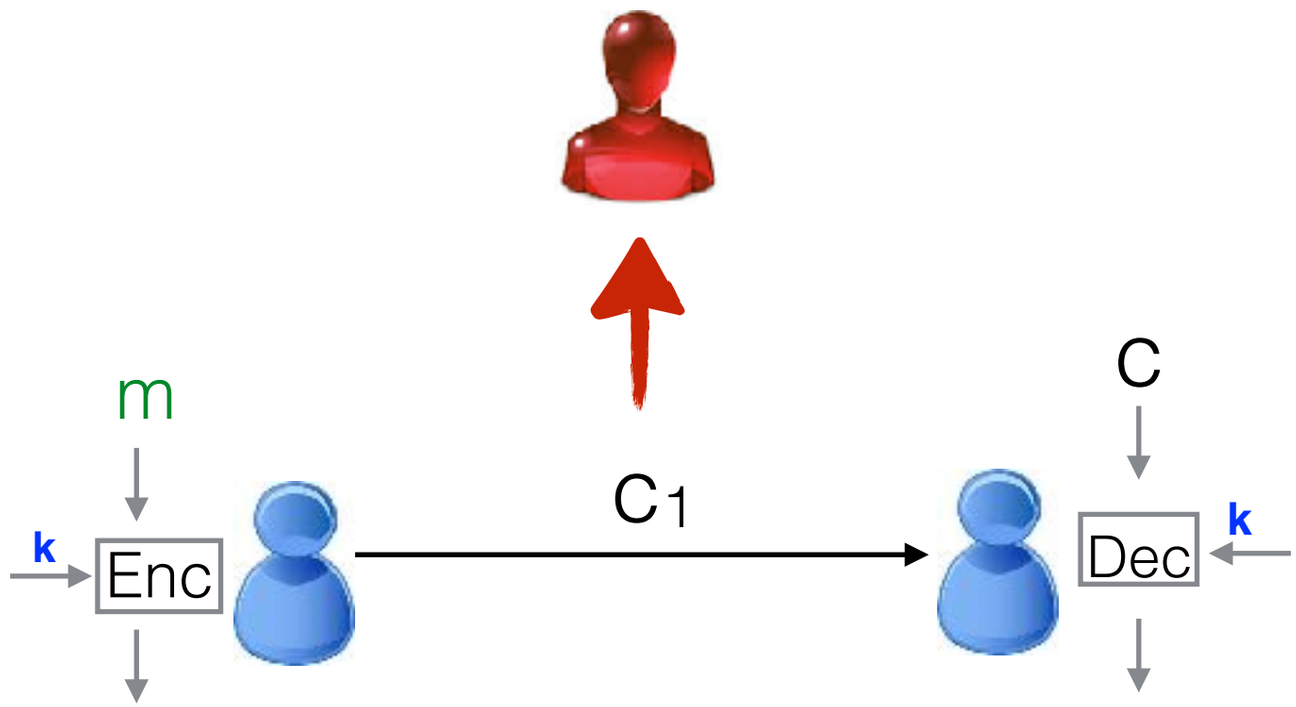
- **Definition**

  Security in presence of an eavesdropper of single cipher text

- **Assumptions**

- **Scheme/proof**

In the real world ….



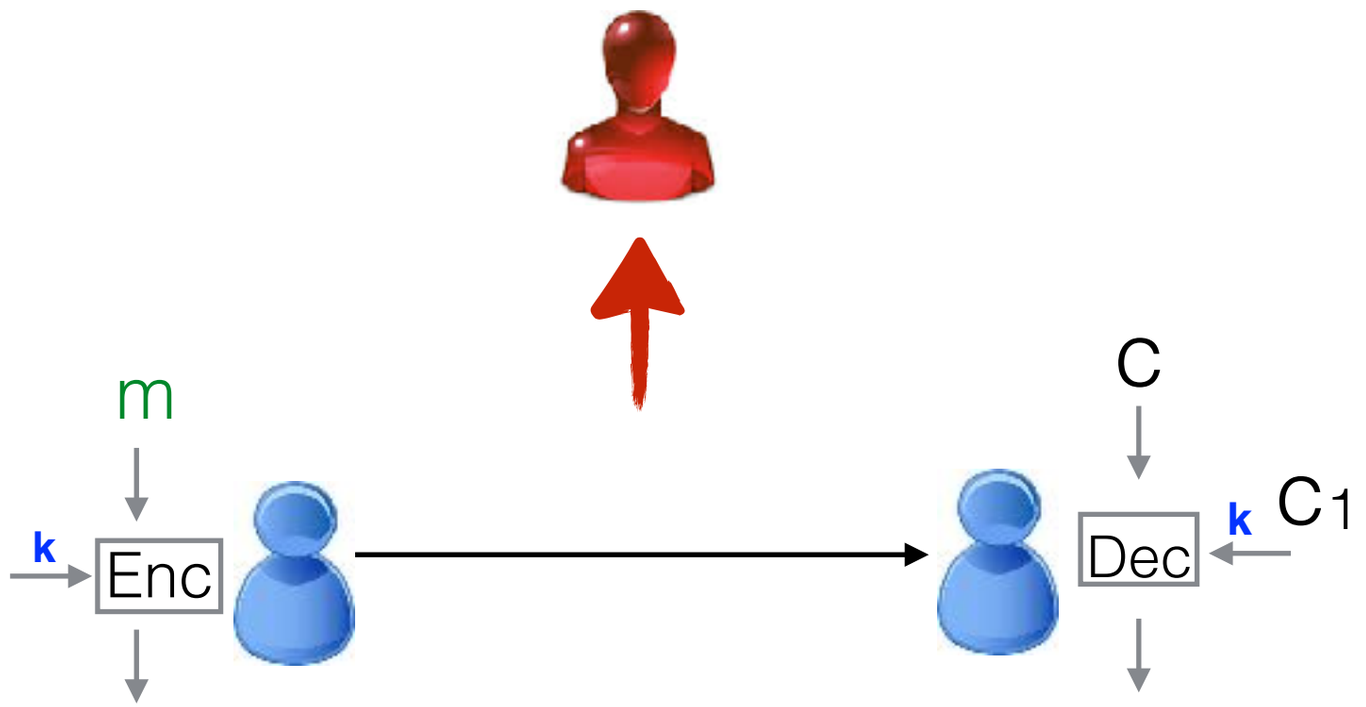
- **Definition**

  Security in presence of an eavesdropper of single cipher text

- **Assumptions**

- **Scheme/proof**

In the real world ….



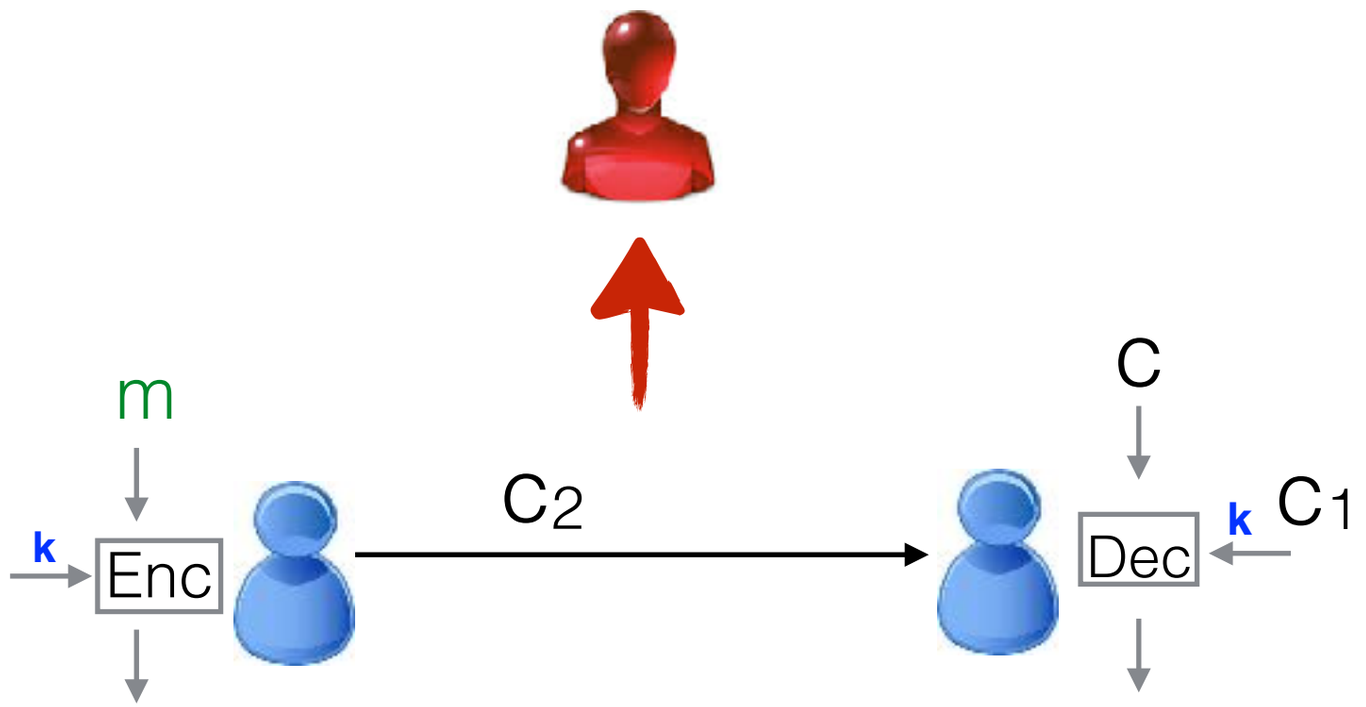
- **Definition**

  Security in presence of an eavesdropper of single cipher text

- **Assumptions**

- **Scheme/proof**

In the real world ….



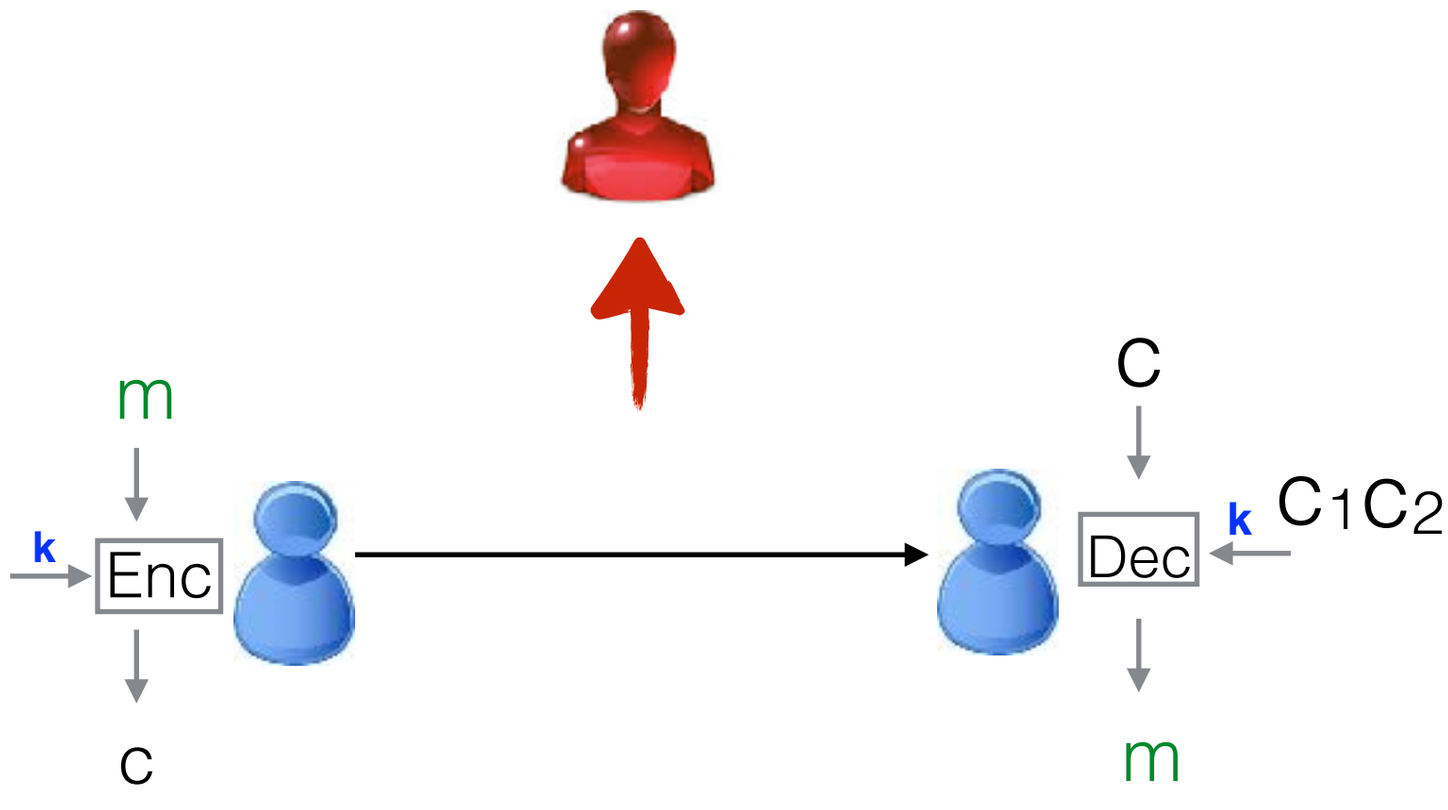
- **Definition**

  Security in presence of an eavesdropper of single cipher text

- **Assumptions**

- **Scheme/proof**

In the real world ….



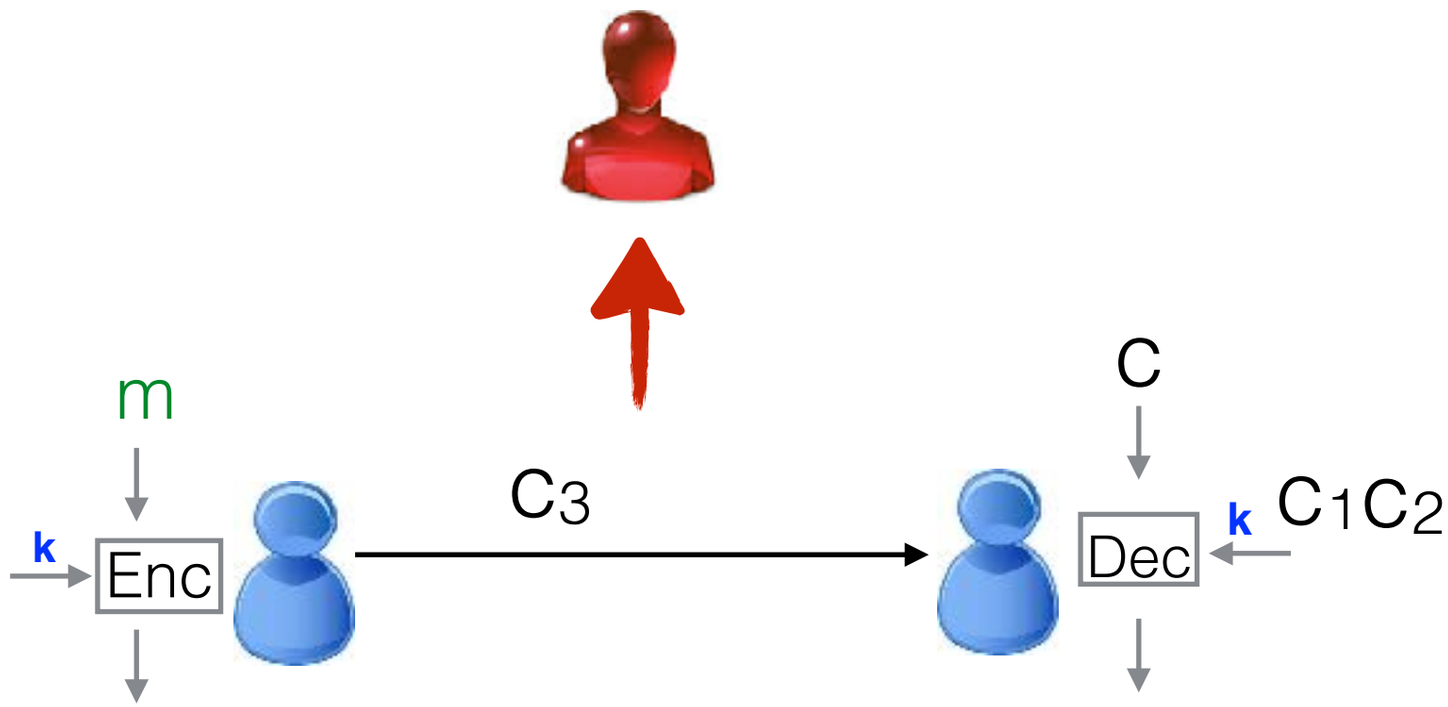
- **Definition**

  Security in presence of an eavesdropper of single cipher text

- **Assumptions**

- **Scheme/proof**

In the real world ….



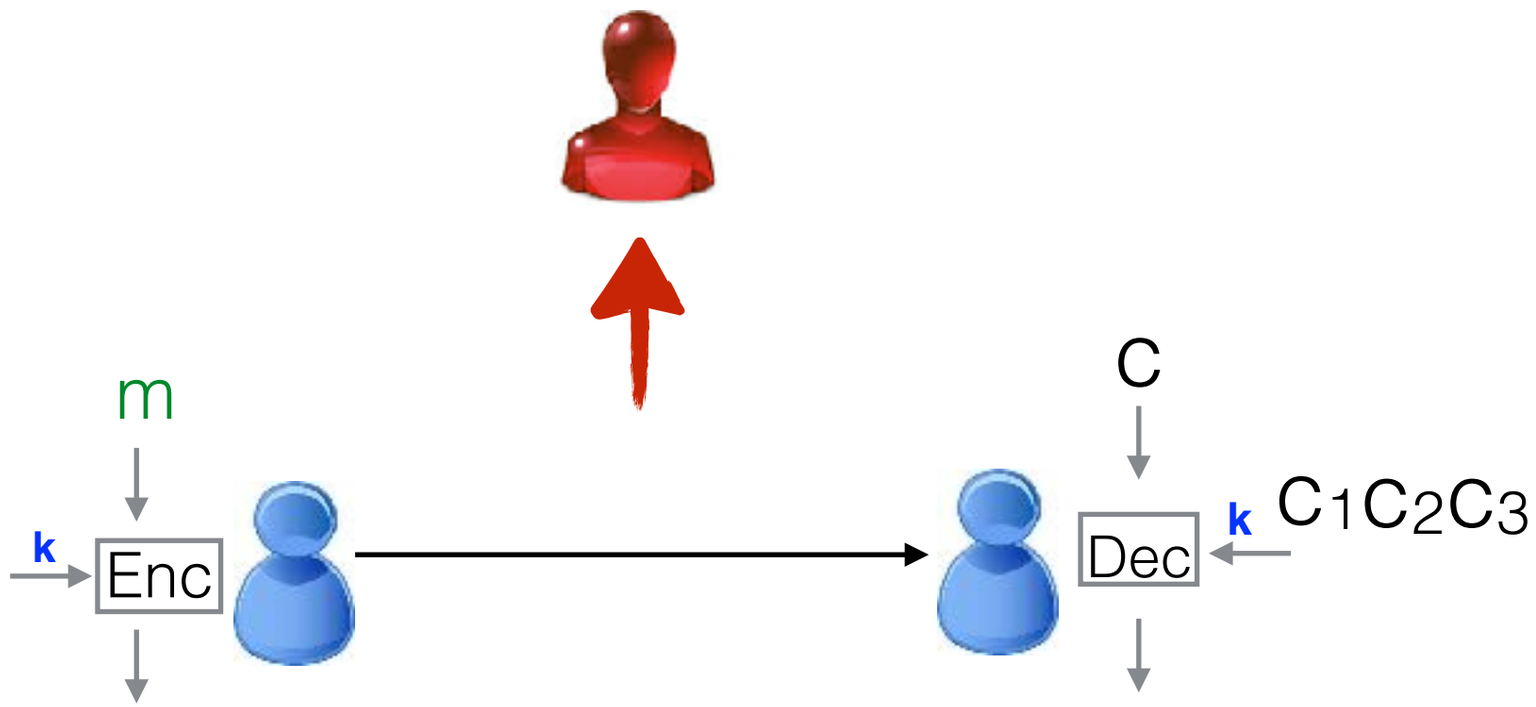
- **Definition**

  Security in presence of an eavesdropper of single cipher text

- **Assumptions**

- **Scheme/proof**

In the real world ….



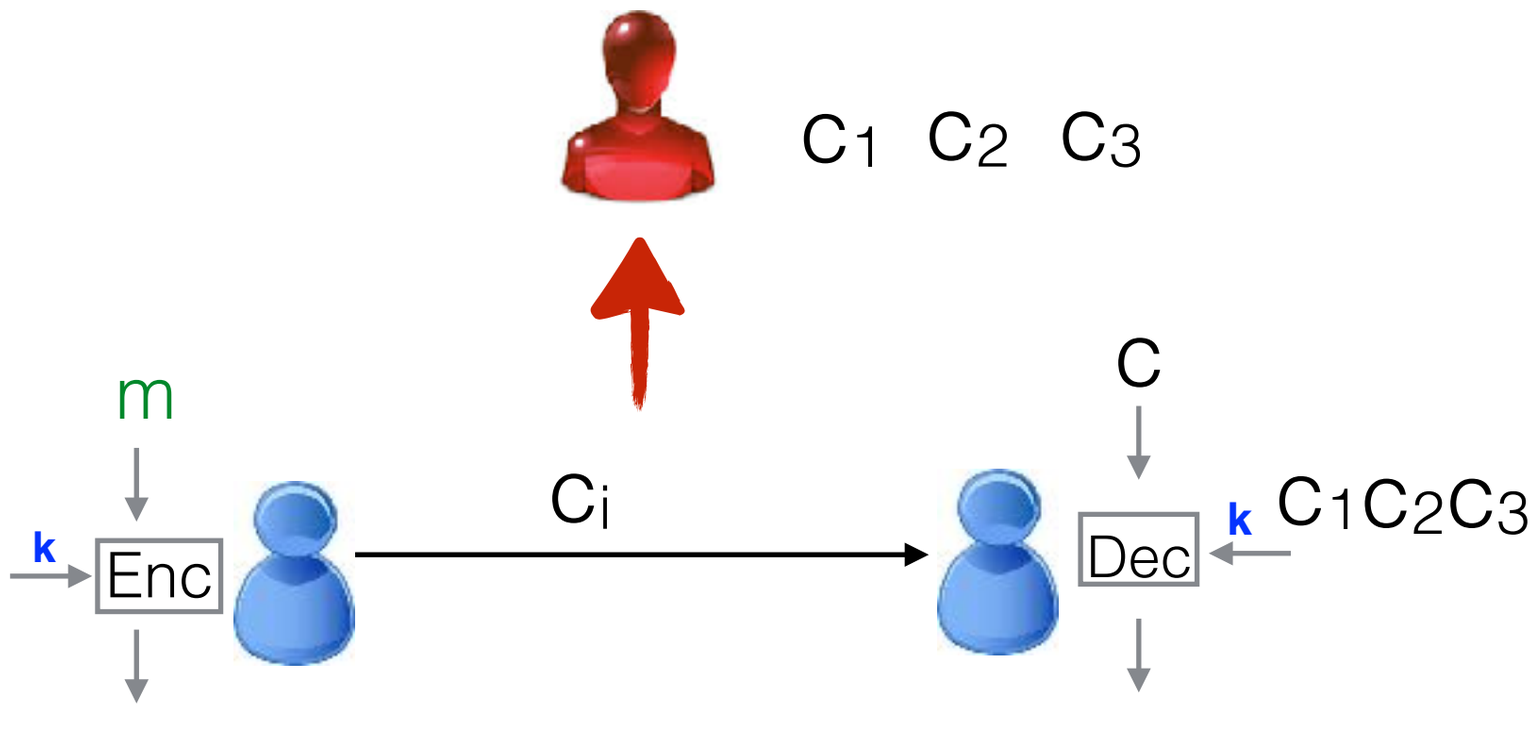
- **Definition**

  Security in presence of an eavesdropper of single cipher text

- **Assumptions**

- **Scheme/proof**

In the real world ….



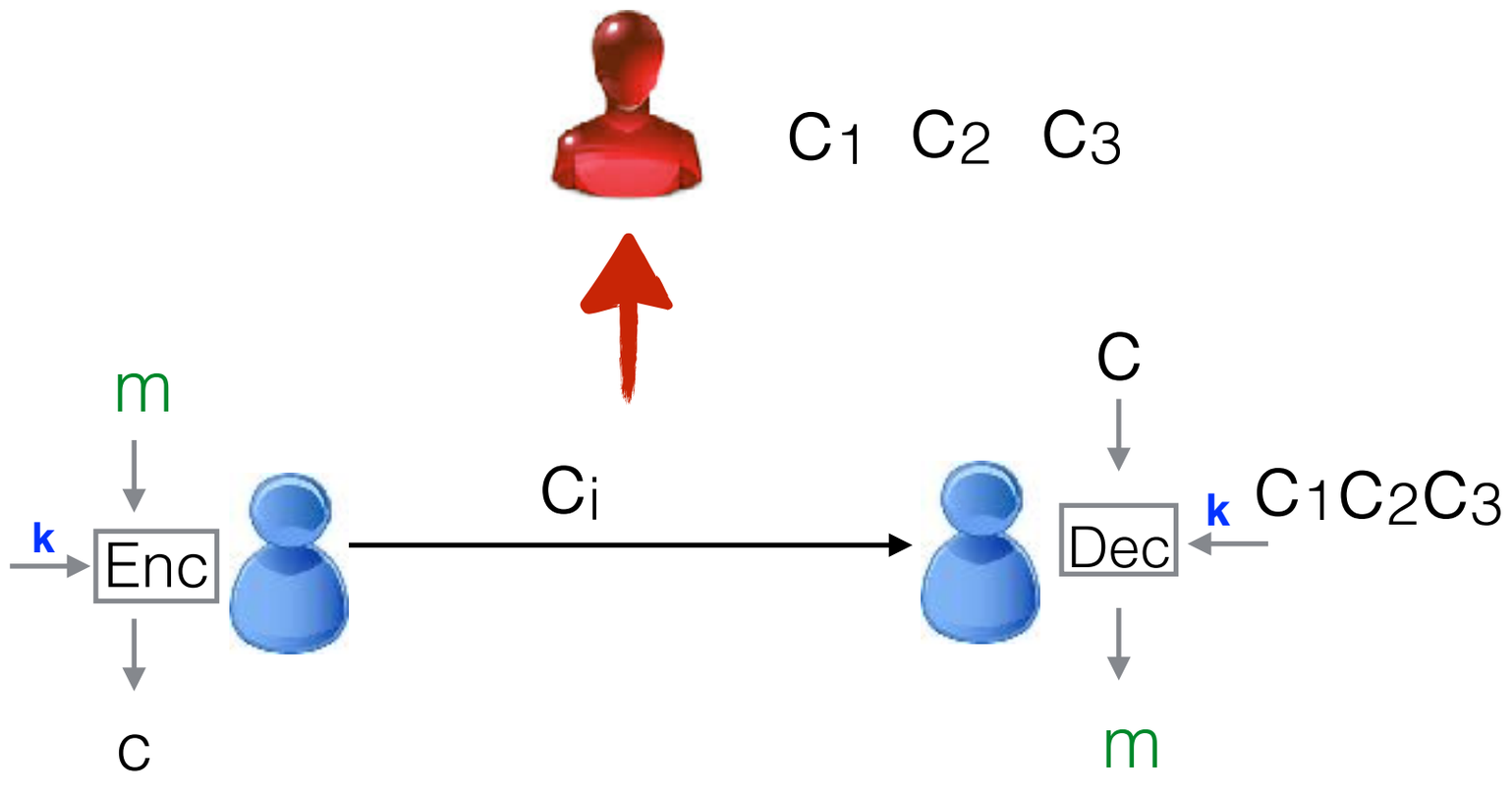
- **Definition**

  Security in presence of an eavesdropper of single cipher text

- **Assumptions**

- **Scheme/proof**

In the real world ….

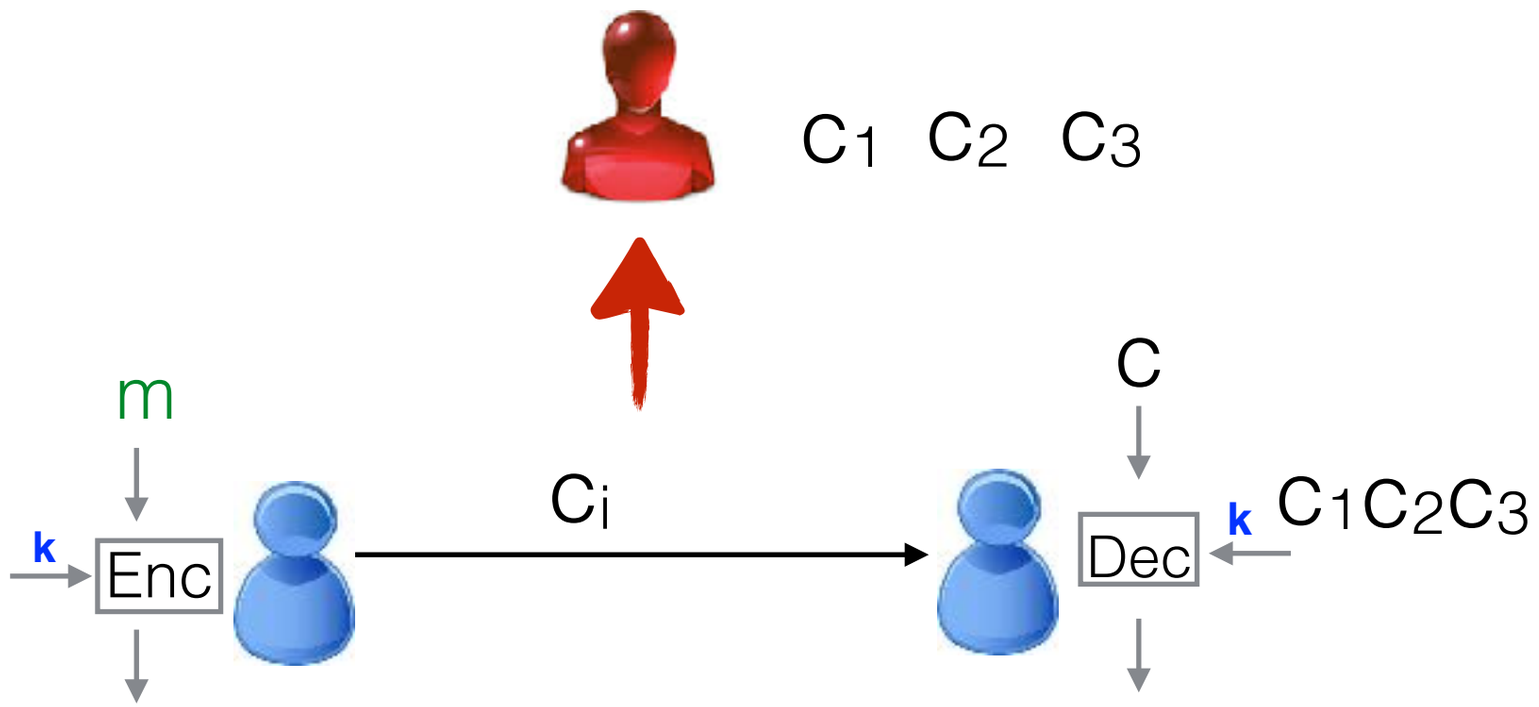


Security in presence of an eavesdropper of single cipher text

- **Assumptions**

- **Scheme/proof**

In the real world ….

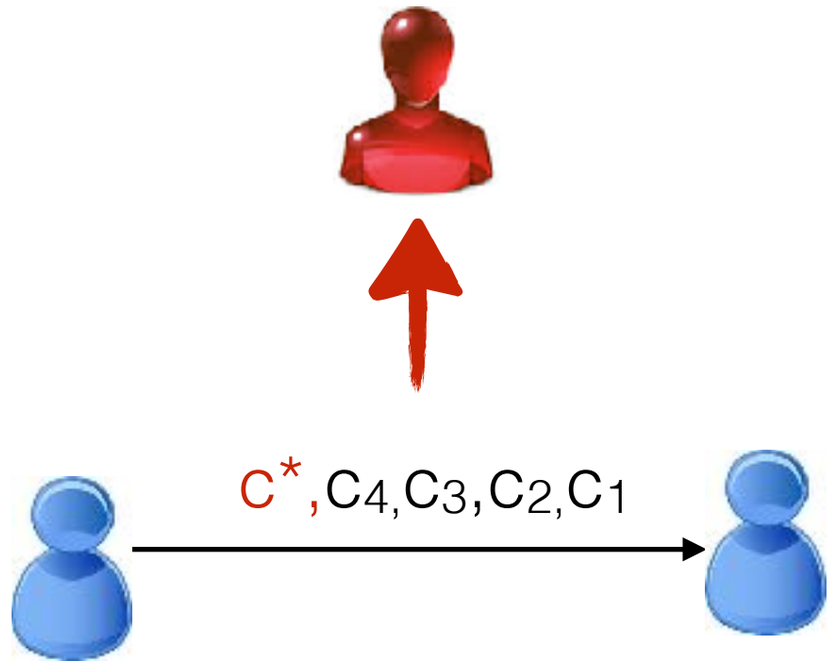


- Assumptions
- Scheme/proof

Today

- Definition
- Assumptions
- Scheme + Proofs!

new
realistic adversary



Today

- **Definition**

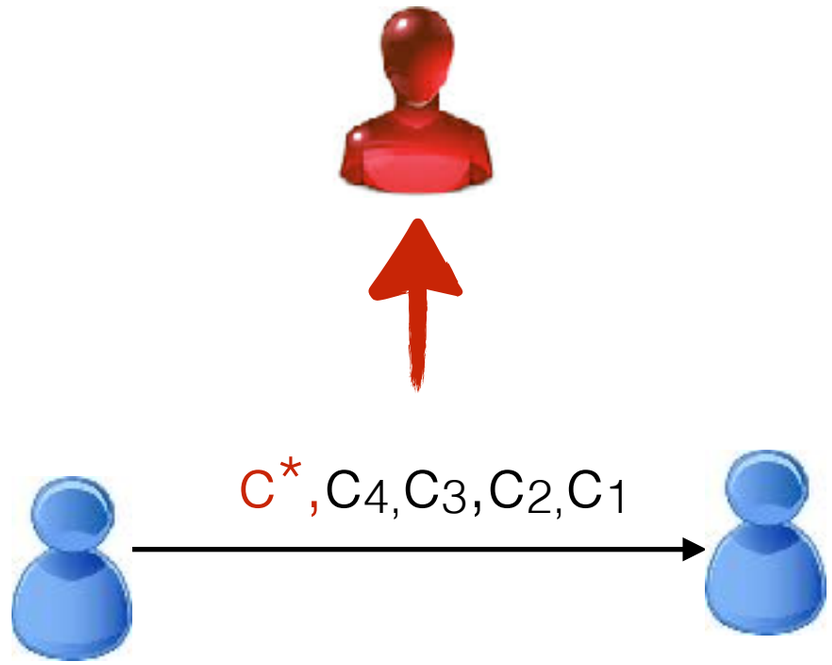
  Chosen Plaintext Attack

  (CPA) Security

- **Assumptions**

- **Scheme + Proofs!**

new
realistic adversary



Today

- **Definition**

  Chosen Plaintext Attack

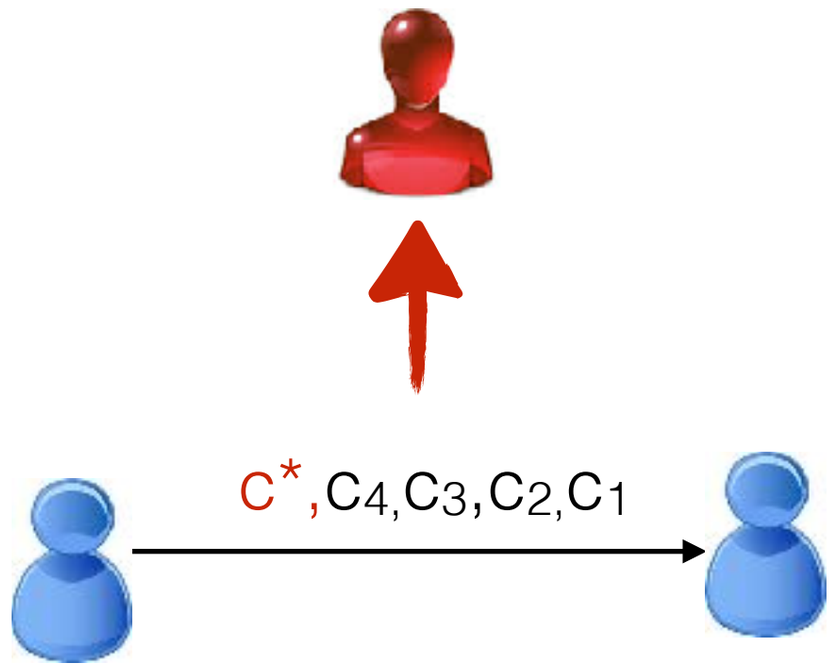
  (CPA) Security

- **Assumptions**

  Pseudorandom Functions

- **Scheme + Proofs!**

new
realistic adversary



Today

- **Definition**

  Chosen Plaintext Attack

  (CPA) Security

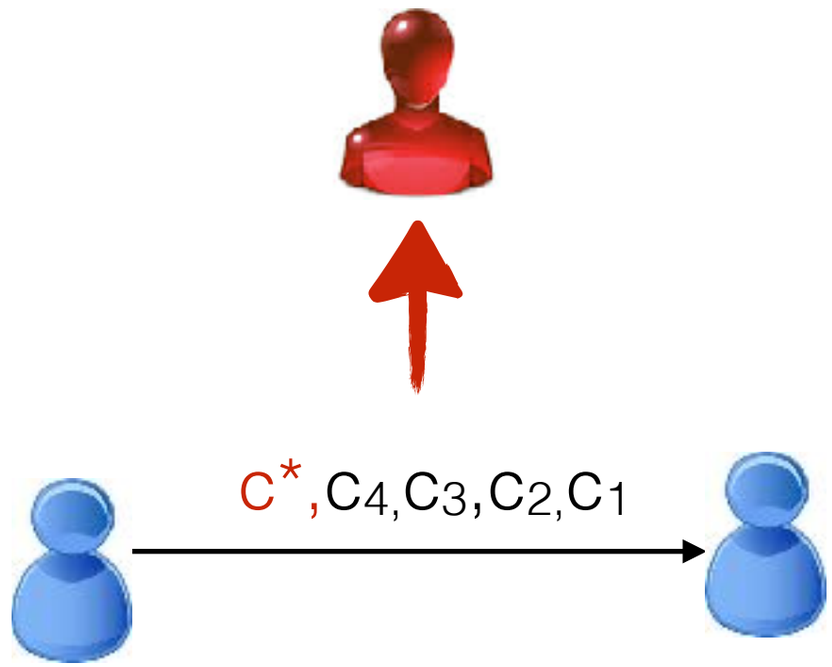
- **Assumptions**

  Pseudorandom Functions

  PRG —> PRF

- **Scheme + Proofs!**

new
realistic adversary



Today

- **Definition**

  Chosen Plaintext Attack

  (CPA) Security

- **Assumptions**

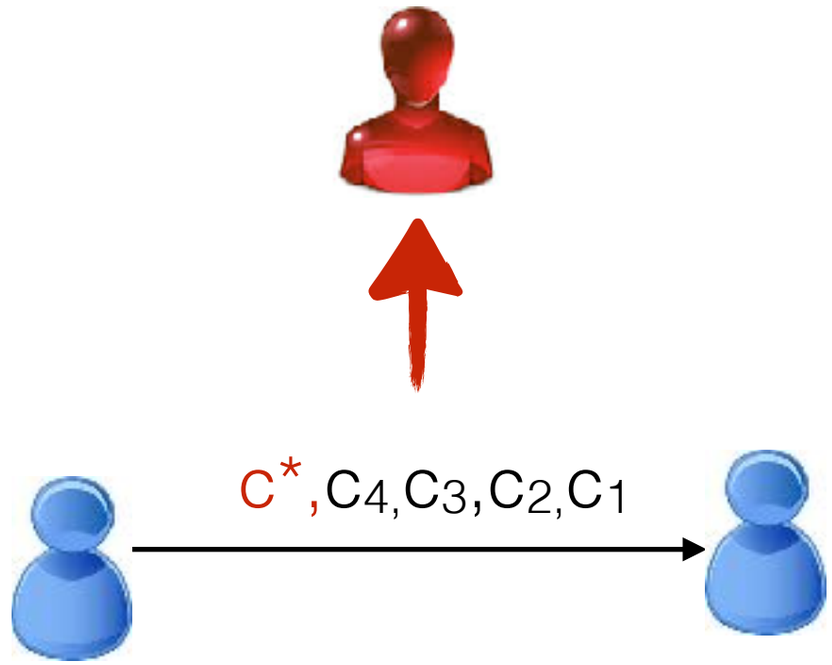
  Pseudorandom Functions

  PRG —> PRF

- **Scheme + Proofs!**

  Encryption scheme from PRF

new
realistic adversary



Today

- **Definition**

  Chosen Plaintext Attack

  (CPA) Security

- **Assumptions**

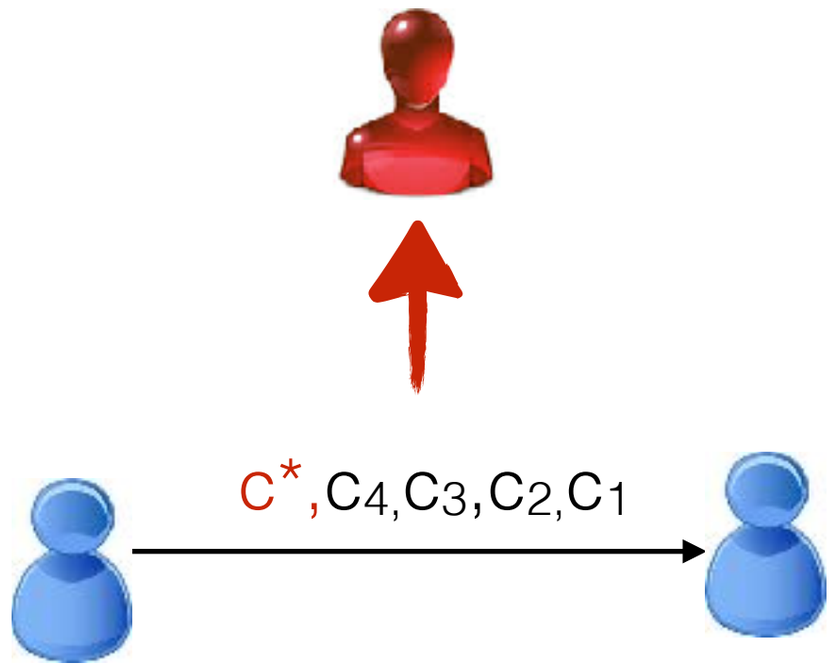
  Pseudorandom Functions

  PRG —> PRF

- **Scheme + Proofs!**

  Encryption scheme from PRF

new
realistic adversary

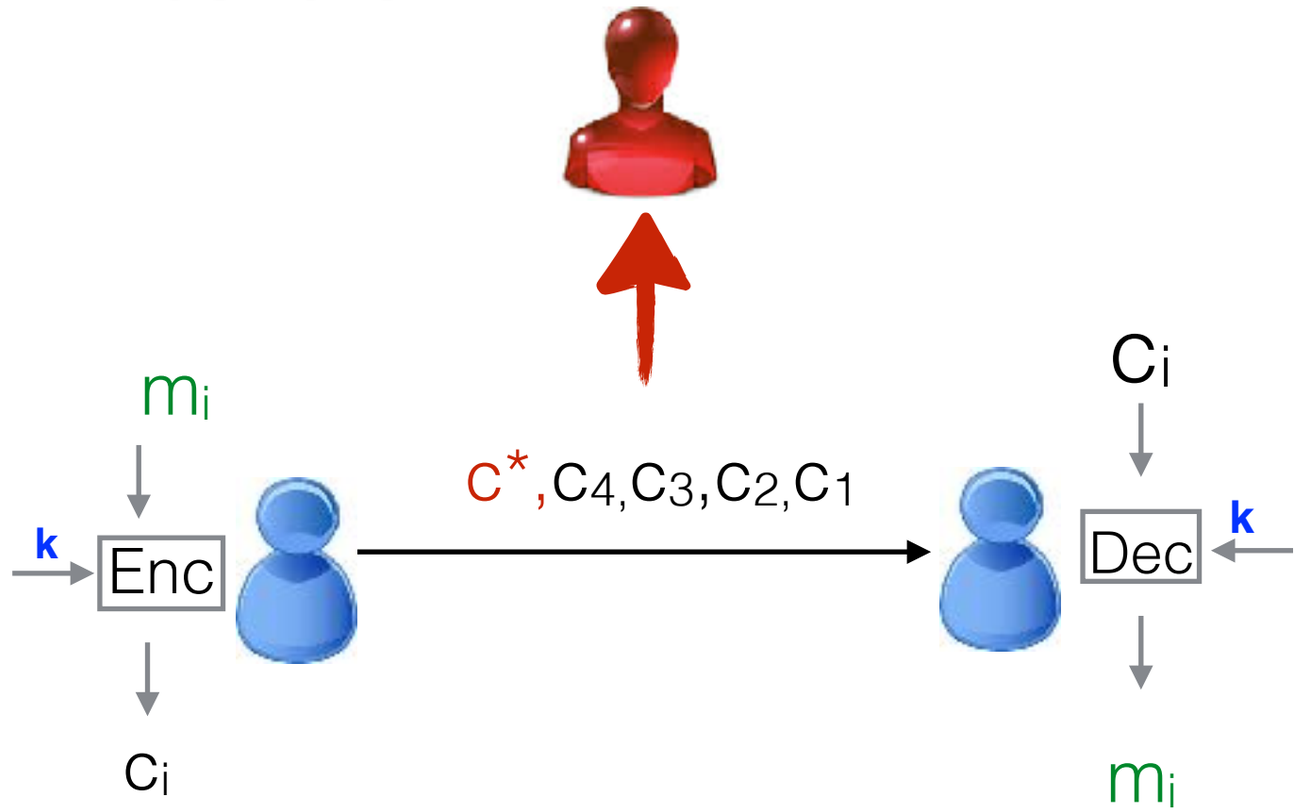


✦ **Definition**

Chosen Plaintext Attack

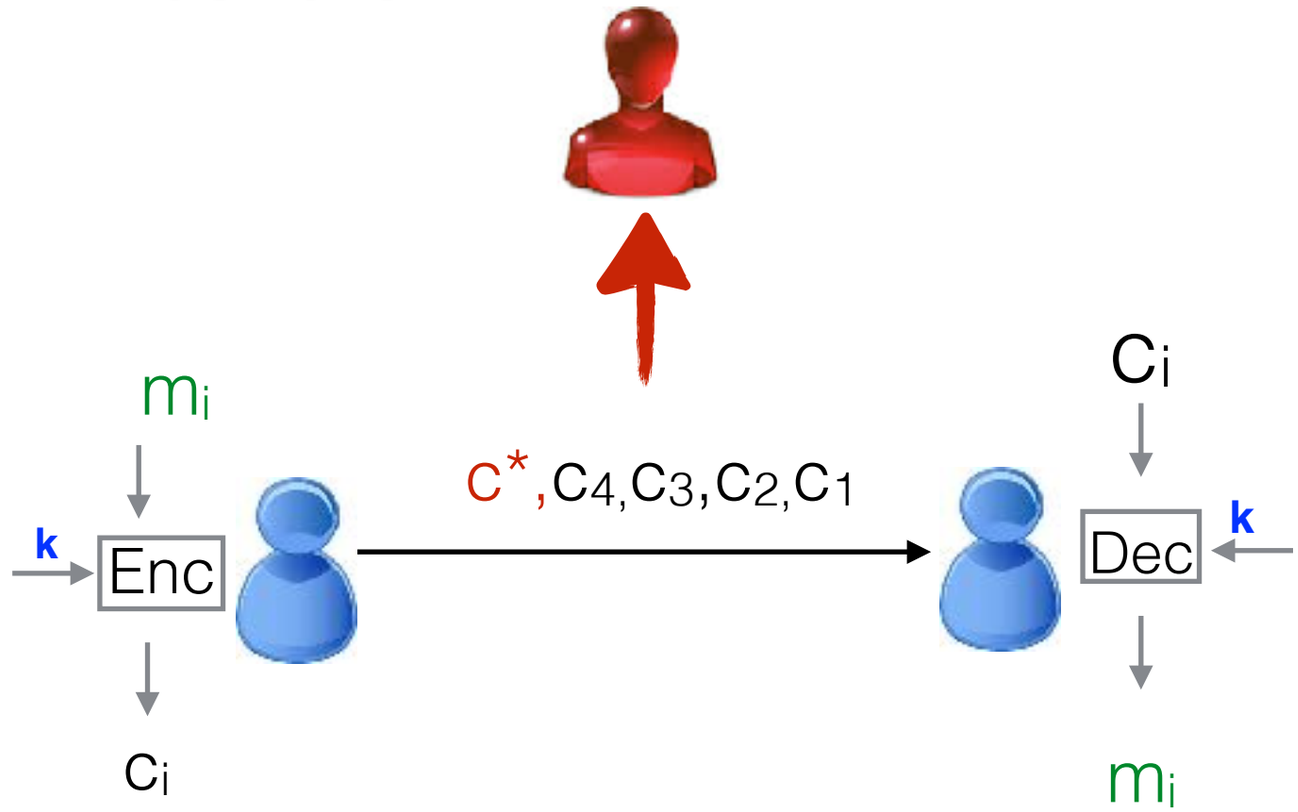
(CPA) Security

Definition: Intuition



Intuitive Definition:

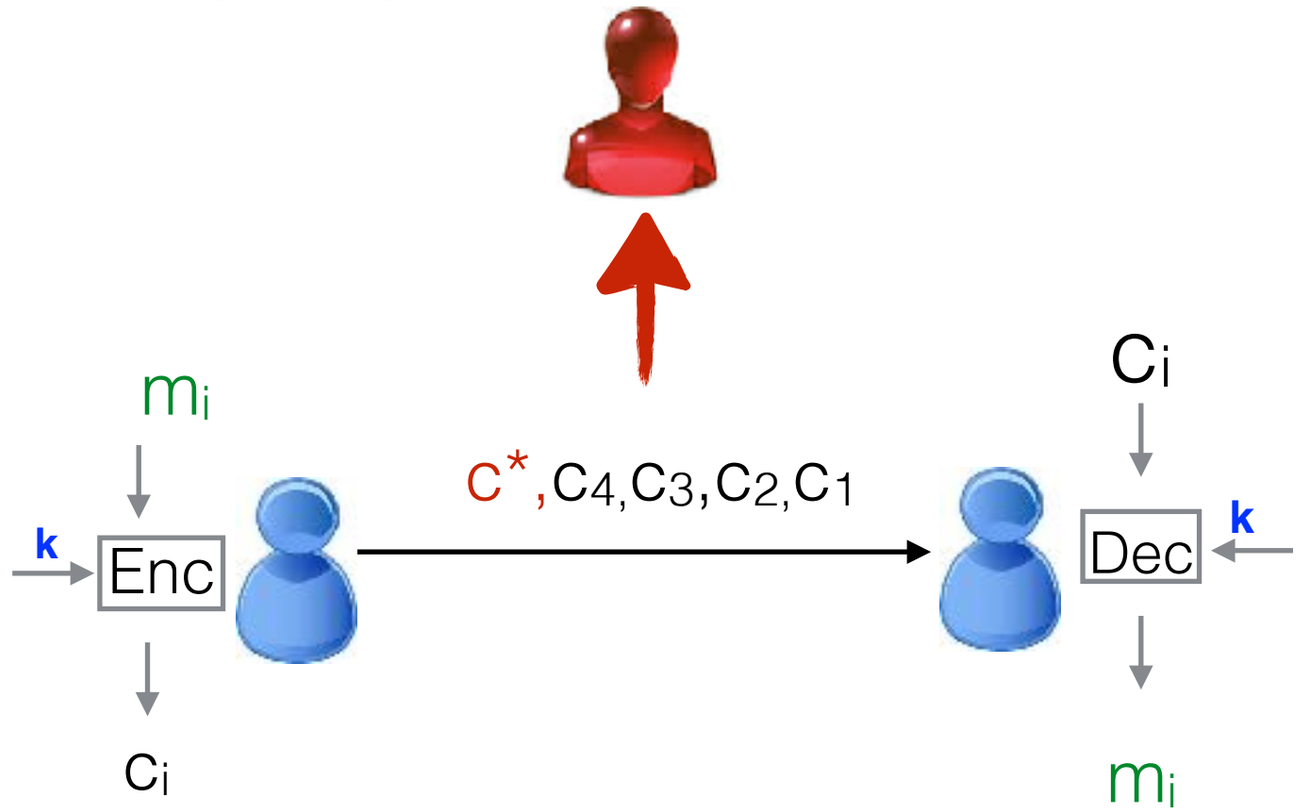
Definition: Intuition



Intuitive Definition:

the adversary does not learn any (additional) information about the message encrypted in **c***

Definition: Intuition

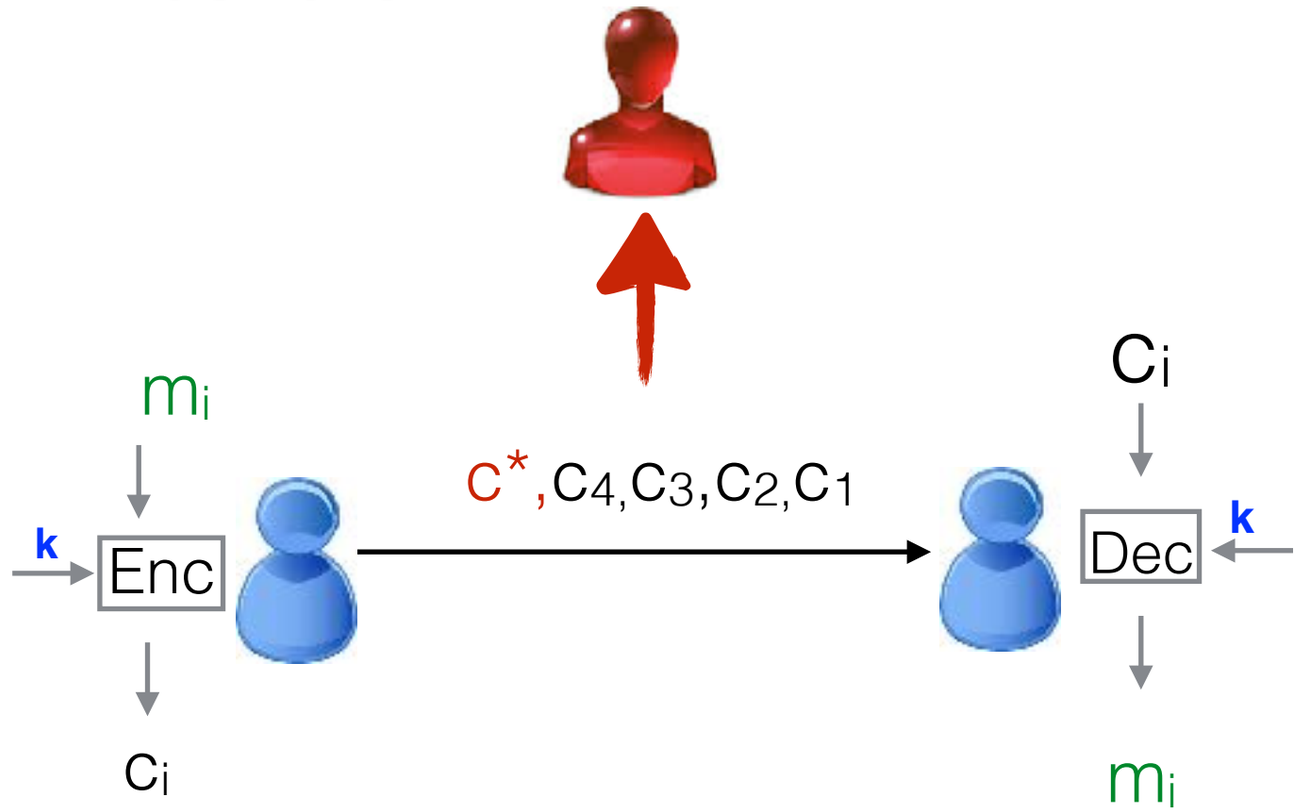


Intuitive Definition:

Even *after observing many ciphertexts c1, c2, c3…*

the adversary does not learn any (additional) information about the message encrypted in **c***

Definition: Intuition



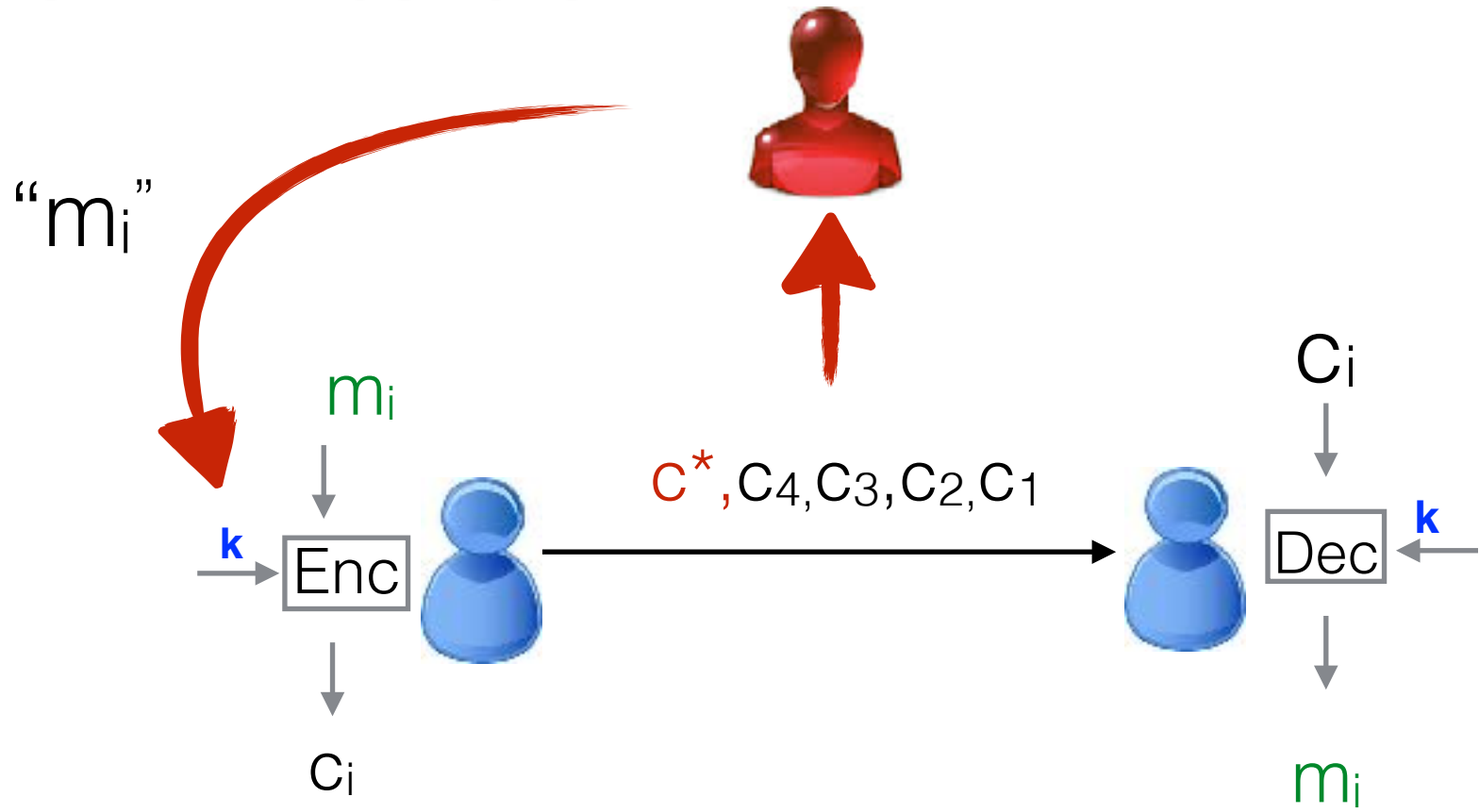
Intuitive Definition:

Even *after observing many ciphertexts c1, c2, c3…*

on messages of her choice!!!

the adversary does not learn any (additional) information about the message encrypted in **c***

Definition: Intuition



Intuitive Definition:

Even *after observing many ciphertexts c1, c2, c3…*

on messages of her choice!!!

the adversary does not learn any (additional) information about the message encrypted in **c***

Is this definition too strong?

Real world Attacks:

World War II

(a) US- Intelligence
(b) British Intelligence

Chosen Plaintex Attack:

Formal Definition

Eavesdropper

PrivK-eav

Eavesdropper

PrivK-eav

m_0 m_1

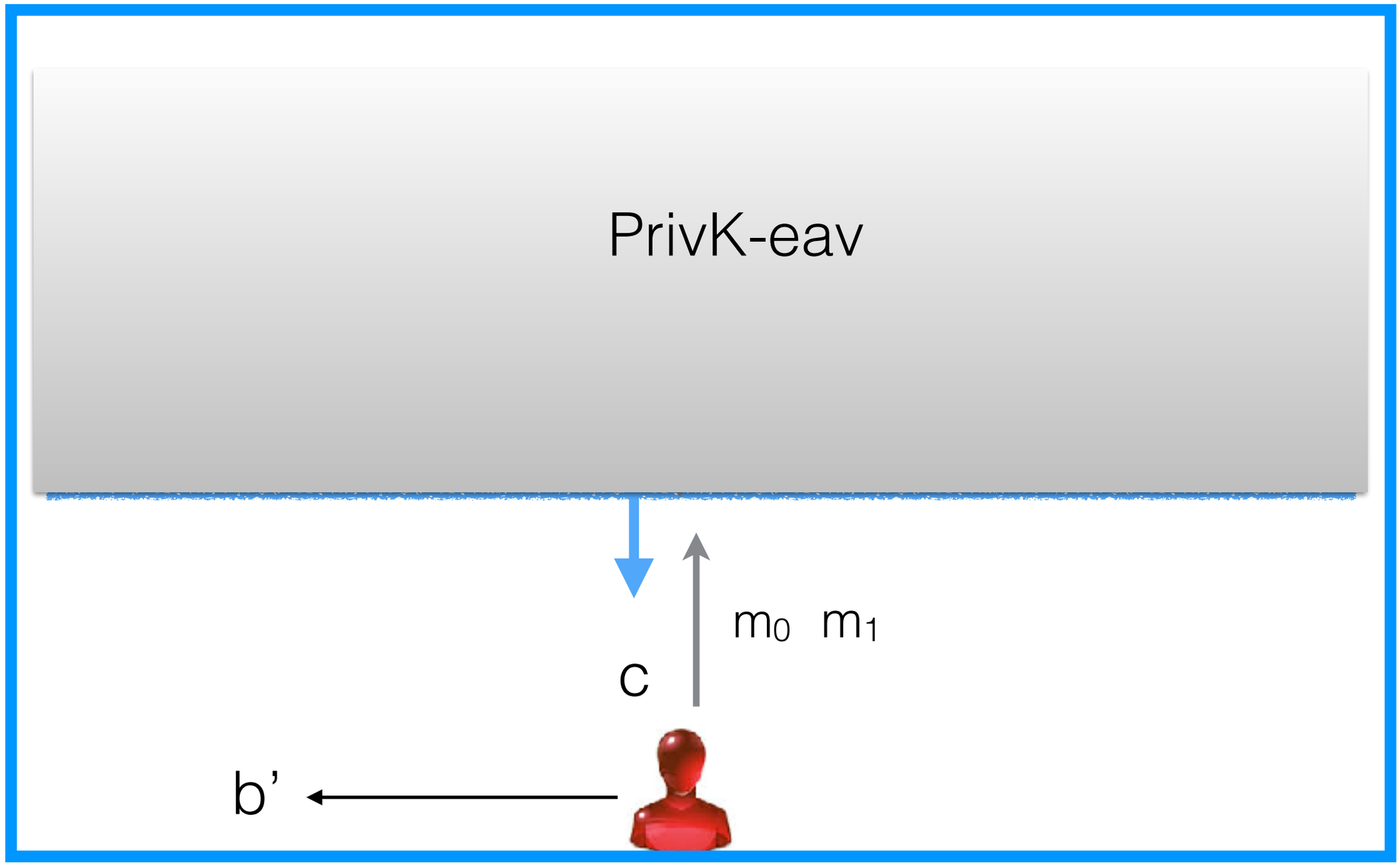
Eavesdropper

PrivK-eav

m_0 m_1

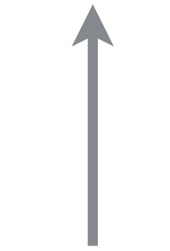
c

Eavesdropper



New Definition

PrivK -
CPA-Game



“training”

…

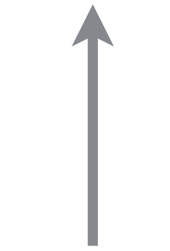
New Definition

PrivK -
CPA-Game

"training"

New Definition

PrivK -
CPA-Game



“training”

…

New Definition

PrivK -
CPA-Game

c_1

“training”

...

New Definition

PrivK -
CPA-Game

c_1

m_2

“training”

…

New Definition

PrivK -
CPA-Game

c_1

“training”

...

New Definition

PrivK -
CPA-Game

c_1
c_2

“training”

...

New Definition

PrivK -
CPA-Game

c_1
c_2

c_3 m_3

. . .

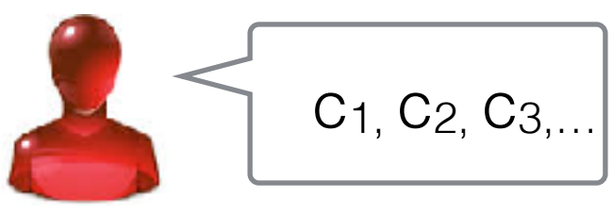
. .

"training"

New Definition

PrivK -
CPA-Game

"challenge"



New Definition

PrivK -
CPA-Game

m_0 m_1 "challenge"

$C_1, C_2, C_3, \ldots$

New Definition

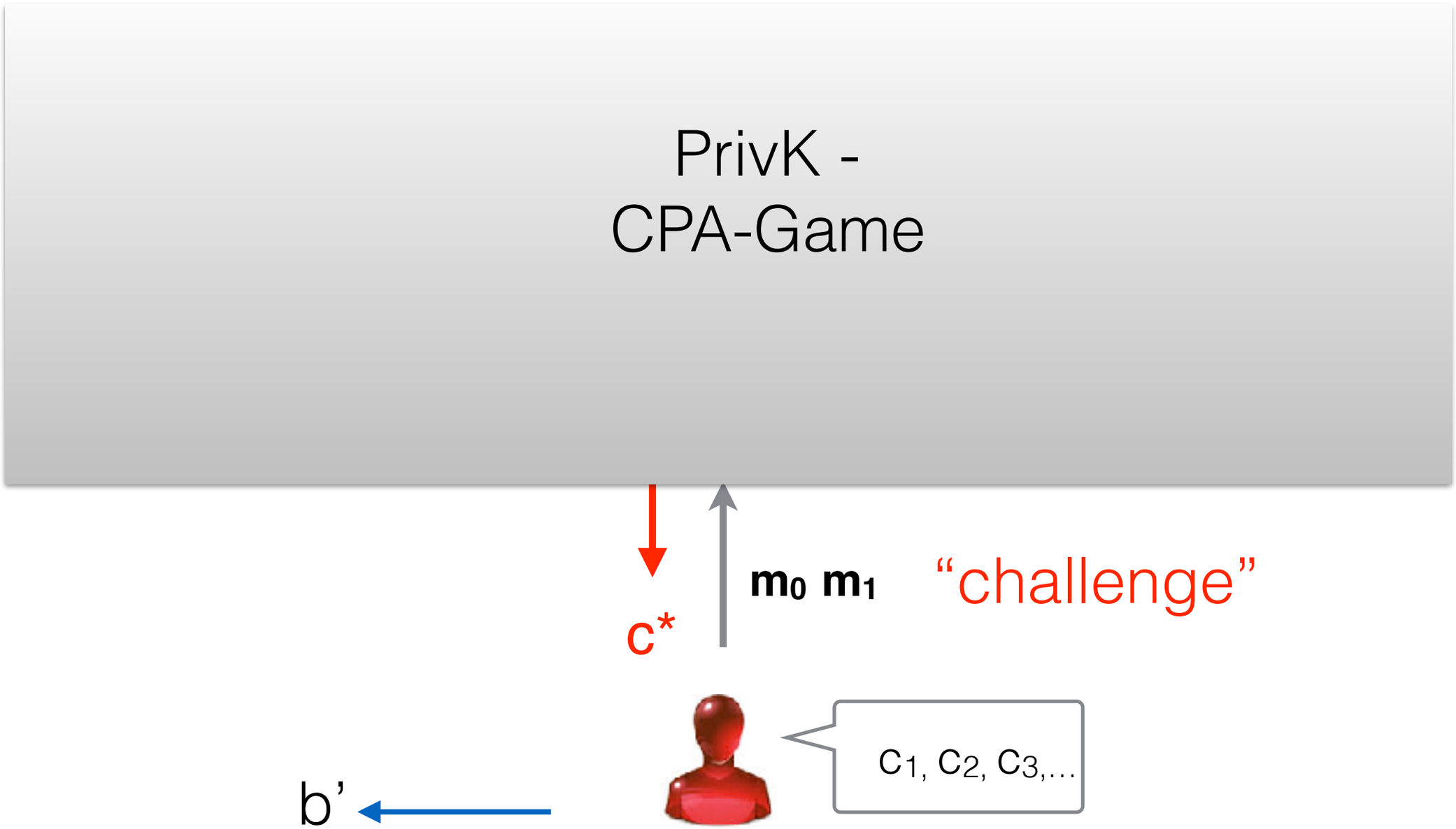
PrivK -
CPA-Game

m_0 m_1 “challenge”

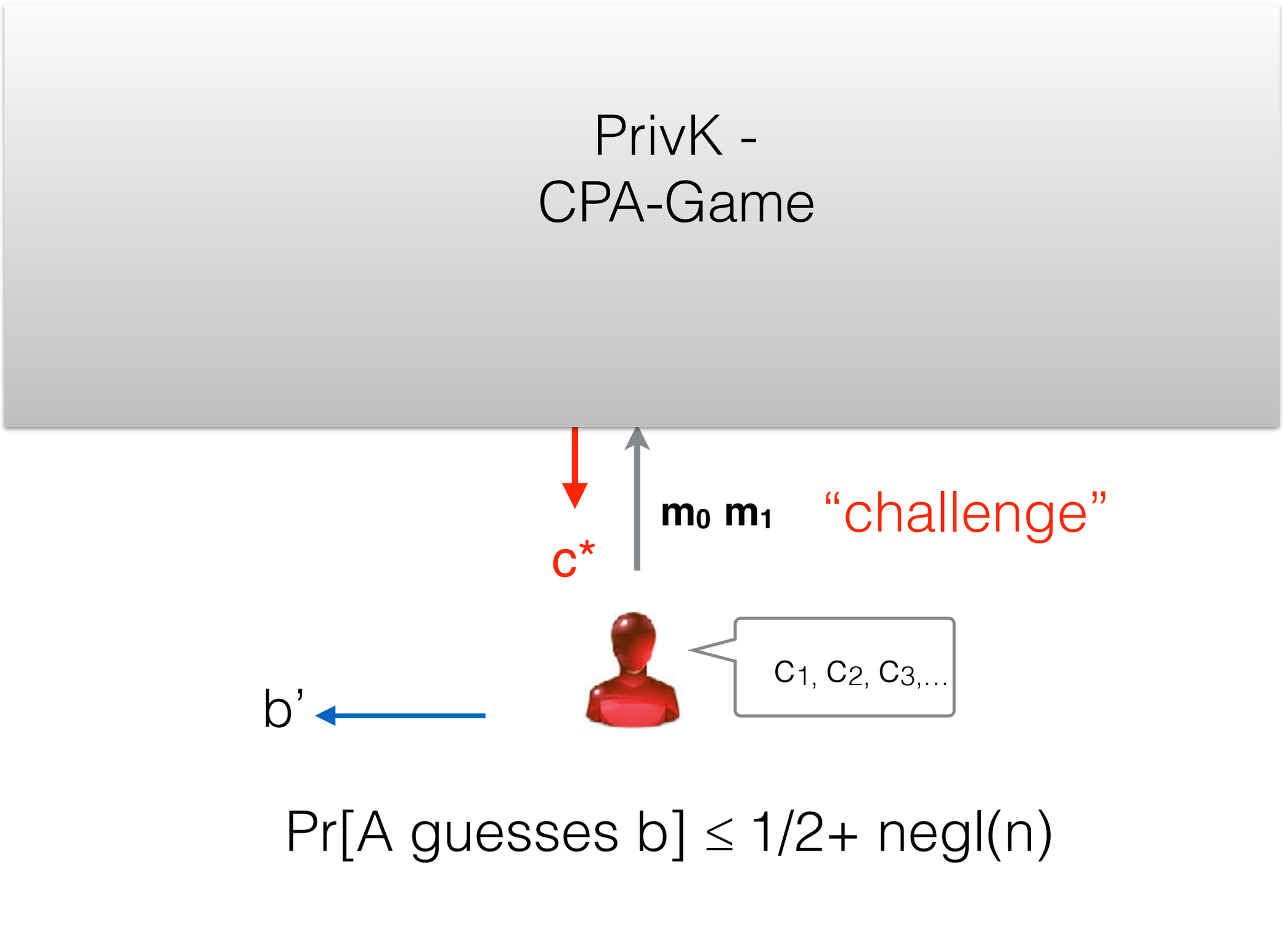
c^*

$c_1, c_2, c_3, \ldots$

New Definition

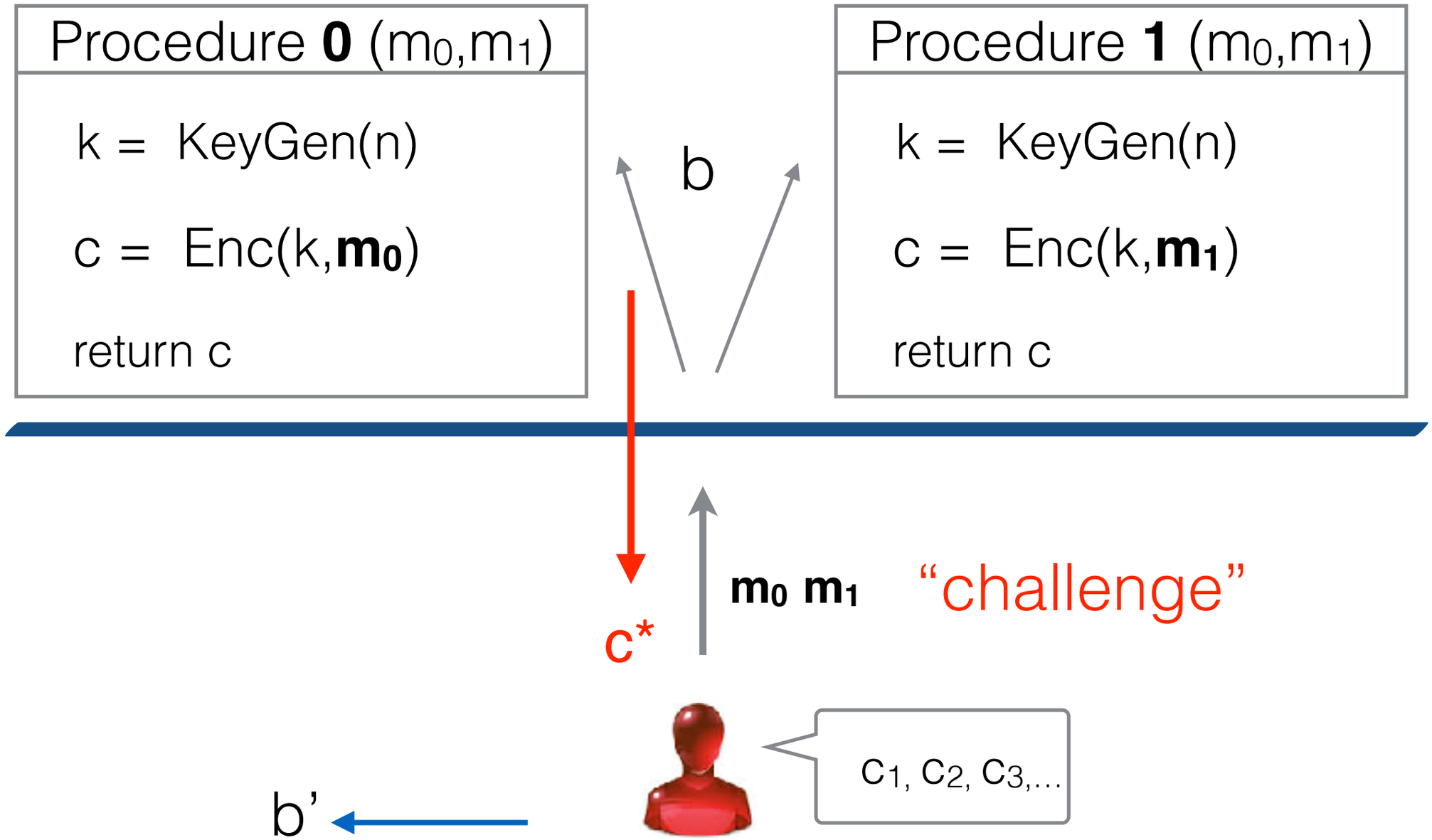


New Definition



Pr[A guesses b] ≤ 1/2+ negl(n)

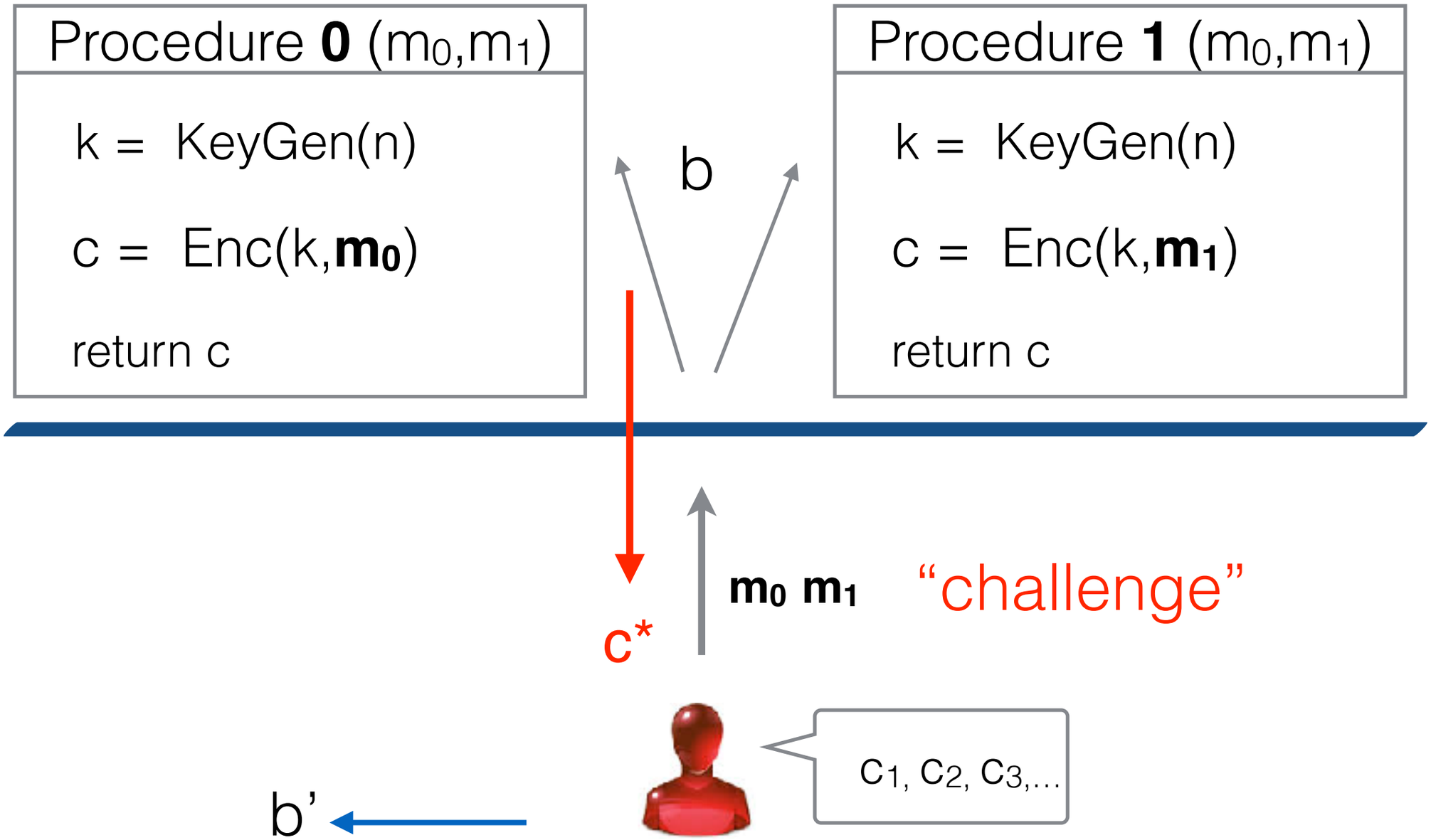
New Definition



Pr[A guesses b] ≤ 1/2+ negl(n)

Indistinguishable CPA-security =

Chosen Plaintext Attack



Pr[A guesses b] ≤ 1/2+ negl(n)

The CPA indistinguishability experiment $\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}(n)$**:**

1. *A random key k is generated by running* $\mathsf{Gen}(1^n)$.
2. *The adversary $\mathcal{A}$ is given input 1^n and oracle access to* $\mathsf{Enc}_k(\cdot)$, *and outputs a pair of messages m_0, m_1 of the same length.*
3. *A random bit $b \leftarrow \{0,1\}$ is chosen, and then a ciphertext $c \leftarrow \mathsf{Enc}_k(m_b)$ is computed and given to $\mathcal{A}$. We call c the* challenge ciphertext.
4. *The adversary $\mathcal{A}$ continues to have oracle access to* $\mathsf{Enc}_k(\cdot)$, *and outputs a bit b'.*
5. *The output of the experiment is defined to be 1 if $b' = b$, and 0 otherwise. (In case* $\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}(n) = 1$, *we say that $\mathcal{A}$* succeeded.)

Can we construct a scheme
that is secure under this definition?

Can we construct a scheme
that is secure under this definition?

Yes but it has to be randomized.

Deterministic Enc (k,)

input output

Randomized Enc (k,)

input output

Deterministic Enc (k,)

input

output

m_1

Randomized Enc (k,)

input

output

Deterministic Enc (k,)

input output

m_1 →

Randomized Enc (k,)

input output

Deterministic Enc (k,)

input output

m_1 → c_1

Randomized Enc (k,)

input output

Deterministic Enc (k,)

input output

m_1 → c_1

m_1

Randomized Enc (k,)

input output

Deterministic Enc (k,)

input output

m_1 → c_1

m_1 →

Randomized Enc (k,)

input output

Deterministic Enc (k,)

input output

m_1 → c_1

m_1 → c_1

Randomized Enc (k,)

input output

Deterministic Enc (k,)

input output

m_1 → c_1

m_1 → c_1

m_2

Randomized Enc (k,)

input output

Deterministic Enc (k,)

input output

m_1 → c_1

m_1 → c_1

m_2 →

Randomized Enc (k,)

input output

Deterministic Enc (k,)

input output

m_1 → c_1

m_1 → c_1

m_2 → c_2

Randomized Enc (k,)

input output

Deterministic Enc (k,)

input	output
m_1	c_1
m_1	c_1
m_2	c_2
..	..
..	..

Randomized Enc (k,)

input	output

Deterministic Enc (k,)

input		output
m_1	→	c_1
m_1	→	c_1
m_2	→	c_2
..		..
..		..

Randomized Enc (k,)

input	output
m_1	

Deterministic Enc (k,)

input		output
m_1	→	c_1
m_1	→	c_1
m_2	→	c_2
..		..
..		..

Randomized Enc (k,)

input		output
m_1	→	

Deterministic Enc (k,)

input	output
m_1	c_1
m_1	c_1
m_2	c_2
..	..
..	..

Randomized Enc (k,)

input	output
m_1	y

Deterministic Enc (k,)

input	output
m_1	c_1
m_1	c_1
m_2	c_2
..	..
..	..

Randomized Enc (k,)

input	output
m_1	y
m_1	

Deterministic Enc (k,)

input	output
m_1	c_1
m_1	c_1
m_2	c_2
..	..
..	..

Randomized Enc (k,)

input	output
m_1	y
m_1	

Deterministic Enc (k,)

input		output
m_1	→	c_1
m_1	→	c_1
m_2	→	c_2
..		..
..		..

Randomized Enc (k,)

input		output
m_1	→	y
m_1	→	w

Deterministic Enc (k,)

input		output
m_1	→	c_1
m_1	→	c_1
m_2	→	c_2
..		..
..		..

Randomized Enc (k,)

input		output
m_1	→	y
m_1	→	w
m_2		

Deterministic Enc (k,)

input	output
m_1	c_1
m_1	c_1
m_2	c_2
..	..
..	..

Randomized Enc (k,)

input	output
m_1	y
m_1	w
m_2	

Deterministic Enc (k,)

input		output
m_1	→	c_1
m_1	→	c_1
m_2	→	c_2
..		..
..		..

Randomized Enc (k,)

input		output
m_1	→	y
m_1	→	w
m_2	→	z

Deterministic Enc (k,)

input		output
m_1	→	c_1
m_1	→	c_1
m_2	→	c_2
..		..
..		..

Randomized Enc (k,)

input		output
m_1	→	y
m_1	→	w
m_2	→	z
..		..
..		..

Deterministic Enc (k,)

input	output
m_1	c_1
m_1	c_1
m_2	c_2
..	..
..	..

Deterministic:
depends only on m and k

Randomized Enc (k,)

input	output
m_1	y
m_1	w
m_2	z
..	..
..	..

Randomized:
depends on m and k and fresh random values

In class exercise.

Theorem. If an encryption scheme has a deterministic* Enc function, then it cannot be CPA-secure

In class exercise.

Theorem. If an encryption scheme has a deterministic* Enc function, then it cannot be CPA-secure

---

- What does it mean that Enc is deterministic?

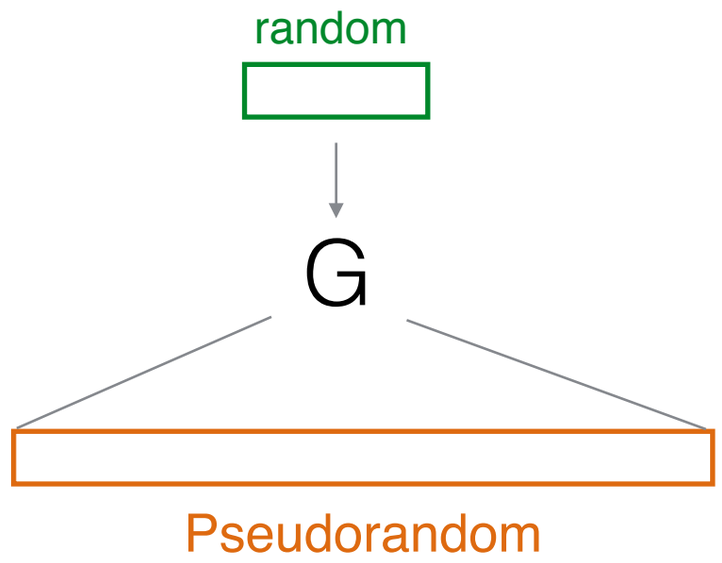
- How can a CPA-adversary exploit that to win the game?

Attack

How to construct a CPA-secure Encryption scheme?

Pseudo - OTP

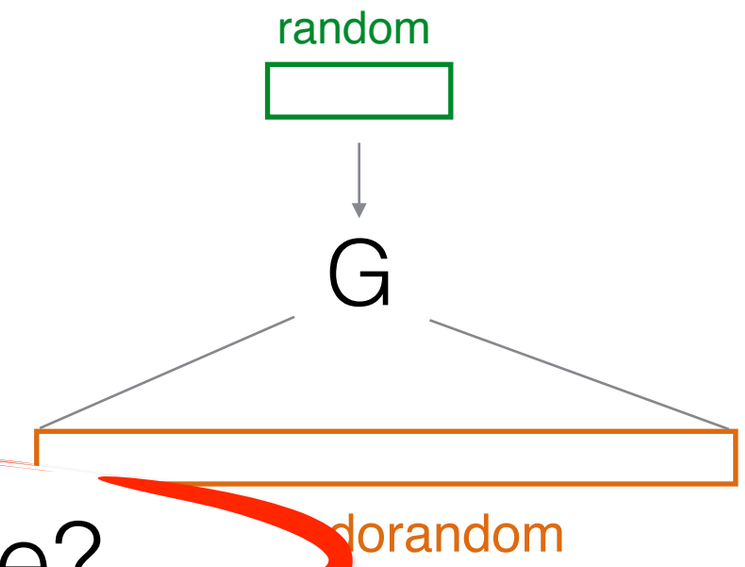
$G : \{0,1\}^n \rightarrow \{0,1\}^{l(n)}$



Gen (n)	Enc (m, k)	Dec (c,k)
$s \leftarrow \{0,1\}^n$		
$k \leftarrow G(s)$	$c = m \oplus k$	$m = c \oplus k$

Pseudo - OTP

$G : \{0,1\}^n \rightarrow \{0,1\}^{l(n)}$

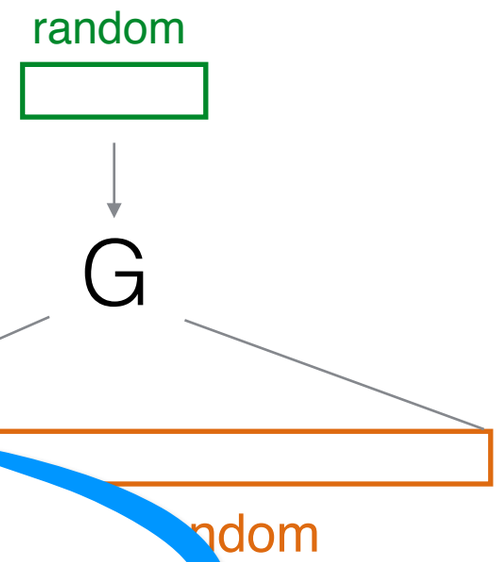


Is Pseudo-OTP CPA-secure?

Gen (n)	Enc (m, k)	Dec (c,k)
$s \leftarrow \{0,1\}^n$		
$k \leftarrow G(s)$	$c = m \oplus k$	$m = c \oplus k$

Pseudo - OTP

$G : \{0,1\}^n \rightarrow \{0,1\}^{l(n)}$



How can we make Pseudo-OTP CPA -Secure?

Gen (n)	Enc (m, k)	Dec (c,k)
$s \leftarrow \{0,1\}^n$		
$k \leftarrow G(s)$	$c = m \oplus k$	$m = c \oplus k$

Today

- **Definition**

  Chosen Plaintext Attack

  (CPA) Security

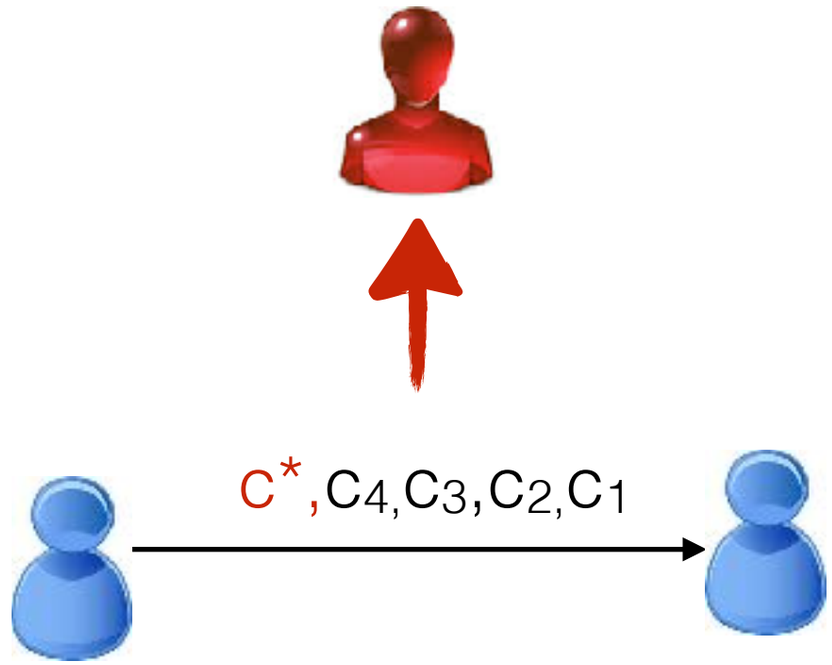
- **Assumptions**

  Pseudorandom Functions

- **Scheme + Proofs!**

  Encryption scheme from PRF

new
realistic adversary



Today

- **Definition**

  Chosen Plaintext Attack

  (CPA) Security

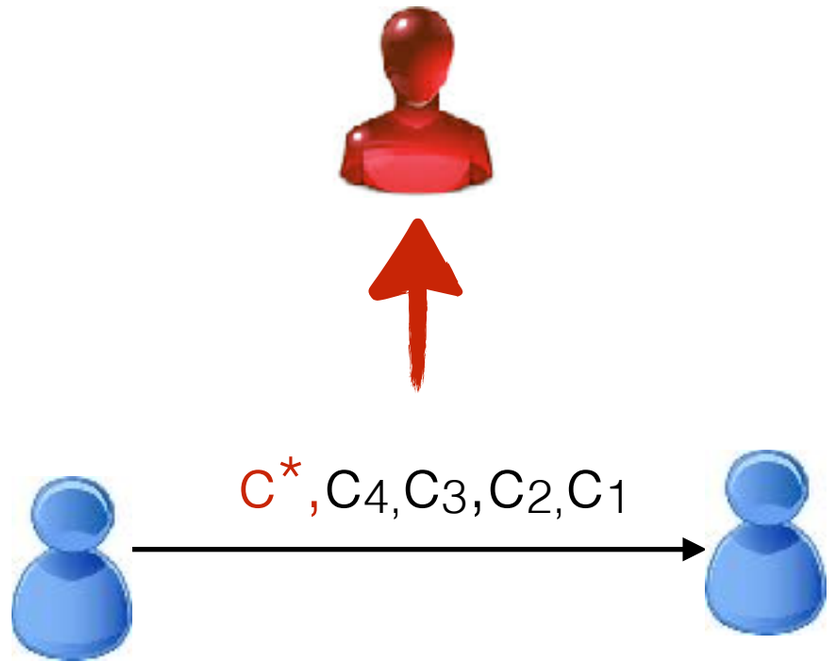
- **Assumptions**

  Pseudorandom Functions

- **Scheme + Proofs!**

  Encryption scheme from PRF

new
realistic adversary



Pseudo-random Functions

Pseudo-random Generator

Pseudo-random Functions

Pseudo-random Generator

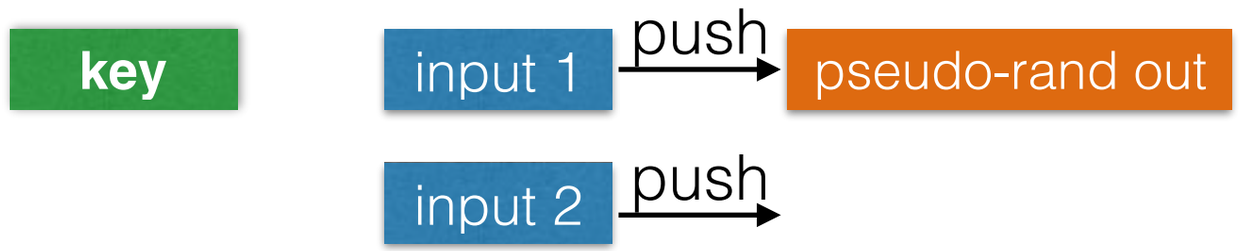
Pseudo-random Functions

key

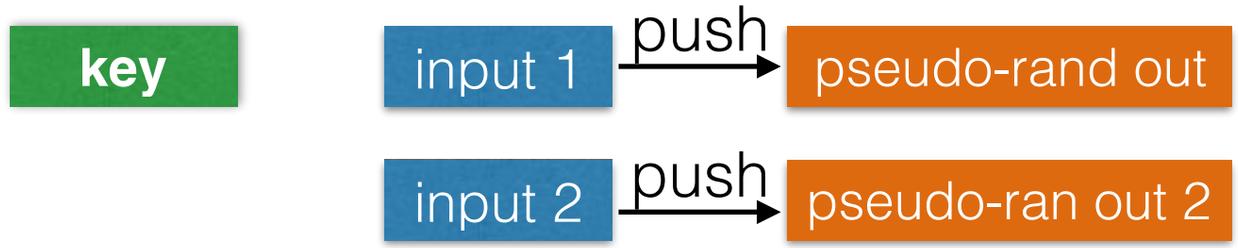
Pseudo-random Functions

Pseudo-random Functions

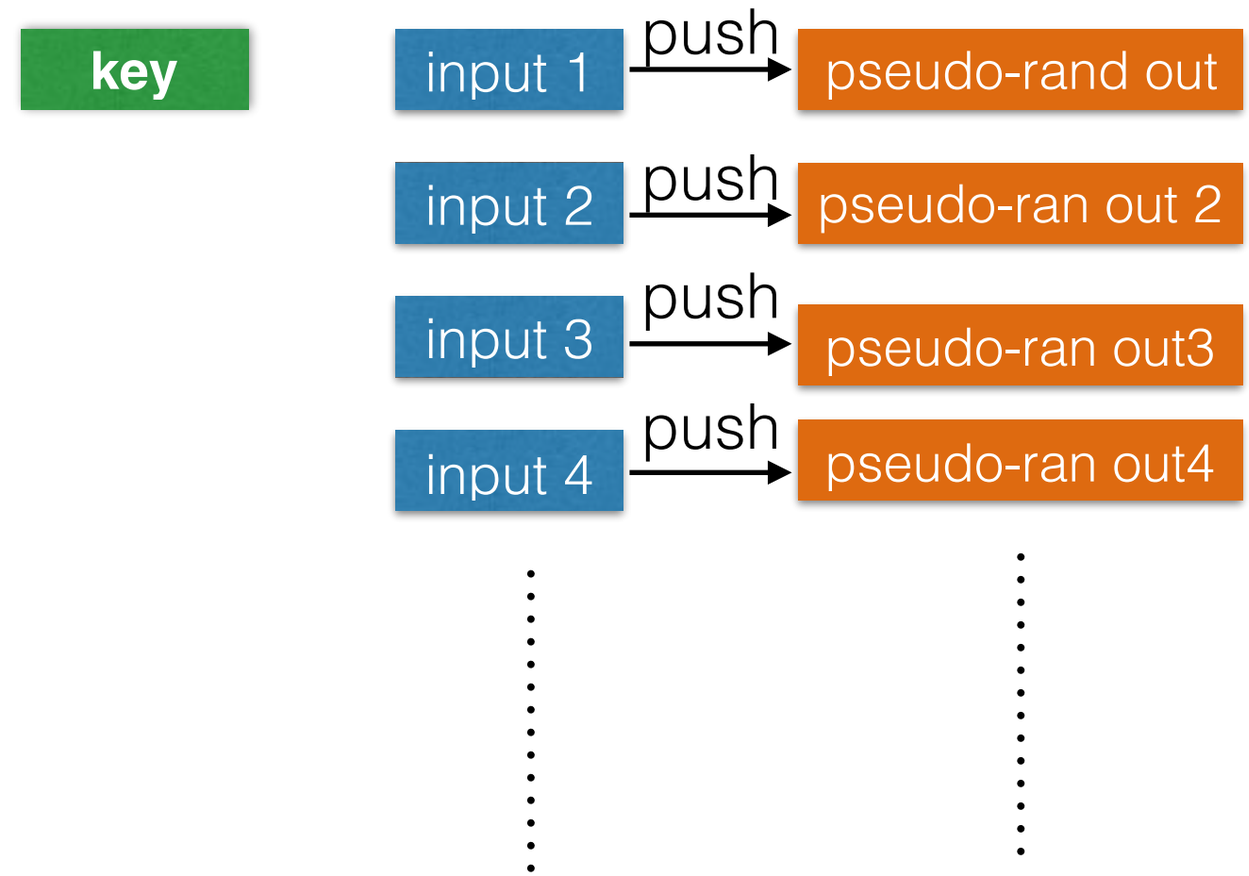
Pseudo-random Functions



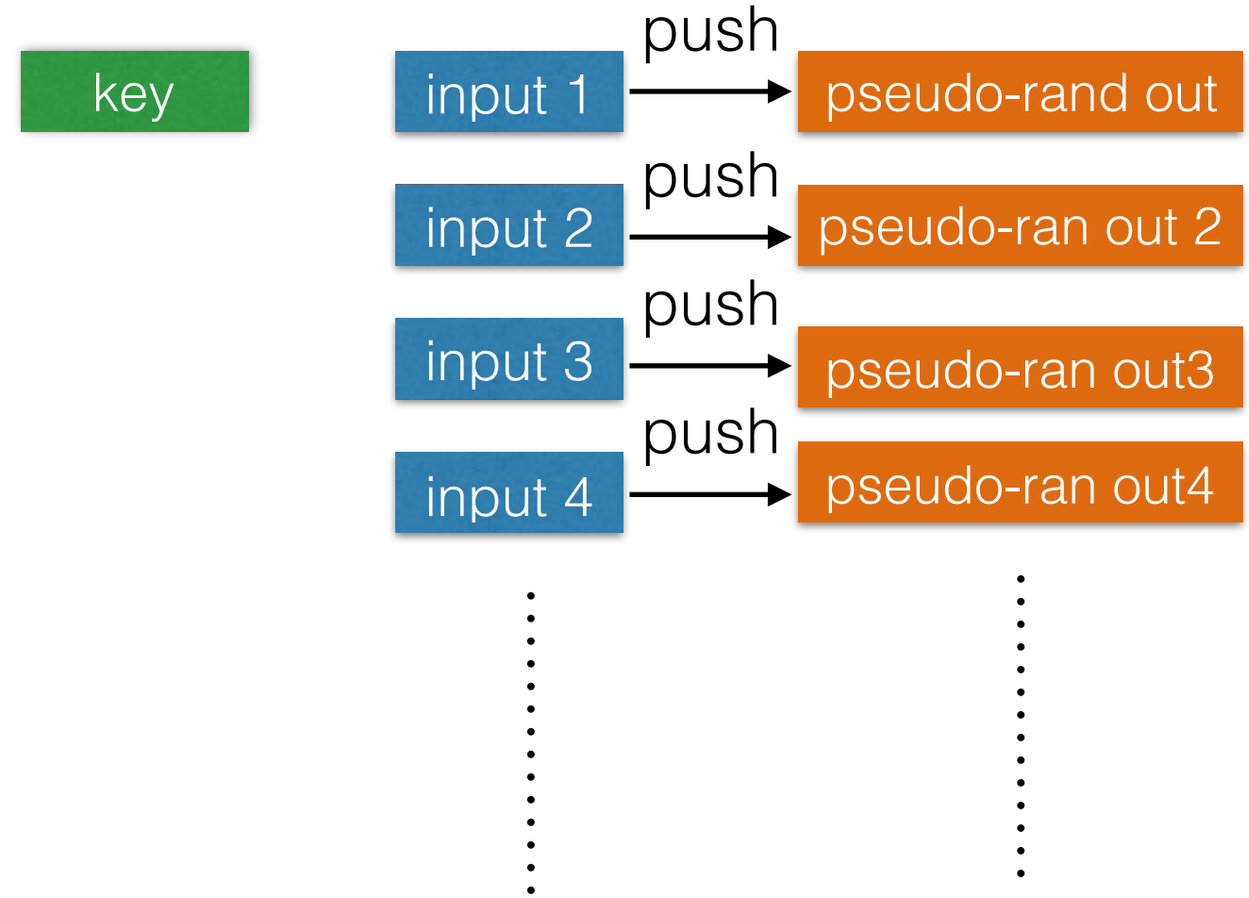
Pseudo-random Functions



Pseudo-random Functions

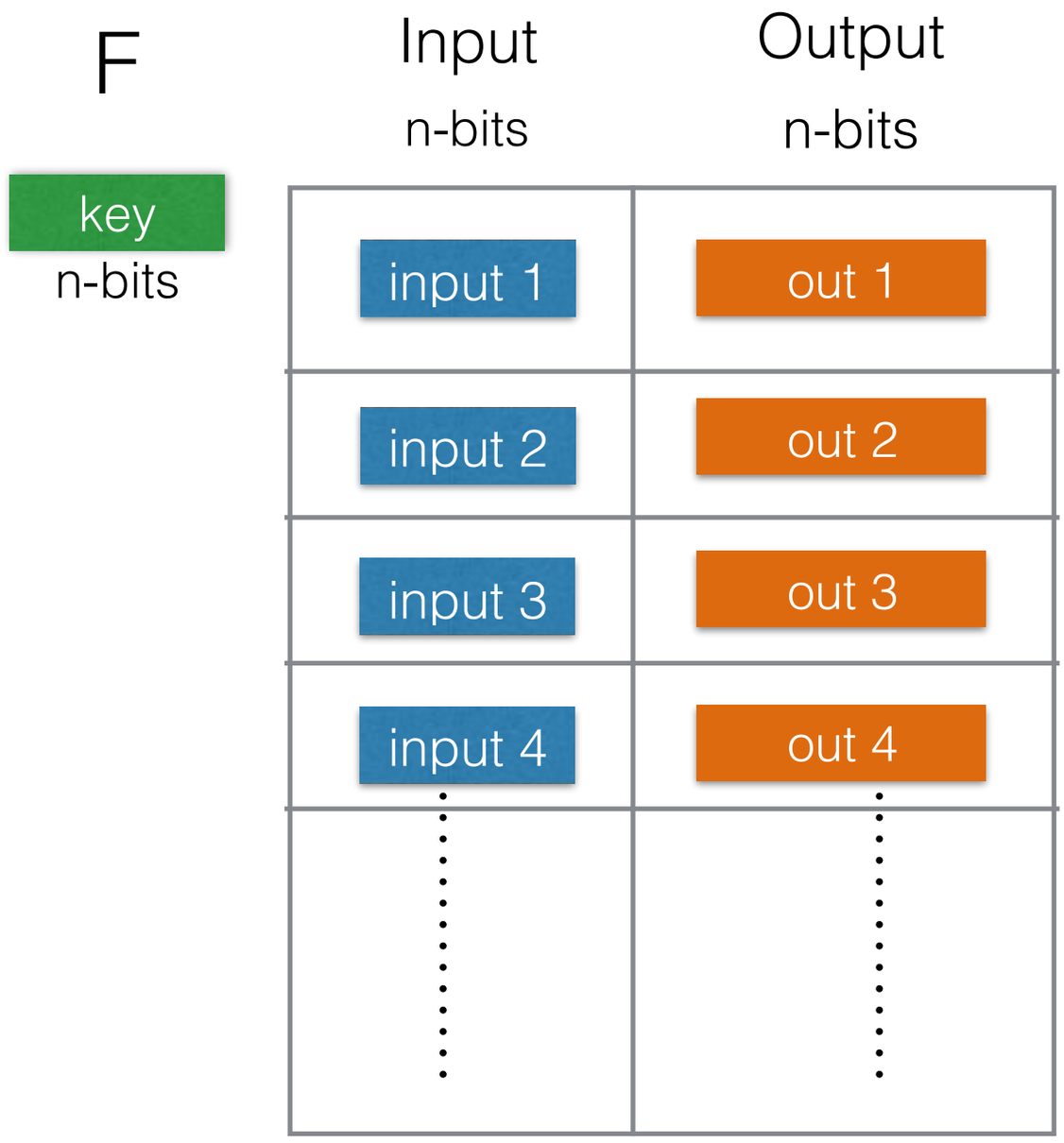


Pseudo-random Functions



Pseudo-random Functions

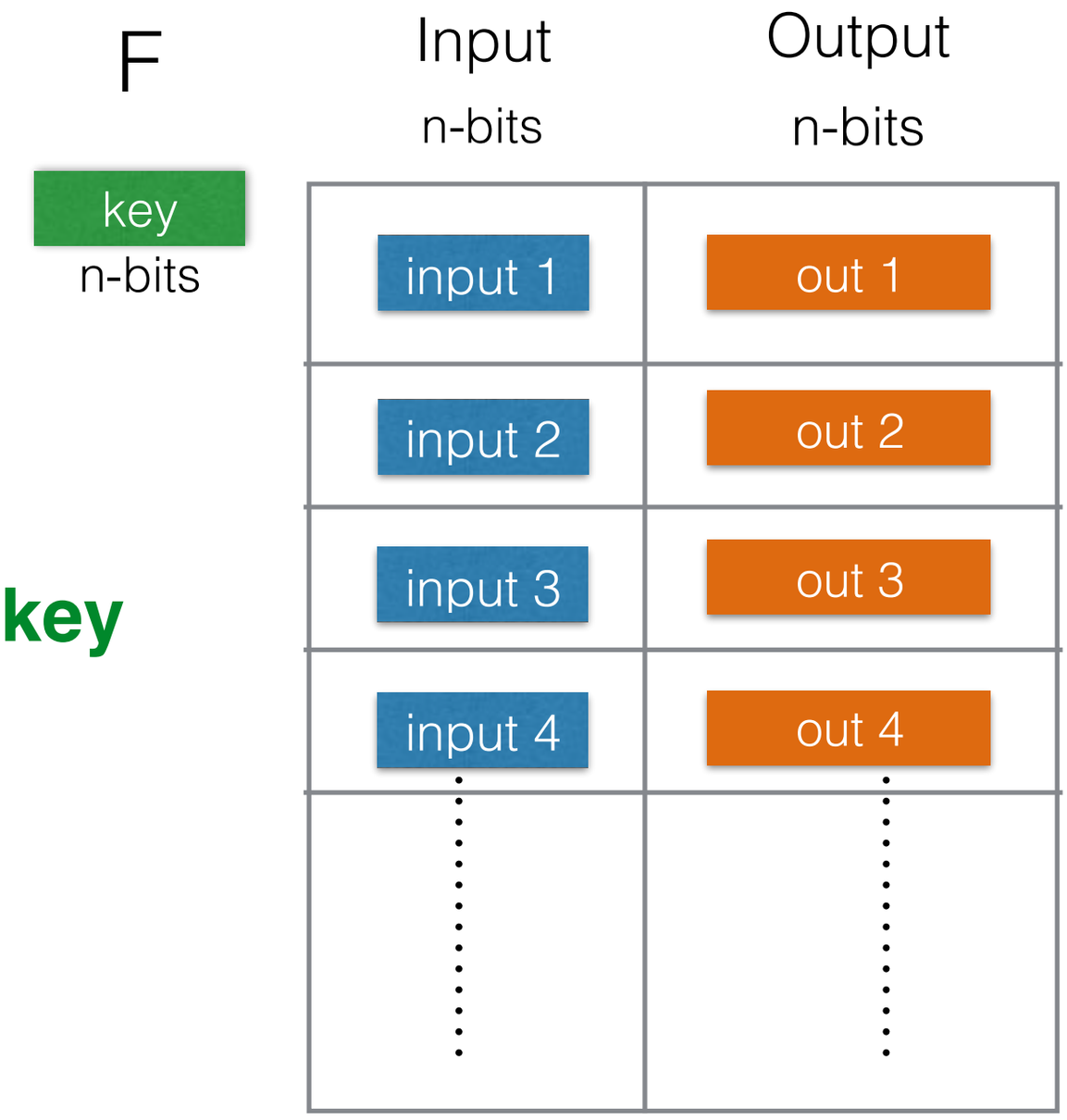
$$F_k(x) = y$$



Pseudo-random Functions

$$F_k(x) = y$$

- Deterministic after fixed **key**

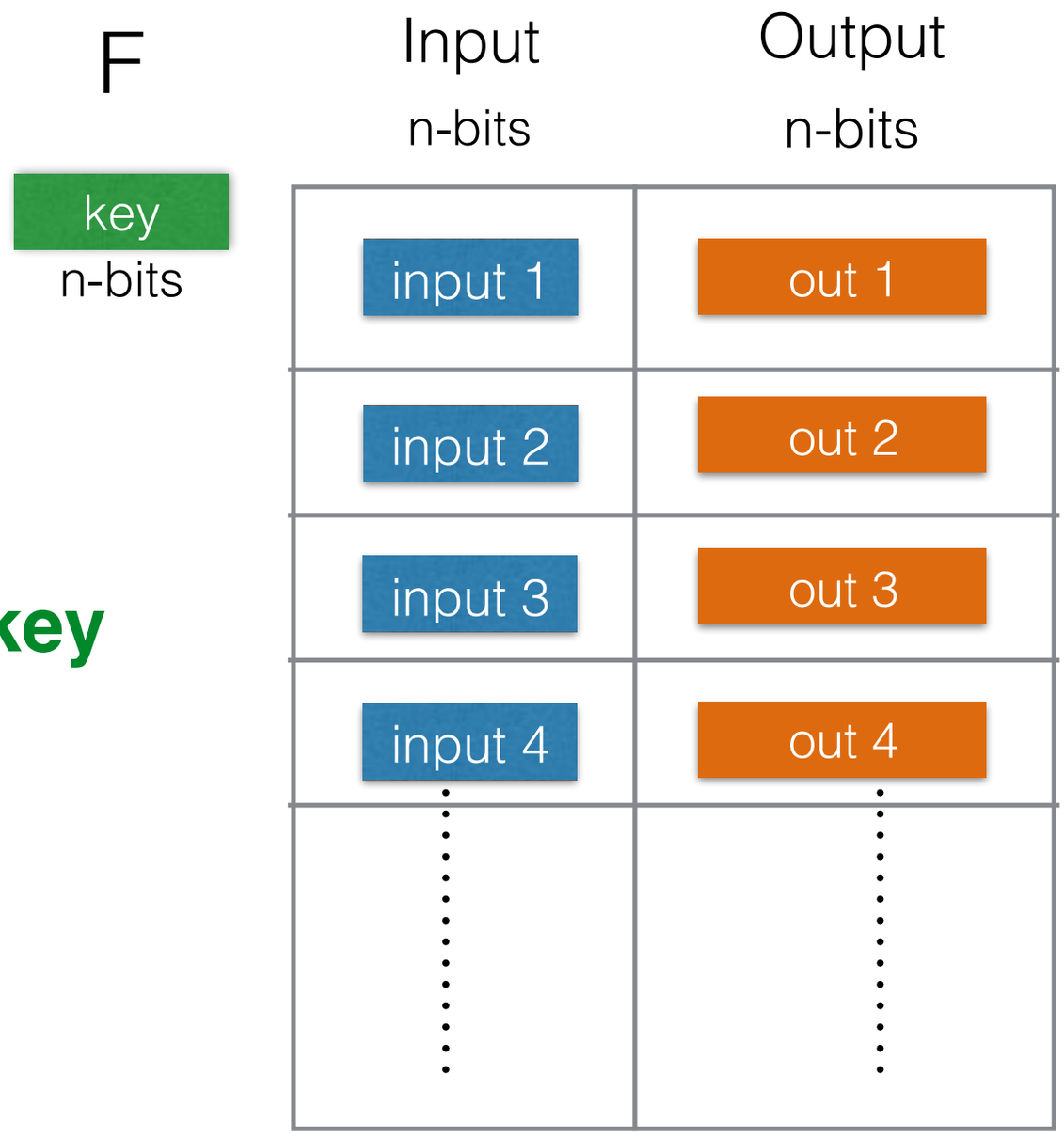


Pseudo-random Functions

$$F_k(x) = y$$

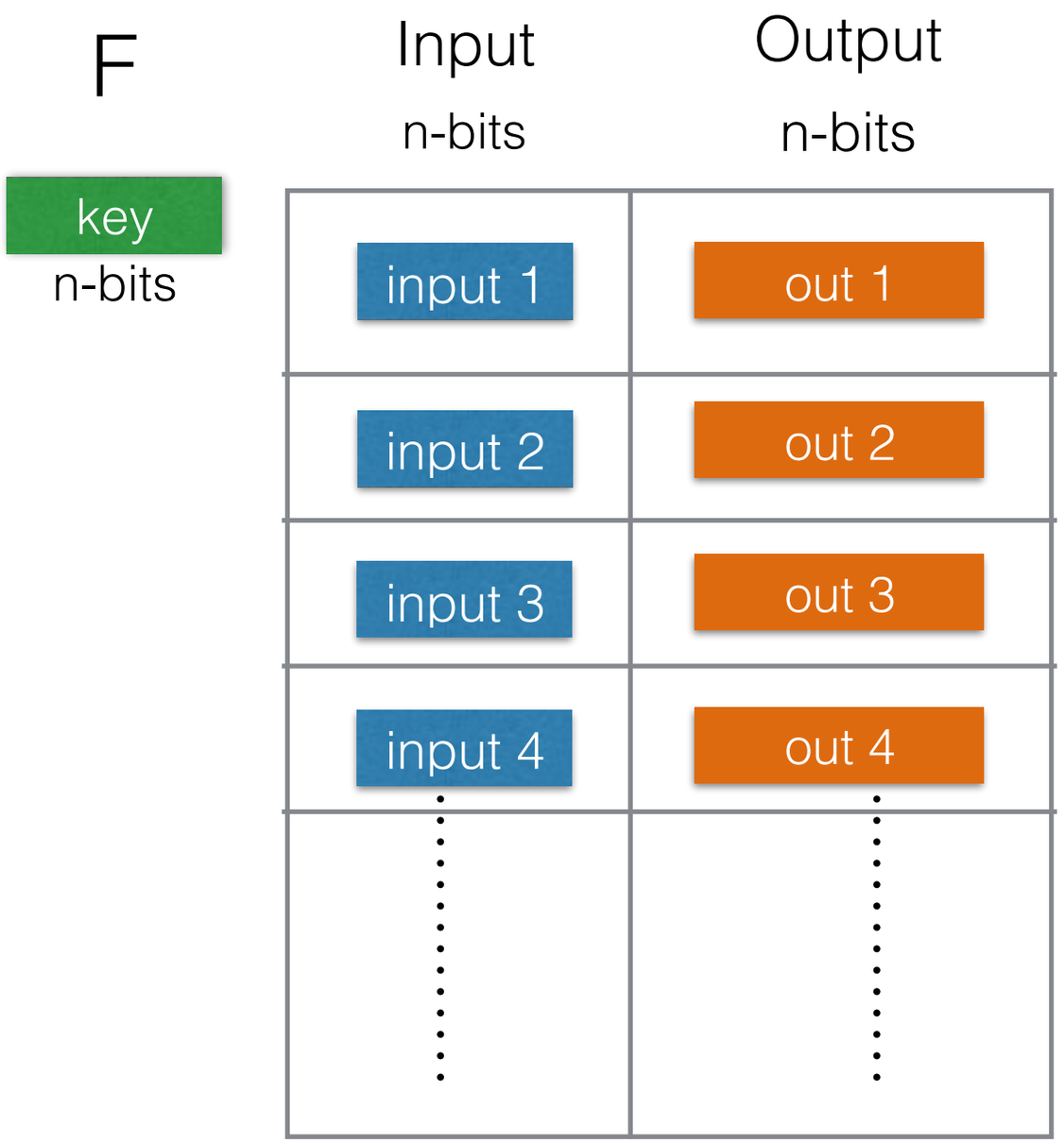
- Deterministic after fixed **key**

- We don't write down the truth table

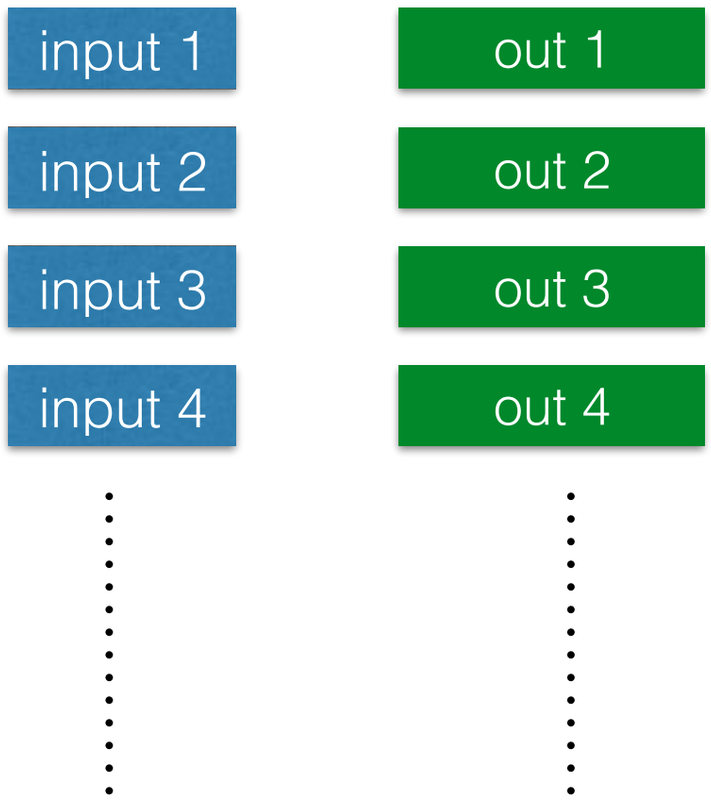


Truly Random Functions

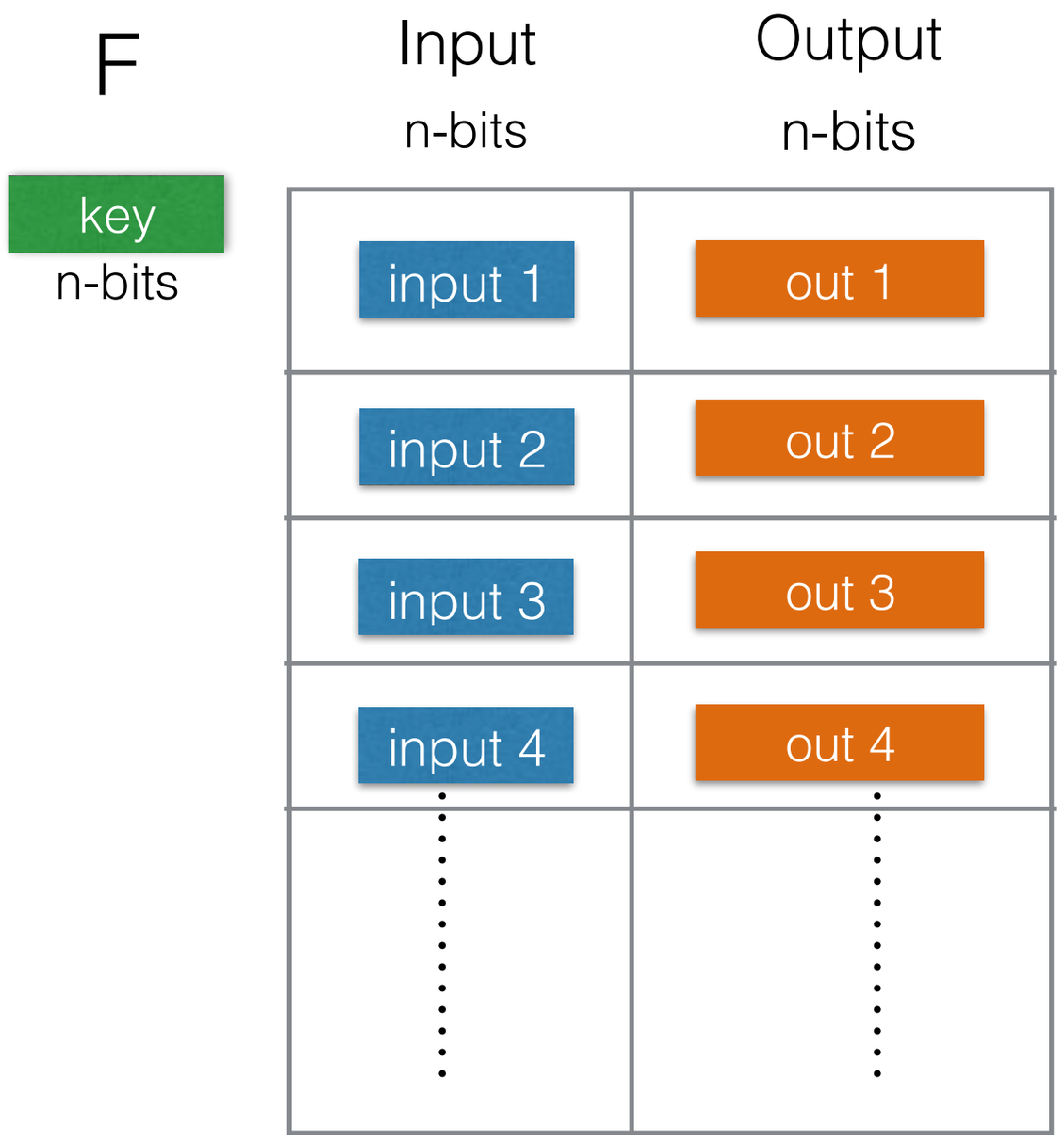
Pseudo-random Functions



Truly Random Functions

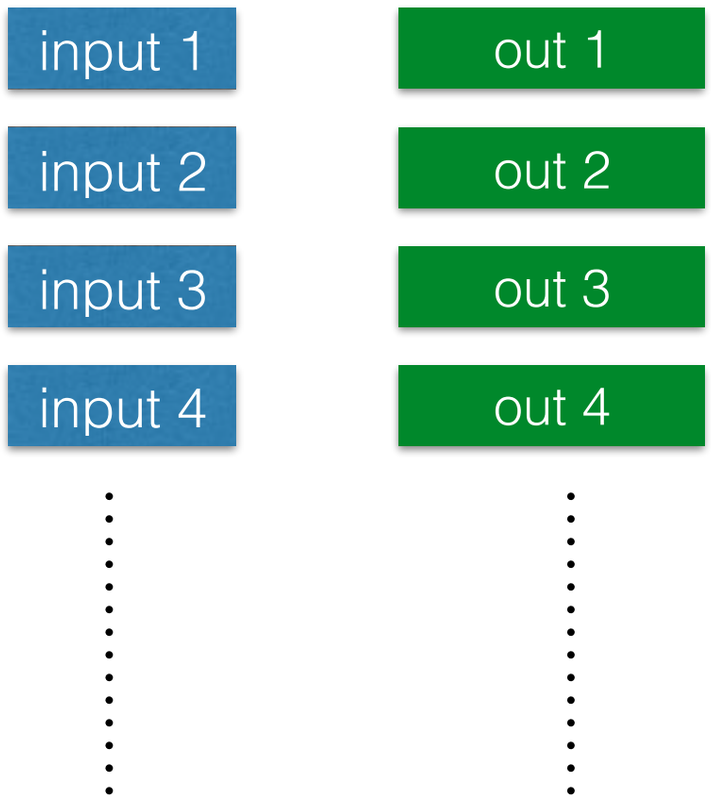


Pseudo-random Functions

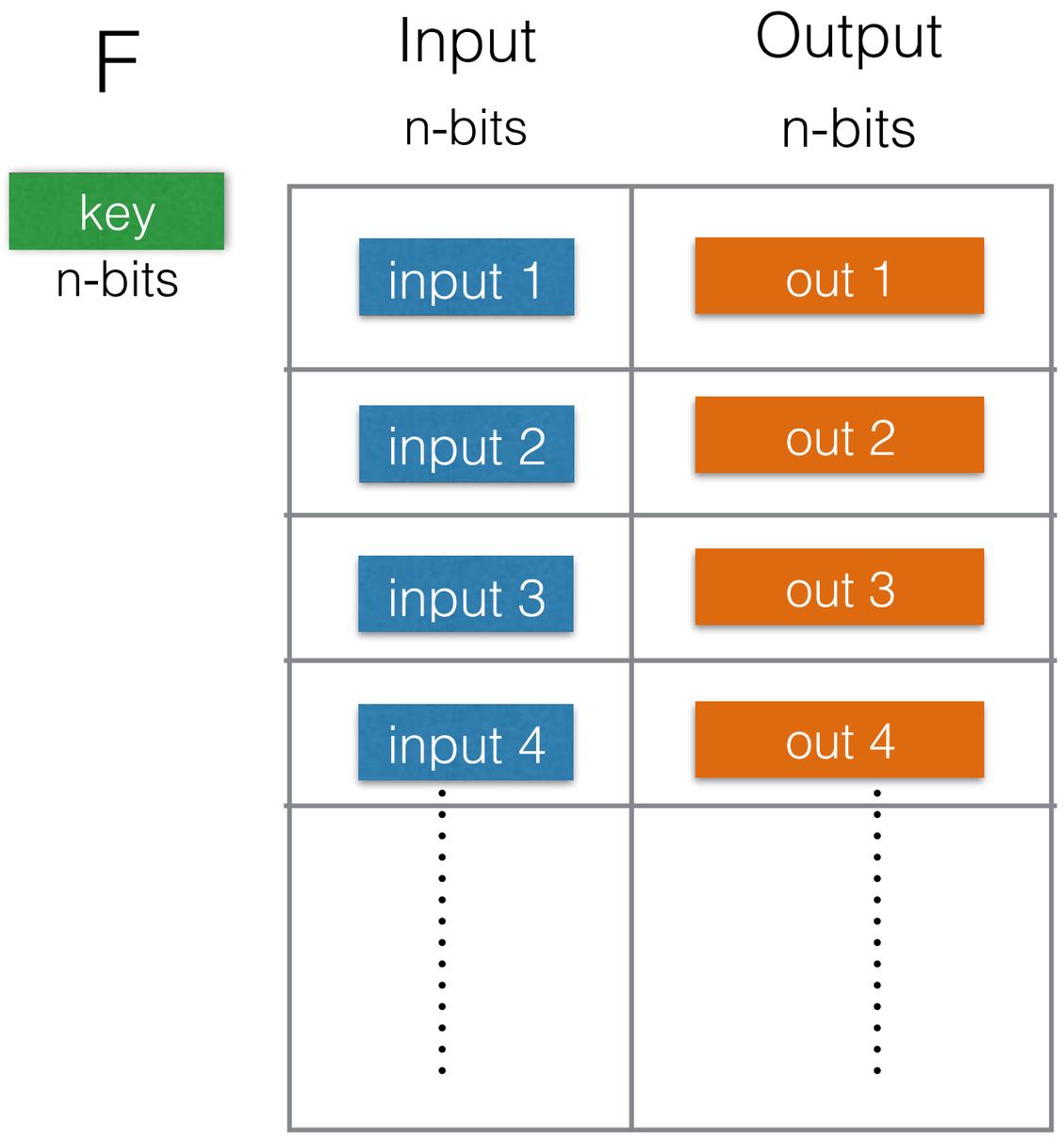


Is a PRF deterministic?

Truly Random Functions

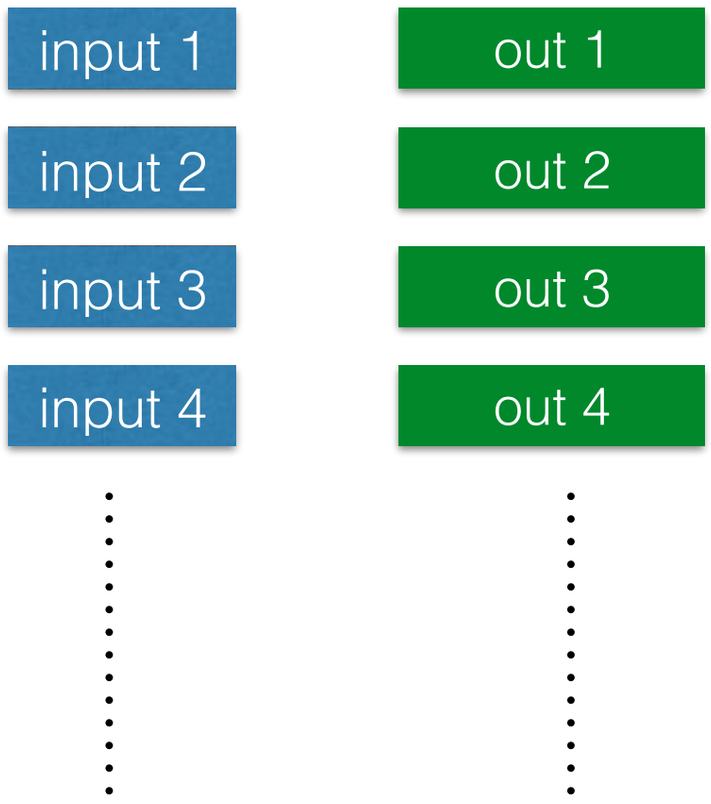


Pseudo-random Functions



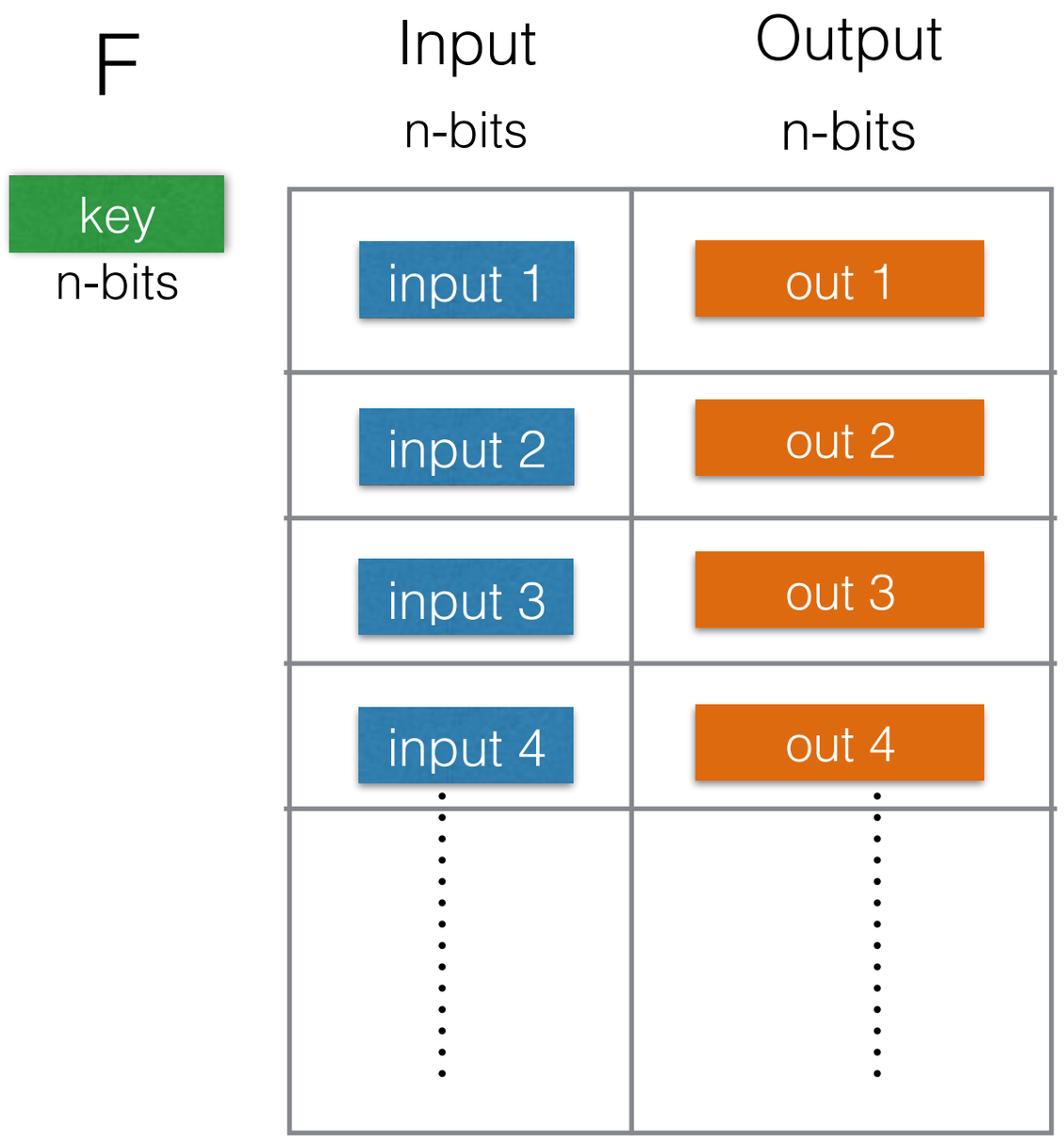
Is a Truly Random Function deterministic?

Truly Random Functions



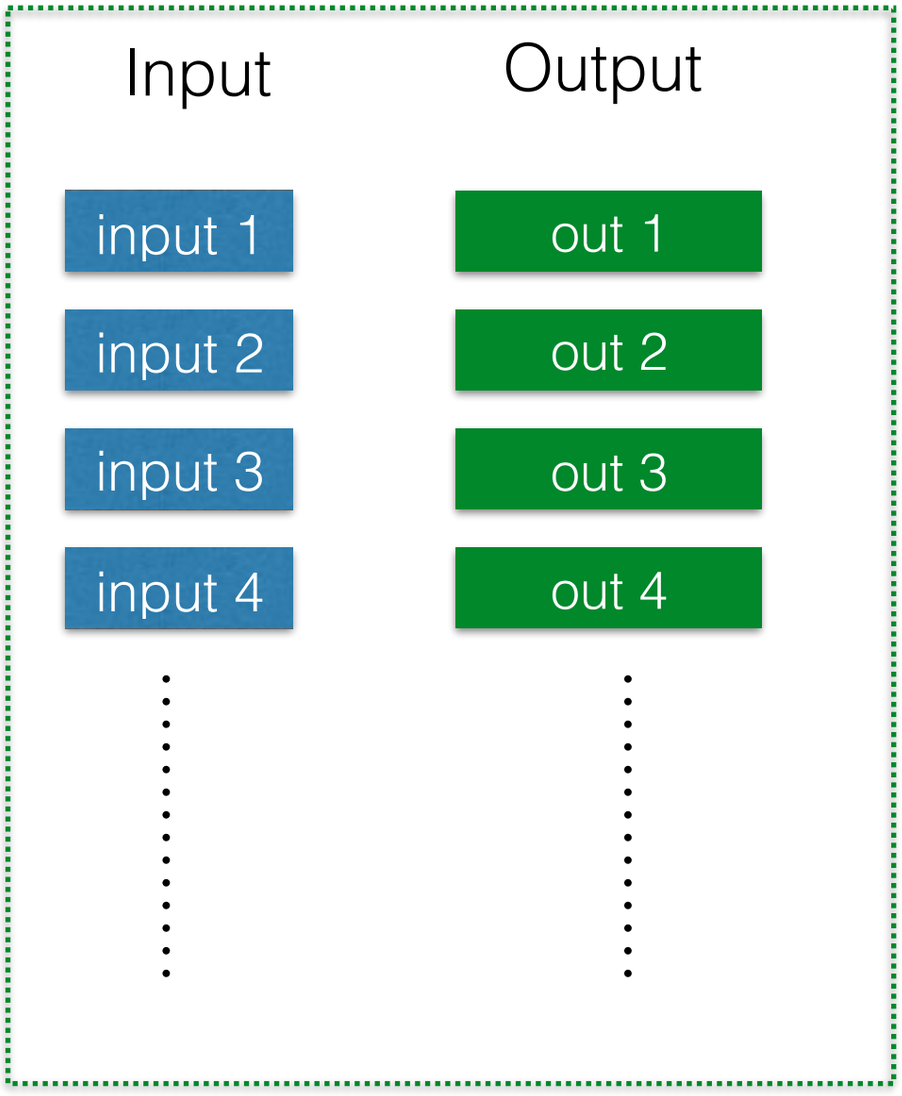
Is a PRF deterministic?

Pseudo-random Functions



PRF GAME!

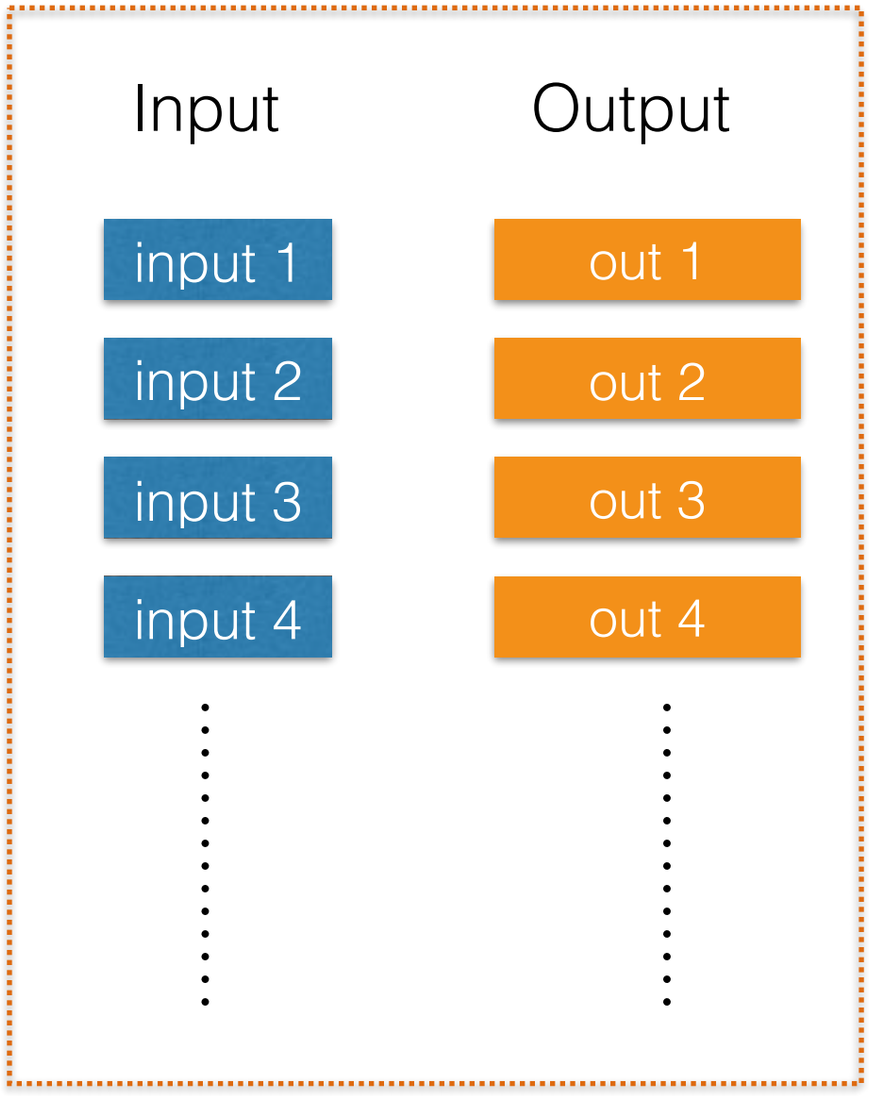
Distribution **Truly** Random Functions



key

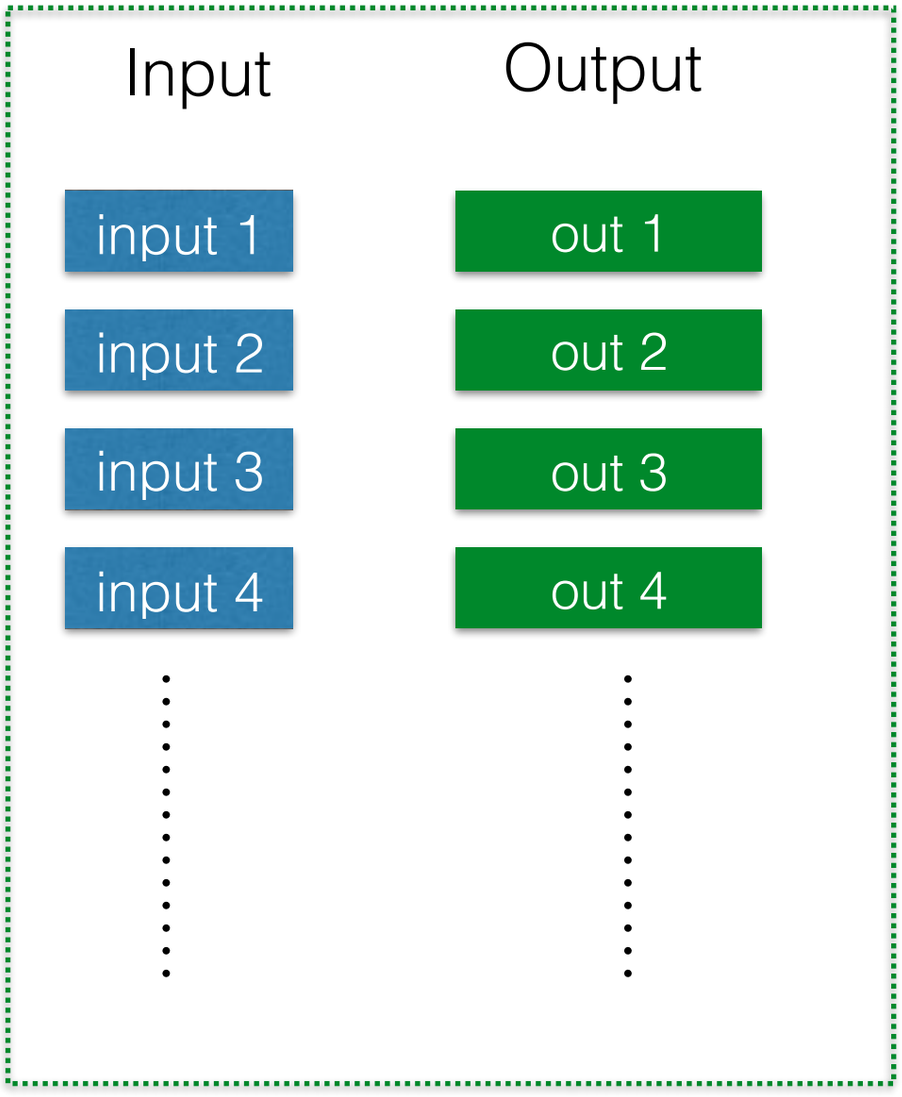
n-bits

Distribution Pseudo-random Functions

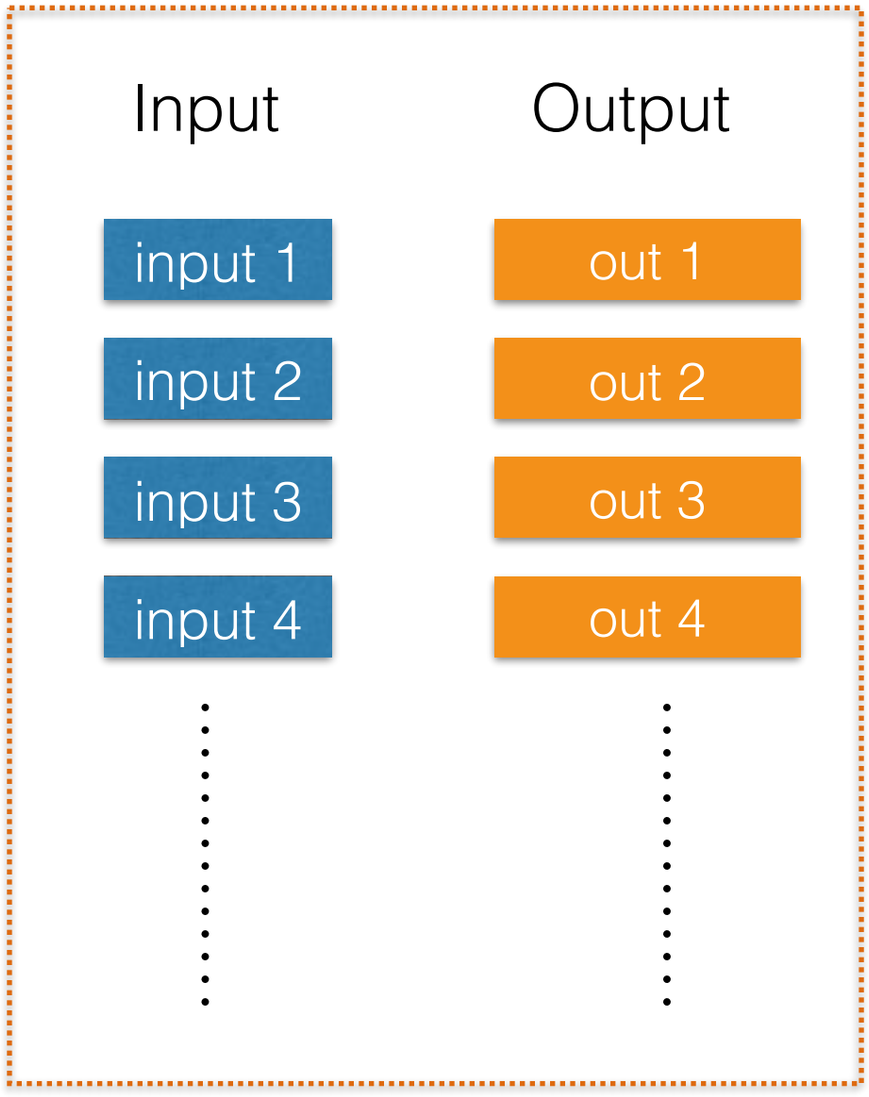


PRF GAME!

Distribution **Truly** Random Functions

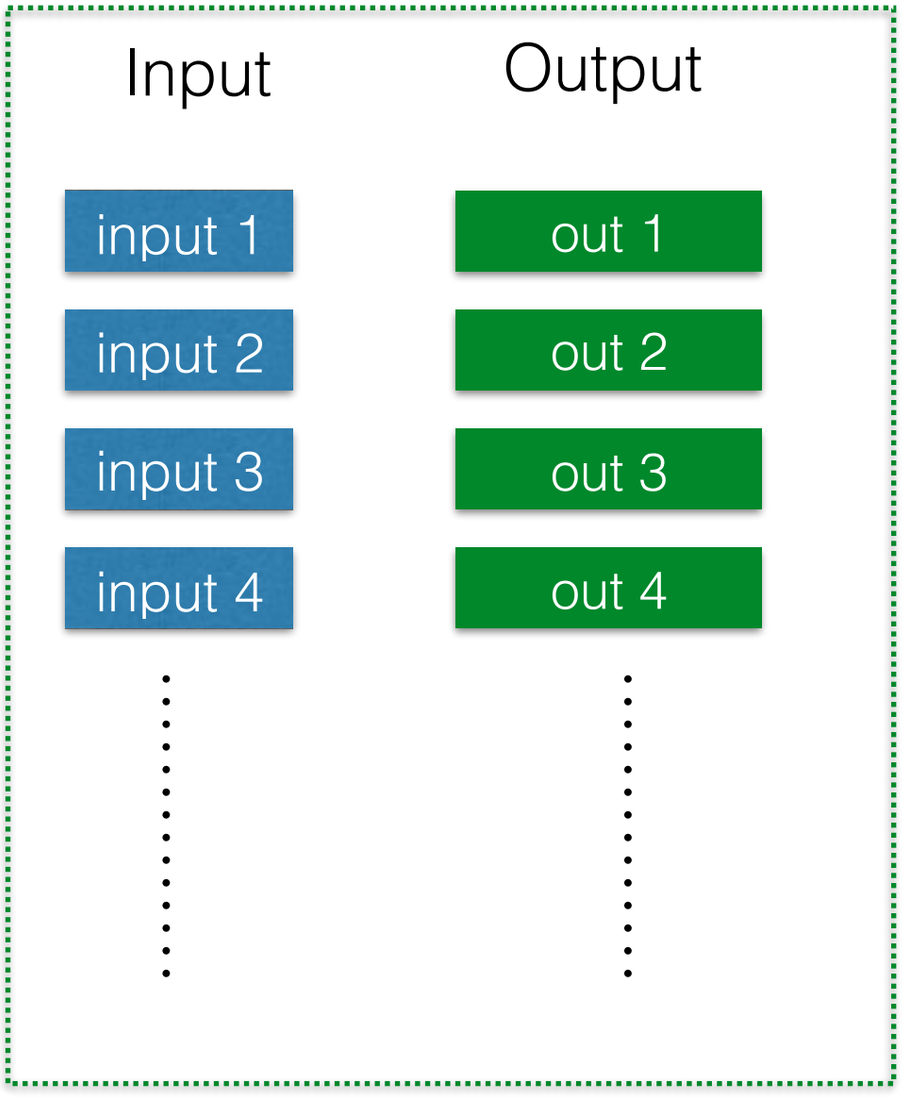


Distribution Pseudo-random Functions

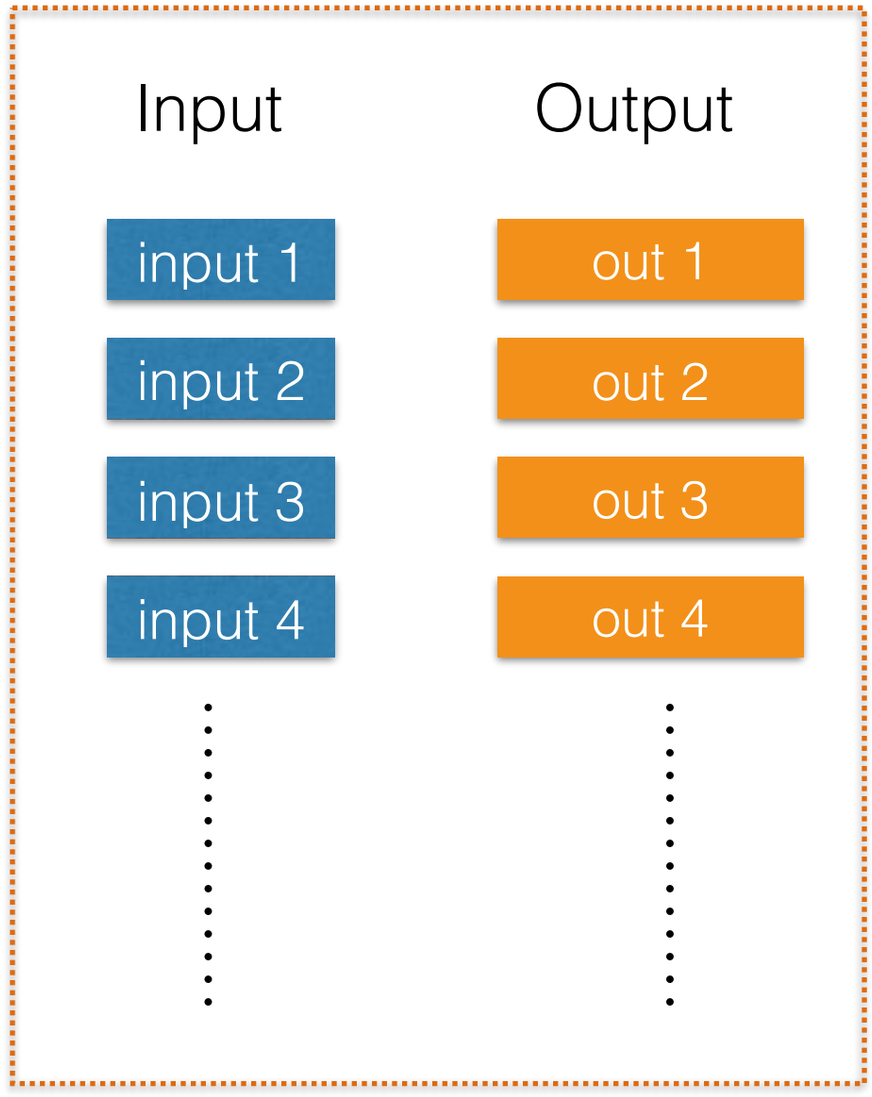


PRF GAME!

Distribution **Truly** Random Functions



Distribution Pseudo-random Functions



Indistinguishability of Pseudorandom functions

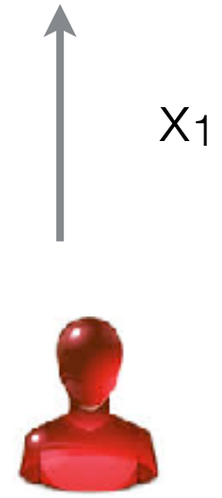
Formally

Pseudo-random function

PRF-Game

Pseudo-random function

PRF-Game



Pseudo-random function

PRF-Game

Pseudo-random function

PRF-Game

y_1

Pseudo-random function

PRF-Game

y_1

x_2

Pseudo-random function

PRF-Game

y_1

Pseudo-random function

PRF-Game

y_1

y_2

Pseudo-random function

PRF-Game

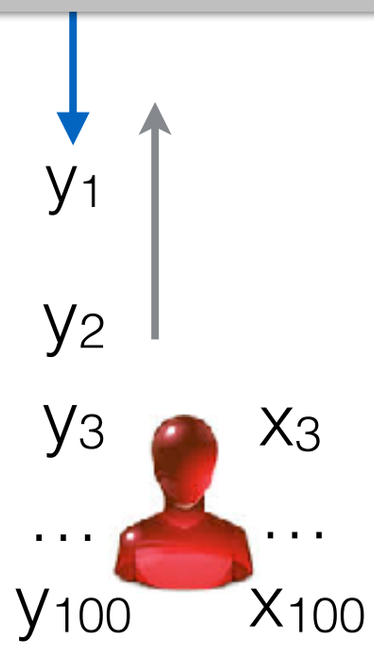
y_1

y_2

y_3 x_3

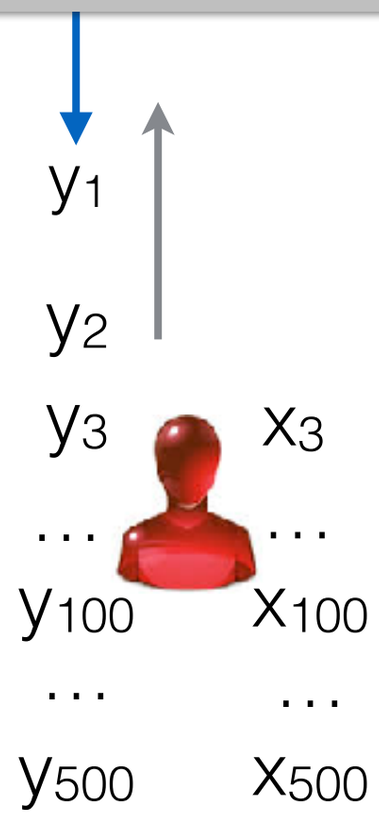
Pseudo-random function

PRF-Game



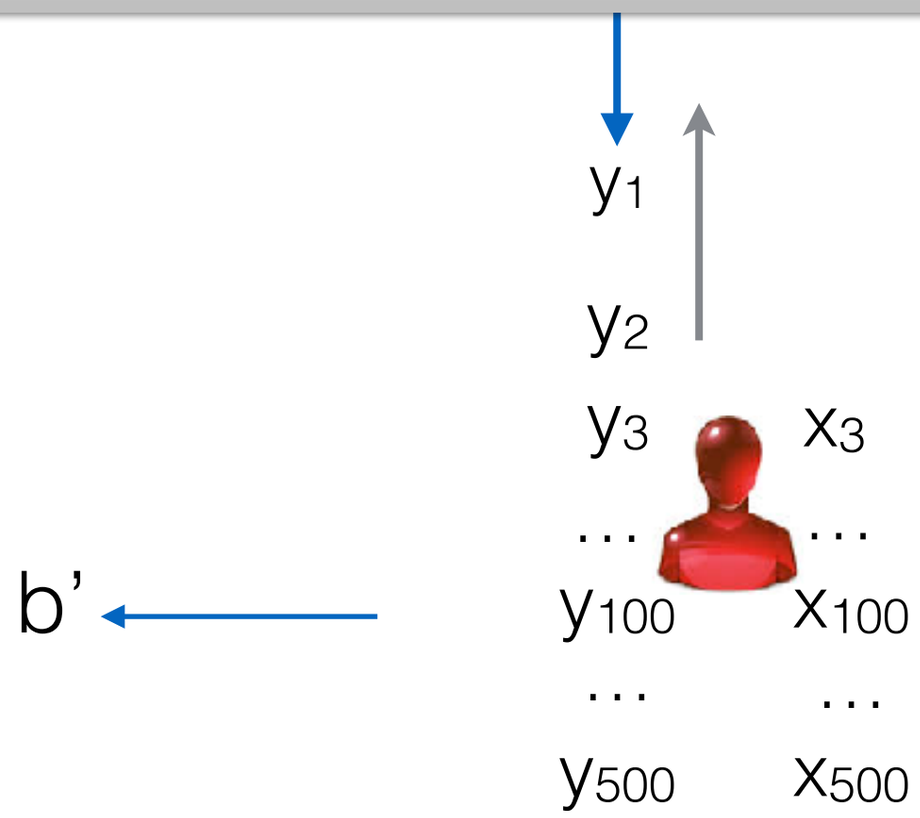
Pseudo-random function

PRF-Game

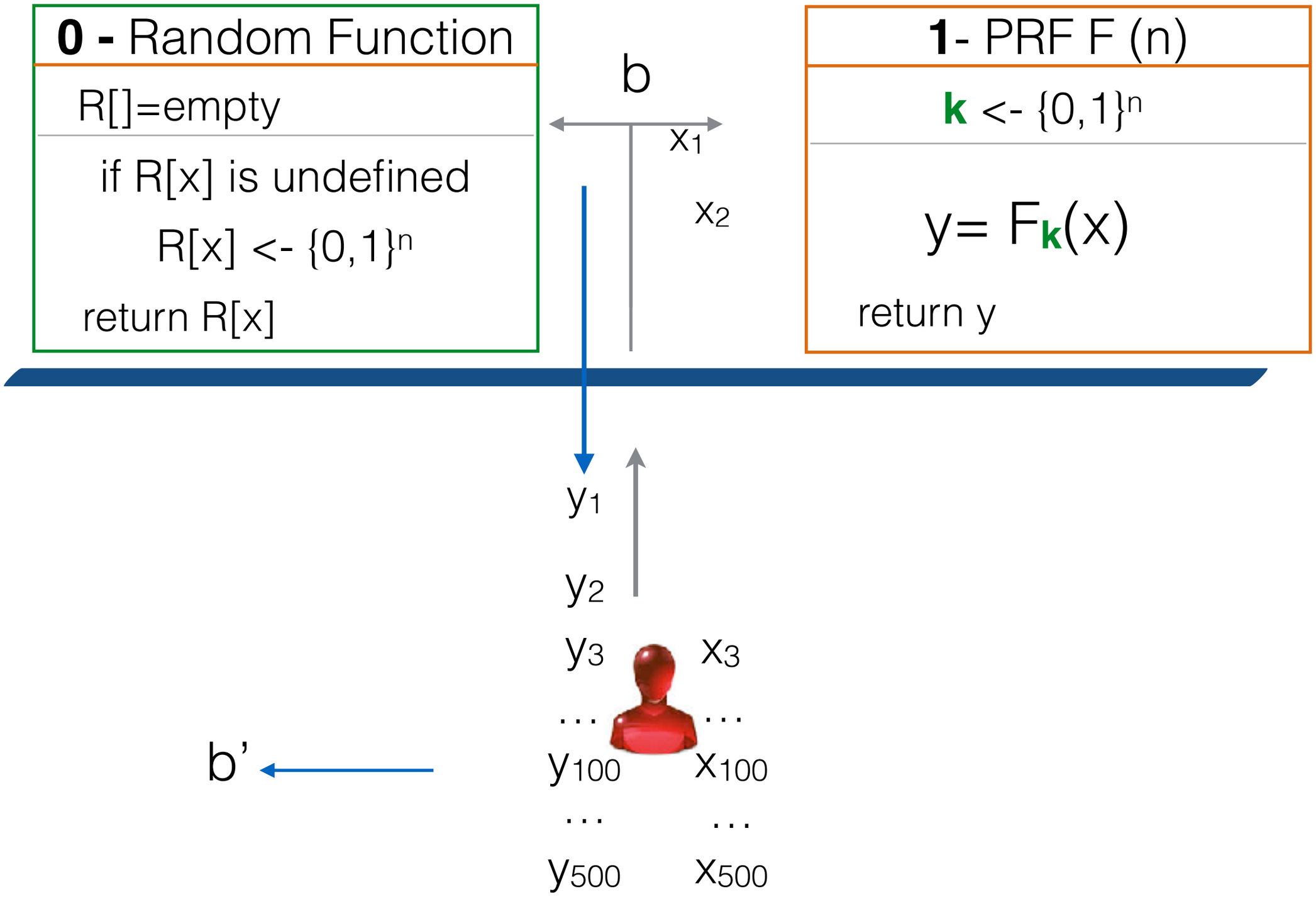


Pseudo-random function

PRF-Game



Pseudo-random function



Pseudo-random function

0 - Random Function

R[]=empty

if R[x] is undefined

R[x] <- $\{0,1\}^n$

return R[x]

b

x_1

x_2

1- PRF F (n)

k <- $\{0,1\}^n$

$y = F_{\mathbf{k}}(x)$

return y

Oracle

y_1

y_2

y_3 x_3

… …

y_{100} x_{100}

… …

y_{500} x_{500}

b’

Understanding the definition

MyF(k,x)

1. compute $y = k \oplus x$
2. output y

Understanding the definition

MyF(k,x)

1. compute $y = k \oplus x$
2. output y

Is MyF a pseudorandom function?

DEFINITION 3.24 *Let $F : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ be an efficient, length-preserving, keyed function. We say F is a* pseudorandom function *if for all probabilistic polynomial-time distinguishers D, there exists a negligible function* negl *such that:*

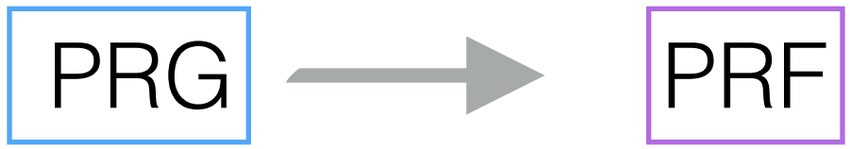
$$\left| \Pr[D^{F_k(\cdot)}(1^n) = 1] - \Pr[D^{f_n(\cdot)}(1^n) = 1] \right| \leq \mathsf{negl}(n),$$

where $k \leftarrow \{0,1\}^n$ is chosen uniformly at random and f_n is chosen uniformly at random from the set of functions mapping n-bit strings to n-bit strings.

How to construct a PRF?

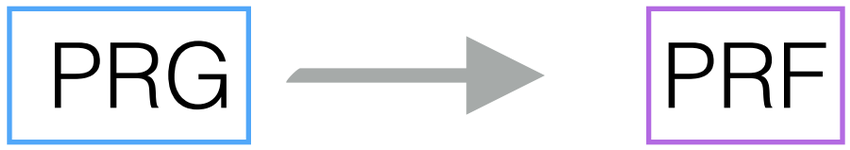
How to construct a PRF?

Theory:

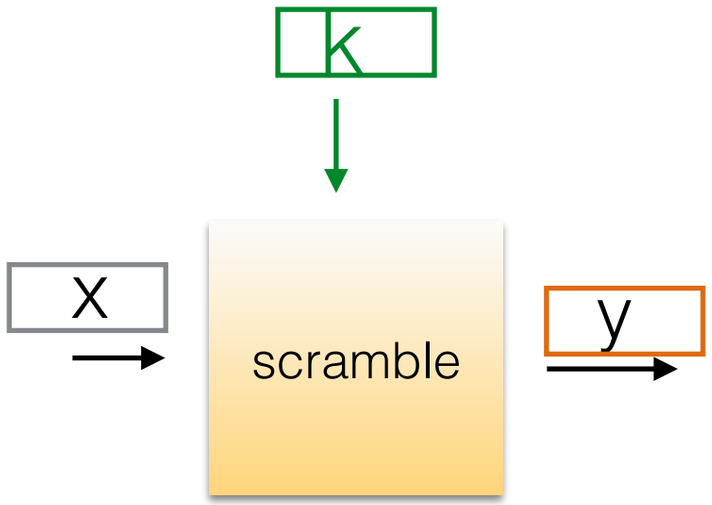


How to construct a PRF?

Theory:

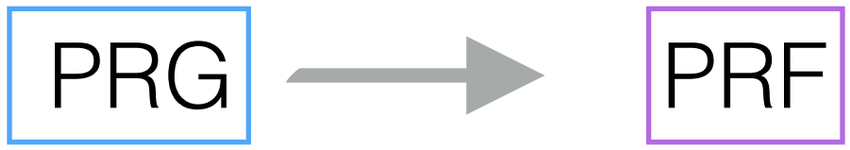


Practice:

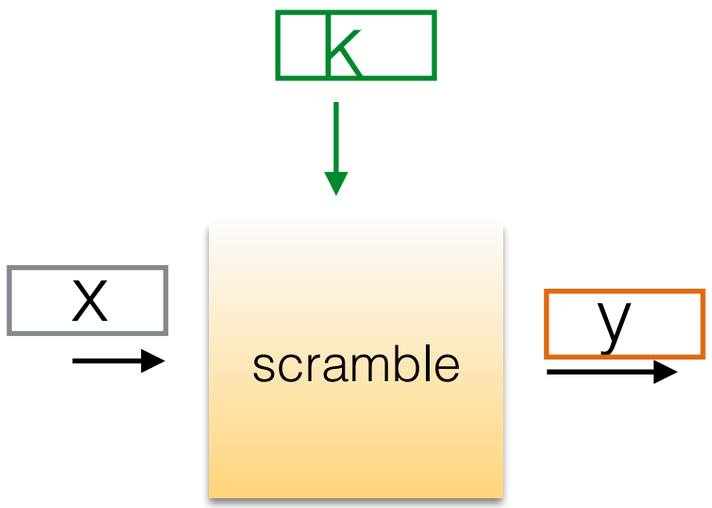


How to construct a PRF?

Theory:



Practice:



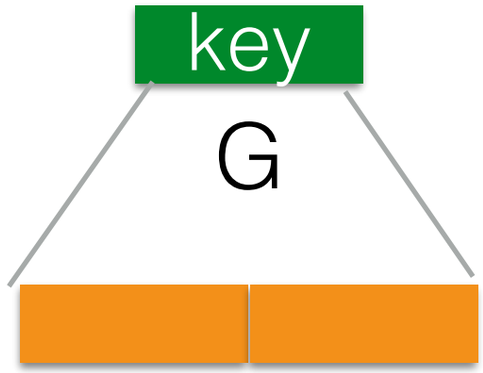
Theory:

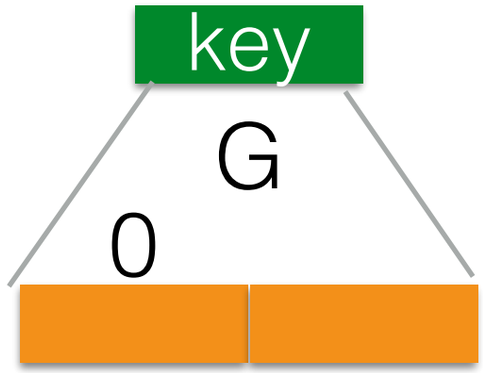
Weaker building blocks allow to build
sophisticated primitives

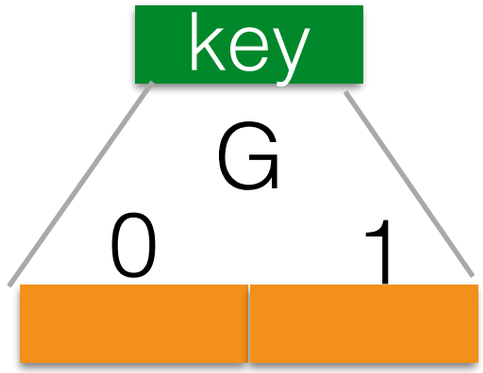
Theory: PRG -> PRF

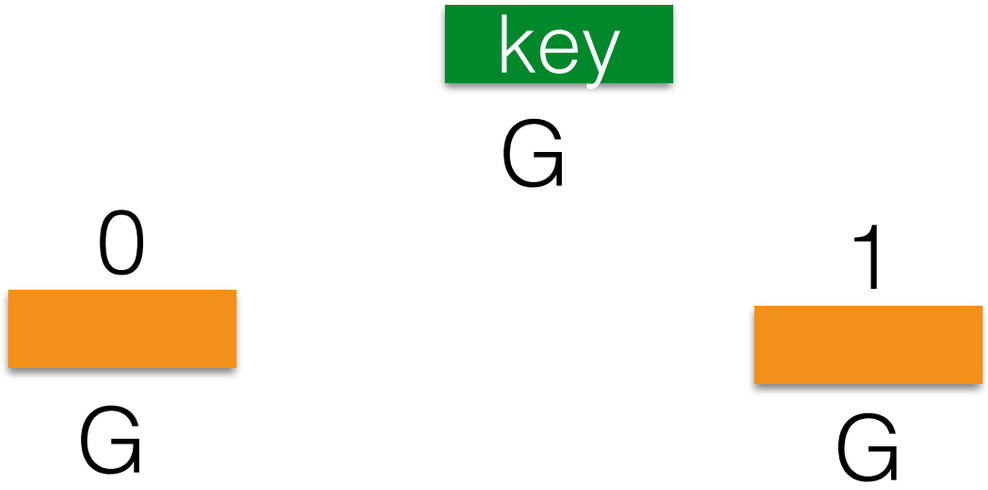
Theory: PRG -> PRF

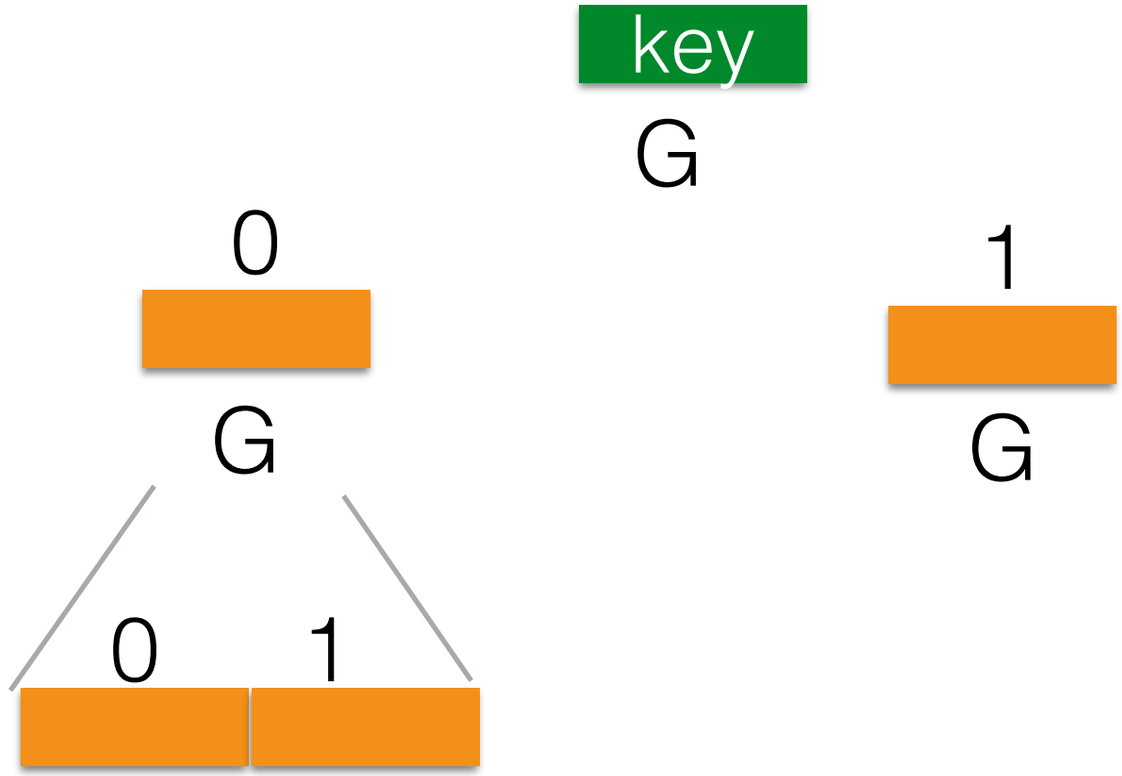
Any idea?

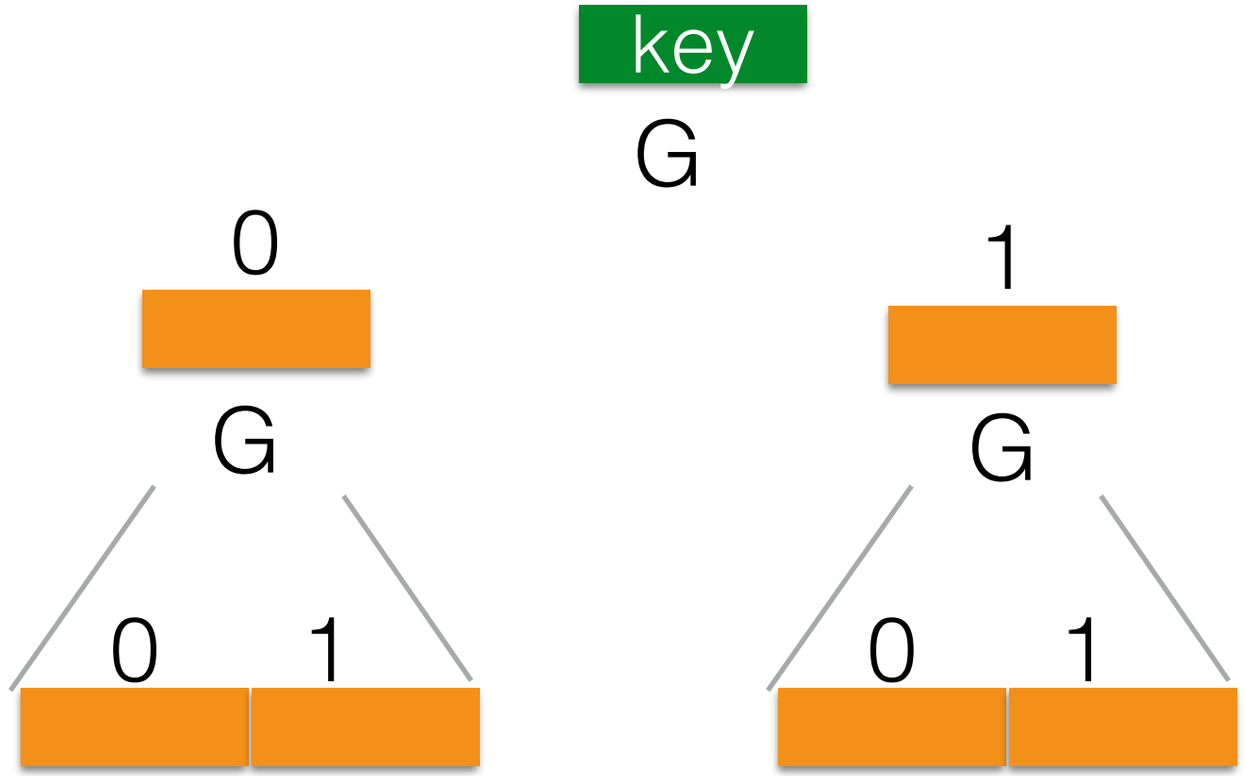

key
G

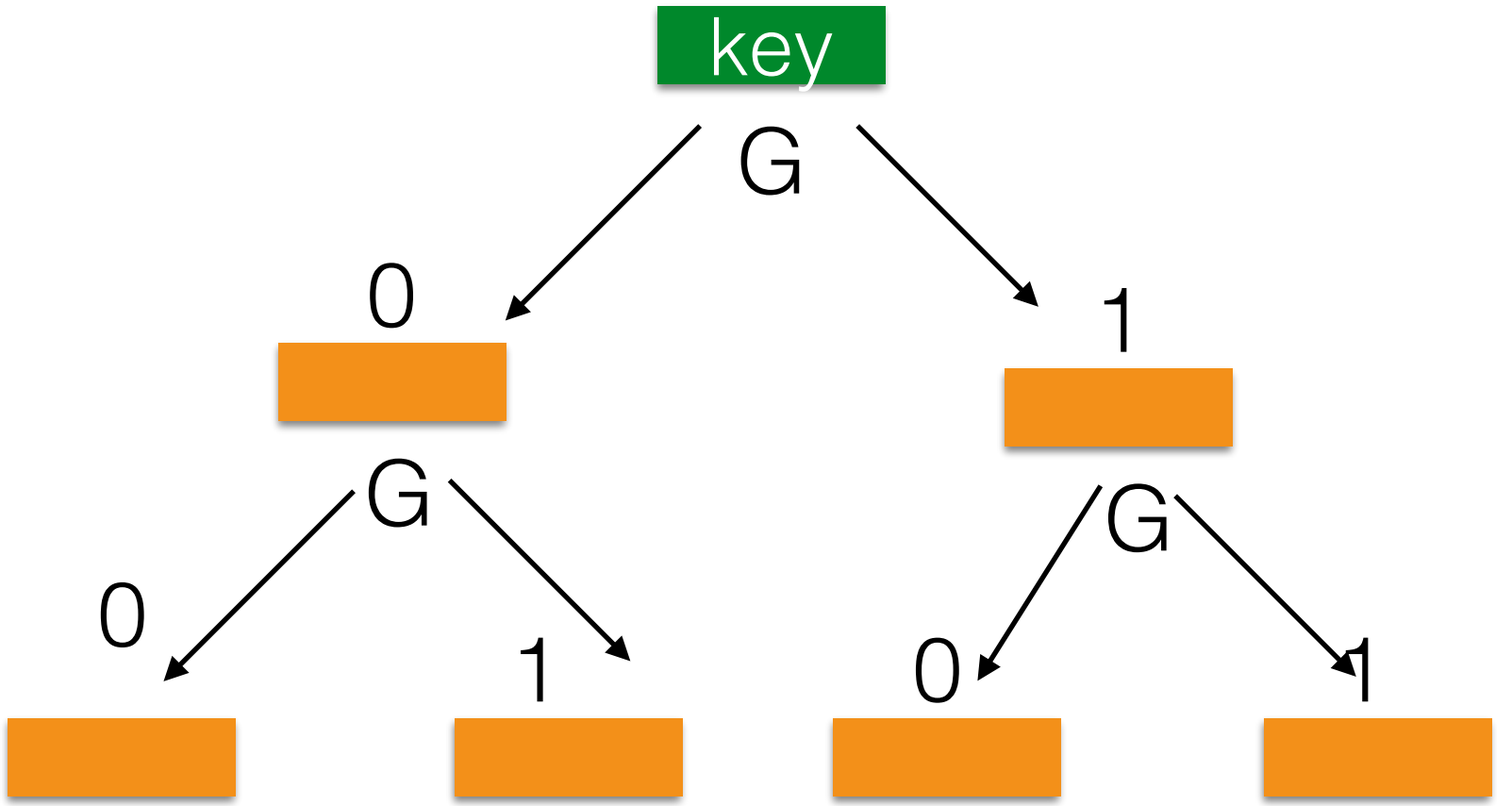

key
G
0

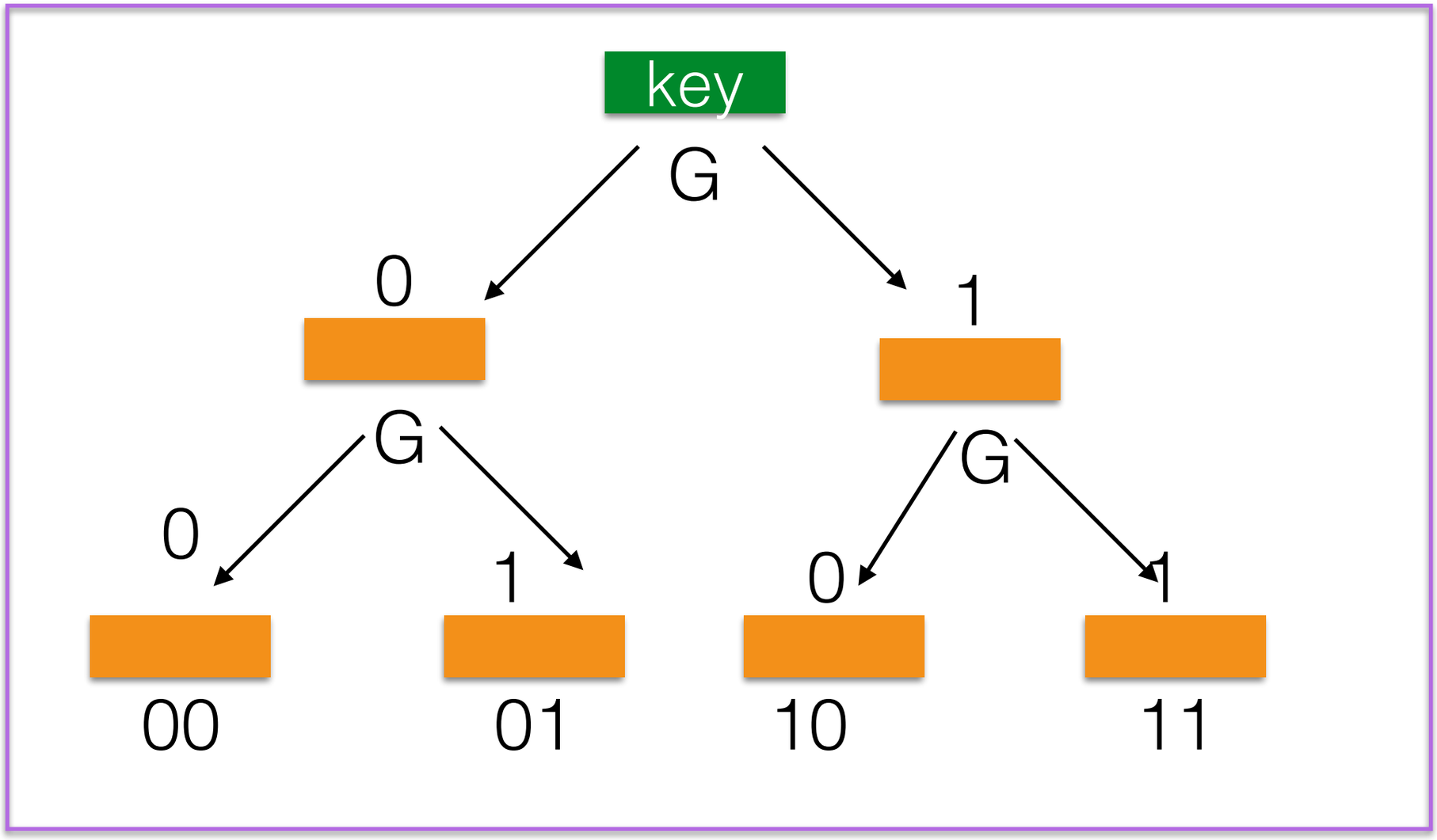

key
G
0
1

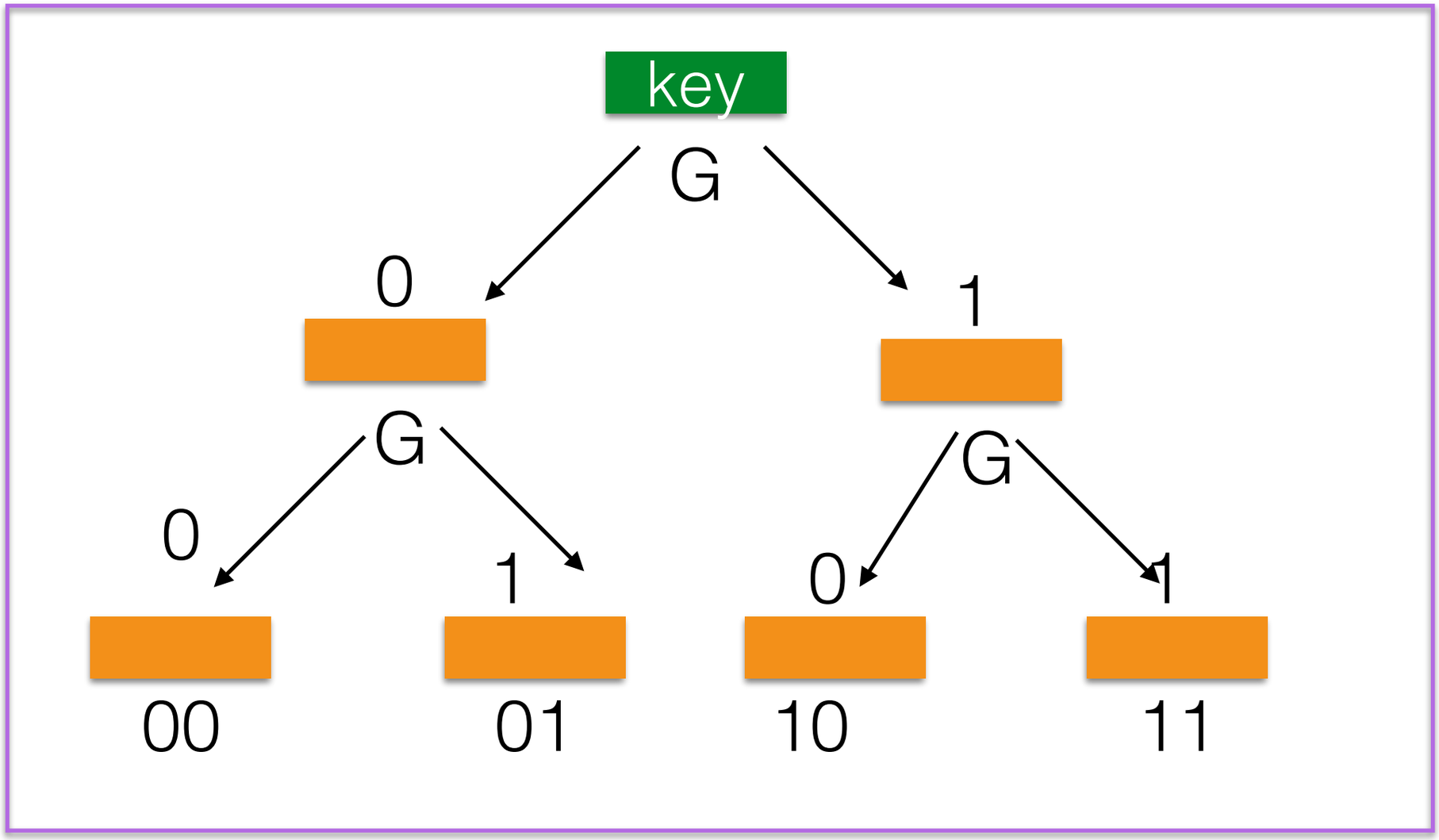

key
G
0
G
1
G


key
G
0
G
1
G
0
1


key
G
0
G
0
1
1
G
0
1
0
1


key
G
0
1
G
G
0
1
0
1
00
01
10
11

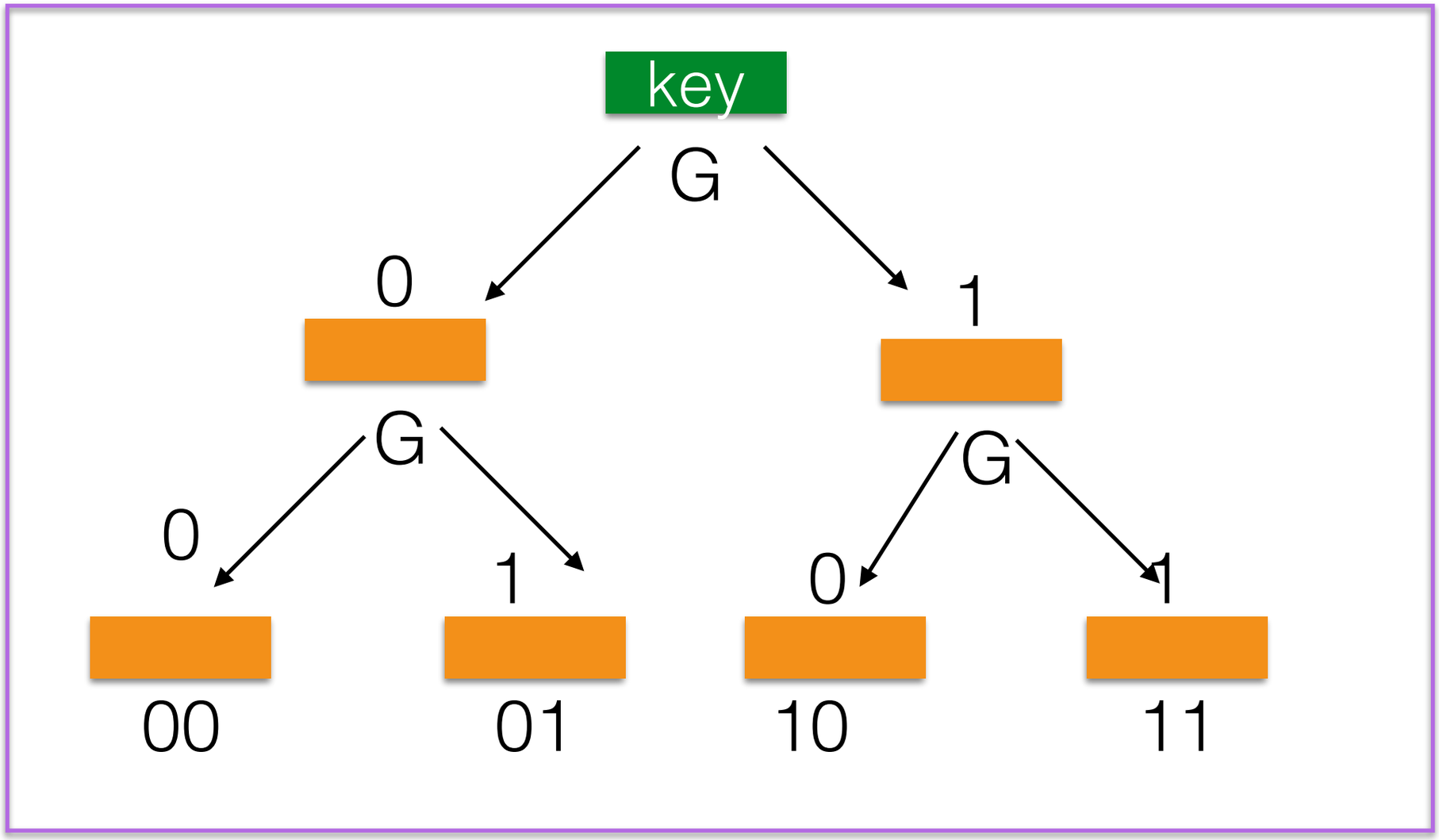

key
G
0
1
G
G
0
1
0
1
00
01
10
11



F(k, x)

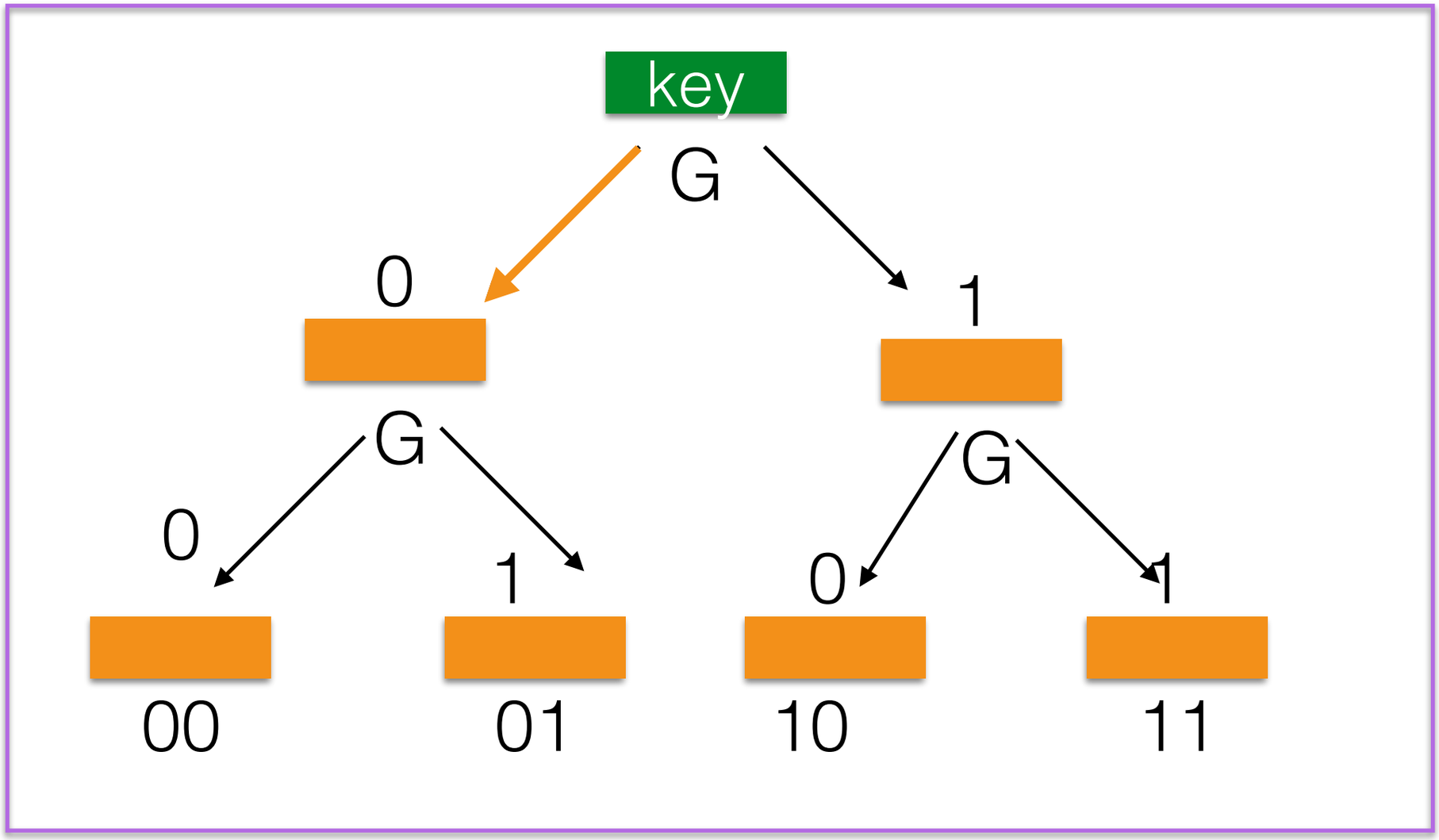
PRF for 2-bit inputs

F(k, 01)



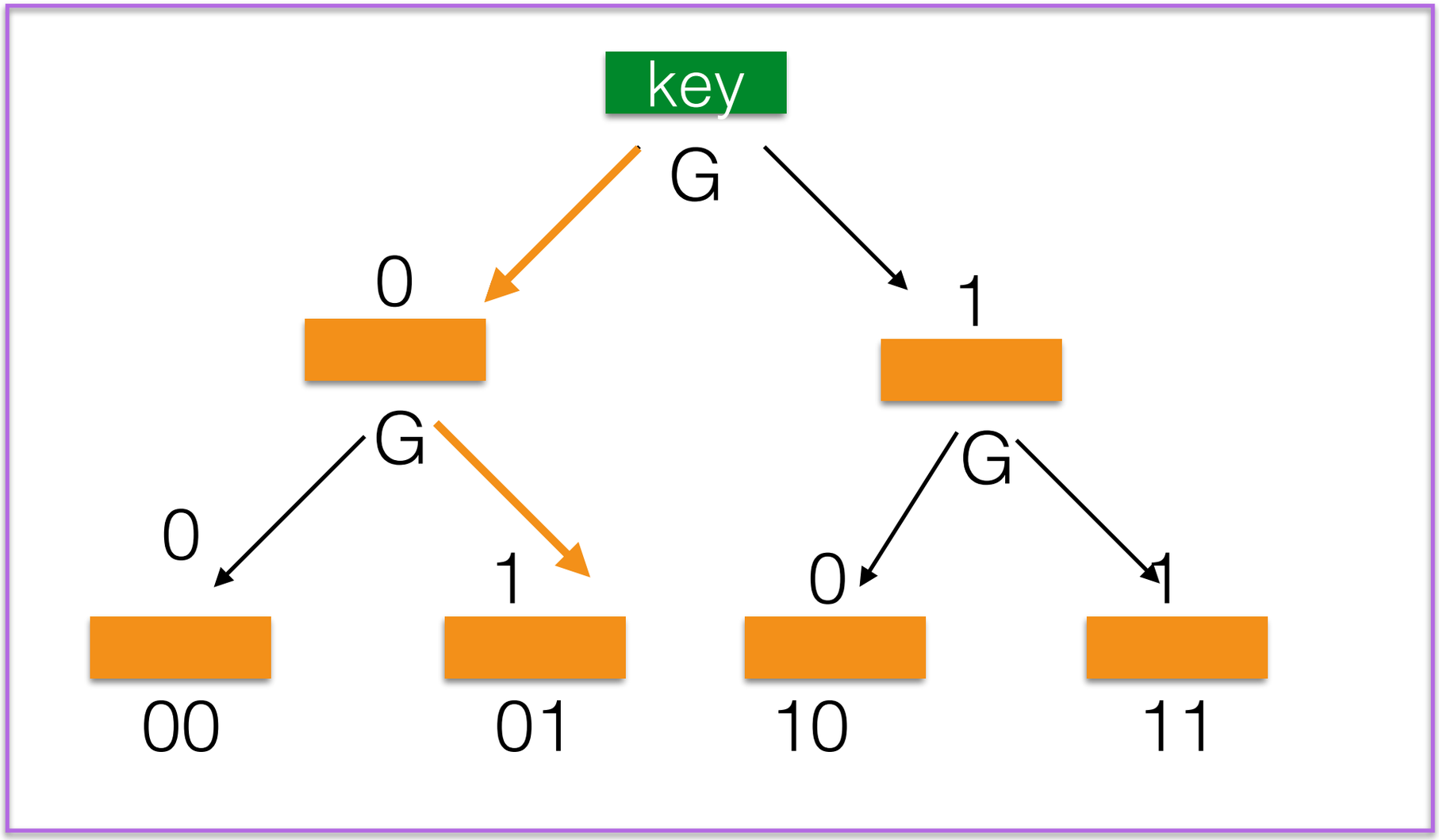
It can be extended for **n** bits

F(k, 01)



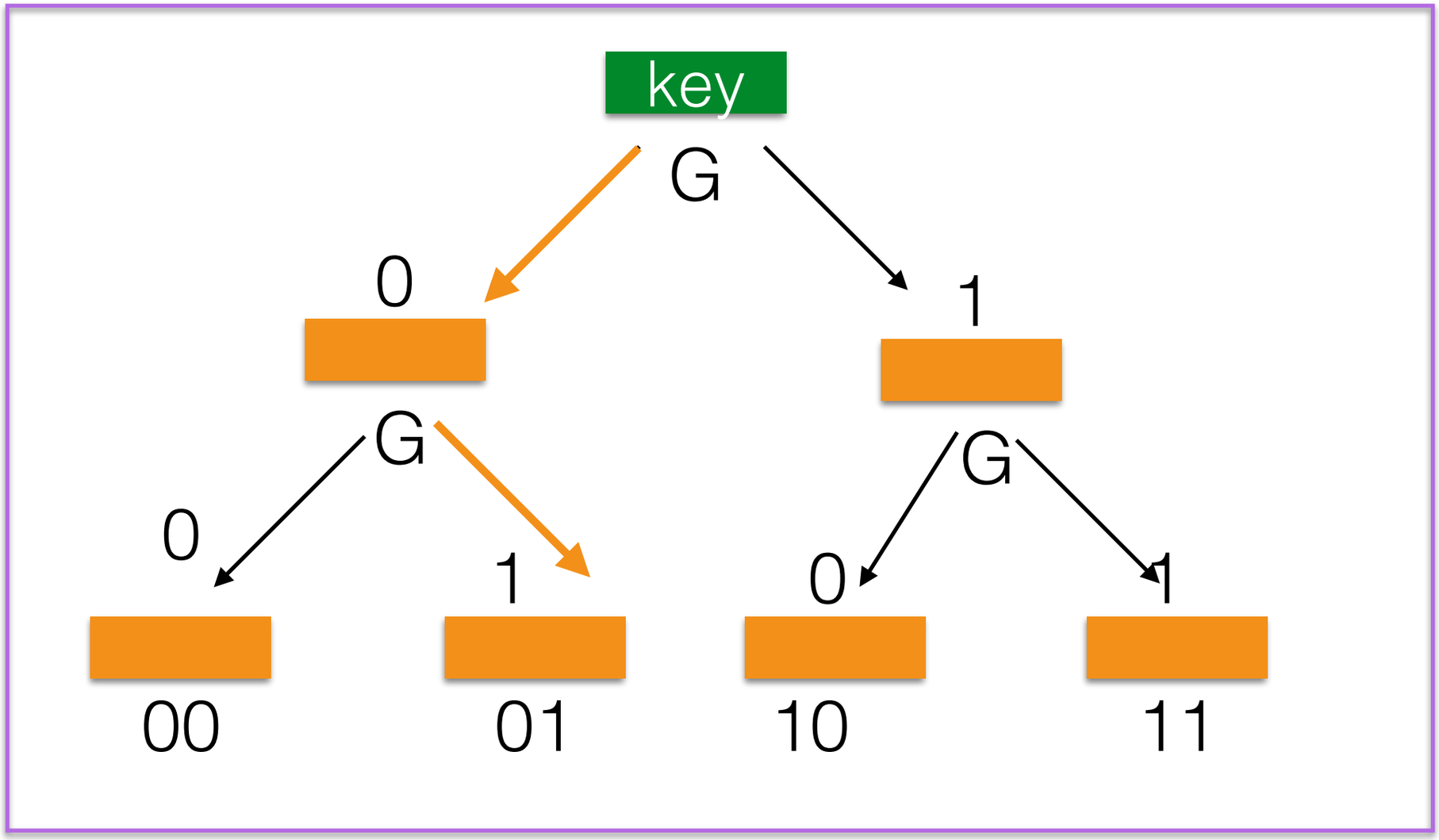
It can be extended for **n** bits

F(k, 01)



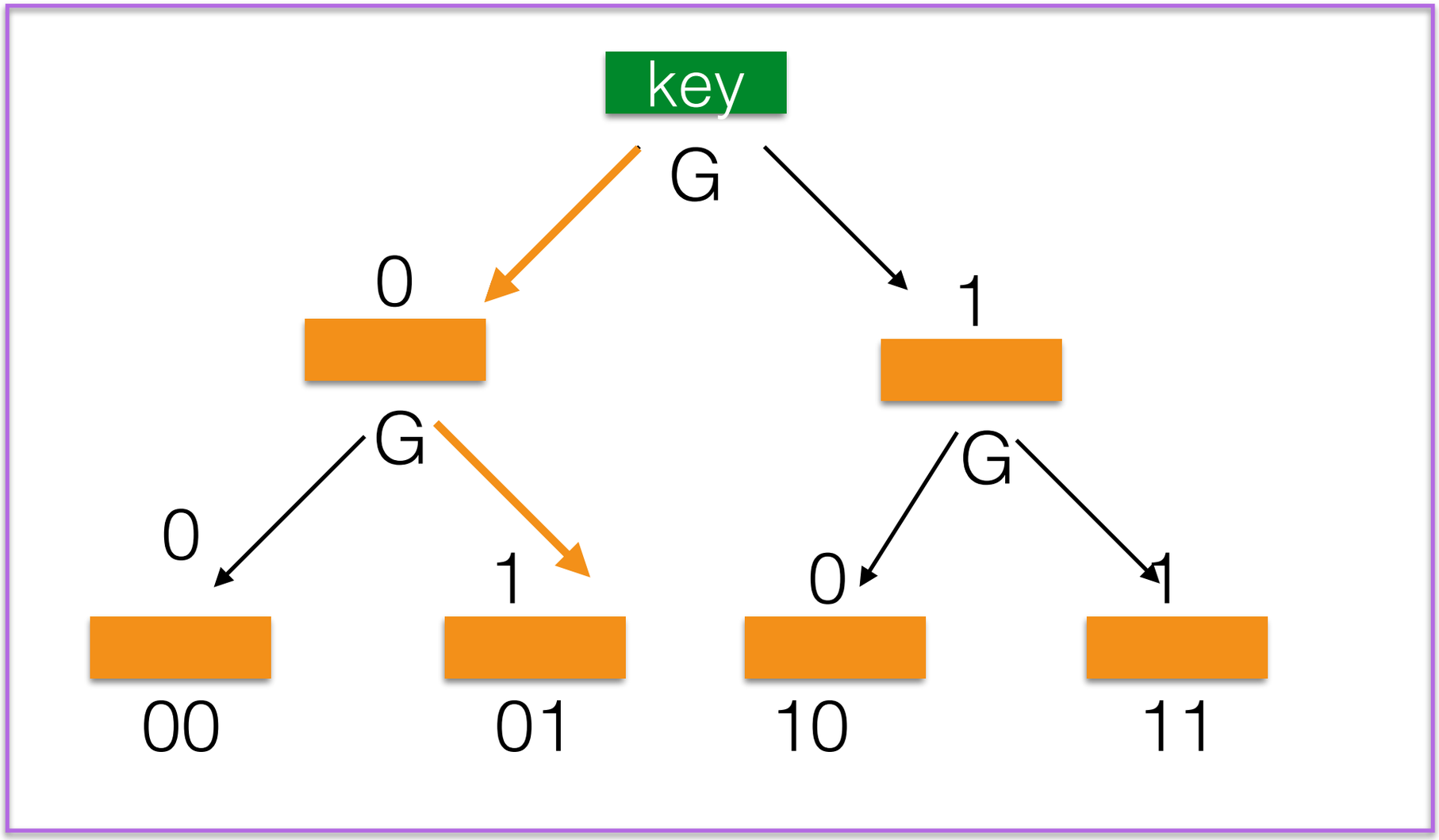
It can be extended for **n** bits

F(k, 01)



It can be extended for **n** bits

F(k, x)

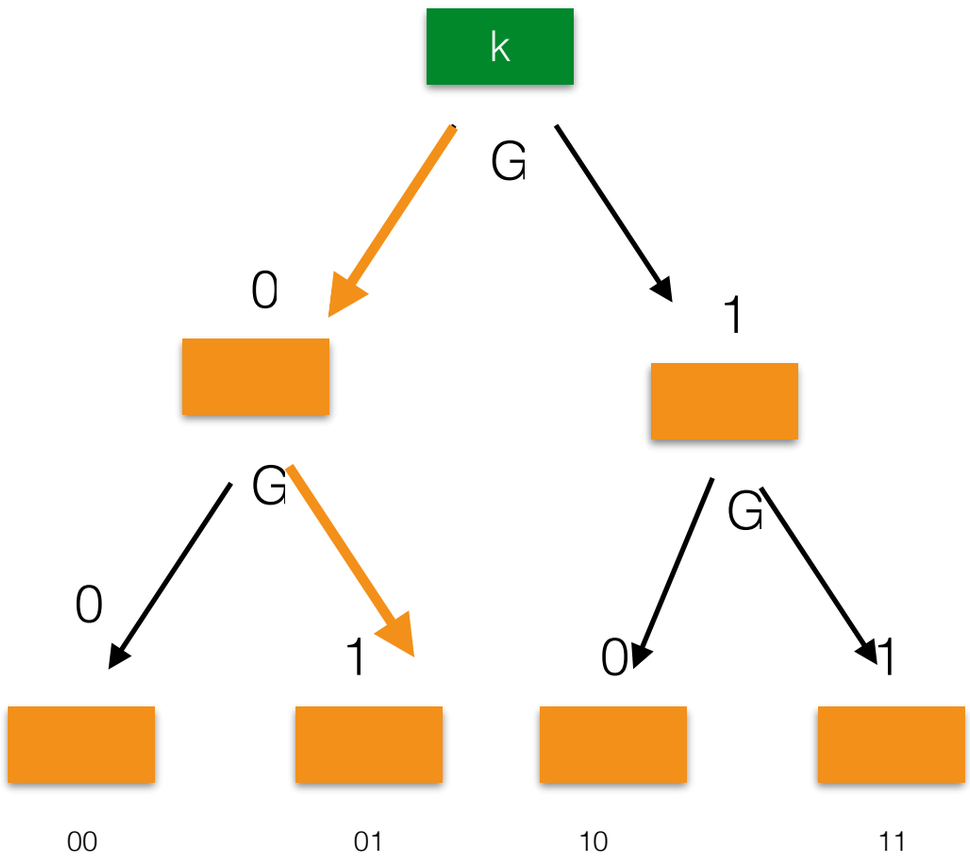


Theorem:

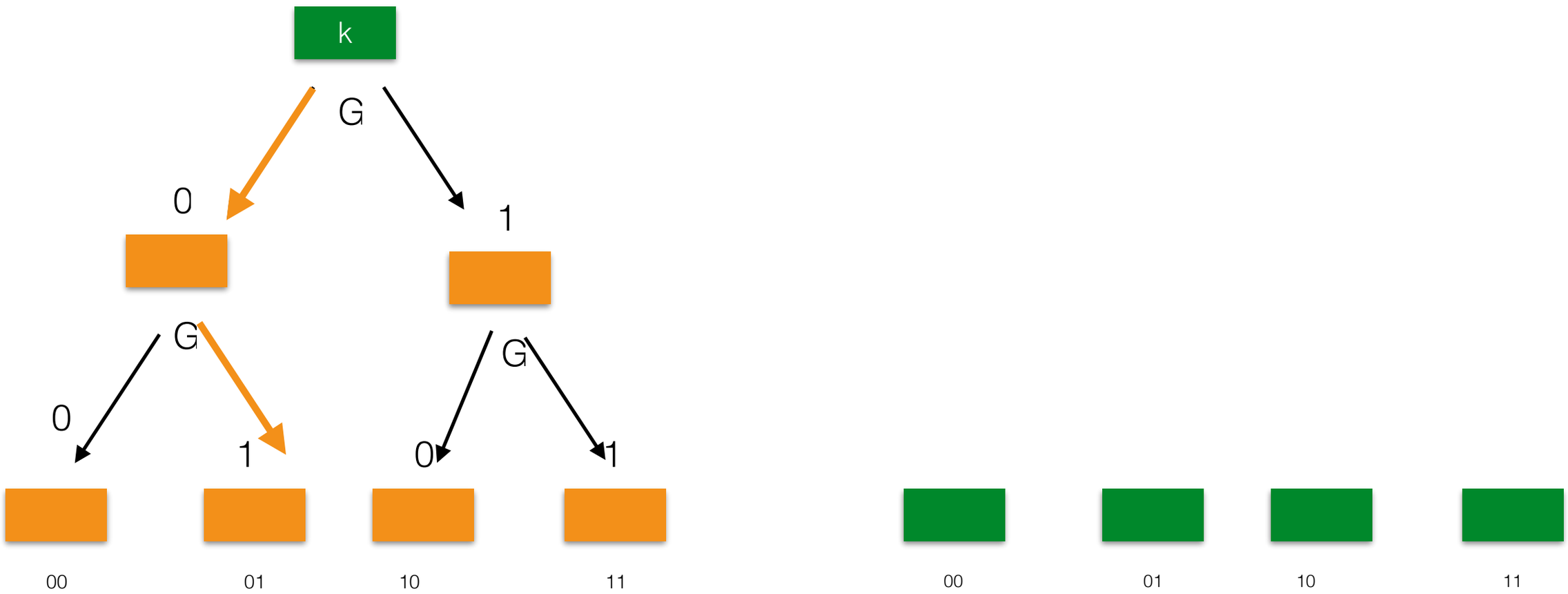
Assume that G is a PRG.

Then F is a Pseudo-random Function.

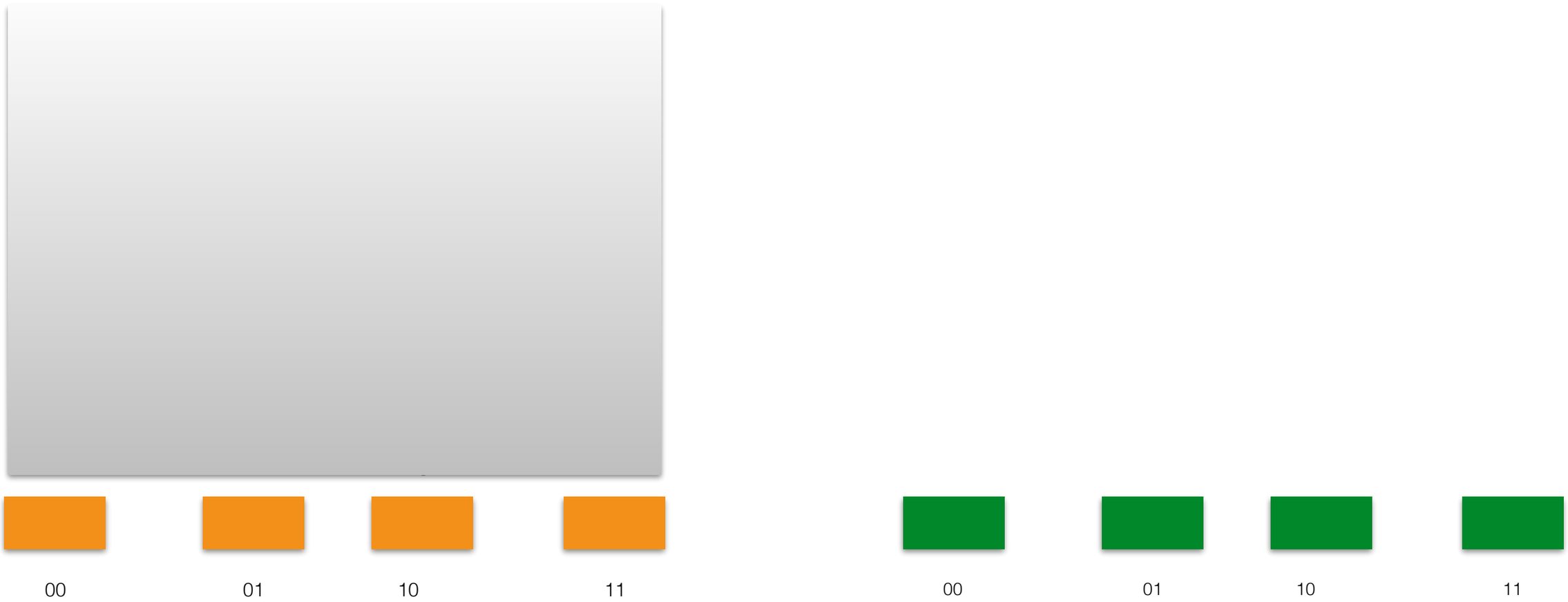
Proof: By hybrid arguments



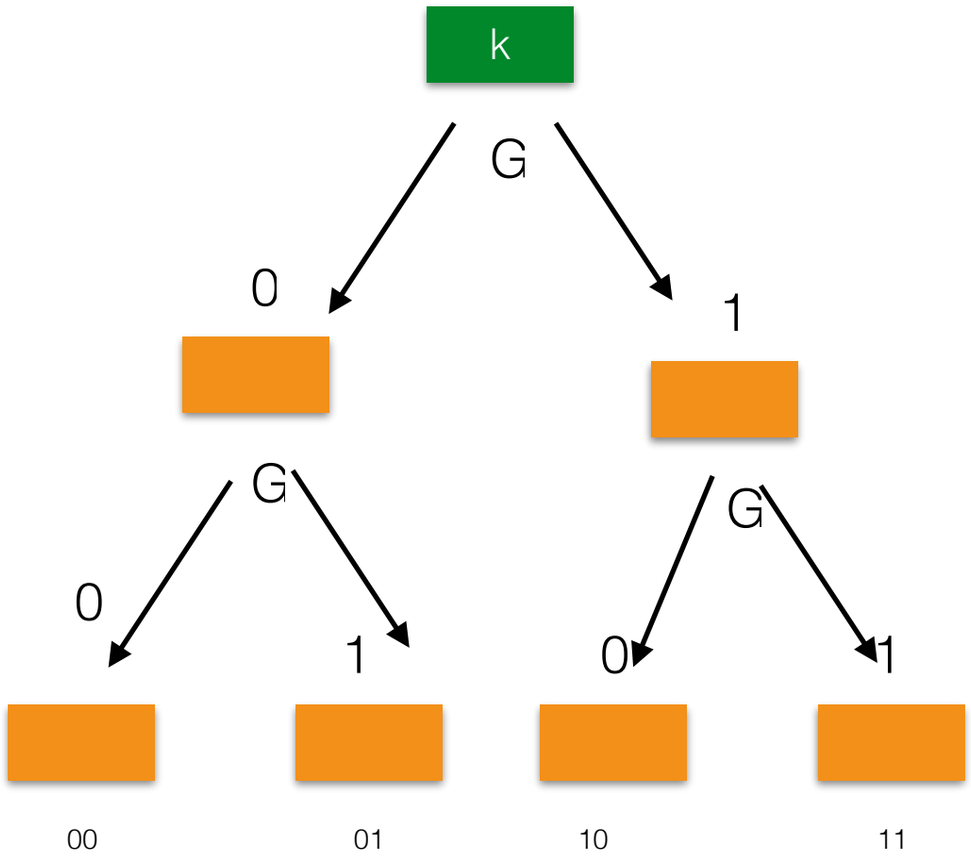
Proof: By hybrid arguments



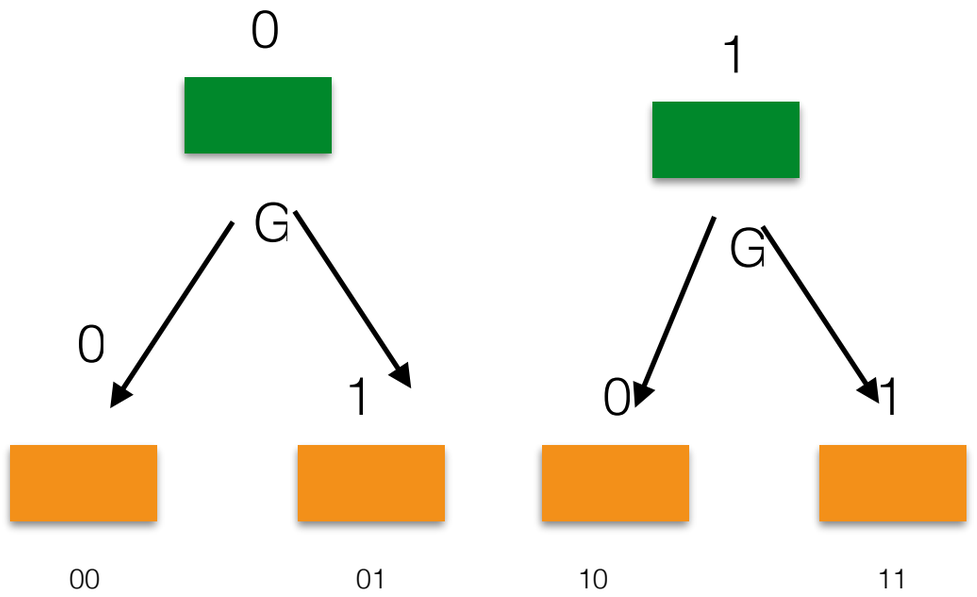
Proof: By hybrid arguments



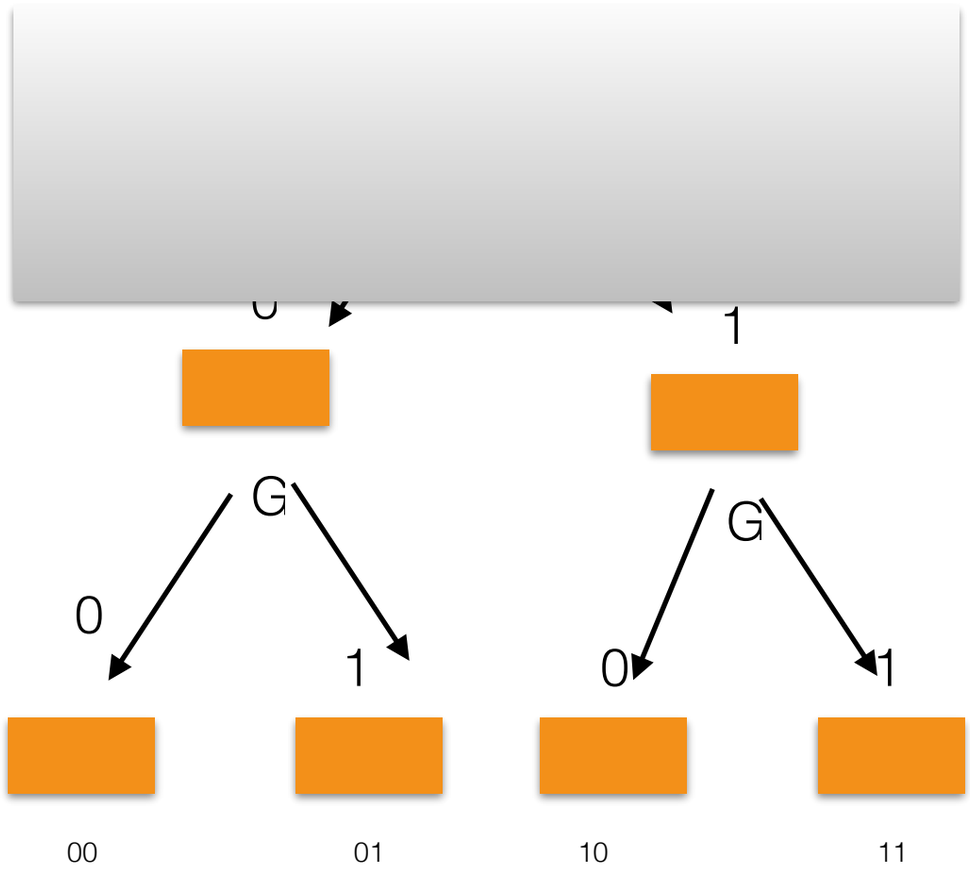
Proof: By hybrid arguments



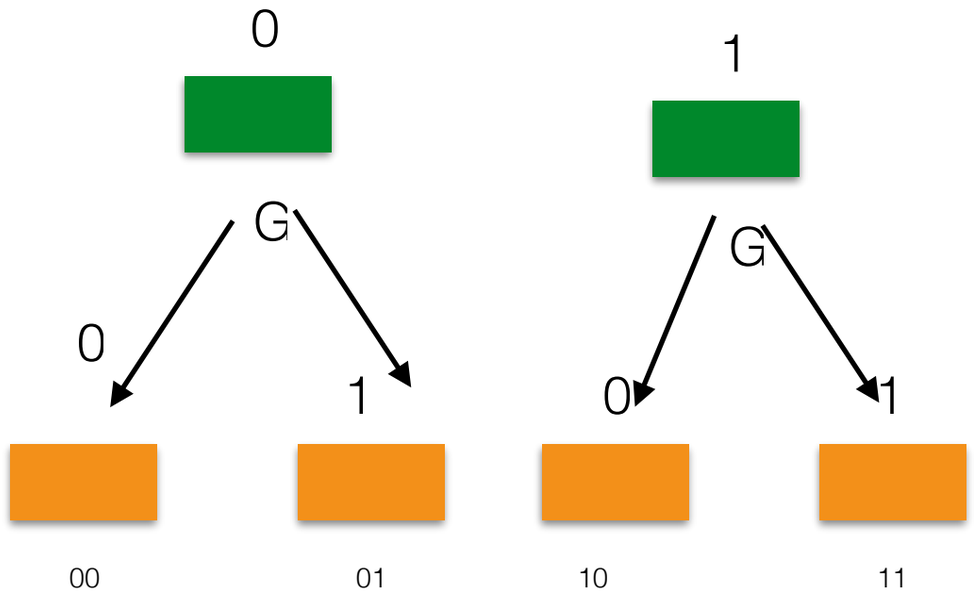
Hybrid game 1



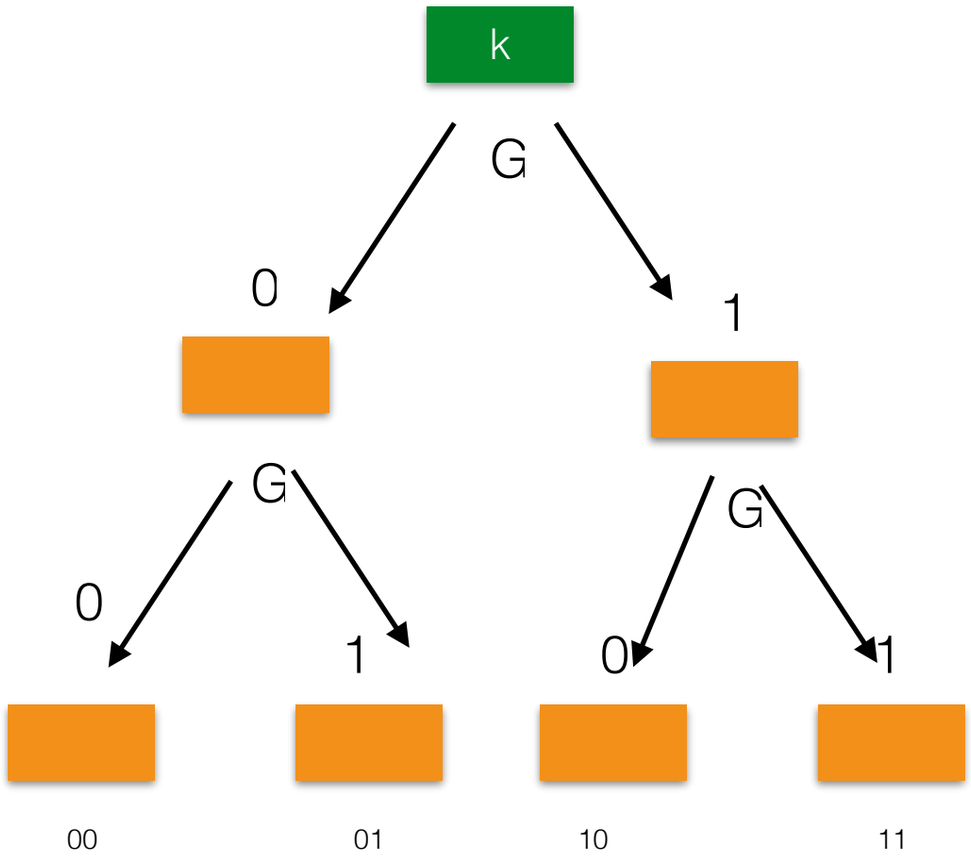
Proof: By hybrid arguments



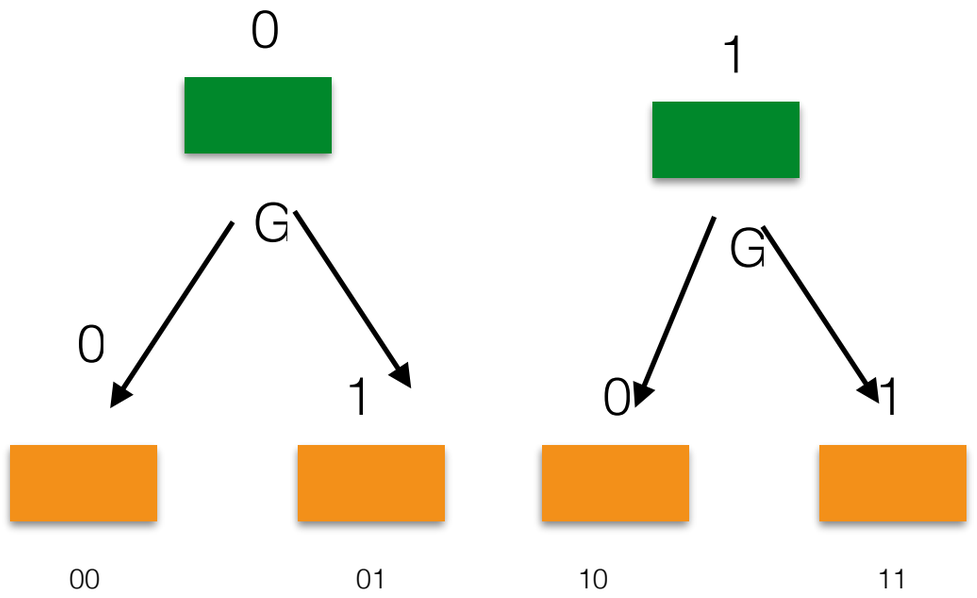
Hybrid game 1



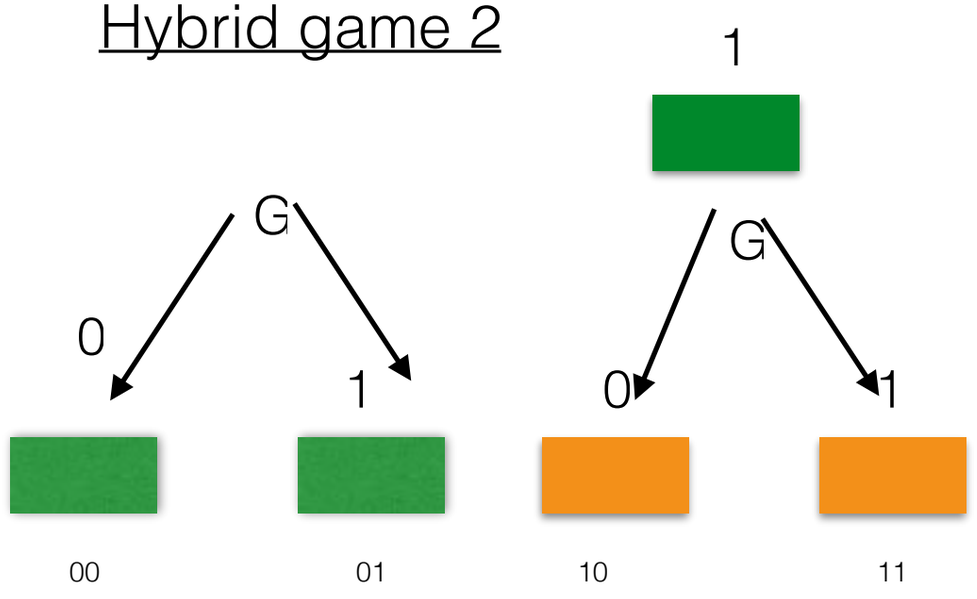
Proof: By hybrid arguments



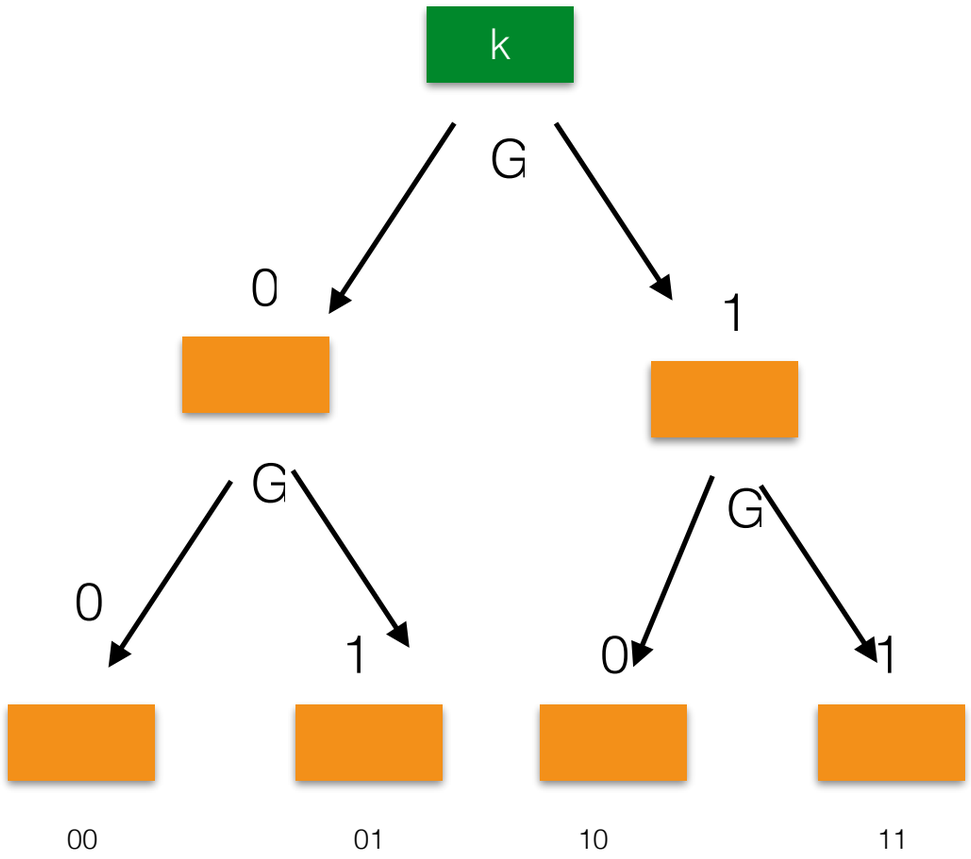
Hybrid game 1



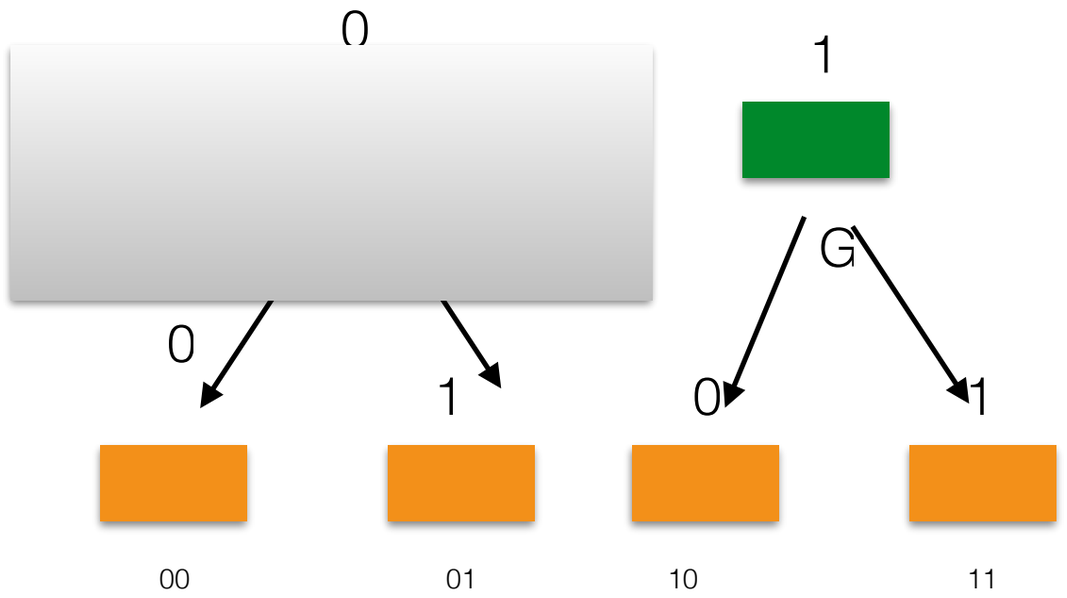
Hybrid game 2



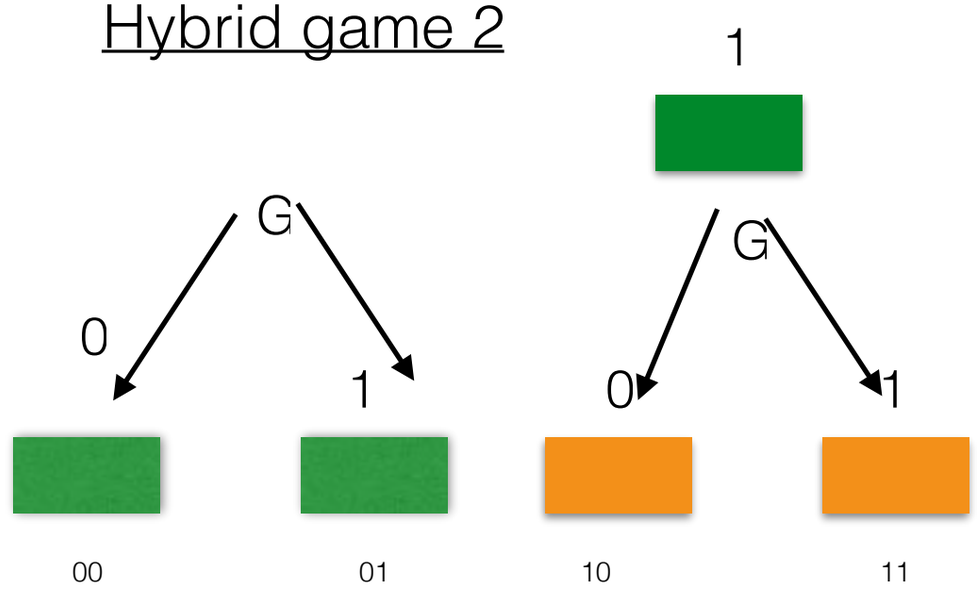
Proof: By hybrid arguments



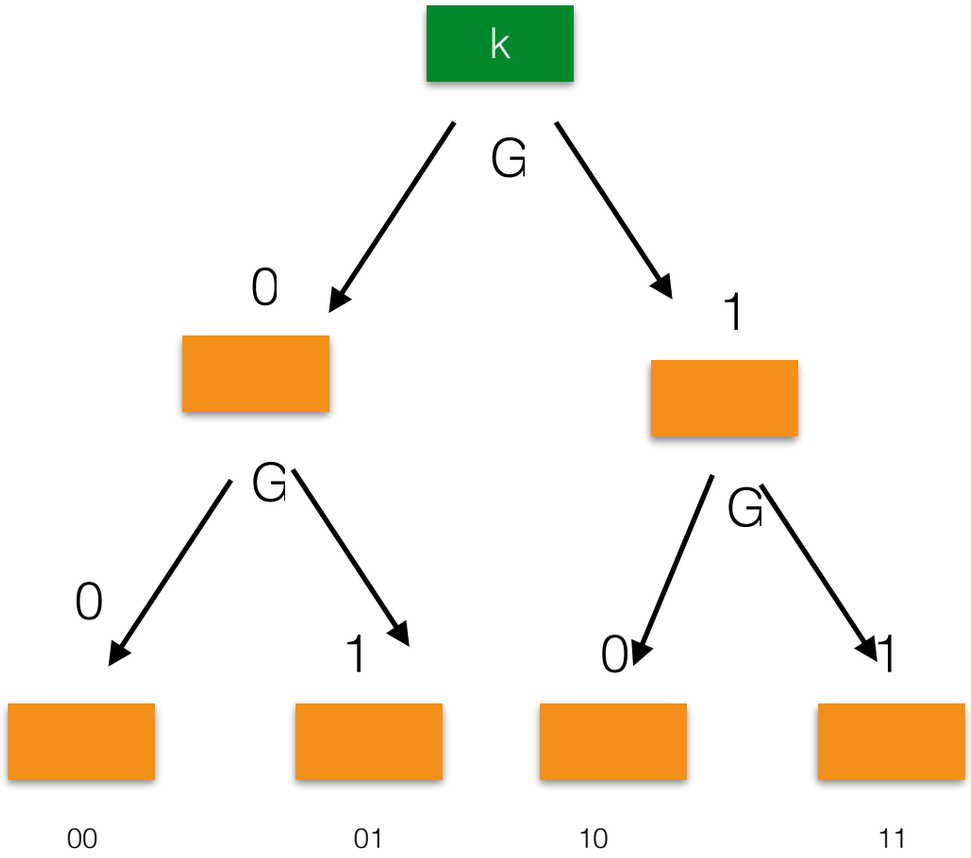
Hybrid game 1



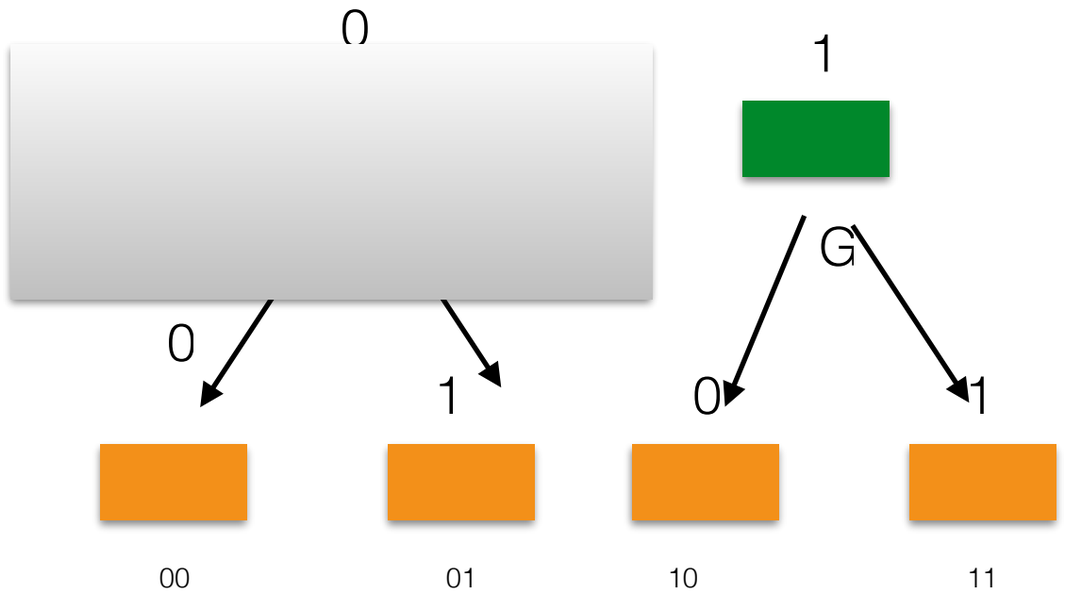
Hybrid game 2



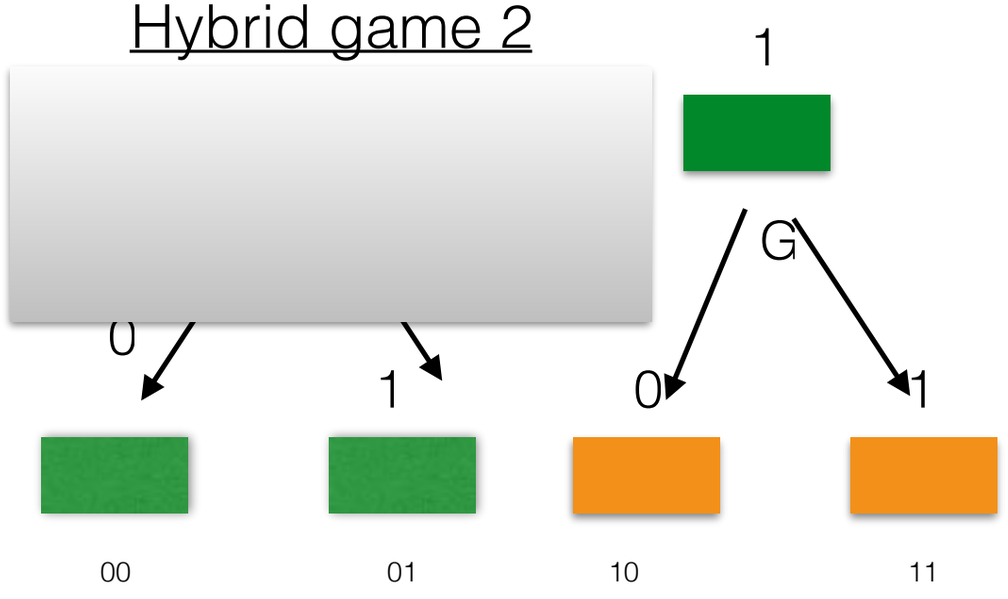
Proof: By hybrid arguments



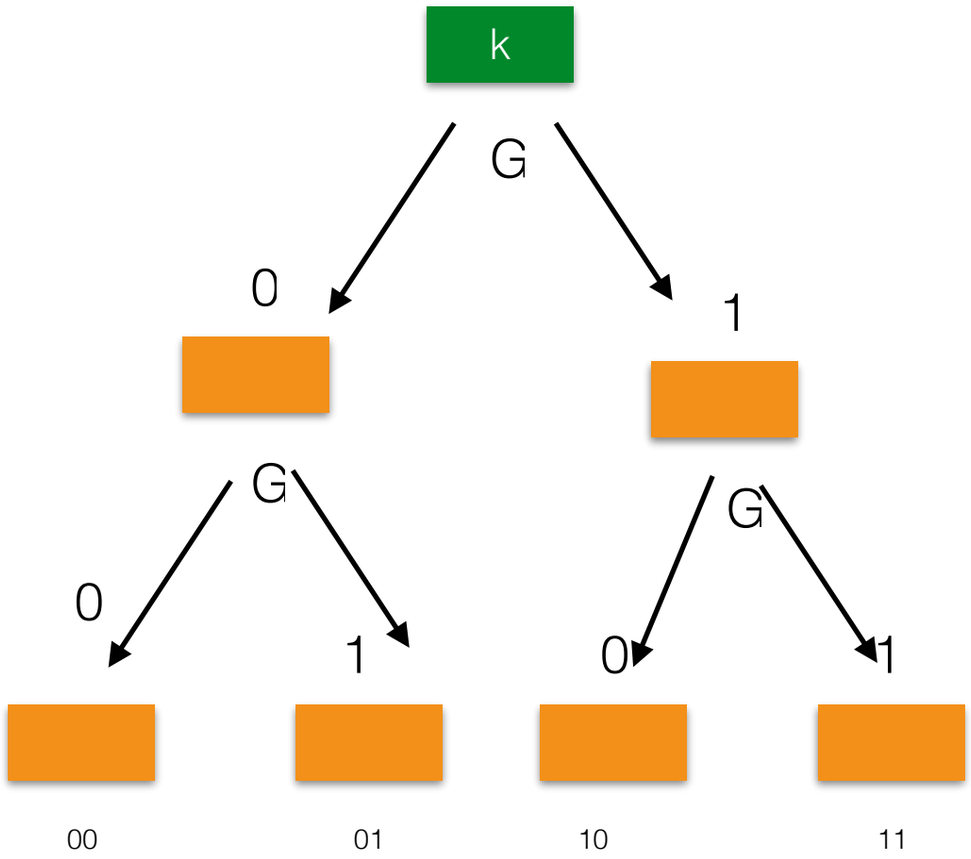
Hybrid game 1



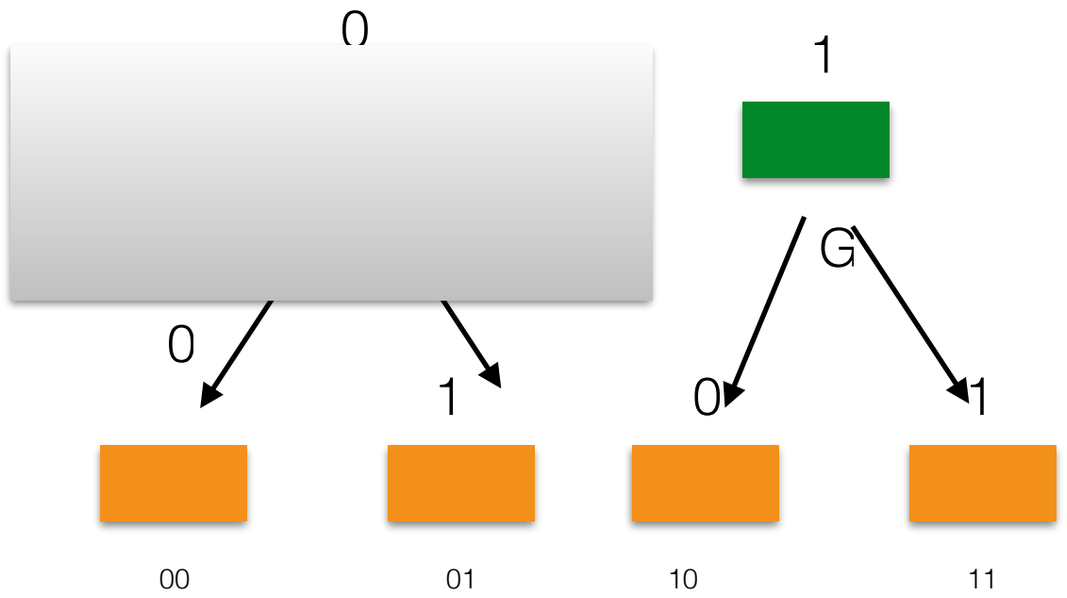
Hybrid game 2



Proof: By hybrid arguments

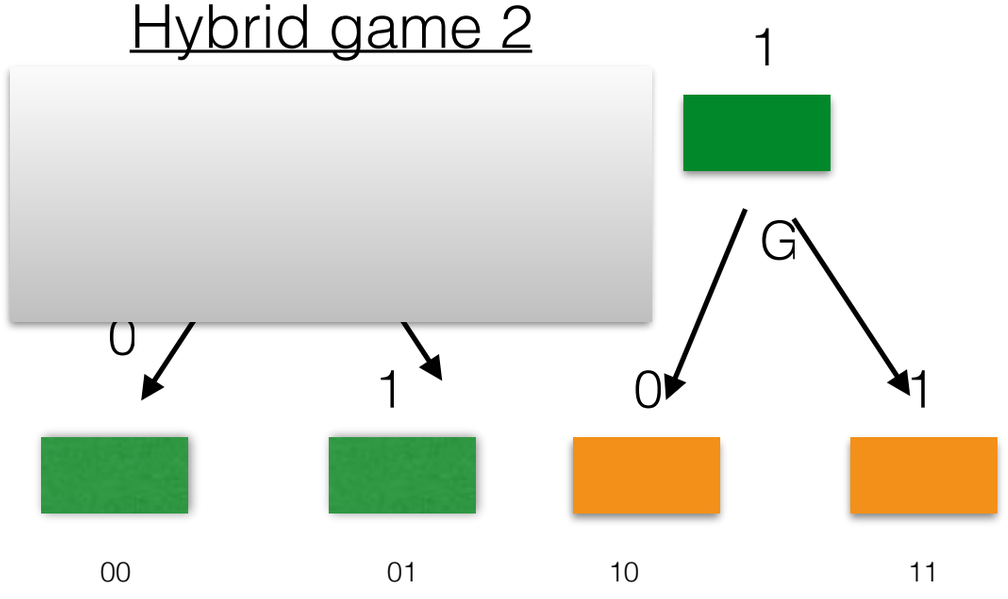


Hybrid game 1



How many hybrids for n bits?

Hybrid game 2



Today

- **Definition**

  Chosen Plaintext Attack

  (CPA) Security

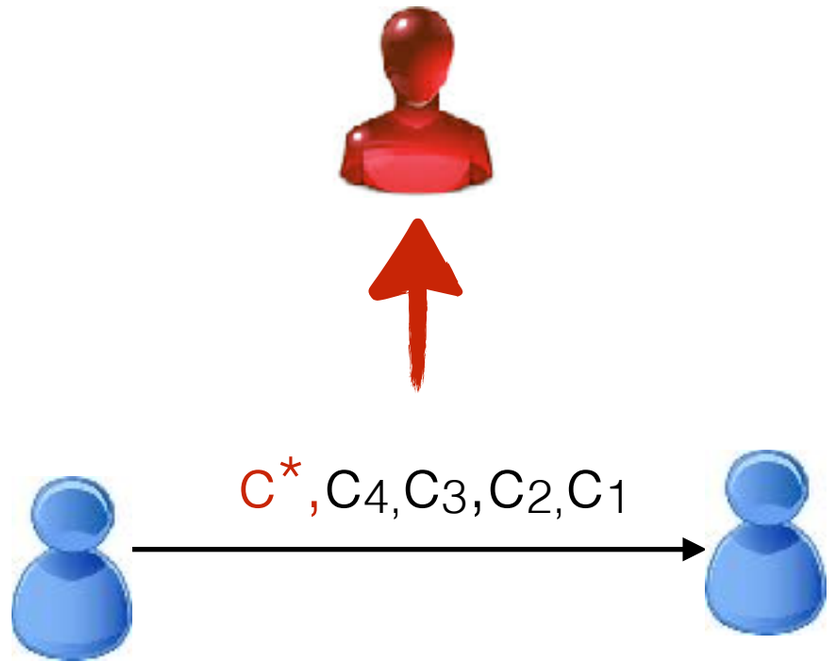
- **Assumptions**

  Pseudorandom Functions

  PRG —> PRF

- **Scheme + Proofs!**

new
realistic adversary



Today

- **Definition**

  Chosen Plaintext Attack

  (CPA) Security

- **Assumptions**

  Pseudorandom Functions

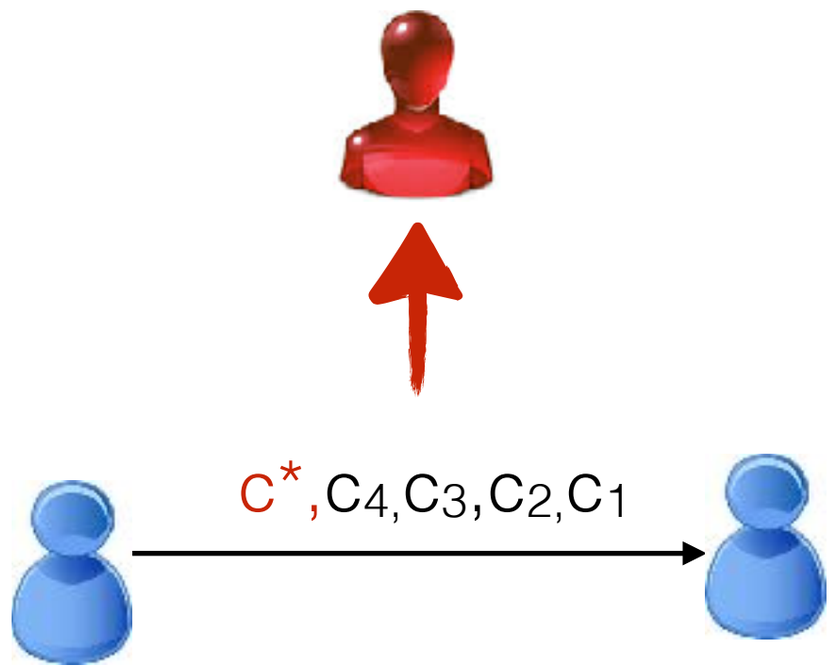
  PRG —> PRF

- **Scheme + Proofs!**

  Encryption scheme from PRF

pseudo- OTP (PRG)	—>	pseudo “many time” pads (PRF)

new
realistic adversary



- Tutorial on Proofs
- Homework 1 (rules)

NO CLASS next week