COS 433/Math 473: Cryptography

Mark Zhandry

Princeton University

Fall 2020

Announcements/Reminders

Last day to submit HW1

HW2 will be posted today

- Due September 29

PR1 Due October 6

Previously on COS 433…

Defining Pseudorandom Generator (PRG)

Syntax:

- Seed space $\mathbf{S_\lambda}$
- Output space $\mathbf{X_\lambda}$
- $\mathbf{G}$: $\mathbf{S_\lambda} \rightarrow \mathbf{X_\lambda}$ (deterministic)

Correctness:

- $|s|=\log|S_\lambda|$, $|x|=\log|X_\lambda|$ polynomial in λ,
- $|X_\lambda| > 2\times|S_\lambda|$
- Running time of $\mathbf{G}$ polynomial in λ

Security of PRGs

Definition: $G: S_\lambda \rightarrow X_\lambda$ is a **secure pseudorandom generator** (PRG) if:

- For all  running in polynomial time, $\exists$ negl **ε,**

$\Big|$ Pr[ **$(G(s))=1 : s \leftarrow S_\lambda]$**

$-$ Pr[ **$(x)=1 : x \leftarrow X_\lambda] \Big| \leq \varepsilon(\lambda)$**

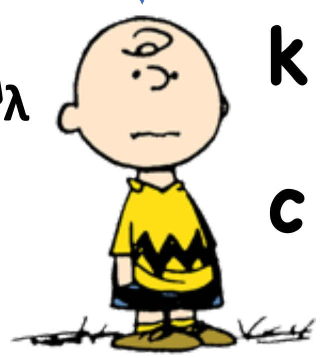
Security

Assume towards contradiction that there is a  and non-negligible **ε** such that

b

$m_0,\ m_1 \in M_\lambda$

c



$k \leftarrow S_\lambda$

$c \leftarrow G(k) \oplus m_b$

b'

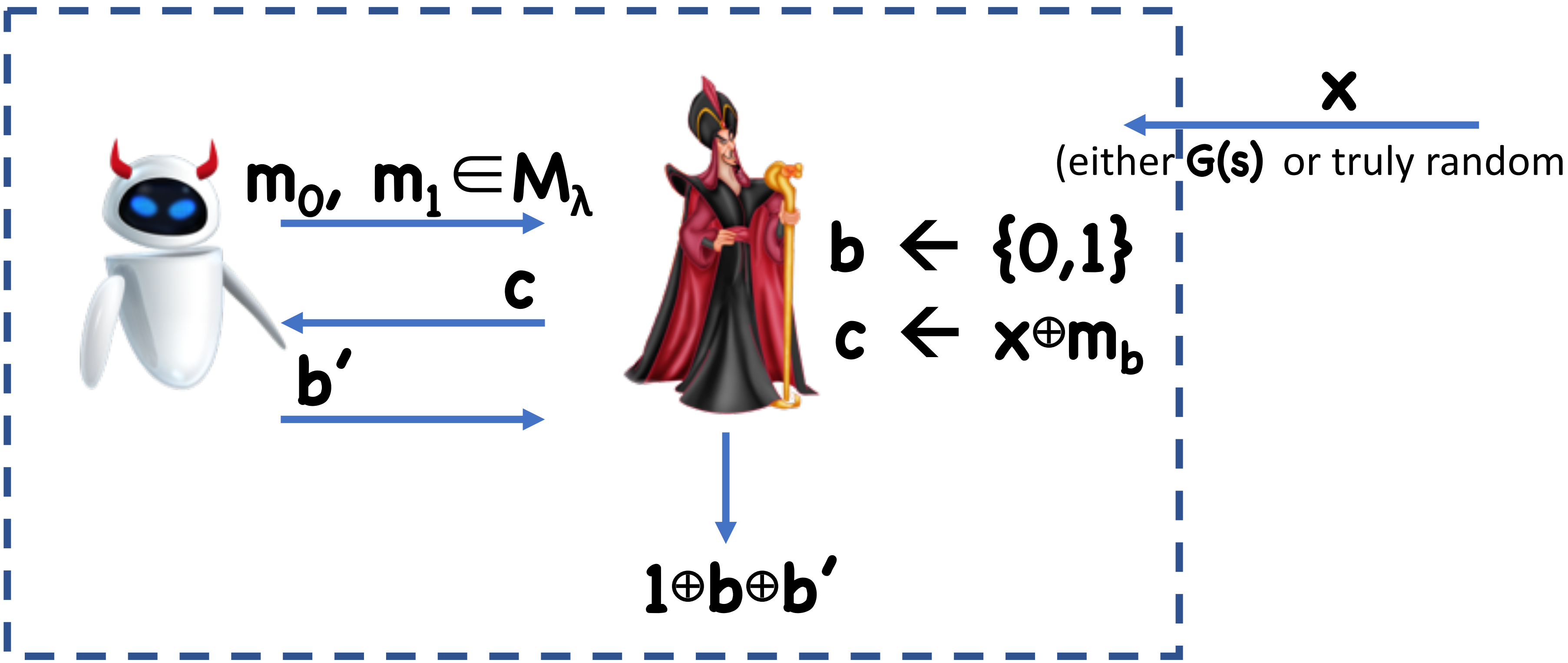
$|Pr[W_0] - Pr[W_1]| \geq \varepsilon$, non-negligible

W_b: $b' = 1$ in IND-Exp_b

Security

Use  to build .  will run  as a subroutine,

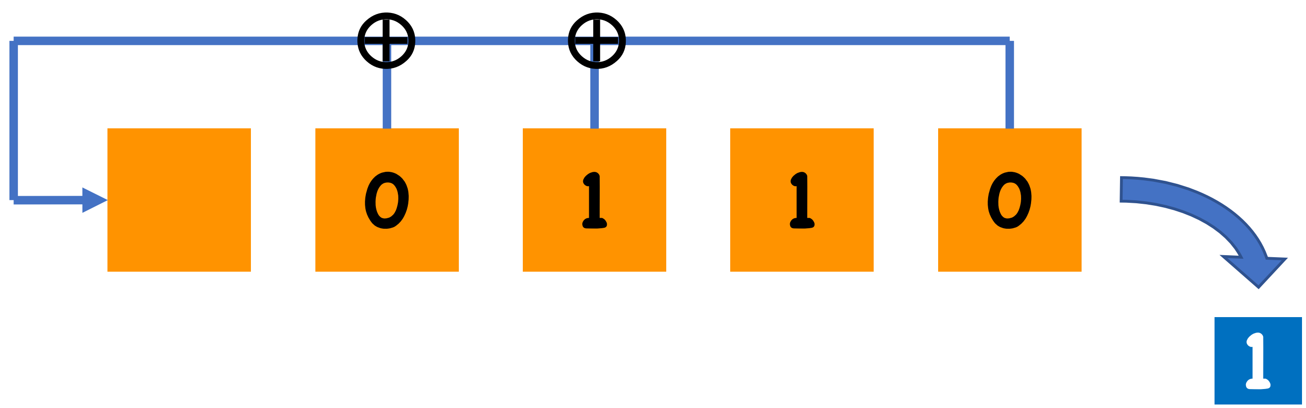
and pretend to be 



Insecure: Linear Feedback Shift Registers

In each step,

- Last bit of state is removed and outputted
- Rest of bits are shifted right
- First bit is XOR of subset of remaining bits



PRGs should be Unpredictable

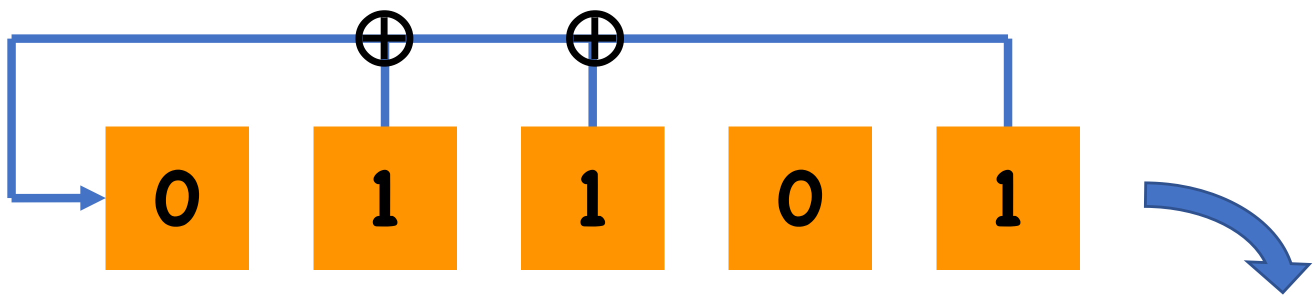
More generally, it should be hard, given some bits of output, to predict subsequent bits

Definition: $\mathbf{G:S_\lambda \rightarrow \{0,1\}^{n(\lambda)}}$ is **unpredictable** if, for all polynomial time  and any $\mathbf{p=p(\lambda)}$, $\exists$ negligible $\boldsymbol{\varepsilon}$ such that:

$\Big|\ \Pr[G(s)_{p+1} \leftarrow$  $(G(s)_{[1,p]})\ ] - ½\ \Big| \le \varepsilon(\lambda)$

Linearity

Problem: LFSR's are linear



$$\text{state}' = \begin{pmatrix} 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix} \bullet \text{state} \pmod 2$$

$$\text{output} = (0\ 0\ 0\ 0\ 1) \bullet \text{state} \pmod 2$$

LFSR period

Period = number of bits before state repeats

After one period, output sequence repeats

Therefore, should have extremely long period

- Ideally almost 2^λ
- Possible to design LFSR's with period $2^\lambda-1$

Today: Constructing Software PRGs

Hardware vs Software

PRGs based on LFSR's are very fast in hardware

Unfortunately, not easily amenable to software

RC4

Fast software based PRG

Resisted attack for several years

No longer considered secure, but still widely used

RC4

State = permutation on **[256]** plus two integers
- Permutation stored as **256**-byte array **S**

Init(16-byte **k)**:
- For **i=0,…,255**
  S[i] = i
- **j = 0**
- **For i=0,…,255**
  j = j + S[i] + k[i mod 16] (mod 256)
  Swap **S[i]** and **S[j]**
- Output **(S,0,0)**

RC4

GetBits(S,i,j):

- **i++ (mod 256)**
- **j+= S[i] (mod 256)**
- Swap **S[i]** and **S[j]**
- **t = S[i] + S[j] (mod 256)**
- Output **(S,i,j), S[t]**

New state   Next output byte

Insecurity of RC4

Second byte of output is slightly biased towards 0

- $\Pr[\text{second byte} = 0^8] \approx 2/256$
- Should be $1/256$

Means RC4 is not secure according to our definition

-  outputs 1 iff second byte is equal to 0^8
- Advantage: $\approx 1/256$

Not a serious attack in practice, but demonstrates some structural weakness

Insecurity of RC4

Possible to extend attack to actually recover the input **k** in some use cases

- The seed is set to **(IV, k)** for some initial value **IV**
- Encrypt messages as **RC4(IV,k)⊕m**
- Also give **IV** to attacker
- Cannot show security assuming RC4 is a PRG

Can be used to completely break WEP encryption standard

PRGs Today

LFSRs and RC4 should not be used for cryptographic purposes, though RC4 still widely used

As course goes on, will see more PRGs

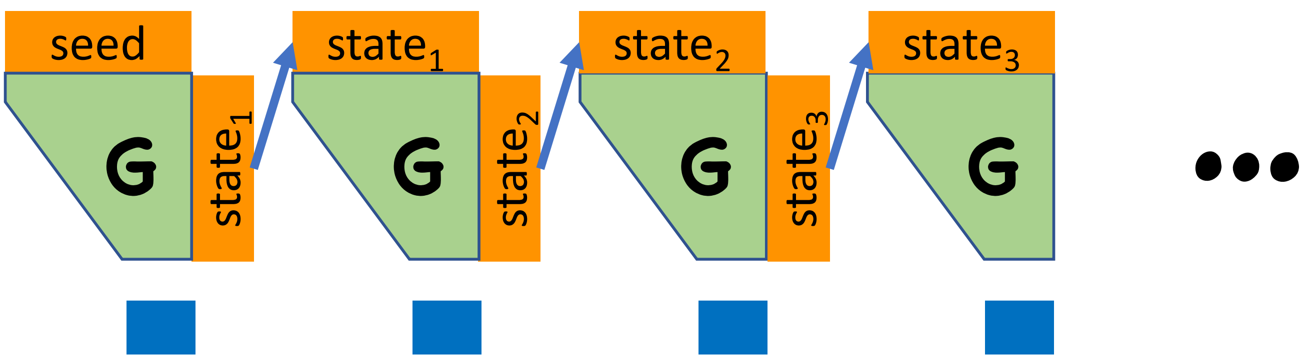
Length Extension for PRGs

Suppose I give you a PRG $\mathbf{G:\{0,1\}^{\lambda} \rightarrow \{0,1\}^{\lambda+1}}$

On it's own, not very useful: can only compress keys by 1 bit

But, we can use it to build PRGs with *arbitrarily-long* outputs!

Extending the Stretch of a PRG

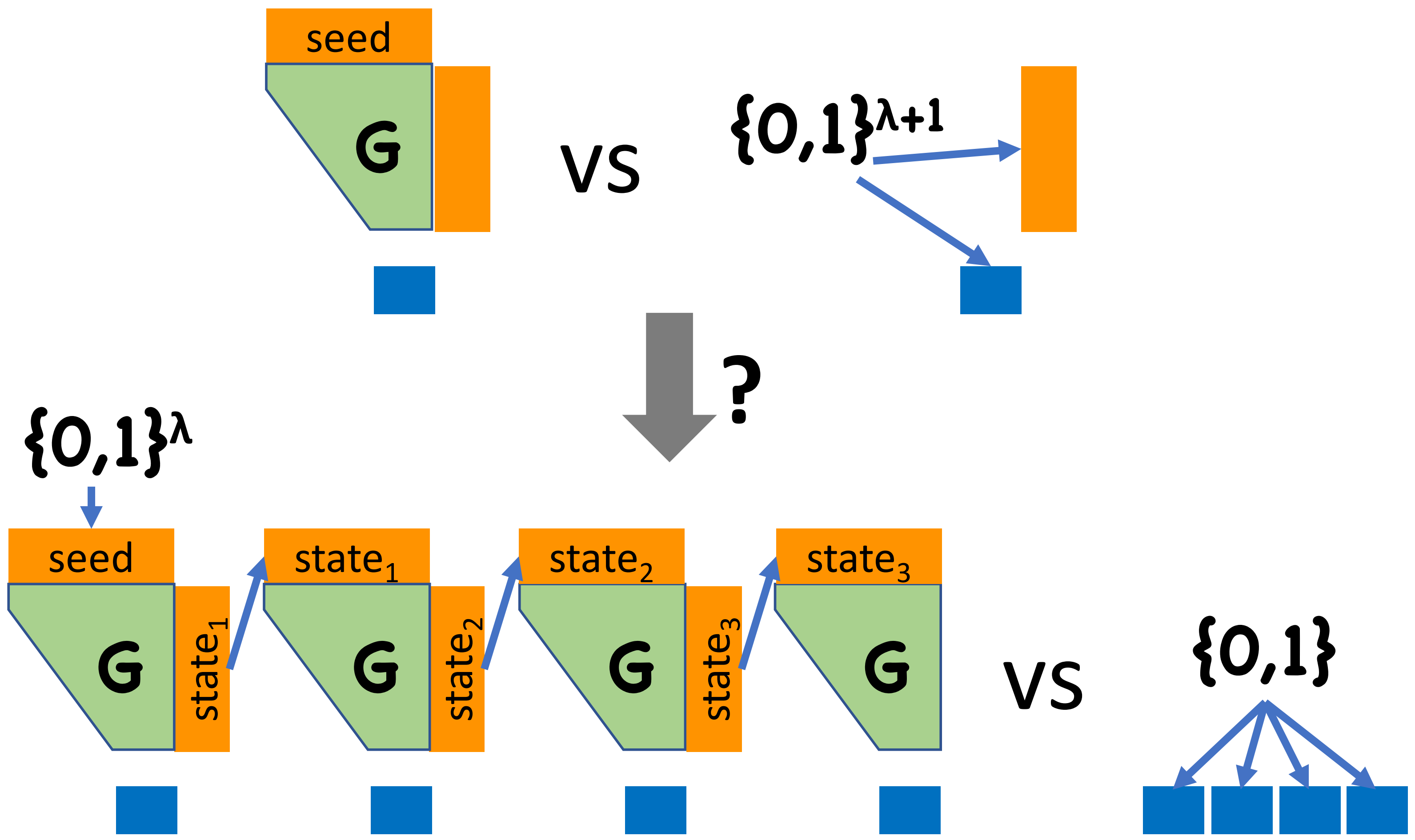


Security Proof

Assume towards contradiction  that breaks big PRG

Goal: build adversary  that breaks **G**

Problem?



Hybrid Arguments

Ubiquitous in crypto proofs

 distinguishes between two cases

- Call them $\mathbf{H_0}$ and $\mathbf{H_t}$

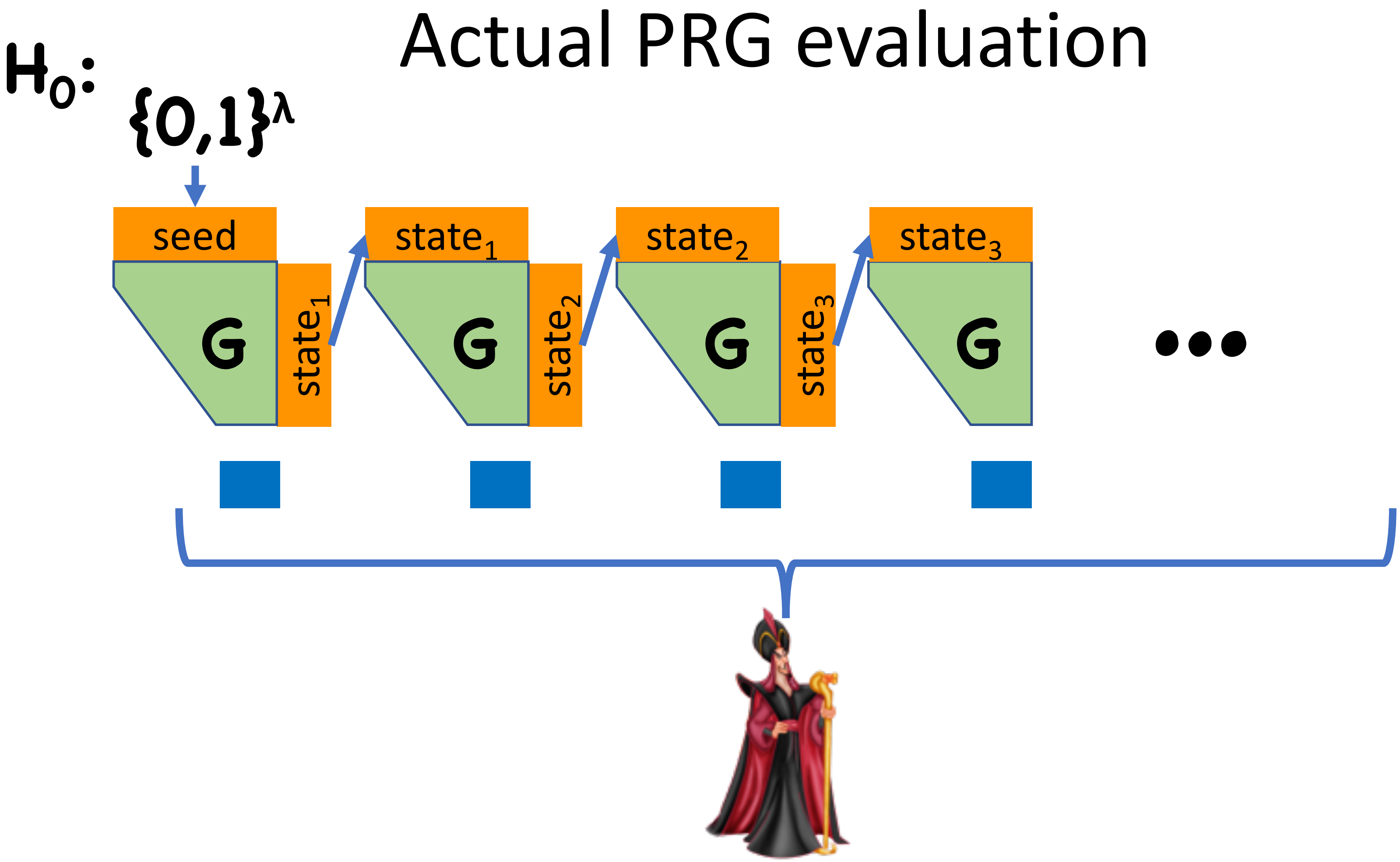
Devise intermediate experiments $\mathbf{H_1,\ldots,H_{t-1}}$ that "interpolate" between $\mathbf{H_0}$ and $\mathbf{H_t}$

- Only change one thing at a time

Use triangle inequality to conclude that  distinguishes $\mathbf{H_{i-1}}$ and $\mathbf{H_i}$

- Use such a distinguisher to build 

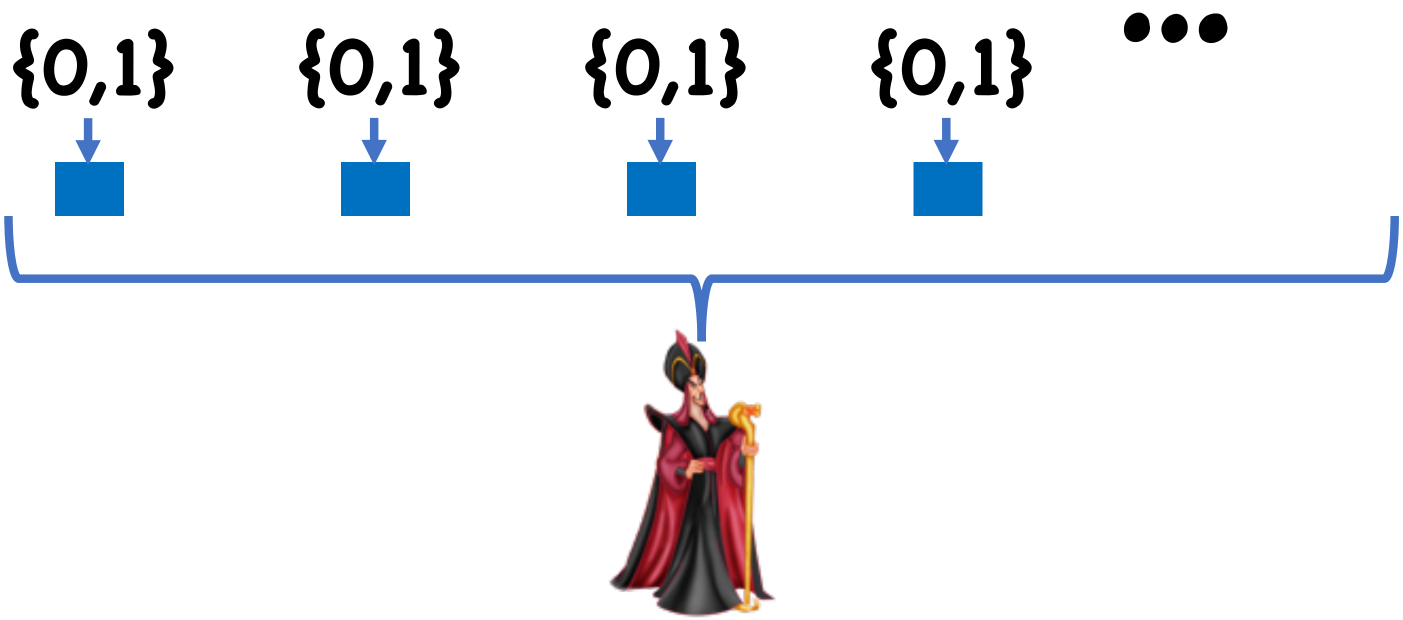
Proof by Hybrids



Security Proof

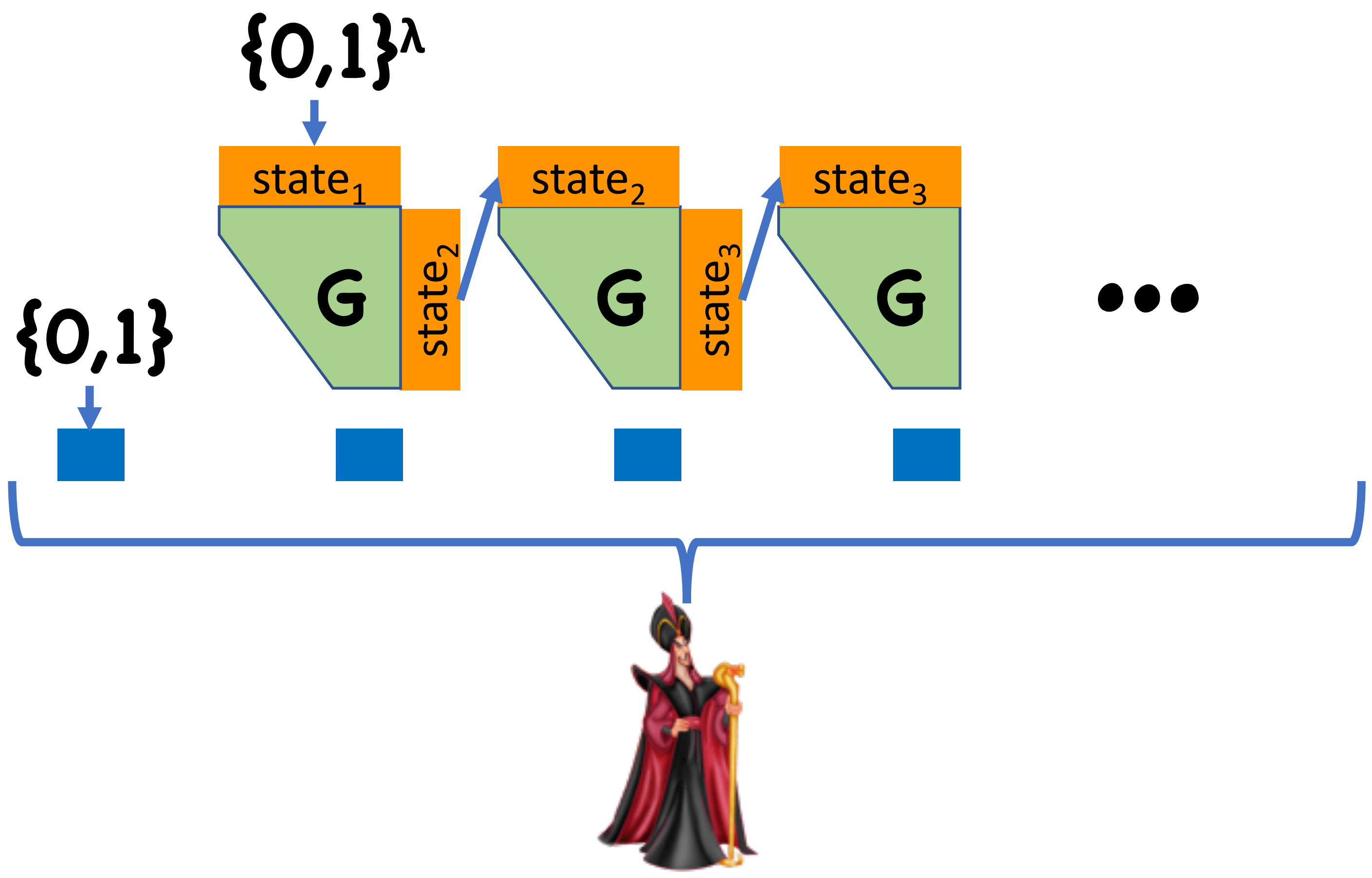
H_t:

Truly Random Values



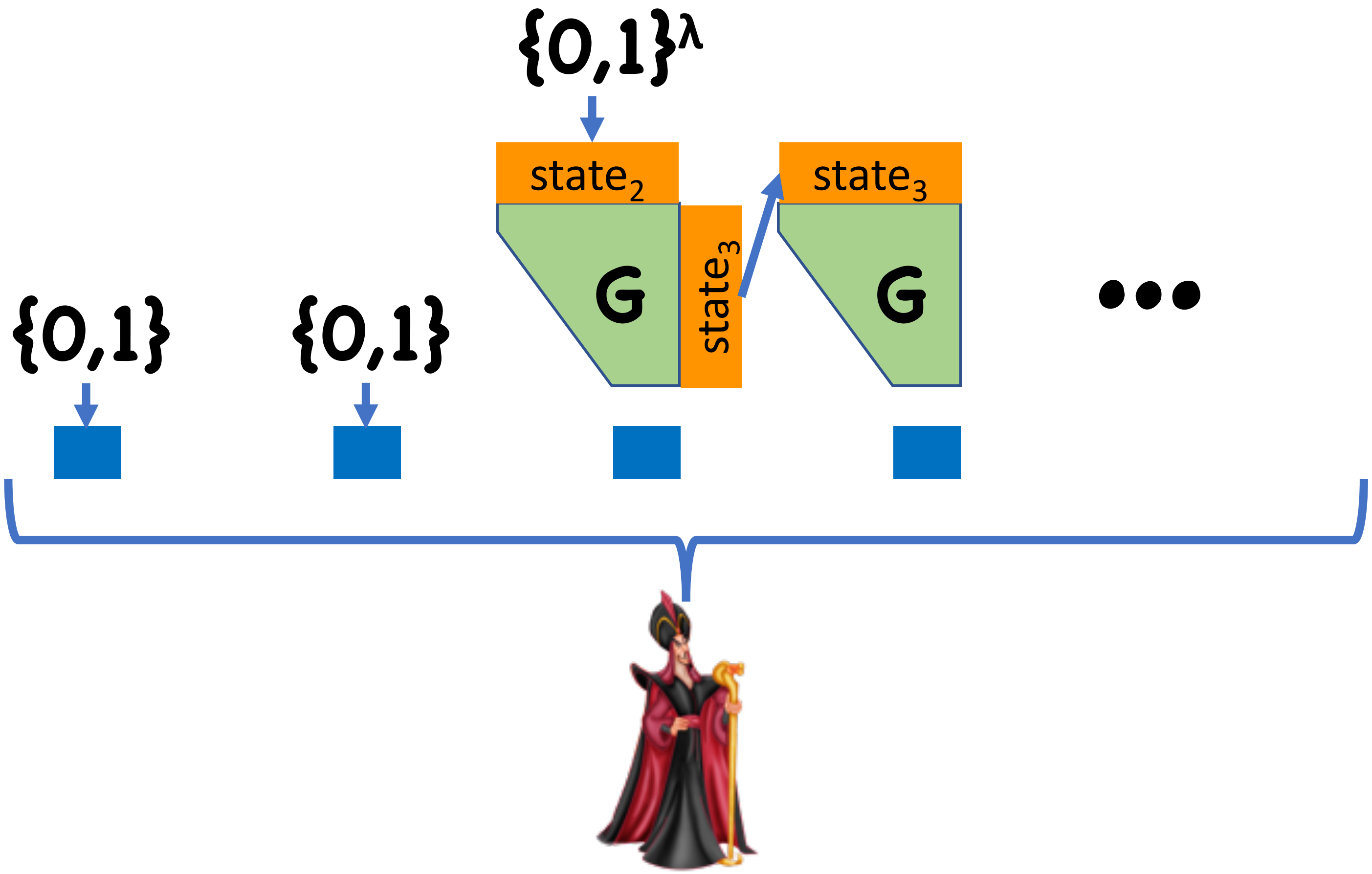
Security Proof

H_1:



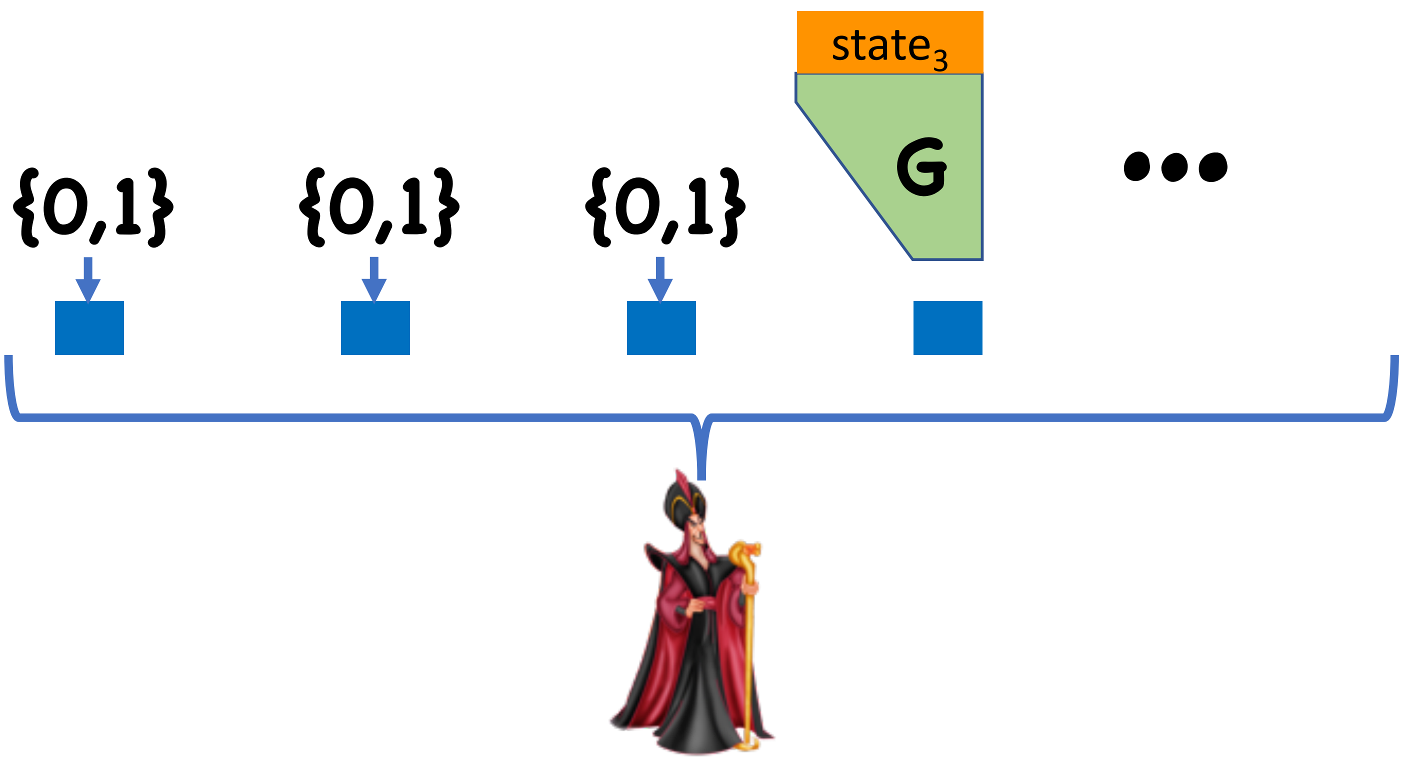
Security Proof

H_2:



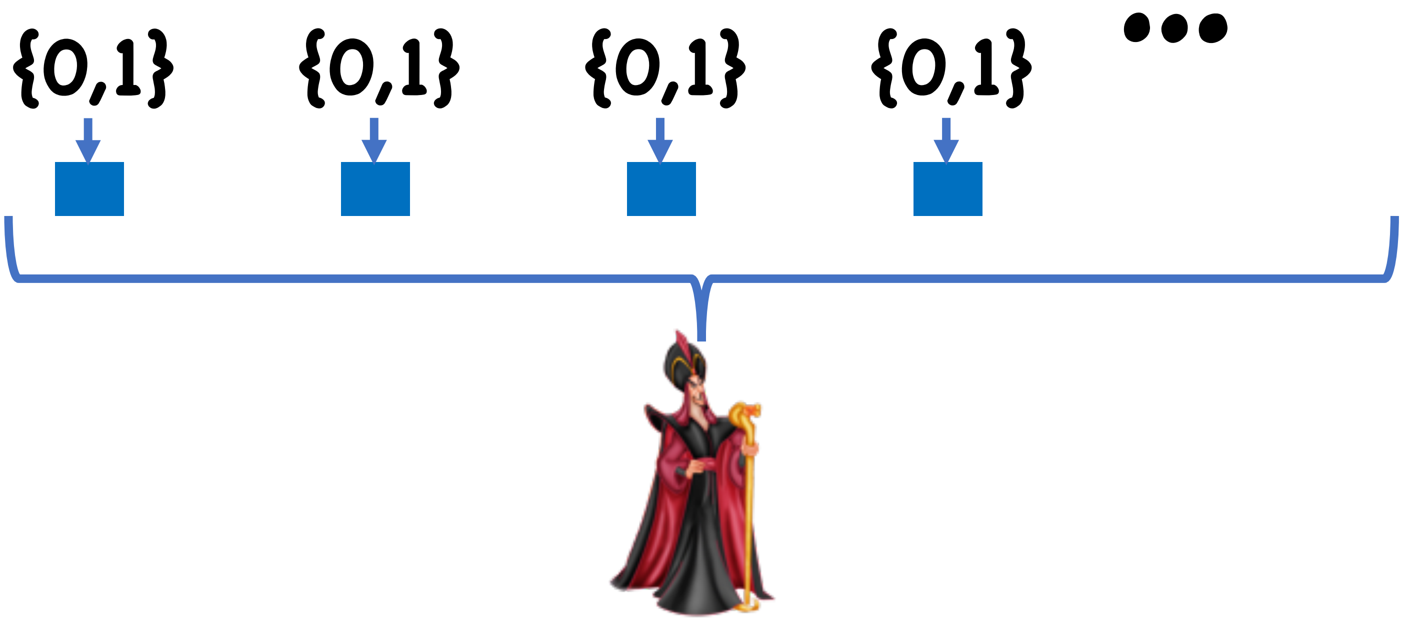
Security Proof

H_2:



Security Proof

$\mathbf{H_t}$:



Security Proof

$\mathbf{H_0}$ corresponds to pseudorandom $\mathbf{x}$
$\mathbf{H_t}$ corresponds to truly random $\mathbf{x}$

Let $\mathbf{q_i = Pr[}$  $\mathbf{(x)=1 : x \leftarrow H_i\ ]}$

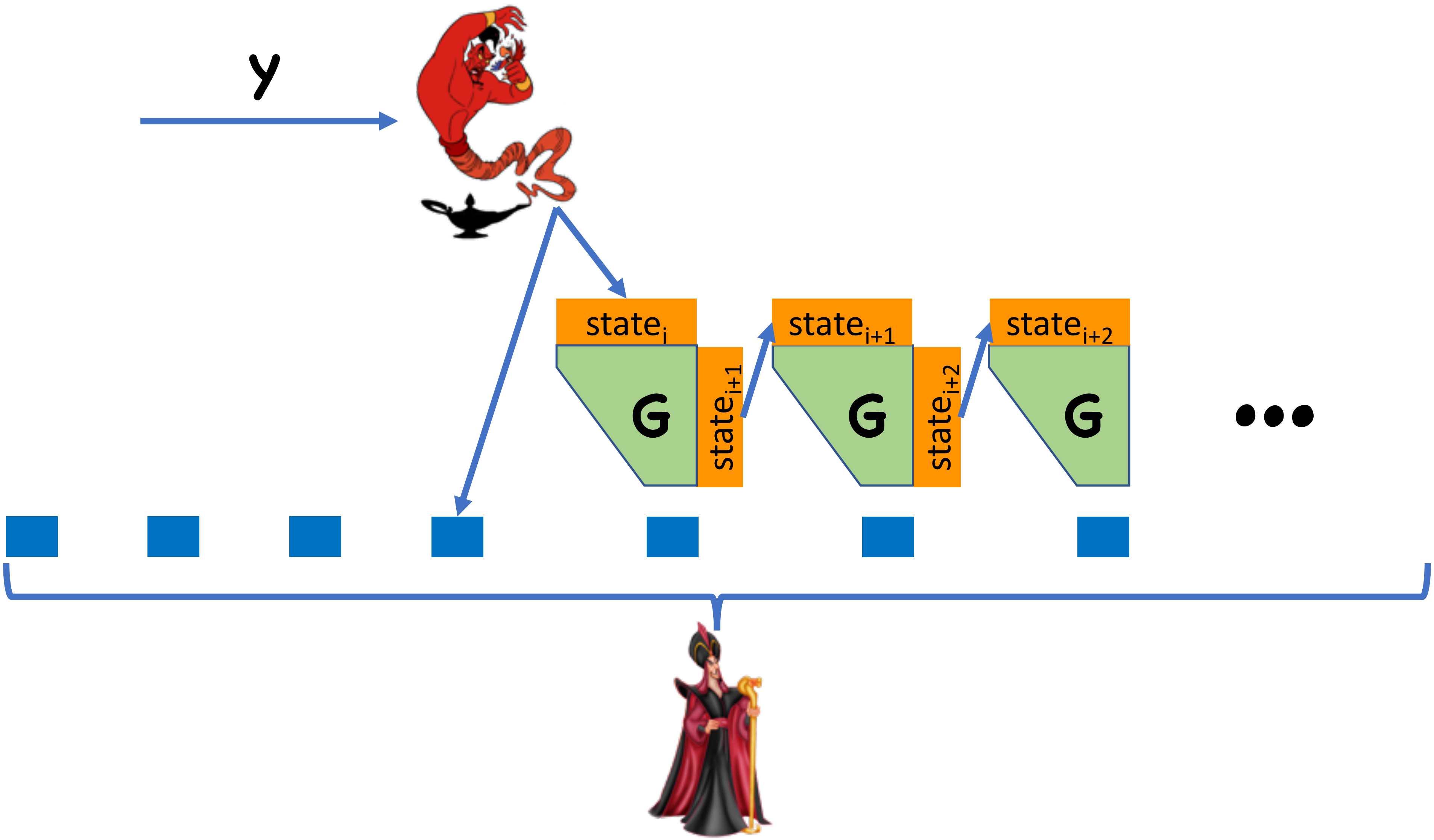
By assumption, $|q_t - q_0| > \varepsilon$

Triangle ineq:

$$|q_t - q_0| \leq |q_1 - q_0| + |q_2 - q_1| + \ldots + |q_t - q_{t-1}|$$

$\Rightarrow\ \exists\, \mathbf{i}$ s.t. $|q_i - q_{i-1}| > \varepsilon/t$

Security Proof



Security Proof

Analysis

- If **y = G(s),** then  sees $\mathbf{H}_{i-1}$

  $\Rightarrow$ **Pr[**  **outputs 1] =** q_{i-1}

  $\Rightarrow$ **Pr[**  **outputs 1] =** q_{i-1}

- If **y** is random, then  sees $\mathbf{H}_{i}$

  $\Rightarrow$ **Pr[**  **outputs 1] =** q_{i}

  $\Rightarrow$ **Pr[**  **outputs 1] =** q_{i}

Hybrids Recap

Useful whenever you can’t directly map between experiments

Only change one thing at a time, change corresponds to security of building block

- Not always obvious what hybrid sequence should be

Summary So Far

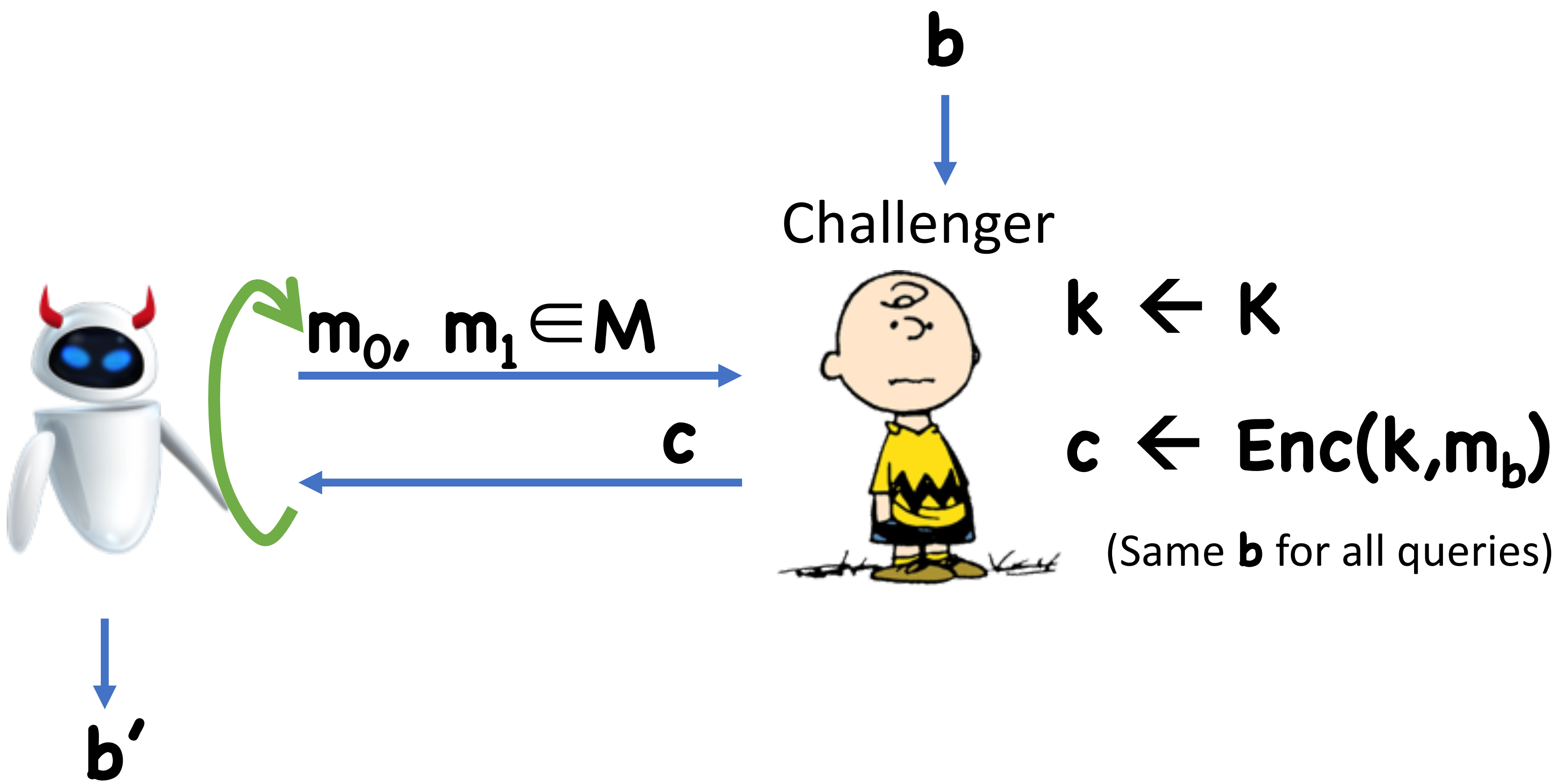
Stream ciphers = Encrytpion with PRG

- Secure encryption for arbitrary length, number of messages (though we did not completely prove it)

However, implementation difficulties due to having to maintaining state

Multiple Message Security

Left-or-Right Experiment



$\mathrm{LoR\text{-}Exp}_b($  $, \lambda)$

LoR Security Definition

Definition: (Enc, Dec) has **Left-or-Right indistinguishability** if, for all  running in polynomial time, $\exists$ negligible $\boldsymbol{\varepsilon}$ such that:

$\Big|\ \Pr[1 \leftarrow \text{LoR-Exp}_0($  $, \lambda)\ ]$

$\quad - \Pr[1 \leftarrow \text{LoR-Exp}_1($  $, \lambda)\ ]\ \Big| \leq \varepsilon(\lambda)$

Alternate Notion: CPA Security

What if adversary can additionally learn encryptions of messages of her choice?

Examples:

- Midway Island, WWII:
  - US cryptographers discover Japan is planning attack on a location referred to as “AF”
  - Guess that “AF” meant Midway Island
  - To confirm suspicion, sent message in clear that Midway Island was low on supplies
  - Japan intercepted, and sent message referencing “AF”

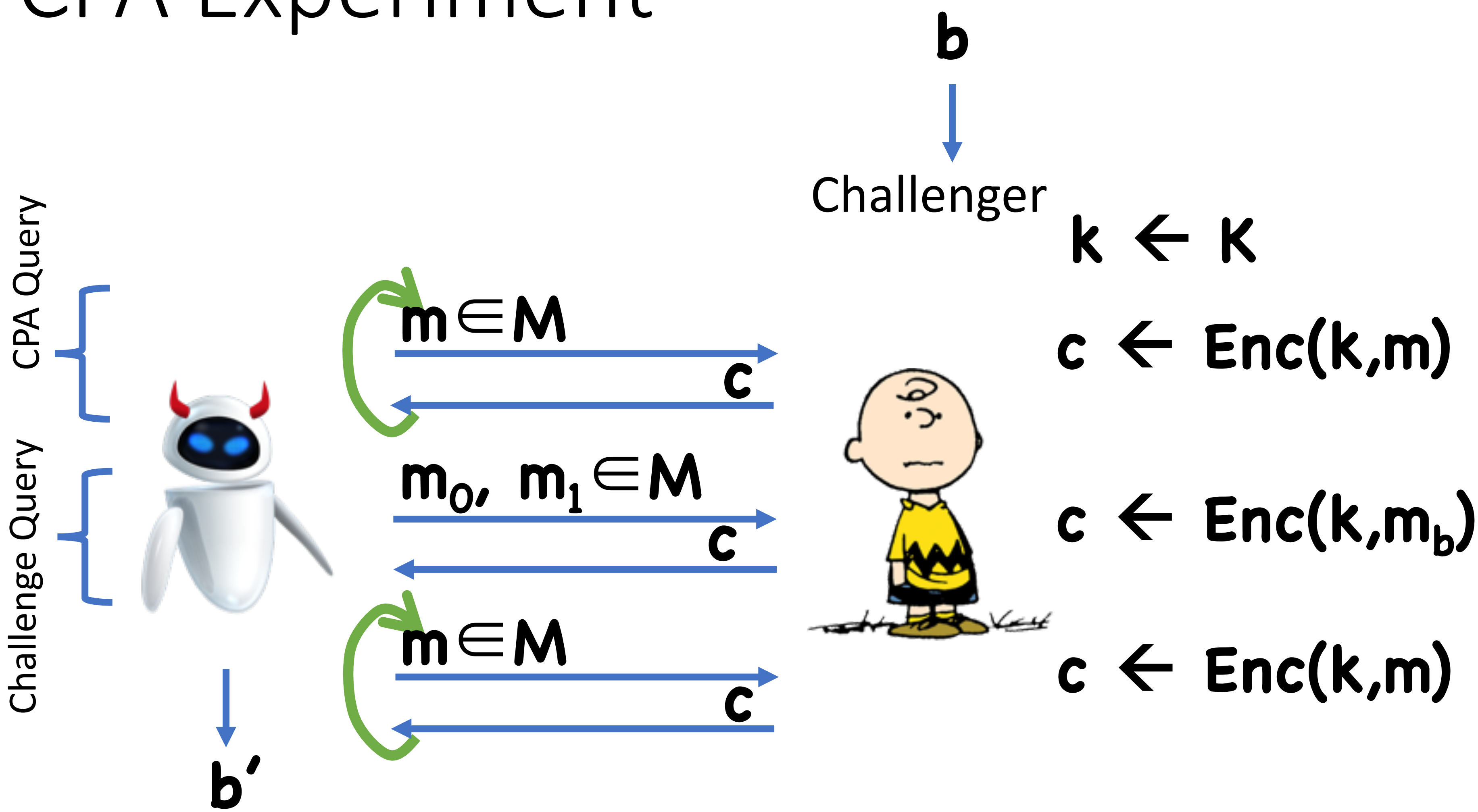
Alternate Notion: CPA Security

What if adversary can additionally learn encryptions of messages of her choice?

Examples:

- Mines, WWII:
  - Allies would lay mines at specific locations
  - Wait for Germans to discover mine
  - Germans would broadcast warning message about the mines, encrypted with Enigma
  - Would also send an “all clear” message once cleared

CPA Experiment



$\text{CPA-Exp}_b(\;)$

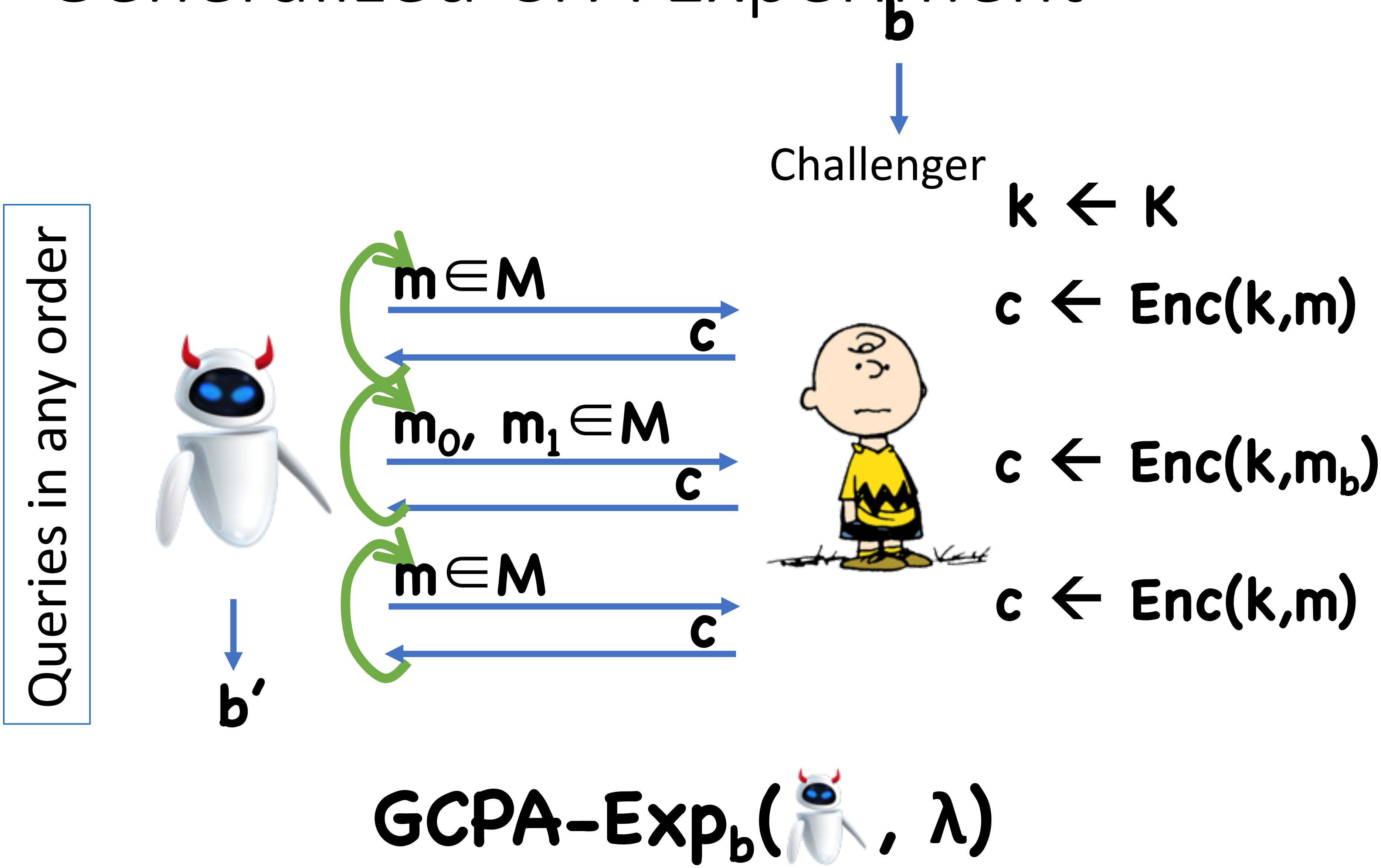
CPA Security Definition

Definition: (Enc, Dec) is **CPA Secure** if, for all  running in polynomial time, $\exists$ negligible **ε** such that:

| Pr[1←CPA-Exp$_0$(, λ)]

– Pr[1←CPA-Exp$_1$(, λ)] | ≤ ε(λ)

Generalized CPA Experiment



GCPA Security Definition

Definition: (Enc, Dec) is **Generalized CPA Secure** if, for all unning in polynomial time, $\exists$ negligible **ε** such that:

| $\Pr[1 \leftarrow \text{GCPA-Exp}_0($  $, \lambda)]$

$- \Pr[1 \leftarrow \text{GCPA-Exp}_1($  $, \lambda)]$ | $\leq \varepsilon(\lambda)$

Equivalences

Theorem:

Left-or-Right indistinguishability

$\Updownarrow$

CPA-security

$\Updownarrow$

Generalized CPA-security

Proof

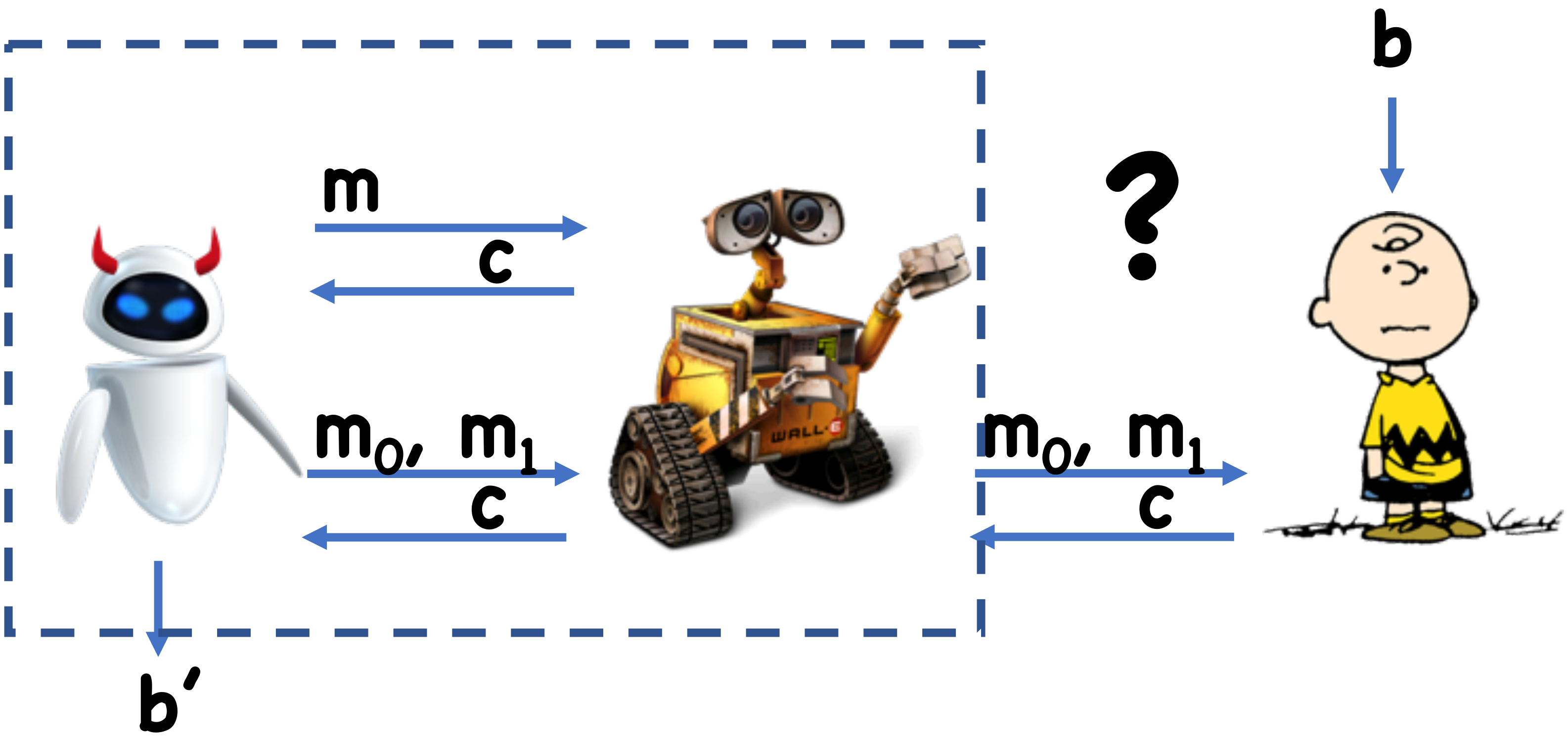
Generalized CPA-security → CPA-security

- Trivial: any adversary in the CPA experiment is also an adversary for the generalized CPA experiment that just doesn't take advantage of the ability to make multiple challenge/LoR queries

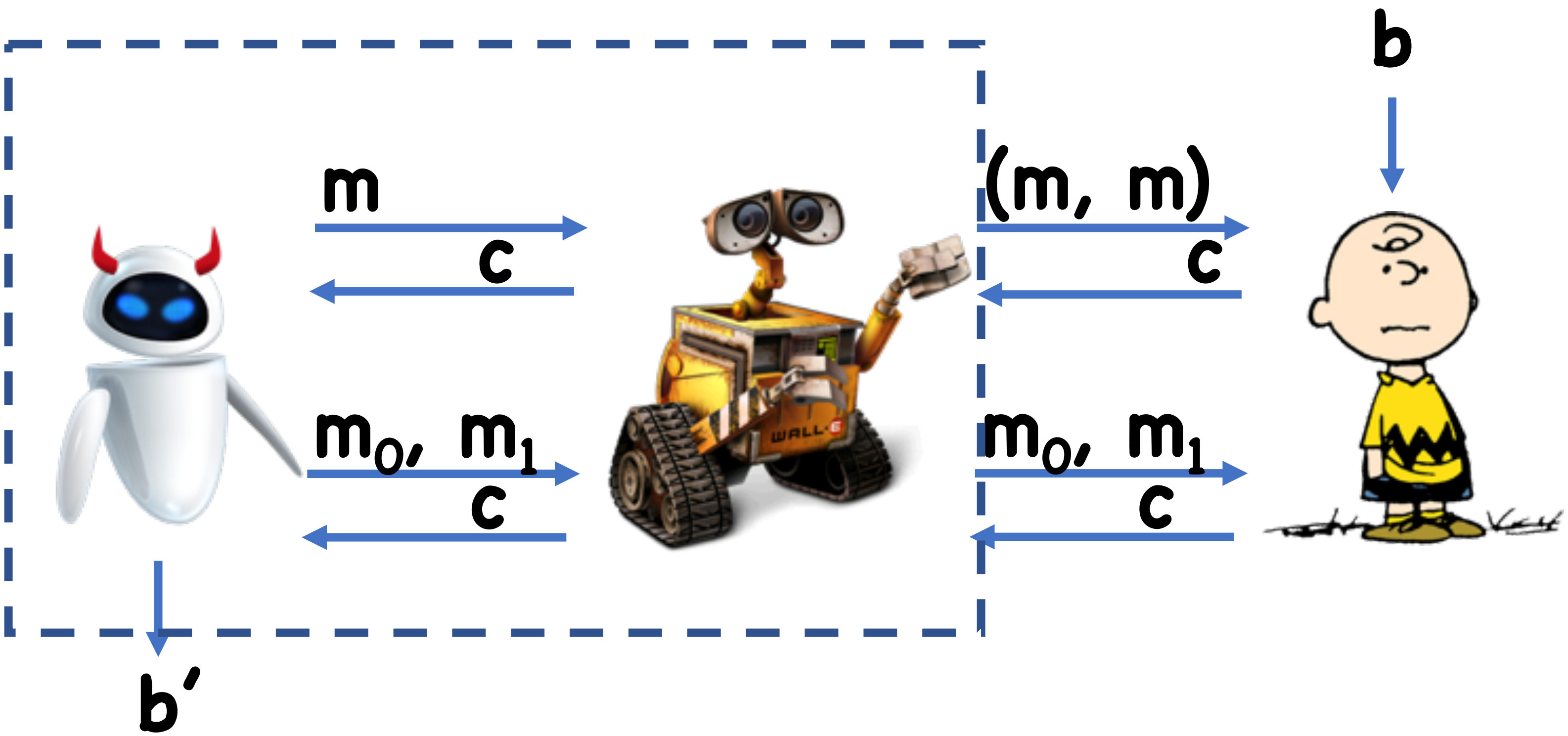
Proof

Left-or-Right → Generalized CPA

- Assume towards contradiction that we have an adversary  for the generalized CPA experiment
- Construct an adversary  that runs  as a subroutine, and breaks the Left-or-Right indistinguishability



$\Pr[1 \leftarrow \text{LoR-Exp}_b($$, \lambda)] = \Pr[1 \leftarrow \text{GCPA-Exp}_b($$, \lambda)]$



Pr[1←LoR-Exp$_b$(, λ)] = Pr[1←GCPA-Exp$_b$(, λ)]

Proof

Left-or-Right → Generalized CPA

| Pr[1←LoR-Exp$_0$(, λ)]

– Pr[1←LoR-Exp$_1$(, λ)] |

= | Pr[1←GCPA-Exp$_0$(, λ)]

– Pr[1←GCPA-Exp$_1$(, λ)] | = ε

Proof

(regular) CPA → Left-or-Right

- Assume towards contradiction that we have an adversary  for the **LoR Indistinguishability**

- Hybrids!

Hybrid **i**:

m_0, m_1

$k \leftarrow K$

If at most **i** queries so far,

$c \leftarrow Enc(k, m_0)$

If more than **i** queries so far,

$c \leftarrow Enc(k, m_1)$

c

b'

Proof

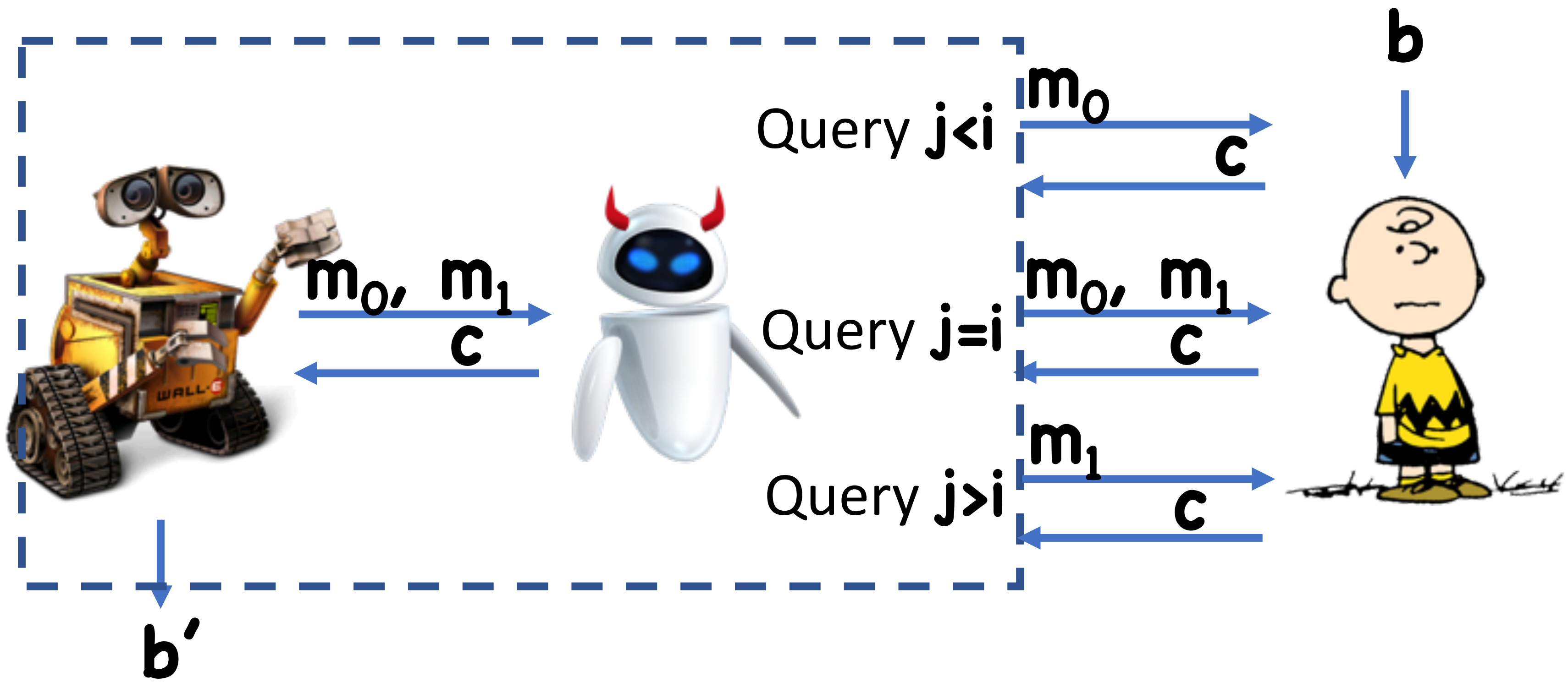
(regular) CPA → Left-or-Right

- Hybrid **0** is identical to **LoR-Exp$_1$(****, λ)**

- Hybrid **q** is identical to **LoR-Exp$_0$(****, λ)**

- We know that  distinguishes Hybrid **q** and Hybrid **0** with advantage **ε**

  $\Rightarrow$ $\exists$ **i** s.t. distinguishes Hybrid **i** and Hybrid **i–1** with advantage **ε/q**



Pr[1←CPA-Exp$_b$(**, λ)] = Pr[1←** in Hybrid **i-b]**

Proof

(regular) CPA → Left-or-Right

| Pr[1←CPA-Exp$_0$( , λ)]

− Pr[1←CPA-Exp$_1$( , λ)] |

= | Pr[1←  in Hybrid **i**]

− Pr[1←  in Hybrid **i−1**] | $\geq \varepsilon/q$

Equivalences

Theorem:

Left-or-Right indistinguishability

$\Updownarrow$

CPA-security

$\Updownarrow$

Generalized CPA-security

Therefore, you can use whichever notion you like best

Next time: how to construct

Announcements/Reminders

Last day to submit HW1

HW2 will be posted today

- Due September 29

PR1 Due October 6