

COS433/Math 473: Cryptography

Mark Zhandry

Princeton University

Spring 2017

Announcements

Homework 2 posted, due Feb 21

Last Time

Multiple message security

Statistical Secrecy

Unfortunately, for everything we've looked at so far,
 $|key| \geq |total\ information\ encrypted|$

Today

$|\text{key}| \geq |\text{total information encrypted}|$ is necessary
for statistical security

Computational Security

Stream Ciphers and PRGs

Notation

For a probabilistic algorithm **A**, we write **A(x; r)** to denote running **A** on input **x**, using randomness **r**

- When thought of as a function of its input and randomness, **A** is deterministic

Statistical Secrecy

Definition: A scheme **(Enc, Dec)** has **statistical secrecy for n messages** if \exists negligible function ϵ s.t. \forall two sequences of messages $(m_0^{(i)})_{i \in [n]}$, $(m_1^{(i)})_{i \in [n]} \in M^n$

$$\Delta \left[(Enc(K_\lambda, m_0^{(i)}))_{i \in [n]}, (Enc(K_\lambda, m_1^{(i)}))_{i \in [n]} \right] < \epsilon(\lambda)$$

Theorem: Suppose **(Enc,Dec)** has plaintext space $\mathbf{M}_\lambda = \{0,1\}^{n(\lambda)}$ and key space $\mathbf{K}_\lambda = \{0,1\}^{t(\lambda)}$. Moreover, assume it is statistically secure for \mathbf{d} messages. Then:

$$t(\lambda) \geq \mathbf{d} \ n(\lambda)$$

In other words, the key must be at least as long as the total length of all messages encrypted

Proof Idea

Use an encryption protocol to build a compression protocol



m



$m' \leftarrow \text{Comp}(m)$

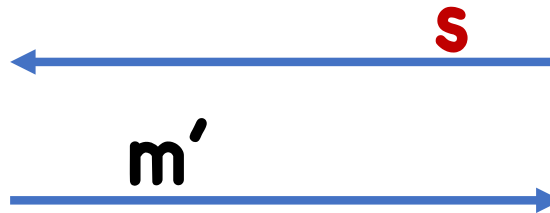
$m \leftarrow \text{Decomp}(m')$

Goal: $|m'| < |m|$

For Now: Easier Goal



m



$s \leftarrow \text{Setup}()$



$m' \leftarrow \text{Comp}(s, m)$

$m \leftarrow \text{Decomp}(s, m')$

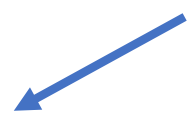
Goal: $|m'| < |m|$

The Protocol

Let \mathbf{m}_0 be some message in \mathbf{M}_λ

Setup():

- Choose random $\mathbf{k}_0 \leftarrow \mathbf{K}$
- Let $\mathbf{c}_1 \leftarrow \text{Enc}(\mathbf{k}_0, \mathbf{m}_0), \dots, \mathbf{c}_d \leftarrow \text{Enc}(\mathbf{k}_0, \mathbf{m}_0)$
- Output $(\mathbf{c}_1, \dots, \mathbf{c}_d)$


 In \mathbf{M}_λ^d

Comp($(\mathbf{c}_1, \dots, \mathbf{c}_d), (\mathbf{m}_1, \dots, \mathbf{m}_d)$):

- Find $\mathbf{k}, \mathbf{r}_1, \dots, \mathbf{r}_d$ such that $\mathbf{c}_i = \text{Enc}(\mathbf{k}, \mathbf{m}_i; \mathbf{r}_i) \quad \forall i$
- If no such values exist, abort
- Output \mathbf{k}

The Protocol

Let \mathbf{m}_0 be some message in \mathbf{M}_λ

Comp($(\mathbf{c}_1, \dots, \mathbf{c}_d)$, $(\mathbf{m}_1, \dots, \mathbf{m}_d)$):  In \mathbf{M}_λ^d

- Find $\mathbf{k}, \mathbf{r}_1, \dots, \mathbf{r}_d$ such that $\mathbf{c}_i = \text{Enc}(\mathbf{k}, \mathbf{m}_i; \mathbf{r}_i) \quad \forall i$
- If no such values exist, abort
- Output \mathbf{k}

Decomp($(\mathbf{c}_1, \dots, \mathbf{c}_d)$, \mathbf{k}):

- Compute $\mathbf{m}_i = \text{Dec}(\mathbf{k}, \mathbf{c}_i)$
- Output $(\mathbf{m}_1, \dots, \mathbf{m}_d)$

Analysis of Protocol

If **Comp** succeeds, **Decomp** must succeed by correctness

- Since $\mathbf{c}_i = \text{Enc}(\mathbf{k}, \mathbf{m}_i; \mathbf{r}_i)$, $\text{Dec}(\mathbf{k}, \mathbf{c}_i)$ must give \mathbf{m}_i

Therefore, must figure out when **Comp** succeeds

Claim: There exists a negligible function $\epsilon(\lambda)$ such that, for any sequence of messages $\mathbf{m}_1, \dots, \mathbf{m}_d$, **Comp** succeeds with probability at least $1 - \epsilon(\lambda)$

(Probability over the randomness used by **Setup()**)

Claim: There exists a negligible function $\epsilon(\lambda)$ such that, for any sequence of messages $\mathbf{m}_1, \dots, \mathbf{m}_d$, **Comp** succeeds with probability at least $1 - \epsilon(\lambda)$

Proof:

- Suppose **Comp** succeeds with probability $1 - p$ for messages $\mathbf{m}_1, \dots, \mathbf{m}_d$
- Let $\mathbf{A}(\mathbf{c}_1, \dots, \mathbf{c}_d)$ be the algorithm that runs **Comp** $((\mathbf{c}_1, \dots, \mathbf{c}_d), (\mathbf{m}_1, \dots, \mathbf{m}_d))$ and outputs **1** if **Comp** succeeds
- If $\mathbf{c}_i = \mathbf{Enc}(\mathbf{k}_0, \mathbf{m}_i)$, then $\Pr[\mathbf{A}(\mathbf{c}_1, \dots, \mathbf{c}_d) = 1] = 1$
- If $\mathbf{c}_i = \mathbf{Enc}(\mathbf{k}_0, \mathbf{m}_0)$, then $\Pr[\mathbf{A}(\mathbf{c}_1, \dots, \mathbf{c}_d) = 1] = 1 - p$
- By statistical security of **Enc**, p must be negligible

Claim: There exists a negligible function $\epsilon(\lambda)$ such that, for any sequence of messages $\mathbf{m}_1, \dots, \mathbf{m}_d$, **Comp** succeeds with probability at least $1 - \epsilon(\lambda)$

Claim: There exists a negligible function $\epsilon(\lambda)$ such that, for **a random** sequence of messages $\mathbf{m}_1, \dots, \mathbf{m}_d$, **Comp** succeeds with probability at least $1 - \epsilon(\lambda)$

(Probability over the randomness used by **Setup()**
and the random choices of $\mathbf{m}_1, \dots, \mathbf{m}_d$)

Next step: Removing Setup

We know:

$$\Pr[\text{Comp succeeds: } (c_1, \dots, c_d) \leftarrow \text{Setup()}, m_i \leftarrow M_\lambda] \geq 1 - \varepsilon(\lambda)$$

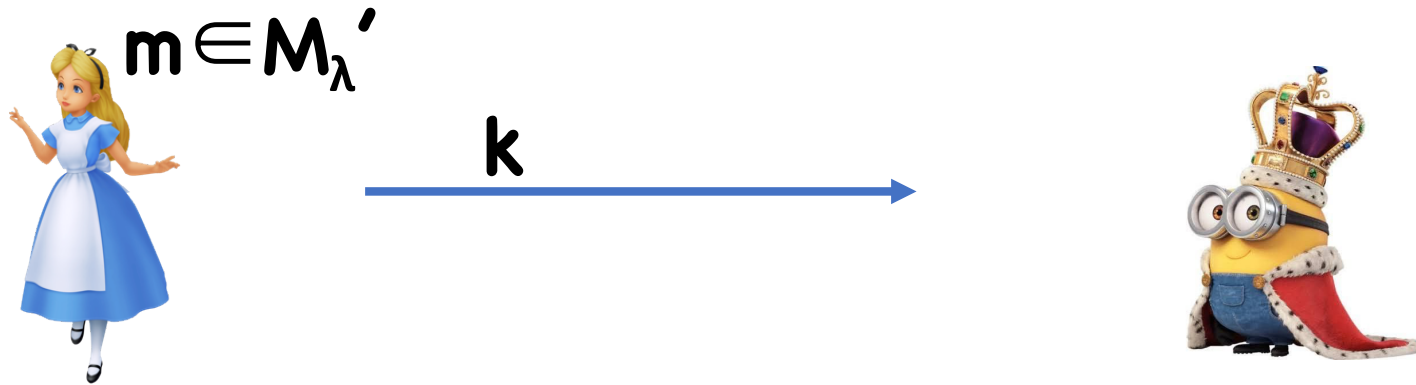
Therefore, there must exist *some* (c_1^*, \dots, c_d^*) such that

$$\Pr[\text{Comp succeeds: } m_i \leftarrow M_\lambda] \geq 1 - \varepsilon(\lambda)$$

Define: $M_\lambda' = \{(m_1, \dots, m_d): \text{Comp succeeds}\}$

- Note that $|M_\lambda'| \geq (1 - \varepsilon(\lambda)) |M_\lambda|^d$

The Protocol



Find k, r_1, \dots, r_d such that
 $c_i^* = \text{Enc}(k, m_i; r_i) \quad \forall i$

For each i ,
Let $m_i \leftarrow \text{Dec}(k, c_i^*)$
Output (m_1, \dots, m_d)

By previous analysis,

- Alice always successfully compresses
- Bob always successfully decompresses

Final Touches

Can compress messages in \mathbf{M}_λ' into keys in \mathbf{K}_λ

Therefore, it must be that $|\mathbf{M}_\lambda'| \leq |\mathbf{K}_\lambda|$

Meaning $t = \log |\mathbf{K}_\lambda|$

$$\begin{aligned} &\geq \log |\mathbf{M}_\lambda'| \\ &\geq \log [(1-\varepsilon(\lambda)) |\mathbf{M}_\lambda|^d] \\ &= d \log |\mathbf{M}_\lambda| + \log [1-\varepsilon(\lambda)] \\ &\geq dn - 2\varepsilon(\lambda) \\ &\geq dn \text{ (as long as } \varepsilon(\lambda) < \tfrac{1}{2}) \end{aligned}$$

Takeaway

If you don't want to physically exchange keys frequently, you cannot obtain statistical security

So, now what?

Computational Security

If it takes a billion years to decrypt the message,
that's ok

How long is okay?

- Practitioner: 2^{80} , 2^{128} , or maybe 2^{256} computational steps
- Theorist? superpolynomial

Defining Security

Consider an attacker as a probabilistic polynomial time algorithm (Turing machine)

- Cobham's thesis: captures anything computable in the real world

Attacker gets to choose the messages

All attacker has to do is distinguish them

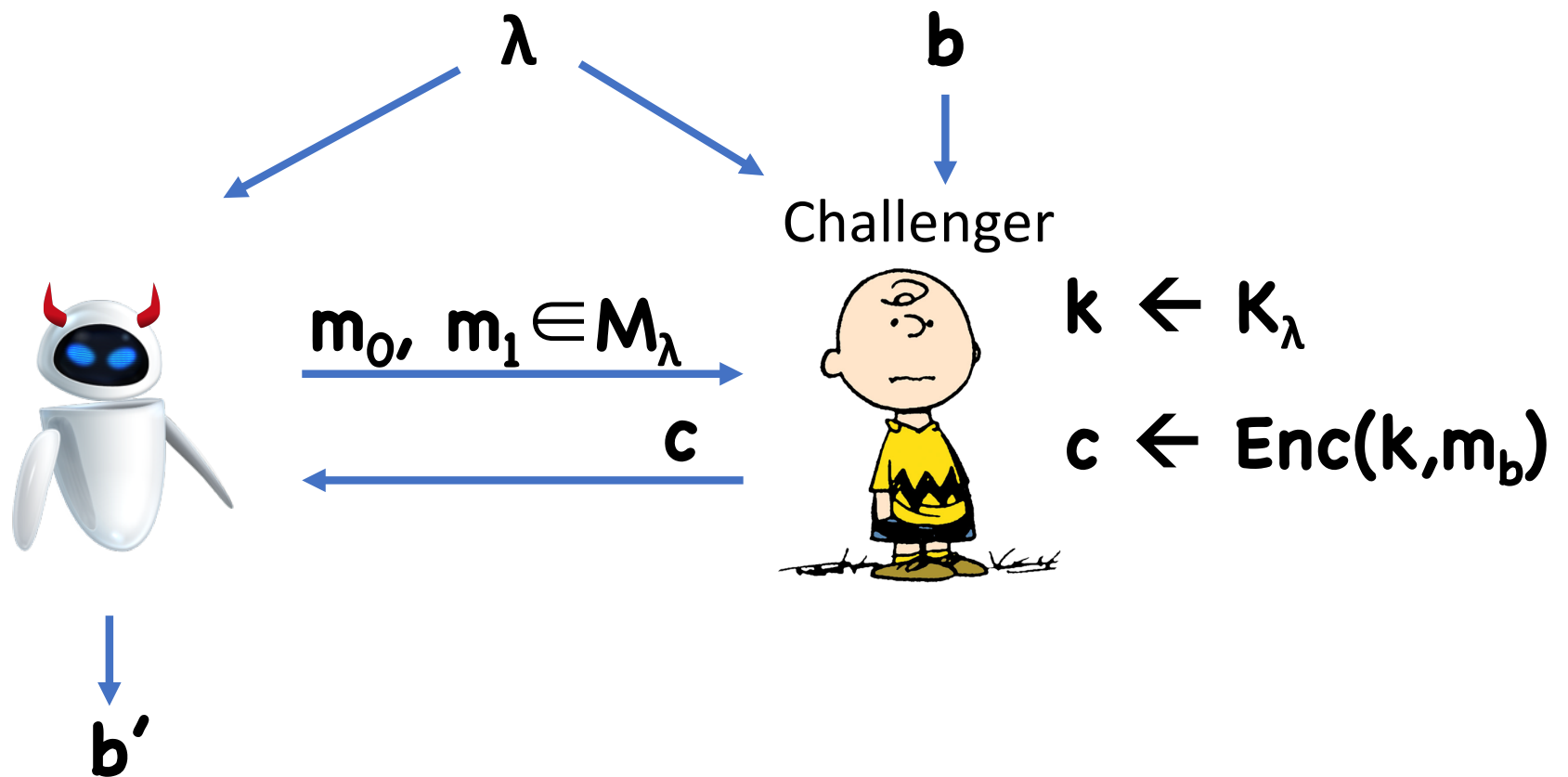
Our impossibility gives an exponential-time algorithm → attack doesn't apply in this setting

Defining Security

Efficiency of algorithms


- Doesn't make sense to allow **(Enc, Dec)** to run in superpoly time, but not the adversary
- Therefore, all algorithms are also poly time (efficient)

Security Experiment/Game



$\text{IND-Exp}_b(\text{robot}, \lambda)$

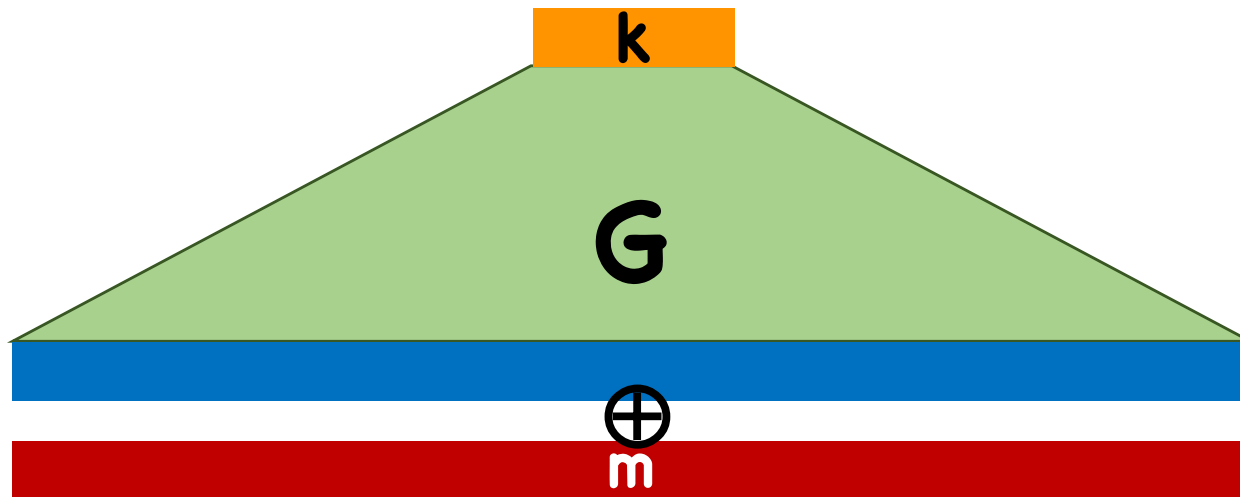
Security Definition

Definition: (Enc, Dec) has **ciphertext indistinguishability** if, for all probabilistic polynomial time (PPT) , there exists a negligible function ϵ such that

$$\left| \Pr[1 \leftarrow \text{IND-Exp}_0(\text{robot}, \lambda)] - \Pr[1 \leftarrow \text{IND-Exp}_1(\text{robot}, \lambda)] \right| \leq \epsilon(\lambda)$$


Construction with $|k| \ll |m|$


Idea: use OTP, but have key generated by some expanding function G




What Do We Want Out of **G**?

Definition: $G:\{0,1\}^* \rightarrow \{0,1\}^*$ is a **secure pseudorandom generator (PRG)** if:

- **G** is computable in polynomial time
- **G** applied to λ bit strings produces strings of length $s(\lambda) > \lambda$
- **G** is deterministic
- For all **PPT** , \exists **negl** ϵ such that:

$$\left| \Pr[\text{ (G(s))=1 : s \leftarrow \{0,1\}^\lambda] \right.$$

$$\left. - \Pr[\text{ (x)=1 : x \leftarrow \{0,1\}^{s(\lambda)}] \right| \leq \epsilon(\lambda)$$

Secure PRG \rightarrow Ciphertext Indistinguishability

$$K_\lambda = \{0,1\}^\lambda$$

$$M_\lambda = \{0,1\}^{s(\lambda)}$$

$$C_\lambda = \{0,1\}^{s(\lambda)}$$

$$\text{Enc}(k,m) = \text{PRG}(k) \oplus m$$

$$\text{Dec}(k,c) = \text{PRG}(k) \oplus c$$

Security?

Intuitively, security is obvious:

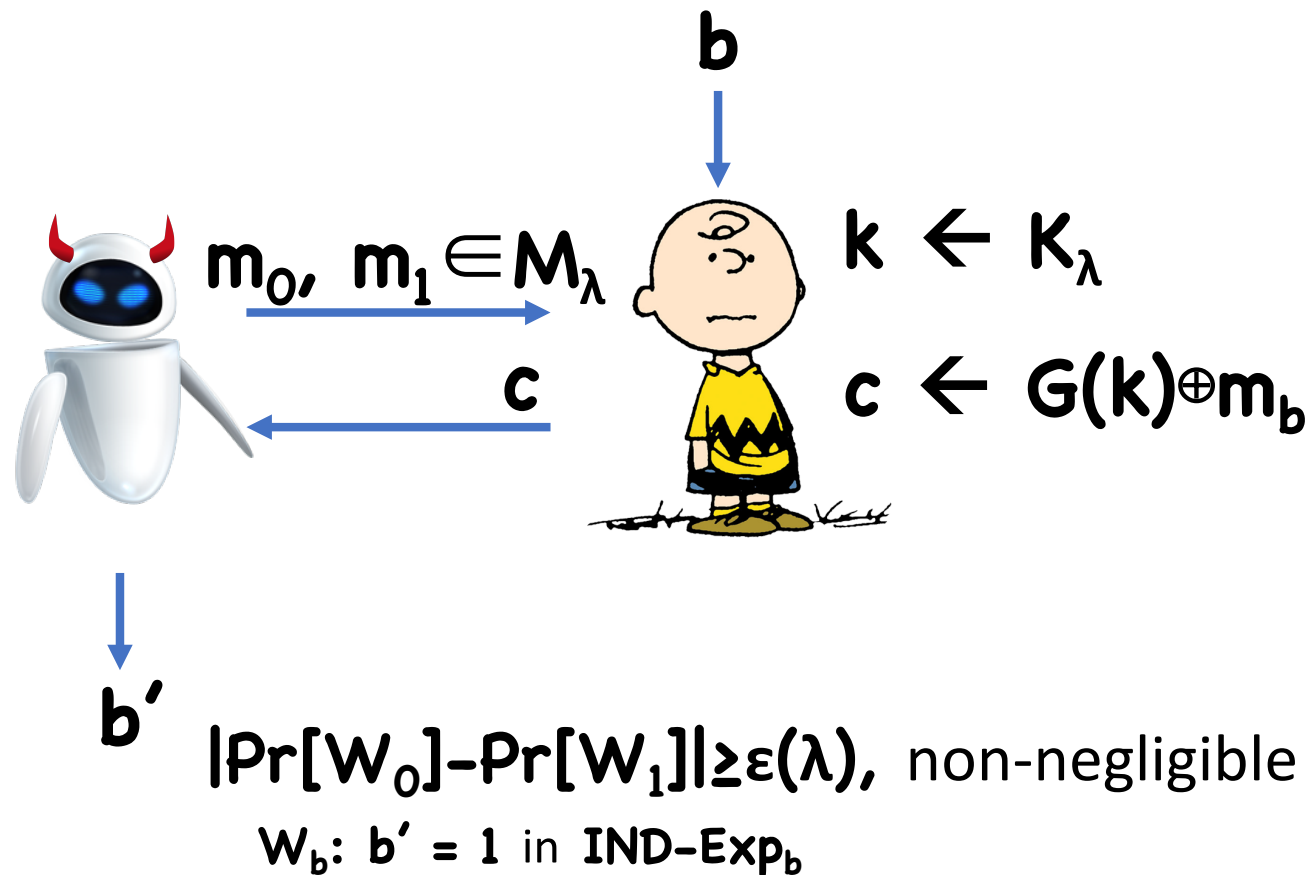
- $\text{PRG}(k)$ "looks" random, so should completely hide \mathbf{m}
- However, formalizing this argument is non-trivial.

Solution: reductions

- Assume toward contradiction an adversary for the encryption scheme, derive an adversary for the PRG

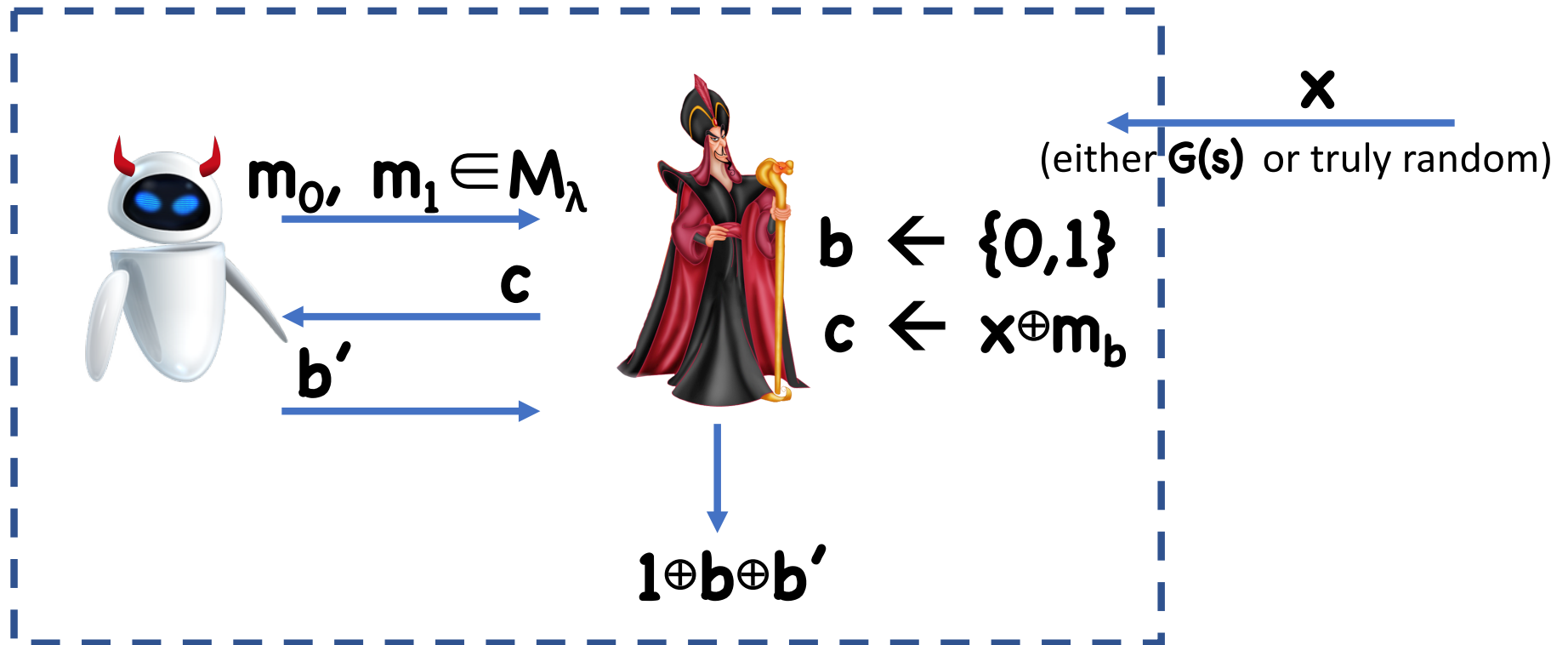
Security

Assume towards contradiction that there is a PPT  such that



Security

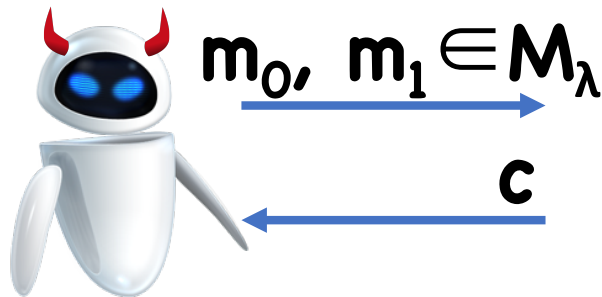
Use  to build .  will run  as a subroutine, and pretend to be .



Security

Case 1: $\mathbf{x} = \text{PRG}(\mathbf{s})$ for a random seed \mathbf{s}

-  “sees” IND-Exp_b for a random bit \mathbf{b}



$$\mathbf{b} \leftarrow \{0,1\}$$

$$\mathbf{s} \leftarrow K_\lambda$$

$$\mathbf{c} \leftarrow \text{PRG}(\mathbf{s}) \oplus m_b$$

\downarrow
 \mathbf{b}'

Security

Case 1: $\mathbf{x} = \text{PRG}(\mathbf{s})$ for a random seed \mathbf{s}

-  “sees” IND-Exp_b for a random bit b

- $\Pr[1 \oplus b \oplus b' = 1] = \Pr[b = b']$

$$= \frac{1}{2} \Pr[b' = 1 \mid b = 1]$$

$$+ \frac{1}{2} (1 - \Pr[b' = 1 \mid b = 0])$$

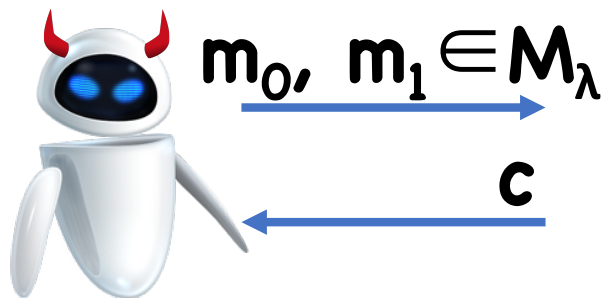
$$= \frac{1}{2} (1 + \Pr[W_0] - \Pr[W_1])$$

$$= \frac{1}{2} (1 \pm \epsilon(\lambda))$$

Security

Case 2: \mathbf{x} is truly random

-  “sees” OTP encryption




$$\begin{aligned} b &\leftarrow \{0,1\} \\ x &\leftarrow \{0,1\}^{s(\lambda)} \\ c &\leftarrow x \oplus m_b \end{aligned}$$

\downarrow
 b'

Security

Case 2: \mathbf{x} is truly random

-  “sees” OTP encryption
- Therefore $\Pr[b'=1 \mid b=0] = \Pr[b'=1 \mid b=1]$
- $\Pr[1 \oplus b \oplus b' = 1] = \Pr[b = b']$
$$= \frac{1}{2} \Pr[b'=1 \mid b=1]$$
$$+ \frac{1}{2} (1 - \Pr[b'=1 \mid b=0])$$
$$= \frac{1}{2}$$

Security

Putting it together:

- $\Pr[\text{👑}(G(s))=1:s \leftarrow \{0,1\}^\lambda] = \frac{1}{2}(1 \pm \epsilon(\lambda))$
- $\Pr[\text{👑}(x)=1:x \leftarrow \{0,1\}^{s(\lambda)}] = \frac{1}{2}$
- Absolute Difference: $\frac{1}{2}\epsilon(\lambda)$, non-negligible
 \Rightarrow Contradiction!

An Alternate Proof: Hybrids

Idea: define sequence of “hybrid” experiments
“between” **IND-Exp₀** and **IND-Exp₁**

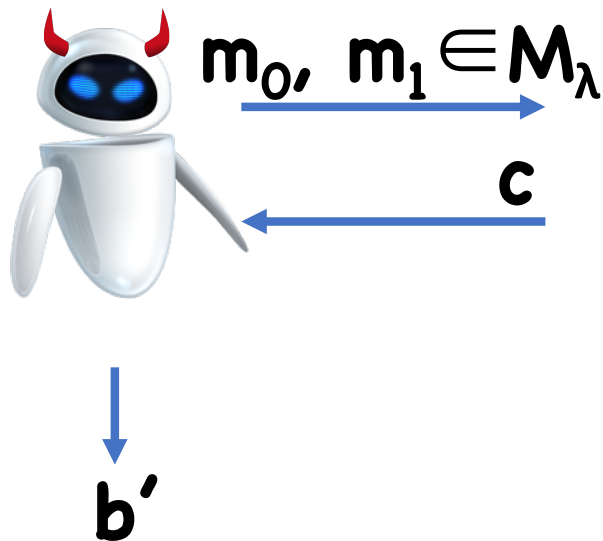
In each hybrid, make small change from previous
hybrid

Hopefully, each small change is undetectable

Using triangle inequality, overall change from **IND-Exp₀** and **IND-Exp₁** is undetectable

An Alternate Proof: Hybrids

Hybrid 0: IND-Exp₀

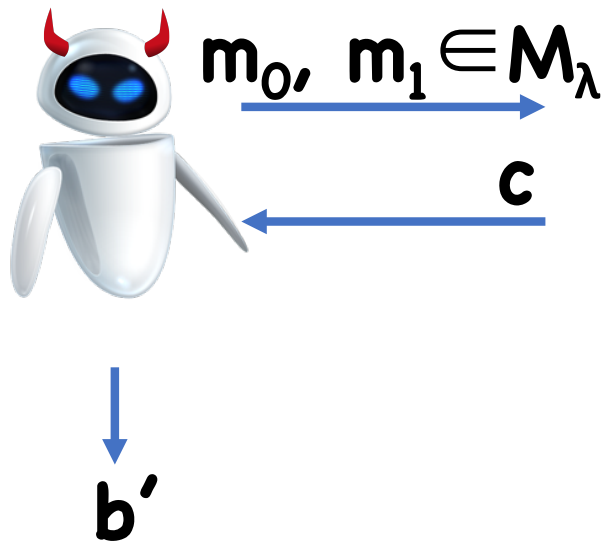


$$k \leftarrow K_\lambda$$

$$c \leftarrow G(k) \oplus m_0$$

An Alternate Proof: Hybrids

Hybrid 1:

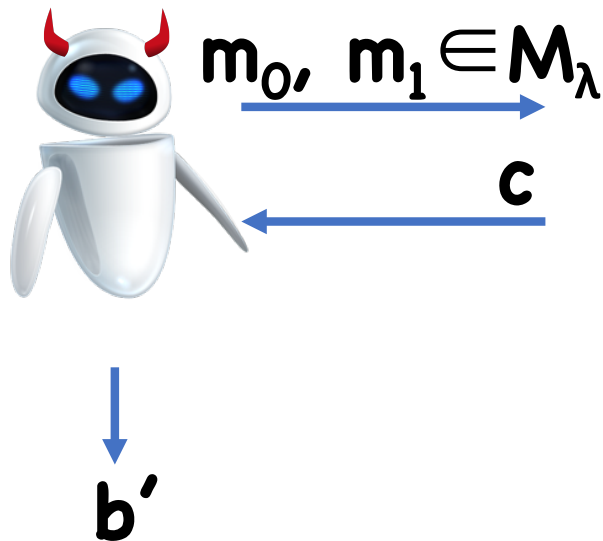


$$x \leftarrow \{0,1\}^{s(\lambda)}$$

$$c \leftarrow x \oplus m_0$$

An Alternate Proof: Hybrids

Hybrid 2:

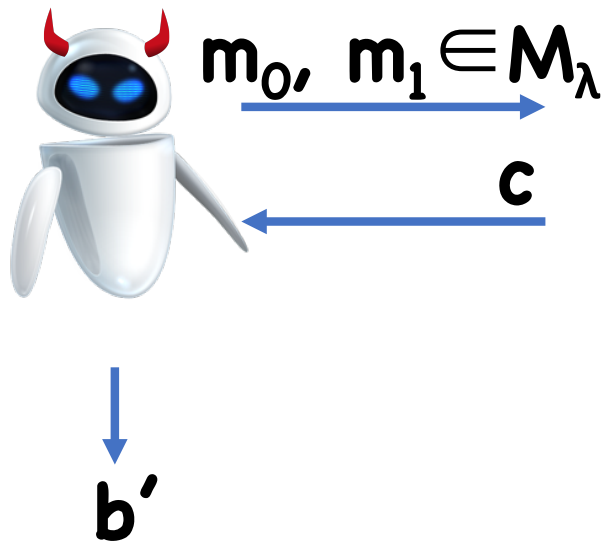


$$x \leftarrow \{0,1\}^{s(\lambda)}$$

$$c \leftarrow x \oplus m_1$$

An Alternate Proof: Hybrids

Hybrid 3: IND-Exp₁



$$k \leftarrow K_\lambda$$

$$c \leftarrow G(k) \oplus m_1$$

An Alternate Proof: Hybrids

$$\begin{aligned} & | \Pr[b'=1 : \text{IND-Exp}_0] - \Pr[b'=1 : \text{IND-Exp}_1] | \\ &= | \Pr[b'=1 : \text{Hyb } 0] - \Pr[b'=1 : \text{Hyb } 3] | \\ &\leq | \Pr[b'=1 : \text{Hyb } 0] - \Pr[b'=1 : \text{Hyb } 1] | \\ &\quad + | \Pr[b'=1 : \text{Hyb } 1] - \Pr[b'=1 : \text{Hyb } 2] | \\ &\quad + | \Pr[b'=1 : \text{Hyb } 2] - \Pr[b'=1 : \text{Hyb } 3] | \end{aligned}$$

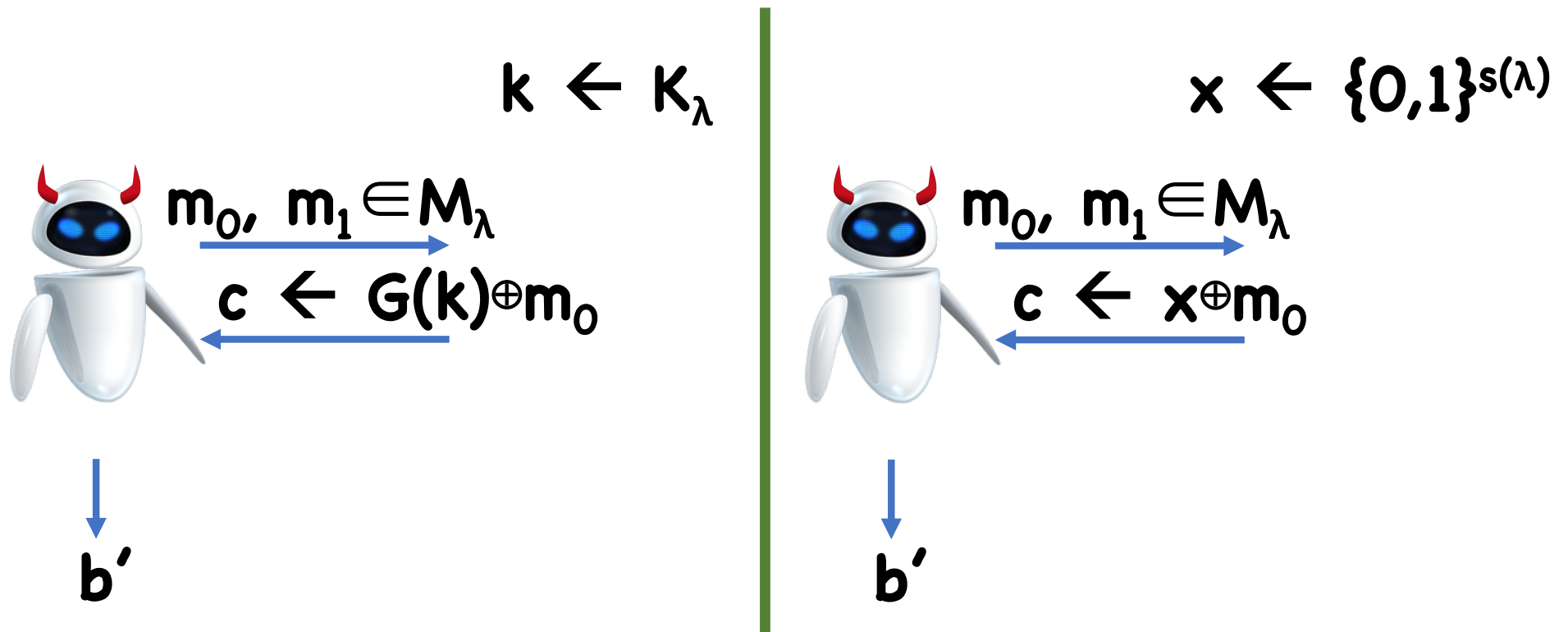
If $|\Pr[b'=1 : \text{IND-Exp}_0] - \Pr[b'=1 : \text{IND-Exp}_1]| \geq \varepsilon(\lambda)$,

Then for some $i=0,1,2$,



$$|\Pr[b'=1 : \text{Hyb } i] - \Pr[b'=1 : \text{Hyb } i+1]| \geq \varepsilon(\lambda)/3$$

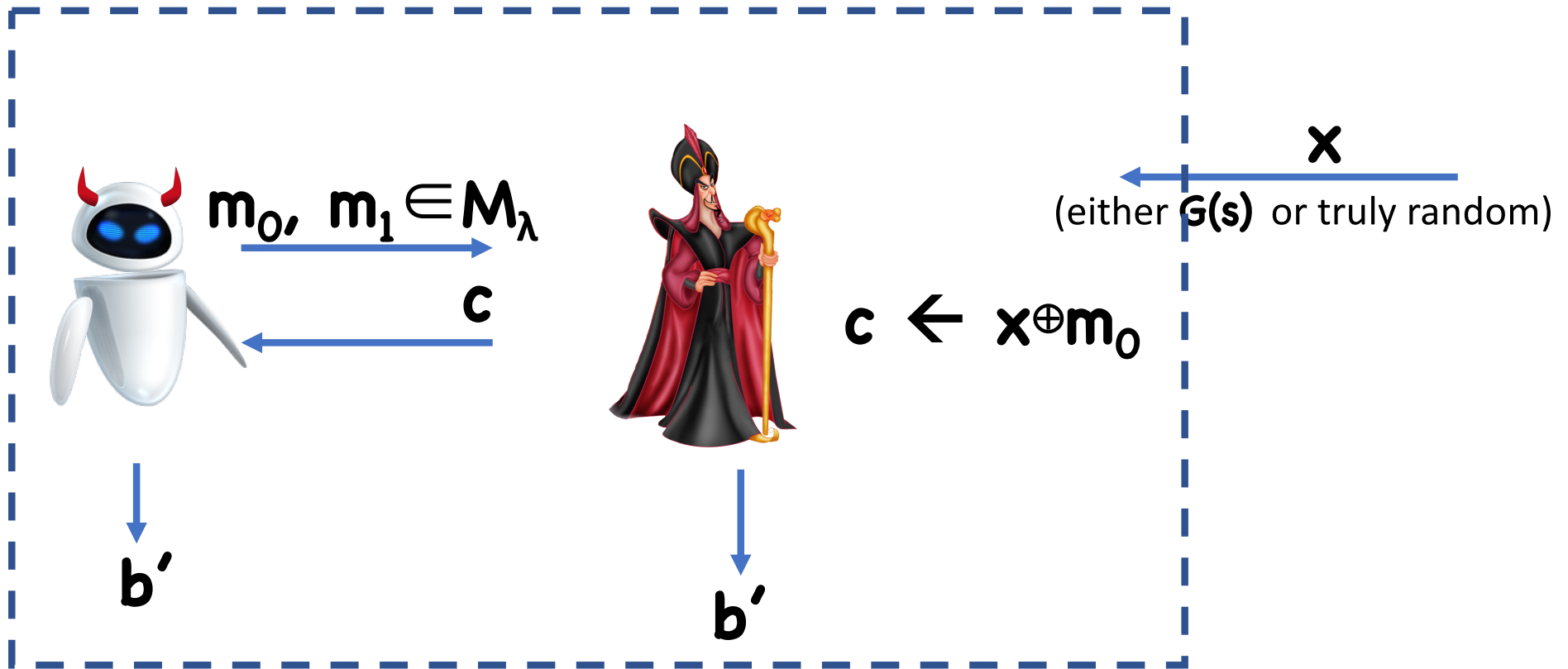
An Alternate Proof: Hybrids

Suppose  distinguishes **Hybrid 0** from **Hybrid 1** with advantage $\epsilon(\lambda)/3$





An Alternate Proof: Hybrids

Suppose  distinguishes **Hybrid 0** from **Hybrid 1** with advantage $\epsilon(\lambda)/3 \Rightarrow$ Construct 



An Alternate Proof: Hybrids

Suppose  distinguishes **Hybrid 0** from **Hybrid 1** with advantage $\epsilon(\lambda)/3 \Rightarrow$ Construct 

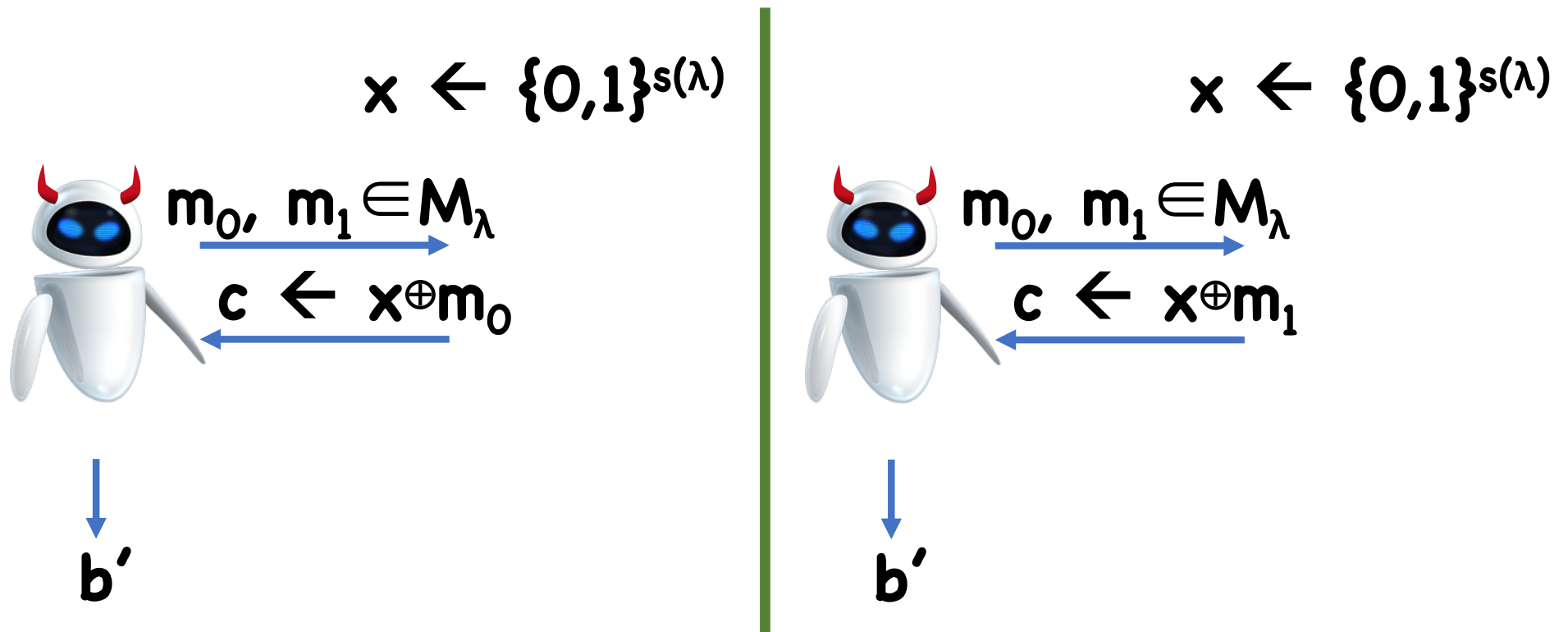
If  is given $G(s)$ for a random s ,  sees **Hybrid 0**

If  is given x for a random x ,  sees **Hybrid 1**

Therefore, advantage of  is equal to advantage of  which is at least $\epsilon(\lambda)/3 \Rightarrow$ Contradiction!

An Alternate Proof: Hybrids

Suppose  distinguishes **Hybrid 1** from **Hybrid 2** with advantage $\epsilon(\lambda)/3$



An Alternate Proof: Hybrids

Suppose  distinguishes **Hybrid 1** from **Hybrid 2**
with advantage $\epsilon/3$

$b \leftarrow \{0,1\}$

$x \leftarrow \{0,1\}^{s(\lambda)}$

Impossible by OTP security

m_0

$c \leftarrow$

m_0

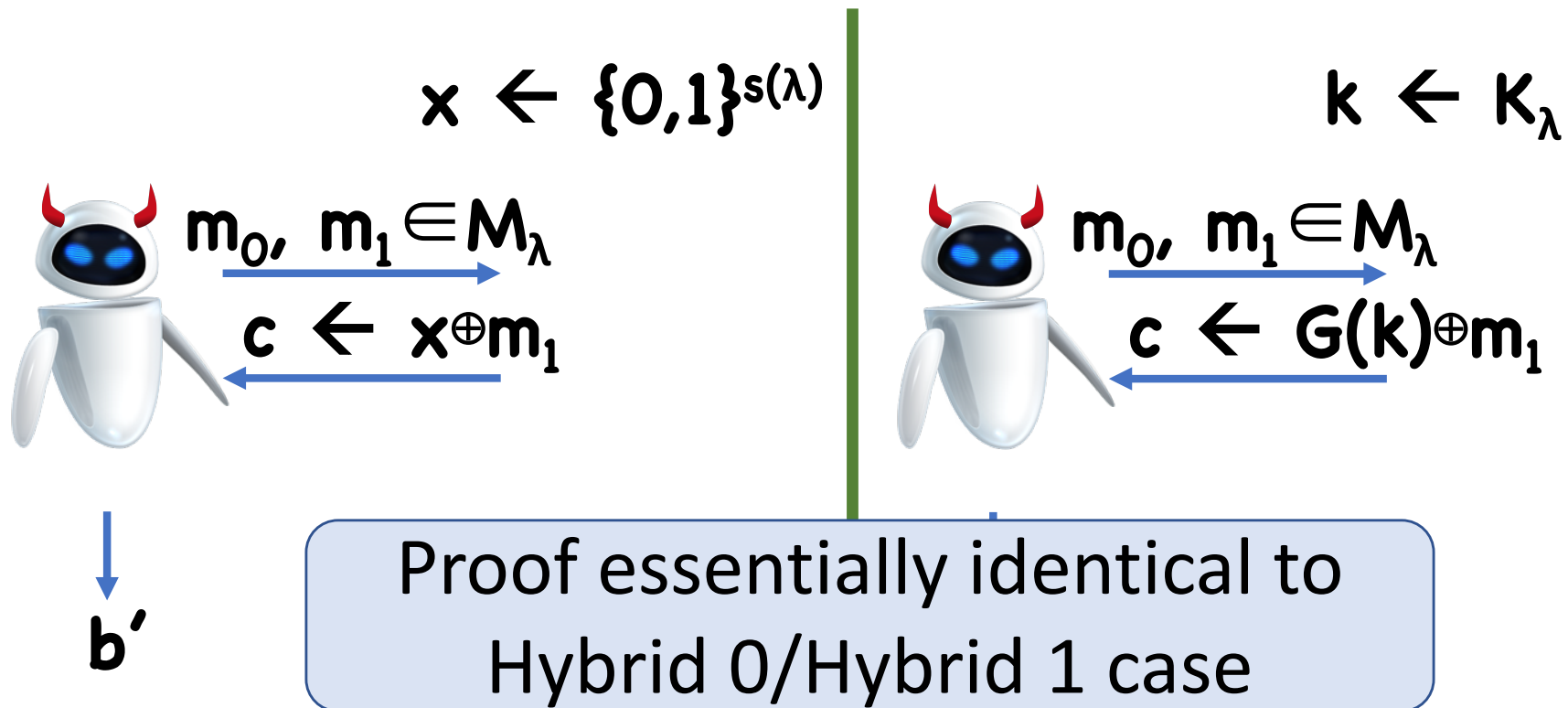
$x \oplus m_1$

b'

b'

An Alternate Proof: Hybrids

Suppose  distinguishes **Hybrid 2** from **Hybrid 3** with advantage $\epsilon(\lambda)/3$





PRG Discussion

Do we need to restrict to PPT  ?

YES!

Reason:

- Impossibility for statistically secure encryption gives exponential-time adversary . Apply reduction to get exponential-time 

PRG Discussion

Do we need to restrict to PPT ?



YES!

Reason:

- More obvious: Brute force attack

For each \mathbf{s} in $\{0,1\}^\lambda$:

If $G(\mathbf{s}) = \mathbf{x}$, output 1

If no \mathbf{s} found, output 0



x
←
(either $G(\mathbf{s})$ or truly random)

PRG Discussion

For each \mathbf{s} in $\{0,1\}^\lambda$:

If $G(\mathbf{s}) = \mathbf{x}$, output 1

If no \mathbf{s} found, output 0



\mathbf{x}
←
(either $G(\mathbf{s})$ or truly random)

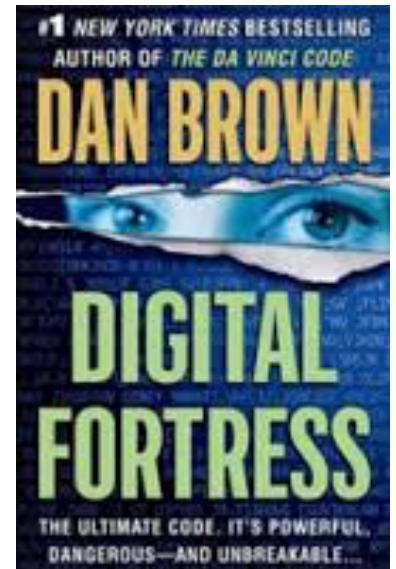
If $\mathbf{x} = G(\mathbf{s})$ for random \mathbf{s} , will always output **1**

If \mathbf{x} is truly random, will output **1** with probability at most $2^{\lambda-s(\lambda)} \leq \frac{1}{2}$

- $2^{s(\lambda)}$ possible values of \mathbf{x} , only $\leq 2^s$ possible values of $G(\mathbf{s})$

“Bergofsky Principle”

(Not a real principle)



(False) Bergofsky Principle: you can brute force any cryptosystem to learn the key

Can you think of an example that contradicts this?

Brute Forcing OTP?

Say I see the ciphertext

ciphertext: **AKFLRKATEOMH**

I try all keys, and I see that

key: **ARMLPAAABOQU**

message: **attackatdawn**

Is this the right key/message? Who knows...

key: **ARMLPAAABUVX**

message: **attackatdusk**

When Is Brute Force Possible?

Possible:

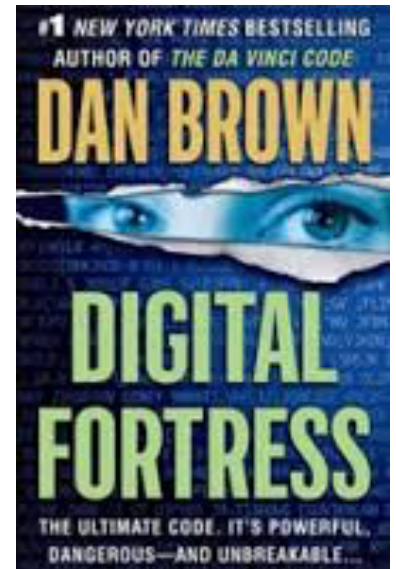
- PRGs
- Shift cipher (unlikely two shifts give valid English)
- Substitution cipher
- Encryption where $|key| < |message|$

Impossible:

- OTP
- Anything else? Anagrams

“Bergofsky Principle”

(Not a real principle)



If you want to find a secret from a finite set, and if given a candidate secret from that set it is possible to tell if the secret is correct, then you can always find the secret given enough time.

Do PRGs Exist?

If **P=NP**, the answer is NO

- Language **L** = $\{x: \exists s, G(s)=x\}$
- **s** is an efficiently verifiable witness that $x \in L$
- Therefore, **L** \in **NP**
- If **P=NP**, can decide **L** in polynomial time
 \Rightarrow break PRG security of **G**

Do PRGs Exist?

Therefore, we need to at least assume **P \neq NP**

Fortunately, most people believe **P \neq NP**

But, huge open question:

Can we build PRGs assuming **P \neq NP**?

Big difficulty:

P \neq NP is a worst case assumption, whereas
PRGs are average case

Does Crypto Exist?

In most cases, crypto requires **$P \neq NP$**

- We can usually efficiently check if a key is correct
- Therefore, if **$P = NP$** , we can also find the key in efficiently
- There are some exceptions: notable example is OTP

However, most crypto seems to require something stronger

Again, **$P \neq NP$** is worst case, whereas most crypto definitions are average case

$P \neq NP$ is necessary but not sufficient for most crypto

Assumptions in Cryptography

For most crypto, will need to make certain computational assumptions

- E.g. that G is a PRG

Obviously, unsatisfying state of affairs

To gain confidence in assumption, need to perform extensive cryptanalysis

Assumptions in Cryptography

To gain confidence in assumption, need to perform extensive cryptanalysis

- Expensive and time consuming

Ideally, we would only do this once

- Don't want to have to perform cryptanalysis every time we design a new scheme

Provable Security

Major goal in cryptography:

Use one component (e.g. PRG) for many cryptographic tasks (e.g. Encryption), with security proof assuming just the security of the component

When we say to “prove” security, we mean relative to the assumption that building block is secure

Exactly what this means should generally be clear from context

Summary

Computational assumptions crucial to cryptography

First building block: PRGs

- Use to build encryption where $|key| \ll |message|$

Security proofs by reduction

- Hybrid arguments

Next Time

Stream Ciphers

Some design principles behind PRGs