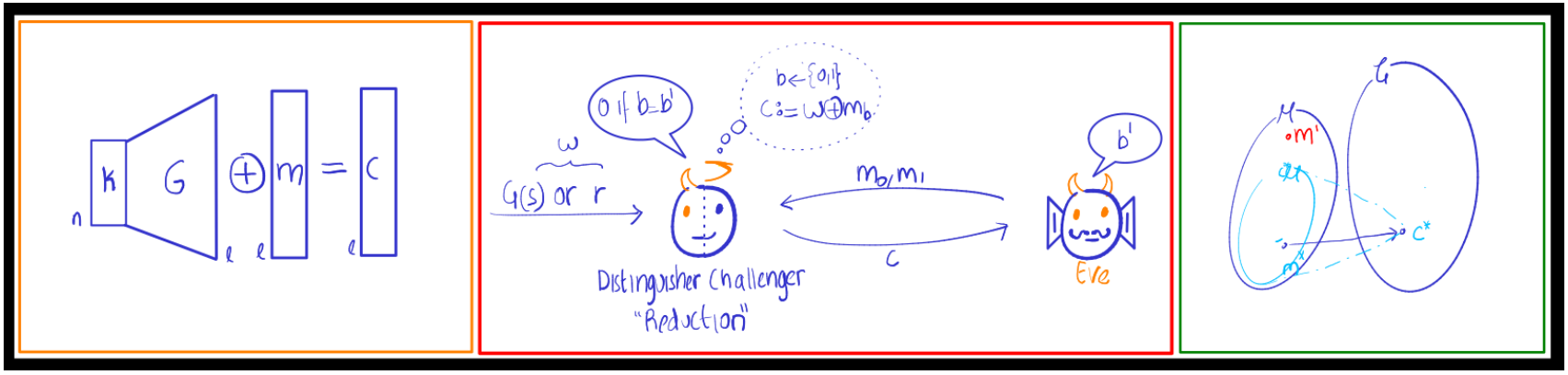


CS783: Theoretical Foundations of Cryptography

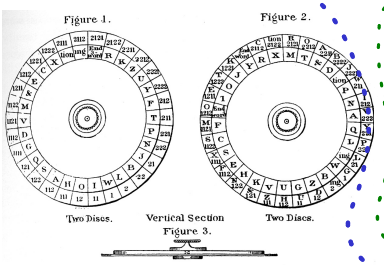
Lecture 3 (30/Jul/24)

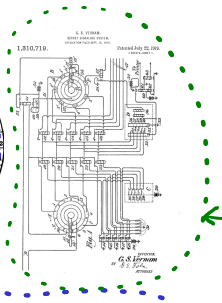
Instructor: Chethan Kamath

Recall from Last Lecture...

MODULE I
(Shared keys)

For a large part of history





Some historical ciphers

Perfect secrecy and one-time pad

Ego quoque sum conscius secreti

ooga

Past

~0

~1930s

Birth of information theory & CS

Foundations of modern cryptography

Credit for images: Wikipedia

Recall from Last Lecture...

General *template*:

1. Identify the task — secret communication with shared keys
2. Come up with precise threat model M (a.k.a security model) — Perfect secrecy against eavesdroppers
   - Adversary/Attack: What are the adversary's capabilities?
   - Security Goal: What does it mean to be secure?
3. Construct a scheme Π — One-time pad
4. Formally prove that Π in secure in model M

Plan for This Lecture...

of longer messages

General *template*:

secret communication with shared keys

1. Identify the task
2. Come up with precise threat model M (a.k.a security model)
   - Adversary/Attack: What are the adversary's capabilities?
   - Security Goal: What does it mean to be secure?

   Perfect secrecy against eavesdroppers
3. Construct a scheme Π

   One-time pad
4. Formally prove that Π in secure in model M

Plan for This Lecture...

General *template*:

1. Identify the task
2. Come up with precise threat model M (a.k.a security model)
   - Adversary/Attack: What are the adversary's capabilities?
   - Security Goal: What does it mean to be secure?
3. Construct a scheme Π
4. Formally prove that Π in secure in model M

secret communication with shared keys

of longer messages

Computational

~~Perfect~~ secrecy against eavesdroppers

Computational

One-time pad

Plan for This Lecture...

General *template*:

1. Identify the task
2. Come up with precise threat model M (a.k.a security model)
   - Adversary/Attack: What are the adversary's capabilities?
   - Security Goal: What does it mean to be secure?
3. Construct a scheme Π
4. Formally prove that Π in secure in model M

secret communication with shared keys

of longer messages

Computational

~~Perfect~~ secrecy against eavesdroppers

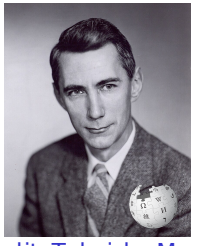
Computational

One-time pad

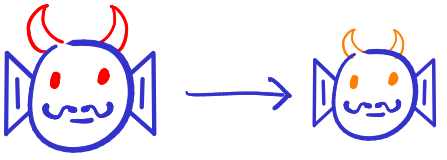
under certain assumption

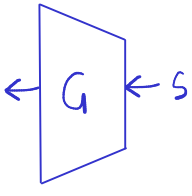
↳ First security reduction!

Plan for This Lecture...



Credit: Tekniska Museet

1. Limitations of Perfect Secrecy: Shannon's Impossibility
2. Bypassing Shannon's Impossibility 
3. Pseudo-Random Generators (PRGs) and Computational OTP



Plan for This Lecture

1. Limitations of Perfect Secrecy: Shannon's Impossibility
2. Bypassing Shannon's Impossibility
3. Pseudo-Random Generators (PRGs) and Computational OTP

Shannon's Impossibility

Theorem 1 (Shannon'49)

Let $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *be any perfectly-secret encryption scheme with message space* $\mathcal{M}$ *and key-space* $\mathcal{K}$*. Then* $|\mathcal{K}| \geq |\mathcal{M}|$*.*

Shannon's Impossibility

Theorem 1 (Shannon'49)

Let $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *be any perfectly-secret encryption scheme with message space* $\mathcal{M}$ *and key-space* $\mathcal{K}$*. Then* $|\mathcal{K}| \geq |\mathcal{M}|$*.*

Proof Sketch.  Idea: proof by contradiction.

Shannon's Impossibility

Theorem 1 (Shannon'49)

Let $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *be any perfectly-secret encryption scheme with message space* $\mathcal{M}$ *and key-space* $\mathcal{K}$*. Then* $|\mathcal{K}| \geq |\mathcal{M}|$.

Proof Sketch.  Idea: proof by contradiction.

Assume for contradiction that $|K| < |M|$

Shannon's Impossibility

Theorem 1 (Shannon'49)

Let $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *be any perfectly-secret encryption scheme with message space* $\mathcal{M}$ *and key-space* $\mathcal{K}$*. Then* $|\mathcal{K}| \geq |\mathcal{M}|$*.*

Proof Sketch.  Idea: proof by contradiction.

Assume for contradiction that $|K| < |M|$

Goal: show that Π not perfectly secure

Shannon's Impossibility

Theorem 1 (Shannon'49)

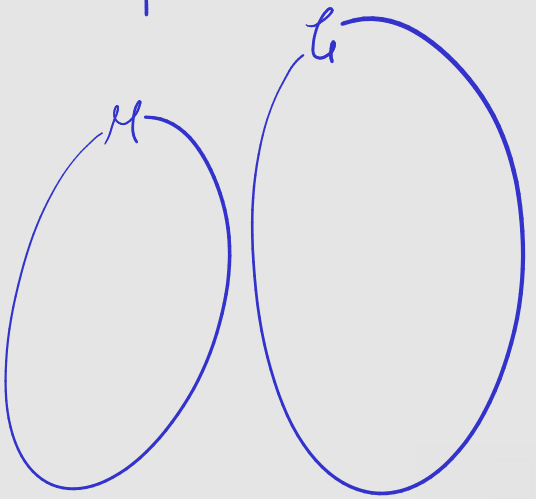
Let $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be any perfectly-secret encryption scheme with message space $\mathcal{M}$ and key-space $\mathcal{K}$. Then $|\mathcal{K}| \geq |\mathcal{M}|$.

Proof Sketch. Idea: proof by contradiction.

Assume for contradiction that $|K| < |M|$

Goal: show that Π not perfectly secure

Fix any message $m^* \in M$ and c^* in m^*'s ciphertext-space



Shannon's Impossibility

Theorem 1 (Shannon'49)

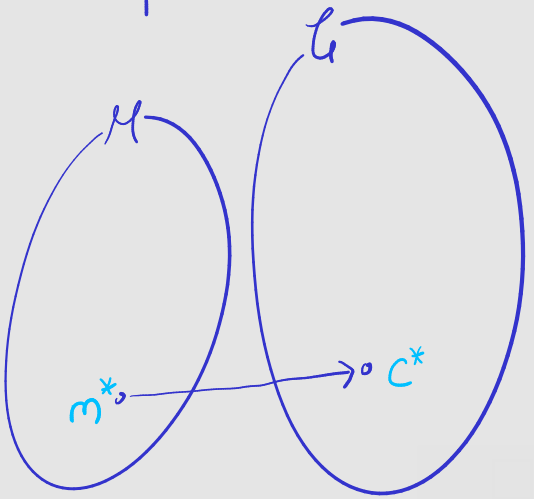
Let $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *be any perfectly-secret encryption scheme with message space* $\mathcal{M}$ *and key-space* $\mathcal{K}$*. Then* $|\mathcal{K}| \geq |\mathcal{M}|$*.*

Proof Sketch. Idea: proof by contradiction.

Assume for contradiction that $|K| < |M|$

Goal: show that Π not perfectly secure

Fix any message $m^* \in M$ and c^* in m^*'s ciphertext-space



Shannon's Impossibility

Theorem 1 (Shannon'49)

Let $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be any perfectly-secret encryption scheme with message space $\mathcal{M}$ and key-space $\mathcal{K}$. Then $|\mathcal{K}| \geq |\mathcal{M}|$.

Proof Sketch. Idea: proof by contradiction.

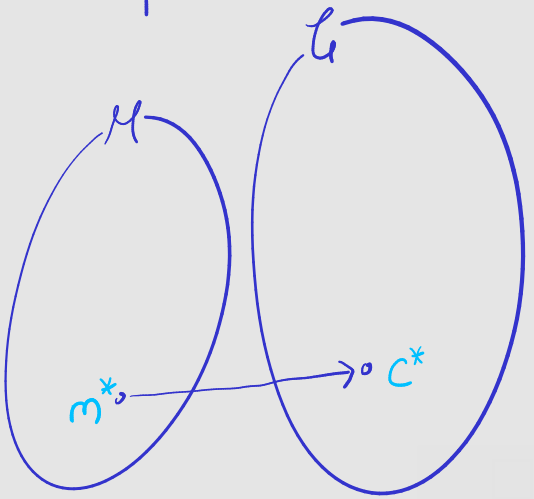
Assume for contradiction that $|K| < |M|$

Goal: show that Π not perfectly secure

Fix any message $m^* \in M$ and c^* in m^*'s ciphertext-space

Consider set $M_{c^*} \subseteq M$ defined as

$$\{m \in M : \exists k \in K \text{ s.t. } Dec(k, c^*) = m\}$$



Shannon's Impossibility

Theorem 1 (Shannon'49)

Let $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be any perfectly-secret encryption scheme with message space $\mathcal{M}$ and key-space $\mathcal{K}$. Then $|\mathcal{K}| \geq |\mathcal{M}|$.

Proof Sketch. Idea: proof by contradiction.

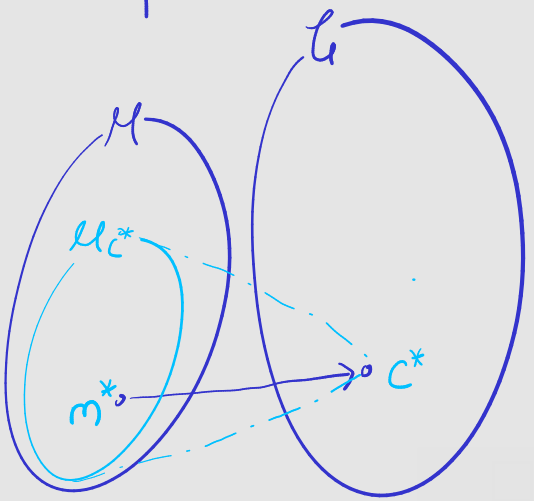
Assume for contradiction that $|K| < |M|$

Goal: show that Π not perfectly secure

Fix any message $m^* \in M$ and c^* in m^*'s ciphertext-space

Consider set $M_{c^*} \subseteq M$ defined as

$$\{m \in M : \exists k \in K \text{ s.t. } Dec(k, c^*) = m\}$$



Shannon's Impossibility

Theorem 1 (Shannon'49)

Let $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be any perfectly-secret encryption scheme with message space $\mathcal{M}$ and key-space $\mathcal{K}$. Then $|\mathcal{K}| \geq |\mathcal{M}|$.

Proof Sketch. Idea: proof by contradiction.

Assume for contradiction that $|\mathcal{K}| < |\mathcal{M}|$

Goal: show that Π not perfectly secure

Fix any message $m^* \in \mathcal{M}$ and c^* in m^*'s ciphertext-space

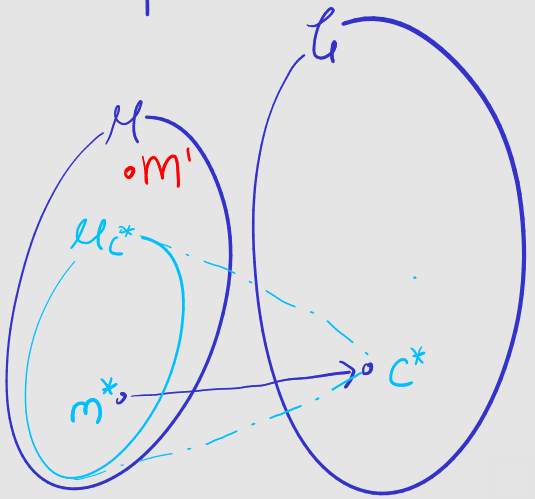
Consider set $M_{c^*} \subseteq \mathcal{M}$ defined as

$\{m \in \mathcal{M} : \exists k \in \mathcal{K} \text{ s.t. } \mathsf{Dec}(k, c^*) = m\}$ ← ? Why?

Since $|M_{c^*}| \leq |\mathcal{K}| < |\mathcal{M}|$,

$\exists m' \in \mathcal{M} \setminus M_{c^*}$: c^* never decrypts to m'

$\Rightarrow (1/2)$



Shannon's Impossibility

Theorem 1 (Shannon'49)

Let $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be any perfectly-secret encryption scheme with message space $\mathcal{M}$ and key-space $\mathcal{K}$. Then $|\mathcal{K}| \geq |\mathcal{M}|$.

Proof Sketch. Idea: proof by contradiction.

(2/2)

Consider (m^*, m') and $\mathsf{Eve}_{c^*}(c) := \begin{cases} \text{'left'} & \text{if } c = c^* \\ \text{'right'} & \text{otherwise} \end{cases}$

Shannon's Impossibility

Theorem 1 (Shannon'49)

Let $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be any perfectly-secret encryption scheme with message space $\mathcal{M}$ and key-space $\mathcal{K}$. Then $|\mathcal{K}| \geq |\mathcal{M}|$.

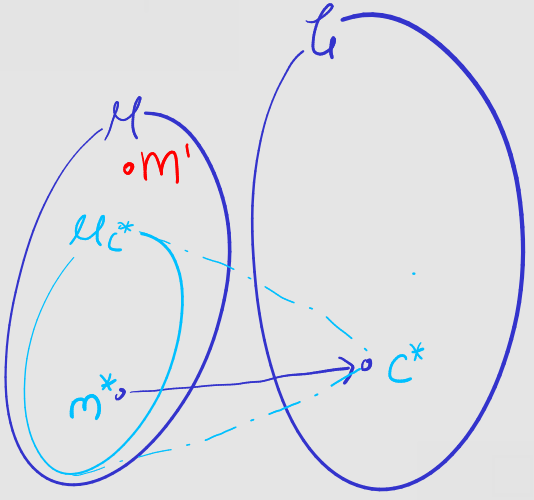
Proof Sketch. Idea: proof by contradiction.

(2/2)

Consider (m^*, m') and $\mathsf{Eve}_{c^*}(c) := \begin{cases} \text{'left'} & \text{if } c = c^* \\ \text{'right'} & \text{otherwise} \end{cases}$

We have:

1) for m^*: $\Pr_{\substack{k \leftarrow \mathsf{Gen} \\ c \leftarrow \mathsf{Enc}(m^*)}} [\mathsf{Eve}_{c^*}(c) = \text{'left'}] > 0$



Shannon's Impossibility

Theorem 1 (Shannon'49)

Let $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be any perfectly-secret encryption scheme with message space $\mathcal{M}$ and key-space $\mathcal{K}$. Then $|\mathcal{K}| \geq |\mathcal{M}|$.

Proof Sketch. Idea: proof by contradiction.

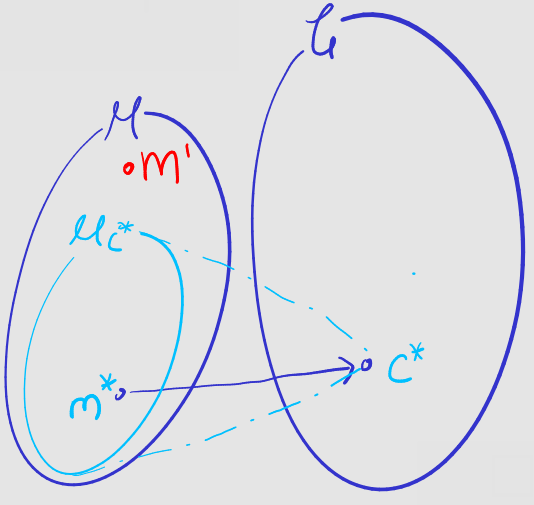
(2/2)

Consider (m^*, m') and $\mathrm{Eve}_{c^*}(c) := \begin{cases} \text{'left'} & \text{if } c = c^* \\ \text{'right'} & \text{otherwise} \end{cases}$

We have:

i) for m^*: $\Pr_{\substack{k \leftarrow \mathsf{Gen} \\ c \leftarrow \mathsf{Enc}(m^*)}} [\mathrm{Eve}_{c^*}(c) = \text{'left'}] > 0$

ii) for m': $\Pr_{\substack{k \leftarrow \mathsf{Gen} \\ c \leftarrow \mathsf{Enc}(m')}} [\mathrm{Eve}_{c^*}(c) = \text{'left'}] = 0$



Shannon's Impossibility

Theorem 1 (Shannon'49)

Let $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *be any perfectly-secret encryption scheme with message space* $\mathcal{M}$ *and key-space* $\mathcal{K}$*. Then* $|\mathcal{K}| \geq |\mathcal{M}|$*.*

Proof Sketch. Idea: proof by contradiction.

(2/2)

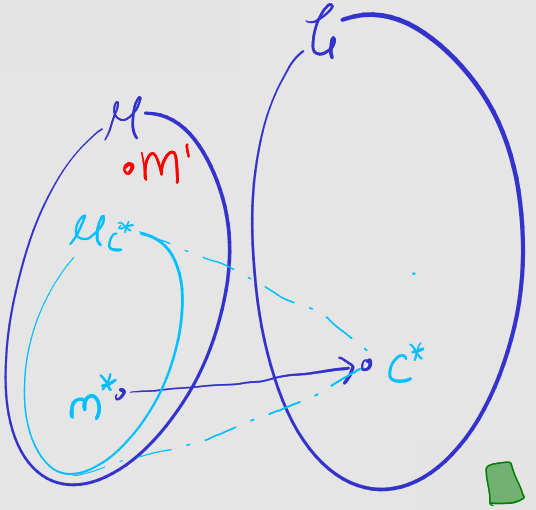
Consider (m^*, m') and $\mathsf{Eve}_{c^*}(c) := \begin{cases} \text{'left'} & \text{if } c = c^* \\ \text{'right'} & \text{otherwise} \end{cases}$

We have:

i) for m^*: $\Pr_{\substack{k \leftarrow \mathsf{Gen} \\ c \leftarrow \mathsf{Enc}(m^*)}} [\mathsf{Eve}_{c^*}(c) = \text{'left'}] > 0$

ii) for m': $\Pr_{\substack{k \leftarrow \mathsf{Gen} \\ c \leftarrow \mathsf{Enc}(m')}} [\mathsf{Eve}_{c^*}(c) = \text{'left'}] = 0$

$\Rightarrow$ Π is not perfectly secure ↯



What Do We Do in Face of Shannon's Impossibility?

Defintion 1 (SKE Imitation Game)

An SKE $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is perfectly-secret if for every eavesdropper Eve and pair of messages* $(m_0, m_1) \in \mathcal{M}$*:*

$$\Pr_{\substack{k \leftarrow \mathsf{Gen} \\ c \leftarrow \mathsf{Enc}(k, m_0)}} [\mathsf{Eve}(c) = 0] - \Pr_{\substack{k \leftarrow \mathsf{Gen} \\ c \leftarrow \mathsf{Enc}(k, m_1)}} [\mathsf{Eve}(c) = 0] = 0$$

What Do We Do in Face of Shannon's Impossibility?

- You compromise.
  - Kerckhoffs' principle: *"The system should be, if not theoretically unbreakable, unbreakable in practice."*

Defintion 1 (SKE Imitation Game)

An SKE $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is perfectly-secret if for every eavesdropper Eve and pair of messages* $(m_0, m_1) \in \mathcal{M}$*:*

$$\Pr_{\substack{k \leftarrow \mathsf{Gen} \\ c \leftarrow \mathsf{Enc}(k, m_0)}} [\mathsf{Eve}(c) = 0] - \Pr_{\substack{k \leftarrow \mathsf{Gen} \\ c \leftarrow \mathsf{Enc}(k, m_1)}} [\mathsf{Eve}(c) = 0] = 0$$

What Do We Do in Face of Shannon's Impossibility?

- You compromise.
  - Kerckhoffs' principle: *"The system should be, if not theoretically unbreakable, unbreakable in practice."*

Defintion 1 (SKE Imitation Game)

An SKE $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is perfectly-secret if for every eavesdropper* Eve *and pair of messages* $(m_0, m_1) \in \mathcal{M}$*:*

$$\Pr_{\substack{k \leftarrow \mathsf{Gen} \\ c \leftarrow \mathsf{Enc}(k, m_0)}}[\mathsf{Eve}(c) = 0] - \Pr_{\substack{k \leftarrow \mathsf{Gen} \\ c \leftarrow \mathsf{Enc}(k, m_1)}}[\mathsf{Eve}(c) = 0] = 0$$

- Compromise two aspects of Defintion 1:
  1. Restrict to *computationally*-bounded Eve
  2. Allow "slack": Eve may distinguish, but with "very small" prob.

What Do We Do in Face of Shannon's Impossibility?

- You compromise.
  - Kerckhoffs' principle: *"The system should be, if not theoretically unbreakable, unbreakable in practice."*

Defintion 1 (SKE Imitation Game)

An SKE $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is perfectly-secret if for every eavesdropper* Eve *and pair of messages* $(m_0, m_1) \in \mathcal{M}$*:*

$$\Pr_{\substack{k \leftarrow \mathsf{Gen} \\ c \leftarrow \mathsf{Enc}(k, m_0)}}[\mathsf{Eve}(c) = 0] - \Pr_{\substack{k \leftarrow \mathsf{Gen} \\ c \leftarrow \mathsf{Enc}(k, m_1)}}[\mathsf{Eve}(c) = 0] = 0$$

- Compromise two aspects of Defintion 1:
  1. Restrict to *computationally*-bounded Eve
  2. Allow "slack": Eve may distinguish, but with "very small" prob.
- Turns out both compromises necessary!

Plan for This Lecture

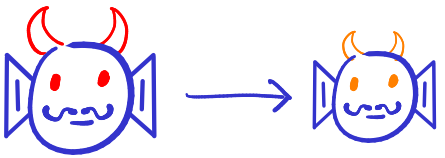
1 Limitations of Perfect Secrecy: Shannon's Impossibility

2 Bypassing Shannon's Impossibility

3 Pseudo-Random Generators (PRGs) and Computational OTP

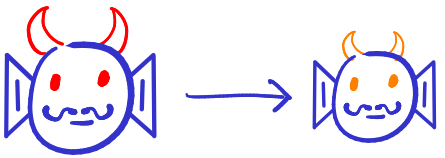
First Compromise: Computationally Bound Eve...

- Restrict to *probabilistic polynomial-time* (PPT) Eves:
  randomised Eve that runs in time $p(n)$, for some polynomial p



First Compromise: Computationally Bound Eve...

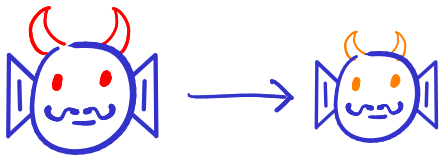
- Restrict to *probabilistic polynomial-time* (PPT) Eves: randomised Eve that runs in time $p(n)$, for some polynomial p



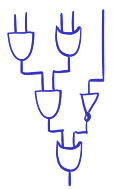
- Why PPT?
  - "Captures" efficient computation
  - The exact model of computation (Turing Machines, Random Access Machine) doesn't matter
    - Church-Turing thesis: all models of computation are polynomially equivalent
  - Polynomials have nice closure properties
  - Randomness allowed since it is allowed for honest algorithms

First Compromise: Computationally Bound Eve...

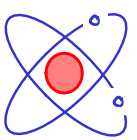
- Restrict to *probabilistic polynomial-time* (PPT) Eves: randomised Eve that runs in time $p(n)$, for some polynomial p



- Why PPT?
  - "Captures" efficient computation
  - The exact model of computation (Turing Machines, Random Access Machine) doesn't matter
    - Church-Turing thesis: all models of computation are polynomially equivalent
  - Polynomials have nice closure properties
  - Randomness allowed since it is allowed for honest algorithms
- Some stronger models for Eve:



  - Polynomial-sized family of circuits: allows "non-uniform" advice
  - Quantum polynomial-time algorithms (Lecture 10)



First Compromise: Computationally Bound Eve...

Candidate Defintion 1

An SKE $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is computationally-secret if for every PPT eavesdropper Eve* 

$$\Pr_{\substack{(m_0,m_1)\leftarrow \mathsf{Eve} \\ k\leftarrow \mathsf{Gen} \\ c\leftarrow \mathsf{Enc}(k,m_0)}}[\mathsf{Eve}(c)=0] - \Pr_{\substack{(m_0,m_1)\leftarrow \mathsf{Eve} \\ k\leftarrow \mathsf{Gen} \\ c\leftarrow \mathsf{Enc}(k,m_1)}}[\mathsf{Eve}(c)=0] = 0$$

First Compromise: Computationally Bound Eve...

Candidate Defintion 1

An SKE $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is computationally-secret if for every PPT eavesdropper Eve*

$$\Pr_{\substack{(m_0,m_1)\leftarrow \mathsf{Eve} \\ k\leftarrow \mathsf{Gen} \\ c\leftarrow \mathsf{Enc}(k,m_0)}}[\mathsf{Eve}(c)=0] - \Pr_{\substack{(m_0,m_1)\leftarrow \mathsf{Eve} \\ k\leftarrow \mathsf{Gen} \\ c\leftarrow \mathsf{Enc}(k,m_1)}}[\mathsf{Eve}(c)=0] = 0$$

Exercise 1

Show that Shannon's impossibility extends to Candidate Defintion 1.

- *Hint: use similar strategy to Theorem 1.*

First Compromise: Computationally Bound Eve...

Candidate Defintion 1

An SKE $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is computationally-secret if for every PPT eavesdropper Eve*

$$\Pr_{\substack{(m_0,m_1)\leftarrow \mathsf{Eve} \\ k\leftarrow \mathsf{Gen} \\ c\leftarrow \mathsf{Enc}(k,m_0)}}[\mathsf{Eve}(c)=0] - \Pr_{\substack{(m_0,m_1)\leftarrow \mathsf{Eve} \\ k\leftarrow \mathsf{Gen} \\ c\leftarrow \mathsf{Enc}(k,m_1)}}[\mathsf{Eve}(c)=0] = 0$$

Exercise 1

Show that Shannon's impossibility extends to Candidate Defintion 1.

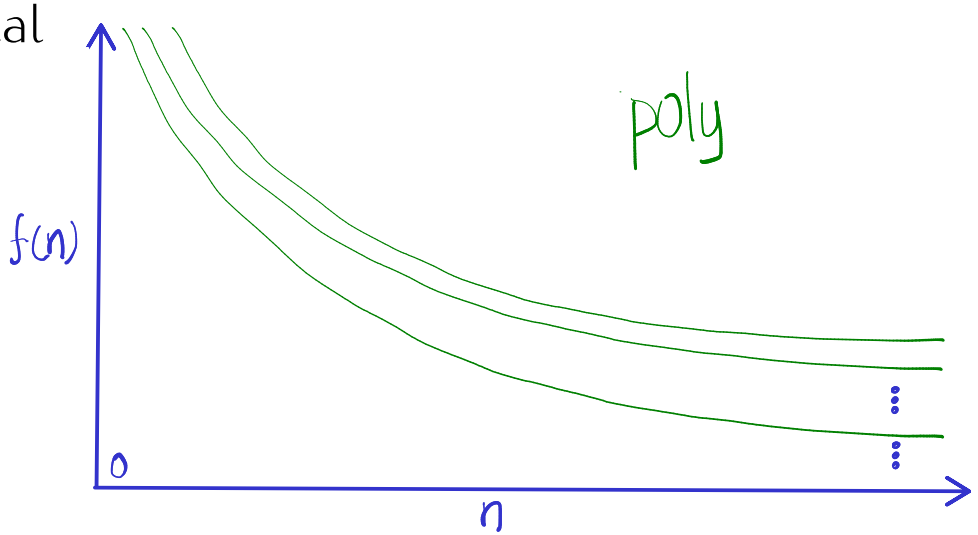
- *Hint: use similar strategy to Theorem 1.*
- *Take-away: Eve can distinguish with a "very small" probability*

Second Compromise: Allow "Slack"...

- Okay if Eve distinguishes with "very small" probability

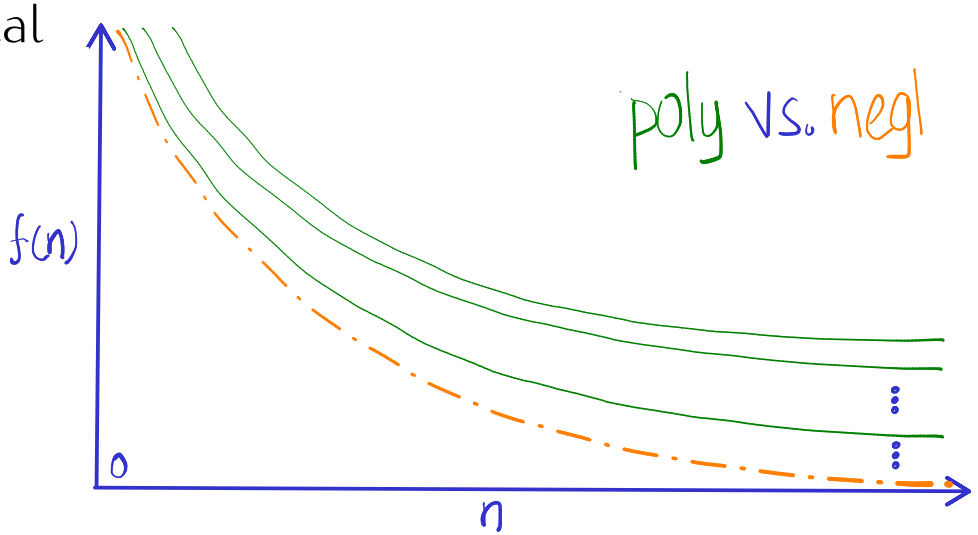
Second Compromise: Allow "Slack"...

- Okay if Eve distinguishes with "very small" probability
- Quantify "very small" using *negligible* function:
  - Intuitive def.: function eventually smaller than *every* inverse polynomial



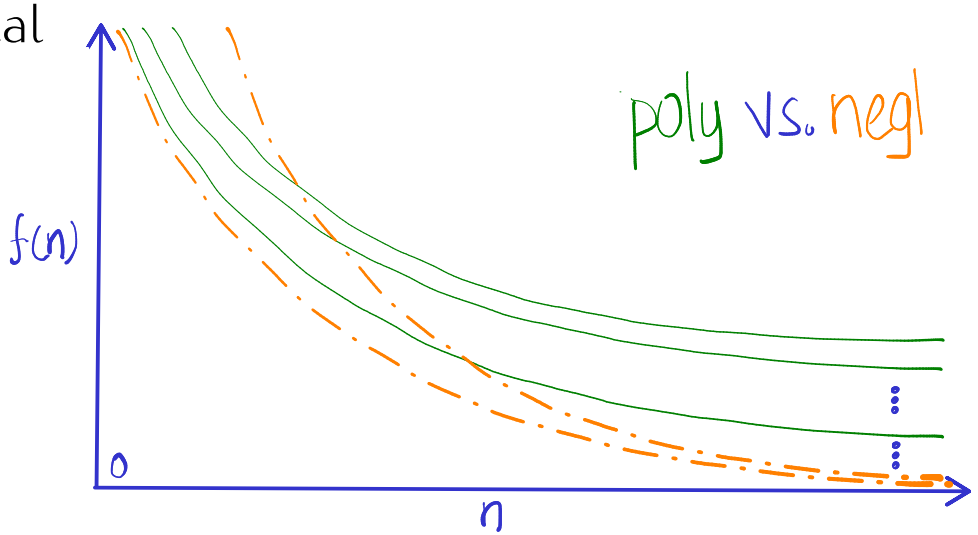
Second Compromise: Allow "Slack"...

- Okay if Eve distinguishes with "very small" probability
- Quantify "very small" using *negligible* function:
  - Intuitive def.: function eventually smaller than *every* inverse polynomial



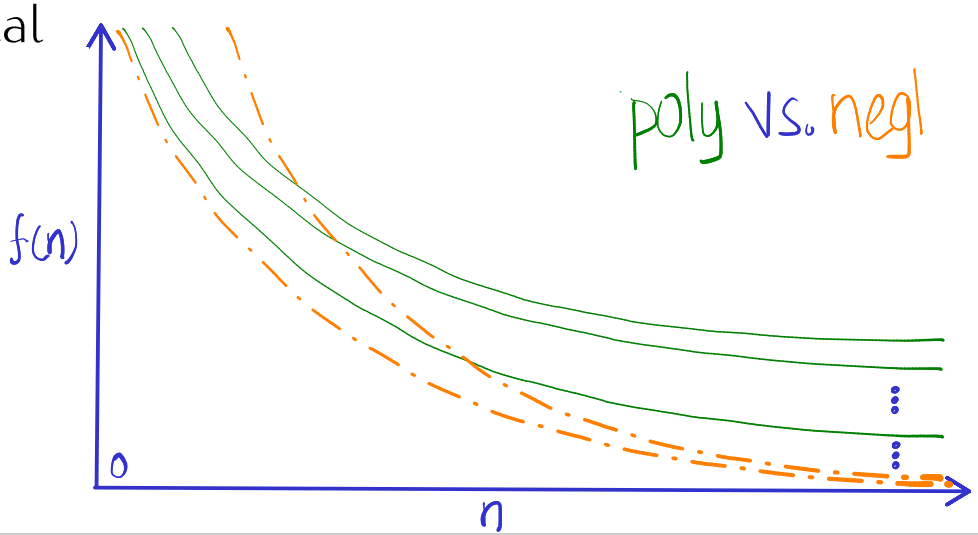
Second Compromise: Allow "Slack"...

- Okay if Eve distinguishes with "very small" probability
- Quantify "very small" using *negligible* function:
  - Intuitive def.: function eventually smaller than *every* inverse polynomial



Second Compromise: Allow "Slack"...

- Okay if Eve distinguishes with "very small" probability
- Quantify "very small" using *negligible* function:
  - Intuitive def.: function eventually smaller than *every* inverse polynomial

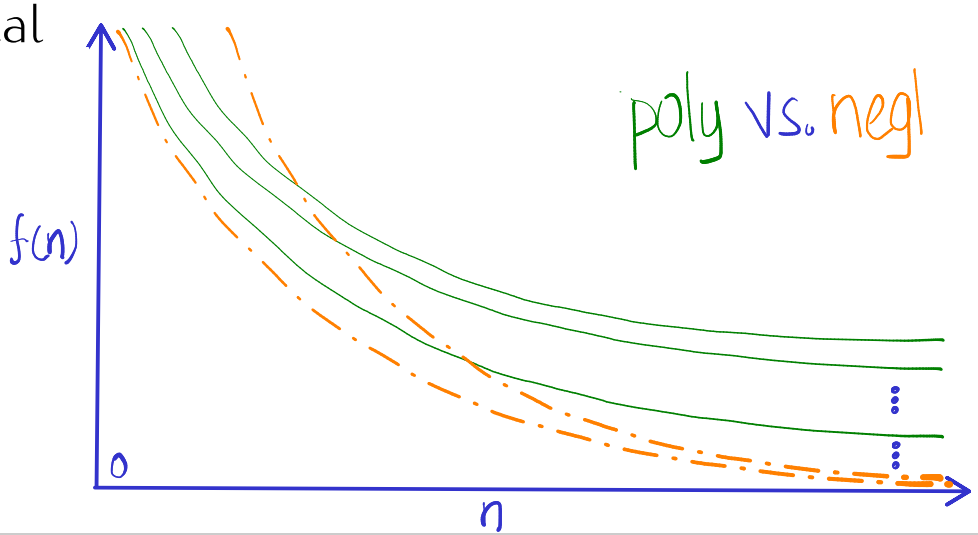


Defintion 2

A function $f : \mathbb{N} \to \mathbb{R}^+$ *is negligible if for every polynomial* p *and sufficiently large* n, $f(n) < 1/p(n)$ *holds.*

Second Compromise: Allow "Slack"...

- Okay if Eve distinguishes with "very small" probability
- Quantify "very small" using *negligible* function:
  - Intuitive def.: function eventually smaller than *every* inverse polynomial



Defintion 2

A function $f : \mathbb{N} \to \mathbb{R}^+$ is negligible if for every polynomial p and sufficiently large n, $f(n) < 1/p(n)$ holds.

- Why negligible? Like PPT, it behaves nicely.

Second Compromise: Allow "Slack"...

Defintion 2

A function $f : \mathbb{N} \to \mathbb{R}^+$ is negligible if for every polynomial p and sufficiently large n, $f(n) < 1/p(n)$ holds.

Negligible or not?

1. $f_1(n) := 1/314159n^{314159}$
2. $f_2(n) := 1/2^n$

Second Compromise: Allow "Slack"...

Defintion 2

A function $f : \mathbb{N} \to \mathbb{R}^+$ is negligible if for every polynomial p and sufficiently large n, $f(n) < 1/p(n)$ holds.

Negligible or not?

1. $f_1(n) := 1/314159n^{314159}$
2. $f_2(n) := 1/2^n$ → "inverse exponential"
3. $f_3(n) := \begin{cases} 1/2^n & \text{for odd } n \\ 1/314159n^{314159} & \text{for even } n \end{cases}$ "hybrid of f_1 & f_2"

Second Compromise: Allow "Slack"...

Defintion 2

A function $f : \mathbb{N} \to \mathbb{R}^+$ is negligible if for every polynomial p and sufficiently large n, $f(n) < 1/p(n)$ holds.

Negligible or not?

1. $f_1(n) := 1/314159n^{314159}$
2. $f_2(n) := 1/2^n$ → "inverse exponential"
3. $f_3(n) := \begin{cases} 1/2^n & \text{for odd } n \\ 1/314159n^{314159} & \text{for even } n \end{cases}$ "hybrid of f_1 & f_2"
4. $f_4(n) := n^{-log(n)}$

Second Compromise: Allow "Slack"...

Defintion 2

A function $f : \mathbb{N} \to \mathbb{R}^+$ is negligible if for every polynomial p and sufficiently large n, $f(n) < 1/p(n)$ holds.

Negligible or not?

1. $f_1(n) := 1/314159n^{314159}$
2. $f_2(n) := 1/2^n$ ⇝ "inverse exponental"
3. $f_3(n) := \begin{cases} 1/2^n & \text{for odd } n \\ 1/314159n^{314159} & \text{for even } n \end{cases}$ "hybrid of f_1 & f_2"
4. $f_4(n) := n^{-log(n)}$ ⇝ "inverse quasi-polynomial"
5. $f_5(n) := p(n)/2^n$, for a very large polynomial $p(n)$
6. $f_6(n) := n^{log(n)}/2^n$

Second Compromise: Allow "Slack"...

Defintion 2

A function $f : \mathbb{N} \to \mathbb{R}^+$ is negligible if for every polynomial p and sufficiently large n, $f(n) < 1/p(n)$ holds.

Negligible or not?

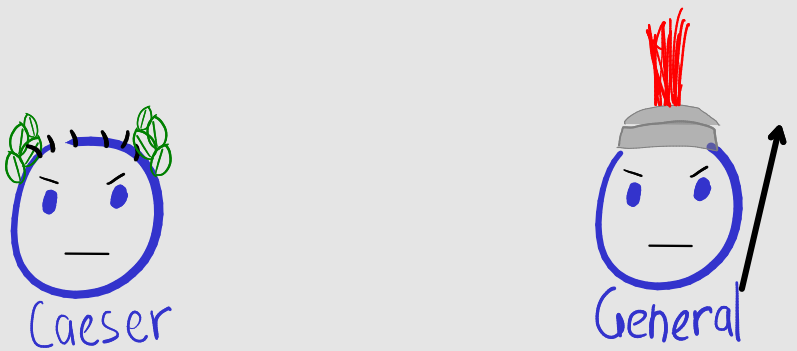
1. $f_1(n) := 1/314159n^{314159}$
2. $f_2(n) := 1/2^n$ ⇝ "inverse exponental"
3. $f_3(n) := \begin{cases} 1/2^n & \text{for odd } n \\ 1/314159n^{314159} & \text{for even } n \end{cases}$ "hybrid of f_1 & f_2"
4. $f_4(n) := n^{-log(n)}$ ⇝ "inverse quasi-polynomial"
5. $f_5(n) := p(n)/2^n$, for a very large polynomial $p(n)$
6. $f_6(n) := n^{log(n)}/2^n$

- To show that $f(n)$ is *non-negligible*, show that there exists a polynomial p such that $f(n) > 1/p(n)$ for *infinitely often* ns.

Incorporating Security Parameter into SKE Definition

Definition 2 (Shared/Symmetric-Key Encryption (SKE))

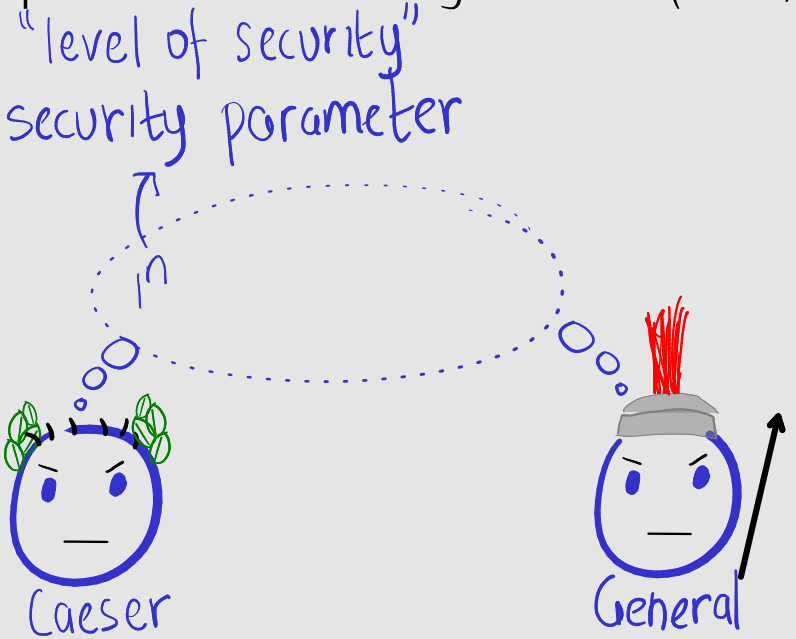
An SKE Π is a triple of efficient algorithms (**Gen**, **Enc**, **Dec**) with the following syntax:



Incorporating Security Parameter into SKE Definition

Definition 2 (Shared/Symmetric-Key Encryption (SKE))

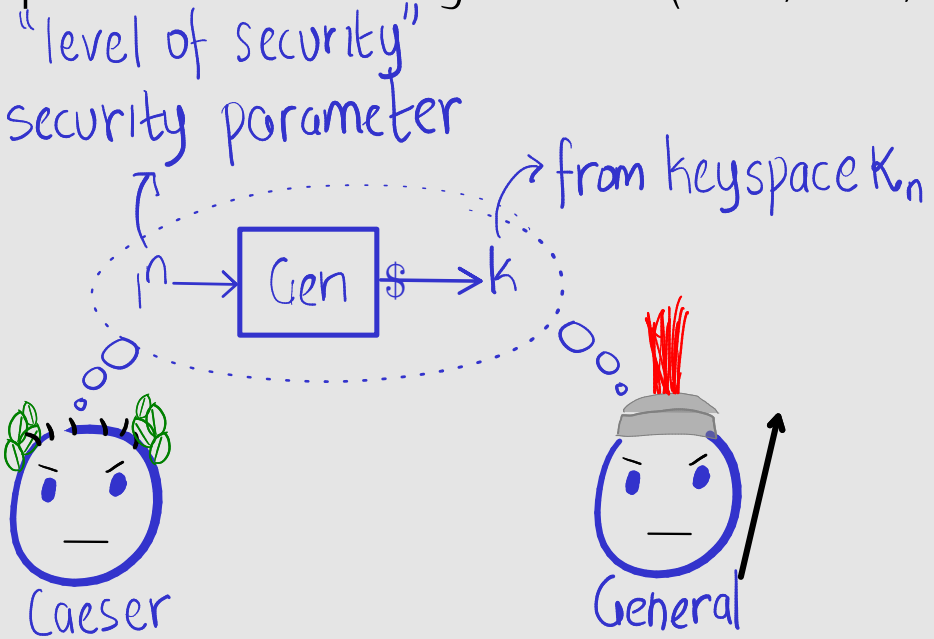
An SKE Π is a triple of efficient algorithms (**Gen**, **Enc**, **Dec**) with the following syntax:



Incorporating Security Parameter into SKE Definition

Definition 2 (Shared/Symmetric-Key Encryption (SKE))

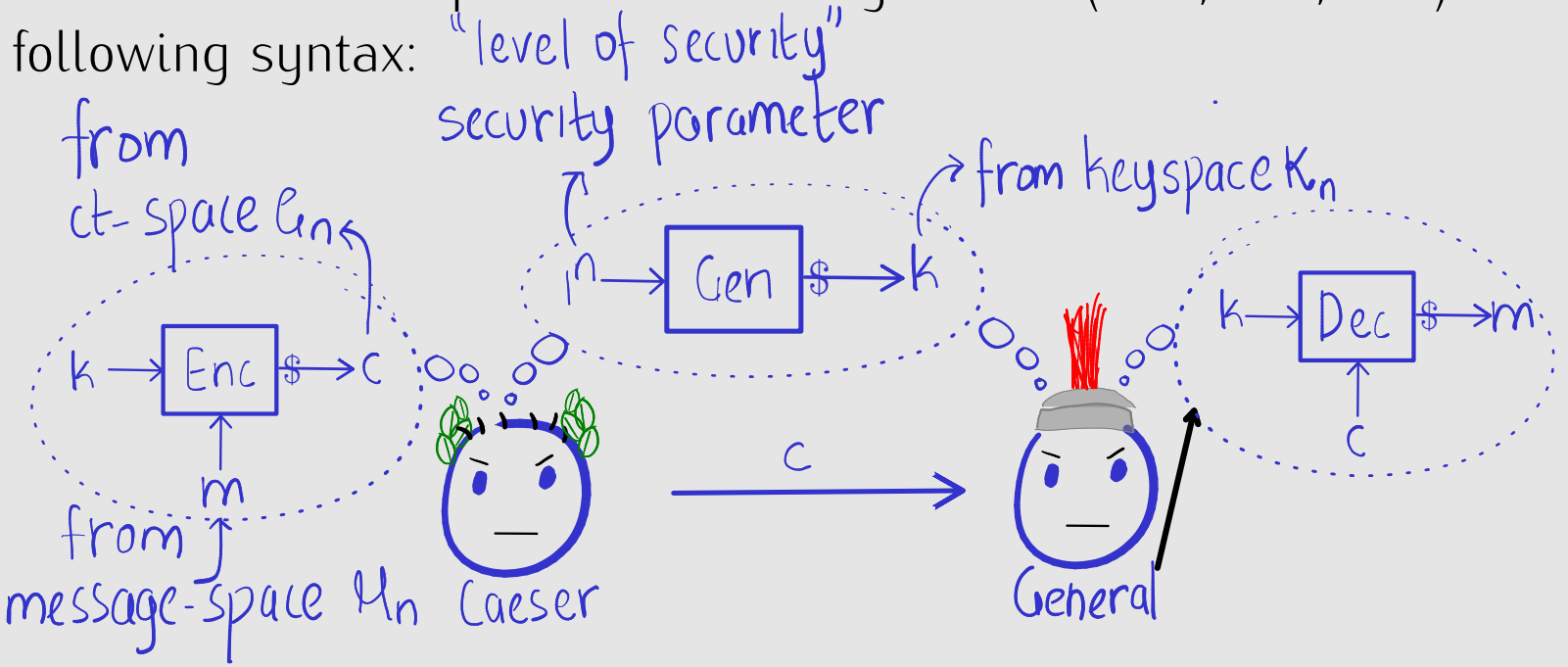
An SKE Π is a triple of efficient algorithms (**Gen**, **Enc**, **Dec**) with the following syntax:



Incorporating Security Parameter into SKE Definition

Definition 2 (Shared/Symmetric-Key Encryption (SKE))

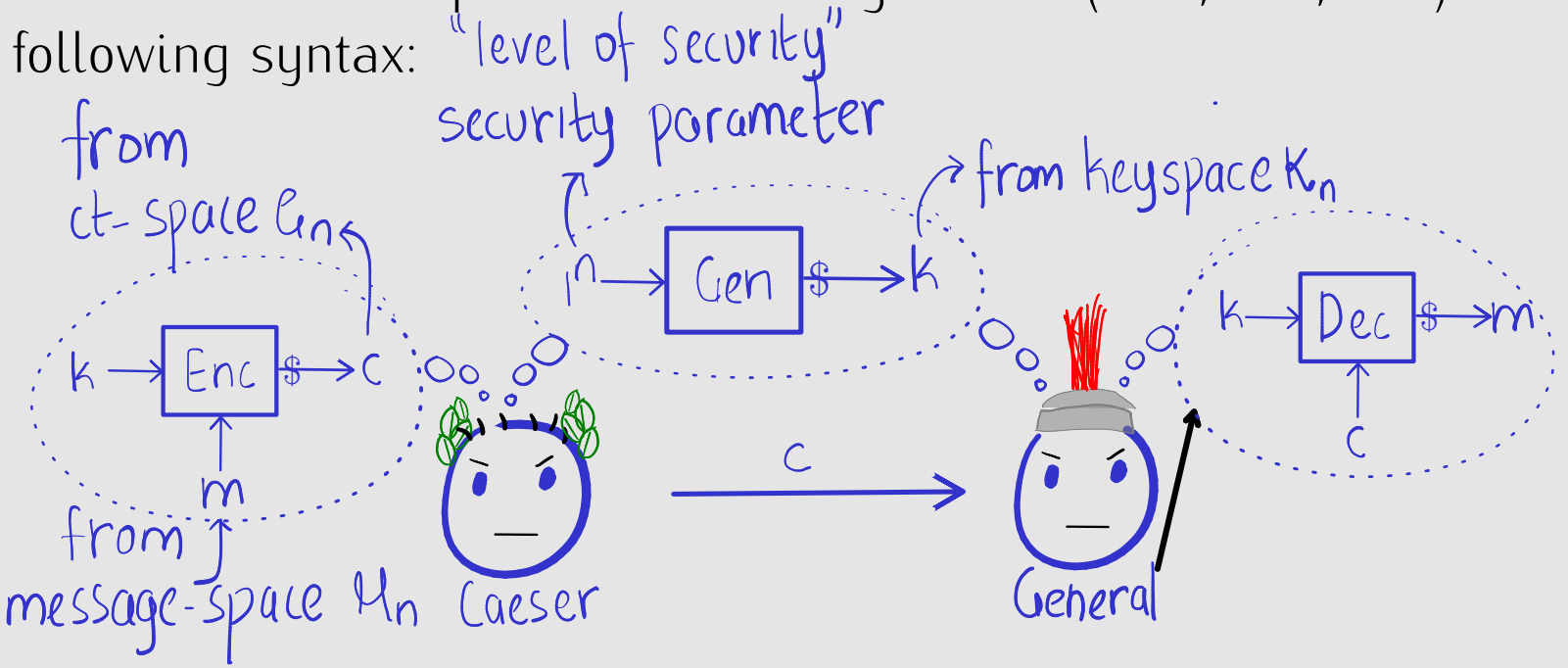
An SKE Π is a triple of efficient algorithms (**Gen**, **Enc**, **Dec**) with the following syntax:



Incorporating Security Parameter into SKE Definition

Definition 2 (Shared/Symmetric-Key Encryption (SKE))

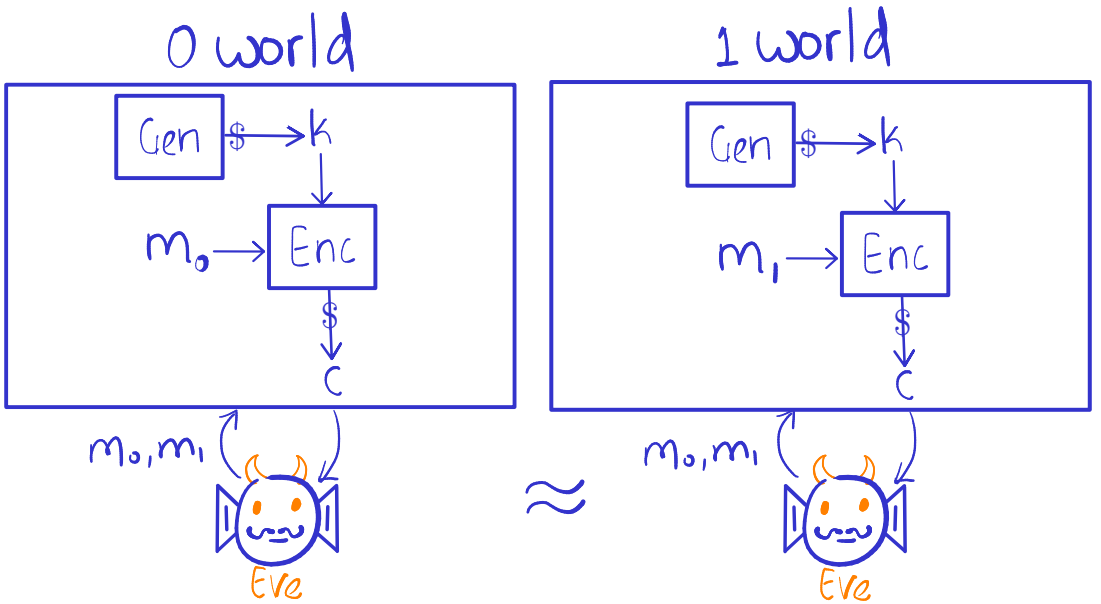
An SKE Π is a triple of efficient algorithms $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ with the following syntax:



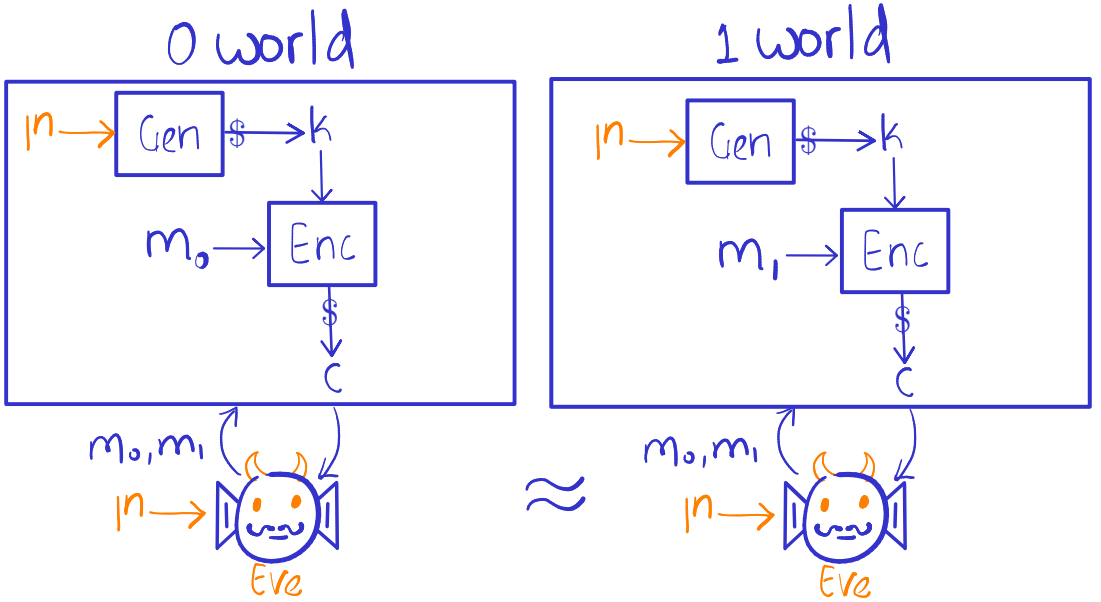
- Correctness of decryption: for every $n \in \mathbb{N}$, message $m \in \mathcal{M}_n$,

$$\Pr_{k \leftarrow \mathsf{Gen}(1^n), c \leftarrow \mathsf{Enc}(k,m)} [\mathsf{Dec}(k, c) = m] = 1$$

Let's Finally Define Computational Secrecy...



Let's Finally Define Computational Secrecy...



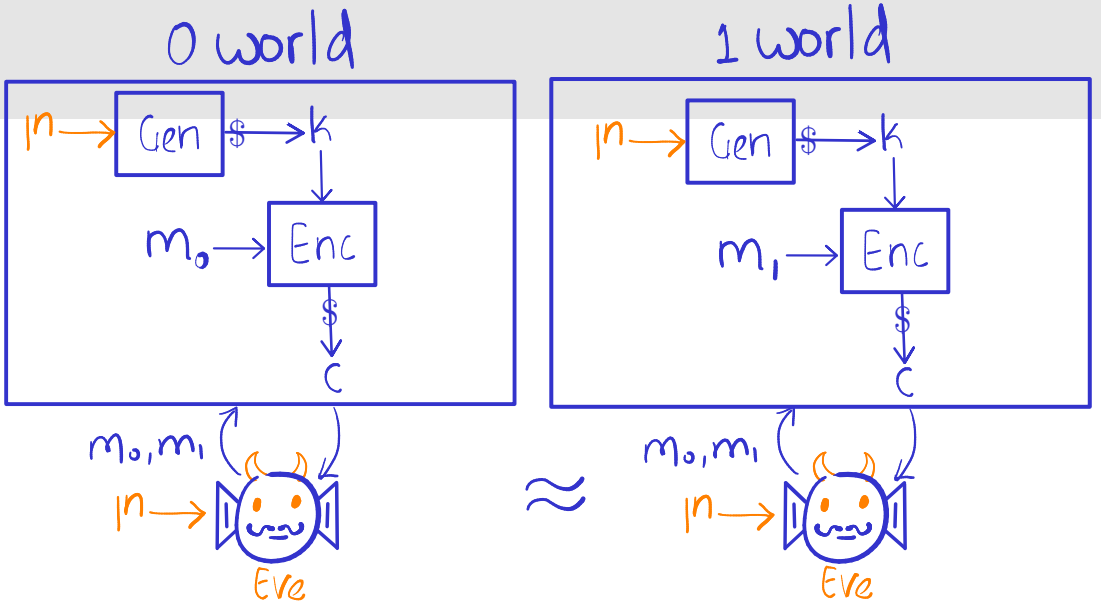
Let's Finally Define Computational Secrecy...

Defintion 3 (SKE Imitation Game for PPT Eves)

An SKE $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is computationally-secret if for every PPT eavesdropper Eve*

$$\delta(n) := \left| \Pr_{\substack{(m_0,m_1)\leftarrow \mathsf{Eve}(1^n) \\ k\leftarrow \mathsf{Gen}(1^n) \\ c\leftarrow \mathsf{Enc}(k,m_0)}} [\mathsf{Eve}(c) = 0] - \Pr_{\substack{(m_0,m_1)\leftarrow \mathsf{Eve}(1^n) \\ k\leftarrow \mathsf{Gen}(1^n) \\ c\leftarrow \mathsf{Enc}(k,m_1)}} [\mathsf{Eve}(c) = 0] \right|$$

is negligible.



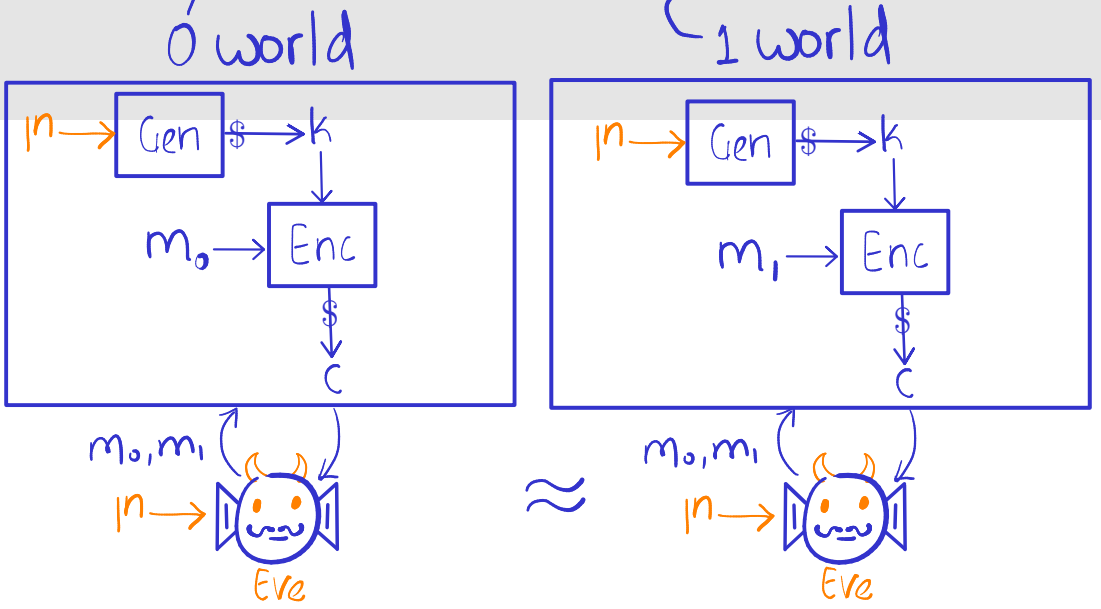
Let's Finally Define Computational Secrecy...

Defintion 3 (SKE Imitation Game for PPT Eves)

An SKE $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is computationally-secret if for every PPT eavesdropper Eve*

$$\delta(n) := \left| \Pr_{\substack{(m_0,m_1)\leftarrow \mathsf{Eve}(1^n) \\ k\leftarrow \mathsf{Gen}(1^n) \\ c\leftarrow \mathsf{Enc}(k,m_0)}} [\mathsf{Eve}(c) = 0] - \Pr_{\substack{(m_0,m_1)\leftarrow \mathsf{Eve}(1^n) \\ k\leftarrow \mathsf{Gen}(1^n) \\ c\leftarrow \mathsf{Enc}(k,m_1)}} [\mathsf{Eve}(c) = 0] \right|$$

is negligible.



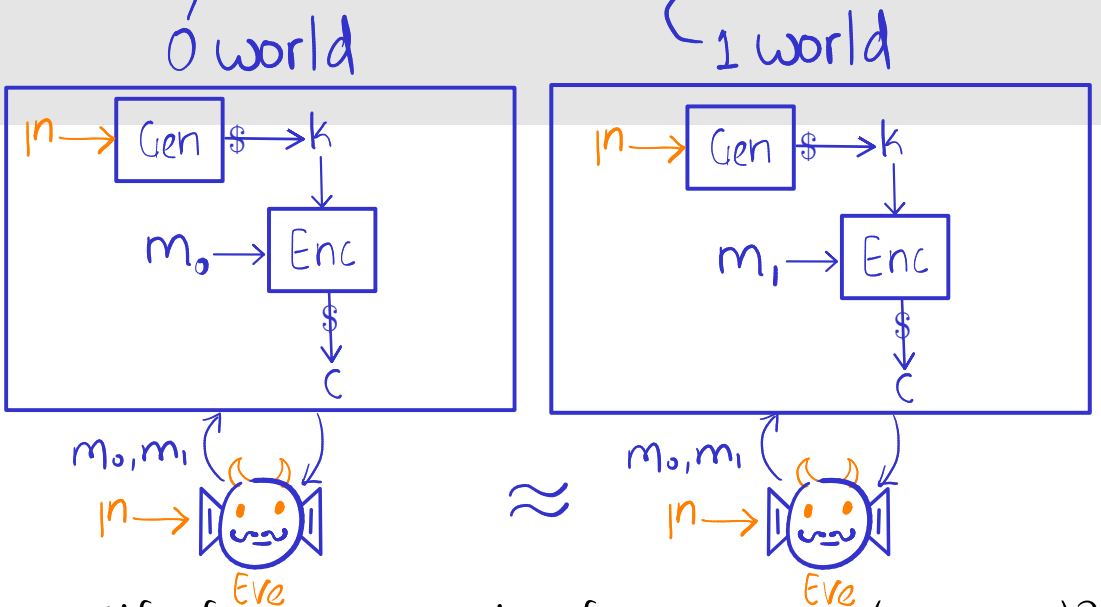
Let's Finally Define Computational Secrecy...

Defintion 3 (SKE Imitation Game for PPT Eves)

An SKE $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is computationally-secret if for every PPT eavesdropper Eve*

$$\delta(n) := \left| \Pr_{\substack{(m_0,m_1)\leftarrow \mathsf{Eve}(1^n) \\ k\leftarrow \mathsf{Gen}(1^n) \\ c\leftarrow \mathsf{Enc}(k,m_0)}} [\mathsf{Eve}(c) = 0] - \Pr_{\substack{(m_0,m_1)\leftarrow \mathsf{Eve}(1^n) \\ k\leftarrow \mathsf{Gen}(1^n) \\ c\leftarrow \mathsf{Enc}(k,m_1)}} [\mathsf{Eve}(c) = 0] \right|$$

is negligible.



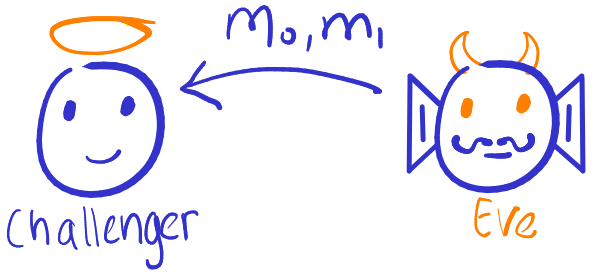
? What if we quantify for every pair of messages (m_0, m_1)?

Let's Finally Define Computational Secrecy...

A Second Way

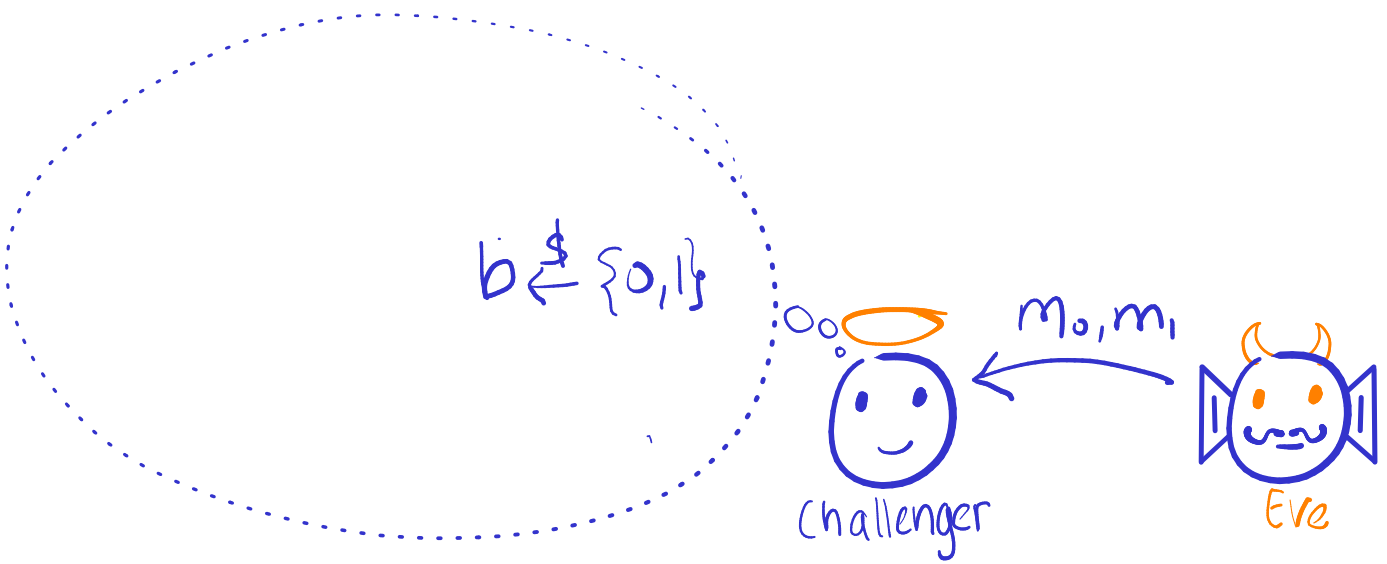
Let's Finally Define Computational Secrecy...

A Second Way



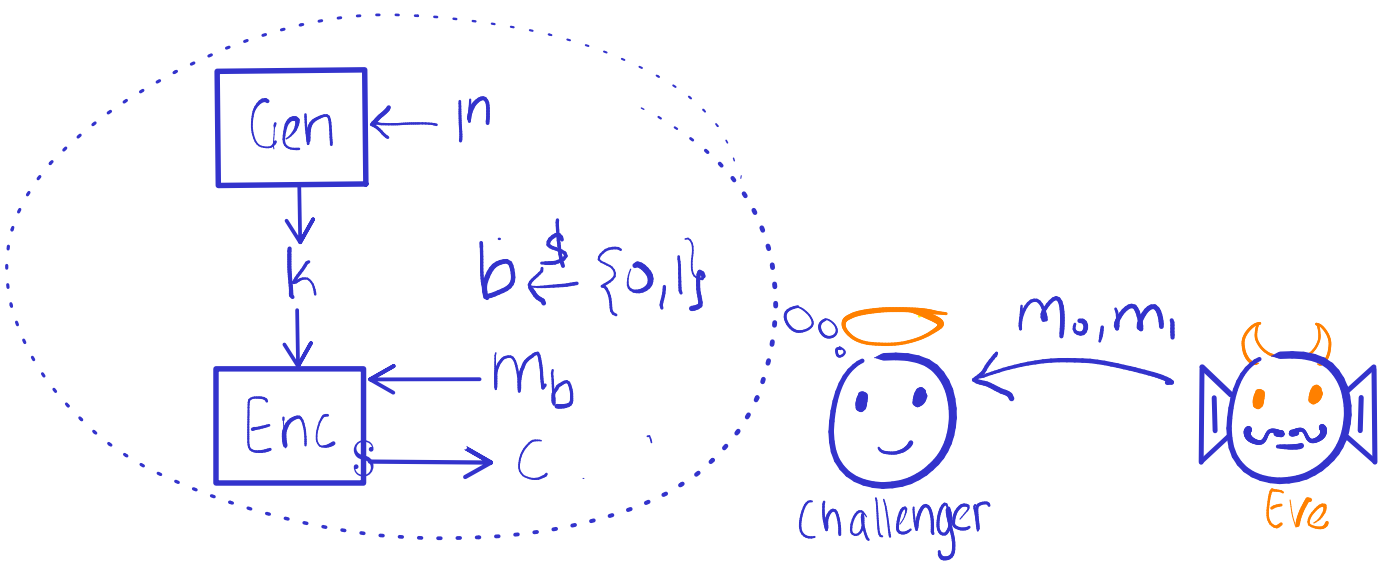
Let's Finally Define Computational Secrecy...

A Second Way



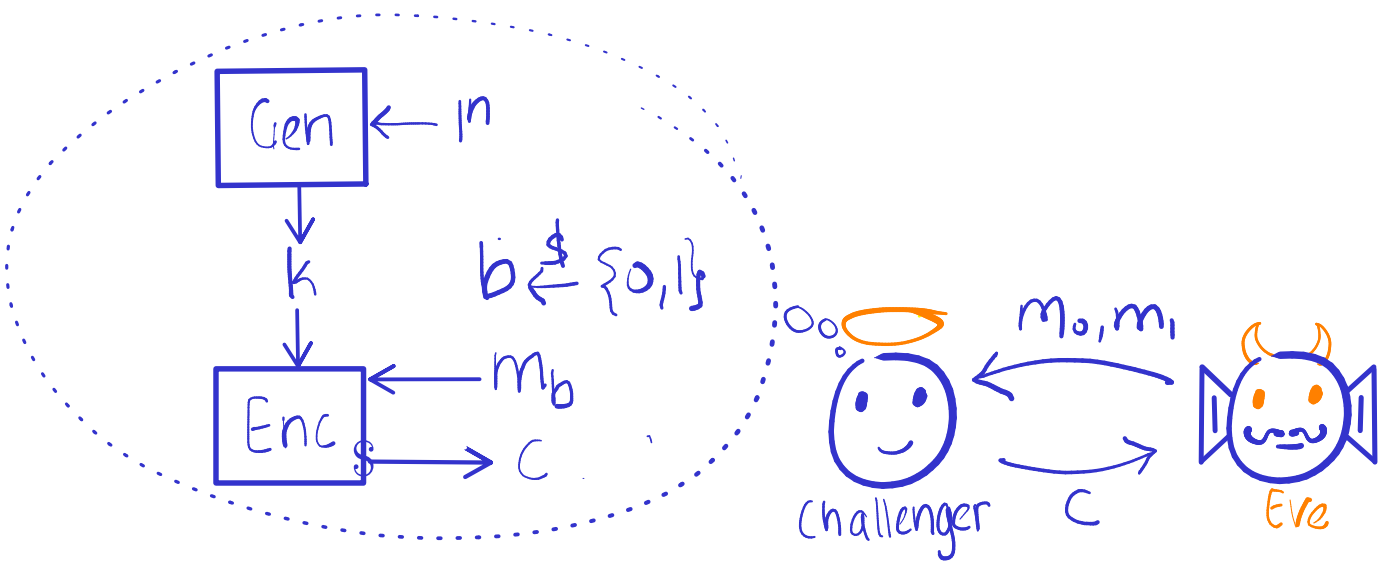
Let's Finally Define Computational Secrecy...

A Second Way



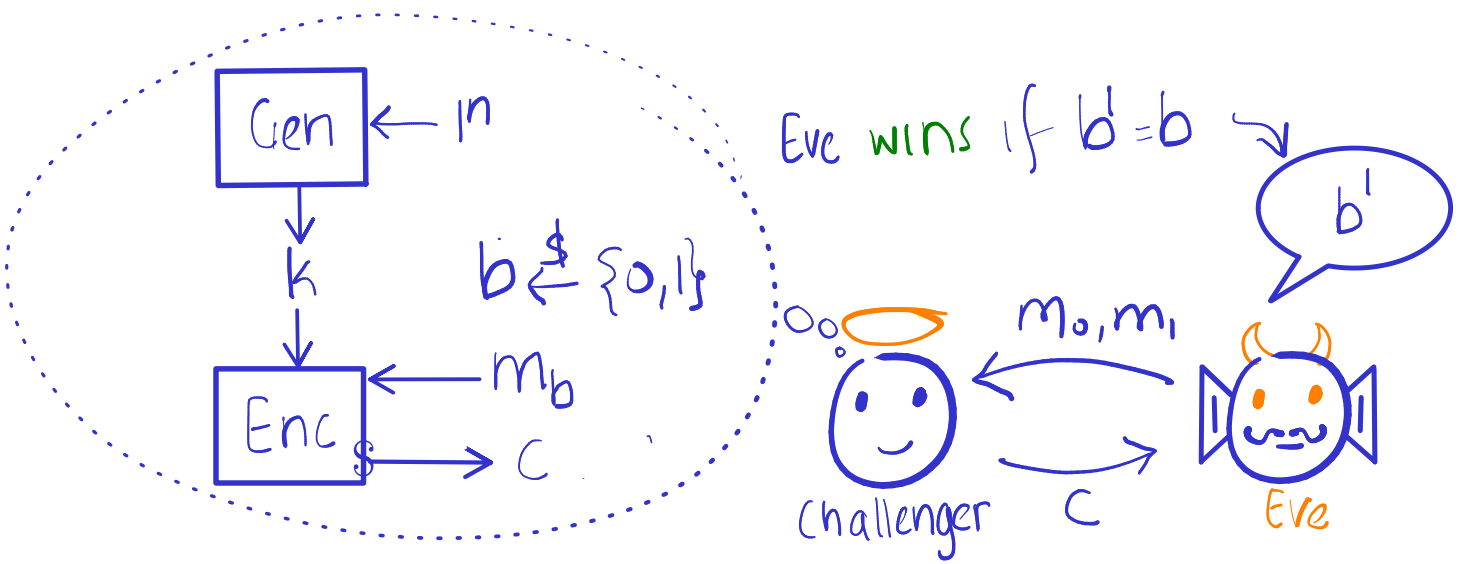
Let's Finally Define Computational Secrecy...

A Second Way



Let's Finally Define Computational Secrecy...

A Second Way



Let's Finally Define Computational Secrecy...

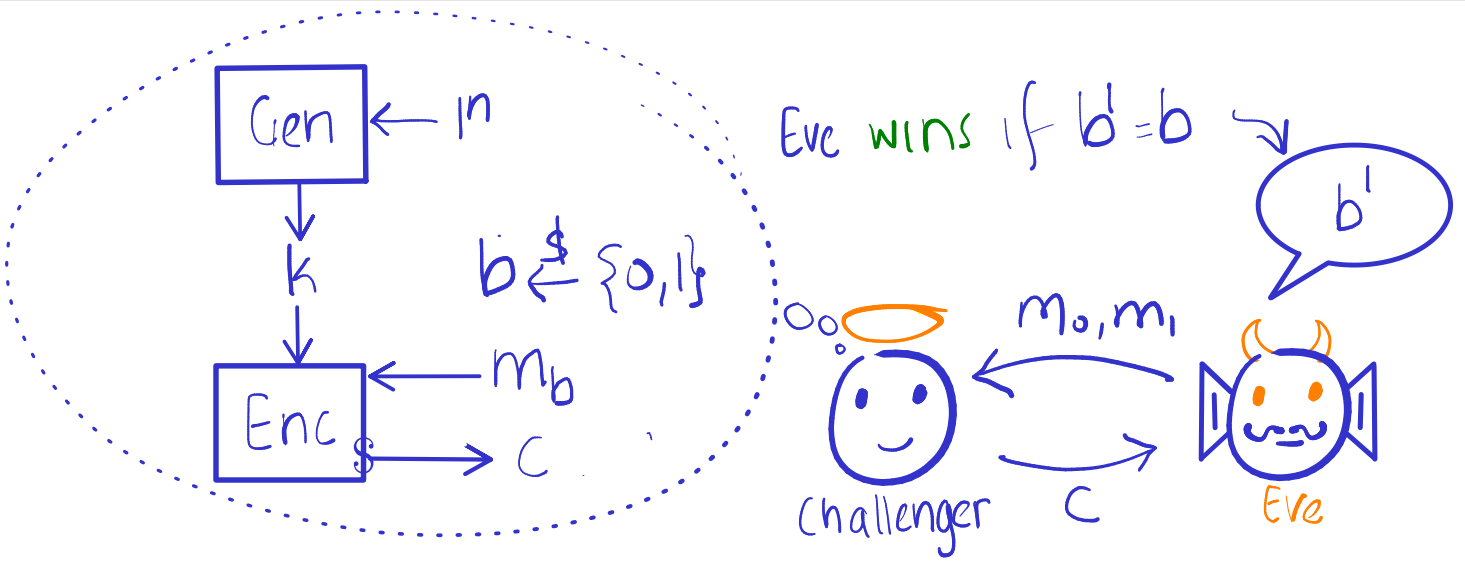
A Second Way

Defintion 4 (Adversarial Indistinguishability for PPT Eves)

An SKE $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is computationally-secret if for every PPT eavesdropper Eve*

$$\delta(n) := \Pr_{\substack{(m_0,m_1)\leftarrow \mathsf{Eve}(1^n) \\ k\leftarrow \mathsf{Gen}(1^n) \\ b\leftarrow\{0,1\} \\ c\leftarrow \mathsf{Enc}(k,m_b)}}[\mathsf{Eve}(c) = b] - \frac{1}{2}$$

is negligible.



The Two Definitions are Equivalent!

Claim 1 (Other direction exercise!)

Defintion 4 implies Defintion 3.

Proof.

Definition 4 implies for every PPT Eve the following is negligible

$$\Pr\left[\begin{array}{l} m_0, m_1 \leftarrow \mathrm{Eve}(1^n) \\ k \leftarrow \mathrm{Gen}(1^n),\ b \leftarrow \{0,1\} \\ c \leftarrow \mathrm{Enc}(k, m_b) \end{array} : \mathrm{Eve}(c) = b \right] - \frac{1}{2}$$

The Two Definitions are Equivalent!

Claim 1 (Other direction exercise!)

Defintion 4 implies Defintion 3.

Proof.

Definition 4 implies for every PPT Eve the following is negligible

$$\Pr\left[\mathsf{Eve}(c)=0, b=0 \text{ or } \mathsf{Eve}(c)=1, b=1\right] - \frac{1}{2}$$

$m_0, m_1 \leftarrow \mathsf{Eve}(1^n)$
$k \leftarrow \mathsf{Gen}(1^n), b \leftarrow \{0,1\}$
$c \leftarrow \mathsf{Enc}(k, m_b)$

The Two Definitions are Equivalent!

Claim 1 (Other direction exercise!)

Defintion 4 implies Defintion 3.

Proof.

Definition 4 implies for every PPT Eve the following is negligible

$$\Pr\left[\mathsf{Eve}(c)=0, b=0\right] + \Pr\left[\mathsf{Eve}(c)=1, b=1\right] - \frac{1}{2}$$

$m_0, m_1 \leftarrow \mathsf{Eve}(1^n)$
$k \leftarrow \mathsf{Gen}(1^n),\ b \leftarrow \{0,1\}$
$c \leftarrow \mathsf{Enc}(k, m_b)$

$m_0, m_1 \leftarrow \mathsf{Eve}(1^n)$
$k \leftarrow \mathsf{Gen}(1^n),\ b \leftarrow \{0,1\}$
$c \leftarrow \mathsf{Enc}(k, m_b)$

The Two Definitions are Equivalent!

Claim 1 (Other direction exercise!)

Defintion 4 implies Defintion 3.

Proof.

Definition 4 implies for every PPT Eve the following is negligible

$$\Pr\left[\begin{array}{l}m_0,m_1\leftarrow \mathrm{Eve}(1^n)\\ k\leftarrow \mathrm{Gen}(1^n),\ b\leftarrow\{0,1\}\\ c\leftarrow \mathrm{Enc}(k,m_b)\end{array} : \mathrm{Eve}(c)=0, b=0\right] + \Pr\left[\begin{array}{l}m_0,m_1\leftarrow \mathrm{Eve}(1^n)\\ k\leftarrow \mathrm{Gen}(1^n),\ b\leftarrow\{0,1\}\\ c\leftarrow \mathrm{Enc}(k,m_b)\end{array} : \mathrm{Eve}(c)=1, b=1\right] - \tfrac{1}{2}$$

$\Downarrow$

$$\frac{1}{2}\left[\Pr\left[\begin{array}{l}m_0,m_1\leftarrow \mathrm{Eve}(1^n)\\ k\leftarrow \mathrm{Gen}(1^n)\\ c\leftarrow \mathrm{Enc}(k,m_0)\end{array} : \mathrm{Eve}(c)=0\right] + \Pr\left[\begin{array}{l}m_0,m_1\leftarrow \mathrm{Eve}(1^n)\\ k\leftarrow \mathrm{Gen}(1^n)\\ c\leftarrow \mathrm{Enc}(k,m_1)\end{array} : \mathrm{Eve}(c)=1\right]\right] - \tfrac{1}{2} \text{ is negl.}$$

The Two Definitions are Equivalent!

Claim 1 (Other direction exercise!)

Defintion 4 implies Defintion 3.

Proof.

Definition 4 implies for every PPT Eve the following is negligible

$$\Pr\left[\begin{array}{l} m_0, m_1 \leftarrow \mathrm{Eve}(1^n) \\ k \leftarrow \mathrm{Gen}(1^n),\ b \leftarrow \{0,1\} \\ c \leftarrow \mathrm{Enc}(k, m_b) \end{array} : \mathrm{Eve}(c) = b\right] - \frac{1}{2}$$

$$\Pr\left[\begin{array}{l} m_0, m_1 \leftarrow \mathrm{Eve}(1^n) \\ k \leftarrow \mathrm{Gen}(1^n),\ b \leftarrow \{0,1\} \\ c \leftarrow \mathrm{Enc}(k, m_b) \end{array} : \mathrm{Eve}(c) = b\right] - \frac{1}{2}$$

$$\Downarrow$$

$$\frac{1}{2}\left|\Pr\left[\begin{array}{l} m_0, m_1 \leftarrow \mathrm{Eve}(1^n) \\ k \leftarrow \mathrm{Gen}(1^n) \\ c \leftarrow \mathrm{Enc}(k, m_0) \end{array} : \mathrm{Eve}(c) = 0\right] + 1 - \Pr\left[\begin{array}{l} m_0, m_1 \leftarrow \mathrm{Eve}(1^n) \\ k \leftarrow \mathrm{Gen}(1^n) \\ c \leftarrow \mathrm{Enc}(k, m_1) \end{array} : \mathrm{Eve}(c) = 0\right]\right| - \frac{1}{2} \text{ is negl.}$$

The Two Definitions are Equivalent!

Claim 1 (Other direction exercise!)

Defintion 4 implies Defintion 3.

Proof.

Definition 4 implies for every PPT Eve the following is negligible

$$\Pr\left[\mathrm{Eve}(c)=0, b=0\right] + \Pr\left[\mathrm{Eve}(c)=1, b=1\right] - \tfrac{1}{2}$$

$m_0, m_1 \leftarrow \mathrm{Eve}(1^n)$
$k \leftarrow \mathrm{Gen}(1^n),\ b \leftarrow \{0,1\}$
$c \leftarrow \mathrm{Enc}(k, m_b)$

$m_0, m_1 \leftarrow \mathrm{Eve}(1^n)$
$k \leftarrow \mathrm{Gen}(1^n),\ b \leftarrow \{0,1\}$
$c \leftarrow \mathrm{Enc}(k, m_b)$

$\Downarrow$

$$\left| \Pr\left[\mathrm{Eve}(c)=0\right] - \Pr\left[\mathrm{Eve}(c)=0\right] \right| \text{ is negl.}$$

$m_0, m_1 \leftarrow \mathrm{Eve}(1^n)$
$k \leftarrow \mathrm{Gen}(1^n)$
$c \leftarrow \mathrm{Enc}(k, m_0)$

$m_0, m_1 \leftarrow \mathrm{Eve}(1^n)$
$k \leftarrow \mathrm{Gen}(1^n)$
$c \leftarrow \mathrm{Enc}(k, m_1)$

The Two Definitions are Equivalent!

Claim 1 (Other direction exercise!)

Defintion 4 implies Defintion 3.

Proof.

Definition 4 implies for every PPT Eve the following is negligible

$$\Pr\left[\mathrm{Eve}(C)=0, b=0\right] + \Pr\left[\mathrm{Eve}(C)=1, b=1\right] - \tfrac{1}{2}$$

$m_0, m_1 \leftarrow \mathrm{Eve}(1^n)$
$K \leftarrow \mathrm{Gen}(1^n),\ b \leftarrow \{0,1\}$
$C \leftarrow \mathrm{Enc}(K, m_b)$

$m_0, m_1 \leftarrow \mathrm{Eve}(1^n)$
$K \leftarrow \mathrm{Gen}(1^n),\ b \leftarrow \{0,1\}$
$C \leftarrow \mathrm{Enc}(K, m_b)$

$\Downarrow$

$$\left| \Pr\left[\mathrm{Eve}(C)=0\right] - \Pr\left[\mathrm{Eve}(C)=0\right] \right| \text{ is negl.}$$

$m_0, m_1 \leftarrow \mathrm{Eve}(1^n)$
$K \leftarrow \mathrm{Gen}(1^n)$
$C \leftarrow \mathrm{Enc}(K, m_0)$

$m_0, m_1 \leftarrow \mathrm{Eve}(1^n)$
$K \leftarrow \mathrm{Gen}(1^n)$
$C \leftarrow \mathrm{Enc}(K, m_1)$

? Where did I cheat?

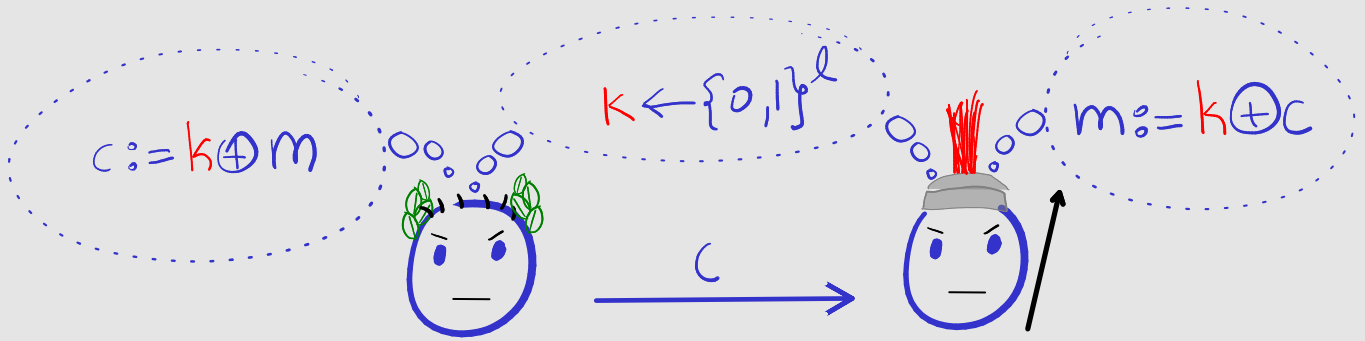
Plan for This Lecture

1. Limitations of Perfect Secrecy: Shannon's Impossibility
2. Bypassing Shannon's Impossibility
3. Pseudo-Random Generators (PRGs) and Computational OTP

↖ Our first cryptographic assumption!

Recall One-Time Pad (Vernam's Cipher)

Construction 1 (Message space $\{0,1\}^\ell$)

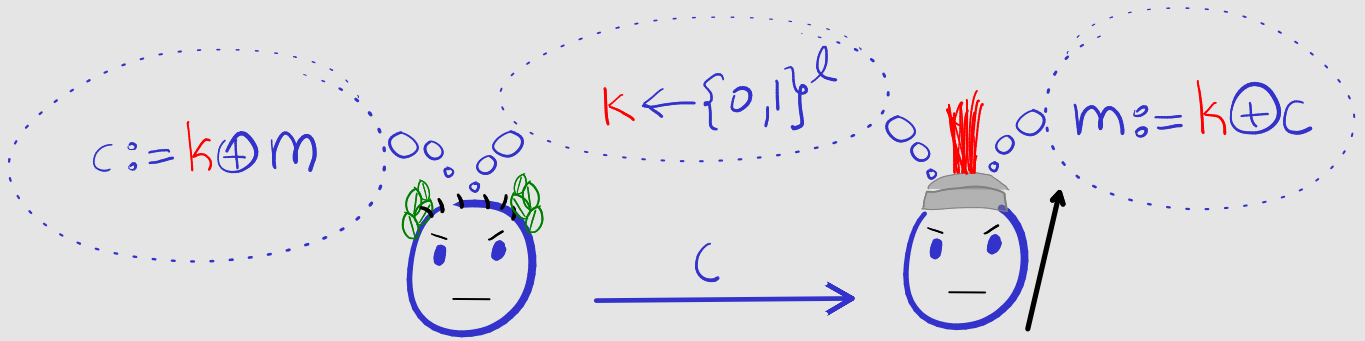


Pseudocode 1 (Message space $\{0,1\}^\ell$)

- *Key generation* $\mathsf{Gen}(1^\ell)$*: output* $k \leftarrow \{0,1\}^\ell$
- *Encryption* $\mathsf{Enc}(k, m)$*: output* $c := k \oplus m$
- *Decryption* $\mathsf{Dec}(k, c)$*: output* $m := k \oplus c$

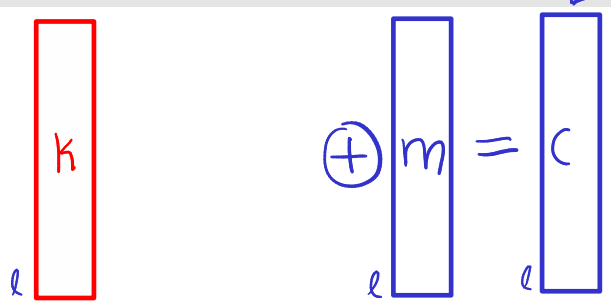
Recall One-Time Pad (Vernam's Cipher)

Construction 1 (Message space $\{0,1\}^\ell$)



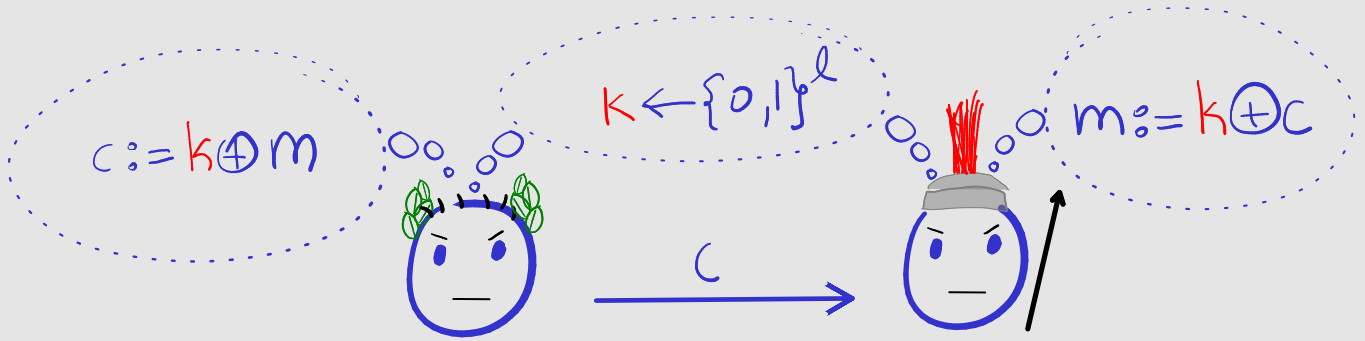
Pseudocode 1 (Message space $\{0,1\}^\ell$)

- *Key generation* $\mathsf{Gen}(1^\ell)$: *output* $k \leftarrow \{0,1\}^\ell$
- *Encryption* $\mathsf{Enc}(k, m)$: *output* $c := k \oplus m$
- *Decryption* $\mathsf{Dec}(k, c)$: *output* $m := k \oplus c$



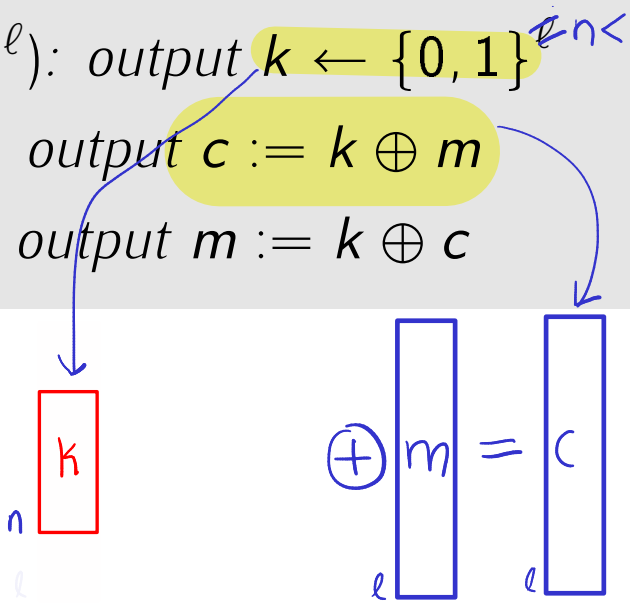
Recall One-Time Pad (Vernam's Cipher)

Construction 1 (Message space $\{0,1\}^\ell$)



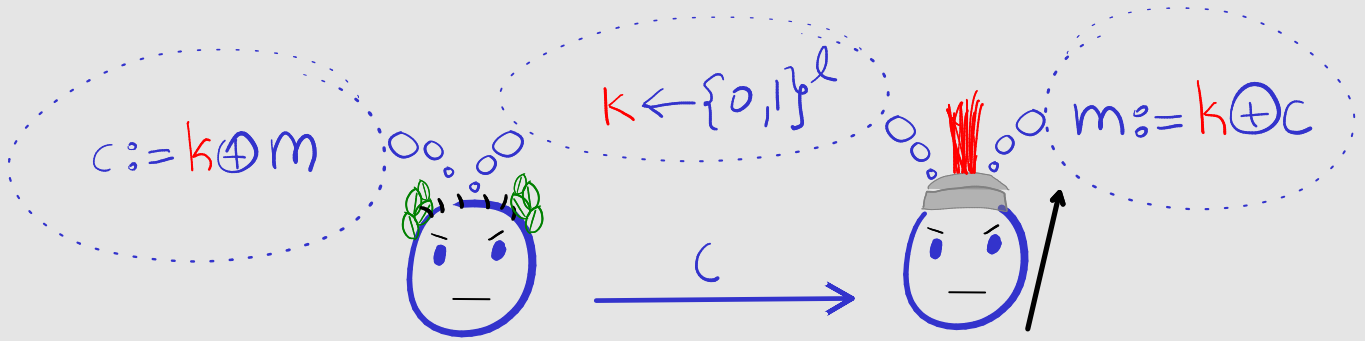
Pseudocode 1 (Message space $\{0,1\}^\ell$)

- *Key generation* $\mathsf{Gen}(1^\ell)$: *output* $k \leftarrow \{0,1\}^n$ $n<\ell$
- *Encryption* $\mathsf{Enc}(k, m)$: *output* $c := k \oplus m$
- *Decryption* $\mathsf{Dec}(k, c)$: *output* $m := k \oplus c$



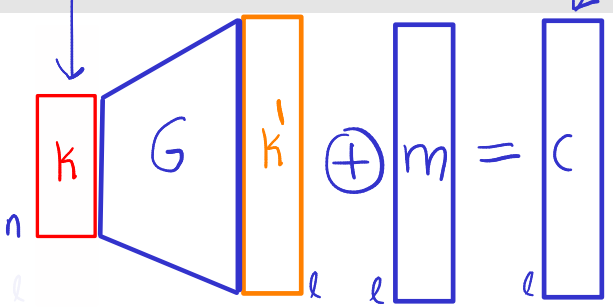
Recall One-Time Pad (Vernam's Cipher)

Construction 1 (Message space $\{0,1\}^\ell$)



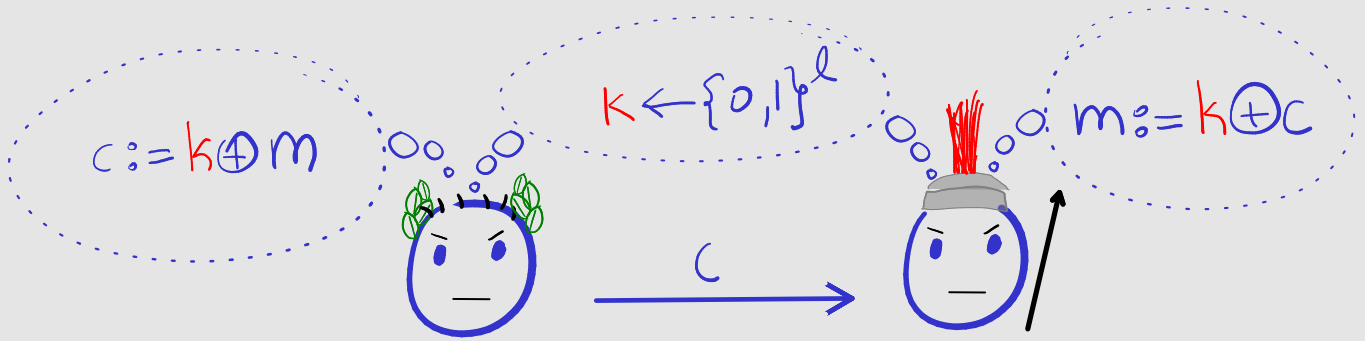
Pseudocode 1 (Message space $\{0,1\}^\ell$)

- *Key generation* $\mathsf{Gen}(1^\ell)$: *output* $k \leftarrow \{0,1\}^n$ $n<\ell$
- *Encryption* $\mathsf{Enc}(k, m)$: *output* $c := k' \oplus m$
- *Decryption* $\mathsf{Dec}(k, c)$: *output* $m := k \oplus c$



Recall One-Time Pad (Vernam's Cipher)

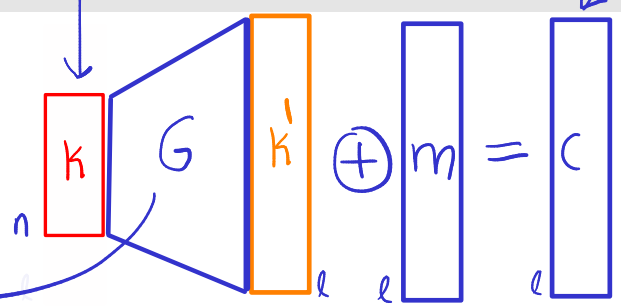
Construction 1 (Message space $\{0,1\}^\ell$)



Pseudocode 1 (Message space $\{0,1\}^\ell$)

- *Key generation* $\mathsf{Gen}(1^\ell)$: *output* $k \leftarrow \{0,1\}^n$ $n<\ell$
- *Encryption* $\mathsf{Enc}(k,m)$: *output* $c := k' \oplus m$
- *Decryption* $\mathsf{Dec}(k,c)$: *output* $m := k \oplus c$

Requirement:
G expands $k \to k'$ such that k' "seems random" to

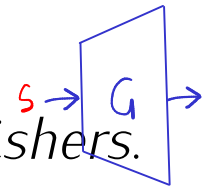


Pseudo-Random Generator (PRG)...

- Intuitive definition: expanding function whose output (on uniformly random input) "seems random" to PPT *distinguishers*.

Pseudo-Random Generator (PRG)...

- Intuitive definition: expanding function whose output (on uniformly random input) "seems random" to PPT *distinguishers*.

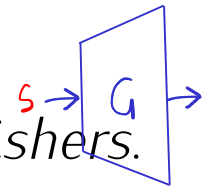


Defintion 5 (PRG, via Imitation Game)

Let G *be an efficient deterministic algorithm that for any* $n \in \mathbb{N}$ *and input* $s \in \{0,1\}^n$*, outputs a string of length* $\ell(n) > n$*.*

Pseudo-Random Generator (PRG)...

- Intuitive definition: expanding function whose output (on uniformly random input) "seems random" to PPT *distinguishers*.



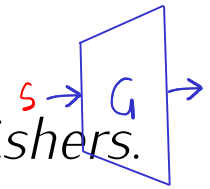
Defintion 5 (PRG, via Imitation Game)

Let G *be an efficient deterministic algorithm that for any* $n \in \mathbb{N}$ *and input* $s \in \{0,1\}^n$*, outputs a string of length* $\ell(n) > n$.

"expansion factor or stretch"

Pseudo-Random Generator (PRG)...

- Intuitive definition: expanding function whose output (on uniformly random input) "seems random" to PPT *distinguishers*.



Defintion 5 (PRG, via Imitation Game)

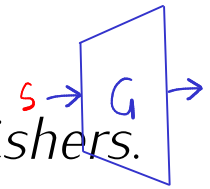
Let G *be an efficient deterministic algorithm that for any* $n \in \mathbb{N}$ *and input* $s \in \{0,1\}^n$*, outputs a string of length* $\ell(n) > n$*.*
G *is PRG if for every PPT distinguisher* D

$$\delta(n) := \left| \Pr_{s \leftarrow \{0,1\}^n}[\mathsf{D}(G(s)) = 0] - \Pr_{r \leftarrow \{0,1\}^{\ell(n)}}[\mathsf{D}(r) = 0] \right|$$

is negligible.

Pseudo-Random Generator (PRG)...

- Intuitive definition: expanding function whose output (on uniformly random input) "seems random" to PPT *distinguishers*.



Defintion 5 (PRG, via Imitation Game)

Let G *be an efficient deterministic algorithm that for any* $n \in \mathbb{N}$ *and input* $s \in \{0,1\}^n$*, outputs a string of length* $\ell(n) > n$*.*
G *is PRG if for every PPT distinguisher* D

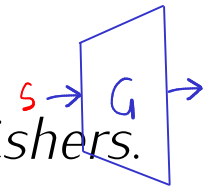
$$\delta(n) := \left| \Pr_{s \leftarrow \{0,1\}^n}[D(G(s)) = 0] - \Pr_{r \leftarrow \{0,1\}^{\ell(n)}}[D(r) = 0] \right|$$

pseudorandom world random world

is negligible.

Pseudo-Random Generator (PRG)

- Intuitive definition: expanding function whose output (on uniformly random input) "seems random" to PPT *distinguishers*.



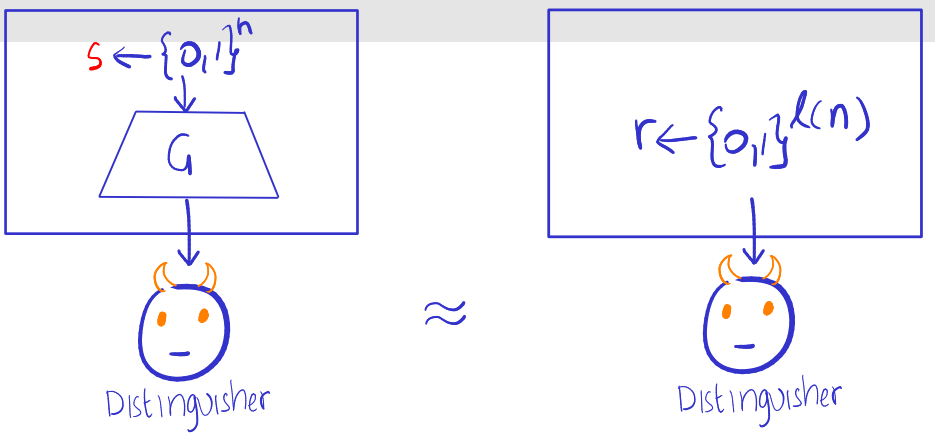
Defintion 5 (PRG, via Imitation Game)

Let G *be an efficient deterministic algorithm that for any* $n \in \mathbb{N}$ *and input* $s \in \{0,1\}^n$*, outputs a string of length* $\ell(n) > n$*.*
G *is PRG if for every PPT distinguisher* D

$$\delta(n) := \left| \Pr_{s \leftarrow \{0,1\}^n}[D(G(s)) = 0] - \Pr_{r \leftarrow \{0,1\}^{\ell(n)}}[D(r) = 0] \right|$$

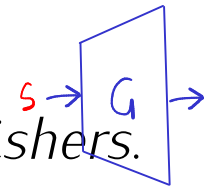
pseudorandom world random world

is negligible.



Pseudo-Random Generator (PRG)...

- Intuitive definition: expanding function whose output (on uniformly random input) "seems random" to PPT *distinguishers*.



Defintion 5 (PRG, via Imitation Game)

Let G *be an efficient deterministic algorithm that for any* $n \in \mathbb{N}$ *and input* $s \in \{0,1\}^n$*, outputs a string of length* $\ell(n) > n$*.*
G *is PRG if for every PPT distinguisher* D

$$\delta(n) := \left| \Pr_{s \leftarrow \{0,1\}^n}[D(G(s)) = 0] - \Pr_{r \leftarrow \{0,1\}^{\ell(n)}}[D(r) = 0] \right|$$

pseudorandom world

random world

is negligible.

Exercise 2

1. *Write up "adversarial indistinguishability" definition of PRG.*
2. *Show that the two definitions are equivalent.*

Pseudo-Random Generator...

Defintion 5 (PRG, via Imitation Game)

Let G *be an efficient deterministic algorithm that for any* $n \in \mathbb{N}$ *and input* $s \in \{0,1\}^n$*, outputs a string of length* $\ell(n) > n$*.*

Pseudo-Random Generator...

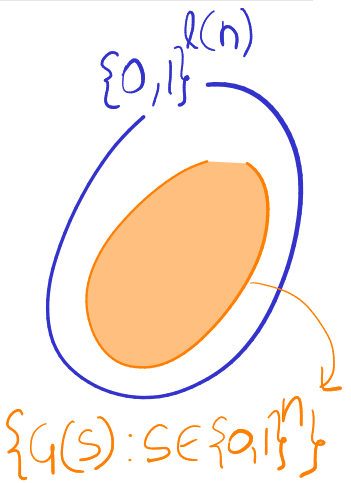
Defintion 5 (PRG, via Imitation Game)

Let G *be an efficient deterministic algorithm that for any* $n \in \mathbb{N}$ *and input* $s \in \{0,1\}^n$*, outputs a string of length* $\ell(n) > n$*.*
G *is PRG if for every PPT distinguisher* D

$$\delta(n) := \left| \Pr_{s \leftarrow \{0,1\}^n}[\mathsf{D}(G(s)) = 0] - \Pr_{r \leftarrow \{0,1\}^{\ell(n)}}[\mathsf{D}(r) = 0] \right|$$

is negligible.

- Let's check if you understood the notion of PRG
  - How can an *unbounded* distinguisher break PRG?
  - If $\mathsf{P} = \mathsf{NP}$ can PRGs exist?



Pseudo-Random Generator...

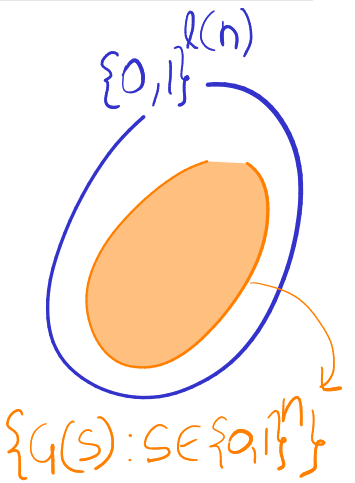
Defintion 5 (PRG, via Imitation Game)

Let G *be an efficient deterministic algorithm that for any* $n \in \mathbb{N}$ *and input* $s \in \{0,1\}^n$*, outputs a string of length* $\ell(n) > n$*.*
G *is PRG if for every PPT distinguisher* D

$$\delta(n) := \left| \Pr_{s \leftarrow \{0,1\}^n}[D(G(s)) = 0] - \Pr_{r \leftarrow \{0,1\}^{\ell(n)}}[D(r) = 0] \right|$$

is negligible.

- Let's check if you understood the notion of PRG
  - How can an *unbounded* distinguisher break PRG?
  - If $\mathsf{P} = \mathsf{NP}$ can PRGs exist?
  - Is G a PRG or not? Below G_1 and G_2 are PRGs
    1. $G(s) := G_1(s)0$
    2. $G(s_1 s_2) := G_1(s_1)G_2(s_2)$



Pseudo-Random Generator...

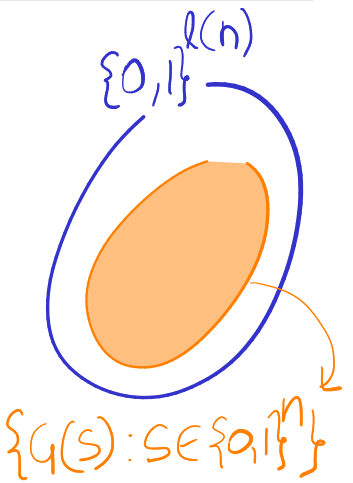
Defintion 5 (PRG, via Imitation Game)

Let G *be an efficient deterministic algorithm that for any* $n \in \mathbb{N}$ *and input* $s \in \{0,1\}^n$*, outputs a string of length* $\ell(n) > n$*.*
G *is PRG if for every PPT distinguisher* D

$$\delta(n) := \left| \Pr_{s \leftarrow \{0,1\}^n}[D(G(s)) = 0] - \Pr_{r \leftarrow \{0,1\}^{\ell(n)}}[D(r) = 0] \right|$$

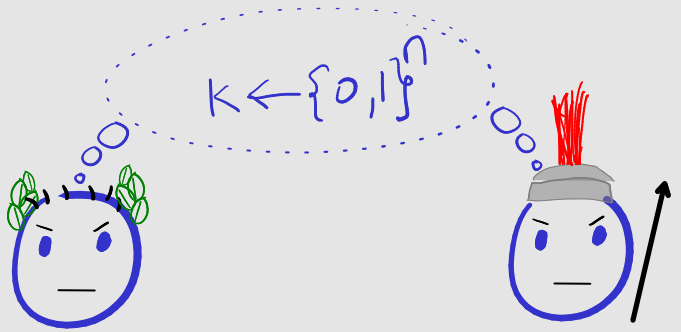
is negligible.

- Let's check if you understood the notion of PRG
  - How can an *unbounded* distinguisher break PRG?
  - If $\mathsf{P} = \mathsf{NP}$ can PRGs exist?
  - Is G a PRG or not? Below G_1 and G_2 are PRGs
    1. $G(s) := G_1(s)0$
    2. $G(s_1 s_2) := G_1(s_1)G_2(s_2)$
    3. $G(s) := G_1(s)G_2(s)$
    4. $G(s) := G_1(s) \oplus G_2(s)$



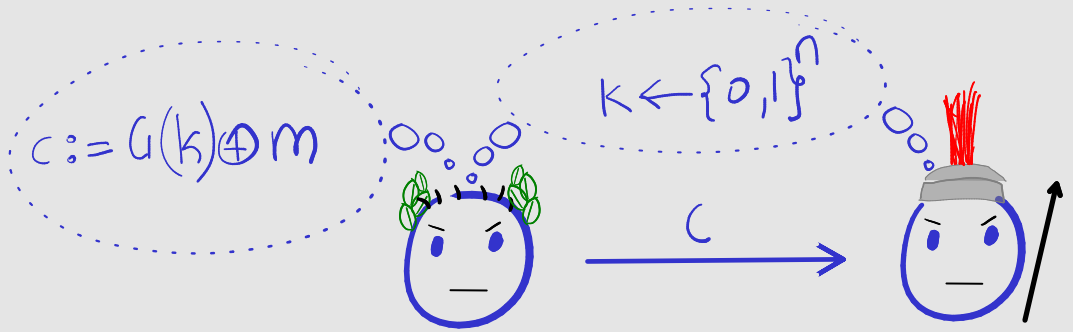
"Computational" One-Time Pad from PRG G

Construction 2 (Message space $\{0,1\}^{\ell(n)}$)



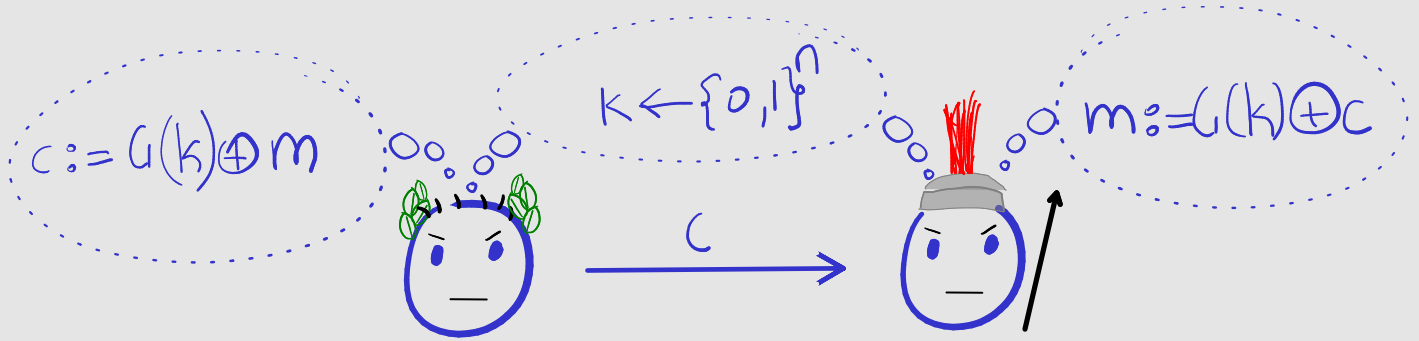
“Computational” One-Time Pad from PRG G

Construction 2 (Message space $\{0,1\}^{\ell(n)}$)



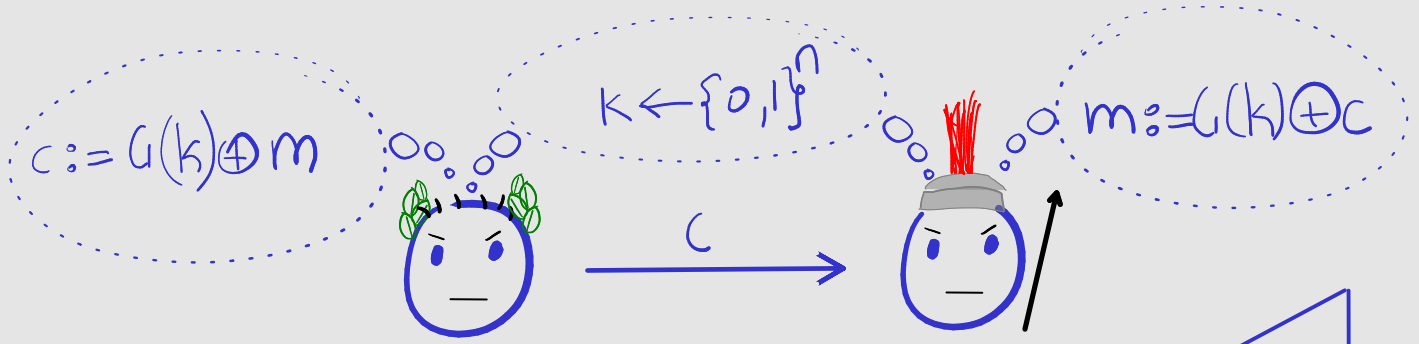
“Computational” One-Time Pad from PRG G

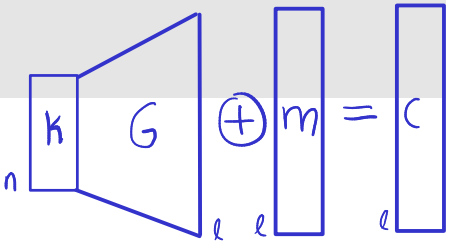
Construction 2 (Message space $\{0,1\}^{\ell(n)}$)



“Computational” One-Time Pad from PRG G

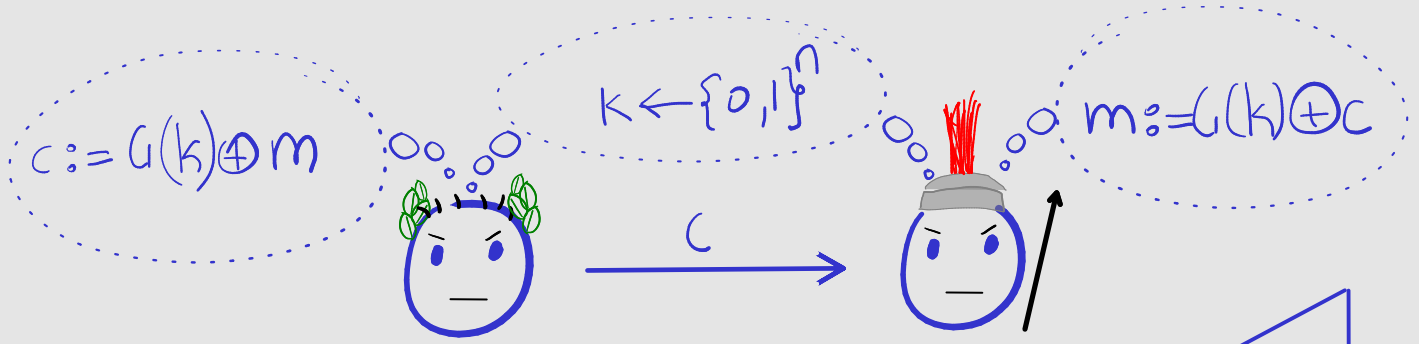
Construction 2 (Message space $\{0,1\}^{\ell(n)}$)

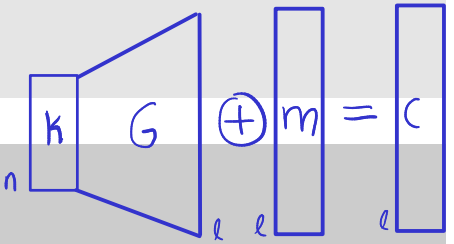




"Computational" One-Time Pad from PRG G

Construction 2 (Message space $\{0,1\}^{\ell(n)}$)



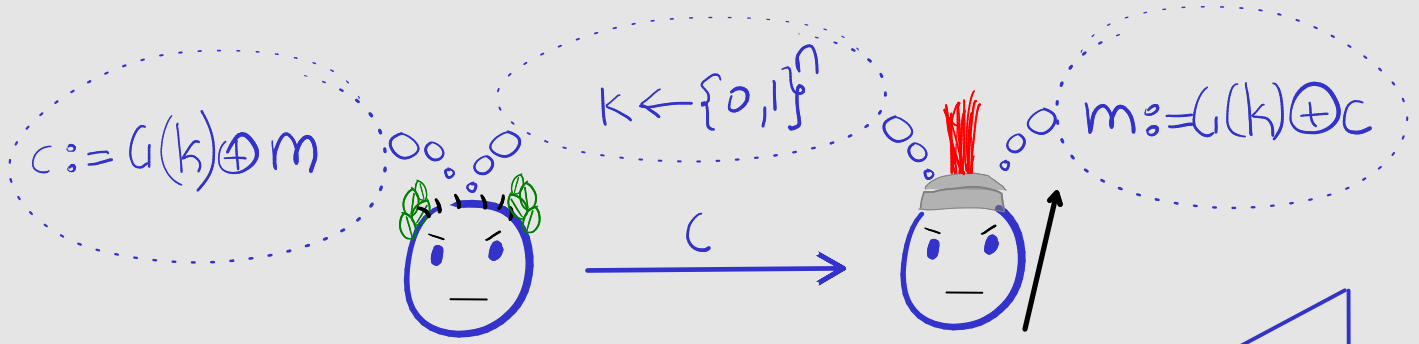


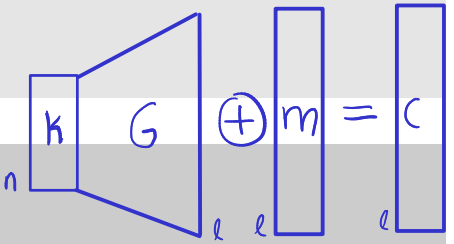
Pseudocode 2 (Message space $\{0,1\}^{\ell(n)}$)

- *Key generation* $\mathsf{Gen}(1^n)$*: output* $k \leftarrow \{0,1\}^n$
- *Encryption* $\mathsf{Enc}(k, m)$*: output* $c := G(k) \oplus m$
- *Decryption* $\mathsf{Dec}(k, c)$*: output* $m := G(k) \oplus c$

"Computational" One-Time Pad from PRG G

Construction 2 (Message space $\{0,1\}^{\ell(n)}$)





Pseudocode 2 (Message space $\{0,1\}^{\ell(n)}$)

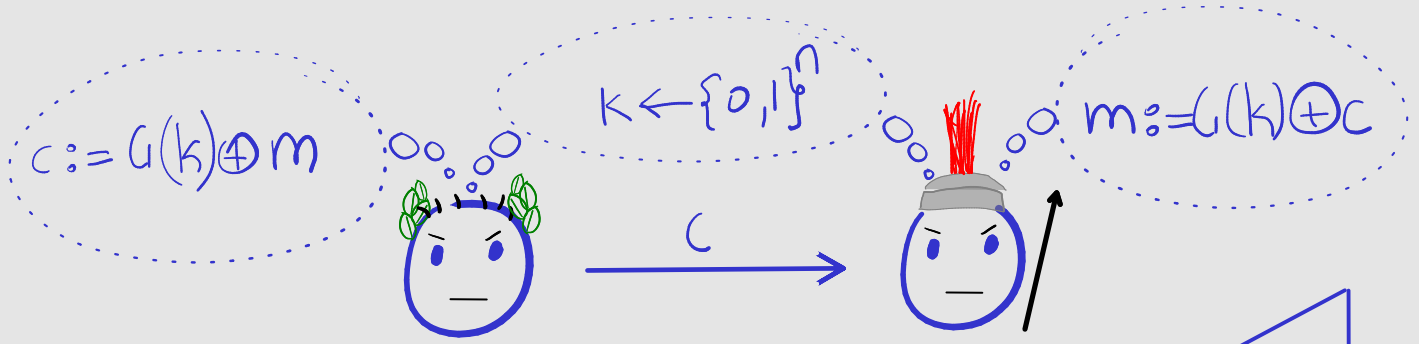
- *Key generation* $\mathsf{Gen}(1^n)$*: output* $k \leftarrow \{0,1\}^n$
- *Encryption* $\mathsf{Enc}(k, m)$*: output* $c := G(k) \oplus m$
- *Decryption* $\mathsf{Dec}(k, c)$*: output* $m := G(k) \oplus c$

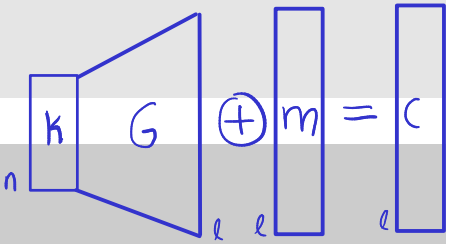
- Correctness of decryption: for every $n \in \mathbb{N}$ and $m \in \{0,1\}^{\ell(n)}$,

$$\Pr_{k \leftarrow \{0,1\}^n}\left[G(k) \oplus \left(G(k) \oplus m\right) = m\right]$$

"Computational" One-Time Pad from PRG G

Construction 2 (Message space $\{0,1\}^{\ell(n)}$)





Pseudocode 2 (Message space $\{0,1\}^{\ell(n)}$)

- *Key generation* $\mathsf{Gen}(1^n)$*: output* $k \leftarrow \{0,1\}^n$
- *Encryption* $\mathsf{Enc}(k, m)$*: output* $c := G(k) \oplus m$
- *Decryption* $\mathsf{Dec}(k, c)$*: output* $m := G(k) \oplus c$

- Correctness of decryption: for every $n \in \mathbb{N}$ and $m \in \{0,1\}^{\ell(n)}$,

$$\Pr_{k \leftarrow \{0,1\}^n}\left[G(k) \oplus (G(k) \oplus m) = m\right] = \Pr_{k \leftarrow \{0,1\}^n}\left[m = m\right] = 1$$

Proof of Computational Secrecy...

Theorem 3

Assuming G *is a PRG, Construction 2 is computationally secret.*

Proof *by reduction.* $\exists$Eve breaking Construction 2 $\Rightarrow$ $\exists$D for G.

Proof of Computational Secrecy...

Theorem 3

Assuming G *is a PRG, Construction 2 is computationally secret.*

Proof *by reduction.* ∃Eve breaking Construction 2 ⇒ ∃D for G.

Intuition: consider the following four worlds

World 0

$$\Pr_{s \leftarrow \{0,1\}^n}[\mathrm{Eve}(G(s) \oplus m_0) = 0]$$

World 1

$$\Pr_{s \leftarrow \{0,1\}^n}[\mathrm{Eve}(G(s) \oplus m_1) = 0]$$

$$\Pr_{r \leftarrow \{0,1\}^{\ell(n)}}[\mathrm{Eve}(r \oplus m_0) = 0]$$

World 0'

$$\Pr_{r \leftarrow \{0,1\}^{\ell(n)}}[\mathrm{Eve}(r \oplus m_1) = 0]$$

World 1'

Proof of Computational Secrecy...

Theorem 3

Assuming G *is a PRG, Construction 2 is computationally secret.*

Proof *by reduction.* ∃Eve breaking Construction 2 ⇒ ∃D for G.

Intuition: consider the following four worlds

Target

World 0

$\Pr_{s \leftarrow \{0,1\}^n}[\text{Eve}(G(s) \oplus m_0) = 0]$

World 1

$\Pr_{s \leftarrow \{0,1\}^n}[\text{Eve}(G(s) \oplus m_1) = 0]$

$\Pr_{r \leftarrow \{0,1\}^{\ell(n)}}[\text{Eve}(r \oplus m_0) = 0]$

World 0'

$\Pr_{r \leftarrow \{0,1\}^{\ell(n)}}[\text{Eve}(r \oplus m_1) = 0]$

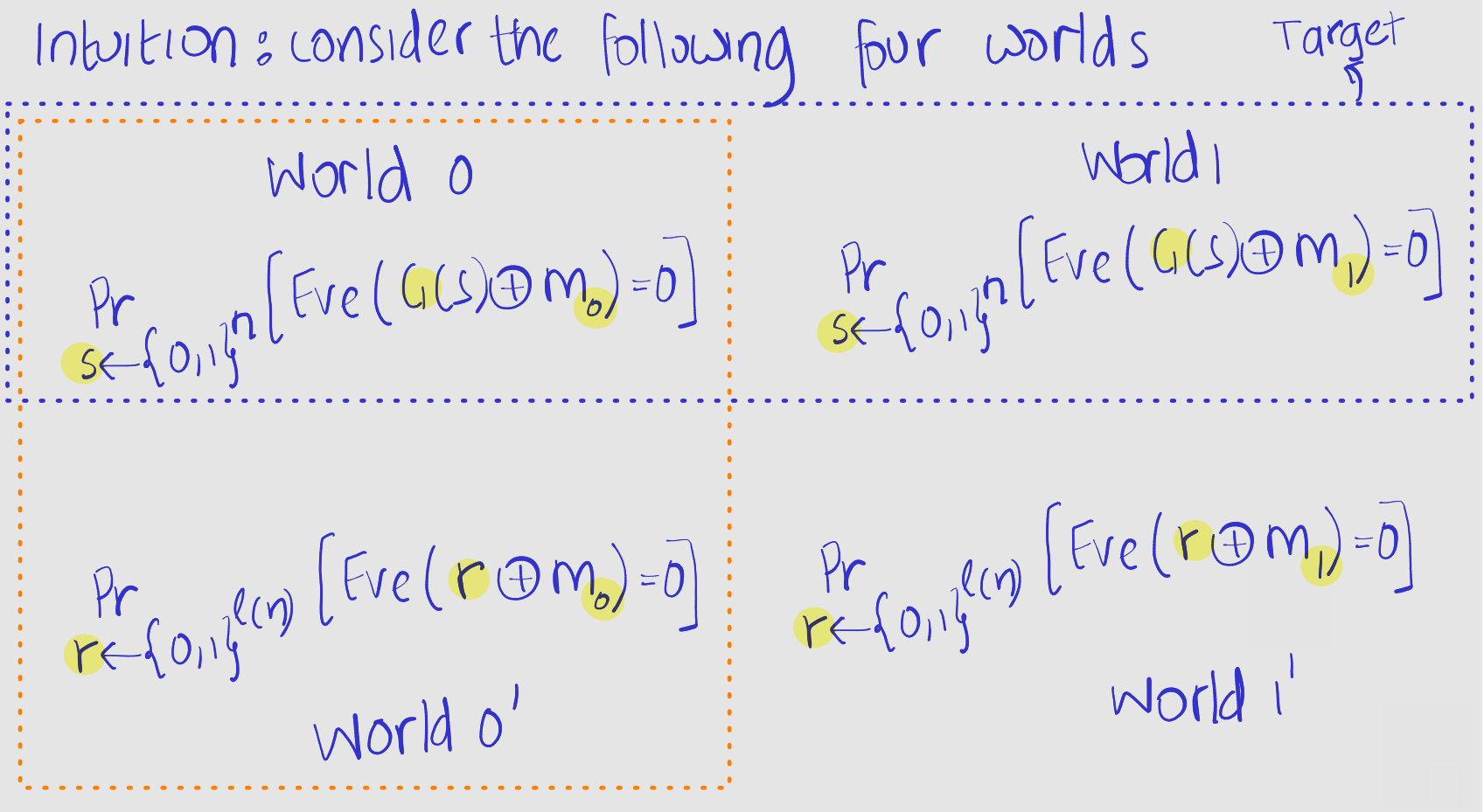
World 1'

Proof of Computational Secrecy...

Theorem 3

Assuming G *is a PRG, Construction 2 is computationally secret.*

Proof *by reduction*. ∃Eve breaking Construction 2 ⇒ ∃D for G.

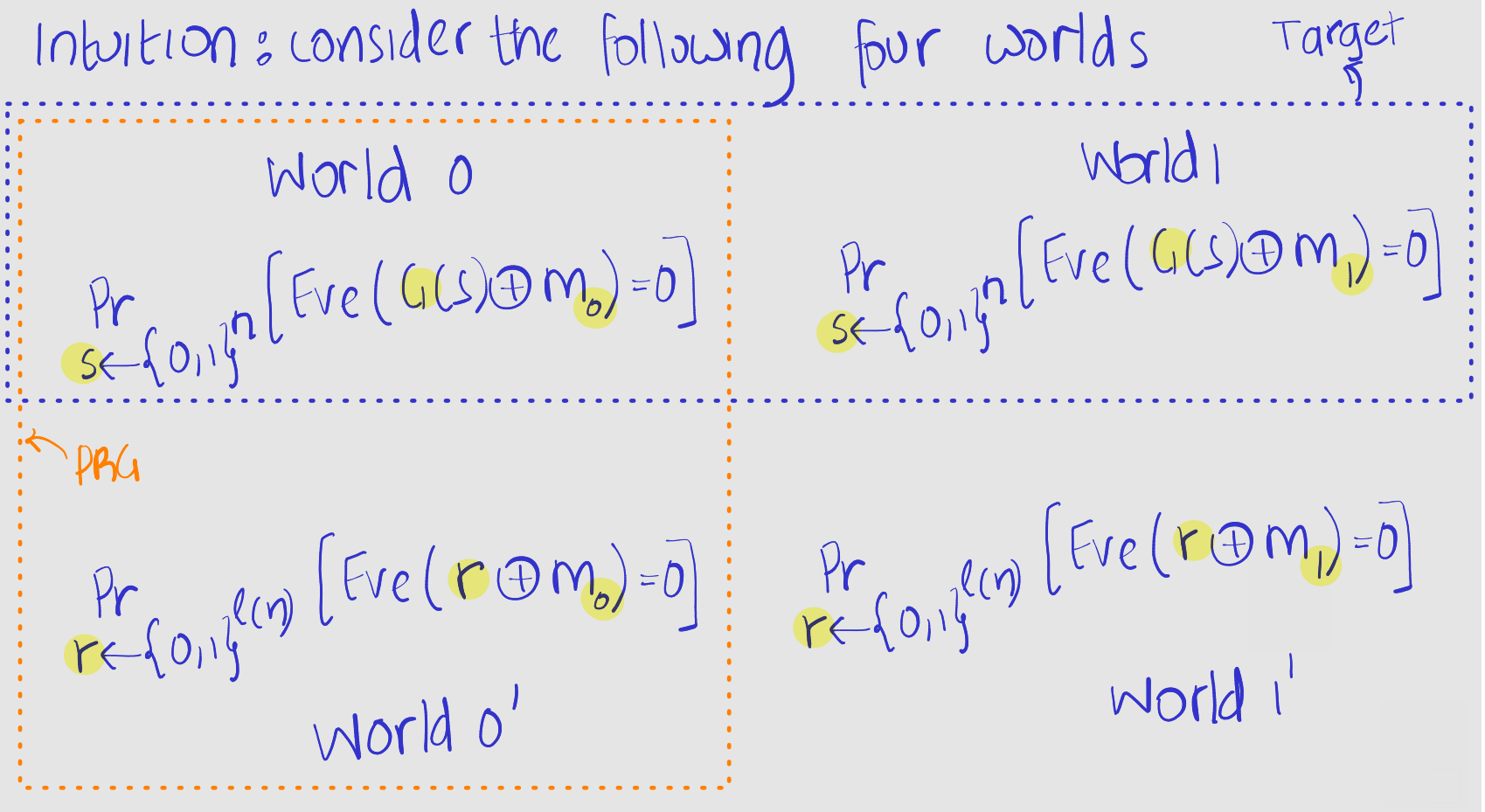


Proof of Computational Secrecy...

Theorem 3

Assuming G *is a PRG, Construction 2 is computationally secret.*

Proof *by reduction*. ∃Eve breaking Construction 2 ⇒ ∃D for G.

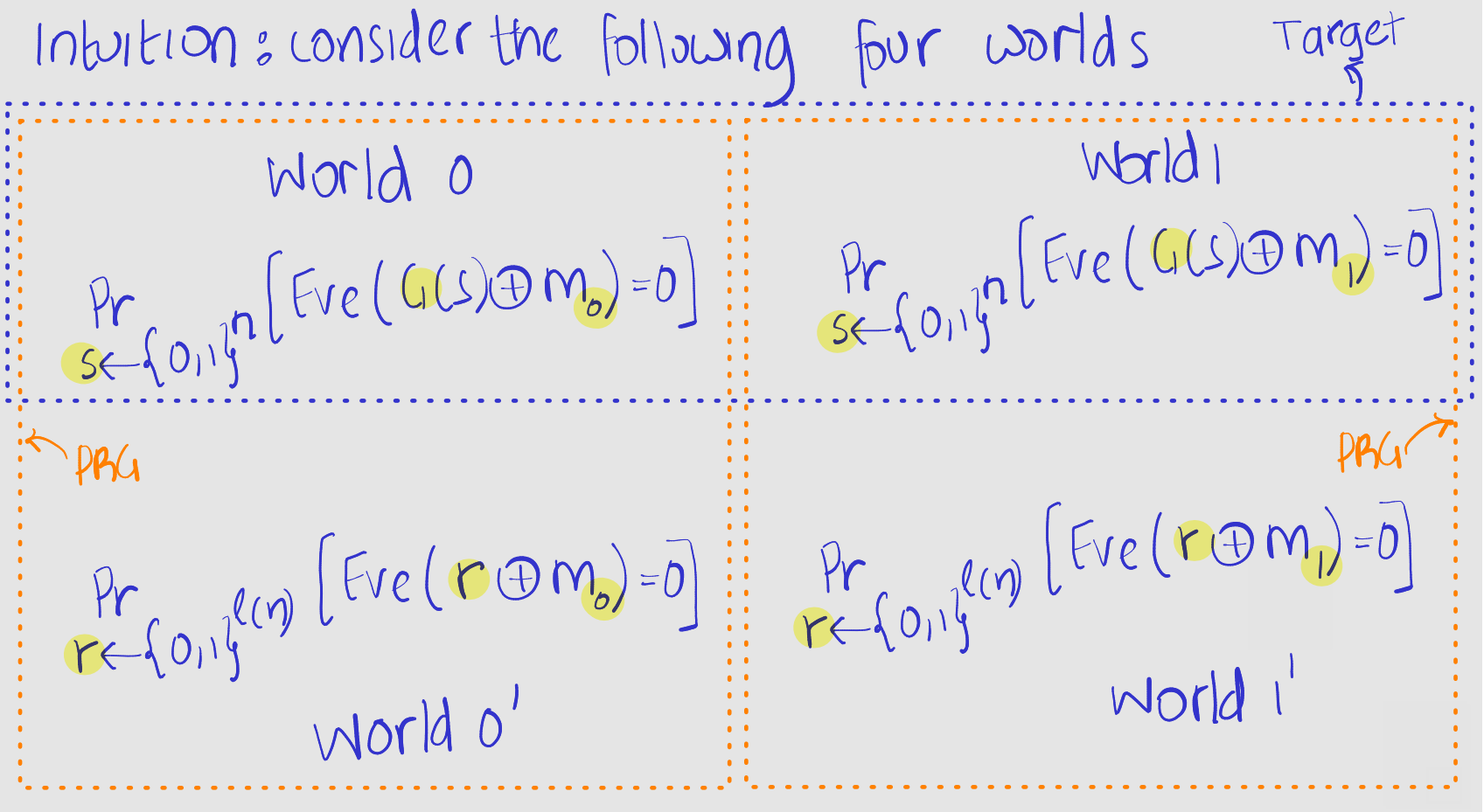


Proof of Computational Secrecy...

Theorem 3

Assuming G *is a PRG, Construction 2 is computationally secret.*

Proof *by reduction*. ∃Eve breaking Construction 2 ⇒ ∃D for G.

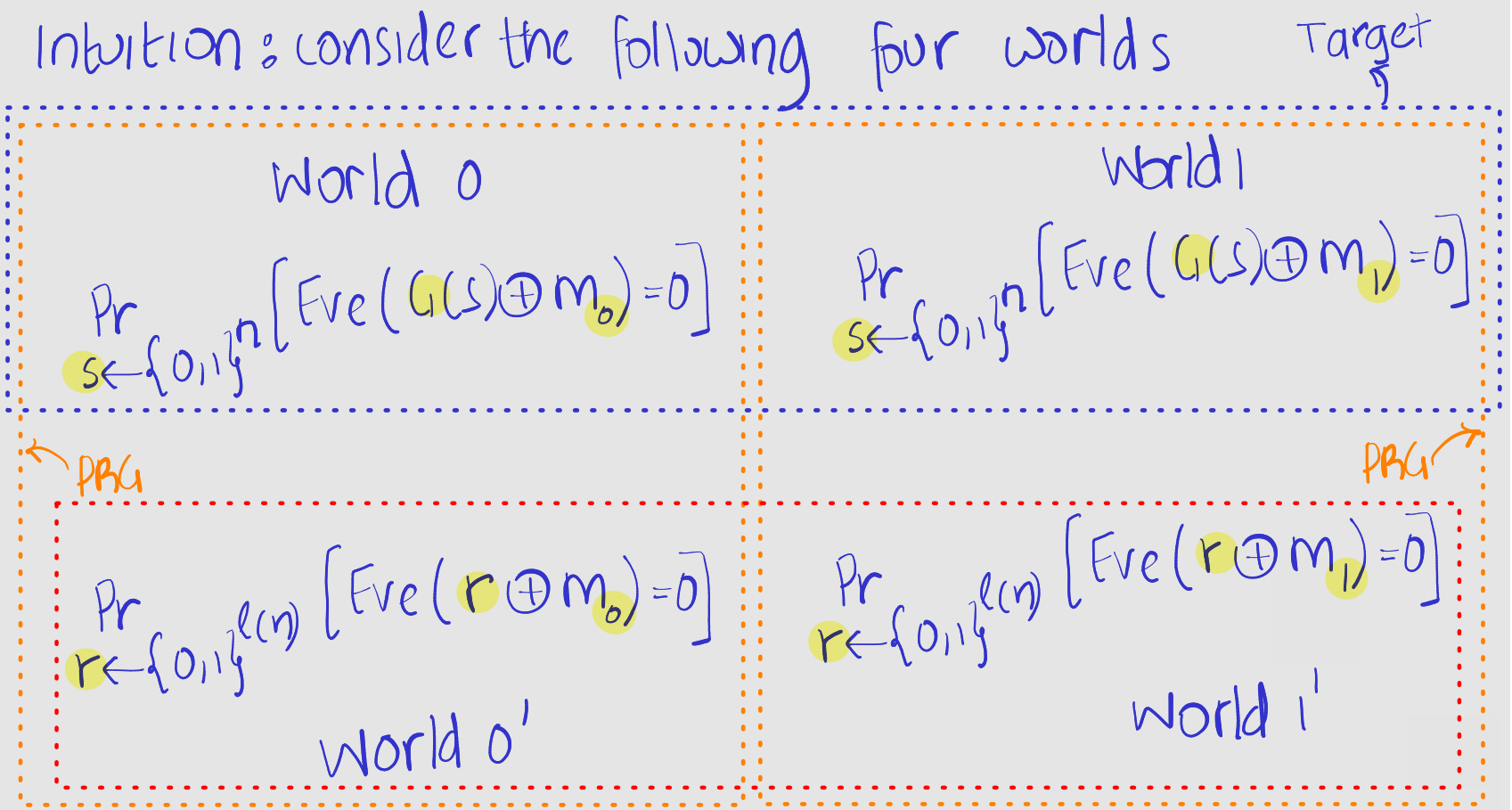


Proof of Computational Secrecy...

Theorem 3

Assuming G *is a PRG, Construction 2 is computationally secret.*

Proof *by reduction*. ∃Eve breaking Construction 2 ⇒ ∃D for G.

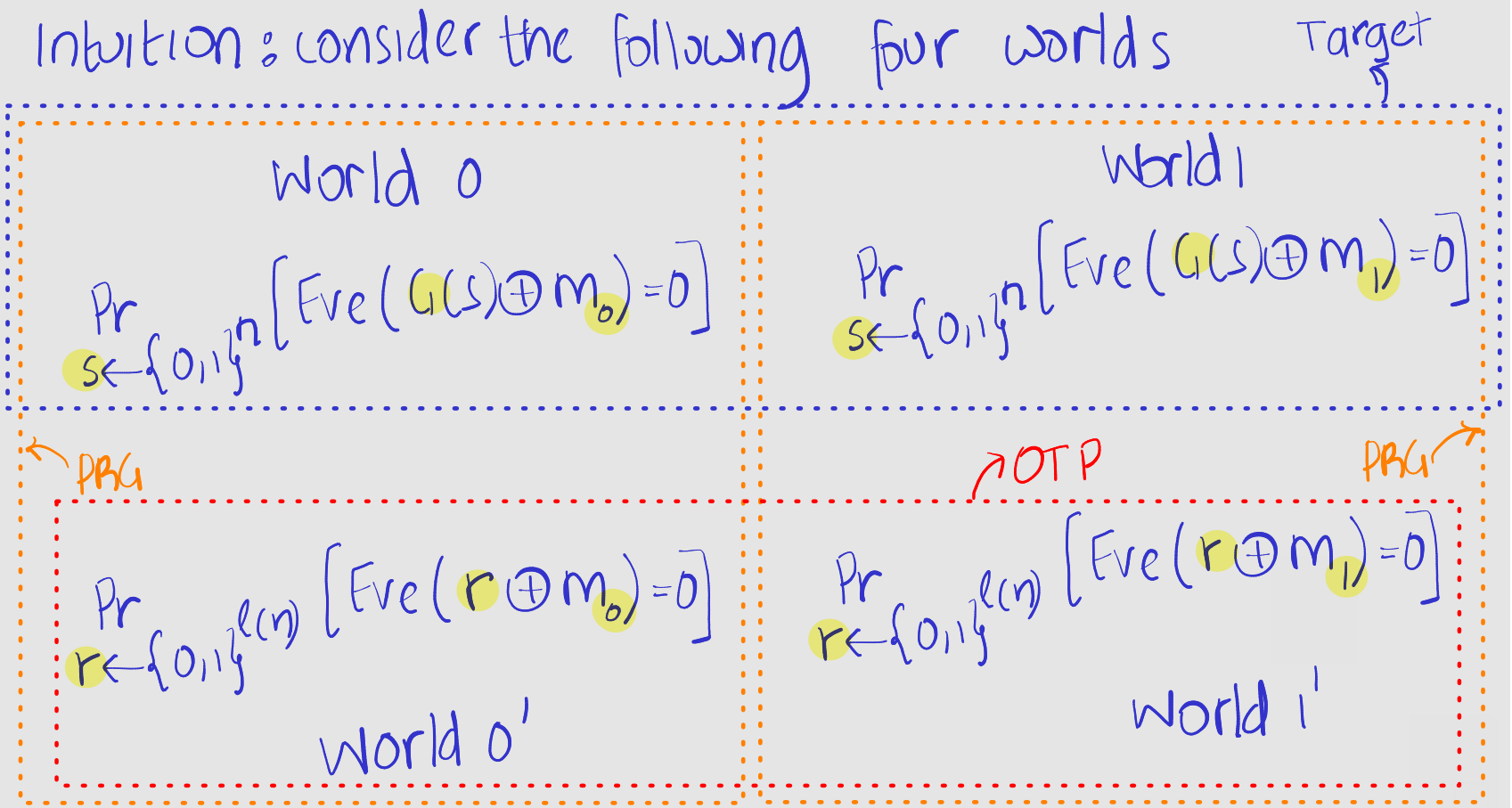


Proof of Computational Secrecy...

Theorem 3

Assuming G *is a PRG, Construction 2 is computationally secret.*

Proof *by reduction.* ∃Eve breaking Construction 2 ⇒ ∃D for G.



Proof of Computational Secrecy...

Theorem 3

Assuming G *is a PRG, Construction 2 is computationally secret.*

Proof *by reduction.* ∃Eve breaking Construction 2 ⇒ ∃D for G.

Proof of Computational Secrecy

Theorem 3

Assuming G *is a PRG, Construction 2 is computationally secret.*

Proof *by reduction.* ∃Eve breaking Construction 2 ⇒ ∃D for G.

Proof of Computational Secrecy

Theorem 3

Assuming G *is a PRG, Construction 2 is computationally secret.*

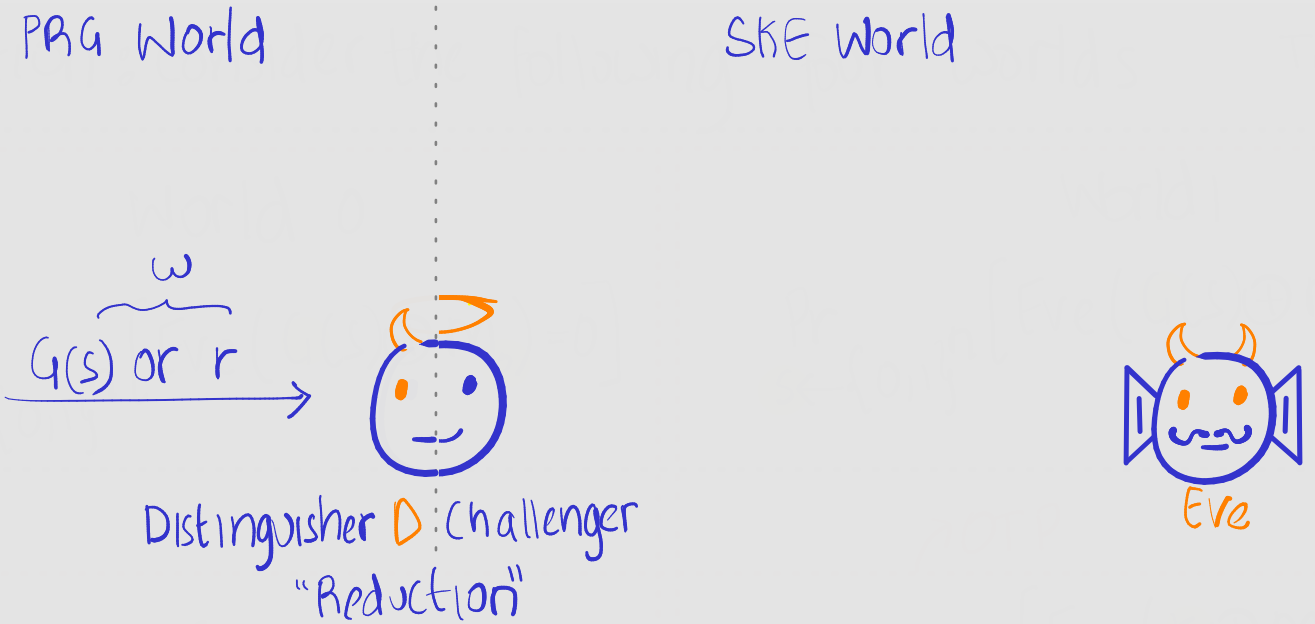
Proof *by reduction.* ∃Eve breaking Construction 2 ⇒ ∃D for G.

Proof of Computational Secrecy...

Theorem 3

Assuming G *is a PRG, Construction 2 is computationally secret.*

Proof *by reduction.* ∃Eve breaking Construction 2 ⇒ ∃D for G.

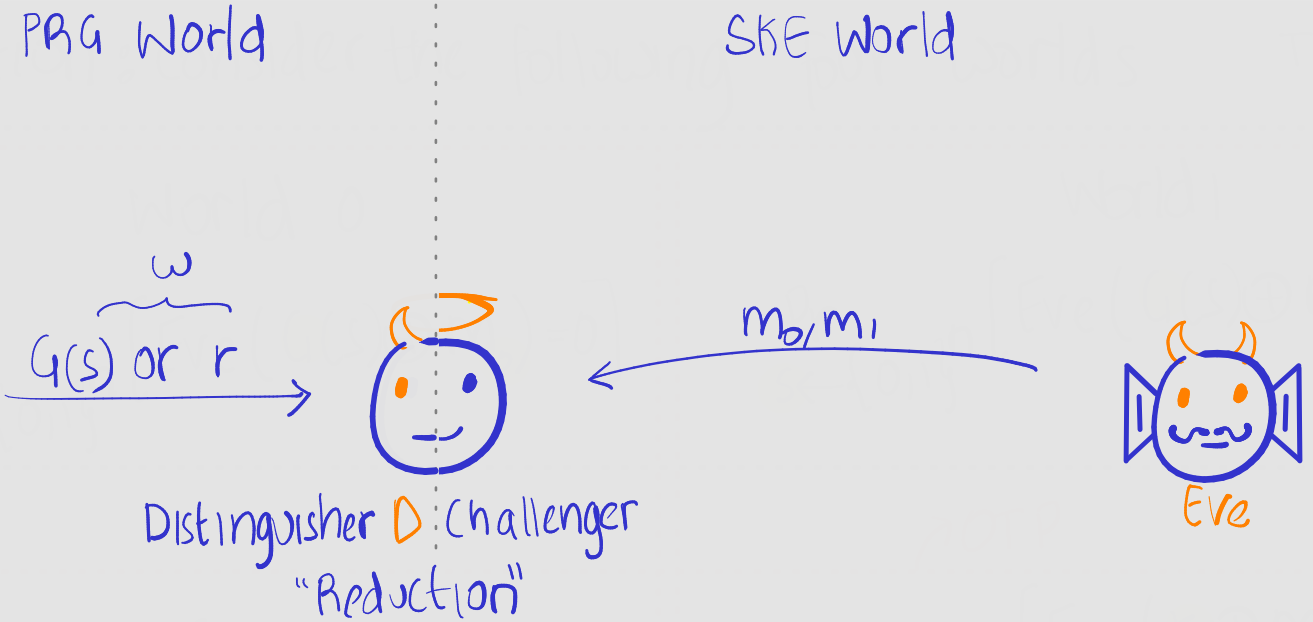


Proof of Computational Secrecy...

Theorem 3

Assuming G *is a PRG, Construction 2 is computationally secret.*

Proof *by reduction.* ∃Eve breaking Construction 2 ⇒ ∃D for G.

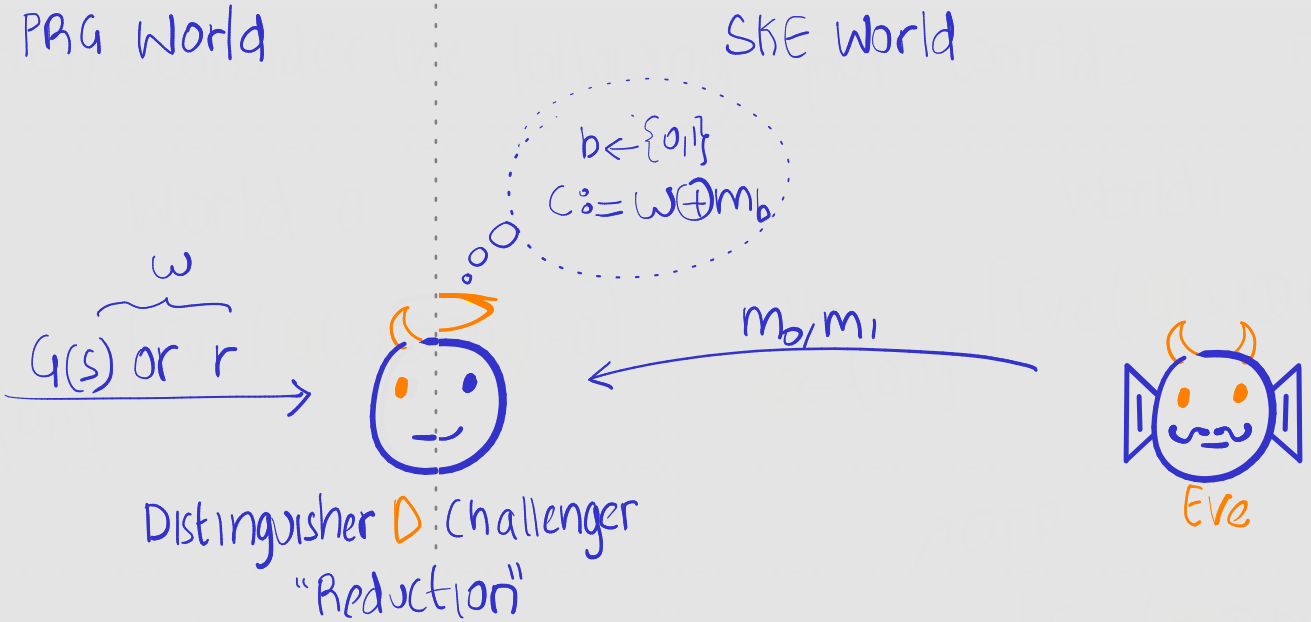


Proof of Computational Secrecy...

Theorem 3

Assuming G *is a PRG, Construction 2 is computationally secret.*

Proof *by reduction.* ∃Eve breaking Construction 2 ⇒ ∃D for G.

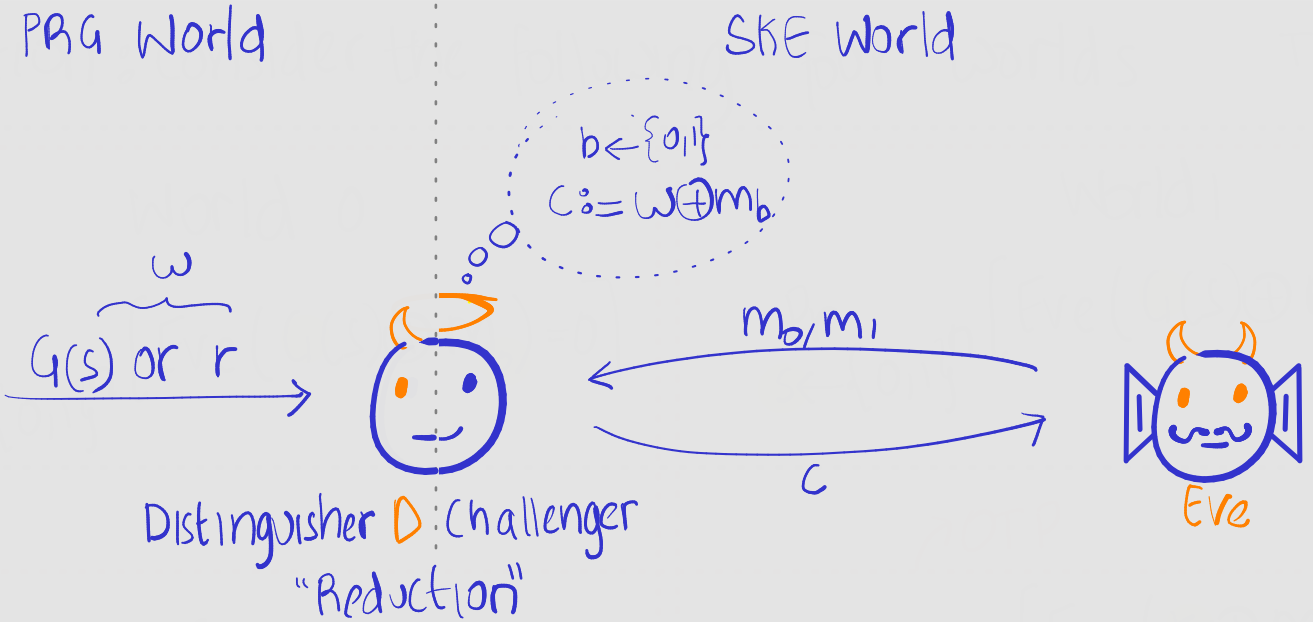


Proof of Computational Secrecy...

Theorem 3

Assuming G *is a PRG, Construction 2 is computationally secret.*

Proof *by reduction.* ∃Eve breaking Construction 2 ⇒ ∃D for G.

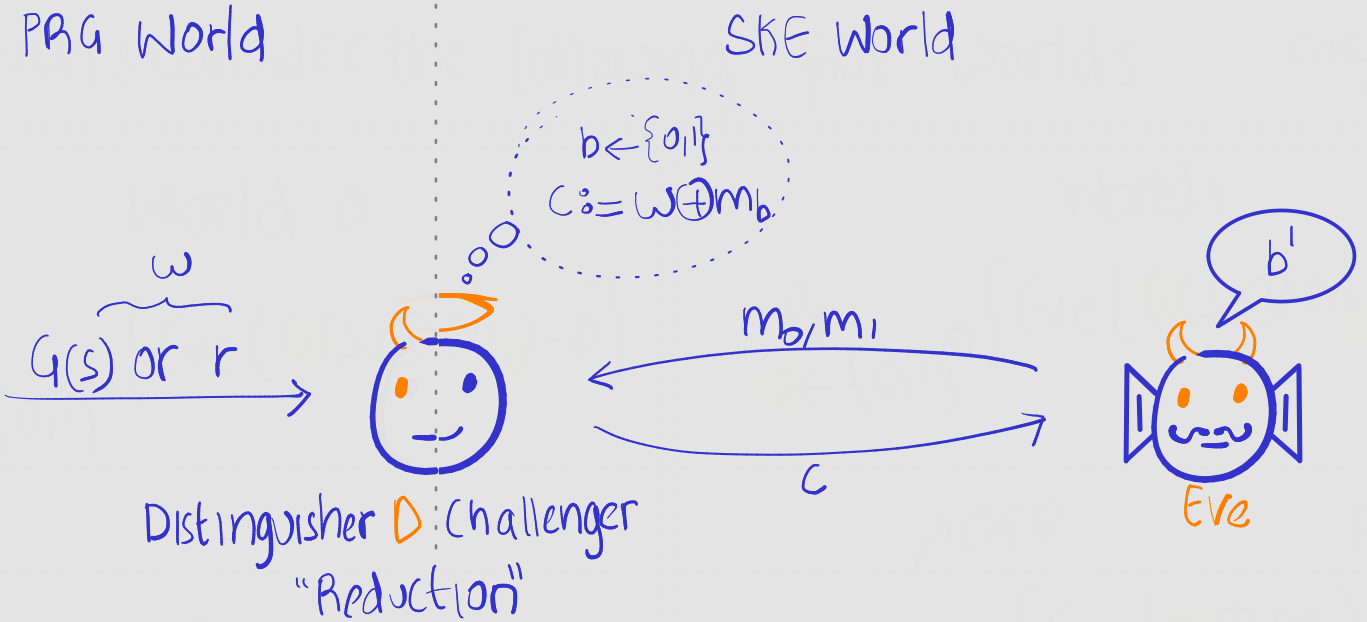


Proof of Computational Secrecy

Theorem 3

Assuming G *is a PRG, Construction 2 is computationally secret.*

Proof *by reduction.* ∃Eve breaking Construction 2 ⇒ ∃D for G.

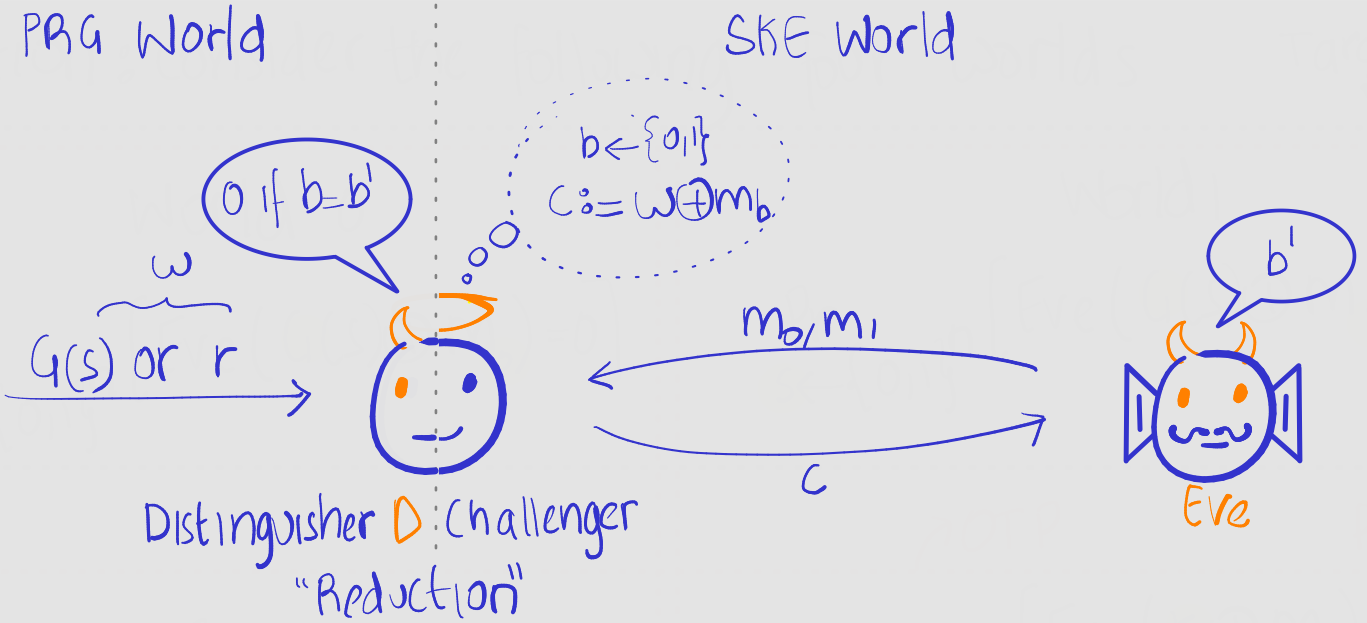


Proof of Computational Secrecy

Theorem 3

Assuming G *is a PRG, Construction 2 is computationally secret.*

Proof *by reduction.* ∃Eve breaking Construction 2 ⇒ ∃D for G.

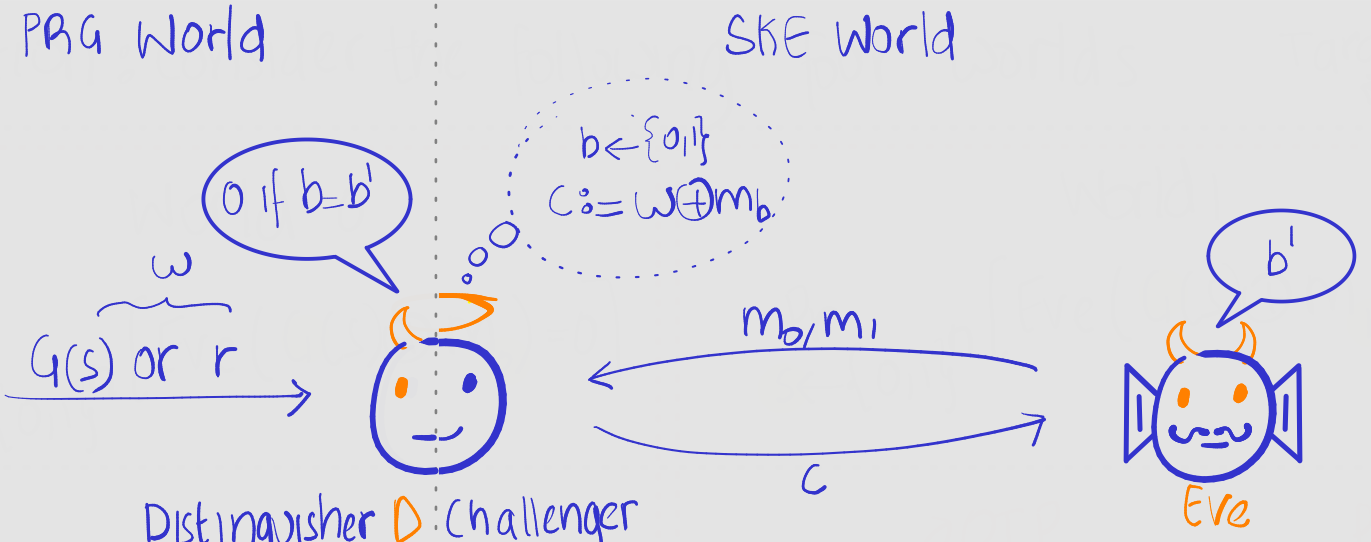


Proof of Computational Secrecy

Theorem 3

Assuming G *is a PRG, Construction 2 is computationally secret.*

Proof *by reduction.* ∃Eve breaking Construction 2 ⇒ ∃D for G.



Analysis

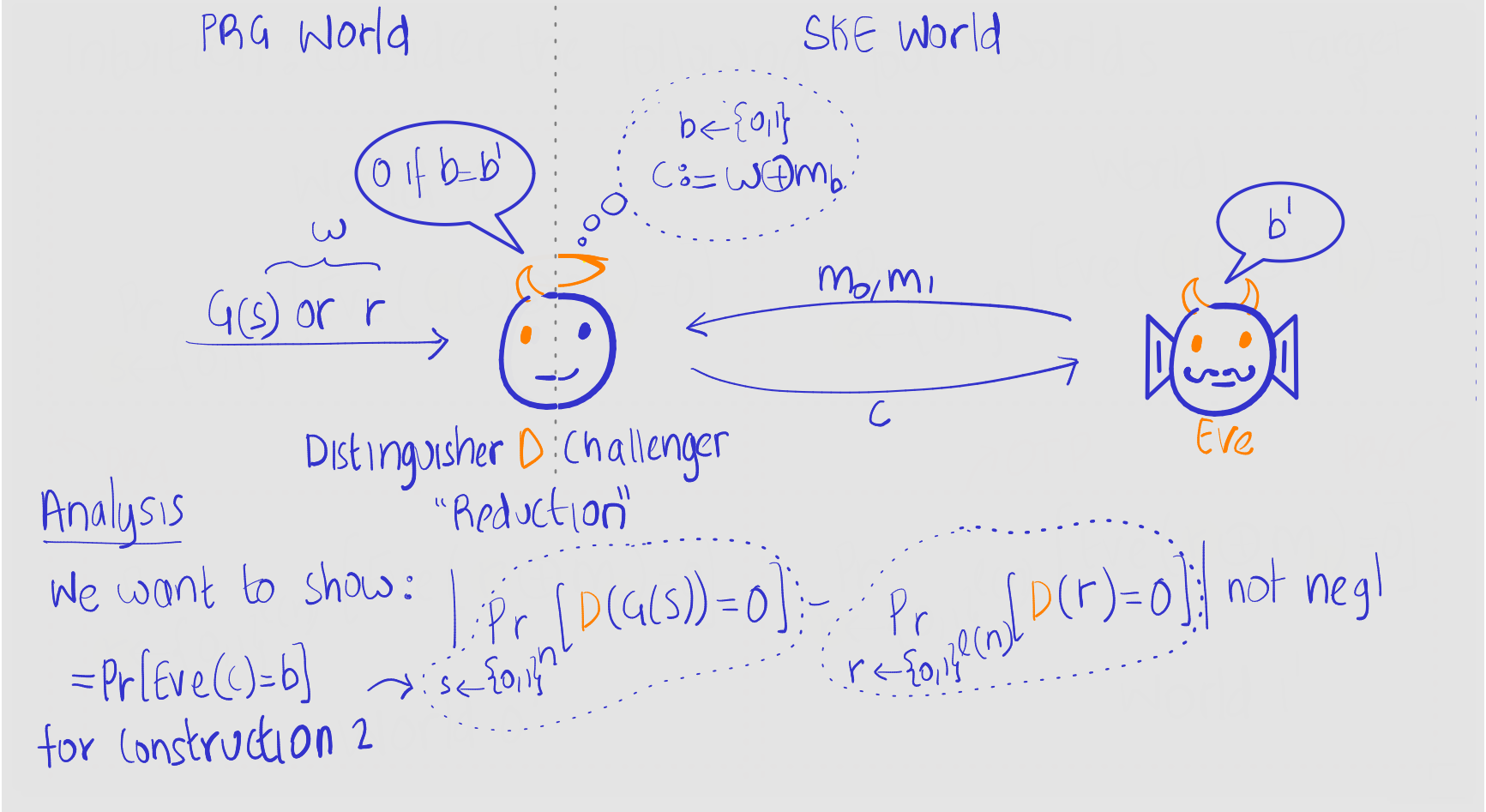
We want to show: $\left| \Pr_{s \leftarrow \{0,1\}^n}[D(G(s))=0] - \Pr_{r \leftarrow \{0,1\}^{\ell(n)}}[D(r)=0] \right|$ not negl

Proof of Computational Secrecy

Theorem 3

Assuming G *is a PRG, Construction 2 is computationally secret.*

Proof *by reduction.* ∃Eve breaking Construction 2 ⇒ ∃D for G.

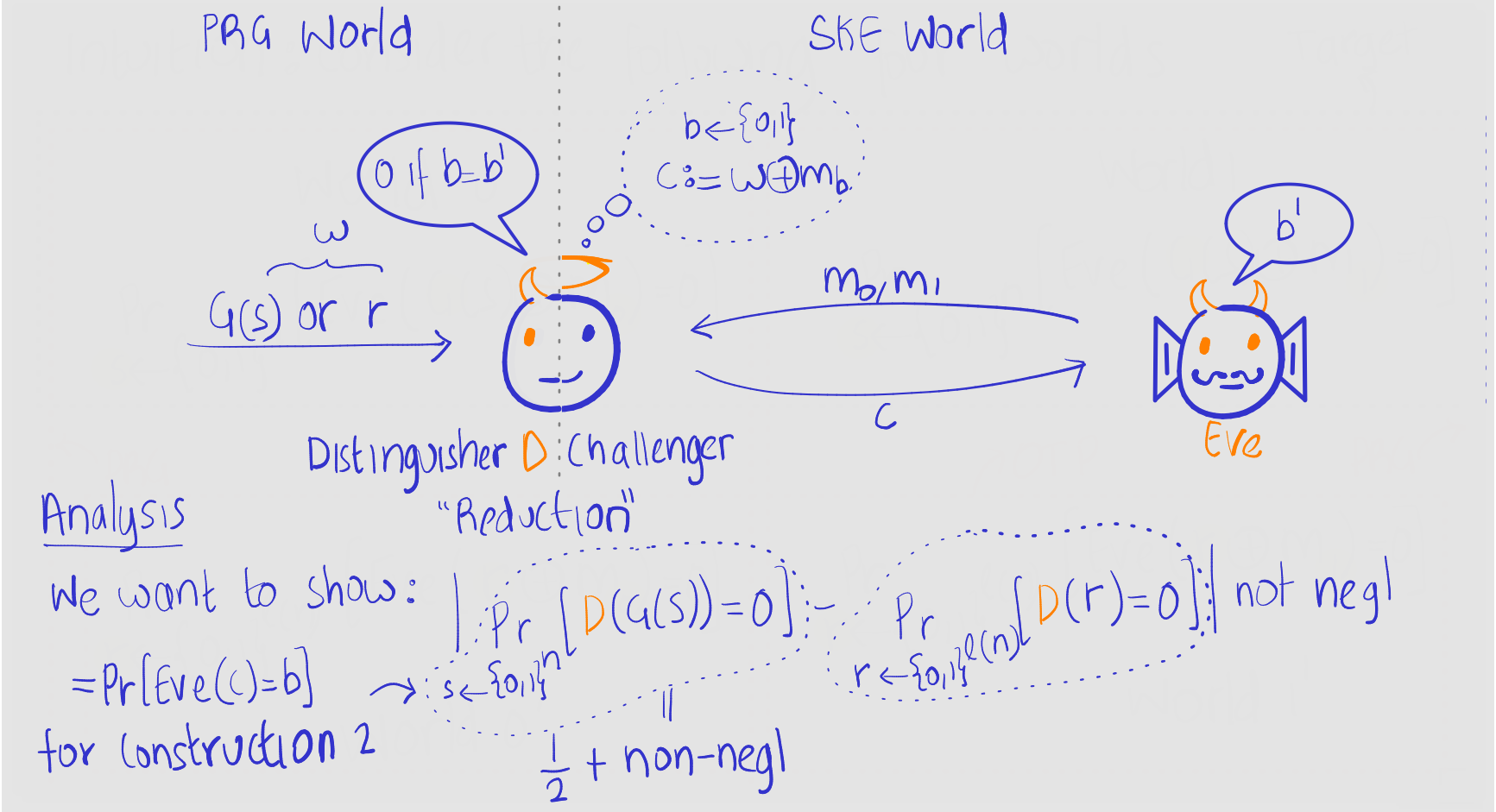


Proof of Computational Secrecy

Theorem 3

Assuming G *is a PRG, Construction 2 is computationally secret.*

Proof *by reduction.* ∃Eve breaking Construction 2 ⇒ ∃D for G.

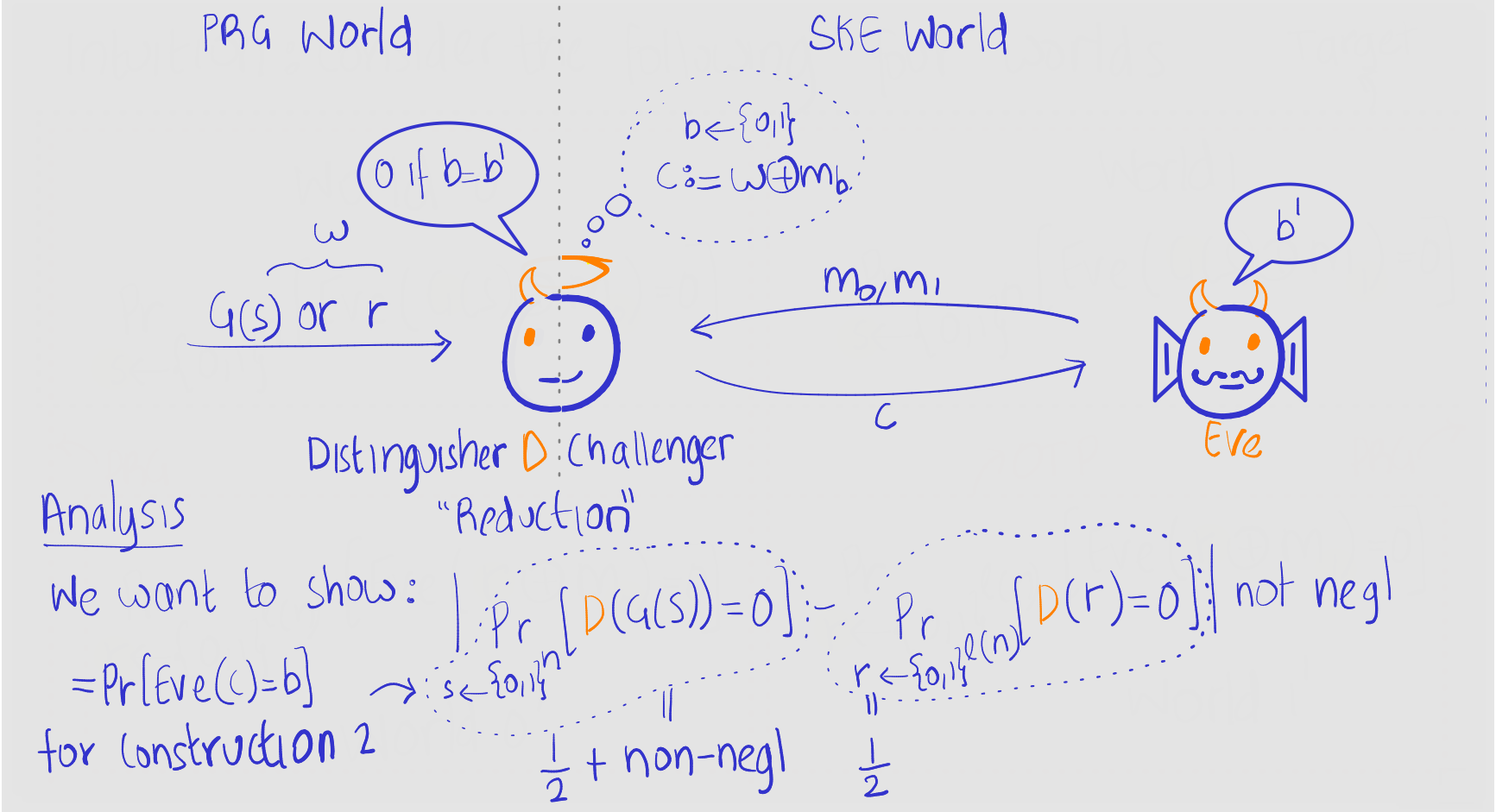


Proof of Computational Secrecy

Theorem 3

Assuming G *is a PRG, Construction 2 is computationally secret.*

Proof *by reduction.* ∃Eve breaking Construction 2 ⇒ ∃D for G.

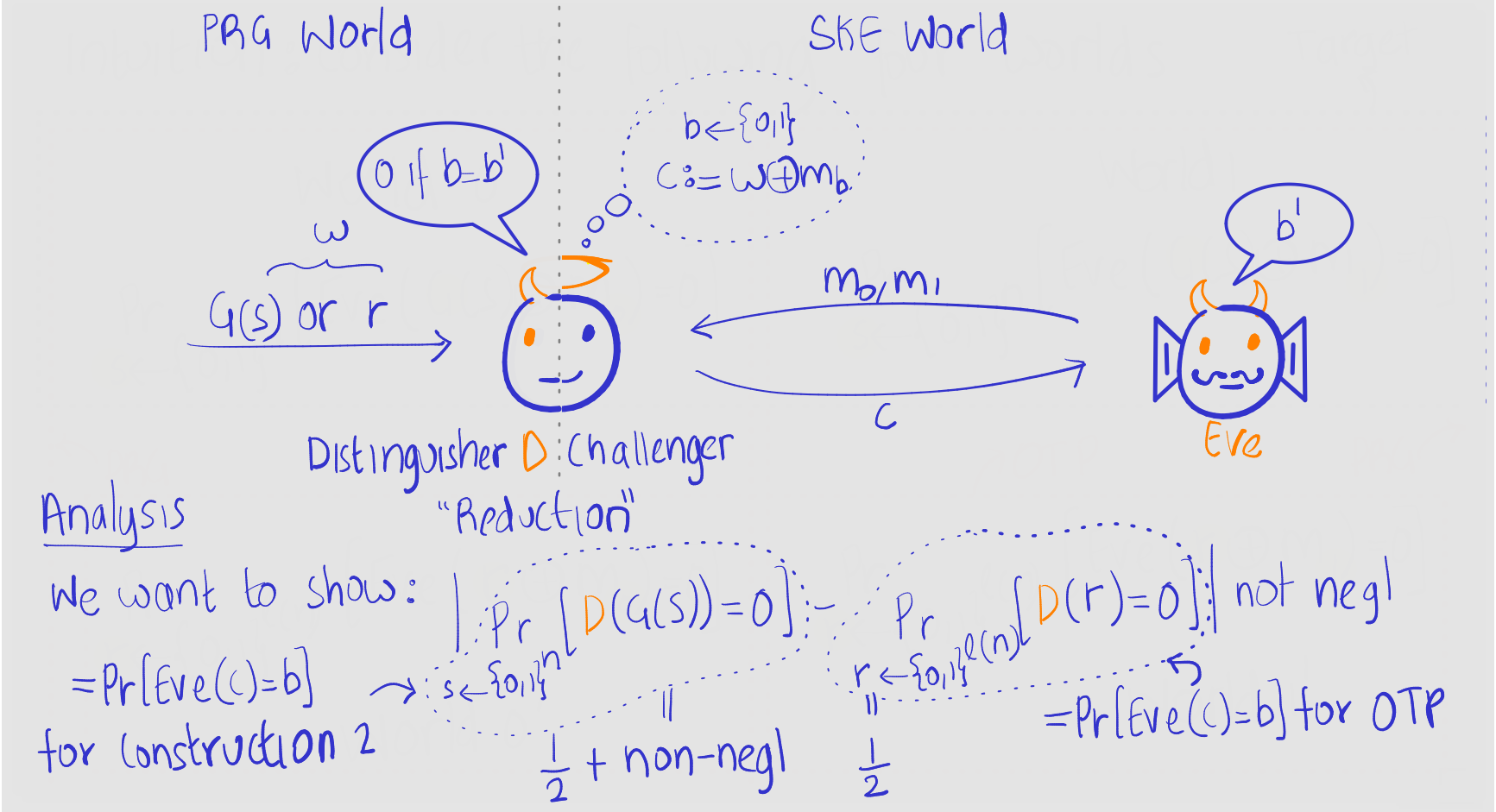


Proof of Computational Secrecy

Theorem 3

Assuming G *is a PRG, Construction 2 is computationally secret.*

Proof *by reduction.* ∃Eve breaking Construction 2 ⇒ ∃D for G.

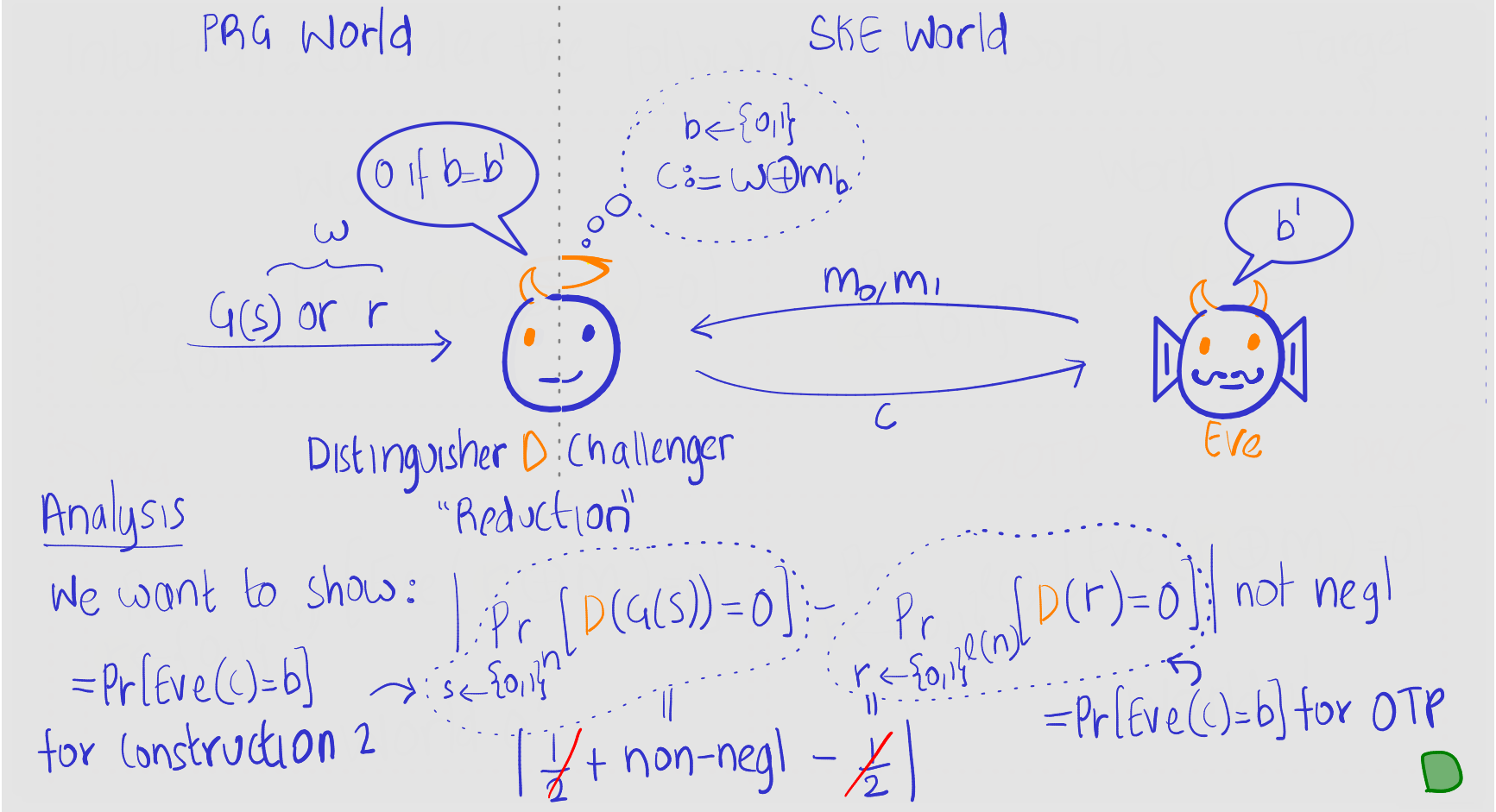


Proof of Computational Secrecy

Theorem 3

Assuming G *is a PRG, Construction 2 is computationally secret.*

Proof *by reduction.* ∃Eve breaking Construction 2 ⇒ ∃D for G.



Proof of Computational Secrecy...

Exercise 3 (Formalise proof of Theorem 3)

Write down the proof formally:

1. *Why does the reduction work?*
2. *In the analysis, explicitly write down expression for "not negligible".*

Proof of Computational Secrecy...

Exercise 3 (Formalise proof of Theorem 3)

Write down the proof formally:

1. *Why does the reduction work?*
2. *In the analysis, explicitly write down expression for "not negligible".*

Exercise 4 (Strengthening Theorem 3)

- *Understand how to model non-uniform adversaries in the circuit model.*

- *Show that assuming PRG secure against non-uniform distinguishers, one can achieve computational OTP against non-uniform eavesdroppers.*
- *Does the reduction work for quantum adversaries?*

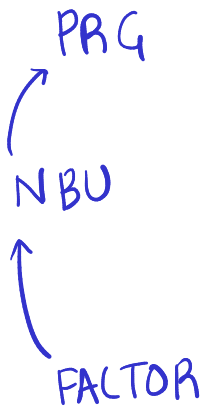
Do PRGs Exist?

- Recall from earlier that they don't if $\mathsf{P} = \mathsf{NP}$.
  - Thus they only exist *conditioned* on $\mathsf{P} \neq \mathsf{NP}$.

Do PRGs Exist?

- Recall from earlier that they don't if $\mathsf{P} = \mathsf{NP}$.
  - Thus they only exist *conditioned* on $\mathsf{P} \neq \mathsf{NP}$.

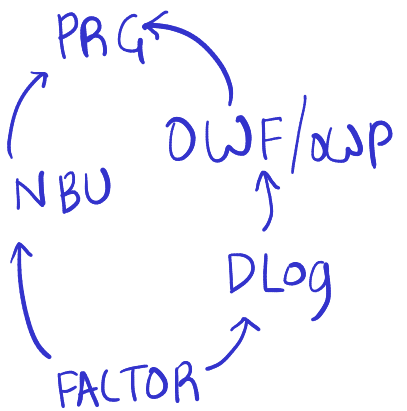
- Lecture 4 (next): PRGs from *next-bit unpredictable (NBU) functions* (stream ciphers)
  - Next-bit unpredictability can be achieved assuming hardness of *factoring* integers



Do PRGs Exist?

- Recall from earlier that they don't if $\mathsf{P} = \mathsf{NP}$.
  - Thus they only exist *conditioned* on $\mathsf{P} \neq \mathsf{NP}$.

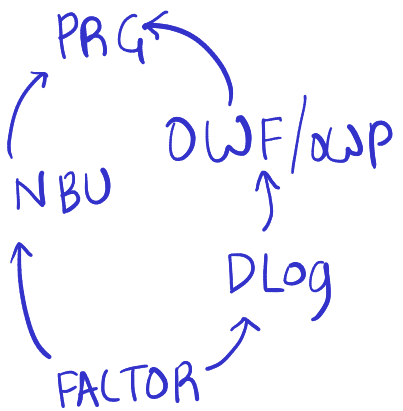
- Lecture 4 (next): PRGs from *next-bit unpredictable (NBU) functions* (stream ciphers)
  - Next-bit unpredictability can be achieved assuming hardness of *factoring* integers
- Lecture 7: pseudorandomness from *one-wayness*
  - One-wayness can be achieved under weaker assumptions (e.g., discrete logarithm)



Do PRGs Exist?

- Recall from earlier that they don't if $\mathsf{P} = \mathsf{NP}$.
  - Thus they only exist *conditioned* on $\mathsf{P} \neq \mathsf{NP}$.

- Lecture 4 (next): PRGs from *next-bit unpredictable (NBU) functions* (stream ciphers)
  - Next-bit unpredictability can be achieved assuming hardness of *factoring* integers
- Lecture 7: pseudorandomness from *one-wayness*
  - One-wayness can be achieved under weaker assumptions (e.g., discrete logarithm)



- Note. If pseudorandomness against *fixed-poly.* distinguishers suffices, then we can construct PRG under *complexity-theoretic* assumptions
  - Look up Nisan-Wigderson PRGs!

Applications of PRG

- Saw application of PRG to construct SKE

Applications of PRG

- Saw application of PRG to construct SKE
- Other applications
  - Fundamental to cryptography since most algorithms are randomised: helps reduce the amount of "pure" randomness required
  - Derandomisation, i.e., turn a randomised algorithm into deterministic
  - Non-cryptographic PRGs (e.g., LFSR): simulation in physics
    - But broken in cryptographic sense

To Recap

- We started off with Shannon's impossibility

To Recap

- We started off with Shannon's impossibility
- Learned how to overcome Shannon's impossibility via PRGs
  - Learned two new notions: PPT and negligible
  - There exist alternative ways
    - E.g.: bounded-storage model, where Eve's storage is limited
- Saw construction of computational OTP
  - First security reduction!

To Recap

- We started off with Shannon's impossibility
- Learned how to overcome Shannon's impossibility via PRGs
  - Learned two new notions: PPT and negligible
  - There exist alternative ways
    - E.g.: bounded-storage model, where Eve's storage is limited
- Saw construction of computational OTP
  - First security reduction!
- The security definitions can be seen through the lens of:

Defintion 6 (computational indistinguishability)

Two distributions X_0 and X_1 are computationally indistinguishable if for every PPT distinguisher D,

$$\delta(n) := \left| \Pr_{x \leftarrow X_0}[D(x) = 0] - \Pr_{x \leftarrow X_1}[D(x) = 0] \right|$$

is negligible.

Next Lecture

- Yet another way to look at PRGs: next-bit unpredictability (stream ciphers)
  - Concrete construction of PRG
- Length-extension for PRG
- First hybrid (security) argument!
- Introduce pseudo-random functions (PRFs)

Next Lecture

- Yet another way to look at PRGs: next-bit unpredictability (stream ciphers)
  - Concrete construction of PRG
- Length-extension for PRG
- First hybrid (security) argument!
- Introduce pseudo-random functions (PRFs)

More Questions?

References

1. [KL14, §3.1-3.3] for details about this lecture
2. [Gol01, §1.3] for a thorough treatment of the computational model used in cryptography (including a discussion of the non-uniform circuit model).
3. Foundational works on pseudorandomness were done by Blum and Micali [BM84] Yao [Yao82].
4. You can read about how PRGs are used for derandomisation in [AB09, Chapter 20]. This is also a great source for reading about complexity-theoretic (i.e., Nisan-Wigderson) PRG.
5. You can read about how Shannon's impossibility can be bypassed in the bounded storage model in [Mau92]


Sanjeev Arora and Boaz Barak.
Computational Complexity - A Modern Approach.
Cambridge University Press, 2009.


Manuel Blum and Silvio Micali.
How to generate cryptographically strong sequences of pseudo-random bits.
SIAM J. Comput., 13(4):850–864, 1984.


Oded Goldreich.
The Foundations of Cryptography - Volume 1: Basic Techniques.
Cambridge University Press, 2001.


Jonathan Katz and Yehuda Lindell.
Introduction to Modern Cryptography (3rd ed.).
Chapman and Hall/CRC, 2014.


Ueli M. Maurer.
Conditionally-perfect secrecy and a provably-secure randomized cipher.
Journal of Cryptology, 5(1):53–66, January 1992.


Andrew Chi-Chih Yao.
Theory and applications of trapdoor functions (extended abstract).
In *23rd FOCS*, pages 80–91. IEEE Computer Society Press, November 1982.