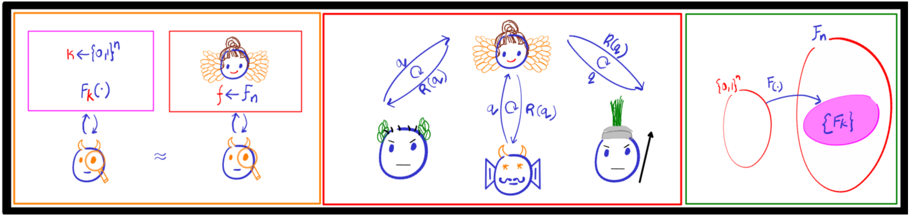


CS409m: Introduction to Cryptography

Lecture 07 (22/Aug/25)

Instructor: Chethan Kamath

Recall from Previous Lecture...

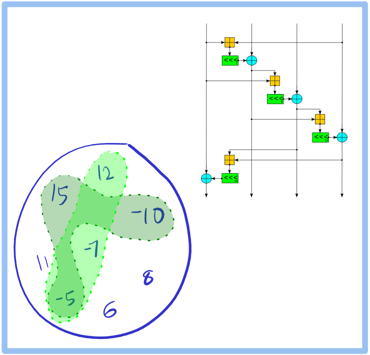
- Task: secure communication of *long messages* with shared keys
- Threat model: computational secrecy against eavesdroppers

Recall from Previous Lecture...

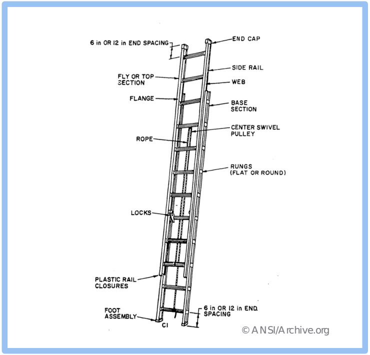
- Task: secure communication of very *long messages* with shared keys
- Threat model: computational secrecy against eavesdroppers

- Task: secure communication of very *long messages* with shared keys
- Threat model: computational secrecy against eavesdroppers

How to Construct PRGs?



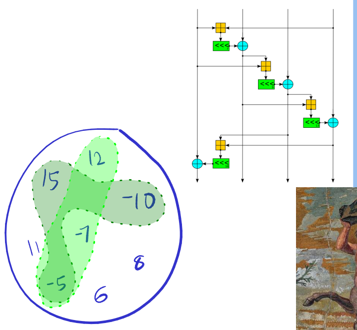
PRG Length Extension



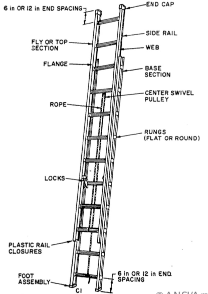
Recall from Previous Lecture . . .

- Task: secure communication of very *long messages* with shared keys
- Threat model: computational secrecy against eavesdroppers

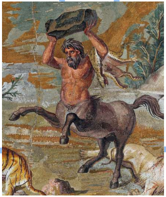
How to Construct PRGs?



PRG Length Extension



© ANSI/Archive.org


© Wikipedia

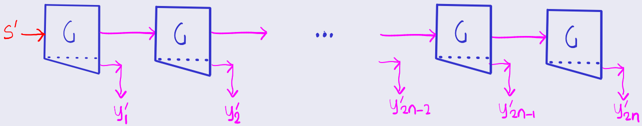
Main tool: hybrid argument

Recall from Previous Lecture...

Theorem 1

If G is a PRG, then so is G′.

Proof. ∃ distinguisher D for G ⇐ ∃ distinguisher D′ for G′.



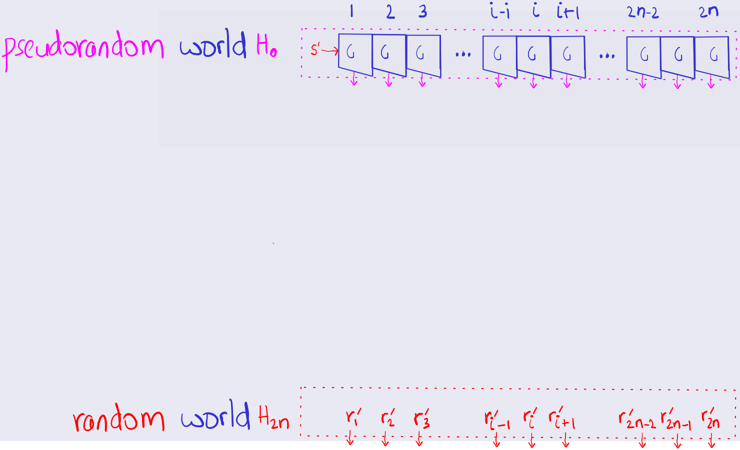
∎

Recall from Previous Lecture...

Theorem 1

If G is a PRG, then so is G′.

Proof. ∃ distinguisher D for G ⇐ ∃ distinguisher D′ for G′.

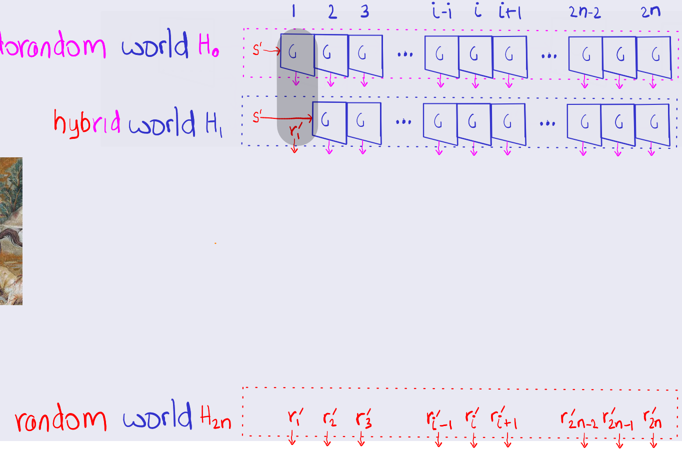


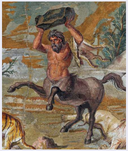
Recall from Previous Lecture...

Theorem 1

If G is a PRG, then so is G′.

Proof. ∃ distinguisher D for G ⇐ ∃ distinguisher D′ for G′.



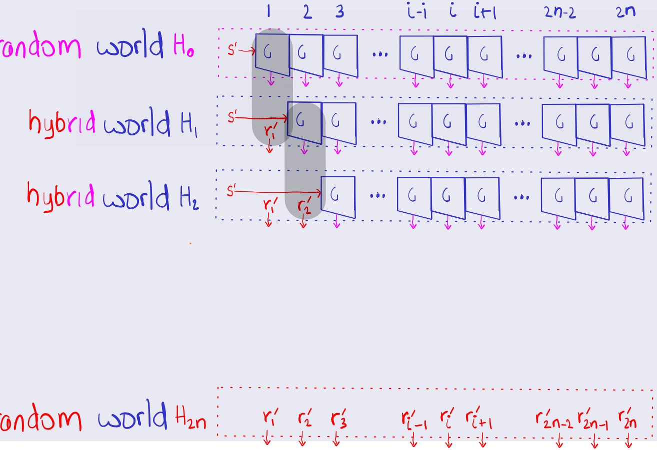


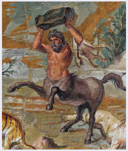
© Wikipedia

Theorem 1

If G is a PRG, then so is G′.

Proof. $\exists$ distinguisher D for G $\Leftarrow$ $\exists$ distinguisher D′ for G′.

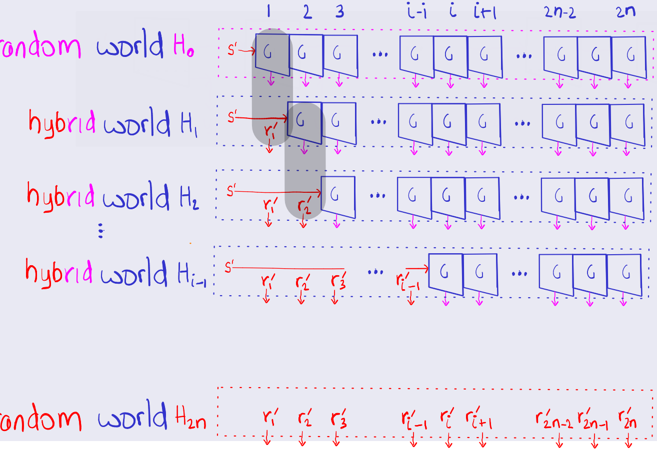


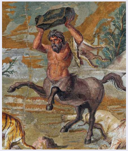


Theorem 1

If G is a PRG, then so is G′.

Proof. ∃ distinguisher D for G ⇐ ∃ distinguisher D′ for G′.





© Wikipedia

Recall from Previous Lecture...

Theorem 1

If G is a PRG, then so is G′.

Proof. $\exists$ distinguisher D for G $\Leftarrow$ $\exists$ distinguisher D′ for G′.

Two worlds of PRG G′:

pseudorandom world H_0

hybrid world H_1

hybrid world H_2

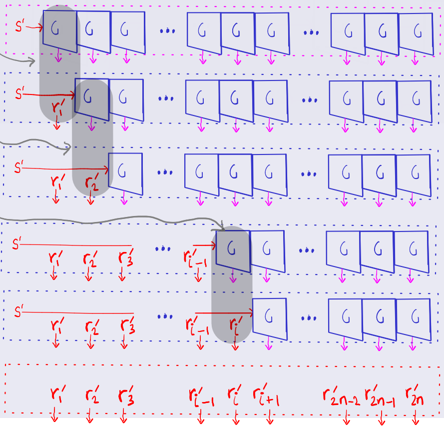
⋮

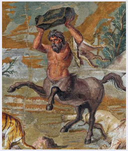
hybrid world H_{i-1}

hybrid world H_i

⋮

random world H_{2n}





© Wikipedia

Recall from Previous Lecture...

Theorem 1

If G is a PRG, then so is G′.

Proof. ∃ distinguisher D for G ⇐ ∃ distinguisher D′ for G′.

Two worlds of PRG G!

pseudorandom world H_0

hybrid world H_1

hybrid world H_2

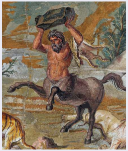
⋮

hybrid world H_{i-1}

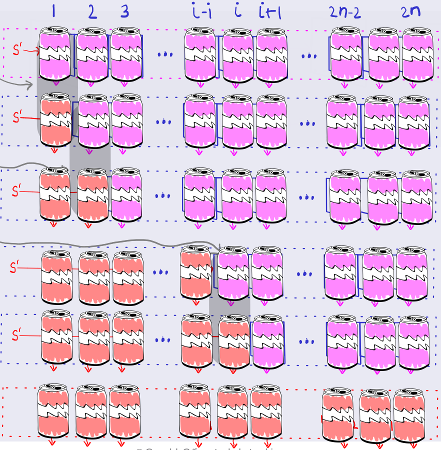
hybrid world H_i

⋮

random world H_{2n}



© Wikipedia



© Gerald_G/freestockphotos.biz

Theorem 1

If G is a PRG, then so is G′.

Proof. ∃ distinguisher D for G ⇐ ∃ distinguisher D′ for G′.

PRG G′

Distinguisher D

Distinguisher D′

Theorem 1

If G is a PRG, then so is G′.

Proof. $\exists$ distinguisher D for G $\Leftarrow$ $\exists$ distinguisher D′ for G′.

PRG G

PRG G′

Distinguisher D
"Reduction"

Distinguisher D′

Theorem 1

If G is a PRG, then so is G′.

Proof. $\exists$ distinguisher D for G $\Leftarrow$ $\exists$ distinguisher D′ for G′.

PRG G

PRG G′

$\underbrace{\omega = \omega^- \| \omega^+}$

G(s) or random r $\rightarrow$

Distinguisher D

"Reduction"

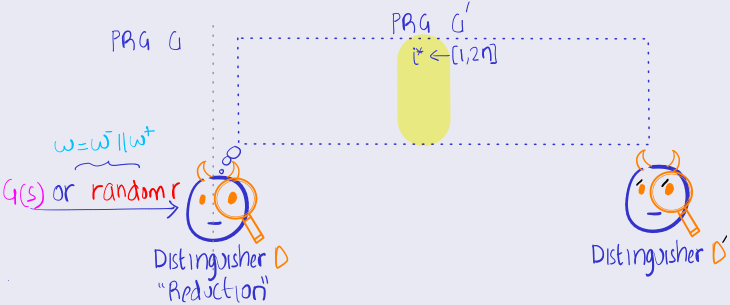
Distinguisher D′

Recall from Previous Lecture...

Theorem 1

If G is a PRG, then so is G′.

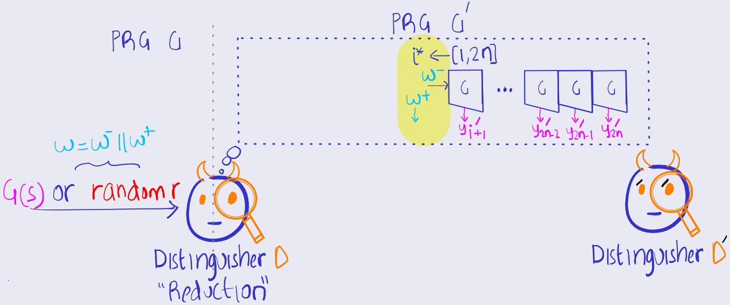
Proof. ∃ distinguisher D for G ⇐ ∃ distinguisher D′ for G′.



Theorem 1

If G is a PRG, then so is G′.

Proof. ∃ distinguisher D for G ⇐ ∃ distinguisher D′ for G′.

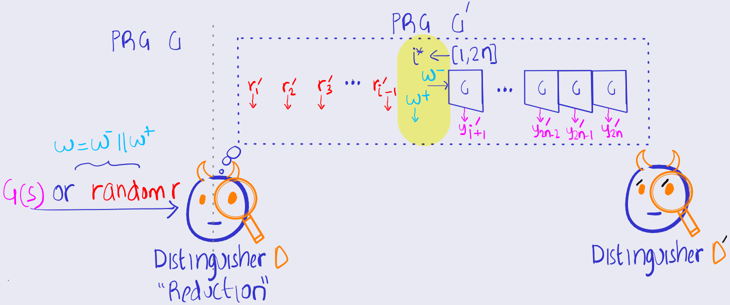


Recall from Previous Lecture...

Theorem 1

If G is a PRG, then so is G′.

Proof. ∃ distinguisher D for G ⇐ ∃ distinguisher D′ for G′.

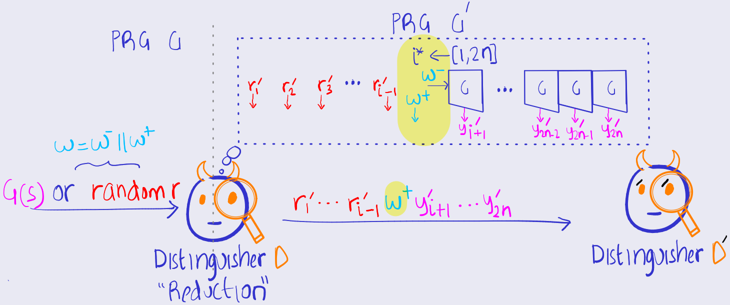


Recall from Previous Lecture...

Theorem 1

If G is a PRG, then so is G′.

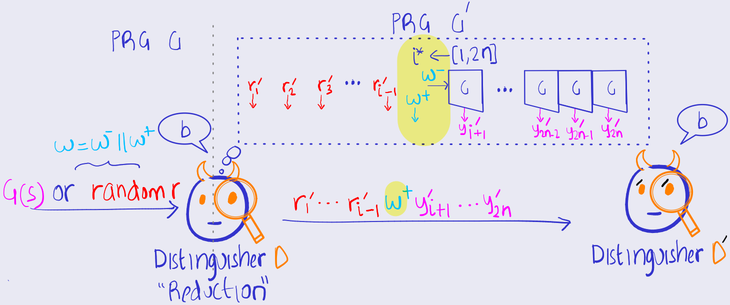
Proof. ∃ distinguisher D for G ⇐ ∃ distinguisher D′ for G′.



Theorem 1

If G is a PRG, then so is G′.

Proof. ∃ distinguisher D for G ⇐ ∃ distinguisher D′ for G′.

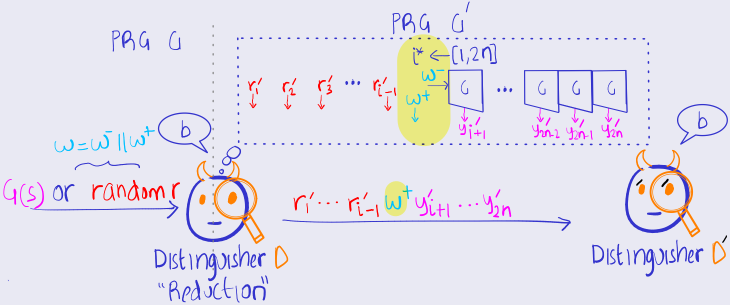


Recall from Previous Lecture...

Theorem 1

If G is a PRG, then so is G′.

Proof. ∃ distinguisher D for G ⇐ ∃ distinguisher D′ for G′.



Analysis: similar to claims 1 and 2.

$\Pr[D'(H_0)=0] - \Pr[D'(H_{2n})=0] \geq \delta \Rightarrow$ $\qquad \Pr[i^*=i] = 1/2n$

$\exists i \in [0, 2n-1]$ such that $\Pr[D'(H_i)=0] - \Pr[D'(H_{i+1})=0] \geq \delta/2n$

Let's Take Stock of Theorem 1

- Construction and Theorem 1 work for any polynomial stretch
  - What happens if we stretch it *exponentially*?

Let's Take Stock of Theorem 1

- Construction and Theorem 1 work for any polynomial stretch
  - What happens if we stretch it *exponentially*?

- There is also a "loss in pseudorandomness"
  - D′ distinguishes with some probability $1/p(n) \Rightarrow$ D distinguishes with probability only $\approx 2n/p(n)$

Let's Take Stock of Theorem 1

- Construction and Theorem 1 work for any polynomial stretch

  What happens if we stretch it *exponentially*?

- There is also a "loss in pseudorandomness"
  - D′ distinguishes with some probability $1/p(n) \Rightarrow$ D distinguishes with probability only $\approx 2n/p(n)$
  - More the stretch, greater the loss

- More generally: "loss in security" of a security reduction
  - One way to measure how "wasteful" the reduction is

Let's Take Stock of Theorem 1

- Construction and Theorem 1 work for any polynomial stretch
  - What happens if we stretch it *exponentially*?

- There is also a "loss in pseudorandomness"
  - D′ distinguishes with some probability $1/p(n) \Rightarrow$ D distinguishes with probability only $\approx 2n/p(n)$
  - More the stretch, greater the loss

- More generally: "loss in security" of a security reduction
  - One way to measure how "wasteful" the reduction is

Exercise 1

- Think of a less wasteful reduction strategy for Theorem 1. Do you feel it is possible?
- Maybe need a different construction?

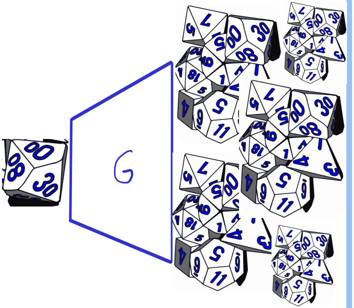
Plan for Today's Lecture

- Task: secure communication of *multiple messages* with shared keys
- Threat model: computational secrecy against eavesdroppers

Plan for Today's Lecture

- Task: secure communication of *multiple messages* with shared keys
- Threat model: computational secrecy against eavesdroppers

Pseudo-Random Function (PRF)



© Openclipart/Freesvg.org

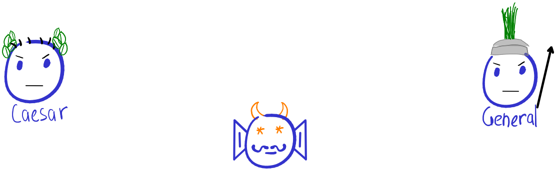
GGM PRF

© Biodiversity Heritage Library

© Oded Goldreich

Let's Encrypt Many Messages Using PRG G...

- Setting: Caesar and his general share a key $k \in \{0,1\}^n$ and want to secretly communicate n messages from $\{0,1\}^n$ in presence of eavesdropper Eve*

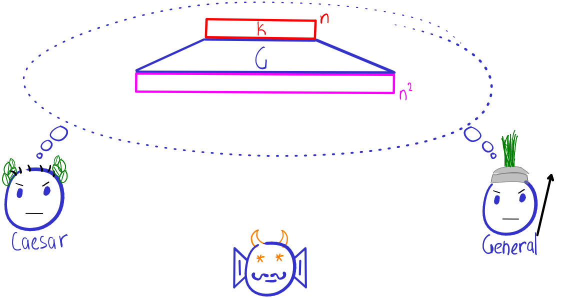


Let's Encrypt Many Messages Using PRG G

$= n$

$n \to n^2$

- Setting: Caesar and his general share a key $k \in \{0,1\}^n$ and want to secretly communicate n messages from $\{0,1\}^n$ in presence of eavesdropper Eve*

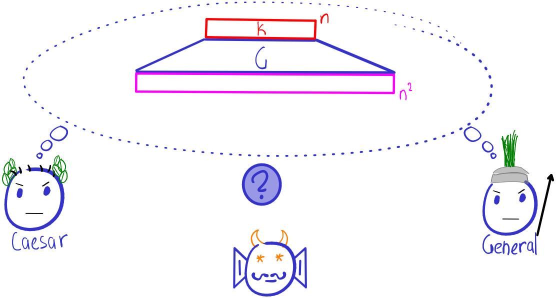


Let's Encrypt Many Messages Using PRG G

$= n$

$n \to n^2$

- Setting: Caesar and his general share a key $k \in \{0,1\}^n$ and want to secretly communicate n messages from $\{0,1\}^n$ in presence of eavesdropper Eve*

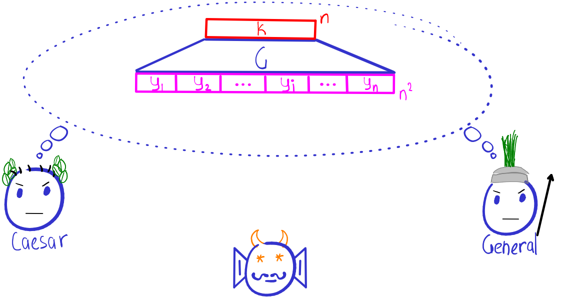


Let's Encrypt Many Messages Using PRG G

$= n$

$n \to n^2$

- Setting: Caesar and his general share a key $k \in \{0,1\}^n$ and want to secretly communicate n messages from $\{0,1\}^n$ in presence of eavesdropper Eve*



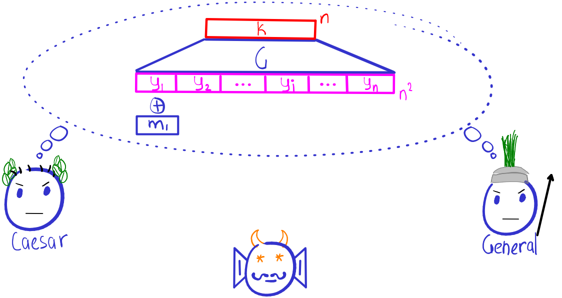
- SKE construction: use output of G as n pseudorandom OTPs

Let's Encrypt Many Messages Using PRG G

$= n$

$n \rightarrow n^2$

- Setting: Caesar and his general share a key $k \in \{0,1\}^n$ and want to secretly communicate n messages from $\{0,1\}^n$ in presence of eavesdropper Eve*



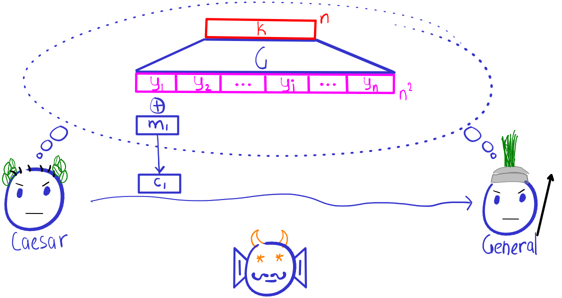
- SKE construction: use output of G as n pseudorandom OTPs

Let's Encrypt Many Messages Using PRG G

$\parallel$
n

$n \to n^2$

- Setting: Caesar and his general share a key $k \in \{0,1\}^n$ and want to secretly communicate n messages from $\{0,1\}^n$ in presence of eavesdropper Eve*



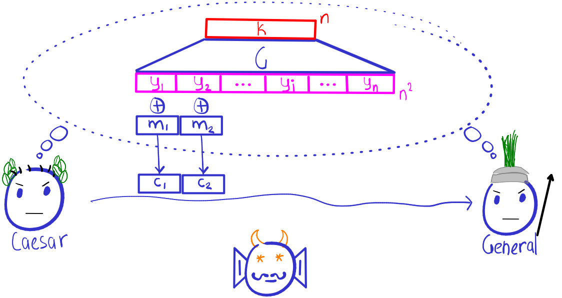
- SKE construction: use output of G as n pseudorandom OTPs

Let's Encrypt Many Messages Using PRG G

$= n$

$n \to n^2$

- Setting: Caesar and his general share a key $k \in \{0,1\}^n$ and want to secretly communicate n messages from $\{0,1\}^n$ in presence of eavesdropper Eve*

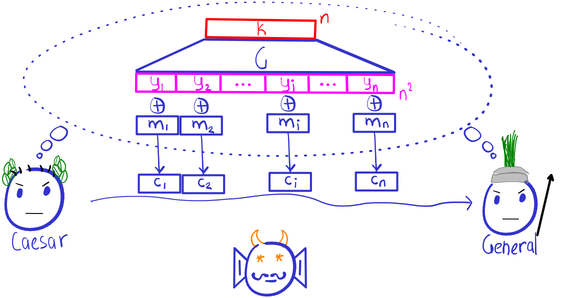


- SKE construction: use output of G as n pseudorandom OTPs

Let's Encrypt Many Messages Using PRG G

$= n$ $\qquad$ $n \to n^2$

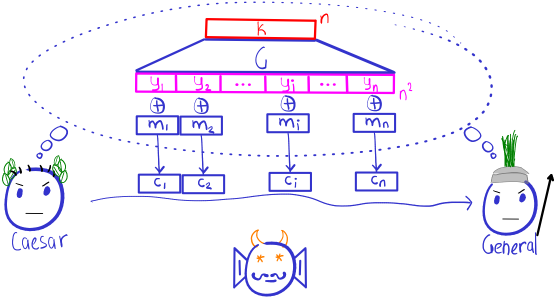
- Setting: Caesar and his general share a key $k \in \{0,1\}^n$ and want to secretly communicate n messages from $\{0,1\}^n$ in presence of eavesdropper Eve*



- SKE construction: use output of G as n pseudorandom OTPs

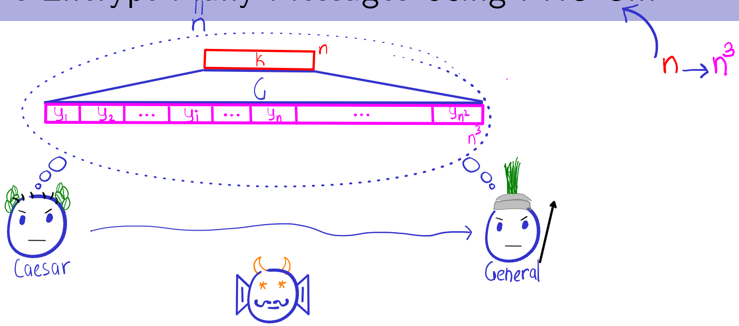
Let's Encrypt Many Messages Using PRG G

- Setting: Caesar and his general share a key $k \in \{0,1\}^n$ and want to secretly communicate n messages from $\{0,1\}^n$ in presence of eavesdropper Eve*



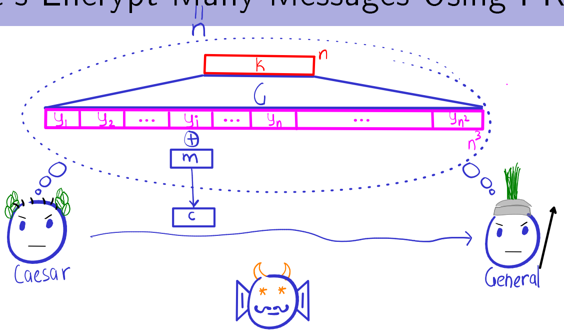
- SKE construction: use output of G as n pseudorandom OTPs
- Problem: construction *stateful*; synchrony must be maintained
  - We lose correctness if (e.g.) ciphertexts delivered out of order
  - ? Come up with a scenario that leads to loss of secrecy

Let's Encrypt Many Messages Using PRG G...

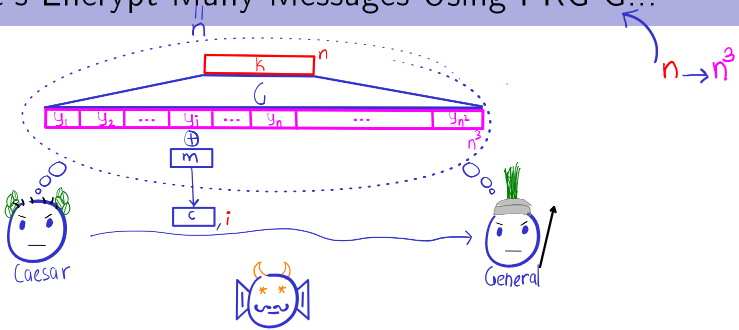


What if the stretch is n^3?

$n \rightarrow n^3$

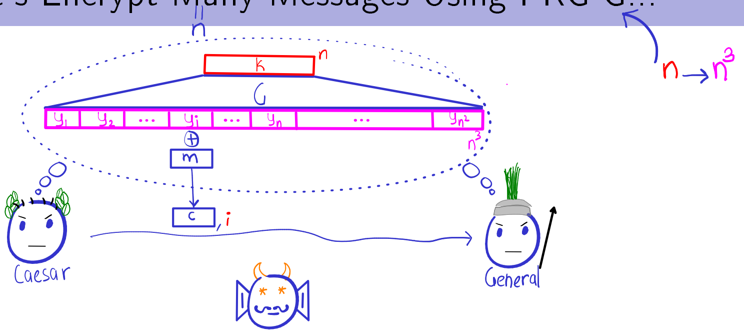


What if the stretch is n^3? Use OTP at random index $i \in [1, n^2]$



What if the stretch is n^3? Use OTP at random index $i \in [1, n^2]$

Let's Encrypt Many Messages Using PRG G...

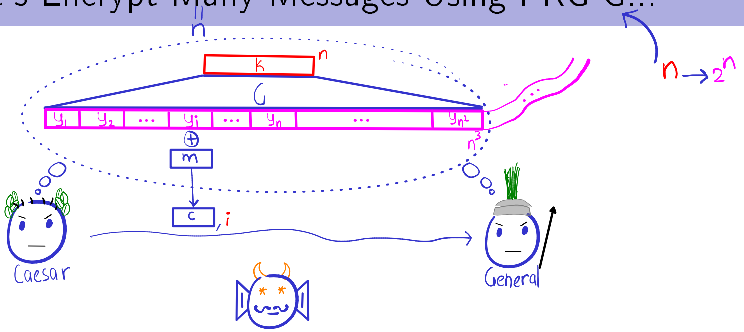


What if the stretch is n^3? Use OTP at random index $i \in [1, n^2]$

Problem? Collision

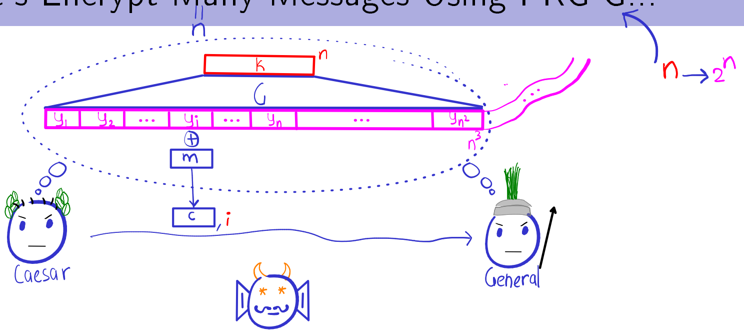
- Underlying problem: only poly. pseudorandom OTPs available

Let's Encrypt Many Messages Using PRG G...

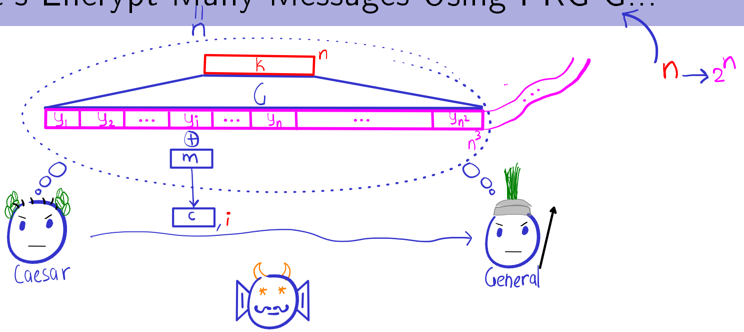


- What if the stretch is n^3? Use OTP at random index $i \in [1, n^2]$
  - Problem? Collision
- Underlying problem: only poly. pseudorandom OTPs available
- What if we stretch the PRG exponentially?

Let's Encrypt Many Messages Using PRG G...



- What if the stretch is n^3? Use OTP at random index $i \in [1, n^2]$
  - Problem? Collision
- Underlying problem: only poly. pseudorandom OTPs available
- What if we stretch the PRG exponentially?
  - Not all pseudorandom OTPs are efficiently "accessible"



- What if the stretch is n^3? Use OTP at random index $i \in [1, n^2]$
  - Problem? Collision
- Underlying problem: only poly. pseudorandom OTPs available
- What if we stretch the PRG exponentially?
  - Not all pseudorandom OTPs are efficiently "accessible"
- Need "PRG" with
  1. Exponential stretch
  2. Output bits "efficiently" accessible (also called locality)

Let's Encrypt Many Messages Using an Oracle in the Sky

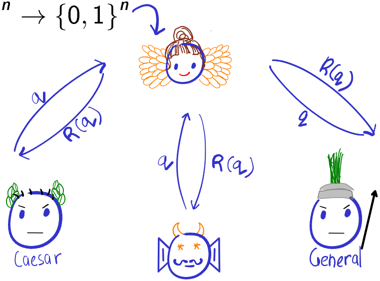
- Setting:
  - Caesar and his general have shared a key $k \in \{0,1\}^n$
  - Everyone (including Eve*) has access to a *random function oracle* $R : \{0,1\}^{2n} \to \{0,1\}^n$

Let's Encrypt Many Messages Using an Oracle in the Sky

- Setting:
  - Caesar and his general have shared a key $k \in \{0,1\}^n$
  - Everyone (including Eve*) has access to a *random function oracle* $R : \{0,1\}^{2n} \rightarrow \{0,1\}^n$

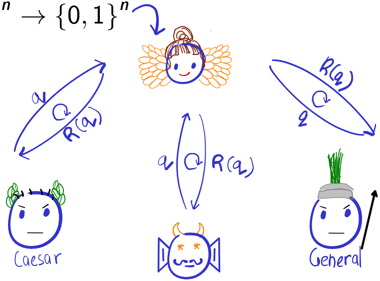
Let's Encrypt Many Messages Using an Oracle in the Sky

- Setting:
  - Caesar and his general have shared a key $k \in \{0,1\}^n$
  - Everyone (including Eve*) has access to a *random function oracle* $R : \{0,1\}^{2n} \to \{0,1\}^n$



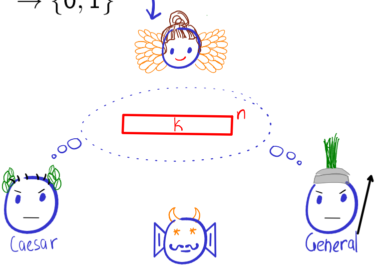
Let's Encrypt Many Messages Using an Oracle in the Sky

- Setting:
  - Caesar and his general have shared a key $k \in \{0,1\}^n$
  - Everyone (including Eve*) has access to a *random function oracle* $R : \{0,1\}^{2n} \to \{0,1\}^n$



Let's Encrypt Many Messages Using an Oracle in the Sky

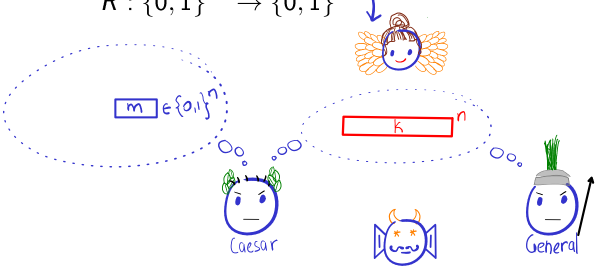
- Setting:
  - Caesar and his general have shared a key $k \in \{0,1\}^n$
  - Everyone (including Eve*) has access to a *random function oracle* $R : \{0,1\}^{2n} \to \{0,1\}^n$



How will you construct a stateless encryption scheme given R?

Let's Encrypt Many Messages Using an Oracle in the Sky

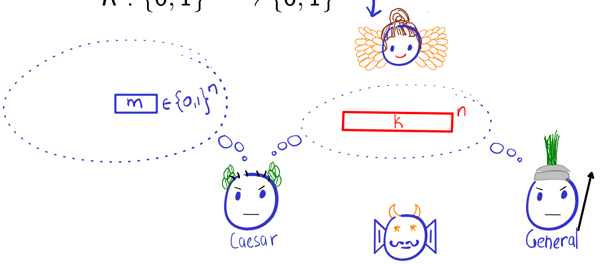
- Setting:
  - Caesar and his general have shared a key $k \in \{0,1\}^n$
  - Everyone (including Eve*) has access to a *random function oracle* $R : \{0,1\}^{2n} \to \{0,1\}^n$



How will you construct a stateless encryption scheme given R?

Let's Encrypt Many Messages Using an Oracle in the Sky

- Setting:
  - Caesar and his general have shared a key $k \in \{0,1\}^n$
  - Everyone (including Eve*) has access to a *random function oracle* $R : \{0,1\}^{2n} \to \{0,1\}^n$

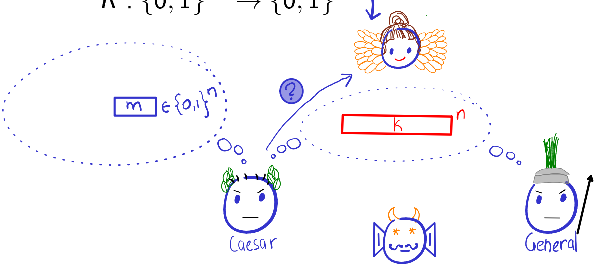


How will you construct a stateless encryption scheme given R?

Hint: R helps generate *exponentially-many* random OTPs

Let's Encrypt Many Messages Using an Oracle in the Sky

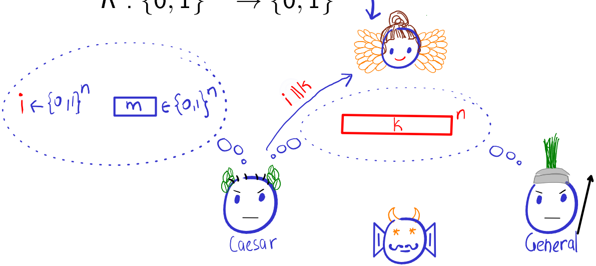
- Setting:
  - Caesar and his general have shared a key $k \in \{0,1\}^n$
  - Everyone (including Eve*) has access to a *random function oracle* $R : \{0,1\}^{2n} \to \{0,1\}^n$



? How will you construct a stateless encryption scheme given R?
  - Hint: R helps generate *exponentially-many* random OTPs

Let's Encrypt Many Messages Using an Oracle in the Sky

- Setting:
  - Caesar and his general have shared a key $k \in \{0,1\}^n$
  - Everyone (including Eve*) has access to a *random function oracle* $R : \{0,1\}^{2n} \to \{0,1\}^n$

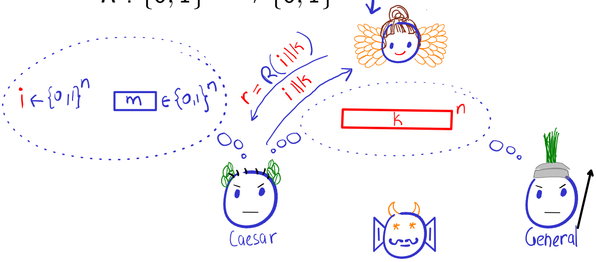


How will you construct a stateless encryption scheme given R?

Hint: R helps generate *exponentially-many* random OTPs

Let's Encrypt Many Messages Using an Oracle in the Sky

- Setting:
  - Caesar and his general have shared a key $k \in \{0,1\}^n$
  - Everyone (including Eve*) has access to a *random function oracle* $R : \{0,1\}^{2n} \to \{0,1\}^n$

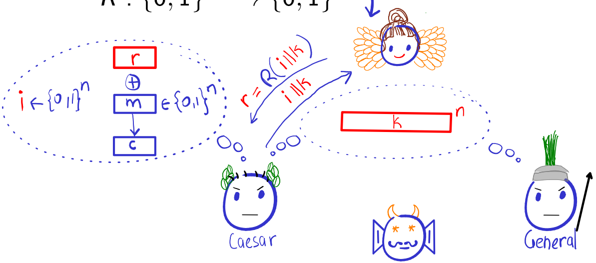


How will you construct a stateless encryption scheme given R?

Hint: R helps generate *exponentially-many* random OTPs

Let's Encrypt Many Messages Using an Oracle in the Sky

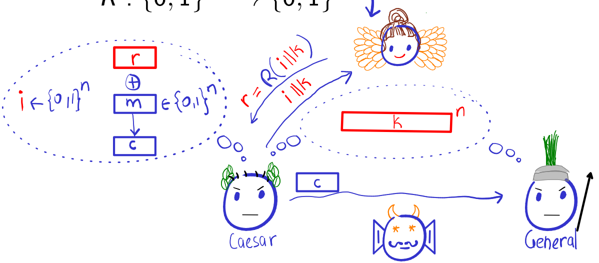
- Setting:
  - Caesar and his general have shared a key $k \in \{0,1\}^n$
  - Everyone (including Eve*) has access to a *random function oracle* $R : \{0,1\}^{2n} \to \{0,1\}^n$



❓ How will you construct a stateless encryption scheme given R?
  - Hint: R helps generate *exponentially-many* random OTPs

Let's Encrypt Many Messages Using an Oracle in the Sky

- Setting:
  - Caesar and his general have shared a key $k \in \{0,1\}^n$
  - Everyone (including Eve*) has access to a *random function oracle* $R : \{0,1\}^{2n} \to \{0,1\}^n$

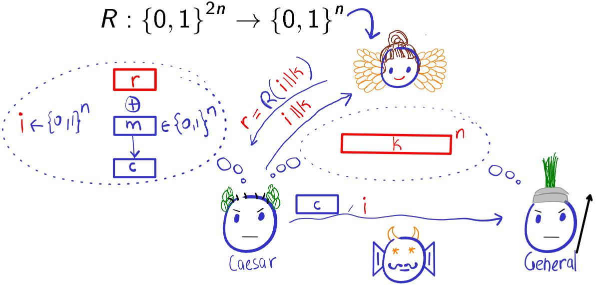


How will you construct a stateless encryption scheme given R?

Hint: R helps generate *exponentially-many* random OTPs

Let's Encrypt Many Messages Using an Oracle in the Sky

- Setting:
  - Caesar and his general have shared a key $k \in \{0,1\}^n$
  - Everyone (including Eve*) has access to a *random function oracle* $R : \{0,1\}^{2n} \to \{0,1\}^n$

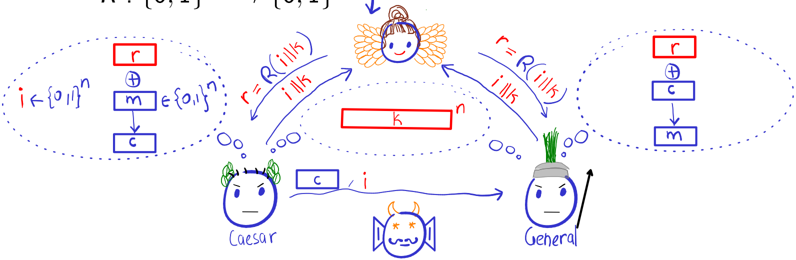


How will you construct a stateless encryption scheme given R?

Hint: R helps generate *exponentially-many* random OTPs

Let's Encrypt Many Messages Using an Oracle in the Sky

- Setting:
  - Caesar and his general have shared a key $k \in \{0,1\}^n$
  - Everyone (including Eve*) has access to a *random function oracle* $R : \{0,1\}^{2n} \to \{0,1\}^n$

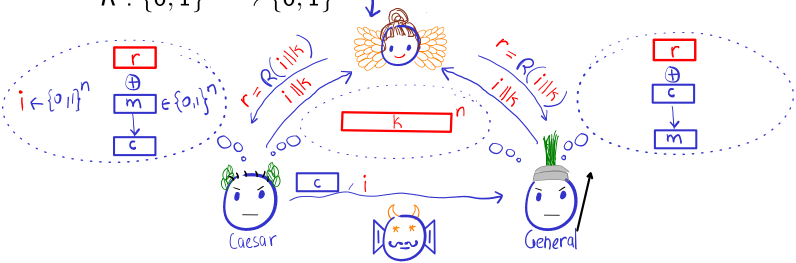


How will you construct a stateless encryption scheme given R?

Hint: R helps generate *exponentially-many* random OTPs

Let's Encrypt Many Messages Using an Oracle in the Sky

- Setting:
  - Caesar and his general have shared a key $k \in \{0,1\}^n$
  - Everyone (including Eve*) has access to a *random function oracle* $R : \{0,1\}^{2n} \to \{0,1\}^n$



How will you construct a stateless encryption scheme given R?

Hint: R helps generate *exponentially-many* random OTPs

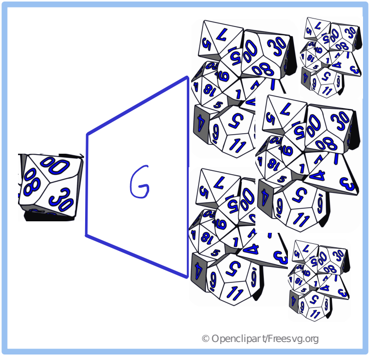
Exercise 2

What if Caesar and his general did not have the shared key k? Can they still do something given the oracle in the sky?

Plan for Today's Lecture

- Task: secure communication of *multiple messages* with shared keys
- Threat model: computational secrecy against eavesdroppers

Pseudo-Random Function (PRF)



GGM PRF

PRF: Computational Analogue of Oracle in the Sky

- A function F that "seems like" a random function oracle to PPT distinguishers

PRF: Computational Analogue of Oracle in the Sky

- A function F that "seems like" a random function oracle to PPT distinguishers
- More formally:
  - F_k sampled at random from a (smallish) family of functions

 $\approx \left\{F_k : \{0,1\}^n \to \{0,1\}^n\right\}_{k\in\{0,1\}^n}$

  - A random function, sampled from the set of *all* functions $\mathcal{F}_n$

$\{f : \{0,1\}^n \to \{0,1\}^n\}$

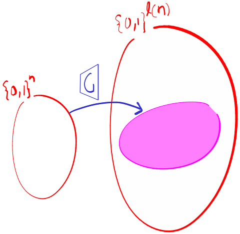
PRF: Computational Analogue of Oracle in the Sky

- A function F that “seems like” a random function oracle to PPT distinguishers
- More formally:
  - F_k sampled at random from a (smallish) family of functions $\{F_k : \{0,1\}^n \to \{0,1\}^n\}_{k\in\{0,1\}^n}$

$\approx$

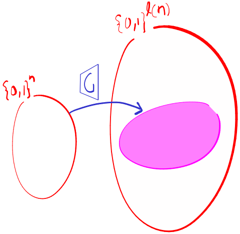
  - A random function, sampled from the set of *all* functions $\mathcal{F}_n$

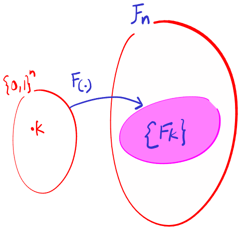
$\{f : \{0,1\}^n \to \{0,1\}^n\}$



- A function F that "seems like" a random function oracle to PPT distinguishers
- More formally:
  - F_k sampled at random from a (smallish) family of functions
    $\approx \left\{F_k : \{0,1\}^n \to \{0,1\}^n\right\}_{k\in\{0,1\}^n}$
  - A random function, sampled from the set of *all* functions $\mathcal{F}_n$

$\{f : \{0,1\}^n \to \{0,1\}^n\}$





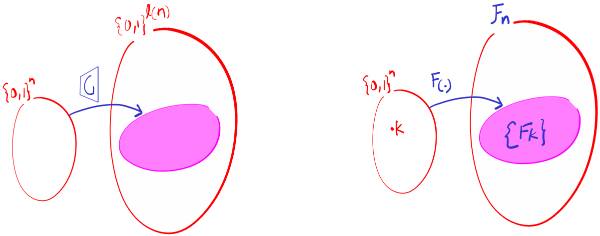
PRF: Computational Analogue of Oracle in the Sky

- A function F that "seems like" a random function oracle to PPT distinguishers
- More formally:
  - F_k sampled at random from a (smallish) family of functions

$\approx \left\{F_k : \{0,1\}^n \to \{0,1\}^n\right\}_{k \in \{0,1\}^n}$

  - A random function, sampled from the set of *all* functions $\mathcal{F}_n$

$\{f : \{0,1\}^n \to \{0,1\}^n\}$



Number of functions in $\{F_k\}$ vs. number of functions $\mathcal{F}_n$?

- Why is it still useful?
  - Helps generate exponentially-many *pseudorandom* OTPs

How Exactly to Define Pseudorandomess for Functions?...

- A function F that "seems like" random function oracle to PPT distinguishers

How Exactly to Define Pseudorandomess for Functions? ...

- A function F that "seems like" random function oracle to PPT distinguishers
- Recall how we defined pseudorandomness for PRG (Lecture 05)

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.

How Exactly to Define Pseudorandomess for Functions?

- A function F that "seems like" random function oracle to PPT distinguishers
- Recall how we defined pseudorandomness for PRG (Lecture 05)

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.

How Exactly to Define Pseudorandomess for Functions?...

- A function F that "seems like" random function oracle to PPT distinguishers
- Recall how we defined pseudorandomness for PRG (Lecture 05)

> 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.*
>
> pseudorandom world random world

Can we give the distinguisher *full description* of the function (e.g., as a table)?

- No, then it becomes easy to distinguish
- How? (Recall: run-time measured w.r.to size of input)

How Exactly to Define Pseudorandomess for Functions?

- A function F that "seems like" random function oracle to PPT distinguishers
- Recall how we defined pseudorandomness for PRG (Lecture 05)

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.

pseudorandom world

random world

- Can we give the distinguisher *full description* of the function (e.g., as a table)?
  - No, then it becomes easy to distinguish
  - How? (Recall: run-time measured w.r.to size of input)
- Way around:
  - Distinguisher given *oracle* access to the functions
  - One query=one unit of running time $\rightarrow$ efficient PPT distinguisher can only make polynomially-many queries

How Exactly to Define Pseudorandomess for Functions?...

Definition 1 (Two worlds)

A family of functions $\{F_k : \{0,1\}^n \to \{0,1\}^n\}_{k \in \{0,1\}^n}$ is a PRF if for every PPT *oracle distinguisher* D

$$\delta(n) := \left| \Pr_{k \leftarrow \{0,1\}^n}[\mathsf{D}^{F_k(\cdot)}(1^n) = 0] - \Pr_{f \leftarrow \mathcal{F}_n}[\mathsf{D}^{f(\cdot)}(1^n) = 0] \right|$$

is negligible.

How Exactly to Define Pseudorandomess for Functions?...

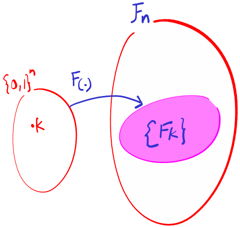
Definition 1 (Two worlds)

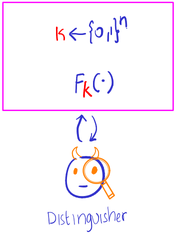
A family of functions $\{F_k : \{0,1\}^n \to \{0,1\}^n\}_{k\in\{0,1\}^n}$ is a PRF if for every PPT *oracle distinguisher* D

$$\delta(n) := \left| \Pr_{k\leftarrow\{0,1\}^n}[D^{F_k(\cdot)}(1^n) = 0] - \Pr_{f\leftarrow\mathcal{F}_n}[D^{f(\cdot)}(1^n) = 0] \right|$$

pseudorandom world

is negligible.





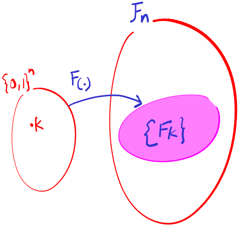
Definition 1 (Two worlds)

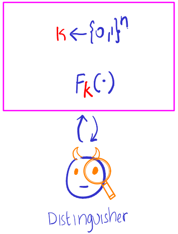
A family of functions $\{F_k : \{0,1\}^n \to \{0,1\}^n\}_{k \in \{0,1\}^n}$ is a PRF if for every PPT *oracle distinguisher* D

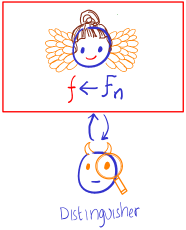
$$\delta(n) := \left| \Pr_{k \leftarrow \{0,1\}^n}[D^{F_k(\cdot)}(1^n) = 0] - \Pr_{f \leftarrow \mathcal{F}_n}[D^{f(\cdot)}(1^n) = 0] \right|$$

pseudorandom world   random world

is negligible.







Let's Check if You Understood Definition 1

PRF or not? Below $F^{(1)}$ and $F^{(2)}$ are PRFs

1. $F_k(x) := k \oplus x$
2. $F_{k_1 \| k_2}(x) := F^{(1)}_{k_1}(x) \| F^{(2)}_{k_2}(x)$
3. $F_k(x_1 \| x_2) := F^{(1)}_k(x_1) \| F^{(2)}_k(x_2)$

Let's Check if You Understood Definition 1

PRF or not? Below $F^{(1)}$ and $F^{(2)}$ are PRFs

1. $F_k(x) := k \oplus x$
2. $F_{k_1 \| k_2}(x) := F^{(1)}_{k_1}(x) \| F^{(2)}_{k_2}(x)$
3. $F_k(x_1 \| x_2) := F^{(1)}_{k}(x_1) \| F^{(2)}_{k}(x_2)$

PRG or not? Below, F is a PRF

1. $\mathrm{G}(s) := F_s(1) \| F_s(2) \| \cdots \| F_s(n-1) \| F_s(n)$
2. $\mathrm{G}(s) := F_s(2^0) \| F_s(2^1) \cdots \| F_s(2^{n-1}) \| F_s(2^n)$
3. $\mathrm{G}(s) := F_1(s) \| F_2(s) \| \cdots \| F_{n-1}(s) \| F_n(s)$

Let's Check if You Understood Definition 1

PRF or not? Below $F^{(1)}$ and $F^{(2)}$ are PRFs

1. $F_k(x) := k \oplus x$
2. $F_{k_1 \| k_2}(x) := F^{(1)}_{k_1}(x) \| F^{(2)}_{k_2}(x)$
3. $F_k(x_1 \| x_2) := F^{(1)}_k(x_1) \| F^{(2)}_k(x_2)$

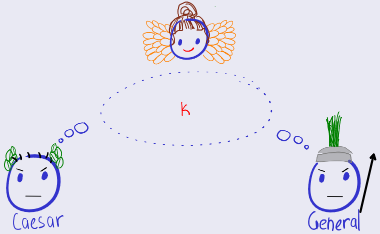
PRG or not? Below, F is a PRF

1. $\mathsf{G}(s) := F_s(1) \| F_s(2) \| \cdots \| F_s(n-1) \| F_s(n)$
2. $\mathsf{G}(s) := F_s(2^0) \| F_s(2^1) \cdots \| F_s(2^{n-1}) \| F_s(2^n)$
3. $\mathsf{G}(s) := F_1(s) \| F_2(s) \| \cdots \| F_{n-1}(s) \| F_n(s)$

Exercise 3

In all the "yes" cases above, formally prove; in all the "no" cases, describe a counter-example.

Construction 1 (Replace random oracle with PRF)



(Stateless) Symmetric-Key Encryption from PRF

Construction 1 (Replace random oracle with PRF)

(Stateless) Symmetric-Key Encryption from PRF

Construction 1 (Replace random oracle with PRF)

(Stateless) Symmetric-Key Encryption from PRF

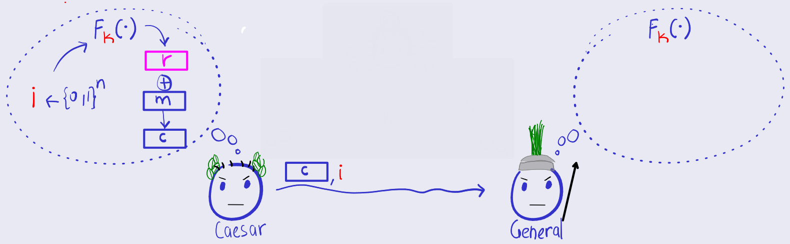
Construction 1 (Replace random oracle with PRF)

(Stateless) Symmetric-Key Encryption from PRF

Construction 1 (Replace random oracle with PRF)

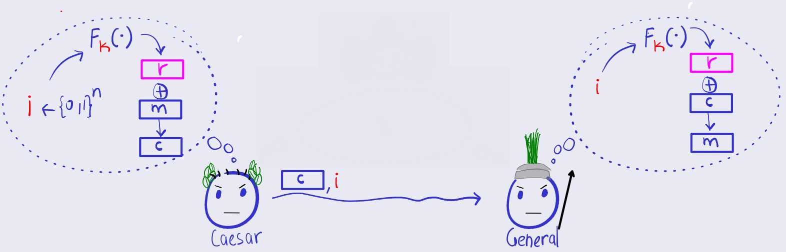
(Stateless) Symmetric-Key Encryption from PRF

Construction 1 (Replace random oracle with PRF)



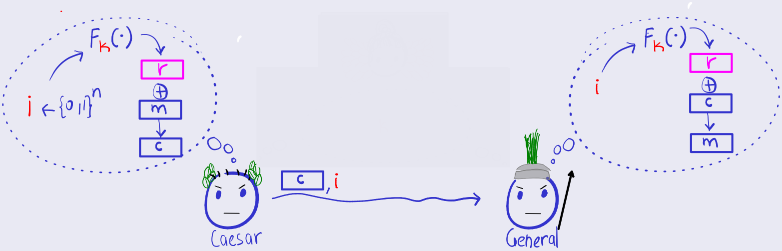
(Stateless) Symmetric-Key Encryption from PRF

Construction 1 (Replace random oracle with PRF)



(Stateless) Symmetric-Key Encryption from PRF

Construction 1 (Replace random oracle with PRF)



Note: encryption is randomised and thus length of ciphertext is longer than plaintext (first such scheme in this course)

Exercise 4 (Hint: reduction similar to computational OTP)

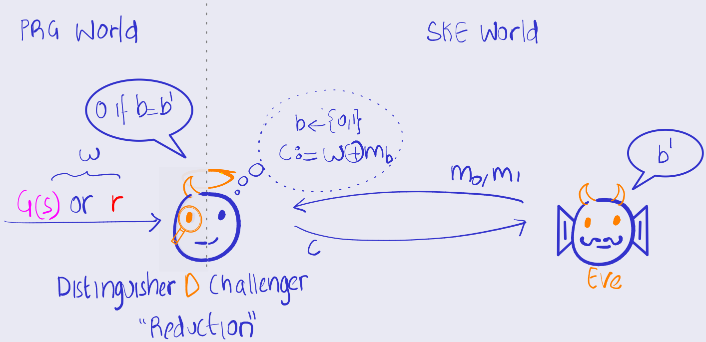
1. Formulate the eavesdropper threat model for multiple encryptions
2. Prove that Construction 1 is secure against eavesdroppers

Hint: Reduction Similar to Computational OTP...

Theorem 2 (Recall, Lecture 05-06)

If G is a PRG, then Comp. OTP is comp. secret against eavesdroppers

Proof by reduction. $\exists$D for G $\Leftarrow$ $\exists$Eve breaking Computational OTP.



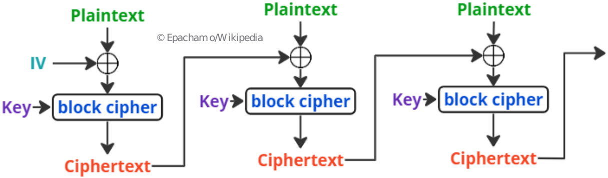
□

PRFs IRL

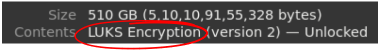
- Coming up: theoretical construction, but inefficient for practice
- Practical PRFs: block ciphers like AES
  - Usually only support certain key-sizes (128, 192, 256)
  - Supported by most libraries (e.g., OpenSSL, NaCl) and even implemented on modern processors (AES-NI)

PRFs IRL

- Coming up: theoretical construction, but inefficient for practice
- Practical PRFs: block ciphers like AES
  - Usually only support certain key-sizes (128, 192, 256)
  - Supported by most libraries (e.g., OpenSSL, NaCl) and even implemented on modern processors (AES-NI)
- For encrypting larger messages (e.g., for disk encryption) “modes of operation” used (Coming up in Lecture 08!)
  - E.g: Cipher block-chaining (CBC) mode



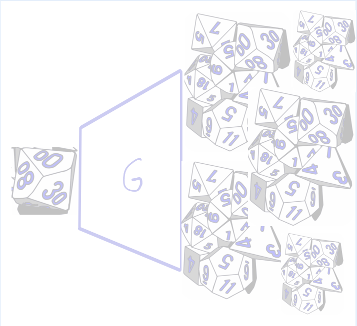
  - My laptop uses LUKS for disk encryption, which uses AES-XTS



Plan for Today's Lecture

- Task: secure communication of *multiple messages* with shared keys
- Threat model: computational secrecy against eavesdroppers

NEW Pseudo-Random Function (PRF)



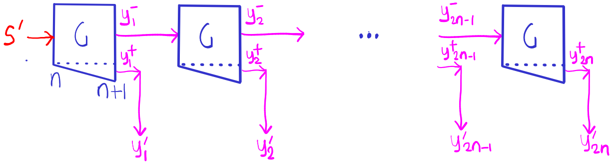
© Openclipart/Freesvg.org

NEW GGM PRF

© Biodiversity Heritage Library

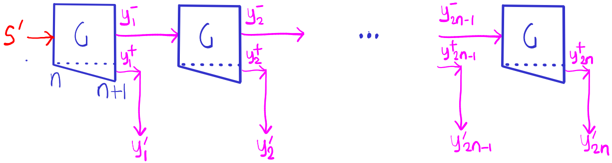
© Oded Goldreich

Let's Try to Construct a PRF



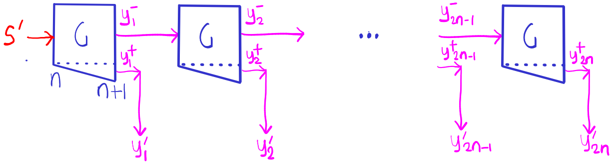
- Recall construction of length-extending PRG from last lecture
- Recall the problem with expanding exponentially:
  - Takes exponential time to access most pseudorandom OTPs

Let's Try to Construct a PRF



- Recall construction of length-extending PRG from last lecture
- Recall the problem with expanding exponentially:
  - Takes exponential time to access most pseudorandom OTPs
- Need "PRG" with
  1. Exponential stretch
  2. Output bits "efficiently" accessible (also called locality)
- How to reconcile the two requirements?
  - Hint: Use length-doubling PRG

Let's Try to Construct a PRF



- Recall construction of length-extending PRG from last lecture
- Recall the problem with expanding exponentially:
  - Takes exponential time to access most pseudorandom OTPs

Need "PRG" with

1. Exponential stretch
2. Output bits "efficiently" accessible (also called locality)

How to reconcile the two requirements?

- Hint: Use length-doubling PRG
- Use binary tree instead of chain!

Tree-Based Construction from Length-Doubling PRG G

$n \to n^2$

Construction 2 (GGM PRF $\{F_k : \{0,1\}^n \to \{0,1\}^n\}_{k \in \{0,1\}^n}$)

Tree-Based Construction from Length-Doubling PRG G

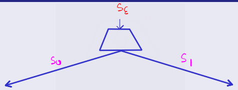
$n \to n^2$

Construction 2 (GGM PRF $\{F_k : \{0,1\}^n \to \{0,1\}^n\}_{k\in\{0,1\}^n}$)

Tree-Based Construction from Length-Doubling PRG G

$n \to n^2$

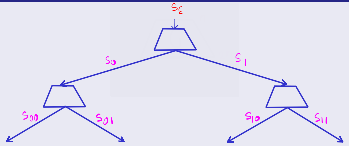
Construction 2 (GGM PRF $\{F_k : \{0,1\}^n \to \{0,1\}^n\}_{k \in \{0,1\}^n}$)



Tree-Based Construction from Length-Doubling PRG G

$n \to n^2$

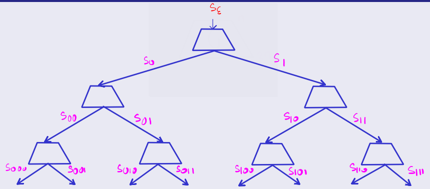
Construction 2 (GGM PRF $\{F_k : \{0,1\}^n \to \{0,1\}^n\}_{k \in \{0,1\}^n}$)



Tree-Based Construction from Length-Doubling PRG G

$n \to n^2$

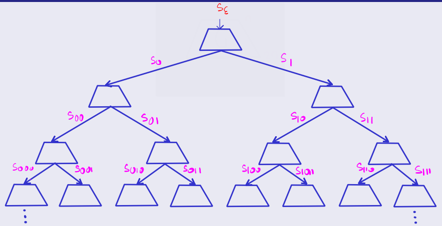
Construction 2 (GGM PRF $\{F_k : \{0,1\}^n \to \{0,1\}^n\}_{k \in \{0,1\}^n}$)



Tree-Based Construction from Length-Doubling PRG G

$n \to n^2$

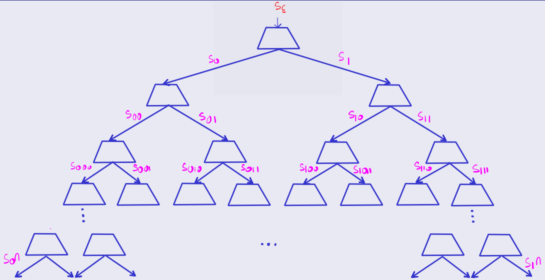
Construction 2 (GGM PRF $\{F_k : \{0,1\}^n \to \{0,1\}^n\}_{k \in \{0,1\}^n}$)



Tree-Based Construction from Length-Doubling PRG G

$n \to n^2$

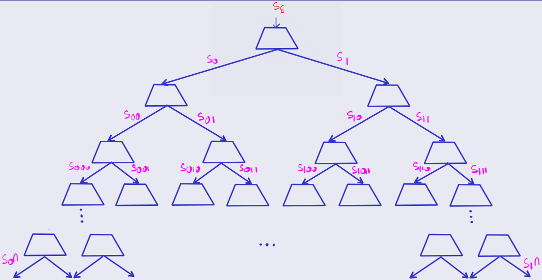
Construction 2 (GGM PRF $\{F_k : \{0,1\}^n \to \{0,1\}^n\}_{k \in \{0,1\}^n}$)



Tree-Based Construction from Length-Doubling PRG G

$n \to n^2$

Construction 2 (GGM PRF $\{F_k : \{0,1\}^n \to \{0,1\}^n\}_{k\in\{0,1\}^n}$)

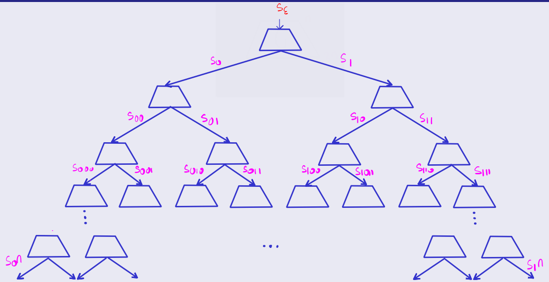


- Define $F_k(x) =$

Tree-Based Construction from Length-Doubling PRG G

$n \to n^2$

Construction 2 (GGM PRF $\{F_k : \{0,1\}^n \to \{0,1\}^n\}_{k\in\{0,1\}^n}$)



- Define $F_k(x) = s_x$ with $s_\varepsilon := k$

Exercise 5

1. Write down the construction formally.
2. What if we use d-ary tree instead of binary tree?

How do We Prove that Construction 2 is a PRF?...

Theorem 3

If G is a length-doubling PRG, then Construction 2 is a PRF.

Proof. First attempt: off-the-shelf hybrid argument.

Strategy: replace, breadth-first, pseudorandom by random

How do We Prove that Construction 2 is a PRF? . . .

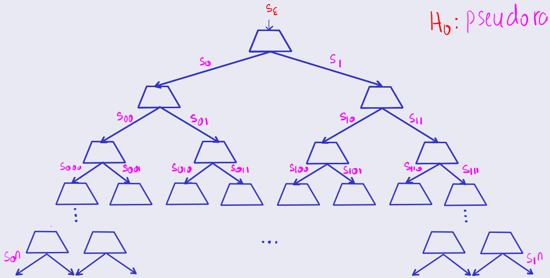
Theorem 3

If G is a length-doubling PRG, then Construction 2 is a PRF.

Proof. First attempt: off-the-shelf hybrid argument.

Strategy: replace, breadth-first, pseudorandom by random

H_0: pseudorandom world



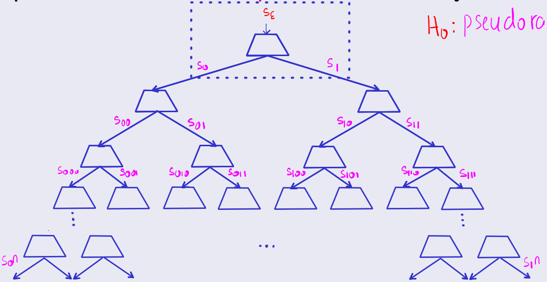
How do We Prove that Construction 2 is a PRF?

Theorem 3

If G is a length-doubling PRG, then Construction 2 is a PRF.

Proof. First attempt: off-the-shelf hybrid argument.

Strategy: replace, breadth-first, pseudorandom by random



H_0: pseudorandom world

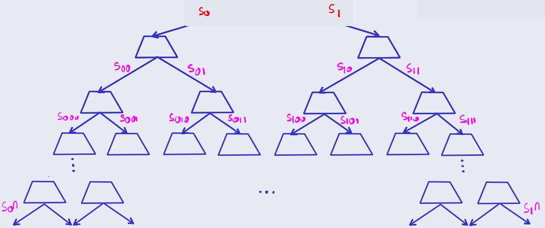
Theorem 3

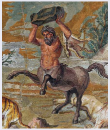
If G is a length-doubling PRG, then Construction 2 is a PRF.

Proof. First attempt: off-the-shelf hybrid argument.

Strategy: replace, breadth-first, pseudorandom by random

H_1: hybrid world





©Wikipedia

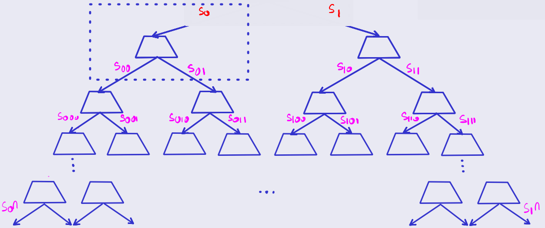
Theorem 3

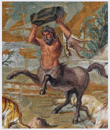
If G is a length-doubling PRG, then Construction 2 is a PRF.

Proof. First attempt: off-the-shelf hybrid argument.

Strategy: replace, breadth-first, pseudorandom by random

H_1 : hybrid world





©Wikipedia

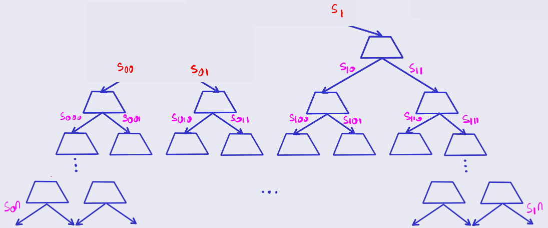
Theorem 3

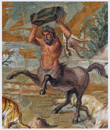
If G is a length-doubling PRG, then Construction 2 is a PRF.

Proof. First attempt: off-the-shelf hybrid argument.

Strategy: replace, breadth-first, pseudorandom by random

H_2: hybrid world





©Wikipedia

How do We Prove that Construction 2 is a PRF?...

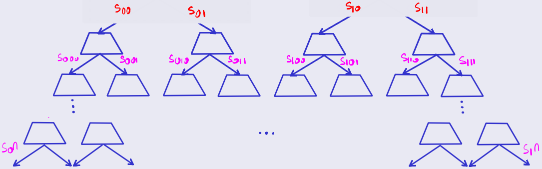
Theorem 3

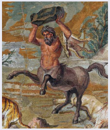
If G is a length-doubling PRG, then Construction 2 is a PRF.

Proof. First attempt: off-the-shelf hybrid argument.

Strategy: replace, breadth-first, pseudorandom by random

H_3: hybrid world





©Wikipedia

How do We Prove that Construction 2 is a PRF?...

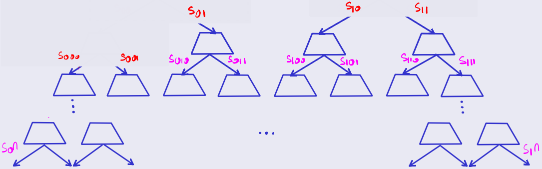
Theorem 3

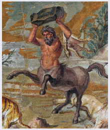
If G is a length-doubling PRG, then Construction 2 is a PRF.

Proof. First attempt: off-the-shelf hybrid argument.

Strategy: replace, breadth-first, pseudorandom by random

H_3: hybrid world




©Wikipedia

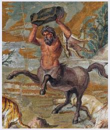
How do We Prove that Construction 2 is a PRF?...

Theorem 3

If G is a length-doubling PRG, then Construction 2 is a PRF.

Proof. First attempt: off-the-shelf hybrid argument.

Strategy: replace, breadth-first, pseudorandom by random



©Wikipedia

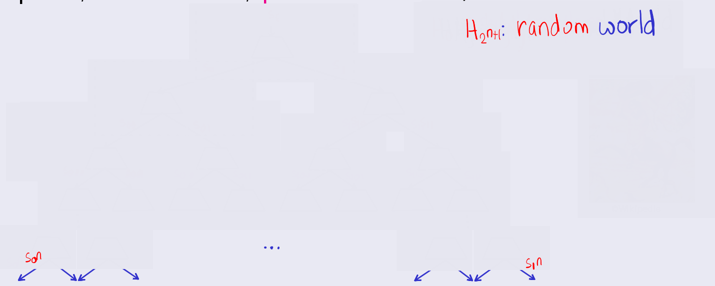
How do We Prove that Construction 2 is a PRF?...

Theorem 3

If G is a length-doubling PRG, then Construction 2 is a PRF.

Proof. First attempt: off-the-shelf hybrid argument.

Strategy: replace, breadth-first, pseudorandom by random



How do We Prove that Construction 2 is a PRF?

Theorem 3

If G is a length-doubling PRG, then Construction 2 is a PRF.

Proof. First attempt: off-the-shelf hybrid argument.

Strategy: replace, breadth-first, pseudorandom by random

Hybrid argument: if D can distinguish H_0 from $H_{2^{n+1}}$

$\Downarrow$

$\exists i \in [0, 2^{n+1}-1]$ such that D distinguishes H_i from H_{i+1}

How do We Prove that Construction 2 is a PRF?...

Theorem 3

If G is a length-doubling PRG, then Construction 2 is a PRF.

Proof. First attempt: off-the-shelf hybrid argument.

Strategy: replace, breadth-first, pseudorandom by random

Hybrid argument: if D can distinguish H_0 from $H_{2^{n+1}}$ w/ pr. δ

$$\Downarrow$$

$\exists i \in [0, 2^{n+1}-1]$ such that D distinguishes H_i from H_{i+1} w/ pr. $\delta/2^{n+1}$

How do We Prove that Construction 2 is a PRF?

Theorem 3

If G is a length-doubling PRG, then Construction 2 is a PRF.

Proof. First attempt: off-the-shelf hybrid argument.

Strategy: replace, breadth-first, pseudorandom by random

Hybrid argument: if D can distinguish H_0 from $H_{2^{n+1}}$ w/ pr. δ

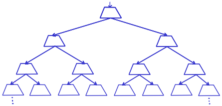
$\Downarrow$

$\exists i \in [0, 2^{n+1}-1]$ such that D distinguishes H_i from H_{i+1} w/ pr. $\delta / 2^{n+1}$

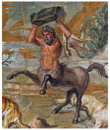
Problem: exponential number of hybrids

Solution: hybrid argument with on-the-fly/lazy sampling!

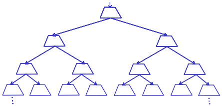
Recap/Next Lecture

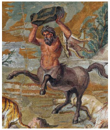


- Defined and constructed PRFs
  - GGM tree-based construction from length-doubling PRGs
  - Another application of hybrid argument



©Wikipedia



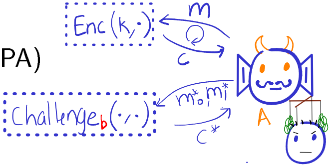


©Wikipedia

- Defined and constructed PRFs
  - GGM tree-based construction from length-doubling PRGs
  - Another application of hybrid argument

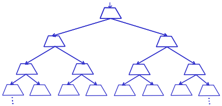
©Wikipedia

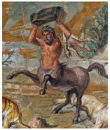
- Constructed a *stateless* SKE from PRF
  - Next lecture: chosen-plaintext attack (CPA)



Recap/Next Lecture

- Defined and constructed PRFs
  - GGM tree-based construction from length-doubling PRGs
  - Another application of hybrid argument

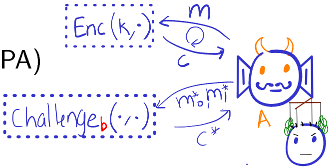




©Wikipedia

©Wikipedia

- Constructed a *stateless* SKE from PRF
  - Next lecture: chosen-plaintext attack (CPA)



- Other applications of PRFs
  - Authentication (coming up: Lecture 09)
  - Natural proofs: barrier to resolving the P $\stackrel{?}{=}$ NP question

Further Reading

1. PRFs were introduced in [GGM84], where the namesake construction from PRGs was also presented.
2. [Gol01, §3.6] for a formal proof of Theorem 3
3. [KL14, §3.5] for a formal description of Construction 1.
4. To read more about natural proofs, and the role of PRFs there [Aar03, §4] or [Cho11] are good sources.

Scott Aaronson.
Is P versus NP formally independent?
Bull. EATCS, 81:109–136, 2003.

Timothy Y Chow.
What is... a natural proof?
Notices of the AMS, 58(11):1586–1587, 2011.

Oded Goldreich, Shafi Goldwasser, and Silvio Micali.
How to construct random functions (extended abstract).
In *25th FOCS*, pages 464–479. IEEE Computer Society Press, October 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.