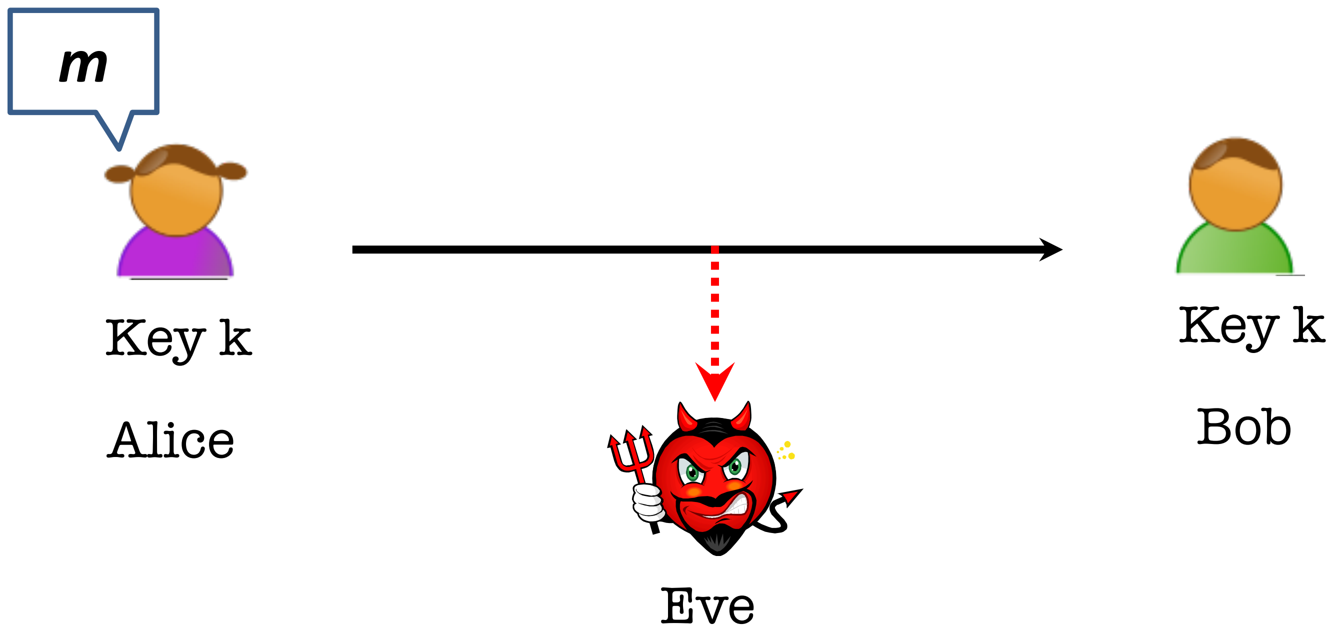
MIT 6.875 & Berkeley CS276

Foundations of Cryptography
Lecture 2

Administrivia

- Piazza Time-zone Survey & Office hours
- PS1 Released, due Sept 15

The Secure Communication Problem



- Alice and Bob have a common key k

- Algorithms (Gen, Enc, Dec)
- Correctness: Dec(k, Enc(k,m)) = m
- Security: No Eve learns anything about m.

How to Define Security

Perfect secrecy: A Posteriori = A Priori

For all m, c: $\Pr[M = m \mid E(K, M) = c] = \Pr[M = m]$

Perfect indistinguishability:

For all m_0, m_1, c: $\Pr[E(K, m_0) = c] = \Pr[E(K, m_1) = c]$

The two definitions are equivalent!

Is there a perfectly secure scheme?

- **One-time Pad**: $E(k,m) = k \oplus m$
- **However**: Keys are as long as Messages
- **WORSE, Shannon's theorem**:
  for **any** perfectly secure scheme, $|\text{key}| \geq |\text{message}|$.

Can we overcome Shannon's conundrum?

Let's first rewrite...

<u>Perfect indistinguishability</u>: *as a Turing test*

For all m_0, m_1, c: $\Pr[E(K, m_0) = c] = \Pr[E(K, m_1) = c]$

World 0:

$k \leftarrow \mathcal{K}$

$c = E(k, m_0)$

World 1:

$k \leftarrow \mathcal{K}$

$c = E(k, m_1)$

is a **distinguisher**.

For all EVE and all m_0, m_1: $\Pr[k \leftarrow \mathcal{K}; c = E(k, m_0)\colon EVE(c) = 0]$
$= \Pr[k \leftarrow \mathcal{K}; c = E(k, m_1)\colon EVE(c) = 0]$

Let's first rewrite…

Perfect indistinguishability: *as a Turing test*

For all m_0, m_1, c: $\Pr[E(K, m_0) = c] = \Pr[E(K, m_1) = c]$

World 0:

k ← $\mathcal{K}$

$c = E(k, m_0)$

World 1:

k ← $\mathcal{K}$

$c = E(k, m_1)$

is a **distinguisher**.

For all EVE and all m_0, m_1:

$$\Pr[k \leftarrow \mathcal{K}; b \leftarrow \{0,1\};\ c = E(k, m_b):\ EVE(c) = b] = 1/2$$

The Axiom of Modern Crypto

Feasible Computation = Probabilistic polynomial-time*

(**p.p.t.** = Probabilistic polynomial-time)

(polynomial in a security parameter n)

So, Alice and Bob are **fixed** p.p.t. algorithms.

(e.g., run in time n^2)

Eve is **any** p.p.t. algorithm.

(e.g., run in time n^4, or n^100, or n^10000,…)

* in recent years, quantum polynomial-time

Computational Indistinguishability (take 1)

World 0:

$k \leftarrow \mathcal{K}$

$c = E(k, m_0)$

World 1:

$k \leftarrow \mathcal{K}$

$c = E(k, m_1)$

 is a **p.p.t.** distinguisher.

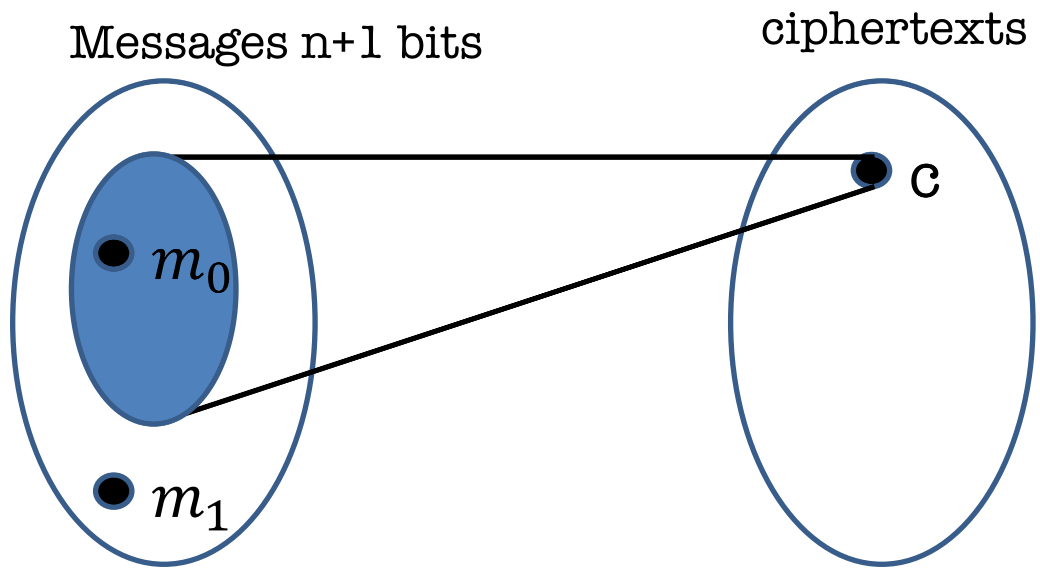
For all **p.p.t.** EVE and all m_0, m_1:

$$\Pr[k \leftarrow \mathcal{K}; b \leftarrow \{0,1\}; c = E(k, m_b): EVE(c) = b] = 1/2$$

Still subject to Shannon's impossibility!

Still subject to Shannon's impossibility!

Set of messages consistent with c = {D(k,c): all k}



Consider Eve that picks a random key k and

outputs 0 if D(k,c) = m_0 **w.p $\geq 1/2^n$**

outputs 1 if D(k,c) = m_1 **w.p = 0**

and a random bit if neither holds.

Bottomline: Pr[EVE succeeds] $\geq 1/2 + 1/2^n$

New Notion: Negligible Functions

Functions that grow slower than 1/p(n) for any polynomial p.

Definition: A function $\mu\colon \mathbb{N} \to \mathbb{R}$ is **negligible** if
for every polynomial function p,
there exists an n_0 s.t.
for all $n > n_0$:

$$\mu(n) < 1/p(n)$$

Key property: Events that occur with negligible probability look ***to poly-time algorithms*** like they ***never*** occur.

New Notion: Negligible Functions

Functions that grow slower than 1/p(n) for any polynomial p.

Definition: A function $\mu: \mathbb{N} \rightarrow \mathbb{R}$ is **negligible** if
for every polynomial function p,
there exists an n_0 s.t.
for all $n > n_0$:

$$\mu(n) < 1/p(n)$$

Question: Let $\mu(n) = 1/n^{\log n}$. Is μ negligible?

New Notion: Negligible Functions

Functions that grow slower than 1/p(n) for any polynomial p.

Definition: A function μ: $\mathbb{N} \rightarrow \mathbb{R}$ is **negligible** if
for every polynomial function p,
there exists an n_0 s.t.
for all $n > n_0$:

$$\mu(n) < 1/p(n)$$

Question: Let $\mu(n) = 1/n^{100}$ if n is prime and
$\mu(n) = 1/2^n$ otherwise. Is μ negligible?

Computational Indistinguishability

World 0:

k $\leftarrow \mathcal{K}$

$c = E(k, m_0)$

World 1:

k $\leftarrow \mathcal{K}$

$c = E(k, m_1)$

 is a distinguisher.

For all **p.p.t.** EVE, **there is a negligible function μ**

s.t. for all m_0, m_1:

$$\Pr[k \leftarrow \mathcal{K}; b \leftarrow \{0,1\}; c = E(k, m_b) : EVE(c) = b] \leq \frac{1}{2} + \mu(n)$$

Our First Crypto Tool:
Pseudorandom Generators (PRG)

PRG Definition

A function $G: \{0,1\}^n \rightarrow \{0,1\}^{n+1}$ is a pseudorandom generator if for no p.p.t. EVE can distinguish between $G(U_n)$ and U_{n+1}.

U_n= uniform distribution on n bits.

U_{n+1}= uniform distribution on n+1 bits.

PRG Definition

A function $G: \{0,1\}^n \rightarrow \{0,1\}^{n+1}$ is a **pseudorandom generator** if for for all p.p.t. EVE, there is a negligible function μ s.t.

$$|\Pr[y \leftarrow U_{n+1}: EVE(y) = 0] - \Pr[x \leftarrow U_n ; \mathrm{y} = \mathrm{G(x)}: \mathrm{EVE(y)} = 0]| \le \mu(\mathrm{n})$$

Question: What happens to this definition if EVE is unbounded?

PRG $\Longrightarrow$ Overcoming Shannon's Conundrum

(or, How to Encrypt n+1 bits using an n-bit key)

$Gen(1^n)$: Generate a random n-bit key k.

$Enc(k, m)$ where m is an **$(\boldsymbol{n}+\mathbf{1})$-bit** message:

Expand k into a (n+1)-bit pseudorandom string $\mathrm{k}' = G(\mathrm{k})$

One-time pad with k': ciphertext is $k' \oplus m$

$Dec(k, c)$ outputs $\mathrm{G}(k) \oplus c$

Correctness:

$Dec(k, c)$ outputs $\mathrm{G}(k) \oplus c = \mathrm{G}(k) \oplus G(k) \oplus \mathrm{m} = \mathrm{m}$

PRG $\Longrightarrow$ Overcoming Shannon's Conundrum

Security: by contradiction.

Suppose for contradiction that there is a p.p.t. EVE, a polynomial function p and m_0, m_1 $s.t.$

$$\Pr[k \leftarrow \mathcal{K}; b \leftarrow \{0,1\};\ c = E(k, m_b):\ EVE(c) = b] \geq \frac{1}{2} + 1/p(n)$$

PRG $\Longrightarrow$ Overcoming Shannon's Conundrum

Security: by contradiction.

Suppose for contradiction that there is a p.p.t. EVE, a polynomial function p and m_0, m_1 $s.t.$

$$\rho = \Pr[k \leftarrow \{0,1\}^n ; b \leftarrow \{0,1\}; c = G(k) \oplus m_b : EVE(c) = b] \geq \frac{1}{2} + 1/p(n)$$

Let $\rho' = \Pr[k' \leftarrow \{0,1\}^{n+1} ; b \leftarrow \{0,1\}; c = k' \oplus m_b : EVE(c) = b] = \frac{1}{2}$

This will give us a distinguisher EVE' for G, contradicting the assumption that G is a pseudorandom generator. QED.

PRG $\Longrightarrow$ Overcoming Shannon's Conundrum

Distinguisher EVE' for G.

Get as input a string y, run EVE(y$\oplus m_b$) for a random b, and let EVE's output be b'. Output "PRG" if b=b' and "RANDOM" otherwise.

$$\Pr[EVE' \, outputs \text{ "}PRG\text{" } | \, y \, is \, pseudorandom] \\ = \rho \geq \frac{1}{2} + 1/p(n)$$

$$\Pr[EVE' \, outputs \text{ "}PRG\text{" } | \, y \, is \, random] = \rho' = \frac{1}{2}$$

Therefore, $\Pr[EVE' \, outputs \text{ "}PRG\text{" } | \, y \, is \, pseudorandom] - \Pr[EVE' \, outputs \text{ "}PRG\text{" } | \, y \, is \, random] \geq 1/p(n)$ $\blacksquare$

PRG $\Longrightarrow$ Overcoming Shannon's Conundrum

(or, How to Encrypt n+1 bits using an n-bit key)

$\boldsymbol{Q1}$: Do PRGs exist?

(Exercise: If P=NP, PRGs do not exist.)

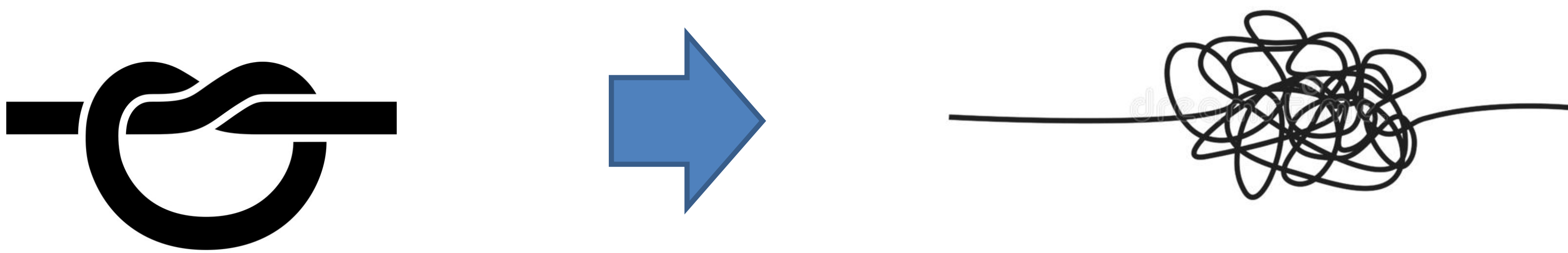
$\boldsymbol{Q2}$: How do we encrypt n^{100} message bits with n key bits?

(Length extension: If there is a PRG that stretches by one bit, there is one that stretches by polynomially many bits)

Constructing PRGs: Two Methodologies

<u>The Practical Methodology</u>

1. Start from a design framework
(e.g. “appropriately chosen functions composed appropriately many times look random”)

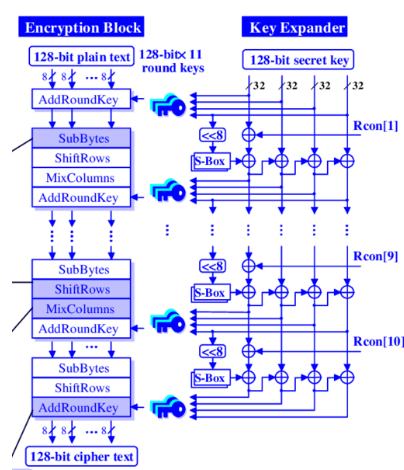


Constructing PRGs: Two Methodologies

<u>The Practical Methodology</u>

1. Start from a design framework
(e.g. "appropriately chosen functions composed appropriately many times look random")

2. Come up with a candidate construction



Rijndael
(now the Advanced Encryption Standard)

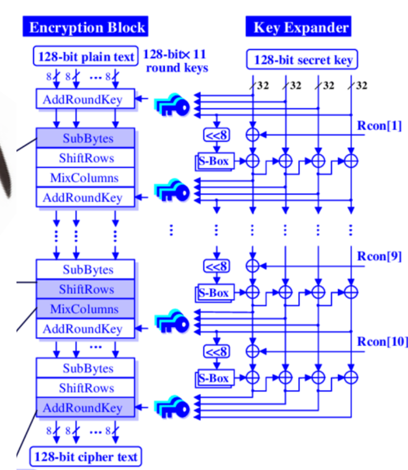
Constructing PRGs: Two Methodologies

The Practical Methodology

1. Start from a design framework
(e.g. “appropriately chosen functions composed appropriately many times look random”)

2. Come up with a candidate construction

3. Do extensive cryptanalysis.

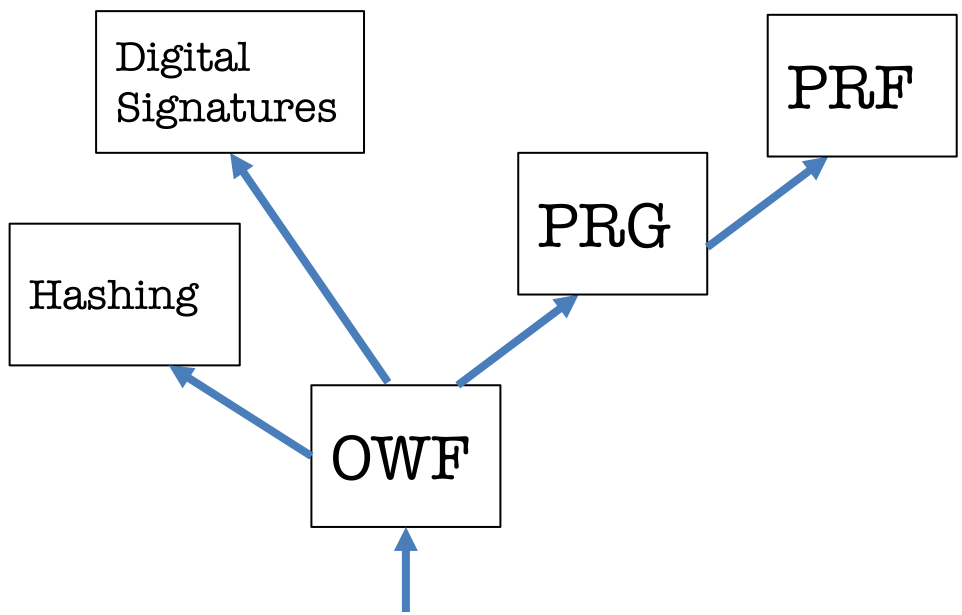


Constructing PRGs: Two Methodologies

The Foundational Methodology (much of this course)

Reduce to simpler primitives.

"Science wins either way" –Silvio Micali



well-studied, average-case hard, problems

Constructing PRGs: Two Methodologies

The Foundational Methodology (much of this course)

A PRG Candidate from the hardness of Subset-sum:

$$G(a_1, \ldots, a_n, x_1, \ldots, x_n) = (a_1, \ldots, a_n, \sum_{i=1}^{n} x_i a_i \bmod 2^{n+1})$$

where a_i are random (n+1)-bit numbers, and x_i are random bits.

Beautiful Function:

If G is a one-way function, then G is a PRG (Pset 1).

If lattice problems are hard on the worst-case, G is a PRG (6.876 Fall18 / CS294-168 Spring20)

PRG $\Longrightarrow$ Overcoming Shannon's Conundrum

(or, How to Encrypt n+1 bits using an n-bit key)

$\boldsymbol{Q1}$**:** Do PRGs exist?

(Exercise: If P=NP, PRGs do not exist.)

$\boldsymbol{Q2}$**:** How do we encrypt n^{100} message bits with n key bits?

(Length extension: If there is a PRG that stretches by one bit, there is one that stretches by polynomially many bits)

Length extension: One bit to Many bits

Let G: $\{0,1\}^n \rightarrow \{0,1\}^{n+1}$ be a pseudorandom generator.

Goal: use G to generate **many** pseudorandom bits.

Length extension: One bit to Many bits

Let G: $\{0,1\}^n \to \{0,1\}^{n+1}$ be a pseudorandom generator.

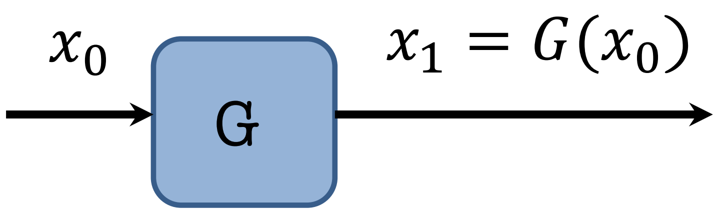
Goal: use G to generate **poly many** pseudorandom bits.

Length extension: One bit to Many bits

Let G: $\{0,1\}^n \rightarrow \{0,1\}^{n+1}$ be a pseudorandom generator.

Goal: use G to generate **poly many** pseudorandom bits.

Construction of G'(x_0)

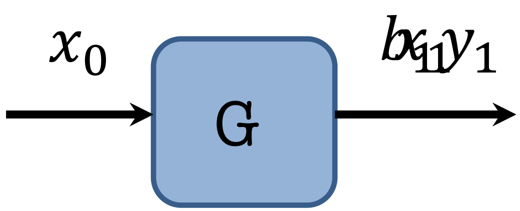


Length extension: One bit to Many bits

Let G: $\{0,1\}^n \rightarrow \{0,1\}^{n+1}$ be a pseudorandom generator.

Goal: use G to generate **poly many** pseudorandom bits.

<u>Construction of G'(x_0)</u>

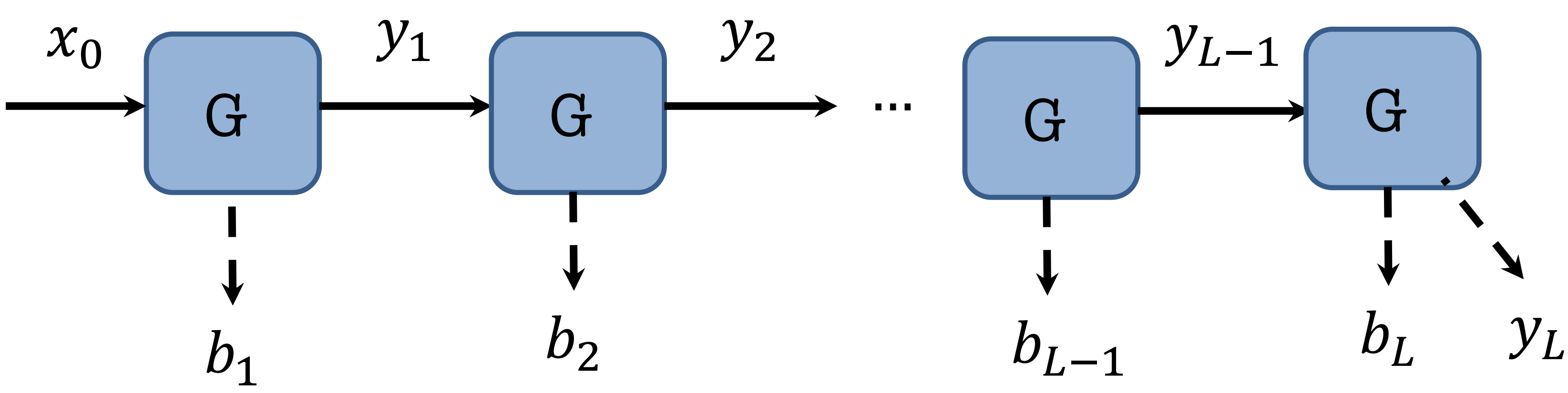


Length extension: One bit to Many bits

Let G: $\{0,1\}^n \rightarrow \{0,1\}^{n+1}$ be a pseudorandom generator.

Goal: use G to generate **poly many** pseudorandom bits.

Construction of G'(x_0) Output $b_1\ b_2\ b_3\ b_4\ b_5 \ldots y_L$.



Also called a stream cipher by the applied people.

Are we all set with encryption?

To encrypt the i-th bit, use the i-th pseudorandom bit.

Two problems:

1. Runtime (an efficiency issue)

2. Need to remember state (a security issue)

In a couple of weeks, Shafi will solve both problems in one shot.

Next Lecture:

Define one-way functions (OWF),

Hardcore bits (HCB),

Goldreich-Levin Theorem: every OWF has a HCB.

Show that OWF $\Rightarrow$ PRG
(how to construct a PRG from any OWF*)