CIS 5560

Cryptography
Lecture 3

Course website:

pratyushmishra.com/classes/cis-5560-s25/

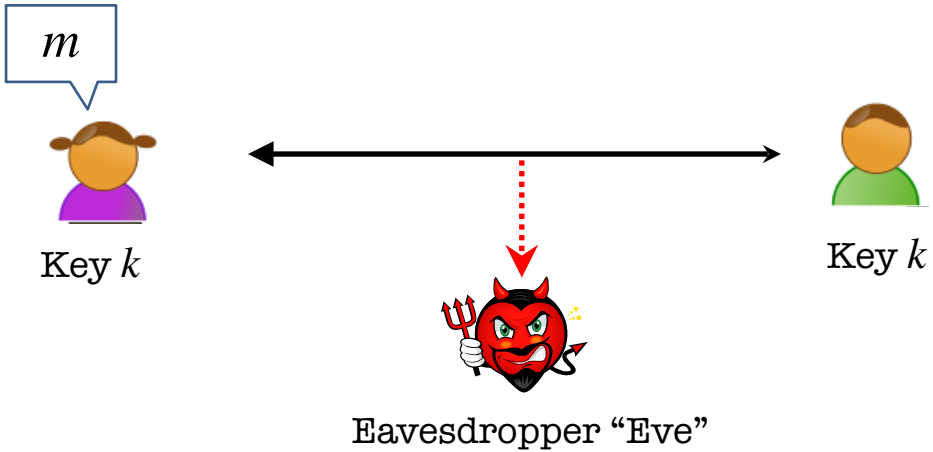
Slides adapted from Dan Boneh and Vinod Vaikuntanathan

Announcements

- **HW 0 is out;** due Friday, Jan 31 at 5PM on Gradescope
  - Covers modular arithmetic, basic probability, Caesar cipher
- Office Hours:
  - Pratyush: Friday 12-1PM

Recap of last lecture

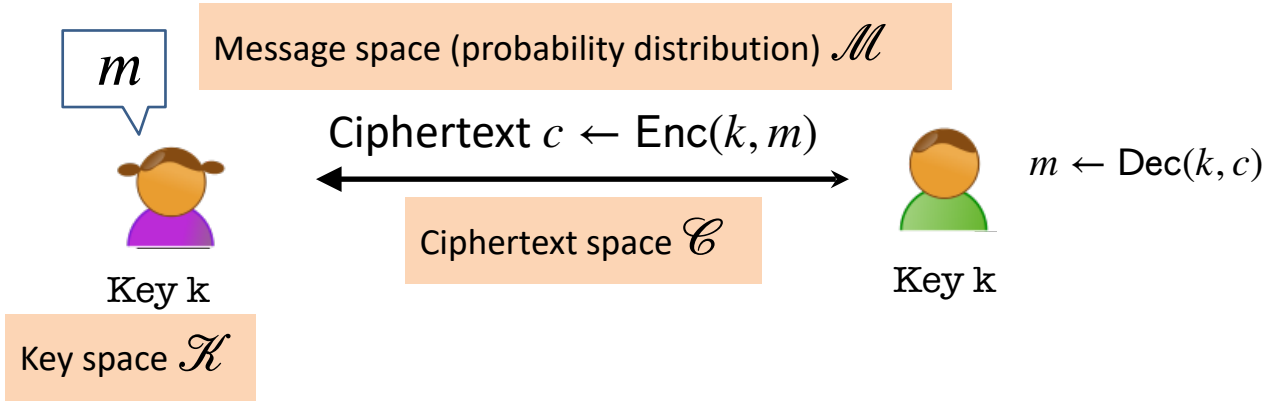
Secure Communication



Alice wants to send a message m to Bob without revealing it to Eve.

Key Notion: Secret-key Encryption

(or Symmetric-key Encryption)



Three (possibly randomized) polynomial-time algorithms:

- **Key Generation Algorithm:** $\mathrm{Gen}(1^k) \rightarrow k$
- **Encryption Algorithm:** $\mathrm{Enc}(k, m) \rightarrow c$
- **Decryption Algorithm:** $\mathrm{Dec}(k, c) \rightarrow m$

Perfect Secrecy is Achievable

The One-time Pad Construction:

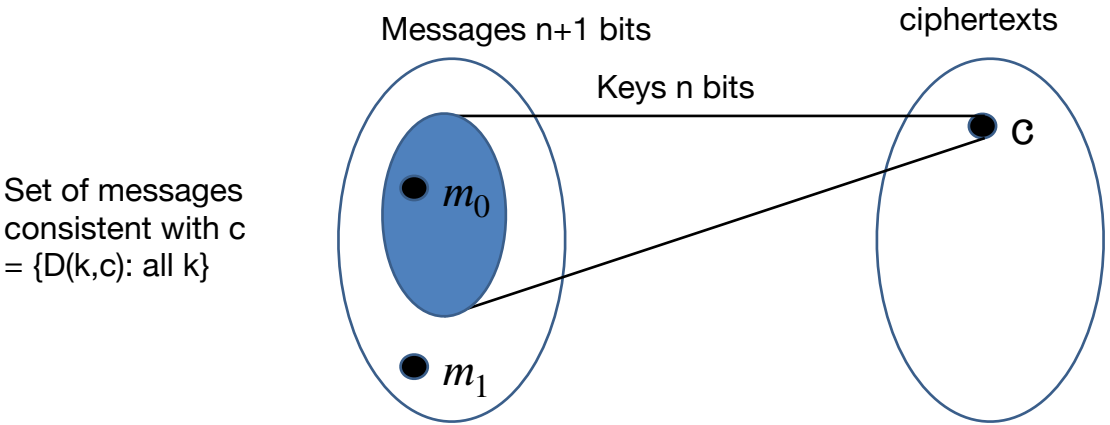
Gen: *Choose an n-bit string k at random, i.e.* $k \leftarrow \{0,1\}^n$

Enc(k, m) *with* $\mathcal{M} = \{0,1\}^n$*: Output* $c = m \oplus k$

Dec(k, c): *Output* $m = c \oplus k$

THEOREM: For any perfectly secure encryption scheme,

$$|\mathcal{K}| \geq |\mathcal{M}|$$



Each cipher text can correspond to at most 2^n messages, but message space contains 2^{n+1} possible messages!

So it is possible (and likely!) that a given cipher text can *never* decrypt to m_1!

$$\Pr[\text{Enc}(\mathcal{K}, m_1) = c] = 0$$

The Axiom of ~~Modern Crypto~~ Life

Feasible Computation = randomized polynomial-time* algorithms

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

(polynomial in a security parameter n)

Why Perfect Indistinguishability?

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

Why do we call it indistinguishability?

World 0:

$k \leftarrow \mathcal{K}$

$c = \text{Enc}(k, m_0)$

World 1:

$k \leftarrow \mathcal{K}$

$c = \text{Enc}(k, m_1)$

For all m_0, m_1, c : $\Pr[\text{world 0}] = \Pr[\text{world 1}]$

Ok, but why do we care? What does it matter whether we are in world 0 or world 1?

Perfect Indistinguishability from Eve's POV

Let's bring Eve into this definition.
It's not really important whether or not we are in world 0 or world 1, but rather whether *Eve* can tell whether we are in world 0 or world 1

World 0:

$k \leftarrow \mathcal{K}$

$c = \text{Enc}(k, m_0)$

World 1:

$k \leftarrow \mathcal{K}$

$c = \text{Enc}(k, m_1)$

Eve is an **all-powerful distinguisher**.
She needs to decide whether c came from World 0 or World 1.

For every Eve and all m_0, m_1,

$$\Pr\left[\text{Eve says that we are in world 0}\right] = \Pr\left[\text{Eve says that we are in world 1}\right]$$

Perfect Indistinguishability from Eve's POV

Let's formalize what it means for Eve to guess correctly:

World 0:

$k \leftarrow \mathcal{K}$

$c = \mathsf{Enc}(k, m_0)$

World 1:

$k \leftarrow \mathcal{K}$

$c = \mathsf{Enc}(k, m_1)$

Eve is an **all-powerful distinguisher**.

She needs to decide whether c came from World 0 or World 1.

For every Eve and all m_0, m_1,

$$\Pr\left[\mathsf{Eve}(c) = 0 \,\middle|\, \begin{matrix} k \leftarrow \mathcal{K} \\ c = \mathsf{Enc}(k, m_0) \end{matrix}\right] = \Pr\left[\mathsf{Eve}(c) = 1 \,\middle|\, \begin{matrix} k \leftarrow \mathcal{K} \\ c = \mathsf{Enc}(k, m_1) \end{matrix}\right]$$

Perfect Indistinguishability from Eve's POV

Equivalently,

World 0:

$k \leftarrow \mathcal{K}$

$c = \mathsf{Enc}(k, m_0)$

World 1:

$k \leftarrow \mathcal{K}$

$c = \mathsf{Enc}(k, m_1)$

Eve is an **all-powerful distinguisher**.

She needs to decide whether c came from World 0 or World

For every Eve and all m_0, m_1,

$$\left| \Pr\left[\mathsf{Eve}(c) = 0 \,\middle|\, \begin{matrix} k \leftarrow \mathcal{K} \\ c = \mathsf{Enc}(k, m_0) \end{matrix}\right] - \Pr\left[\mathsf{Eve}(c) = 1 \,\middle|\, \begin{matrix} k \leftarrow \mathcal{K} \\ c = \mathsf{Enc}(k, m_1) \end{matrix}\right] \right| = 0$$

Called adversary's "advantage"

Perfect Indistinguishability from Eve's POV, Take 2

We can rewrite this into an equivalent form with just one probability. Essentially, if Eve can't distinguish between either world, it means that she is right half the time, and wrong half the time.

World 0:

$k \leftarrow \mathcal{K}$

$c = \mathsf{Enc}(k, m_0)$

World 1:

$k \leftarrow \mathcal{K}$

$c = \mathsf{Enc}(k, m_1)$

Eve is an **all-powerful distinguisher**.

She needs to decide whether c came from World 0 or World 1.

For every Eve and m_0, m_1, $\Pr\left[\mathsf{Eve}(c) = b \;\middle|\; \begin{array}{r} k \leftarrow \mathcal{K} \\ b \leftarrow \{0,1\} \\ c = \mathsf{Enc}(k, m_b) \end{array}\right] = \frac{1}{2}$

So what can we do with this framing?

The Key Idea:

Computationally Bounded Adversaries

The Axiom of ~~Modern Crypto~~ Life

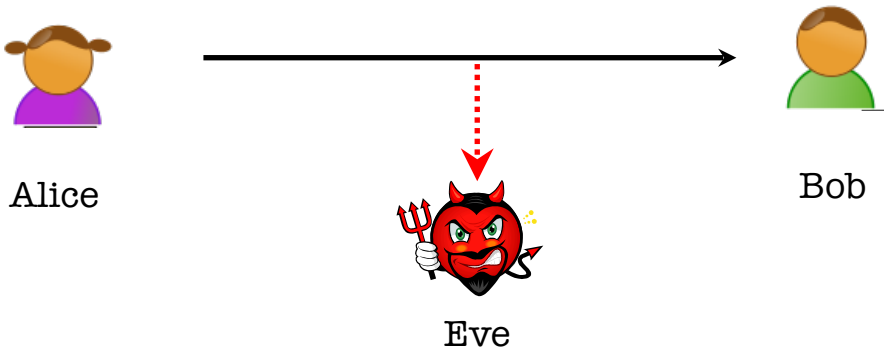
Feasible Computation = randomized polynomial-time* algorithms

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

(polynomial in a security parameter n)

* in recent years, quantum polynomial-time

Secure Communication



Running time of Alice and Bob?

Fixed p.p.t. (e.g., run in time $O(n^2)$)

Running time of Eve?

Arbitrary p.p.t. (e.g., run in time $O(n^2)$ or $O(n^4)$ or $O(n^{1000})$)

Computational Indistinguishability (take 1)

World 0:

$k \leftarrow \mathcal{K}$

$c = \mathsf{Enc}(k, m_0)$

World 1:

$k \leftarrow \mathcal{K}$

Eve is a **PPT distinguisher**.
She needs to decide whether

Doesn't work: we saw counter-example to this last class

For every **PPT** Eve and m_0, m_1,

$$\left| \Pr\left[\mathsf{Eve}(c) = 0 \,\middle|\, \begin{matrix} k \leftarrow \mathcal{K} \\ c = \mathsf{Enc}(k, m_0) \end{matrix}\right] - \Pr\left[\mathsf{Eve}(c) = 1 \,\middle|\, \begin{matrix} k \leftarrow \mathcal{K} \\ c = \mathsf{Enc}(k, m_1) \end{matrix}\right] \right| = 0$$

Computational Indistinguishability (take 2)

World 0:

$k \leftarrow \mathcal{K}$

$c = \mathsf{Enc}(k, m_0)$

World 1:

$k \leftarrow \mathcal{K}$

$c = \mathsf{Enc}(k, m_1)$

Eve is arbitrary **PPT distinguisher**.

She needs to decide whether c came from World 0 or World 1.

For every **PPT** Eve and m_0, m_1,

$$\left| \Pr\left[\mathsf{Eve}(c) = 0 \,\middle|\, \begin{matrix} k \leftarrow \mathcal{K} \\ c = \mathsf{Enc}(k, m_0) \end{matrix}\right] - \Pr\left[\mathsf{Eve}(c) = 1 \,\middle|\, \begin{matrix} k \leftarrow \mathcal{K} \\ c = \mathsf{Enc}(k, m_1) \end{matrix}\right] \right| = \varepsilon$$

Idea: Eve can only do ε better than random guessing.

How small should ε be?

- In practice:
  - Non-negligible (too large): $1/2^{30}$
  - Negligible: $1/2^{128}$

- In theory, we care about asymptotics:
  - Non-negligible: $\varepsilon > 1/n^2$
  - Negligible: $\varepsilon < 1/p(n)$ for every poly p

Today's Lecture

- Computational indistinguishability
- Negligible functions
- Pseudorandom generators
- Semantic security
- PRGs → Semantically-secure encryption

New Notion: Negligible Functions

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

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

$$\varepsilon(n) < \frac{1}{p(n)}$$

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

Why is this the right notion?

Let Eve's ε be non-negligible $1/n^2$

(i.e. distinguishes wp $1/2 + 1/n^2$)

Eve can distinguish for $1/n^2$ fraction of keys!

Formalization: Negligible Functions

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

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

$$\varepsilon(n) < \frac{1}{p(n)}$$

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

Security Parameter: n *(sometimes λ)*

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

$$\varepsilon(n) < \frac{1}{p(n)}$$

- Runtimes & success probabilities are measured as a function of n.
- ***Want***: Honest parties run in time (fixed) polynomial in n.
- ***Allow***: Adversaries to run in time (arbitrary) polynomial in n,
- ***Require***: adversaries to have success probability negligible in n.

Computational Indistinguishability (take 3)

World 0:

$k \leftarrow \mathcal{K}$

$c = \mathsf{Enc}(k, m_0)$

World 1:

$k \leftarrow \mathcal{K}$

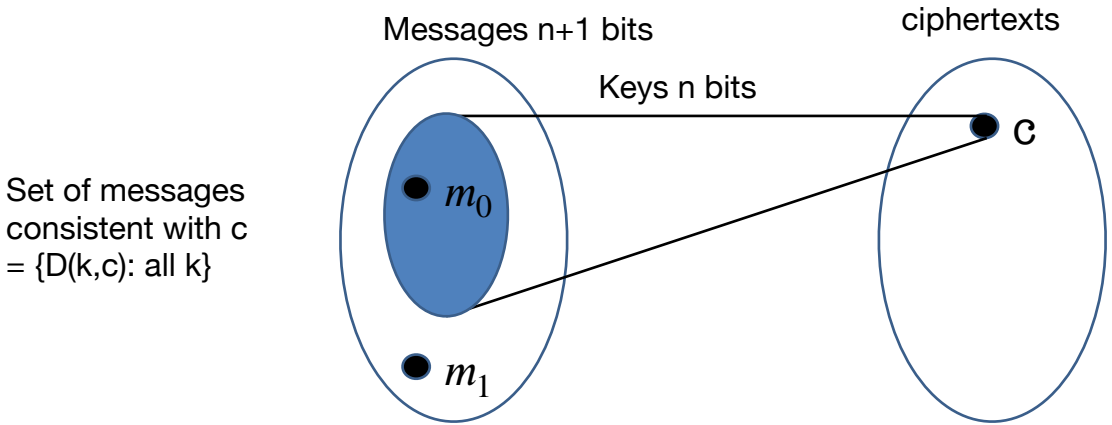
$c = \mathsf{Enc}(k, m_1)$

Eve is arbitrary **PPT distinguisher**.
She needs to decide whether c came from World 0 or World 1.

For every **PPT** Eve, there exists a negligible fn ε, st for all m_0, m_1,

$$\left| \Pr\left[\mathsf{Eve}(c) = 0 \,\middle|\, \begin{matrix} k \leftarrow \mathcal{K} \\ c = \mathsf{Enc}(k, m_0) \end{matrix}\right] - \Pr\left[\mathsf{Eve}(c) = 1 \,\middle|\, \begin{matrix} k \leftarrow \mathcal{K} \\ c = \mathsf{Enc}(k, m_1) \end{matrix}\right] \right| = \varepsilon(n)$$

What about Shannon's impossibility?



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: Eve's advantage $\approx 1/2^n$

Negligible!

Can we achieve this definition?

Yes!

Our First Crypto Tool: Pseudorandom Generators (PRG)

Pseudorandom Generators

Informally: **Deterministic** Programs that stretch a "truly random" seed into a (much) longer sequence of **"seemingly random"** bits.

Q1: How to define "seemingly random"?

Q2: Can such a G exist?

How to **Define** a Strong Pseudo Random Number Generator?

Def 1 [Indistinguishability]

"No polynomial-time algorithm can distinguish between the output of a PRG on a random seed vs. a truly random string"

= "as good as" a truly random string for practical purposes.

Def 2 [Next-bit Unpredictability]

"No polynomial-time algorithm can predict the $(i+1)^{th}$ bit of the output of a PRG given the first i bits, better than chance"

THREE DEFS ALL EQUIVALENT!

Def 3 [Incompressibility]

"No polynomial-time algorithm can compress the output of the PRG into a shorter string"

PRG Def 1: Indistinguishability

Definition [Indistinguishability]:

A deterministic polynomial-time computable function

$$G : \{0,1\}^n \to \{0,1\}^m \text{ is a } \textbf{PRG} \text{ if:}$$

(a) It is expanding: $m > n$ and

(b) for every PPT algorithm D (called a distinguisher) if there is a negligible function ε such that:

$$\left| \Pr[D(G(U_n)) = 1] - \Pr[D(U_m) = 1] \right| = \varepsilon(n)$$

Notation: U_n (resp. U_m) denotes the random distribution on n-bit (resp. m-bit) strings; m is shorthand for $m(n)$.

PRG Def 1: Indistinguishability

Definition [Indistinguishability]:

A deterministic polynomial-time computable function

$$G : \{0,1\}^n \to \{0,1\}^m \text{ is a } \textbf{PRG} \text{ if:}$$

(a) It is expanding: $m > n$ and

(b) for every PPT algorithm D (called a distinguisher) if there is a negligible function ε such that:

$$\Pr\left[D(y_b) = b \;\middle|\; \begin{array}{r} b \leftarrow \{0,1\} \\ x \leftarrow \{0,1\}^n \\ y_0 = G(x) \\ y_1 \leftarrow \{0,1\}^{n+1} \end{array} \right] \leq 1/2 + \varepsilon(n)$$

PRG Def 1: Indistinguishability

WORLD 1:
The Pseudorandom World

$y \leftarrow G(U_n)$

WORLD 2:
The Truly Random World

$y \leftarrow U_m$

PPT Distinguisher gets y but cannot tell which world she is in

Why is this a good definition

Good for all Applications:

As long as we can find truly random seeds, can replace true randomness by the output of PRG(seed) in ANY (polynomial-time) application.

If the application behaves differently, then it constitutes a (polynomial-time) statistical test between PRG(seed) and a truly random string.

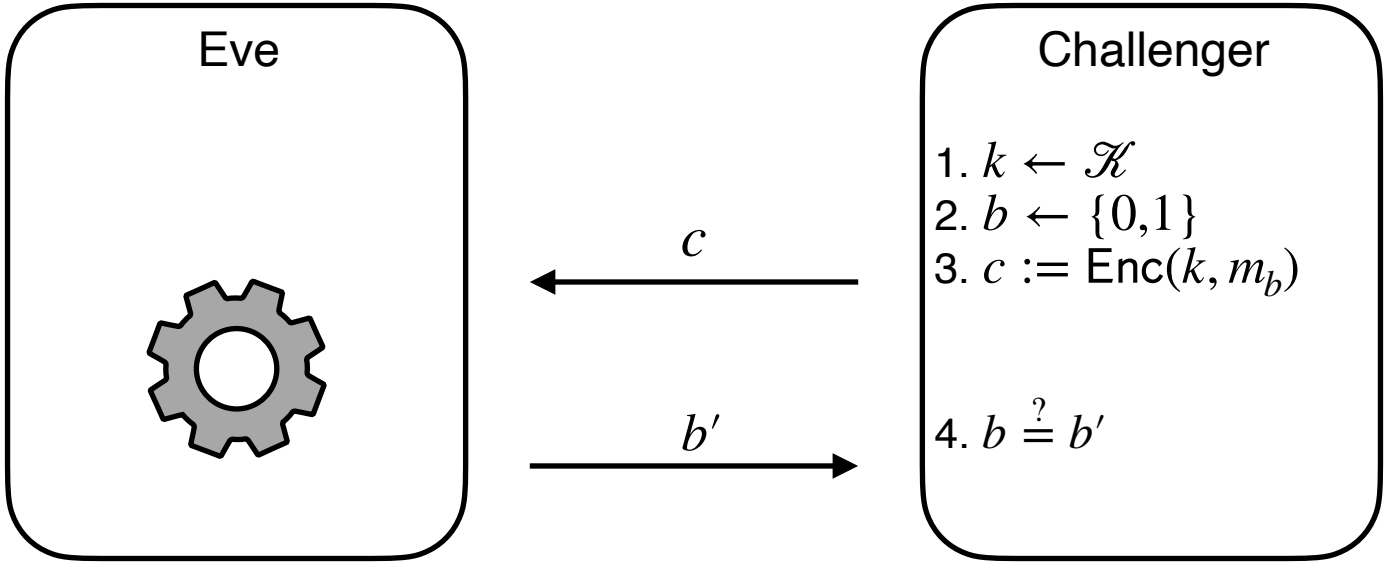
Semantic Security

For every **PPT** Eve, there exists a negligible fn ε, st for all m_0, m_1,

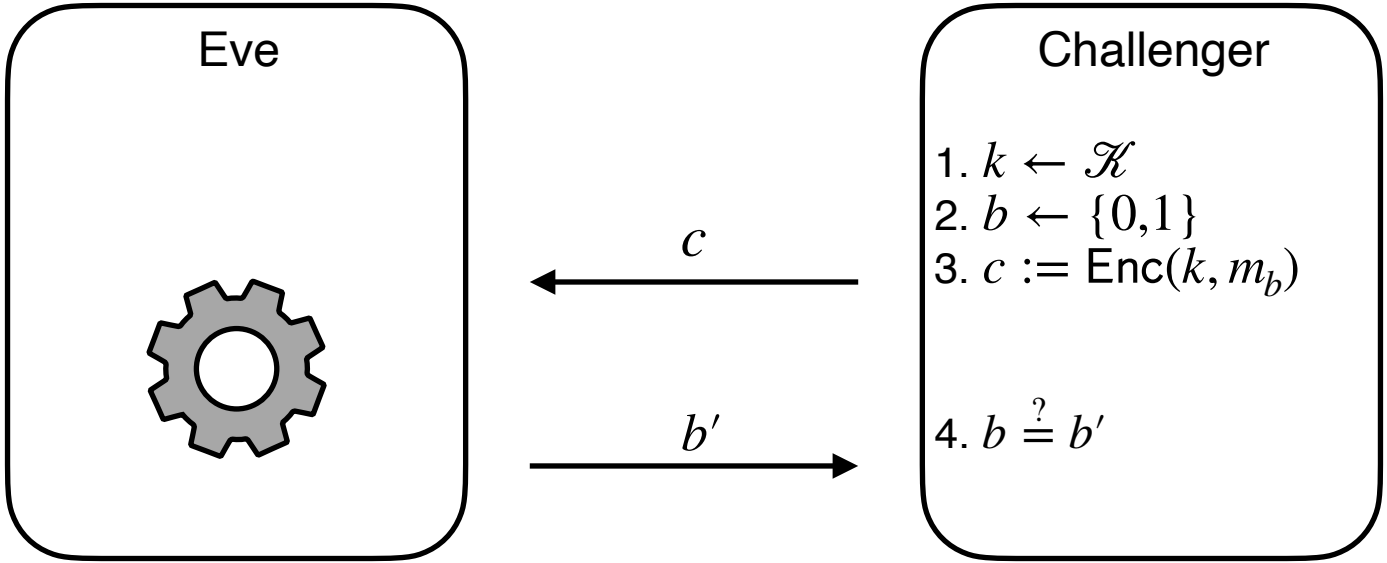
$$\Pr\left[\mathsf{Eve}(c) = b \;\middle|\; \begin{array}{r} k \leftarrow \mathcal{K} \\ b \leftarrow \{0,1\} \\ c := \mathsf{Enc}(k, m_b) \end{array}\right] < \frac{1}{2} + \varepsilon(n)$$

We briefly discussed earlier that we can view this as a game between a “challenger” and the adversary Eve. Let’s flesh that out.

Semantic Security

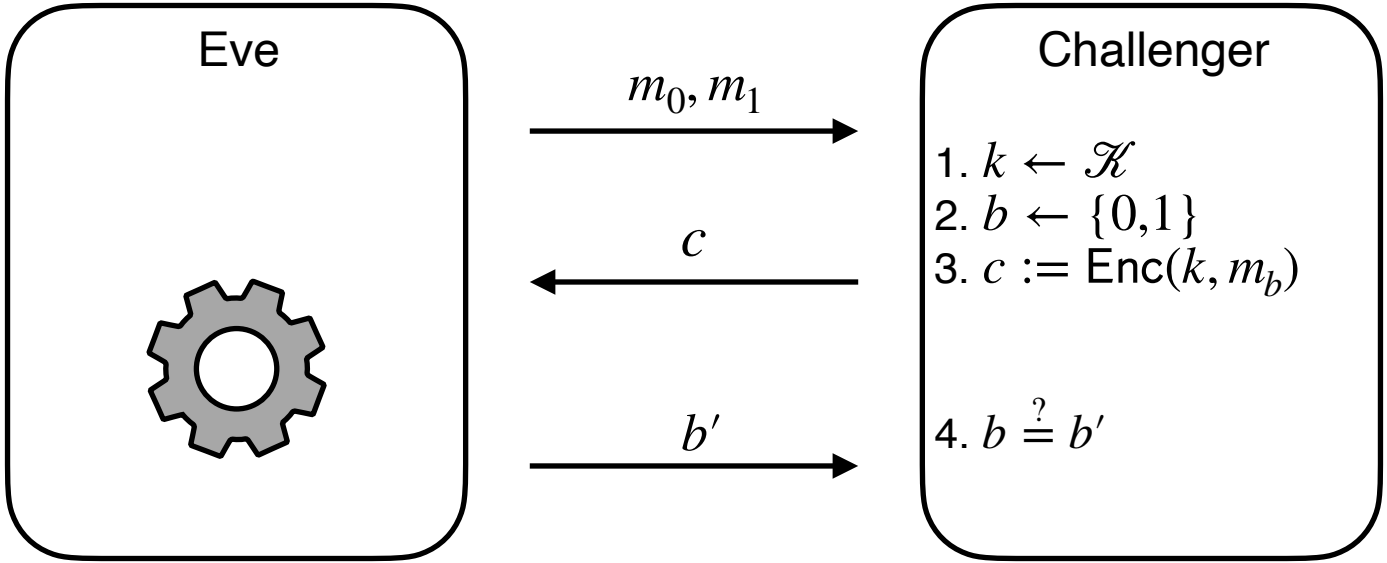


Semantic Security



We had a good question last time: how does Eve even know what the choices for m_0, m_1 are?

Semantic Security



Ans: we'll let Eve choose the messages!

Semantic Security

For every **PPT** Eve, there exists a negligible fn ε such that

$$\Pr\left[\mathsf{Eve}(c) = b \;\middle|\; \begin{array}{r}(m_0, m_1) \leftarrow \mathsf{Eve} \\ k \leftarrow \mathscr{K} \\ b \leftarrow \{0,1\} \\ c := \mathsf{Enc}(k, m_b)\end{array}\right] < \frac{1}{2} + \varepsilon(n)$$

Semantic Security

For every **PPT** Eve, there exists a negligible fn ε such that

$$\Pr\left[\mathsf{Eve}(c) = b \;\middle|\; \begin{array}{r}(m_0, m_1) \leftarrow Eve \\ k \leftarrow \mathcal{K} \\ b \leftarrow \{0,1\} \\ c := \mathsf{Enc}(k, m_b)\end{array}\right] < \frac{1}{2} + \varepsilon(n)$$

Why is this the “right” definition?

Intuitively: even if Eve knows which messages are candidate plaintexts, ciphertext *still* reveals no information!

PRGs → Semantically Secure Encryption

PRG $\Longrightarrow$ Semantically Secure Encryption

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

- Gen$(1^k) \rightarrow k$:
  - Sample an n-bit string at random.
- Enc$(k, m) \rightarrow c$:
  - Expand k to an $n+1$-bit string using PRG: $s = G(k)$
  - Output $c = s \oplus m$
- Dec$(k, c) \rightarrow m$:
  - Expand k to an $n+1$-bit string using PRG: $s = G(k)$
  - Output $m = s \oplus c$

Correctness:

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

PRG $\Longrightarrow$ Semantically Secure Encryption

Security: your first reduction!

Suppose for contradiction that there exists an Eve that breaks our scheme.

That, is assume that there is a p.p.t. Eve, and polynomial function p s.t.

$$\Pr\left[\mathsf{Eve}(c) = b \;\middle|\; \begin{array}{r} (m_0, m_1) \leftarrow \mathsf{Eve} \\ k \leftarrow \mathcal{K} \\ b \leftarrow \{0,1\} \\ c := \mathsf{Enc}(k, m_b) \end{array}\right] > \frac{1}{2} + 1/p(n)$$

PRG $\Longrightarrow$ Semantically Secure Encryption

Security: your first reduction!

Assume that there is a p.p.t. Eve, a polynomial function p and m_0, m_1 s.t.

$$\Pr\left[\mathsf{Eve}(c) = b \;\middle|\; \begin{array}{r} (m_0, m_1) \leftarrow \mathsf{Eve} \\ k \leftarrow \{0,1\}^n \\ b \leftarrow \{0,1\} \\ c := G(k) \oplus m_b \end{array}\right] > \frac{1}{2} + 1/p(n)$$

Let's call this ρ

Compare with $\Pr\left[\mathsf{Eve}(c) = b \;\middle|\; \begin{array}{r} (m_0, m_1) \leftarrow \mathsf{Eve} \\ k' \leftarrow \{0,1\}^{n+1} \\ b \leftarrow \{0,1\} \\ c := k' \oplus m_b \end{array}\right] = \frac{1}{2}$

Let's call this ρ'

Clearly, Eve can break security in
PRG case, but not in OTP world!

Eve can distinguish pseudorandom from random!

Idea: Use Eve to break PRG indistinguishability!

Distinguisher $D(y)$:

1. Get two messages m_0, m_1, from Eve and sample a bit b
2. Compute $b' \leftarrow \text{Eve}(y \oplus m_b)$
3. If $b' = b$, output “0”
4. Otherwise, output “1”

World 0

$$\Pr[D \text{ outputs "0"} \mid b = 0 \ (y \text{ is pseudorandom})] = \Pr[\text{Eve outputs } b' = b \mid b = 0] = \rho \geq 1/2 + 1/p(n)$$

World 1

$$\Pr[D \text{ outputs "1"} \mid b = 1 \ (y \text{ is random})] = \Pr[\text{Eve outputs } b' = b \mid b = 1 = \rho' = 1/2$$

Therefore,

$$\left| \Pr[D \text{ outputs "PRG"} \mid y \text{ is pseudorandom}] - \Pr[D \text{ outputs "PRG"} \mid y \text{ is pseudorandom}] \right| \geq 1/p(n)$$

PRG $\Longrightarrow$ Semantically Secure Encryption

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

$Q1$: Do PRGs exist?

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

$Q2$: How do we encrypt longer messages or many messages with a fixed key?

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

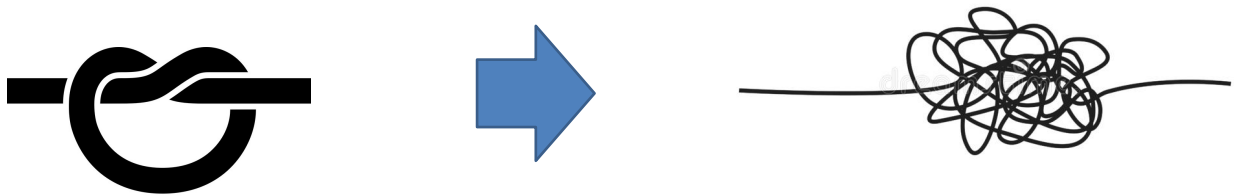
(**Pseudorandom functions**: PRGs with exponentially large stretch and "random access" to the output.)

$Q1$: Do PRGs exist?

Constructing PRGs: Two Methodologies

The Practical Methodology

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

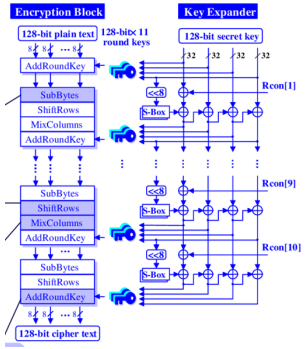


Constructing PRGs: Two Methodologies

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



Rijndael
(now the Advanced Encryption Standard)

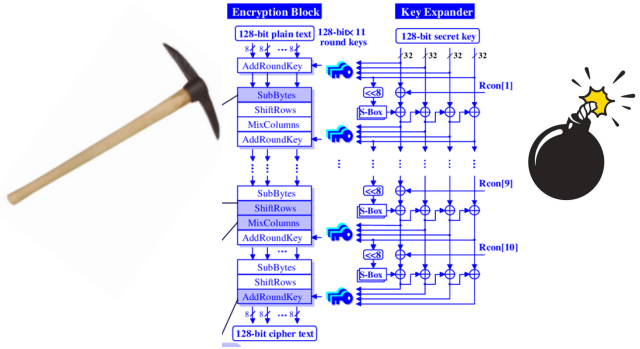
Constructing PRGs: Two Methodologies

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.



Examples

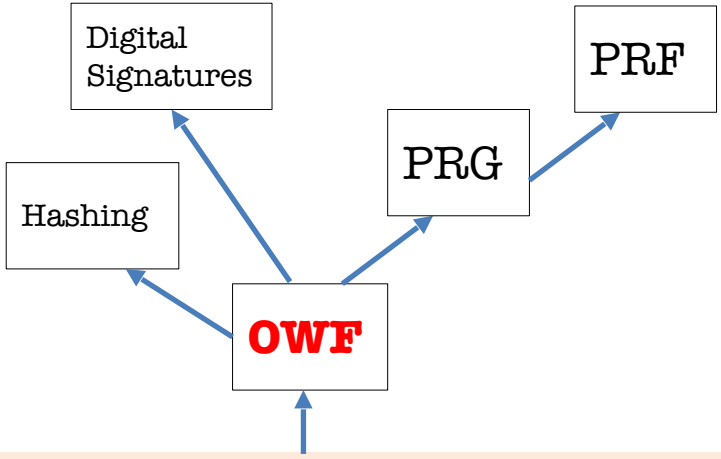
- **RC4: old PRG from 1987**
  - Proposed by Ron Rivest (of RSA fame)
  - **Fast and simple**
  - Used in TLS till 2013
  - However lots of biases
    - e.g. 2nd byte of output has 2/256 chance of being 0.
  - In 2013, attack which made key recovery feasible with just 2^{20} ciphertexts!
  - Finally deprecated in 2015, *28 years* after creation!

Constructing PRGs: Two Methodologies

The Foundational Methodology (much of this course)

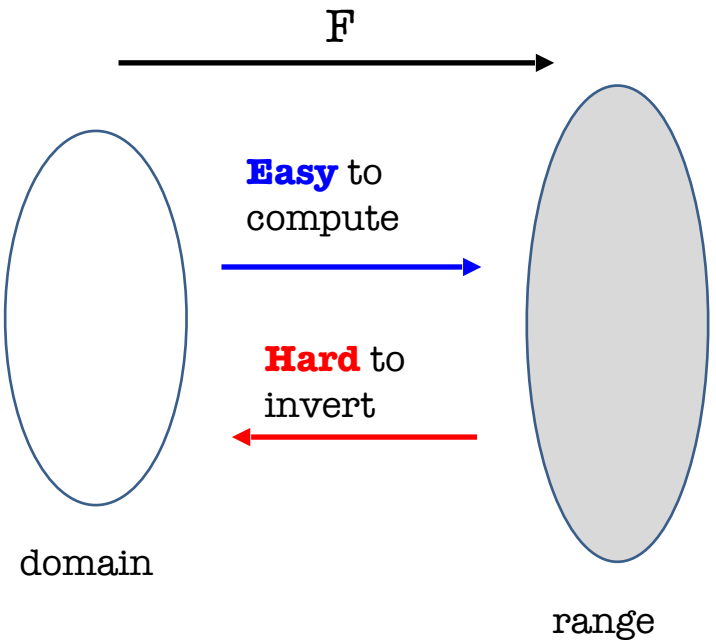
Reduce to simpler primitives.

"Science wins either way" –Silvio Micali



well-studied, average-case hard, problems

One-way Functions (Informally)



Source of all hard problems in cryptography!

What is a good definition?

One-way Functions (Take 1)

A function (family) $\{F_n\}_{n\in\mathbb{N}}$ where $F(\cdot) : \{0,1\}^n \to \{0,1\}^{m(n)}$ is **one-way** if for every p.p.t. adversary A, the following holds:

$$\Pr\left[A(1^n, y) = x \,\middle|\, \begin{array}{c} x \leftarrow \{0,1\}^n \\ y := F_n(x) \end{array}\right] = \mathsf{negl}(n)$$

Consider $F_n(x) = 0$ for all x.

This is one-way according to the above definition.
In fact, impossible to find *the* inverse even if A has unbounded time.

Conclusion: not a useful/meaningful definition.

One-way Functions (Take 1)

A function (family) $\{F_n\}_{n\in\mathbb{N}}$ where $F(\cdot) : \{0,1\}^n \to \{0,1\}^{m(n)}$ is **one-way** if for every p.p.t. adversary A, the following holds:

$$\Pr\left[A(1^n, y) = x \,\middle|\, \begin{array}{c} x \leftarrow \{0,1\}^n \\ y := F_n(x) \end{array}\right] = \mathsf{negl}(n)$$

The Right Definition: Impossible to find ***an*** inverse efficiently.

One-way Functions: The Definition

A function (family) $\{F_n\}_{n\in\mathbb{N}}$ where $F(\cdot) : \{0,1\}^n \to \{0,1\}^{m(n)}$ is **one-way** if for every p.p.t. adversary A, the following holds:

$$\Pr\left[F_n(x') = y \;\middle|\; \begin{array}{r} x \leftarrow \{0,1\}^n \\ y := F_n(x) \\ x' \leftarrow A(1^n, y) \end{array}\right] = \mathsf{negl}(n)$$

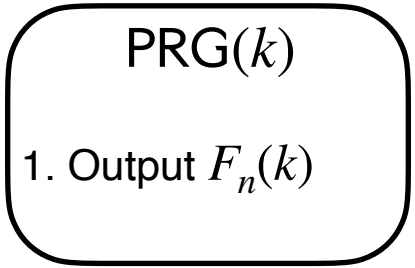
- Can always find *an* inverse with unbounded time
- … but should be hard with probabilistic polynomial time

One-way Permutations:

One-to-one one-way functions with $m(n) = n$.

How to get PRG from OWF?

OWF → PRG, Attempt #1



(Assume $m(n) > n$)

Does this work?

OWF → PRG, Attempt #1

Consider $F_n(x)$ constructed from another OWF F'_n:

1. Compute $y := F'_n(x)$
2. Output $y' := (y_0, 1, y_1, 1, \ldots, y_n, 1)$

PRG(k)

1. Output $F_n(k)$

Is F one-way?

Yes!

Is PRG unpredictable?

No!

Our problem:

OWFs don’t tell us anything about how their inputs are distributed

They are only hard to invert

Next class

- How to get randomness from OWF output
  - How to use this to get PRGs
- How to extend the length of PRGs
- How to get PRGs with “exponentially-large” output