

# Cryptography

Modern cryptography was born in 1970's when computationally easy-to-verify but 'hard-to-solve' problems were discovered.

Cryptography is closely related to some advanced topics in computational complexity.

# Synopsis

1. Computationally Secure Encryption
2. Pseudorandom Generator
3. Pseudorandom Function
4. One-Way Function
5. Zero Knowledge Proof
6. Remark

# Computationally Secure Encryption

An **encryption scheme** is a pair  $(E, D)$  of algorithms such that

$$D_k(E_k(x)) = x$$

for all **key**  $k$  and **plaintext**  $x$ . Obviously  $E_k$  is **one-one** for every  $k$ .

# Shannon's Perfect Secrecy

$(E, D)$  is **perfectly secret** if for every pair  $x, x' \in \{0, 1\}^m$ , the distributions  $E_{U_n}(x)$  and  $E_{U_n}(x')$  are identical.

- ▶  $n$  is the key length.
- ▶  $U_n$  is the uniform distribution over  $\{0, 1\}^n$ .

# One Time Pad Encryption Scheme, Vernan 1917

## Encryption:

- ▶ Plaintext  $x \in \{0, 1\}^n$ .
- ▶ Generate a key  $k \in_{\mathcal{R}} \{0, 1\}^n$ , encrypt  $x$  by  $x \oplus k$ .

## Decryption:

- ▶ Ciphertext  $y \in \{0, 1\}^n$ .
- ▶ The plaintext is recovered by  $y \oplus k$ .

---

If a key  $k$  is used twice, useful information can be derived.



# One Time Pad Encryption Scheme

**Fact.** The one time pad encryption scheme is perfectly secure.

---

It is crucial that the **key** is as long as the **message**.

**Shannon Theorem.** Suppose  $(E, D)$  is an encryption scheme. If  $n < m$ , then there exist  $x, x'$  such that  $E_{U_n}(x)$  and  $E_{U_n}(x')$  differ.

Proof.

A proof can be read off from the proof of Lemma. □

---

- ▶ Perfectly secret encryption scheme is not a practical scenario.
- ▶ Modern cryptography offers a solution.

# Negligible Functions

A function  $\epsilon : \mathbf{N} \rightarrow [0, 1]$  is **negligible** if

$$\forall c. \exists N. \forall n \geq N. \epsilon(n) < \frac{1}{n^c}.$$

In other words  $\epsilon$  is negligible if it tends to 0 faster than  $\frac{1}{p(n)}$  for every polynomial  $p(n)$ .

---

- ▶ Events with negligible probability can be practically ignored.
- ▶  $\epsilon$  is **not** negligible if  $\exists c. \epsilon(n) \geq \frac{1}{n^c}$  for infinitely many  $n$ .

# Computationally Secure Encryption Scheme

An encryption scheme  $(E, D)$  for keys of length  $n$  and messages of length  $m$  is **computationally secure** if for every P-time PTM  $\mathbb{A}$  there is a negligible function  $\epsilon : \mathbf{N} \rightarrow [0, 1]$  such that

$$\left| \Pr_{k \in_R \{0,1\}^n, x \in_R \{0,1\}^m} [\mathbb{A}(E_k(x)) = (i, b) \wedge x_i = b] - \frac{1}{2} \right| \leq \epsilon(n).$$

Is there a computationally secure encryption scheme? The answer is conditional.

**Lemma.** Suppose  $\mathbf{P} = \mathbf{NP}$ . Let  $(E, D)$  be a P-time encryption scheme with key shorter than message. A P-time algorithm  $\mathbb{A}$  exists such that for every message length  $m$ , there is a pair  $x_0, x_1 \in \{0, 1\}^m$  satisfying

$$\Pr_{b \in_R \{0,1\}, k \in_R \{0,1\}^n} [\mathbb{A}(E_k(x_b)) = b] \geq 3/4$$

where  $n$  is the key length and  $n < m$ .

---

1. Let  $S$  be defined as follows:

$$y \in S \text{ iff } \exists k. y = E_k(x_0), \text{ where } x_0 = 0^m.$$

2. If  $\mathbf{P} = \mathbf{NP}$  then  $S$  is P-time decidable by some algorithm  $\mathbb{A}$ .

►  $\mathbb{A}(x) = 0$  iff  $x \in S$ .

3. Let  $D_x = \text{distribution } E_{U_n}(x)$ . Then  $\Pr[\mathbb{A}(D_{x_0})=0] = 1$ .

If  $\Pr[D_x \in S] > \frac{1}{2}$  for all  $x$  then one would have

$$\frac{1}{2} < \Pr_x[\Pr[D_x \in S]] = \Pr_k[\Pr_x[E_k(x) \in S]] \leq \frac{1}{2},$$

where  $\leq$  holds because  $|S| \leq 2^n \leq 2^{m-1}$  by the definition of  $S$  and  $E_k$  is injective.

It follows that  $\Pr[D_{x_1} \in S] \leq \frac{1}{2}$  for some  $x_1 \in \{0, 1\}^m$ . According to the definition of  $\mathbb{A}$ , one has  $\Pr[\mathbb{A}(D_{x_1})=0] \leq \frac{1}{2}$ . Hence

$$\begin{aligned} \Pr_{b,k}[\mathbb{A}(E_k(x_b))=b] &= \frac{1}{2}\Pr[\mathbb{A}(D_{x_0})=0] + \frac{1}{2}\Pr[\mathbb{A}(D_{x_1})=1] \\ &= \frac{1}{2} + \frac{1}{2}\Pr[\mathbb{A}(D_{x_1})=1] \\ &\geq \frac{3}{4}. \end{aligned}$$

$P \neq NP$  is necessary for modern cryptography. We do not know if it is sufficient.



# Pseudorandom Generator

Modern cryptography addresses the long key issue by studying how to generate long keys from short ones.

- ▶ An observer cannot detect efficiently any useful difference between a pseudorandom key and a truly random key.

What is a pseudorandom string? How do we characterize pseudorandom strings?

- ▶ For modern cryptography it suffices that encrypted messages are distributed in a way that looks random to all **efficient** observers.

# Pseudorandom Generator

Let  $G : \{0, 1\}^* \rightarrow \{0, 1\}^*$  and  $\ell : \mathbf{N} \rightarrow \mathbf{N}$  be P-time computable such that  $\ell(n) > n$  for all  $n$  and  $|G(x)| = \ell(|x|)$  for all  $x \in \{0, 1\}^*$ .

---

$G$  is a computationally secure **pseudorandom generator** of **stretch  $\ell(n)$**  if, for every P-time PTM  $\mathbb{A}$ , there exists a negligible function  $\epsilon : \mathbf{N} \rightarrow [0, 1]$  such that

$$|\Pr[\mathbb{A}(G(U_n)) = 1] - \Pr[\mathbb{A}(U_{\ell(n)}) = 1]| \leq \epsilon(n).$$

- 
1. Yao. Theory and Applications of Trapdoor Functions. FOCS 1982.



A pseudorandom generator says nothing about how it is constructed.

# Unpredictability

Let  $G : \{0, 1\}^* \rightarrow \{0, 1\}^*$  be P-time computable with stretch  $\ell(n)$ , where  $\ell : \mathbf{N} \rightarrow \mathbf{N}$  is P-time computable such that  $\forall n. \ell(n) > n$ .

---

We say that  $G$  is **unpredictable** if for every P-time PTM  $\mathbb{B}$  there is a negligible function  $\epsilon : \mathbf{N} \rightarrow [0, 1]$  such that

$$\left| \Pr_{x \in_R \{0,1\}^n, y = G(x), i \in_R [\ell(n)]} [\mathbb{B}(1^n, y_1, \dots, y_{i-1}) = y_i] - \frac{1}{2} \right| \leq \epsilon(n).$$

- 
1. M. Blum, S. Micali. How to Generate Cryptographically Strong Sequences of Pseudorandom Bits. FOCS 1982.



## Unpredictability $\Leftarrow$ Pseudorandomness

Suppose  $G$  is a pseudorandom generator. If it is not unpredictable then there is some  $c$  such that

$$\left| \Pr_{x \in_R \{0,1\}^n, y=G(x), i \in_R [\ell(n)]} [\mathbb{B}(1^n, y_1, \dots, y_{i-1}) = y_i] - \frac{1}{2} \right| \geq \frac{1}{n^c}$$

holds for a P-time PTM  $\mathbb{B}$  for infinitely many  $n$ . Some  $i$  exists such that

$$\left| \Pr_{x \in_R \{0,1\}^n, y=G(x)} [\mathbb{B}(1^n, y_1, \dots, y_{i-1}) = y_i] - \frac{1}{2} \right| \geq \frac{1}{n^c \ell(n)}$$

for infinitely many  $n$ . It follows from  $\Pr[\mathbb{B}(U_{\ell(n)}) = 1] = \frac{1}{2}$  that

$$\Pr[\mathbb{B}(G(U_n)) = 1] - \Pr[\mathbb{B}(U_{\ell(n)}) = 1] \geq \frac{1}{n^c \ell(n)}$$

for infinitely many  $n$ , which is a contradiction.

# Unpredictability $\Rightarrow$ Pseudorandomness

**Theorem** (Yao, 1982). If  $G$  is unpredictable, then it is a pseudorandom generator.

---

1. Yao. Theory and Applications of Trapdoor Functions. FOCS 1982.





# Unpredictability $\Rightarrow$ Pseudorandomness

Let  $\ell : \mathbf{N} \rightarrow \mathbf{N}$  be P-time computable such that  $\ell(n) \geq n$ .

Let  $G : \{0, 1\}^* \rightarrow \{0, 1\}^*$  be P-time computable unpredictable function with stretch  $\ell$ .

---

Suppose  $G$  is not a pseudorandom generator. Then there is some constant  $c$  and some P-time PTM  $\mathsf{A}$  such that, wlog,

$$\Pr[\mathsf{A}(G(U_n)) = 1] - \Pr[\mathsf{A}(U_{\ell(n)}) = 1] \geq \frac{1}{n^c}$$

for infinitely many  $n$ .

## Unpredictability $\Rightarrow$ Pseudorandomness

For  $i \leq \ell(n)$ , the **hybrid** distribution  $\mathcal{D}_i$  is defined as follows:

1. choose  $x \in_{\mathcal{R}} \{0, 1\}^n$  and compute  $y = G(x)$ ;
2. output  $y_1, \dots, y_i, z_{i+1}, \dots, z_{\ell(n)}$  with  $z_{i+1}, \dots, z_{\ell(n)} \in_{\mathcal{R}} \{0, 1\}$ .

We notice that  $\mathcal{D}_0 = U_{\ell(n)}$  and  $\mathcal{D}_{\ell(n)} = G(U_n)$ .

---

Let  $p_i = \Pr[\mathbf{A}(\mathcal{D}_i) = 1]$ . By assumption for infinitely many  $n$ ,

$$p_{\ell(n)} - p_0 = (p_{\ell(n)} - p_{\ell(n)-1}) + (p_{\ell(n)-1} - p_{\ell(n)-2}) + \dots + (p_1 - p_0) \geq \frac{1}{n^c}.$$

# Unpredictability $\Rightarrow$ Pseudorandomness

Algorithm  $\mathbb{B}$  asserts that everything  $\mathbb{A}$  says is correct.

- ▶ Input  $1^n$ ,  $i \in [\ell(n)]$  and  $y_1, \dots, y_{i-1}$ .
  1. randomly generate  $z_i, \dots, z_{\ell(n)}$ ;
  2. compute  $a = \mathbb{A}(y_1, \dots, y_{i-1}, z_i, \dots, z_{\ell(n)})$ ;
  3. output  $z_i$  if  $a = 1$  and  $1 - z_i$  if  $a = 0$ .

---

We are done if we can prove the following inequality

$$\Pr_{x \in_R \{0,1\}^n, y=G(x), i \in_R [\ell(n)]} [\mathbb{B}(1^n, y_1, \dots, y_{i-1}) = y_i] - \frac{1}{2} \geq \frac{1}{n^c \ell(n)},$$

which can be derived if the following holds for every  $i \in [\ell(n)]$ :

$$\Pr_{x \in_R \{0,1\}^n, y=G(x)} [\mathbb{B}(1^n, y_1, \dots, y_{i-1}) = y_i] = \frac{1}{2} + (p_i - p_{i-1}).$$

## Unpredictability $\Rightarrow$ Pseudorandomness

$\mathbb{B}$  predicts  $y_i$  correctly if  $a = 1 \wedge z_i = y_i$  or  $a = 0 \wedge z_i = 1 - y_i$ . This event happens with probability

$$\frac{1}{2} \Pr_{x,y=G(x)}[a = 1 | z_i = y_i] + \frac{1}{2} (1 - \Pr_{x,y=G(x)}[a = 1 | z_i = 1 - y_i]) .$$

Now  $\Pr_{x \in_R \{0,1\}^n, y=G(x)}[a = 1 | z_i = y_i] = p_i$ . On the other hand,

$$\begin{aligned} p_{i-1} &= \Pr[\textcolor{violet}{A}(\mathcal{D}_{i-1}) = 1] \\ &= \Pr[a = 1 | z_i = y_i]/2 + \Pr[a = 1 | z_i = 1 - y_i]/2 \\ &= p_i/2 + \Pr[a = 1 | z_i = 1 - y_i]/2. \end{aligned} \tag{1}$$

We get  $\Pr[a = 1 | z_i = 1 - y_i] = 2p_{i-1} - p_i$  from (1).

**Theorem** Given a pseudorandom generator with stretch  $n^c$ , one can design a computationally secure encryption scheme  $(E, D)$  using  $n$ -length keys for  $n^c$ -length messages.

---

Given a random key of length  $n$ , generate a key of length  $n^c$  using the pseudorandom generator, and then apply the one-time pad encryption scheme.

## Application: Derandomization

If pseudorandom generator exists, then we can construct **subexponential deterministic** algorithms for problems in **BPP**.

- ▶ This is the **derandomization** of **BPP**.

---

The basic idea:

- ▶ Let  $L$  be decided by an  $n^d$ -time PTM  $\mathbb{P}$  with bounded error.
- ▶ For every small  $\epsilon$  let  $c$  be such that  $0 < \frac{d}{c} < \epsilon < 1$ .
- ▶ Apply to all strings of length  $n^{\frac{d}{c}}$  the pseudorandom generator with stretch  $n^c$  and then execute  $\mathbb{P}$  by following the choices prescribed by the produced pseudorandom strings of length  $n^d$ .
- ▶ The algorithm runs in time  $O(2^{n^\epsilon})$

# Pseudorandom Function

Let  $\mathcal{F}_n$  denote the set of all functions of type  $\{0, 1\}^n \rightarrow \{0, 1\}^n$ .

1. Generally  $n2^n$  bits are necessary to specify a function in  $\mathcal{F}_n$ .
2. Consequently its computation is not efficient.

---

We look for an efficient subset  $\mathcal{G}_n$  of  $\mathcal{F}_n$  that appears random.

1. Every element of  $\mathcal{G}_n$  is specified by  $n$  bits.
2. Every element of  $\mathcal{G}_n$  is P-time computable.
3. Yet no P-time PTM can detect noticeable difference between a random element of  $\mathcal{G}_n$  and a random element of  $\mathcal{F}_n$ .

- 
- ▶ There are  $2^{n2^n}$  elements in  $\mathcal{F}_n$ .
  - ▶ There are only  $2^n$  elements in  $\mathcal{G}_n$ .



Pseudorandom functions are pseudorandom generators with exponential stretch.

- ▶ A pseudofunction is a blackbox, a distinguisher can only ask for the values of the function at a **small** number of inputs.

# Pseudorandom Function

Let  $\{f_k\}_{k \in \{0,1\}^*}$  be a family of functions such that

- ▶  $f_k : \{0,1\}^{|k|} \rightarrow \{0,1\}^{|k|}$  for every  $k \in \{0,1\}^*$ , and
- ▶  $f_k(x)$  is P-time computable from  $k, x$ .

---

The family  $\{f_k\}_{k \in \{0,1\}^*}$  is **pseudorandom** if for every P-time probabilistic OTM  $\mathbb{A}$  there is a negligible function  $\epsilon : \mathbf{N} \rightarrow [0, 1]$  such that for all  $n$ ,

$$\left| \Pr_{k \in_{\mathbf{R}} \{0,1\}^n} [\mathbb{A}^{f_k}(1^n) = 1] - \Pr_{g \in_{\mathbf{R}} \mathcal{F}_n} [\mathbb{A}^g(1^n) = 1] \right| \leq \epsilon(n).$$

---

$\mathbb{A}$  needs no input. The string  $1^n$  marks the input length.

# Pseudorandom Generator $\Leftarrow$ Pseudorandom Function

Suppose  $\{f_k\}_{k \in \{0,1\}^*}$  is a pseudorandom family of functions. It follows from definition

- ▶ that for every polynomial  $\ell(n)$ , the map  $G$  defined by

$$k \in \{0,1\}^n \mapsto f_k(1), \dots, f_k(\ell(n)) \in \{0,1\}^{n\ell(n)}$$

is a pseudorandom generator.

## Goldreich-Goldwasser-Micali Theorem.

Suppose that there exists a pseudorandom generator  $G$  with stretch  $\ell(n) = 2n$ . Then there exists a pseudorandom function family.

- 
1. O. Goldreich, S. Goldwasser, S. Micali. How to Construct Random Functions. FOCS 1984.



# Pseudorandom Generator $\Rightarrow$ Pseudorandom Function

Let  $G$  be a pseudorandom generator with stretch  $2n$ .

- ▶  $G_0(x)$  is the first  $n$  bits of  $G(x)$ ;
- ▶  $G_1(x)$  is the last  $n$  bits of  $G(x)$ .

For each seed  $k \in \{0, 1\}^n$  the function  $f_k$  is defined by

$$f_k(x) = G_{x_n}(G_{x_{n-1}}(\dots G_{x_1}(k)\dots)).$$

---

We will prove that  $\{f_k\}_{k \in \{0,1\}^*}$  is a pseudorandom function family.

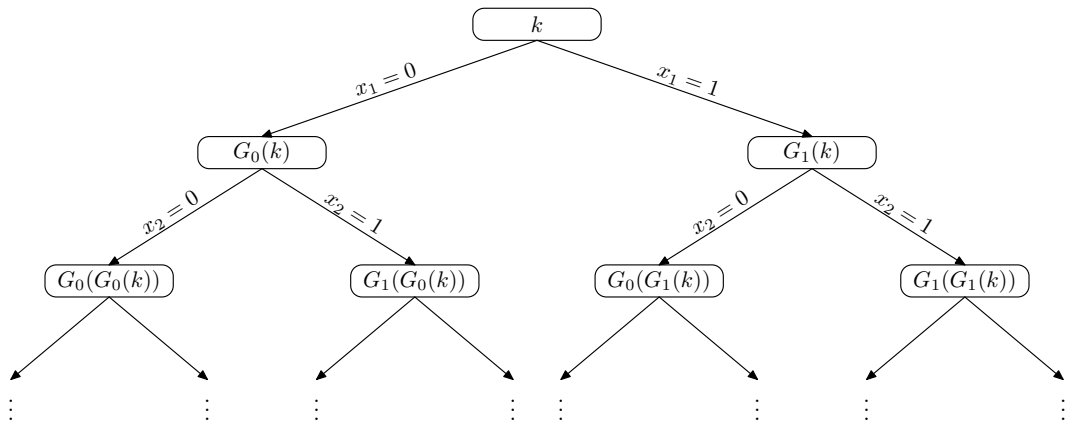


Figure: The Algorithm that Calculates  $f_k(x)$ .

# Pseudorandom Generator $\Rightarrow$ Pseudorandom Function

Let  $\mathbb{A}$  be a  $T$ -time PTM that distinguishes  $\{f_k\}_{k \in \{0,1\}^n}$  and  $\mathcal{F}_n$ . I.e. some  $c$  exists such that the following inequality holds for infinitely many  $n$ ,

$$\Pr_{g \in_R \mathcal{F}_n}[\mathbb{A}^g(1^n) = 1] - \Pr_{k \in_R \{0,1\}^n}[\mathbb{A}^{f_k}(1^n) = 1] \geq \frac{1}{n^c}.$$

---

We construct a P-time PTM  $\mathbb{B}$  that distinguishes  $U_{2n}$  and  $G(U_n)$  with  $\frac{1}{nT} \cdot \frac{1}{n^c}$  bias.

- ▶ Define a **random** implementation  $\mathcal{O}$  of the oracles  $f_{U_n}$  in terms of  $G$ .
  1. generate a seed  $k \in_R \{0,1\}^n$  randomly;
  2. run the algorithm that calculates  $f_k$  on **all** queries.
- ▶ We then modify  $\mathcal{O}$  to get  $\{\mathcal{O}_i\}_{i \leq nT}$  using hybrid approach.

# Pseudorandom Generator $\Rightarrow$ Pseudorandom Function

For  $i \leq nT$  the random oracle  $\mathcal{O}_i$  is defined as follows:

1. In the first  $i$ -th steps generate children randomly.
2. After the first  $i$ -steps generate children pseudo-randomly using  $G$ .
3. The random answers must be **consistent**!

Clearly  $\mathcal{O}_0$  is  $\mathcal{O}$ , and  $\mathcal{O}_{nT}$  is a random function.

---

Let  $p_i = \Pr[\mathbb{A}^{\mathcal{O}_i}(1^n) = 1]$ . Observe that

$$\blacktriangleright p_0 = \Pr_{k \in_R \{0,1\}^n}[\mathbb{A}^{f_k}(1^n) = 1] \text{ and } p_{nT} = \Pr_{g \in_R \mathcal{F}_n}[\mathbb{A}^g(1^n) = 1].$$

By assumption  $p_{nT} - p_0 \geq \frac{1}{n^c}$ .



Algorithm  $\mathbb{B}$ .

1. Input  $k \in \{0, 1\}^{2n}$ .
2. Generate  $i \in_{\mathbb{R}} [nT]$ .
3. Run  $\mathbb{A}^{\mathcal{O}_i}(1^n)$ , with the modification of  $\mathcal{O}_i$  that in the  $i$ -th invocation the two children are the first respectively the last  $n$  bits of  $k$ .

---

The following can be easily verified.

- ▶ If  $k \in_{\mathbb{R}} U_{2n}$ , then  $\mathbb{B}$ 's output is distributed as  $\mathbb{A}^{\mathcal{O}_i}(1^n)$ .
- ▶ If  $k \in_{\mathbb{R}} G(U_n)$ , then  $\mathbb{B}$ 's output is distributed as  $\mathbb{A}^{\mathcal{O}_{i-1}}(1^n)$ .

---

Using hybrid argument,  $\Pr[\mathbb{B}(U_{2n}) = 1] - \Pr[\mathbb{B}(G(U_n)) = 1]$  is

$$\sum_{i \in [nT]} \frac{\Pr[\mathbb{A}^{\mathcal{O}_i}(1^n) = 1]}{nT} - \sum_{i \in [nT]} \frac{\Pr[\mathbb{A}^{\mathcal{O}_{i-1}}(1^n) = 1]}{nT} = \frac{p_{nT}}{nT} - \frac{p_0}{nT} \geq \frac{1}{nT} \cdot \frac{1}{n^c}.$$

## Application: One Key for Many Messages

By Goldreich-Goldwasser-Micali Theorem and Yao's Theorem, the string

$$f_k(r_1)f_k(r_2)f_k(r_3)\dots f_k(r_{\ell(k)})$$

is unpredictable.

---

1. **Alice** encrypts a message  $x \in \{0, 1\}^n$  by choosing  $r \in_R \{0, 1\}^n$  and sends  $(r, f_k(r) \oplus x)$  to **Bob**, where  $k \in \{0, 1\}^n$  is the key.
2. **Bob** receives  $(r, y)$  and calculates  $f_k(r) \oplus y$  to recover  $x$ .

## Application: Message Authentication Code

For the same reason the following protocol is secure.

1. Alice sends  $x$  to Bob.
2. Bob sends  $(x, f_k(x))$  to Alice.
3. Alice receives  $(x, y)$  and checks if  $y = f_k(x)$  to verify that the message has not been corrupted.

## Application: Lower Bound for Machine Learning

In machine learning the goal is to learn a function  $f$  from a sequence of examples  $(r_1, f(r_1)), \dots, (r_k, f(r_k))$ .

- ▶ The existence of pseudorandom function implies that even if  $f$  is P-time computable, there is no way to learn it in P-time.

# One-Way Function

Suppose  $G : \{0, 1\}^* \rightarrow \{0, 1\}^*$  is a pseudorandom generator.

For every P-time PTM  $\mathbb{A}$  there must be a negligible function  $\epsilon : \mathbf{N} \rightarrow [0, 1]$  such that the following holds for every  $n$ ,

$$\Pr_{x \in_R \{0,1\}^n} [\mathbb{A}(1^n, G(x)) = x' \wedge G(x') = G(x)] \leq \epsilon(n).$$

# One-Way Function

A P-time function  $f : \{0, 1\}^* \rightarrow \{0, 1\}^*$  is a **one-way function** if for every P-time PTM  $\mathbb{A}$  there is a negligible function  $\epsilon : \mathbf{N} \rightarrow [0, 1]$  such that for every  $n$ ,

$$\Pr_{x \in_R \{0,1\}^n, y=f(x)}[\mathbb{A}(1^n, y) = x' \wedge f(x') = y] \leq \epsilon(n).$$

Let  $f : \{0, 1\}^* \rightarrow \{0, 1\}^*$  be a P-time computable function such that  $\forall x. |x| \leq |f(x)|$ .

- ▶ If  $\mathbf{P} = \mathbf{NP}$  then  $\{(l, u, y) \mid \exists x. f(x) = y \wedge l \leq x \leq u\} \in \mathbf{P}$ .
- ▶ By divide-and-conquer one can compute  $f^{-1}$  in P-time.

---

The existence of one way function implies  $\mathbf{P} \neq \mathbf{NP}$ .



Integer multiplication is believed to be one-way.

**Theorem.** If one-way **permutations** exist, then for every  $c \in \mathbf{N}$ , there exists a pseudorandom generator with stretch  $S(n) = n^c$ .

1. Q. Yao. Theory and Applications of Trapdoor Functions. FOCS 1982.

---

**Theorem.** If one-way **functions** exist, then for every  $c \in \mathbf{N}$ , there exists a pseudorandom generator with stretch  $S(n) = n^c$ .

1. J. Håstad, R. Impagliazzo, L. Levin and M. Luby. A Pseudorandom Generator from any One-way Function. SIAM Journal on Computing, 28:1364-1396, 1999.

The crucial step is to obtain a pseudorandom generator that extends input by one bit.

- ▶ If  $f$  is a one-way permutation, then  $G(x, r) = f(x), r, x \odot r$  is a pseudorandom generator. Intuitively  $r$  is random,  $f(x)$  is pseudorandom, and the  $(2n+1)$ -th bit cannot be predicted with probability noticeably larger than  $1/2$ .

---

We shall prove **Theorem** using Goldreich-Levin Theorem.

**Goldreich-Levin Theorem.** Suppose  $f : \{0, 1\}^* \rightarrow \{0, 1\}^*$  is a one-way permutation. Then for every P-time PTM  $\mathbb{A}$  there is a negligible function  $\epsilon : \mathbf{N} \rightarrow [0, 1]$  such that

$$\left| \Pr_{x, r \in_{\mathbf{R}} \{0, 1\}^n} [\mathbb{A}(f(x), r) = x \odot r] - \frac{1}{2} \right| \leq \epsilon(n),$$

where  $x \odot r = \sum_{i=1}^n x_i r_i \pmod{2}$ .

- 
1. O. Goldreich, L. Levin. A Hard-Core Predicate for All One-Way Functions. STOC 1989.



---

We call  $x \odot r$  the **hard core bit** of the function  $xr \mapsto f(x)r$ .

Scenario:

- ▶ We know  $f(x)$  and that  $\mathbb{A}(f(x), r)$  approximates  $x \odot r$  to some extent.
- ▶ We hope to recover  $x$ .

---

1. If  $\mathbb{A}(f(x), r) = x \odot r$  for all  $r$ , then it is easy to recover  $x$  by the following algorithm:

- ▶ Run  $\mathbb{A}(f(x), e^1), \dots, \mathbb{A}(f(x), e^n)$ .
- ▶ Paste the resulting  $n$  bits to get  $x$ .

2. Suppose  $\Pr_{r \in_R \{0,1\}^n}[\mathbb{A}(f(x), r) = x \odot r] \geq 0.9$ .

---

Now  $x \odot r$  is uniformly distributed if  $r$  is uniformly distributed. So by union bound

$$\Pr_{r \in_R \{0,1\}^n}[(\mathbb{A}(f(x), r) \neq x \odot r) \vee (\mathbb{A}(f(x), r \oplus e^i) \neq x \odot (r \oplus e^i))] \leq 0.2.$$

The equality  $(x \odot r) \oplus (x \odot (r \oplus e^i)) = x \odot (r \oplus r \oplus e^i) = x \odot e^i = x_i$  implies that

$$\Pr_{r \in_R \{0,1\}^n}[\mathbb{A}(f(x), r) \oplus \mathbb{A}(f(x), r \oplus e^i) = x_i] \geq 0.8, \quad (2)$$

which can be amplified to  $1 - 1/10n$  by majority vote.

---

- If we decrease 0.9 to 0.75, then 0.8 goes down to 0.5, rendering the lower bound in (2) utterly useless.

Algorithm **B**: The input is  $y = f(x)$ .

1.  $m := 200n$ .
2. Choose  $r^1, \dots, r^m \in_{\mathbb{R}} \{0, 1\}^n$ .
3. For  $i$  from 1 to  $n$  do
  - 3.1  $z_1 := \mathbb{A}(f(x), r^1), z'_1 := \mathbb{A}(f(x), r^1 \oplus e^i),$   
 $\dots,$   
 $z_m := \mathbb{A}(f(x), r^m), z'_m := \mathbb{A}(f(x), r^m \oplus e^i).$
  - 3.2 guess that  $x_i$  is the majority value of  $\{z_j \oplus z'_j\}_{j \in [m]}$ .

## Analysis of $\mathbb{B}$ :

1. Let the indicator random variable  $Z_j$  be defined by

$$Z_j(r^j) = \begin{cases} 1, & \text{if } \mathbb{A}(f(x), r^j) = x \odot r^j \text{ and } \mathbb{A}(y, r^j \oplus e^i) = x \odot (r^j \oplus e^i), \\ 0, & \text{otherwise.} \end{cases}$$

2. Clearly  $Z_1, \dots, Z_m$  are independent. Let  $Z = Z_1 + \dots + Z_m$ .
3.  $E[Z_j] \geq 0.8$  and  $E[Z] \geq 0.8m$ .
4.  $\Pr[|Z - E[Z]| \geq 0.3m] \leq 1/(0.3\sqrt{m})^2$  by Chebychev inequality.
5. It follows from  $m = 200n$  that  $\Pr[Z \leq 0.5m] \leq 1/10n$ .

- 
- ▶ Chebychev inequality:  $\Pr[|Z - E[Z]| \geq k\sqrt{\text{Var}(Z)}] \leq 1/k^2$ .
  - ▶  $\text{Var}(Z) = \sum_{j=1}^m \text{Var}(Z_j) \leq m$  since  $\text{Var}(Z_j) \leq 1$  for all  $j$ .



3. Suppose there are constant  $c \in \mathbf{N}$  and P-time PTM  $\mathbb{A}$  such that

$$\Pr_{x,r \in_R \{0,1\}^n} [\mathbb{A}(f(x), r) = x \odot r] - \frac{1}{2} \geq \frac{1}{n^c}$$

for infinitely many  $n$ . There is at least a  $\frac{1}{2n^c}$  fragment of  $x$ 's, the **good**  $x$ 's, such that

$$\Pr_{r \in_R \{0,1\}^n} [\mathbb{A}(f(x), r) = x \odot r] - \frac{1}{2} \geq \frac{1}{2n^c}$$

for infinitely many  $n$ .

---

**Lemma.** Suppose  $a_1, a_2, \dots, a_n \in [0, 1]$  and  $\rho = (\sum_{i \in [n]} a_i)/n$ . There is at least  $\frac{\rho}{2}$  fraction of  $a_i$ 's such that  $a_i \geq \frac{\rho}{2}$ .

The point is that we cannot afford to applying  $\mathbb{A}$  twice for probabilistic reason.

Instead of calculating  $\mathbb{A}(f(x), r^1), \dots, \mathbb{A}(f(x), r^m)$ , we **guess**  $x \odot r^1, \dots, x \odot r^m$ . Because we are guessing the values of these expressions, we do not need to know  $x$ .

---

- ▶ Choose randomly distinct seeds  $s^1, \dots, s^k \in_R \{0, 1\}^n$ .
  - ▶  $\{\bigoplus R\}_{R \subseteq \{s^1, \dots, s^k\}}$  are random and **pairwise** independent.
  - ▶  $\{x \odot \bigoplus R\}_{R \subseteq \{s^1, \dots, s^k\}}$  are completely determined by  $x \odot s^1, \dots, x \odot s^k$ .
- 

We can afford **exhaustive guessing** if  $k = \log m$ .

Algorithm **C**: The input is  $y = f(x)$ .

1. Input  $y \in \{0, 1\}^n$ . Think of  $y$  as  $f(x)$  for some  $x$ .
2.  $m := 10n^{2c+1}$ ;
3.  $k := \log(m)$ ;
4. Generate  $s^1, \dots, s^k \in_{\mathbb{R}} \{0, 1\}^n$ ;
5. Let  $R^1, \dots, R^m$  be subsets of  $\{s^1, \dots, s^k\}$  in a canonical way;
6. For each guess  $w \in \{0, 1\}^k$  do
  - 6.1 for each  $i \in [n]$  do
    - 6.1.1  $x \odot s^1 := w_1, \dots, x \odot s^k := w_k$ ;  
 $z_1 := \bigoplus_{t \in R^1} (x \odot s^t), \dots, z_m := \bigoplus_{t \in R^m} (x \odot s^t)$ ;  
 $z'_1 := \mathbb{A}(y, \bigoplus R^1 \oplus e^i), \dots, z'_m := \mathbb{A}(y, \bigoplus R^m \oplus e^i)$ ;
    - 6.1.2 guess that  $x_i$  is the majority value of  $\{z_j \oplus z'_j\}_{j \in [m]}$ .
  - 6.2  $x := x_1 \dots x_n$ ;
  - 6.3 if  $f(x) = y$ , output  $x$  and halt.

## Analysis of $\mathbb{C}$ :

1. Let the indicator random variable  $Z_j$  be defined by

$$Z_j(r^j) = \begin{cases} 1, & \text{if } \mathbb{A}(y, r^j \oplus e^i) = x \odot (r^j \oplus e^i), \\ 0, & \text{otherwise.} \end{cases}$$

2.  $Z_1, \dots, Z_m$  are pairwise independent and  $E[Z_j] \geq 1/2 + 1/n^c$ .
3. Hence  $E[Z] \geq m/2 + m/n^c$ , where  $Z = Z_1 + \dots + Z_m$ .
4. Using  $\text{Var}(Z) = \sum_{j=1}^m \text{Var}(Z_j) \leq m$ , we derive

$$\begin{aligned} \Pr[|Z - E[Z]| \geq m/n^c] &\leq \Pr[|Z - E[Z]| \geq \frac{\sqrt{m}}{n^c} \sqrt{\text{Var}(Z)}] \\ &\leq \frac{n^{2c}}{m} = \frac{n^{2c}}{10n^{2c+1}} = \frac{1}{10n}. \end{aligned}$$

5. Now  $\Pr[Z \leq m/2] \leq \frac{1}{10n}$  follows from 3 and 4.

**Theorem.** Let  $f$  be a one-way **permutation**. The function mapping  $x, r \in \{0, 1\}^n$  onto

$$r, f^{n^c}(x) \odot r, f^{n^c-1}(x) \odot r, \dots, f^1(x) \odot r$$

is a pseudorandom generator of stretch  $n + n^c$  for every  $c \in \mathbf{N}$ .

---

Let  $\mathbb{A}$  be a P-time PTM such that for  $x, r \in_{\mathbf{R}} \{0, 1\}^n$  and  $i \in_{\mathbf{R}} [n^c]$ ,

$$\Pr[\mathbb{A}(r, f^{n^c}(x) \odot r, f^{n^c-1}(x) \odot r, \dots, f^{i+1}(x) \odot r) = f^i(x) \odot r] - \frac{1}{2} \geq \frac{1}{n^d}$$

for some  $d \in \mathbf{N}$  and infinitely many  $n$ .

continued on the next slide.

$\mathbb{B}(y, r)$ , where  $y, r \in \{0, 1\}^n$ , is designed as follows:

1. Generate  $i \in_{\mathbb{R}} [n^c]$ ;
2. Output  $\mathbb{A}(r, f^{n^c-i}(y) \odot r, \dots, f^1(y) \odot r, y \odot r)$ .

---

The probability that  $\mathbb{B}(f(x), r)$  outputs  $x \odot r$  is the same as

$$\Pr[\mathbb{A}(r, f^{n^c}(x) \odot r, f^{n^c-1}(x) \odot r, \dots, f^{i+1}(x) \odot r) = f^i(x) \odot r].$$

Hence

$$\Pr_{x, r \in_{\mathbb{R}} \{0, 1\}^n}[\mathbb{B}(f(x), r) = x \odot r] - \frac{1}{2} \geq \frac{1}{n^d},$$

contradicting to Goldreich-Levin Theorem.

---

Since  $f$  is a permutation  $r, f^{n^c-i}(x) \odot r, \dots, f^1(x) \odot r, x \odot r$  is the same distribution as  $r, f^{n^c}(x) \odot r, \dots, f^{i+1}(x) \odot r, f^i(x) \odot r$ .

one-way function  $\Leftrightarrow$  pseudorandom generator  $\Leftrightarrow$  unpredictability

## Application: Tossing Coin Over Phone

Suppose  $A$  and  $B$  want to toss a coin over phone. We can apply the following protocol.

---

1.  $A$  chooses  $x, r \in_R \{0, 1\}^n$  and sends  $(f_n(x), r)$  to  $B$ , where  $f_n$  is a one-way permutation known to both parties.
  2.  $B$  chooses  $b \in_R \{0, 1\}$  and sends it to  $A$ .
  3.  $A$  sends  $x$  to  $B$ .
  4.  $A$  and  $B$  agree to use  $b \oplus (x \odot r)$ .
- 

$A$  cannot manipulate the result because it cannot change  $x$ .

$B$  cannot manipulate the result because it did not know  $x$ .

$A$  can make sure that the result is random as long as  $x$  is.

$B$  can make sure that the result is random as long as  $b$  is.



# Zero Knowledge Proof

In an interactive proof for an **NP** problem, say GI, a certificate usually reveals more than the mere fact that a statement is true.

It turns out that it is possible to design an interactive proof system such that a verifier does not learn anything from interaction apart from the fact that a statement is true.

- ▶ In the following definition, Perfect Zero Knowledge requires that no matter what a verifier learns after participating a proof for a statement  $x$ , it could have derived the same thing by itself without participating in any interaction.

---

1. S. Goldwasser, S. Micali and C. Rackoff The Knowledge Complexity of Interactive Proof Systems. STOC, 186-208, 1985.



# Zero Knowledge Proof of **NP** Language

Suppose  $L \in \mathbf{NP}$  and  $\mathbb{M}$  is a P-time TM such that  $x \in L$  if and only if

$$\exists u \in \{0, 1\}^{p(|x|)}. \mathbb{M}(x, u) = 1$$

for some polynomial  $p$ .

# Zero Knowledge Proof of **NP** Language

A pair  $\mathbb{P}, \mathbb{V}$  of interactive P-time PTM's is called a **zero knowledge proof** for  $L$  if they enjoy the following properties.

- ▶ **Completeness.** If  $\mathbb{M}(x, u) = 1$ , then  $\Pr[\text{out}_{\mathbb{V}}(\mathbb{P}(x, u), \mathbb{V}(x))] \geq \frac{2}{3}$ .
- ▶ **Soundness.** If  $x \notin L$ , then  $\Pr[\text{out}_{\mathbb{V}}(\mathbb{P}^*(x, u), \mathbb{V}(x))] \leq \frac{1}{3}$  for all  $\mathbb{P}^*$  and  $u$ .
- ▶ **Perfect Zero Knowledge.** For every P-time interactive PTM  $\mathbb{V}^*$  there is an **expected** P-time PTM  $\mathbb{S}^*$ , called a **simulator**, such that for every  $x \in L$  and a certificate  $u$  for  $x \in L$ , the following holds:

$$\text{out}_{\mathbb{V}^*}(\mathbb{P}(x, u), \mathbb{V}^*(x)) \equiv \mathbb{S}^*(x),$$

meaning that the two random variables are identical even though  $\mathbb{S}^*$  does not have any access to  $u$ .

The idea of **simulation** to demonstrate security is central to many aspects of cryptography.

# Zero Knowledge Proof for Graph Isomorphism

**Public Input:**  $G_0, G_1$  with  $n$  vertices.

**$\mathbb{P}$  knows:** A permutation  $\pi \in [n] \rightarrow [n]$  such that  $G_1 = \pi(G_0)$ .

---

**$\mathbb{P}$**  sends  $H = \pi'(G_1)$  with a random permutation  $\pi' \in_R [n] \rightarrow [n]$ .

**$\mathbb{V}$**  sends a random bit  $b \in_R \{0, 1\}$ .

**$\mathbb{P}$**  sends  $\pi'' = \begin{cases} \pi', & \text{if } b = 1, \\ \pi'\pi, & \text{if } b = 0. \end{cases}$

**$\mathbb{V}$**  checks if  $H = \pi''(G_b)$ .

---

If  $G_0 \simeq G_1$ ,  **$\mathbb{V}$**  accepts with probability one.

If  $G_0 \not\simeq G_1$ ,  **$\mathbb{V}$**  rejects with probability  $\frac{1}{2}$ .

# Zero Knowledge Proof for Graph Isomorphism

Let  $\mathbb{V}^*$  be some verifier's strategy.

---

- ▶ If  $G_0 \simeq G_1$ , then  $\mathbb{P}$ 's first message has the same distribution as the message sent by the following simulator  $\mathbb{S}^*$ :
    - ▶ Generate  $b' \in_{\mathbb{R}} \{0, 1\}$  and  $\pi' \in_{\mathbb{R}} [n] \rightarrow [n]$ ;
    - ▶ Send  $H = \pi'(G_{b'})$  to  $\mathbb{V}^*$ ;
    - ▶ Get some  $b$  from  $\mathbb{V}^*$ ;
    - ▶ If  $b = b'$  then send  $\pi'$  to  $\mathbb{V}^*$  and output whatever  $\mathbb{V}^*$  outputs, otherwise restart  $\mathbb{S}^*$ .
  - ▶ The key point is that  $H$  reveals nothing about  $b'$  if  $G_0 \simeq G_1$ .
- 

If  $\mathbb{V}^*$  runs in P-time, then  $\mathbb{S}^*$  runs in expected P-time.



# Secure Multiparty Computation

Ten people working in a firm want to calculate their average salaries without revealing the salary of any of them.

Remark

Cryptography on weaker assumption (say  $\mathbf{P} \neq \mathbf{NP}$ ) ?

Modern cryptography is founded on something not provable.