# Perfect secrecy. One-time pad. PRGs.

Elena Kirshanova

Course “Information and Network Security”
Lecture 2
March 3, 2020

## Some defs/conventions/notations

### Polynomial time

An algorithm $\mathcal{A}$ is called *polynomial-time*, if for any input $n$, the runtime of $\mathcal{A}$ is $\mathcal{O}(n^k)$ for any fixed $k$.

Examples:

- multiplying two $n$-bit numbers: $\mathcal{O}(n \log n)$ – polynomial time
- factoring an $n$-bit number: $\exp(\mathcal{O}(n^{1/3} \cdot (\log n)^{2/3})$ – subexponential time

## Some defs/conventions/notations

### Polynomial time

An algorithm $\mathcal{A}$ is called *polynomial-time*, if for any input $n$, the runtime of $\mathcal{A}$ is $\mathcal{O}(n^k)$ for any fixed $k$.

Examples:

- multiplying two $n$-bit numbers: $\mathcal{O}(n \log n)$ – polynomial time
- factoring an $n$-bit number: $\exp(\mathcal{O}(n^{1/3} \cdot (\log n)^{2/3})$ – subexponential time

### Probabilistic Polynomial time

An algorithm $\mathcal{A}$ is called *probabilistic polynomial-time* (ppt), if it is polynomial time and uses randomness.

## Some defs/conventions/notations

### Negligible function

A function $f : \mathbb{N} \to \mathbb{R}$ is called *negligible* if for all polynomials $p$ there exists $N \in \mathbb{N}$ so that for any $n \geq N$

$$f(n) < \frac{1}{p(n)}.$$

Examples:

- negligible functions:

$$\frac{1}{2^n}, \ \frac{1}{2^{\sqrt{n}}}, \ \frac{1}{2^{\log^2(n)}}$$

- non-negligible functions:

$$\frac{1}{\log n}, \ \frac{1}{n^2}, \ \frac{1}{2^{\mathcal{O}(\log n)}}$$

## Basic principles of modern cryptography

1. **Kerckhoffs' principle (1883)**: A cryptosystem should remain secure if everything about it, except the key, is publicly known.
2. An adversary cannot derive any information about a plaintext from the corresponding ciphertext
3. The attack model (i.e., what an adversary is allowed to do) must be clearly specified.
   Example: we restrict adversaries to be ppt algorithms
4. The security assumption must as well be clearly specified.

# Basic principles of modern cryptography

1. **Kerckhoffs' principle (1883)**: A cryptosystem should remain secure if everything about it, except the key, is publicly known.
2. An adversary cannot derive any information about a plaintext from the corresponding ciphertext
3. The attack model (i.e., what an adversary is allowed to do) must be clearly specified.
   Example: we restrict adversaries to be ppt algorithms
4. The security assumption must as well be clearly specified.

Security statement:

Under the assumption $X$, the construction $\Pi$ is secure in the $Y$ security model.

## Symmetric Encryption

Let $\lambda$ be a security parameter and $\mathcal{M}, \mathcal{K}, \mathcal{C}$ be resp. the plaintext, key, ciphertext spaces. A *Symmetric Encryption Scheme* $\Pi = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ consists of three ppt algorithms:

- $\mathsf{KeyGen}(1^\lambda) : k \leftarrow \mathcal{K}$
- $\mathsf{Enc}(k, m \in \mathcal{M}) : c \leftarrow \mathsf{Enc}(k, m)$
- $\mathsf{Dec}k, c \in \mathcal{C} \, : \, m' \leftarrow \mathsf{Dec}(k, c)$

## Symmetric Encryption

Let $\lambda$ be a security parameter and $\mathcal{M}, \mathcal{K}, \mathcal{C}$ be resp. the plaintext, key, ciphertext spaces. A *Symmetric Encryption Scheme* $\Pi = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ consists of three ppt algorithms:

- $\mathsf{KeyGen}(1^\lambda) : k \leftarrow \mathcal{K}$
- $\mathsf{Enc}(k, m \in \mathcal{M}) : c \leftarrow \mathsf{Enc}(k, m)$
- $\mathsf{Dec} k, c \in \mathcal{C} : m' \leftarrow \mathsf{Dec}(k, c)$

The scheme $\Pi$ is called *correct* if for all $k \in \mathcal{K}, m \in \mathcal{M}$ :

$$\mathsf{Dec}(k, \mathsf{Enc}(k, m)) == m.$$

# Perfect Secrecy

- Let $\mathcal{M}, \mathcal{K}, \mathcal{C}$ be resp. the plaintext, key, ciphertext spaces
- Let $M \in \mathcal{M}$ be a random variable distributed according to a distribution defined over $\mathcal{M}$: $m$ is taken with probability $\Pr[M = m]$.
- Analogously for $K \in \mathcal{K}, C \in \mathcal{C}$.

# Perfect Secrecy

- Let $\mathcal{M}, \mathcal{K}, \mathcal{C}$ be resp. the plaintext, key, ciphertext spaces
- Let $M \in \mathcal{M}$ be a random variable distributed according to a distribution defined over $\mathcal{M}$: $m$ is taken with probability $\Pr[M = m]$.
- Analogously for $K \in \mathcal{K}, C \in \mathcal{C}$.

**Perfect secrecy**

An encryption scheme $\Pi = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ is called *perfectly secure* if for any distribution over $\mathcal{M}$

$$\Pr\left[M = m | C = c\right] = \Pr\left[M = m\right] \quad \forall m \in \mathcal{M}, c \in \mathcal{C}.$$

Equivalent definition:

$$\Pr\left[\mathsf{Enc}(k, m_0) = c\right] = \Pr\left[\mathsf{Enc}(k, m_1) = c\right] \quad \forall m_0, m_1 \in \mathcal{M}, c \in \mathcal{C}.$$

# One-time pad (or Vernam cipher)

### One-time pad

Let $\mathcal{M}, \mathcal{K}, \mathcal{C} = \{0,1\}^n$ s.t. $n = \lambda$.

- $\mathsf{KeyGen}(1^\lambda) : k \leftarrow \{0,1\}^n$
- $\mathsf{Enc}(k, m \in \{0,1\}^n) : c = k \oplus m$
- $\mathsf{Dec}(k, c \in \{0,1\}^n) : m = k \oplus c$

### Theorem

*OTP is perfectly secure in the Ciphertext-only Attack (COA) model (i.e., an adversary sees only ciphertexts).*

# Price to pay for perfect secrecy

## Size of $\mathcal{K}$

Let $\Pi$ be perfectly secure. Then

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

## Shannon's theorem

Let $\Pi = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ satisfy $|\mathcal{K}| = |\mathcal{M}| = |\mathcal{C}|$. Then $\Pi$ is perfectly secure iff

1. KeyGen chooses $k \leftarrow$ uniformly at random with prob. $\frac{1}{|\mathcal{K}|}$
2. For all $m \in \mathcal{M}$, $c \in \mathcal{C}$, $\exists!\ k \in \mathcal{K} :\ c = \mathsf{Enc}(k, m)$.

See proofs in Katz&Lindell *Introduction to Modern Cryptography*

# Computational Security

## Perfect Secrecy

- Information-theoretic (strong) security against **unbounded** adversary
- Impractical key space size

## Computational Security

- We usually use keys of sizes $128, 256$ bits
- Security against **ppt** adversaries
- Unbounded adversary with access to plaintext-ciphertext pairs $(m_i, c_i)$ can launch an exhaustive search for $k \in \mathcal{K}$ s.t. $\text{Enc}(k, m_i) == c_i \ \forall i$.

## Pseudorandom Generators (PRGs)

Idea: 'Stretch' truly random $\ell$-bits seed $s$ into a longer $L$-bits 'random looking' string Define

$$\begin{aligned} G : \{0,1\}^{\ell} \to \{0,1\}^{L} : \\ s \mapsto G(s) \end{aligned}$$

**Intuition:** an ppt adversary cannot tell the difference between $G(s)$ and $r \leftarrow \{0,1\}^{L}$.

# Statistical tests

A **ls $\Pi$ statistical test** on $\{0,1\}^n$ is an algorithm $A$ s.t. $A(x)$ outputs $0$ (="not random") or $1$ (="random")

1. $A(x) = 1 \quad |\#0(x) - \#1(x)| \leq 10\sqrt{n}$
2. $A(x) = 1 \quad |\#00(x) - n/4| \leq 10\sqrt{n}$
3. $A(x) = 1 \quad \max \mathsf{len}\{1...1(x)\} \leq 10 \log n$

**Formal definition**

A PRG $G$ is an efficient deterministic algorithm that given a seed $s \in \mathcal{S} = \{0,1\}^\ell$ outputs an $r \in \mathcal{R} = \{0,1\}^L$

**Attack game for PRG:**

**Experiment 0:** The Challenger computes $r \in \mathcal{R}$ as

$$s \leftarrow \mathcal{S}$$
$$r \leftarrow G(s)$$

# Secure PRG

## Formal definition

A PRG $G$ is an efficient deterministic algorithm that given a seed $s \in \mathcal{S} = \{0,1\}^\ell$ outputs an $r \in \mathcal{R} = \{0,1\}^L$

**Attack game for PRG:**

**Experiment 0:** The Challenger computes $r \in \mathcal{R}$ as

$$s \leftarrow \mathcal{S}$$
$$r \leftarrow G(s)$$

**Experiment 1:** The Challenger computes $r \in \mathcal{R}$ as

$$r \leftarrow \{0,1\}^L$$

Let $W_b$ be the event that $\mathcal{A}$ outputs $b$.
$\mathcal{A}$'s advantage: $\mathsf{PRGadv}\,[\mathcal{A}, G] = |\Pr[W_0] - \Pr[W_1]|$.

A PRG $G$ is **secure** if PRGadv is negligible for all ppt $\mathcal{A}$.

# Stream Cipher = OTP with keys output by a PRG

Let $G : \{0,1\}^\ell \to \{0,1\}^L$

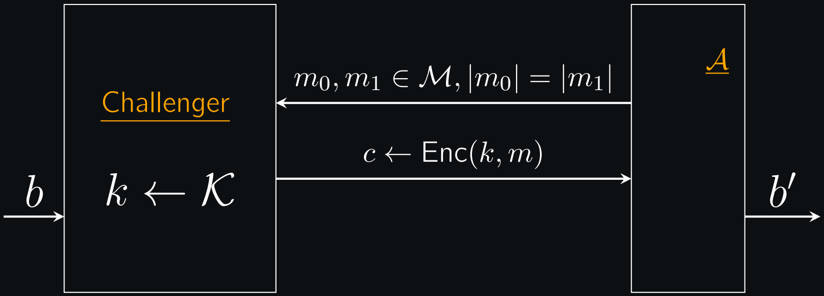
The **stream cipher** $\Pi = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ is defined over

- $\mathcal{K} = \{0,1\}^\ell$
- $\mathcal{M} = \{0,1\}^L$
- $\mathcal{C} = \{0,1\}^L$

For $s \in \{0,1\}^\ell$

- $\mathsf{Enc}(s, m \in \{0,1\}^L) : c = G(s) \oplus m$
- $\mathsf{Dec}(s, c \in \{0,1\}^L) : m = G(s) \oplus c$

# Semantic Security




Let $W_b$ be the event that $\mathcal{A}$ outputs $b$.
$\mathcal{A}$'s advantage: $\mathsf{Adv}\,[\mathcal{A}, \mathsf{Enc}] = |\Pr[W_0] - \Pr[W_1]|$.

$\Pi$ is **semantically secure** if $\mathsf{PRGadv} = \mathrm{negl}(\lambda)$ for all ppt $\mathcal{A}$.

## Semantic Security of $\Pi$

**Theorem**

*If $G$ is a secure PRG, then the stream cipher $\Pi$ constructed from $G$ is semantically secure.*

**Proof strategy:** for any adversary $\mathcal{A}$ against semantic security, $\exists$ an adversary $\mathcal{B}$ against $G$.

# QUESTION!

Let $G$ be a PRG that the last bit of the output is always 0.

Let $\Pi$ be a stream cipher constructed from $G$.

**Is $\Pi$ semantically secure?**

# Constrictions of a PRG: Salsa and ChaCha

- Salsa20,ChaCha20: proposed by D.Bernstein in 2005, 2008
- used in many TLS cipher suits
- Input: $256$-bit seed and a parameter $L$
- Output: $(256 \cdot L)$-bit pseudorandom string

## Constrictions of a PRG: Salsa and ChaCha

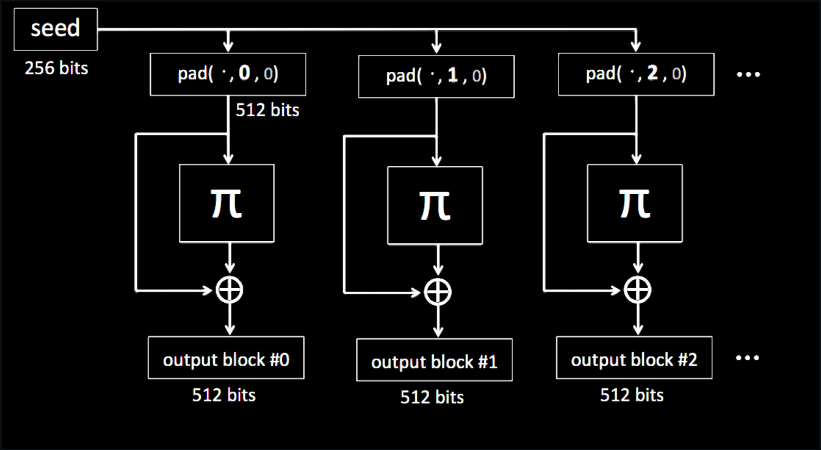
- Salsa20,ChaCha20: proposed by D.Bernstein in 2005, 2008
- used in many TLS cipher suits
- Input: $256$-bit seed and a parameter $L$
- Output: $(256 \cdot L)$-bit pseudorandom string

Two components

1. pad$(s, j, 0)$: takes a seed $s$, a $64$-bit counter $j$ and a $64$-bit nonce
   Output: $512$-bit block
2. a fixed public permutation $\pi : \{0,1\}^{512} \to \{0,1\}^{512}$

See `https://cr.yp.to/chacha.html` for details

# ChaCha PRG




Nonce – the third parameter of **pad**$(s, j, 0)$ is used to convert a PRG into a PRF (useful for encryption of multiple messages).

picture is taken from D.Boneh, V.Shoup A Graduate Course in Applied Cryptography

## (Somewhat) Broken PRGs

1. **linear congruential generators**
   - had been used in glibc, Microsoft Visual Basic, Java
   - notorious for RANDU
   - **not cryptographically secure PRG!**

2. **RC4**
   - proposed by R.Rivest in 1987
   - used to be a part of TLS, 802.11b WEP
   - **not cryptographically secure PRG!**

3. **Linear feedback shift registers**
   - used for protecting movies on DVD disks
   - weakly security PRG (Trivium)

## A Random Number Generator

- In practice, random bits are generated using a random number generator, RNG
- An RNG outputs a sequence of pseudorandom bits
- Unlike PRG, an RNG take additional input (entropy source)
- Example in Linux: /**dev**/**random**
- Entropy is usually taken from hardware (keyboard/mouse events, hardware interrupts, jitters).

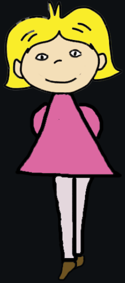
## Application: Coin flipping

Task: throw a fair coin over without interaction

$$G : \{0,1\}^\ell \to \{0,1\}^L$$

Bob

Alice



$r \leftarrow \{0,1\}^L$

$\xleftarrow{r}$

Flips a coin
$b \in \{0,1\}$
$s \in \{0,1\}^\ell$

$$\text{commit}(b,r,s) = \begin{cases} G(s), & b = 0 \\ G(s) \oplus r, & b = 1 \end{cases}$$

$\xrightarrow{\text{commit}(b,r,s)}$