# Stream Ciphers

Stanislav Palúch

Fakula riadenia a informatiky, Žilinská univerzita

25. októbra 2017

## Computational, Unconditional and Perfect Security

### Definition (Computational Security of a Cryptosystem.)

We will say that a cryptosystem is computationally secure if the best known breaking algorithm requires at least $N$ steps where $N$ is a specified large number.
Another attitude:
We will say that a cryptosystem is computationally secure if problem of its breaking is polynomially equivalent to some NP-hard problem.

### Definition (Unconditional security of a Cryptosystem.)

We will say that a cryptosystem is unconditionally secure if it is unbreakable even in the case where an adversary has unlimited resources for solving a problem e.g. breaking a cipher.

### Definition (Perfect Security of a Cryptosystem.)

We will say that a cryptosystem $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ has a perfect secresy if conditional probablility of the event plaintext $x \in \mathcal{P}$ was transmitted under the condition Ciphertext $y \in \mathcal{C}$ was received, is equal to probablity of transmitting plaintext $x$, i.e.

$$P[M = x | C = y] = P[M = x].$$

## Ceasar Cipher Applied to One Character Has a Perfect Security

Suppose we have a cryptosystem $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$
where $\mathcal{P} = \mathbb{Z}_{26}$, $\mathcal{C} = \mathbb{Z}_{26}$, $\mathcal{K} = \mathbb{Z}_{26}$, $E_k(x) = x \oplus k$, $D_k(y) = y \ominus k$.
Let probability distribution of keys is uniform probability distribution i.e.

$$\forall k \in \mathbb{Z}_{26} \quad P[K = k] = \frac{1}{26}.$$

$$P[C = y] = \sum_{k \in \mathcal{K}} P[K = k].P[M = d_k(y)] =$$

$$\sum_{k \in \mathcal{K}} P[K = k].P[M = (y \ominus_{26} k)] =$$

$$\sum_{k \in \mathcal{K}} \frac{1}{26}.P[M = (y \ominus_{26} k)] = \frac{1}{26}.\underbrace{\sum_{k \in \mathcal{K}} P[M = (y \ominus_{26} k)]}_{=1} = \frac{1}{26}$$

$$P[C = y] = \frac{1}{26}$$

## Ceasar Cipher Applied to One Character Has a Perfect Security

$$P[C = y|M = x] = P[K = y \ominus_{26} x] = \frac{1}{26}$$

$$P[M = x|C = y] = \underbrace{\frac{P[M = x].P[C = y|M = x]}{P[C = y]}}_{\text{Bayes theorem: } P(A|B) = \frac{P(A).P(B|A)}{P(B)}} = \frac{P[M = x].\frac{1}{26}}{\frac{1}{26}} = P[M = x]$$

### Theorem

*Ceasar cipher applied to one character has a perfect security if every time another key with uniform probablity distribution is used.*

### Theorem

*Let $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ be a cryptosystem where $|\mathcal{P}| = |\mathcal{C}| = |\mathcal{K}|$. Then this cryptosystem has a peerfect security if and only if every key is used with the same probability $1/|\mathcal{K}|$ and if and only if for every $x \in \mathcal{P}$ a and for every $y \in \mathcal{C}$ there exists exactly one key $k \in \mathcal{K}$ such that $y = e_k(x)$.*

## *Stream cipher*

Principle of a stream cipher:
$x_1, x_2, \dots, x_n, \dots$ – stream of characters of plaintext
$k_1, k_2, \dots, k_n, \dots$ – stream of keys

Corresponding stream of characters of ciphertext is:

$$y_1, y_2, \dots, y_n, \dots = E_{k_1}(x_1), E_{k_2}(x_2), \dots, E_{k_n}(x_n), \dots$$

Ceasar and Vigenèr cihers can be slightly changed in such a way that characters of alphabet $\mathbb{Z}_{26}$ will be represented by a binary code (e.g. 5-bit binarycode of ASCII code) operation $\oplus$ will be replaced by bitwise operation XOR denoted by $\otimes$.

| XOR | 0 | 1 |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 1 | 0 |

Potom

$$E_k(x) = x \otimes k \quad \text{a} \quad D_k(y) = y \otimes k$$

Binary operation $\otimes$ is a associative and commutative operation on $\mathbb{Z}_{26}$ and $(\mathbb{Z}_{26}, \otimes)$ is a commutative group with neutral element 0.

## One time pad - Vernam Cipher

The one time pad (OTP) is a stream cipher which requires the use of a one-time pre-shared key of the same size as the message being sent.

If the key is truly random and is never reused, then the resulting ciphertext will be impossible to decrypt or break.

The one time pad was first described by Frank Miller in 1882, and was re-invented in 1917.

In 1919, U.S. Patent 1,310,719 was issued to Gilbert S. Vernam for one time pad cipher.

We wil use alphabet equal to $A = \mathbb{Z}_2$ – characters will bee 0 or 1.

The set of all plaintexts is the set of all finite sequences of elements of $A = \mathbb{Z}_2$.
The set of all keys is the set of all finite sequences of elements of $A = \mathbb{Z}_2$.
The set of all ciphertexts is the set of all finite sequences of elements of $A = \mathbb{Z}_2$.

## *One time pad - Vernam Cipher*

Let

$x_1, x_2, \dots, x_n, \dots$ – be a stream of characters of plaintext

$k_1, k_2, \dots, k_n, \dots$ – be a stream of keys, $P(k_i = 0) = P(k_i = 1) = \dfrac{1}{2}$

$y_1, y_2, \dots, y_n, \dots$ – be a stream of characters of ciphertext

**Enciphering procedure:**

$$\begin{array}{cccccc}
x_1, & x_2, & x_3, & \dots, & x_i, & \dots \\
k_1, & k_2, & k_3, & \dots, & k_i, & \dots \\
y_1 = x_1 \otimes k_1, & y_2 = x_2 \otimes k_2, & y_3 = x_3 \otimes k_3, & \dots, & y_i = x_i \otimes k_i, & \dots
\end{array}$$

**Deciphering procedure:**

$$\begin{array}{cccccc}
y_1, & y_2, & y_3, & \dots, & y_i, & \dots \\
k_1, & k_2, & k_3, & \dots, & k_i, & \dots \\
y_1 = y_1 \otimes k_1, & y_2 = y_2 \otimes k_2, & y_3 = y_3 \otimes k_3, & \dots, & y_i = y_i \otimes k_i, & \dots
\end{array}$$

If keys $k_1, k_2, \dots, k_n, \dots$ are chosen by chance with uniform probability distribution and i never reused, then there is no chance to break Vernam cipher.

## Attack against Repeated Usage of the Same Key Stream

Drawbacks of Vernam cipher:

- Stream of keys shoule be as long as plaintext
- Stream of key can be used only once

**Attack against Repeated Usage of the Same Key Stream**

Suppose that two plaintext sequences

$$a_1, a_2, \dots, a_n, \dots, \quad b_1, b_2, \dots, b_n, \dots$$

were enciphered by the same stream of keys: $k_1, k_2, \dots, k_n, \dots$.

Cryptanalyst gets two ciphertexts $y_1, y_2, \dots, y_n, \dots,\ z_1, z_2, \dots, z_n, \dots$ such that

$$y_i = a_i \otimes k_i, \quad z_i = b_i \otimes k_i.$$

He calculates a sequence $w_1, w_2, \dots, w_n, \dots,$ where $w_i = y_i \otimes z_i$.
It holds:

$$w_i = y_i \otimes z_i = (a_i \otimes k_i) \otimes (b_i \otimes k_i) = (a_i \otimes b_i) \otimes (k_i \otimes k_i) = (a_i \otimes b_i) \otimes 0 = (a_i \otimes b_i)$$

$$w_i = (a_i \otimes b_i)$$

## Attack against Repeated Usage of the Same Key Stream

$$w_i = (a_i \otimes b_i) \tag{1}$$

The stream $w_1, w_2, \ldots$ is a sequence of characters of plaintext enciphered by the sequence of characters of another plaintext. Such sequnce carries with itself enough information for revealing significant part of both plaintexts (and in final concequence also revealing of key stream)

**Synchronization of Two Ciphertexts Enciphered with the Same Key Stream**

Cryptanalyst taps two ciphertexts enciphered by Vernam cipher by the same key stream:

$$y_1, y_2, \ldots, y_n, \ldots, \quad z_1, z_2, \ldots, z_n, \ldots,$$

but those ciphertexts are shifted one against another by $d$ positions, i.e.

$$y_i = a_i \otimes k_{i+d}, \quad z_i = b_i \otimes k_i.$$

If he calculates a sequence

$$w_i = y_i \otimes z_i = (a_i \otimes k_{i+d}) \otimes (b_i \otimes k_i) = (a_i \otimes b_i) \otimes (k_{i+d} \otimes k_i),$$

thie sequence will be appear as a equence of random bites.

## Synchronization of Two Ciphertexts

However, if he shifts ciphertext $z_1, z_2, \dots, z_n, \dots,$ against ciphertext $y_1, y_2, \dots, y_n, \dots$ by $d$ positions backwards and calculates a sequence

$$w_i = y_i \otimes z_{i+d} = (a_i \otimes k_{i+d}) \otimes (b_{i+d} \otimes k_{i+d}) = (a_i \otimes b_{i+d}) \otimes (k_{i+d} \otimes k_{i+d}) = a_i \otimes b_{i+d},$$

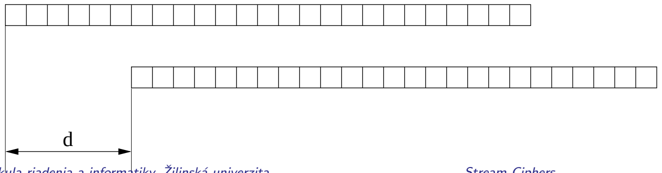
$$w_i = a_i \otimes b_{i+d}, \tag{2}$$

he gets one plaintext enciphered by another plaintext what makes it possible to reveal substantial parts of both plaintexts.

Correct offset $d$ can be determined by the fact that relative number of zeros in the sequence $\{w_i\}_{i=1,2,\dots}$ significantly raises in the case that both ciphertexts are correctly synchronized.

Probability of zero in the case or correct synchronization is equal to the probability of the event that $a_i = b_{i+d}$ wha is equal to correspondent coefficient of coincidence.

Cryptanalyst shifts both ciphertext one against another. If offset $d$ is correct, the numbe of equalities on same positions significantly raises.



## Generation of Genuine Random Number Sequences

The best way how to obtain a genuine sequence of random number is measuring a real physical phenomenon which is sufficiently random, e.g.

- output of a Gieiger-Muller couter
- measuring irregularitie of busy server
- measuring temperature
- measuring of Johnson noise of a resistor or a semiconductor

Result of just mentioned ways are random but not necessarily uniformly distributed.
One wah how to equalize probabilities of 0 and 1 is as follows:

$$\underbrace{00}_{-}|\underbrace{00}_{-}|\underbrace{10}_{1}|\underbrace{11}_{-}|\underbrace{01}_{0}|\underbrace{01}_{0}|\underbrace{00}_{-}|\underbrace{11}_{-}|\underbrace{10}_{1}|\underbrace{00}_{-}|\underbrace{10}_{1}|$$

Another way is:
Assume that $P(k_i = 0) = 1/2 + \epsilon$, $P(k_i = 1) = 1/2 - \epsilon$.
Set $z_i = k_{2i} \otimes k_{2i+1}$.

$$P(z_i = 0) = P(k_{2i} = 0).P(k_{2i+1} = 0) + P(k_{2i} = 1).P(k_{2i+1} = 1) =$$

$$\left(\frac{1}{2} + \epsilon\right)^2 + \left(\frac{1}{2} - \epsilon\right)^2 = \frac{1}{2} + 2\epsilon^2$$

## Pseudorandom Number Generators

Linear congruence generator

$$X_n = (aX_{n-1} + b) \mod m$$

Period max $m - 1$.

Quadratic congruence generator

$$X_n = (aX_{n-1}^2 + bX_{n-1} + c) \mod m$$

Cubuc congruence generatorr

$$X_n = (aX_{n-1}^3 + bX_{n-1}^2 + cX_{n-1} + d) \mod m$$

Joan Boyar proved, that linear and other congruence enerators are cryptographically weak.
**Random generators implemented in programm languages are not destined for use in cryptography.**
**They must not be used in this connectioni!!**

## Generator RC4

RC4 was designed by Ron Rivest of RSA Security in 1987. RC4 was initially a trade secret.

In September 1994 a description of it was anonymously posted to the Cypherpunks mailing list.

The leaked code was confirmed to be genuine as its output was found to match that of proprietary software using licensed RC4.

RC4 became part of some commonly used encryption protocols and standards, such as WEP, SSL and its TLS, until it was prohibited for all versions of TLS by RFC 7465 in 2015, due to the RC4 attacks weakening or breaking RC4 used in SSL/TLS.

RC4 uses 256 S-boxes $S[0], S[1], \dots, S[255]$ containing a permutation of numbers $0 - 255$. Moreover it uses two pointers $i$, $j$ used as indexes into array $S[\,]$.

```
rand()
i=i+1 mod 256
j=j+S[i] mod 256
swap(S[i],S[j])
t=(S[i]+S[j]) mod 256
k=S[t]
return k
```

## Initialization Procedure for RC4

Key can be long up to 256*8=2048 bits. These bits will be repeatidly used to fill all 8-bit numbers $K[0], K[1], \ldots, K[255]$.

Initialization procedure is az follows

```
for i=0 to 255
{
S[i]=i
}
j=0
for i=0 to 255
{
j=(j+S[i]+K[i]) mod 256
swap(S[i],S[j])
}
i=0
j=0
```

There are similar pseudo random generators designed as VMPC.