One-Time Pad or Vernam Cipher

• The one-time pad, which is a provably secure cryptosystem, was developed by Gilbert Vernam in 1918.
• The message is represented as a binary string (a sequence of 0’s and 1’s using a coding mechanism such as ASCII coding.
• The key is a truly random sequence of 0’s and 1’s of the same length as the message.
• The encryption is done by adding the key to the message modulo 2, bit by bit. This process is often called exclusive or, and is denoted by XOR. The symbol $\oplus$ is used.

<table>
<thead>
<tr>
<th>$a$</th>
<th>$b$</th>
<th>$c = a \oplus b$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
One-Time Pad or Vernam Cipher

Example: Let the message be IF then its ASCII code be (1001001 1000110) and the key be (1010110 0110001). The ciphertext can be found exoring message and key bits

Encryption:

1001001 1000110 plaintext
1010110 0110001 key
0011111 1110110 ciphertext = ( v)

Decryption:

0011111 1110110 ciphertext
1010110 0110001 key
1001001 1000110 plaintext
Why One-Time Pad is provably secure?

Or how can we prove it is unbreakable?

• The security depends on the randomness of the key.
• It is hard to define randomness.
• In cryptographic context, we seek two fundamental properties in a binary random key sequence:
  1. Unpredictability: Independent of the number of the bits of a sequence observed, the probability of guessing the next bit is not better than \( \frac{1}{2} \). Therefore, the probability of a certain bit being 1 or 0 is exactly equal to \( \frac{1}{2} \).
  2. Balanced (Equal Distribution): The number of 1’s and 0’s should be equal.
Mathematical Proof

- the probability of a key bit being 1 or 0 is exactly equal to $\frac{1}{2}$.
- The plaintext bits are not balanced. Let the probability of 0 be $x$ and then the probability of 1 turns out to be $1-x$.
- Let us calculate the probability of ciphertext bits.

<table>
<thead>
<tr>
<th>$m_i$</th>
<th>prob.</th>
<th>$k_i$</th>
<th>prob.</th>
<th>$c_i$</th>
<th>prob.</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$x$</td>
<td>0</td>
<td>$\frac{1}{2}$</td>
<td>0</td>
<td>$\frac{1}{2}x$</td>
</tr>
<tr>
<td>0</td>
<td>$x$</td>
<td>1</td>
<td>$\frac{1}{2}$</td>
<td>1</td>
<td>$\frac{1}{2}x$</td>
</tr>
<tr>
<td>1</td>
<td>$1-x$</td>
<td>0</td>
<td>$\frac{1}{2}$</td>
<td>1</td>
<td>$\frac{1}{2}(1-x)$</td>
</tr>
<tr>
<td>1</td>
<td>$1-x$</td>
<td>1</td>
<td>$\frac{1}{2}$</td>
<td>0</td>
<td>$\frac{1}{2}(1-x)$</td>
</tr>
</tbody>
</table>

- We find out the probability of a ciphertext bit being 1 or 0 is equal to $(\frac{1}{2})x + (\frac{1}{2})(1-x) = \frac{1}{2}$. Ciphertext looks like a random sequence.
A Practical One-Time Pad

- A satellite produces and broadcasts several random sequences of bit at a rate fast enough such that no computer can store more than a very small fraction of them.
- Alice & Bob use a PKC to agree on a method of sampling bits from these random sequences.
- They use these bits to form a key for one-time pad.
- Eve, in theory, can break the PKC they used even though doing so is difficult.
- But by the time she breaks it, random bits Alice & Bob collected disappeared and Eve can not decrypt the message since she hasn’t got the resources to store all the random bits that have been broadcast.
• Symmetric-key ciphers
• Encrypt individual characters at a time,
• Faster and less complex in hardware,
• They are desirable in some applications in which
  • buffering is limited
  • bits must be individually processed as they are received.
  • Transmission errors are highly probable.
• Vast amount of theoretical knowledge.
• Various design principles.
• Widely being used at present, will probably be used in the future.
Basic Idea comes from **One-Time-Pad** cipher,

Encryption : \[ c_i = m_i \oplus k_i \quad i = 1,2,3,... \]
- \( m_i \) : plain-text bits.
- \( k_i \) : key (key-stream) bits
- \( c_i \) : cipher-text bits.

Decryption : \[ m_i = c_i \oplus k_i \quad i = 1,2,3,... \]
- Provably Secure.

**Drawback** : Key-stream should be as long as plain-text. Key distribution & Management difficult.

**Solution** : Stream Ciphers (in which key-stream is generated in pseudo-random fashion from relatively short secret key.)
**Randomness**: Closely related to *unpredictability*.

**Pseudo-randomness**: PR sequences appears random to a computationally bounded adversary.

- Stream Ciphers can be modeled as Finite-state machine.

\[ S_{i+1} \rightarrow S_i \rightarrow F \rightarrow G \rightarrow k_i \rightarrow m_i \rightarrow c_i \]

- \( S_i \): state of the cipher at time \( t = i \).
- \( F \): state function.
- \( G \): output function.

Initial state, output and state functions are controlled by the secret key.
1. Synchronous Stream Ciphers

- Key-stream is independent of plain and cipher-text.
- Both sender & receiver must be synchronized.
- Resynchronization can be needed.
- No error propagation.
- Active attacks can easily be detected.

2. Self-Synchronizing Stream Ciphers

- Key-stream is a function of fixed number $t$ of cipher-text bits.
- Limited error propagation (up to $t$ bits).
- Active attacks cannot be detected.
- At most $t$ bits later, it resynchronizes itself when synchronization is lost.
- It helps to diffuse plain-text statistics.
Analysis

• Efforts to evaluate the security of stream ciphers.

  1. Mathematical Analysis
     • Period and Linear Complexity,
     • Security against Correlation Attacks.

  2. Pseudo-randomness Testing
     • Statistical Tests,
     • Linear Complexity,
     • Ziv-Lempel Complexity
     • Maximum Order Complexity,
     • Maurer’s Universal Test.

• In testing, all the tests are applied to as many key-streams of different lengths as possible.
Linear Feedback Shift Register - LFSR

$C(x) = 1 + c_1 x + c_2 x^2 + \cdots + c_L x^L$ : Connection Polynomial

- If $C(x)$ is primitive, LFSR is called *maximum-length*, and the output sequence is called *m-sequence* and its period is $T = 2^L - 1$.

- *m-sequences* have good statistical properties.
- However, they are predictable.
• If 2L successive bits of an m-sequence are known, the shortest LFSR which produces the rest of the sequence can be found using Berlekamp-Massey (BM) algorithm.
• Generally, the length of the shortest LFSR which generates a sequence is called linear complexity.

Stream Cipher Designs Based on LFSRs

• LFSRs generate m-sequence.
• However, “Linearity is the curse of cryptographer.”
• The methods of utilizing LFSRs as building blocks in the stream cipher design.
• The design principle:
  Use other blocks which introduce non-linearity while preserving the statistical properties of m-sequences.
Nonlinear combination Generators

The Combiner Function should be,
1. Balanced,
2. Highly nonlinear,
3. Correlation Immune.
• Utilizing the *algebraic normal form* of the combiner function we can compute the linear complexity of the output sequence.

**Example (Geffe Generator)** :

\[
F(x_1, x_2, x_3) = x_1 x_2 \oplus x_2 x_3 \oplus x_3
\]

If the lengths of the LFSRs are relatively prime and all connection polynomials are primitive, then

\[
L = L_1 L_2 + L_2 L_3 + L_3
\]

\[
T = (2^{L_1} - 1) \cdot (2^{L_2} - 1) \cdot (2^{L_3} - 1)
\]

When we inspect the truth table of the combiner function we gain more insight about the security of Geffe generator.
The combiner function is balanced.
However, the correlation probability,
\[ P(z = x_1) = \frac{3}{4}. \]
Geffe generator is not secure.
Nonlinear Filter Generator

- Upper bound for linear complexity,

\[ L_m = \sum_{i=1}^{m} \binom{L}{i} \quad m : \text{nonlinear order of the filter function.} \]

- When \( L \) and \( m \) are big enough, the linear complexity will become large.
Clock-controlled Generators

• An LFSR can be clocked by the output of another LFSR.
• This introduces an irregularity in clocking of the first LFSR, hence increase the linear complexity of its output.

Example: Shrinking Generator
• Relatively new design.
• However, it is analyzed and it seems secure under certain circumstances.

\[
\begin{align*}
\text{if } & \gcd(L_s, L_A) = 1 \Rightarrow \\
T &= (2^{L_A} - 1) \cdot 2^{L_s-1} \\
L_A \cdot 2^{L_s-2} &< L < L_A \cdot 2^{L_s-1}
\end{align*}
\]
Different Designs

• SEAL, RC4.
  • They use expanded key tables,
  • Fast in software,
  • Look secure,
  • They have not been fully analyzed yet,
  • Efficient analysis tools are not developed.