Full-Time-Pad
Symmetric Stream Cipher

Improved One-Time-Pad Encryption Scheme

Taha Canturk
kibnakanoto@protonmail.com

2024-05-20

Version 1.0

Abstract

`One-Time-Pad` Encrypion Scheme is a secure algorithm but there are 2 main security risks. One, a key cannot be reused. Two, plaintext length equals key length which is very inefficient when dealing with long plaintexts. These 2 security risks only exist due to a lack of confusion and diffusion per ciphertext. As denoted by Claude Shannon in the report he published in 1945, A Mathematical Theory of Cryptography, A secure cryptographic algorithm requires confusion and diffusion. The `Full-Time-Pad` symmetric stream cipher is developed based on the `One-Time-Pad` with solutions to the security risks while maintaining high speed computation. To achieve diffusion, the key is permutated in it's byte array form using a constant permutation matrix. To acheive the confusion, the key is manipulated in it's 32-bit integer representation using Modular **A**ddition in F_p, Bitwise **R**otations, and **X**or (**ARX**). The permutation garuntees that every time there is a manipulation, eacj 32-bit number is made up of a different byte order.

Contents

1 Introduction

1.1 Pre-requisite Terminology

Symmetric	Same key is used for encryption and decryption
Stream	Plaintext is encrypted without seperating it into blocks
plaintext	plain data before encryption
ciphertext	encrypted plaintext
Cipher	Encryption algorithm. Plaintext is transformed into a ciphertext that can only be reversed with a key

1.2 Applications

1.3 Key Generation

1.4 Prerequisite Mathematics

1.5 Vector Permutation

2 Security Vulnerabilities

2.1 Brute-Force

2.1.1 Birthday Attack

2.1.2 Denial of Service (DoS)

2.2 Reverse Engineering the Transformation

2.3 Collision-Resistance

2.3.1 Different Permutation Matrices

2.3.2 Number of Rounds

2.3.3 Constant - F_p - Prime Galois Field Size

2.3.4 Constant - r - Dynamic Rotation Constant

3 Hashing

3.1 Diffusion - Permutation

3.1.1 Vector Permutation

3.1.2 Dynamic vs. Static

3.2 Dynamic Matrix Permutation

3.2.1 Deravation

Python code is in the test/perm.py

Algorithm 1 Dynamic Permutation Matrix Deravation Pseudo-code

Input: an array of incrementing numbers (0-31) A
Output: Most Efficient Permutation Matrix V (16×32)
Begin
$P \leftarrow$ `copy of A`
for $k = 0$ to 4 **do**
    for $i = 0$ to 8 **do**
        $P_i \leftarrow A_{i\times4}$
        $P_{i+8} \leftarrow A_{i\times4+1}$
        $P_{i+16} \leftarrow A_{i\times4+2}$
        $P_{i+24} \leftarrow A_{i\times4+3}$
    end for
    $A \leftarrow$ `copy of P`
    $V.append(P)$
    $C \leftarrow$ `copy of P`
    for $m = 0$ to 3 **do**
        for $i = 0$ to 8 **do**
            for $n = 0$ to 4 **do**
                $P_{i\times4+n} \leftarrow C_{(1+n+m) \bmod 4 + i\times4}$
            end for
        end for
        $V.append(P)$
    end for
    $A \leftarrow$ `copy of P`
end for
Return V

3.2.2 Dynamic Permutation Matrix Values

$$\begin{Bmatrix}
0 & 4 & 8 & 12 & 16 & 20 & 24 & 28 & 1 & 5 & 9 & 13 & 17 & 21 & 25 & 29 & 2 & 6 & 10 & 14 & 18 & 22 & 26 & 30 & 3 & 7 & 11 & 15 & 19 & 23 & 27 & 31 \\
4 & 8 & 12 & 0 & 20 & 24 & 28 & 16 & 5 & 9 & 13 & 1 & 21 & 25 & 29 & 17 & 6 & 10 & 14 & 2 & 22 & 26 & 30 & 18 & 7 & 11 & 15 & 3 & 23 & 27 & 31 & 19 \\
8 & 12 & 0 & 4 & 24 & 28 & 16 & 20 & 9 & 13 & 1 & 5 & 25 & 29 & 17 & 21 & 10 & 14 & 2 & 6 & 26 & 30 & 18 & 22 & 11 & 15 & 3 & 7 & 27 & 31 & 19 & 23 \\
12 & 0 & 4 & 8 & 28 & 16 & 20 & 24 & 13 & 1 & 5 & 9 & 29 & 17 & 21 & 25 & 14 & 2 & 6 & 10 & 30 & 18 & 22 & 26 & 15 & 3 & 7 & 11 & 31 & 19 & 23 & 27 \\
12 & 28 & 13 & 29 & 14 & 30 & 15 & 31 & 0 & 16 & 1 & 17 & 2 & 18 & 3 & 19 & 4 & 20 & 5 & 21 & 6 & 22 & 7 & 23 & 8 & 24 & 9 & 25 & 10 & 26 & 11 & 27 \\
28 & 13 & 29 & 12 & 30 & 15 & 31 & 14 & 16 & 1 & 17 & 0 & 18 & 3 & 19 & 2 & 20 & 5 & 21 & 4 & 22 & 7 & 23 & 6 & 24 & 9 & 25 & 8 & 26 & 11 & 27 & 10 \\
13 & 29 & 12 & 28 & 15 & 31 & 14 & 30 & 1 & 17 & 0 & 16 & 3 & 19 & 2 & 18 & 5 & 21 & 4 & 20 & 7 & 23 & 6 & 22 & 9 & 25 & 8 & 24 & 11 & 27 & 10 & 26 \\
29 & 12 & 28 & 13 & 31 & 14 & 30 & 15 & 17 & 0 & 16 & 1 & 19 & 2 & 18 & 3 & 21 & 4 & 20 & 5 & 23 & 6 & 22 & 7 & 25 & 8 & 24 & 9 & 27 & 10 & 26 & 11 \\
29 & 31 & 17 & 19 & 21 & 23 & 25 & 27 & 12 & 14 & 0 & 2 & 4 & 6 & 8 & 10 & 28 & 30 & 16 & 18 & 20 & 22 & 24 & 26 & 13 & 15 & 1 & 3 & 5 & 7 & 9 & 11 \\
31 & 17 & 19 & 29 & 23 & 25 & 27 & 21 & 14 & 0 & 2 & 12 & 6 & 8 & 10 & 4 & 30 & 16 & 18 & 28 & 22 & 24 & 26 & 20 & 15 & 1 & 3 & 13 & 7 & 9 & 11 & 5 \\
17 & 19 & 29 & 31 & 25 & 27 & 21 & 23 & 0 & 2 & 12 & 14 & 8 & 10 & 4 & 6 & 16 & 18 & 28 & 30 & 24 & 26 & 20 & 22 & 1 & 3 & 13 & 15 & 9 & 11 & 5 & 7 \\
19 & 29 & 31 & 17 & 27 & 21 & 23 & 25 & 2 & 12 & 14 & 0 & 10 & 4 & 6 & 8 & 18 & 28 & 30 & 16 & 26 & 20 & 22 & 24 & 3 & 13 & 15 & 1 & 11 & 5 & 7 & 9 \\
19 & 27 & 2 & 10 & 18 & 26 & 3 & 11 & 29 & 21 & 12 & 4 & 28 & 20 & 13 & 5 & 31 & 23 & 14 & 6 & 30 & 22 & 15 & 7 & 17 & 25 & 0 & 8 & 16 & 24 & 1 & 9 \\
27 & 2 & 10 & 19 & 26 & 3 & 11 & 18 & 21 & 12 & 4 & 29 & 20 & 13 & 5 & 28 & 23 & 14 & 6 & 31 & 22 & 15 & 7 & 30 & 25 & 0 & 8 & 17 & 24 & 1 & 9 & 16 \\
2 & 10 & 19 & 27 & 3 & 11 & 18 & 26 & 12 & 4 & 29 & 21 & 13 & 5 & 28 & 20 & 14 & 6 & 31 & 23 & 15 & 7 & 30 & 22 & 0 & 8 & 17 & 25 & 1 & 9 & 16 & 24 \\
10 & 19 & 27 & 2 & 11 & 18 & 26 & 3 & 4 & 29 & 21 & 12 & 5 & 28 & 20 & 13 & 6 & 31 & 23 & 14 & 7 & 30 & 22 & 15 & 8 & 17 & 25 & 0 & 9 & 16 & 24 & 1
\end{Bmatrix}$$

3.2.3 Other Options

3.3 Confusion - ARX

3.3.1 A - Modular Addition

3.3.2 R - Bitwise Rotation

3.3.3 X - XOR

3.4 Key Transformation

4 Cipher

4.1 Transformation

4.2 Avalanche Effect - Plaintext

4.2.1 Encryption Index

4.3 Long Plaintexts