A Novel Design Method of Stream Ciphers Based on Table-Element Permutation

Hongbo Zou

Neijiang Local Tax Bureau, Sichuan, China
scnjds@163.com

Abstract. In this paper, a new stream ciphers design method (named TEP) is proposed to base on the table-element nonlinear permutation. A number of words are generated by n-LFSRs(linear feedback shift register) input to a table. In the table, every word is dealt with by the nonlinear transforms and the several words are combined with nonlinear function to produce keystream words. The algorithm is simplicity and the secret key is generated rapidly. The result of many simulation experiments show that the keystream by TEP method generating can meet DIEHARD statistics tests. The approach is efficient to design stream ciphers.

Key words: stream cipher, combining function, LFSR, permutation

1 Introduction

Stream ciphers are an important manner of information encryption. One approach of the conventional stream ciphers is to use n-LFSRs in parallel, their output combined using an n-input binary Boolean function . Various properties of such a combining function are critical for ensuring the security of the resultant scheme. It is very difficult for designer to design a function, which not only meet a variety of conditions for randomness, but also resists against all kinds of attack approach. How to design function is central topics to design of the kind of stream ciphers [3, 4].

The RC4 is a stream cipher widely deployed in software applications due to its simplicity and efficiency. 32-bit RC4 requiring 16 Gigabytes of memory, would be incredibly limiting need 4G memory. RC4 can't directly generate 32, 64-bit word flow in PC platform. Furthermore, there are several weaknesses in RC4 which is found in [7]. A number of word-oriented stream ciphers such as Rabbit, Turing, HC-256 [8, 9, 11] etc have been presented in current, but their security had not been proved exhaustively.

A new stream ciphers design method is presented in this paper, named TEP (Table-Element Permutation) algorithm. It is difference to bit-oriented combining function stream cipher, which is a word-oriented combining stream cipher. The main idea of the TEP is the n-LFSRs(Linear Feedback Shift Register) provide randomness 32-bit or 64-bit word mode to input to the table in turn, Every

element of the table is transposed with various nonlinear transform and confused the original linear relationship between a word and others, the random counters choose several words to nonlinear combine in order to generate unpredictable secret key. Principle of TEP see Fig. 1. The result of simulation experiments shows that random keystream is generated by TEP method to meet DIEHARD statistics tests.

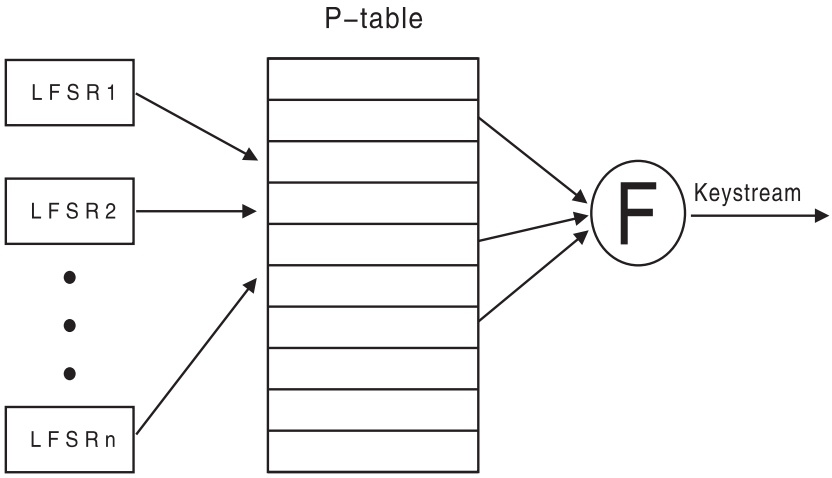

Fig. 1. Principle of TEP stream cipher design method

2 TEP Algorithm describe

TEP Algorithm consists of two steps. The first is initialize procedure, the second step is main program. They all use SP nonlinear transform to confuse internal states of the table-element to generate keystream for encryption. We discuss later. Transposition cryptosystem, an old strategy is adopted to in this paper for SP nonlinear transform design.

2.1 Organization and notation

This paper is organized as follows. We will describe the design method of TEP in detail in section two and the simple cryptanalysis of TEP in section three, and in section four the performance results are presented. We conclude and summarize in section five.

The following operations and functions are used in TEP:

The table P is used in TEP, LFSRi denoted ith LFSR generated 32-bit word. Every element is $P[i] = p_i(0)||p_i(1)||p_i(2)||p_i(3)$, $p_i(0), p_i(1), p_i(2), p_i(3)$ are four

bytes.

P : a table with 256*32 bit elements.Every element is denoted as $P[i], P[i]$ with $0 \leq i < 256$.

$p_i(j)$: jth byte of ith element of P table is denoted as $p_i(j).p_i(j)$ with $0 \leq i < 256, 0 \leq j \leq 3$. Every element split into four bytes.

$+$: means $x + y \bmod 256$, where $0 \leq x, y < 256$.

$\oplus$: exclusive operator.

$\|$: concatenation operator.

mm : random key word of output.

j : is random counter to select element operation and output.

$flag$: two LFSR select mark. $flag = 0$ select LFSR1, $flag = 1$ select LFSR2

$P[i] \ggg r$: Table-element $P[i]$ right cyclical shift r bit, means $((P[i] \gg r) \oplus (P[i] \ll (32 - r))$ where $0 \leq i \leq 255, 0 \leq r \leq 31$.

$Input(LFSRi)$: LFSRi generate 32-bit word to input table.

$Output(mm)$: output 32-bit word mm.

2.2 SP Transform

The transform operator of TEP called SP transform (section permutation) to permute internal one byte (8-bit) between two words. Begin from the first element of the table P, the random counter select one element $P[i]$ in the table, $P[i]$ right cyclical move 11 bits. $P[i + 213]$ right cyclical move 5 bits. Divide element $P[i]$, $P[i+213]$ into four 8-bit quarters with hexadecimal digits of $p_i(0)$, $p_i(1)$, $p_i(2)$, $p_i(3)$ and $p_{i+213}(0)$, $p_{i+213}(1)$, $p_{i+213}(2)$, $p_{i+213}(3)$. The content of $p_{i+213}(0)$, $p_{i+213}(1)$, $p_{i+213}(2)$, $p_{i+213}(3)$ is regarded as address position of words to operate in the table, corresponding elements are $P[p_{i+213}(0)]$, $P[p_{i+213}(1)]$, $P[p_{i+213}(2)]$, $P[p_{i+213}(3)]$. The first byte of $P[p_{i+213}(0)]$ element will permute $p_i(0)$ byte of $P[i]$. The second byte of $P[p_{i+213}(1)]$ element will permute $p_i(1)$ byte of $P[i]$. The third byte of $P[p_{i+213}(2)]$ element will permute $p_i(2)$ byte of $P[i]$. The fourth byte of $P[p_{i+213}(2)]$ element will permute $p_i(3)$ byte of $P[i]$. Last, $P[p_{i+213}(0)]$, $P[p_{i+213}(1)]$, $P[p_{i+213}(2)]$, $P[p_{i+213}(3)]$ and new $P[i]$ right cyclical shift 3, 13, 19, 29, 17-bit respectively. SP transform explicit description is as follow:

$$
\begin{aligned}
P[i] &\ggg 11 \\
P[i+213] &\ggg 5 \\
P[i] &= p_i(0)||p_i(1)||p_i(2)||p_i(3) \\
P[i+213] &= p_{i+213}(0)||p_{i+213}(1)||p_{i+213}(2)||p_{i+213}(3)
\end{aligned}
$$

$$\begin{aligned} P[p_{i+213}(0)] &= a_0(0)||a_0(1)||a_0(2)||a_0(3) \\ P[p_{i+213}(1)] &= a_1(0)||a_1(1)||a_1(2)||a_1(3) \\ P[p_{i+213}(2)] &= a_2(0)||a_2(1)||a_2(2)||a_2(3) \\ P[p_{i+213}(3)] &= a_3(0)||a_3(1)||a_3(2)||a_3(3) \end{aligned}$$

After exchanging, these elements are changed as follow:

$$\begin{aligned} P[i] &= a_i(0)||a_i(1)||a_i(2)||a_i(3) \\ P[p_{i+213}(0)] &= p_i(0)||a_0(1)||a_0(2)||a_0(3) \\ P[p_{i+213}(1)] &= a_1(1)||p_i(1)||a_1(2)||a_1(3) \\ P[p_{i+213}(0)] &= a_2(0)||a_2(1)||p_i(2)||a_2(3) \\ P[p_{i+213}(0)] &= a_3(0)||a_3(1)||a_3(2)||p_i(3) \end{aligned}$$

Next rotation right every elements.

$$\begin{aligned} P[i] &\ggg 17 \\ P[p_{i+213}(0)] &\ggg 3 \\ P[p_{i+213}(1)] &\ggg 13 \\ P[p_{i+213}(2)] &\ggg 21 \\ P[p_{i+213}(3)] &\ggg 29 \end{aligned}$$

Design illuminate: we firstly use rotation-right operator simple change two table-element internal linear relation and avoid same context for next SP operation choosing same table-element. The reason that we choose four bytes of $P[i+213]$ as address of $P[i]$ operation is we don't want to use the context of $P[i]$ as address of itself operation. As for i adding 213 is stochastic, in fact, we think may add the other random number.

2.3 Initialization procedure

Similar to the general combination function stream ciphers algorithm, the new algorithm also uses n-LFSRs as input resources to supply random well word stream. TEP put stochastic one 32-bit word with n-LFSRs generating into a table (or pool) in turn. We use two LFSRs as example to illustrate TEP principle in the paper. TEP first put 128 32-bit words by the first LFSR generating into a ahead half of the table, afterwards input 128 32-bit words by second LFSR producing to behind half of the table. TEP will permute the four sections of every element of the table with SP transform without generating output. TEP repeatedly the same operate from the first element to entire table. After the initialization process has been completed, TEP is ready to generate keystream.

2.4 Main program

The main program will generate and output random sequence key. It also uses SP transform that is previously used at initialization program. Begin from first element, program uses SP transform at first, then select one element $P[j]$ with one random counter to combine other four words with F function, the result of operation to output as keystream. $P[j]$ is replaced with one 32-bit word by two LFSRs generating in turn. New the 32-bit element of input will be transposed with SP. It is not until main program to stop that producing enough number keystream. F function is described later.

2.5 $F(i, j)$ function describe

$F(i, j)$ function is nonlinear transform just as conventional stream ciphers. F function is bit-oriented in conventional stream ciphers, however, it is word-oriented in TEP. It is a key for this kind stream ciphers family to design F function for security. In the paper, F function is designed to use exclusion operator with five 32-bit words that are selected from among the table P by the random counters. The random counters j are derived from internal states of the table. Explicit description is as follows. Divide every element $P[i]$ into four 8-bit quarters with hexadecimal digits of $p_i(0)$, $p_i(1)$, $p_i(2)$, $p_i(3)$. The content of $p_i(0)$, $p_i(1)$, $p_i(2)$, $p_i(3)$ is regarded as address the position of words, corresponding elements are $P[p_i(0)]$, $P[p_i(1)]$, $P[p_i(2)]$, $P[p_i(3)]$, that right cyclical shift 3, 9, 19, 23 bits respectively.

$$\begin{aligned} P[i] &= p_i(0)||p_i(1)||p_i(2)||p_i(3) \\ j &= j + (p_i(0) + p_i(1)) \oplus (p_i(2) + p_i(3)) \\ F(i,j) &= P[p_i(0)] \ggg 3 \oplus P[p_i(1)] \ggg 9 \oplus P[p_i(2)] \ggg 19 \\ &\quad \oplus P[p_i(3)] \ggg 23 \oplus P[j] \ggg 29 \end{aligned}$$

where j initial value is 0.

2.6 Pseudo code describe

Algorithm 1 Initialization Procedure

```
for i = 0 to 127 do
    P[i] ← Input(LFSR1)
end for
for i = 128 to 255 do
    P[i] ← Input(LFSR2)
end for
for i = 0 to 255 do
    SP(P[i])
end for
```

Algorithm 2 Main Program

```
flag ← 0
j ← 0
while don't satisfy stop condition do
  for i = 0 to 255 do
    SP(P[i])
    j ← j + ((p_i(0) + p_i(1)) ⊕ (p_i(2) + p_i(3)))
    mm ← Output(F(i, j))
    if flag = 0 then
      P[j] ← Input(LFSR1)
      flag ← 1
    else
      P[j] ← Input(LFSR2)
      flag ← 0
    end if
    SP(P[j])
  end for
end while
```

The words sequence of mm is the keystream that we need.

The secret key is chosen to be the initial state of the LFSRs.

Design specification: Using the sequence counter ensures that each element of the table P can be updated with SP operator. A 32-bit word with the use of two LFSRs generating alternately input to the table. For each element of input, using SP operator mix up its bit sequence linear relation. The states of internal element of the table are confused with cyclically right shift operator. All elements of the table will be fragment with SP operator repeatedly.

2.7 Encryption and decryption

The key stream is XORed with the message for encryption. The decryption is to XOR the key stream with the cipher text.

3 Security of TEP discuss

3.1 Period property

TEP algorithm hasn't obvious period characteristics or is difficult to predict even though two LFSRs have definite period. Period of TEP not only depend on two LFSRs' period but is decided by the states of table-element. Because every element of the table is operated with cyclical right shift and SP transform repeatedly, even if LFSRs return initially state after circulating a cycle, words element of the table aren't original status in the same position to result in outputting grant difference.

3.2 Resistance

TEP algorithm is different from traditional combining function stream ciphers, it is word-oriented. It is also different from current proposed word-oriented stream cipher such as Rabbit, HC-256, Turing, and so on. It assimilates merit of traditional combiner function stream ciphers that have well randomness. The table of TEP is to open and dynamical update. Output of n-LFSRs is merely resources to satisfy stochastic property, doesn't directly output. It is thus impossible almost all the techniques developed to attack stream ciphers based on LFSR fail on TEP. The output result can resist against linear attacks, differential attacks and other common attack. It needs time for security of TEP to exhaustive analysis. For TEP, SP transform achieves to largely scale bytes permutation between four elements. The output word leaks very small amount of partial information at each step. The large number of elements of the table and a non-linear transforms ensures that the secret key could not be recovered from those that leaked partial information. We think it is impossible to recover the secret key faster than exhaustive key search. We hope that the cryptographic community prove its security or find the weakness, which helps us improve this method in the future.

3.3 Balance

TEP compared with the conventional sequence ciphers algorithm, its mathematical description is more difficult. Randomness of the output of TEP has well mathematical foundation based on LFSR, which is confirmed by a lot of statistical test. Nonlinear transformation function F and SP transform are critical to ensure the algorithm security. A variety of nonlinear transform come into being many different algorithms. In this paper, we adopt a conservative method to design SP operator that merely modify 0, 1 position of elements, which doesn't increase or decrease the number of 0 and 1 in the table. The $F(i, j)$ function has a balance property, so the balance has a theoretical guarantee owing to LFSR balance.

3.4 Parameters and Experiment

The initial states of the LFSRs can't be full 0. We note that although length of the table P is almost universally specified in the literature as being a power of 2, and most often 256, this is not necessary, but more of a convenience. Choosing length of the table just depends on experience in order to TEP algorithm carry out easily and simply. It is too small to resist against cryptographically analysis, but too large will increase the cost of time and space. Generally, scale of the table is chosen from 256, 512, 1024, 2048 elements. Width of the table suit for computer platform 8, 16, 32, 64, 128-bit in order to algorithm run rapidly.

For TEP, we make many simulation tests with Turbo 2.0 for windows. It is difficult and slow for conventional LFSR to achieve with C language, Our experiments replace conventional LFSR with a Galos-LFSR configuration. Galois-LFSR is not cryptographically superior, when it's compared with typical LFSR.

They exhibit some of the same characteristics. If they constructed from primitive polynominals, they have maximal period 2^{n-1} iterations before repeating. In software, Galois- LFSR exploits parallelism and easier achieves with C language, thus it can execute faster.

We investigate the randomness of TEP with DIEHARD statistical tests to choose a variety of key parameters. These tests analyze a single large file from the output of the generator of megabytes or more. Majority of tests in DIEHARD return a p-value, which should be uniform on $[0, 1]$,if the input file contains truly independent random bits.In the paper, LFSR1 is specified 160-bit, LFSR is specified 120-bit in order to easy to implement on 32-bit platform for our simulation experiments.

The primitive polynomial $p_1(x)$ and $p_2(x)$ of LFSR1 and LFSR2 are given below [10]:

(0xAF838772) (0xABE14240) (0x1889A622) (0x14AF19A1) (0xCC2D8086)

(0XB64E4D3F) (0XA8E7331B) (0XD871FA30) (0XD46D4DBA)

The initial states of LFSR1 and LFSR2 or secret key are given below:

(0x56B3E642) (0xA23B5F78) (0x76F4D57) (0xABCDAE12) (0x467C7642)

(0x387C6543) (0xAECDC783) (0x343B3242) (0xC3F7C943)

Choosing the initial state of LFSR1 and LFSR2 is to stochastic, no special intention. Output results can pass DIEHARD statistical tests [6]. The exhaustively test reports will be omitted due to length of the article.

4 Discuss and Conclusions

TEP is well suitability in difference processor platforms. On 8, 16-bit platform, the table of TEP changes into $8 * 256$, $16 * 256$ bits, moreover, the SP transform changes to 4-bit section operator. On 64, 128-bit platforms, the table of TEP changes into $64 * 512$, $128 * 1024$ bits and the SP transform don't change. Initialization progress of TEP will cost an amount of time. LFSR produce a 32-bit word slowly, So TEP produce speed of secret word slight slowly, It is reason that we don't evaluated performance of the TEP algorithm with current word-oriented stream ciphers, but along with the improvement of computer's speed, memory prices have dropped, these are negligible. We only propose design principle of new stream cipher in the paper, which doesn't perfectly scheme. There is much minutia need to be improved in several ways. For example, SP and $F(i, j)$ may changes into the other nonlinear transform such as S-box. Any cipher scheme is the equilibrium of efficient and time. If there is enough time, TEP may use output of the other LFSR as j random counter to replace internal states of the table generating. TEP may also use the key change initial states of the table like RC4. The new word-oriented LFSR and T function have been proposed in [1, 2, 5, 12]. Using new word-oriented LFSR or T-function, Performance of TEP is significantly risen.

References

1. A. Klimov, A. Shamir. A new class of invertible mappings. Workshop on Cryptographic Hardware and Embedded Systems-CHES' 2002, LNCS 2523[C], pp.470 483. Springer, Heidelberg (2003)
2. A. Klimov, A. Shamir. Cryptographic applications of T-functions. Ninth workshop on Selected Areas in Cryptograph-SAC' 2003, LNCS 3006[C], pp. 248 261. Springer, Heidelberg (2004)
3. A. Menezes, P. Van Oorschot, S. Vanstone. Handbook of Applied Cryptography CRC Press, 1996, Chapter 5, 7, 12, 13.
4. B. Schneier. Applied Cryptography Second Edition. J. Wiley Sons Inc, 1996, Chapter 16, 17.
5. B. Tsaban, U. Vishne. Efficient Linear Feedback Shift Rregisters With Maximal Period. Finite Fields Appl, 8(2), pp. 256-267 (2002)
6. G. Marsaglia. Diehard Statistical Tests, `http://stat.fsu.edu/~geo/`
7. I. Mantin, A. Shamir. A Practical Attack on Broadcast RC4. Fast Software Encryption (FSE'01), LNCS 2355, pp.152-164, Springer, Heidelberg(2002)
8. M. Boesgaard, M. Vesterager, T. Pedersen. Rabbit: A New High-Performance Stream Cipher. Fast Software Encryption(FSE'03), LNCS 2887, pp.307-329. Springer, Heidelberg(2003)
9. G,R. Gregory, H. Philip. Turing: A Fast Stream Cipher. Fast Software Encryption 2003, LNCS 2887, pp.290 306. Springer, Heidelberg(2003)
10. J. Arndt. `http://www.jjj.de/mathdata/rand-primpoly.txt`
11. H. Wu. A New Stream Cipher HC-256. Fast Software Encryption(FSE'04), LNCS 3017, pp.226-244. Springer, Heidelberg(2004)
12. G. Zeng, W. Ha,K. He. High Efficiency Feed-Back Shift Register: σ - LFSR. Cryptology ePrint rchive, Report 2007/114, 2007, `http://eprint.iacr.org/2007`