

New Single Cycle T-function and Stream Cipher Proposal

Jin Hong

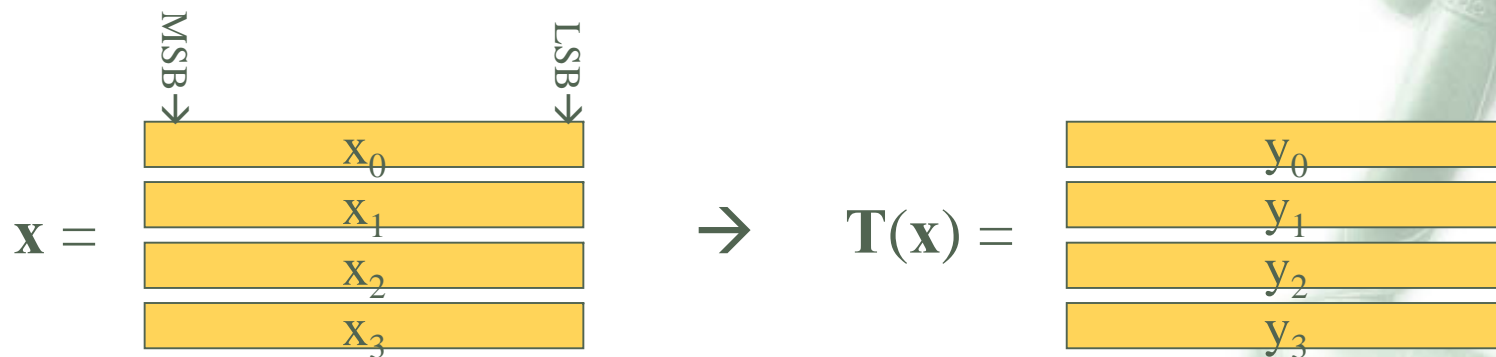
National Security Research Institute

Dong Hoon Lee, Yongjin Yeom, Daewan Han



T-function

- ✿ Introduced by Klimov & Shamir
 - CHES 2002, SAC 2003, FSE 2004
 - Mitra & Sarkar, AsiaCrypt 2004
- ✿ Building block for ciphers
- ✿ multi-word to multi-word function

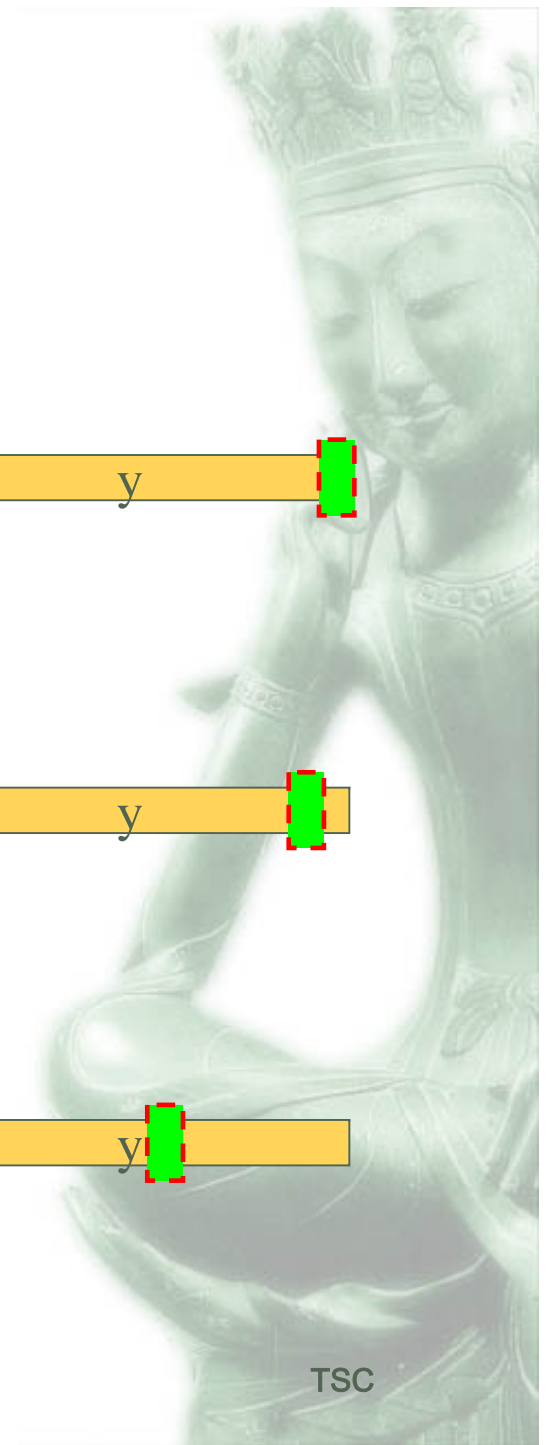
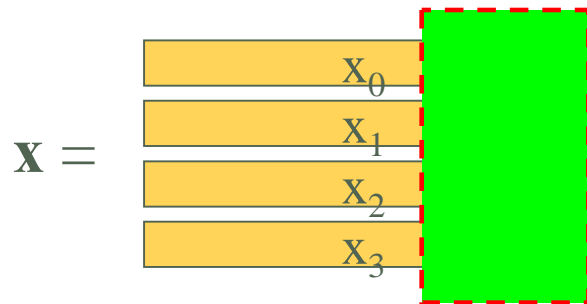
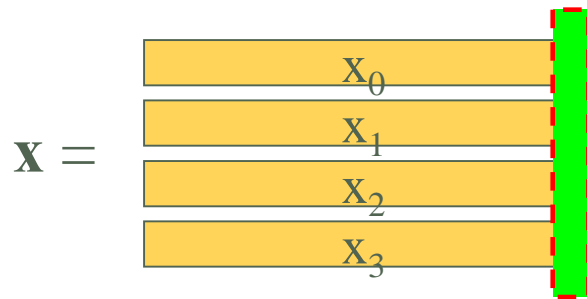
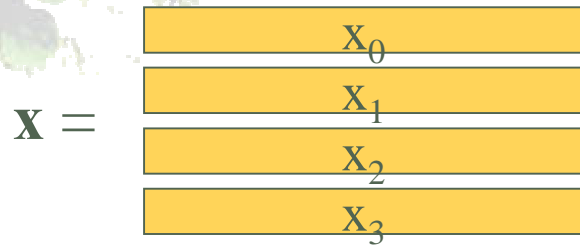


- ✿ want to use as substitute for LFSR
- ✿ analogue of a maximal length LFSR is called a single cycle T-function

T-function



Parameter



Example T-function

$$\begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{pmatrix} \mapsto \begin{pmatrix} x_0 \oplus \alpha(\mathbf{x}) & \oplus (2x_2(x_1 \vee C1)) \\ x_1 \oplus \alpha(\mathbf{x}) \wedge x_0 & \oplus (2x_2(x_3 \vee C3)) \\ x_2 \oplus \alpha(\mathbf{x}) \wedge x_0 \wedge x_1 & \oplus (2x_0(x_3 \vee C3)) \\ x_3 \oplus \alpha(\mathbf{x}) \wedge x_0 \wedge x_1 \wedge x_2 \oplus (2x_0(x_1 \vee C1)) \end{pmatrix}$$

✿ Klimov & Shamir, FSE 2004

✿ x_i : 64 bit words

✿ α : odd parameter

$$\alpha(\mathbf{x}) = p \oplus (p + C0), \quad p = x_0 \wedge x_1 \wedge x_2 \wedge x_3$$

✿ $C0, C1, C3$: integers

✿ This is a single cycle T-function

Closer Look at the Example

$$\begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{pmatrix} \mapsto \begin{pmatrix} x_0 \oplus \alpha(\mathbf{x}) \\ x_1 \oplus \alpha(\mathbf{x}) \wedge x_0 \\ x_2 \oplus \alpha(\mathbf{x}) \wedge x_0 \wedge x_1 \\ x_3 \oplus \alpha(\mathbf{x}) \wedge x_0 \wedge x_1 \wedge x_2 \end{pmatrix}$$

$$\begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{pmatrix} \mapsto \begin{pmatrix} b_0 \oplus 1 \\ b_1 \oplus b_0 \\ b_2 \oplus b_0 \wedge b_1 \\ b_3 \oplus b_0 \wedge b_1 \wedge b_2 \end{pmatrix}$$

✿ On columns with $[\alpha(\mathbf{x})]_i = 0$,
 – $[x]_i \rightarrow [x]_i$.

✿ On columns with $[\alpha(\mathbf{x})]_i = 1$,
 – $[x]_i \rightarrow [x]_i + 1$.

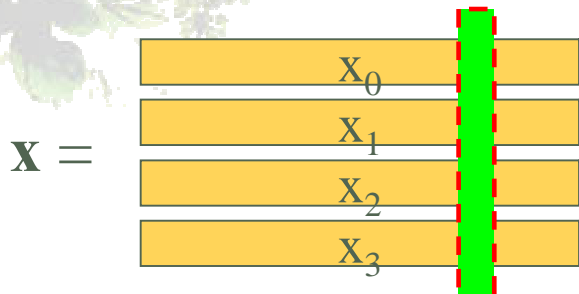
✿ Each column is a counter

✿ When $\alpha(\mathbf{x}) = p \oplus (p+1)$, it is a true counter

$$\begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{pmatrix} \mapsto \begin{pmatrix} x_0 \oplus \alpha(\mathbf{x}) & \oplus (2x_2(x_1 \vee C1)) \\ x_1 \oplus \alpha(\mathbf{x}) \wedge x_0 & \oplus (2x_2(x_3 \vee C3)) \\ x_2 \oplus \alpha(\mathbf{x}) \wedge x_0 \wedge x_1 & \oplus (2x_0(x_3 \vee C3)) \\ x_3 \oplus \alpha(\mathbf{x}) \wedge x_0 \wedge x_1 \wedge x_2 \oplus (2x_0(x_1 \vee C1)) \end{pmatrix}$$

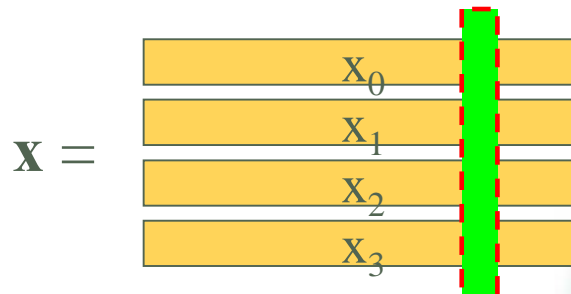
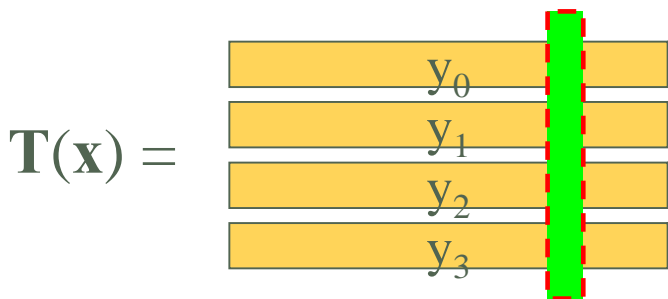
$$\begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{pmatrix} \mapsto \begin{pmatrix} x_0 \oplus \alpha(\mathbf{x}) & \oplus (2x_2(x_1 \vee C1)) \\ x_1 \oplus \alpha(\mathbf{x}) \wedge x_0 & \oplus (2x_2(x_3 \vee C3)) \\ x_2 \oplus \alpha(\mathbf{x}) \wedge x_0 \wedge x_1 & \oplus (2x_0(x_3 \vee C3)) \\ x_3 \oplus \alpha(\mathbf{x}) \wedge x_0 \wedge x_1 \wedge x_2 \oplus (2x_0(x_1 \vee C1)) \end{pmatrix}$$

Natural Extension



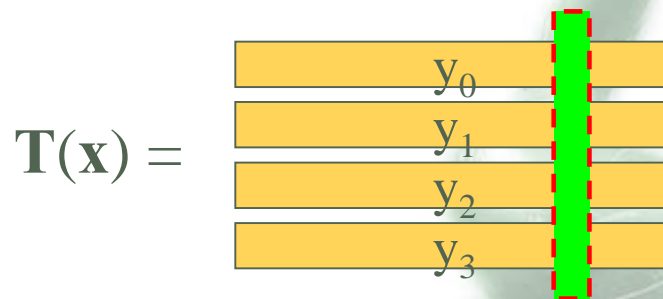
depending on $[\alpha(\mathbf{x})]_i$, apply

$[\mathbf{x}]_i \rightarrow [\mathbf{x}]_i$
or
 $[\mathbf{x}]_i \rightarrow [\mathbf{x}]_{i+1}$



depending on $[\alpha(\mathbf{x})]_i$, apply

$[\mathbf{x}]_i \rightarrow [\mathbf{x}]_i$
or
 $[\mathbf{x}]_i \rightarrow S([\mathbf{x}]_i)$





New Class of T-functions

$$T(\mathbf{x}) = \mathbf{x} \oplus \{ \alpha(\mathbf{x}) \wedge (\mathbf{x} \oplus S(\mathbf{x})) \}$$

✿ If we have

- α : odd parameter,
- S : single cycle S-box,

then T is a single cycle T-function

Comparison of T-functions

$$\begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{pmatrix} \mapsto \begin{pmatrix} x_0 \oplus \alpha(\mathbf{x}) & \oplus (2x_2(x_1 \vee C1)) \\ x_1 \oplus \alpha(\mathbf{x}) \wedge x_0 & \oplus (2x_2(x_3 \vee C3)) \\ x_2 \oplus \alpha(\mathbf{x}) \wedge x_0 \wedge x_1 & \oplus (2x_0(x_3 \vee C3)) \\ x_3 \oplus \alpha(\mathbf{x}) \wedge x_0 \wedge x_1 \wedge x_2 \oplus (2x_0(x_1 \vee C1)) \end{pmatrix}$$

$$\mathbf{T}(\mathbf{x}) = \mathbf{x} \oplus \{ \alpha(\mathbf{x}) \wedge (\mathbf{x} \oplus \mathbf{S}(\mathbf{x})) \}$$

❁ Previous

- Possible to add even parameter
- Each column is a counter

❁ New

- Better mixing within columns
- Possible to manipulate probability of change
- Impossible to add even parameter

The Cipher Proposal

- ✿ Define an odd parameter by setting
 - $\alpha(\mathbf{x}) = p \oplus (p+1) \oplus 2s,$
 - $p = x_0 \wedge x_1 \wedge x_2 \wedge x_3, \quad s = x_0 + x_1 + x_2 + x_3$
- ✿ Choose S-box so that $\mathbf{x} \rightarrow \mathbf{x} \oplus S(\mathbf{x})$ is efficiently realizable (using bit-slice technique)
- ✿ Construct a single cycle T-function

$$T(\mathbf{x}) = \mathbf{x} \oplus \{ \alpha(\mathbf{x}) \wedge (\mathbf{x} \oplus S(\mathbf{x})) \}$$

with these ingredients, acting on four 32bit words

- ✿ After each application of T apply the filter

$$f(\mathbf{x}) = (x_0 \lll 11 + x_1) \lll 14 + (x_0 \lll 13 + x_2) \lll 22 + (x_0 \lll 12 + x_3)$$

to obtain a word-size output

Extra condition on S-box

- For each fixed column i ,

$$\text{Prob}_{\mathbf{x}} ([\alpha(\mathbf{x})]_i = 1) \sim 0.5$$

- For a generic S-box, at each application of T , the state bits change with probability ~ 0.25

- Choose S-box so that it sends even number to odd number and vice versa.

- Then bits x_0 changes with probability ~ 0.5

- Then the filter

$f(\mathbf{x}) = (x_0 \lll 11 + x_1) \lll 14 + (x_0 \lll 13 + x_2) \lll 22 + (x_0 \lll 12 + x_3)$
will take care of probability of change appearing in the output.

- Quantitative argument of this is possible



Comparison of two filter models

❁ LFSR based filter model

- 1 bit output per 1 bit update
- linearity of LFSR allows algebraic attacks

❁ Our T-function based filter model

- one word output per two word update
- linearity removed by S-box, thwarting algebraic attacks
- extra caution needed of the filter because of T-function's weaker randomness



Naïve Security

$$f(\mathbf{x}) = (x_0 \lll 11 + x_1) \lll 14 + (x_0 \lll 13 + x_2) \lll 22 + (x_0 \lll 12 + x_3)$$

- ❁ At most 96 bit security
- ❁ Mixing of lower and higher bits prevents attacks that utilize localness of T-functions



Software Performance

- ❁ T-function based filter model
 - 195 MB/sec
 - Pentium4 2.4GHz, Visual C++ 6.0 (SP6)
- ❁ RC4 (Crypto++ 5.2.1 benchmarks)
 - 113 MB/sec
 - Pentium4 2.1GHz, Visual C++ .NET 2003

Software Implementation

```
unsigned int stream (unsigned int x[4])
{
    unsigned int t1, t2, t3, tmp;

    // odd parameter
    tmp = x[0] & x[1] & x[2] & x[3];
    tmp = tmp ^ (tmp + 1);
    tmp ^= (x[0] + x[1] + x[2] + x[3]) << 1;
    // t = x ⊕ S(x)
    t1 = x[3] ^ ( x[0] ) & ( ~x[2] );
    t3 = x[1] ^ ( ~x[0] ) & ( x[2] );
    t2 = ( ~x[2] ) & ( ~ ( x[1] ^ x[0] ) );
    t2 ^= ( x[2] ) & ( ( ( x[1] ) & ( ~x[0] ) ) ^ ( ( x[3] ) & ( x[0] ) ) );
    // T-function
    x[0] ^= tmp;
    x[1] ^= tmp & t1;
    x[2] ^= tmp & t2;
    x[3] ^= tmp & t3;
    // filter
    tmp = _lrotl( _lrotl(x[0], 11) + x[1], 14);
    tmp += _lrotl( _lrotl(x[0], 13) + x[2], 22);
    tmp += _lrotl(x[0], 12) + x[3] ;

    return tmp;
}
```

$$T(x) = x \oplus \{ \alpha(x) \wedge (x \oplus S(x)) \}$$

Conclusion

- ❁ Previous T-function is analyzed and a new class of single cycle T-function is given.
- ❁ Filter model built on this T-function is proposed.
- ❁ Probability of change was discussed.
- ❁ We know of no critical weakness.
- ❁ The cipher is light and fast in software.
- ❁ Hardware implementations should also be light and good enough.

- ❁ We look forward to your feedback