What do you think about T-functions?

T-79.515 Cryptography: Special Topics

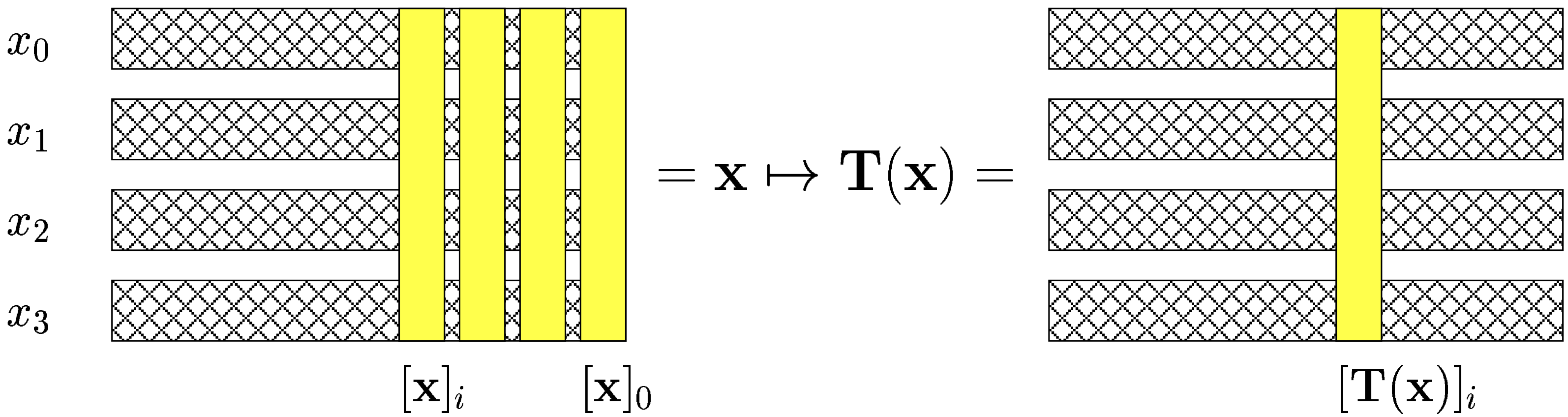
Seminar talk

Emilia Käsper

Talk overview

- The motivation behind T-functions
- Constructing a stream cipher: attempt #1
- Constructing a stream cipher: attempt #2
- T-functions in other applications
- Conclusions and discussion

What is a T-function?



- i^{th} output bits depend only on input bits $[\mathbf{x}]_i, \ldots, [\mathbf{x}]_0$
- $\mathbf{T}$ = **T**riangular
- $+,\ -,\ \times,\ \oplus,\ \vee,\ \wedge$ and their combinations

Some landmarks in the history of T-functions

- 1997: RC6 uses the mapping $x \mapsto 2x^2 + x$ (Rivest et. al.)
- 2002: T-functions as a new class (Klimov and Shamir)
- 2003: A stream cipher proposal (Klimov and Shamir)
- 2004: T-functions go mainstream, several papers, several attacks
- 2005: New applications in block ciphers and hash functions

Why T-functions?

- LFSR-s are “tame” — too well studied
- T-functions are “semi-wild”:

- And they are fast.

Why T-functions? (cont.)

- Mix “crazy” design with provable properties
- Provable single cycle property
- ... but single cycle T-functions are not easy to find

$$x \mapsto 2x^2 + x$$

$$x = \underbrace{x_1 x_2 \ldots x_n}_{n} \underbrace{00 \ldots 0_2}_{n} =: 2^n X$$

$$2x^2 + x = 2 \cdot 2^{2n} X^2 + 2^n X = 2^n X = x$$

Let's construct a stream cipher
[Shamir, Klimov 2004]

- Take the most compact known single cycle mapping:

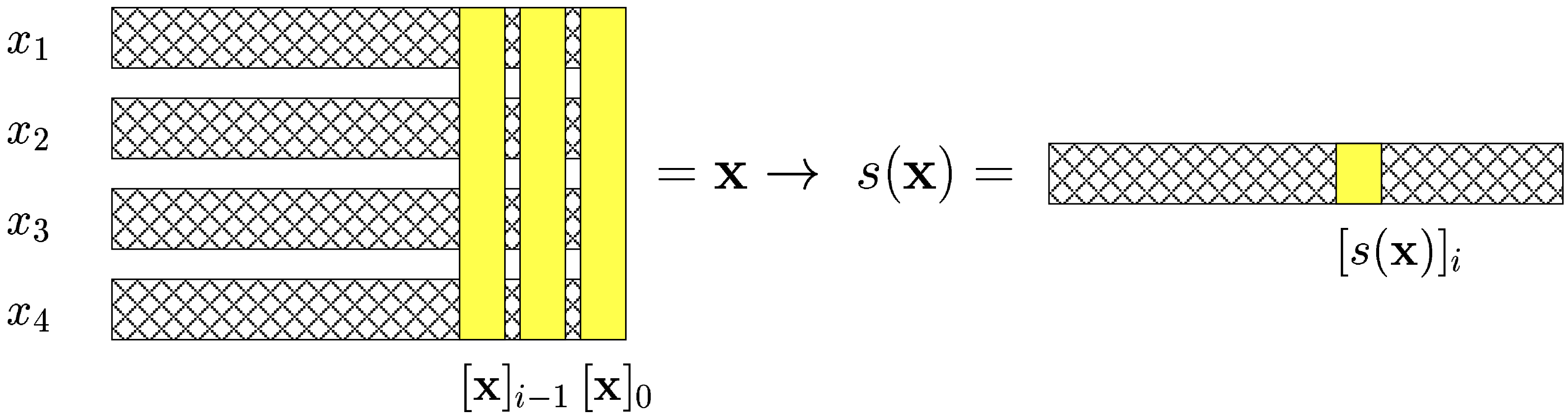
$$\begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{pmatrix} \mapsto \begin{pmatrix} x_0 & \oplus & s \\ x_1 & \oplus & (s \wedge a_0) \\ x_2 & \oplus & (s \wedge a_1) \\ x_3 & \oplus & (s \wedge a_2) \end{pmatrix}$$

$$a_0 = x_0,\ a_i = a_{i-1} \wedge x_i,\ s = s(\mathbf{x}) = (a_3 + C) \oplus a_3,$$

where C is an odd constant and $|x_i| = 64$.

Parameters

$$s = s(\mathbf{x}) = (a_3 + C) \oplus a_3 = (x_0 \wedge x_1 \wedge x_2 \wedge x_3 + C) \oplus (x_0 \wedge x_1 \wedge x_2 \wedge x_3)$$



Let's construct a stream cipher (cont.)

- This is not secure:

$$[s(\mathbf{x})]_0 = 1 \Rightarrow [\mathbf{T}(\mathbf{x})]_0 = [\mathbf{x}]_0 + 1 \pmod{2^m}$$

$$[\mathbf{T}(\mathbf{x})]_i = \begin{cases} [\mathbf{x}]_i & \text{if } [s(\mathbf{x})]_i = 0 \\ [\mathbf{x}]_i + 1 \pmod{2^m} & \text{if } [s(\mathbf{x})]_i = 1. \end{cases}$$

- C = 1 gives a counter!

Let's construct a stream cipher (still cont.)

- Add multiplication

$$\begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{pmatrix} \mapsto \begin{pmatrix} x_0 & \oplus & s & \oplus & (2x_1x_2) \\ x_1 & \oplus & (s \wedge a_0) & \oplus & (2x_2x_3) \\ x_2 & \oplus & (s \wedge a_1) & \oplus & (2x3x_0) \\ x_3 & \oplus & (s \wedge a_2) & \oplus & (2x_0x_1) \end{pmatrix}$$

Let's construct a stream cipher (still cont.)

- Avoid zero-tail attacks on multiplication
- Stay compact

$$\begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{pmatrix} \mapsto \begin{pmatrix} x_0 & \oplus & s & \oplus & (2(x_1 \vee C_1)x_2) \\ x_1 & \oplus & (s \wedge a_0) & \oplus & (2x_2(x_3 \vee C_3)) \\ x_2 & \oplus & (s \wedge a_1) & \oplus & (2(x_3 \vee C_3)x_0) \\ x_3 & \oplus & (s \wedge a_2) & \oplus & (2x_0(x_1 \vee C_1)) \end{pmatrix}$$

We're done!

- This looks secure enough!
- It looks so secure that we'll just take as output the 32 msb-s from each word.
- Bad idea...

An attack on the cipher
[Mitra, Sarkar, Asiacrypt 2004]

$$\mathbf{T}\begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} x_0 & \oplus & s & \oplus & (2(x_1 \vee C_1)x_2) \\ x_1 & \oplus & (s \wedge a_0) & \oplus & (2x_2(x_3 \vee C_3)) \\ x_2 & \oplus & (s \wedge a_1) & \oplus & (2(x3 \vee C_3)x_0) \\ x_3 & \oplus & (s \wedge a_2) & \oplus & (2x_0(x_1 \vee C_1)) \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{pmatrix}$$

- The msb-s of x_i, y_i, a_i and s are known
- Mount a time-memory tradeoff attack on multiplications
- Complexity 2^{40}

Let's construct a stream cipher: Attempt 2

[Hong, Lee et.al, FSE 2005]

- Apply S-boxes on columns <<< non-dogmatic
- A single cycle S-box will not give a single cycle T-function
- Employ parameters to get

$$\mathbf{T}(\mathbf{x}) = (s(\mathbf{x}) \wedge \mathbf{S}(\mathbf{x})) \oplus ((\ s(\mathbf{x}) \wedge \mathbf{S}^2(\mathbf{x})).$$

- This is a single cycle function for certain parameters.

Let's construct a stream cipher: Attempt 2 (cont.)

- Take 4 words, 32 bits each
- Use the T-function as a substitution for an LFSR in a *filter model*

$$f(\mathbf{x}) = (((x_0 \lll 9) + x_1) \lll 15) + ((x_2 \lll 7) + x_3)$$

- Rotations ensure that output from the same S-box does not contribute directly to the same output bit
- Remove possibility of separate handling of memory

This, too, is vulnerable
[FSE 2005 Rump session]

- Distinguishing attack: requires 2^{22} words
- Small-size parameter affecting the whole state:
- Wait for a "nice" output of the parameter, then attack

T-Functions in other applications

- Diffusion layers of block ciphers
- Self-synchronizing hash functions
- Self-synchronizing stream ciphers
- ... (Use your imagination) ...

The general T-function methodology
[Klimov, Shamir]

1. Find a skeleton bitwise mapping from 1-bit inputs to 1-bit outputs with desired property
2. Extend to n-bit words in a natural way
3. Add some parameters to obtain a larger class of mappings and provide mixing
4. Change some $\oplus$ operations to $+$ or $-$

MDS mappings

- The desired property:

$$\phi : X^m \to X^m,\ y = \phi(x), y' = \phi(x')$$

$$D_\phi = \min_{x,x'}(d(x,x') + d(y,y'))$$

- *Maximum Distance Separable* mapping: $D_\phi = m + 1$
- *Almost* MDS mapping: $D_\phi \geqslant m$
- In SPN-s: Large $D_\phi \Rightarrow$ many active S-boxes

Let's construct an almost MDS mapping

1. Find a skeleton bitwise mapping from 1-bit inputs to 1-bit outputs with desired property:

$$\begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix} \times \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{pmatrix} + \begin{pmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \end{pmatrix}$$

$$d(x, x') + d(y, y') \geqslant 4$$

Let's construct an almost MDS mapping (cont.)

2. Extend to n-bit words in a natural way

$$
\begin{aligned}
y_0 &= x_1 \oplus x_2 \oplus x_3 + s_0 \\
y_1 &= x_2 \oplus x_3 \oplus x_0 + s_1 \\
y_2 &= x_3 \oplus x_0 \oplus x_1 + s_2 \\
y_3 &= x_0 \oplus x_1 \oplus x_2 + s_3
\end{aligned}
$$

Let's construct an almost MDS mapping (still cont.)

3. Add some parameters to obtain a larger class of mappings and provide mixing

$$y_0 = (x_1 \oplus x_2 \oplus x_3)(2x_0 + 1)$$
$$y_1 = (x_2 \oplus x_3 \oplus x_0)(2x_1 + 1)$$
$$y_2 = (x_3 \oplus x_0 \oplus x_1)(2x_2 + 1)$$
$$y_3 = (x_0 \oplus x_1 \oplus x_2)(2x_3 + 1)$$

Let's construct an almost MDS mapping (still cont.)

4. Change some $\oplus$ operations to $+$ or $-$

$$y_0 = x_1 + (x_2 \oplus x_3)(2x_0 + 1)$$
$$y_1 = x_2 + (x_3 \oplus x_0)(2x_1 + 1)$$
$$y_2 = x_3 + (x_0 \oplus x_1)(2x_2 + 1)$$
$$y_3 = x_0 + (x_1 \oplus x_2)(2x_3 + 1)$$

Conclusions

- T-functions have some attractive properties
- But T-functions themselves are not secure — they are just building material
- First attempts to give a complete cipher design have failed quite miserably
- A comment from an internet forum:

  > Security can be achieved through usage but effectiveness cannot, it is something that should be there from the beginning.

Do you agree?