New Encryption Algorithm Based on T-Functions (EATF)

Haythem Zorkta and Loay Ali, *Member, IACSIT*

***Abstract*—A new block symmetric encryption algorithm EATF is introduced in this paper. This algorithm is based on a relatively new class of invertible mappings called T-Functions, which used to provide EATF with confusion and diffusion in its permutation and balanced mixing phases. Again T-Functions in used to generate large quasigroups as its encryption core. EATF has multiple operational modes according to its dynamic key length. Analytical study proved EATF strength in all operational modes against brute force attacks, and introduced block encryption times equal to 50 µs.**

***Index Terms*—Private key cryptosystems, quasigroup, string transformations, random permutations.**

I. Introduction

Many cryptographic algorithms designed by a various ideas depend on application environment. A well-known examples are block ciphers or stream ciphers which can use symmetric or asymmetric ciphers. However, many cryptographic algorithms use a mixture of different kinds of operations (e.g. balanced block mixer functions, modular additions or multiplications and bit shifts or rotations) such that they cannot be described easily by some relatively small or simple system of linear or quadratic equations. As these operations are algebraically rather incompatible, it is hard to solve equations which include different ones algebraically [1]-[4].

New class of T-functions [5] is presented in this article, which allows to implement permutation transformations and provide a powerful method for generating a larger set of quasigroups. It has very good ciphering properties and, therefore, it has potential uses in symmetric cryptography.

Additionally an implementation of new symmetric block ciphering algorithm was presented. This algorithm include permutation, balanced block mixer (BBM) and quasigroup encryptor.

However, this paper is not only dedicated to the quite young subject of T-functions. The presented algorithm, insures the effectiveness of the encryption by the immense complexity associated with the task of finding the scrambling transformation.

II. Preliminaries

In this section definitions for quasigroup, T-function and BBM are presented.

Manuscript received June 15, 2012; revised August 1, 2012. This paper was accepted by 4th IEEE International Conference on Computer Science and Information Technology (IEEE ICCSIT 2011)

H. Zorkta and L. Ali are with the Aleppo University- Syria (email: drzorkta@hotmail.com; Laaj_4@yahoo.com).

A. Quasigroup String Transformations

Definition 1. A quasigroup $(Q, *)$ is a groupoid satisfying the law

$$(\forall u, v \in Q)(\exists! x, y \in Q) : u * x = v \,\&\, y * u = v.$$

It follows that for each $a, b \in Q$ there is a unique $x \in Q$ such that $a * x = b$. Then we denote $x = a \backslash_* b$ where $\backslash_*$ is a binary operation in Q (called a left parastrophe of $*$) and the groupoid $(Q, \backslash_*)$ is a quasigroup too. The algebra $(Q, *, \backslash_*)$ satisfies the identities

$$x \backslash_* (x * y) = y,\ x * (x \backslash_* y) = y.$$

Consider an alphabet (i.e., a finite set) Q, and denote by Q^+ the set of all nonempty words (i.e., finite strings) formed by the elements of Q. In this paper, we will use two notifications for the elements of Q^+: $a_1a_2 \ldots a_n$ and $(a_1, a_2, \ldots, a_n)$, where $a_i \in Q$. Let $*$ be a quasigroup operation on the set Q. For each $l \in Q$ we define two functions $e_{l,*}$, $d_{l,*}$: $Q^+ \to Q^+$ as follows:

Definition 2. Let $a_i \in Q$, $M = a_1a_2 \ldots a_n$. Then

$$e_{l,*}(M) = b_1b_2 \ldots b_n \Leftrightarrow$$

$$b_1 = l * a_1,\ b_2 = b_1 * a_2, \ldots, b_n = b_{n-1} * a_n,$$

$$d_{l,*}(M) = c_1c_2 \ldots c_n \Leftrightarrow$$

$$c_1 = l * a_1,\ c_2 = a_1 * a_2, \ldots, c_n = a_{n-1} * a_n,$$

The functions $e_{l,*}$ and $d_{l,*}$ are called the *e*–transformation and the *d*–transformation of Q^+ based on the operation * with leader *l* respectively [6], [7].

Theorem 1. If $(Q, *)$ is a finite quasigroup, then $e_{l,*}$ and $d_{l,\backslash *}$ are mutually inverse permutations of Q^+, i.e.,

$$d_{l,\backslash *}(e_{l,*}(M)) = M = e_{l,*}(d_{l,\backslash *}(M))$$

for each leader $l \in Q$ and for every string $M \in Q^+$.

B. T-Functions:

Definition 3. T-functions are a relatively new class of invertible mappings using a combination of arithmetical operations like addition, subtraction, multiplication and negation together with Boolean operations like OR, XOR and NOT. This helps to design cryptographic schemes resistant to many present day attacks. Example of such a mapping is

$$x \to \sum_i (x \oplus c)_i \left(\bmod 2^n\right) \tag{1}$$

where C is a positive constant, i is an odd number and $\oplus$ represents the logical operation XOR [5]. This turns out to be a permutation of single cycle of length 2^n given certain conditions. In general, a T-function is a mapping in which the i^{th} bit of the output depends on $0, 1, \ldots, i^{th}$ input bits. Using a small number of such primitive operations over n-bit (32, 64, etc) words, efficient cryptographic building blocks can be designed. It is interesting to note that composition of two T-functions will also be a T-function. T-functions is used mainly to create a quasigroup, so construction of large quasigroups from smaller ones is an important problem for many applications.

Example: Let $v: z_2^3 * z_2^3 \rightarrow z_2^3$ be given by

$$v(x, y) = x^2y^2 + 2(x \vee y)$$

here $\vee$ represents the Boolean OR operation and addition and multiplication are defined for mod 2^3.

Let $c = \langle 1,0,0 \rangle \in z_2^3$. We define:

$$x * y = c + (x + y) + 2v(x, y)$$

$$x * y = 4 + x + y + 2x^2y^2 + 4(x \vee y)$$

Based on this, the quasigroup is: (see Table I)

TABLE I: THE QG CREATED USING A T-FUNCTION $V(X, Y)$.

*	0	1	2	3	4	5	6	7
0	4	1	6	3	0	5	2	7
1	1	4	3	6	5	0	7	2
2	6	3	0	5	2	7	4	1
3	3	6	5	0	7	2	1	4
4	0	5	2	7	4	1	6	3
5	5	0	7	2	1	4	3	6
6	2	7	4	1	6	3	0	5
7	7	2	1	4	3	6	5	0

C. *Balanced Block Mixer (BBM)*

BBM is an *m*-input-port *m*-output-port mechanism with the following properties [8]:

1) The mapping is one-to-one: every possible input value (across all input ports) to the mixer produces a different output value (across all output ports), and every possible output value is produced by a different input value.
2) Each output port is a function of all input ports.
3) Any change to any one of the input ports will produce a change to every output port.
4) Stepping any one input port through all possible values (while keeping the other inputs fixed) will step every output port through all possible values.

The overall structure of a Balanced Block Mixer is that of an orthogonal pair of Latin squares [8], which may be implemented either by computation when and as needed, or as explicit look-up tables (see Fig. 1).

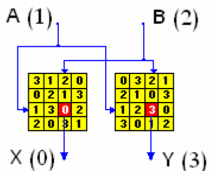

Fig. 1. BBM unit.

So on, If we have two input ports labeled A and B, two output ports labeled X and Y, and some irreducible modulo 2 polynomial P of degree appropriate to the port size, BBM is formed by the equations:

$$X = 3A + 2B \ (mod\ 2)(mod\ P)$$

$$Y = 2A + 3B \ (mod\ 2)(mod\ P)$$

This particular Balanced Block Mixer is a self-inverse, and so can be used without change whether enciphering or deciphering [9].

III. THE PROPOSED CIPHER ALGORITHM (EATF)

A new design for symmetric block cipher algorithm EATF is presented in this paper. EATF is based on new T-Function equations to provide confusion in its permutation phase and diffusion in its BBM phase. Again T-Function in used to generate large quasigroups named here S-Box encryptor as its encryption core. Fig. 2 shows the main structure of EATF.

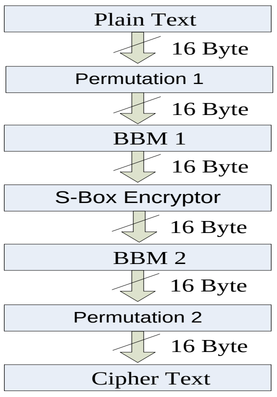

Fig. 2. EATF main structure.

EATF has symmetry design and consist of three main phases, Permutation phase, BBM phase and S-Box Encryptor phase.

A. *Permutation Phase*

Permutation phase receives 16 bytes block from plaintext as input, permute it according to a T-Function equation (1), introduced in section 2.2 and return 16 Bytes as an output (see Table II).

TABLE II: PERMUTATION PHASE。

O = P (x, i, n, c)	

Input:	***x***: 16 Bytes plaintext block;
	i: number of the T-function equation terms (odd integer);
	n: length of the single cycle permutation;
	c: array of **i** different positive numbers.

Output:	*O*: 16 Bytes after permutation.
Process:	For *j*:1 to 16 do $O_j = T(x_j)$

B. *BBM Phase*

A new design of BBM is used in this article (see Fig. 3).

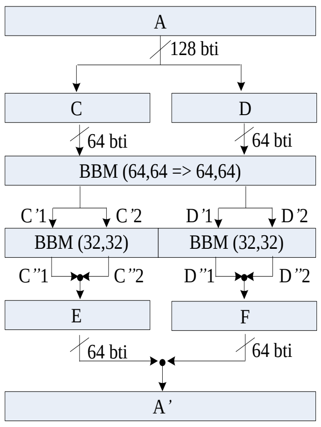

Fig. 3. BBM Structure

Table III explains how this BBM phase works:

TABLE III: BBM PHASE

***O'* = BBM(*A*, *p*1, *p*2, *p*3)**	
Input:	***A*:** 16 Bytes, from permutation phase
	***p*1**: irreducible polynomial with 65 bits
	***p*2, *p*3:** irreducible polynomials with 33 bits.
Output:	O': 16 Bytes after BBM.
Process:	*A* → [*C* (8 odd Bytes), *D* (8 even Bytes)].
	[*C*', *D*'] = BBM([*C*, *D*], *p*1).
	C' → [*C*'1 (4 odd Bytes), *C*'2 (4 even Bytes)].
	D' → [*C*'1 (4 odd Bytes), *D*'2 (4 even Bytes)].
	[*C*"1, *C*"2] = BBM([*C*'1, *C*'2], *p*2).
	[*D*"1, *D*"2] = BBM([*D*'1, *D*'2], *p*3).
	E ← [*C*"1, *C*"2], *F* ← [*D*"1, *D*"2].
	O' ← [*E*, *F*].

C. S-Box Encryptor Phase

In this sub section T-functions used to generate ($2^8 \times 2^8$) S-box Encryptor (see Fig. 4), which perform ciphering operations, as shown Table IV.

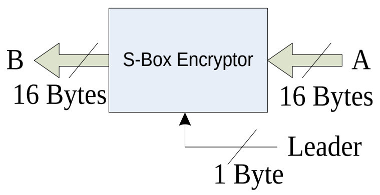

Fig. 4. S-Box Structure

TABLE IV: S-BOX ENCRYPTOR PHASE

***B* = *S*-Box(*A*, leader)**	
Input:	***A***: 16 Bytes form BBM phase;
	Leader: positive integer < 2^8
Output:	**B**: 16 Bytes after S-Box Encryptor.
Encryption:	B_1 = leader * A_1.
	For *i* = 2 to 16 do
	$B_i = B_{i-1} * A_i$.
Decryption:	A_1 = leader $\backslash_{*}$ B_1.
	For i = 2 to 16 do
	$A_i = B_{i-1} \backslash_{*} B_i$.

Finally, EAFT steps in encryption and decryption are illustrated in Tables (V, VI)

TABLE V: EATF ENCRYPTION ALGORITHM STEPS

EATF Encryption Algorithm	
Input:	**PT**: 16 Bytes plaintext block
	Key: (*i*1,*i*2,*i*3,*i*4,*c*1,*c*2,*c*3,*c*4,*p*1..*p*6,leader)
Output:	***CT*:** 16 Bytes ciphertext block
Process:	**C1** = Permut1(*PT*, *i*1, *c*1)
	C2 = BBM1(C1, *p*1, *p*2, *p*3)
	C3 = S-Box (*C*2, leader, *i*2, *c*2, *i*3, *c*3)
	C4 = BBM2(*C*3, *p*4, *p*5, *p*6)
	CT = Permut2(*C*4,*i*4,*c*4)

TABLE VI: EATF DECRYPTION ALGORITHM STEPS

EATF Decryption Algorithm	
Input:	***CT*:** 16 Bytes Ciphertext block
	Key: (*i*1,*i*2,*i*3,*i*4,*c*1,*c*2,*c*3,*c*4,*p*1..*p*6,leader)
Output:	***PT*:** 16 Bytes plaintext block
Process:	***P1*** = Permut2^{-1}(*PT*, *i*4, *c*4)
	P2 = BBM2(*P*1, *p*4, *p*5, *p*6)
	P3= S-Box^{-1}(*P*2, leader, *i*2, *c*2, *i*3, *c*3)
	P4 = BBM1(*P*3, *p*1, *p*2, *p*3)
	PT = Permut1^{-1}(*P*4, *i*1, *c*1)

IV. ANALYSIS STUDY

Time and strength evaluation studies were done on EATF in this paper. All these studies were done on 3GHz CPU with 1 GBytes RAM workstation.

A. EATF Time Analysis

Analysis here concerned about the time of each part of EATF and also about the time overhead which occurred when number of EATF phases is increased.

- *Permutation Phase Time Analysis:*

Permutation time means the time consumed in permutation phase. Moreover, effect of increasing permutation phases is measured. Table VII lists measured times on this part of *EATF* and Fig. 5 shows permutation layers increment vs. permutation time, all these measurements done with 100 Byte plaintext file size.

TABLE VII: PERMUTATION LAYERS NUMBER VS. PERMUTATION TIME

Number of Permutation layers	**Permutation1 Time *[µs]***	**Permutation2 Time *[µs]***	**Total Permutation Time [µs]**
1	45	46	91
2	90	93	183
4	182	181	363
6	271	273	544
8	362	361	723
10	452	451	903
12	543	541	1084
14	633	632	1265
16	721	724	1445

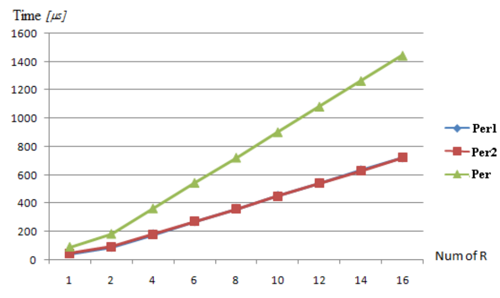

Fig. 5. Permutation layers number vs. Permutation Time

- *S-Box Encryptor Time Analysis:*

S-Box Encryptor Time means the time consumed in S-box Encryptor phase during EATF encryption or decryption process. For security purposes, increasing the number of S-boxes in this phase was considered and the time measures was recorded. Table VIII and Fig. 6 explain these results, all these measurements done with 100 Byte plaintext file size.

TABLE VIII: S-BOX NUMBER VS. S-BOX ENCRYPTOR TIME

Number of S-boxes	**S-Box Encryptor Time [µs]**
1	44
2	91
4	181
6	270
8	362
10	451
12	542
14	634
16	722

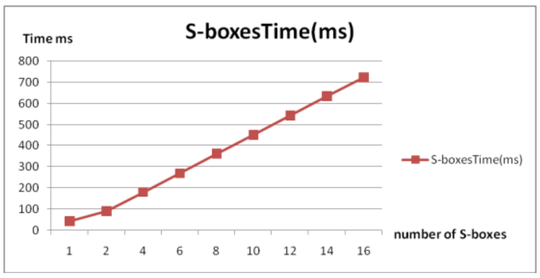

Fig. 6. S-Box number vs. S-Box Encryptor Time

- *EATF Encryption/Decryption Time Study:*

Various file sizes was encrypted, decrypted with EATF. Results show that, time increments proportionally with file size, in both encryption and decryption (see Table IX and Fig. 7).

TABLE IX: ENCRYPTION, DECRYPTION TIMES VS. FILE SIZES.

Input Size [KB]	**Enc Time [ms]**	**Dec Time [ms]**
0.1	0.3	0.31
0.2	0.59	0.60
0.4	1.16	1.18
1	2.85	2.90
3	9.63	9.78
10	32.65	33.10
20	64.84	65.80
50	162.67	164.10
100	325.33	340.66
300	976.00	1036.66
500	1627.33	1729.33
1000	3351.33	3552.66

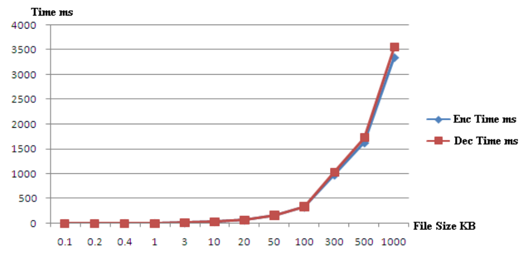

Fig. 7. Encryption, Decryption Times vs. file sizes.

B. *Proposed Cipher Algorithm's Strength*

EATF strength depends on the encryption key, which can be dynamic according to the choice of the T-Function parameters in Permutation and S-Box Encryptor phases, and the BBM irreducible polynomials. There are many combinations for the key components. Table X illustrates the boundary cases beside the normal one for the selection of the key, and shows that EATF key length is comparable to the already exist standards in its minimum and normal cases, while maximum case is reserved for most critical and sensitive data security applications. Table XI shows the estimated times to break EATF using exhaustive search attack with assumption that, there are computer works with computation power from order of 10^{30} key per second.

Moreover, multiple permutation layers and S-Boxes will increase the security of EATF definitely.

V. COMPARATIVE STUDY

Comparative study EATF and the standard Rijndael was done under the same parameters and the results was drawn on Fig. 8.

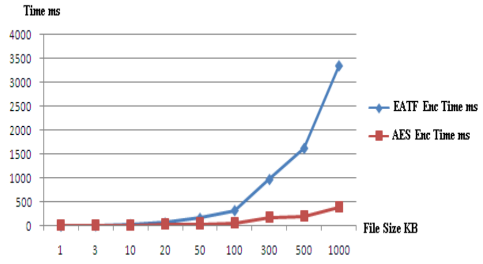

Fig. 8. EATF vs. Rijndael encryption time

This study shows that EATF encryption times is comparable to Rijndael for small file sizes (up to 250 KB).

TABLE XI: EATF KEY LENGTH WITH BOUNDARY DESIGN CASES

EATF Design Cases	**Conditions**	**key length**
Minimum case $T\text{-}F_{Per1} = T\text{-}F_{Per2}$ $BBM_1 = BBM_2$ $T\text{-}F_{1:S\text{-}b} = T\text{-}F_{2:S\text{-}b} = T\text{-}F_{Per}$	$i_1 = i_4 = i$ (16 bits) $c_1 = c_4 = c$ (32 bits) $p_1 = p_4 = p$ (65bits) $p_{2,3} = p_{5,6} = p'$(33bit) $i_2 = i_3 = i$ $c_2 = c_3 = c$ leader = l (8 bits)	154 bits
Normal case $T\text{-}F_{Per1} = T\text{-}F_{Per2}$ $BBM_1 = BBM_2$ $T\text{-}F_{1:S\text{-}b} \neq T\text{-}F_{2:S\text{-}b} \neq T\text{-}F_{Per}$	$i_1 = i_4 = i$ (16 bits) $c_1 = c_4 = c$ (32 bits) $p_1 = p_4 = p$ (65bits) $p_{2,3} = p_{5,6} = p''$ (66 b) $i_2 \neq i_3$ (32 bits) $c_2 \neq c_3$ (64 bits) leader = l (8 bits)	283 bits
Maximum case $T\text{-}F_{Per1} \neq T\text{-}F_{Per2}$ $BBM_1 \neq BBM_2$ $T\text{-}F_{1:S\text{-}b} \neq T\text{-}F_{2:S\text{-}b} \neq T\text{-}F_{Per}$	$i_1 \neq i_4$ (32 bits) $c_1 \neq c_4$ (64 bits) $p_1 \neq p_4$ (130 bits) $p_{2,3} \neq p_{5,6}$ (132 bits) $i_2 \neq i_3$ (32 bits) $c_2 \neq c_3$ (64 bits) leader = l (8 bits)	462 bits

TABLE XI: EATF BROKEN TIMES USING EXHAUSTIVE SEARCH ATTACK

EATF Design Cases	**EATF Broken Times**	**key length**
Minimum case	$\approx 5\times 10^{8}$ year	154 bits
Normal case	$\approx 3\times 10^{47}$ year	283 bits
Maximum case	$\approx 3\times 10^{100}$ year	462 bits

VI. CONCLUSIONS

Analysis study shown that EATF has the following characteristics:

1) Dynamic key length and in its minimum case, exceeds the existing standards.
2) Multiple operational modes, according to the cipher key.
3) Secure against brute force attacks.
4) Encryption time for one plaintext block (128bits) equal to 50 μs.
5) Encryption time increased proportionally with file sizes.

Encryption time analysis recommended to use EATF with file sizes less than 1/4 Mbytes

REFERENCES

[1] A. Klimov and A. Shamir, "A new class of invertible mappings. workshop on cryptographic hardware and embedded systems (CHES)," 2002.

[2] A. Klimov and A. Shamir, "Cryptographic applications of t-functions. selected areas in cryptography (SAC)," 2003.

[3] A. Klimov and A. Shamir, "New cryptographic primitives based on multiword T-functions," *FSE 2004*, 2004.

[4] A. Klimov, "Applications of T-functions in cryptography. PhD Thesis," *Weizmann Institute of Science, Submitted*, 2004. [Online]. Available: http://www.wisdom.weizmann.ac.il/˜ask/

[5] H. Zorkta, “T-Functions,” *15th Scientific Session, SCS-Aleppo,* 2010.

[6] S. K. Pal, “Securing Digital Information using Quasigroups,” Scientific Analysis Group, DRDO, Metcalfe House, Civil Lines, Delhi – 110 054, India (skptech@yahoo.com).

[7] S. Kapoor, “Securing digital information using quasigroups,” Department of Computer Science, University of Delhi, Delhi – 110 007 , India (shivam.dumca@gmail.com).

[8] Compact Encription Algorithum needed Please. [Online]. Available: http://www.ciphersbyritter.com/BBM.HTM.

[9] Formulae for Latin squares of size 2^n. [Online]. Available: http://www.ciphersbyritter.com/JAVASCRP/ACTIVBBM.HTM.

Assoc. Prof. Dr. Haythem Zorkta was bron in 1970, Damascus – Syria (drzorkta@hotmail.com) A lecturer at Aleppo University- Syria. Master and PhD degree at computer networks security from MTC- Cairo- Egypt. A lot of international and local papers at the same field, and a technical reviewer for many international and local conferences. Supervisor for many Master and PhD thesis's.

Eng. Loay Ali was bron in 1980, Damascus-Syria (laaj_4@yahoo.com) Master degree from Aleppo University (2011) at network security .Many local papers and International participation at (2011) 4th IEEE International Conference on Computer Science and Information Technology (IEEE ICCSIT 2011).