J. Software Engineering & Applications, **2010, 3: 50-57**
doi:10.4236/jsea.2010.31006 Published Online January 2010 (http://www.SciRP.org/journal/jsea)

Cryptanalysis of TEA Using Quantum-Inspired Genetic Algorithms

Wei Hu

Department of Computer Science, Houghton College, One Willard Avenue, Houghton, NY, USA.
Email: wei.hu@houghton.edu

Received September 10th, 2009; revised October 9th, 2009; accepted October 16th, 2009.

ABSTRACT

The Tiny Encryption Algorithm (TEA) is a Feistel block cipher well known for its simple implementation, small memory footprint, and fast execution speed. In two previous studies, genetic algorithms (GAs) were employed to investigate the randomness of TEA output, based on which distinguishers for TEA could be designed. In this study, we used quantum-inspired genetic algorithms (QGAs) in the cryptanalysis of TEA. Quantum chromosomes in QGAs have the advantage of containing more information than the binary counterpart of the same length in GAs, and therefore generate a more diverse solution pool. We showed that QGAs could discover distinguishers for reduced cycle TEA that are more efficient than those found by classical GAs in two earlier studies. Furthermore, we applied QGAs to break four-cycle and five-cycle TEAs, a considerably harder problem, which the prior GA approach failed to solve.

Keywords: *Cryptanalysis, Distinguisher, Feistel Block Cipher, Genetic Algorithms, Optimization, Quantum Computing, TEA*

1. Introduction

The Tiny Encryption Algorithm (TEA), a Feistel block cipher notable for its simplicity of description and implementation, was developed by David Wheeler and Roger Needham at the Computer Laboratory of Cambridge University and was first presented at the Fast Software Encryption workshop at Cambridge in 1994 [1]. Its design goal was to minimize the memory footprint and maximize the speed. There is an excellent article on TEA by Shepherd [2].

The following code presents the TEA encode routine in C language:

```
void code(long* v, long* k) {
unsigned long y = v[0], z = v[1], sum = 0, /* set up */
delta = 0x9e3779b9, n = 32 ;
while (n-->0) { /* basic cycle start */
sum += delta ;
y += (z<<4)+k[0] ^ z+sum ^ (z>>5)+k[1] ;
z += (y<<4)+k[2] ^ y+sum ^ (y>>5)+k[3] ;
}
v[0] = y ; v[1] = z ; }
```

where the 128-bit key is stored in four 32-bit blocks k = (k[0], k[1], k[2], k[3]) and data of 64 bits are stored in v = (v[0] ,v[1]).

The quantity delta in the C code is used to ensure that encryption/decryption in each cycle is different. A cycle in TEA is defined as two Feistel rounds. The SHIFT, ADD, and XOR operations in TEA provide the necessary diffusion of the statistics of the plaintext in the ciphertext and confusion between the ciphertext and key value for a secure encryption algorithm.

The simplicity of TEA's key schedule algorithm made itself susceptible to the related-key attacks; in [3] three such attacks were suggested. Soon after this discovery, the original authors of TEA created a revised version of TEA called XTEA to address this weakness [4].

Differential cryptanalysis is a commonly used cryptanalytic technique introduced by Biham and Shamir [5]. It explores the correlations between the difference in an input and the resultant difference at the output of the encryption. The goal is to discover the non-randomness, in the form of differential characteristics, of the cipher, based on which the information about the secret key used in encryption can be uncovered. In a differential cryptanalysis, a large number of plaintext pairs need to be generated following the patterns of differential characteristics of a specific problem. A random selection method will not be the best technique for this purpose. In [4,6,7], genetic algorithms [8] were employed to improve the search process for effective plaintext pairs to attack Data Encryption Standard (DES) [19]. In [9], authors suggested differential attacks on 17-cycle TEA and 23-cycle XTEA.

The impossible differential cryptanalysis proposed in [10] is a special case of differential cryptanalysis. Differential cryptanalysis seeks out the differential characteristics of a cipher with greater than expected probability, but the impossible differential cryptanalysis looks for differential characteristics with probability zero (impossible). In [11] authors conducted the impossible differential cryptanalysis of 12-cycle TEA and 14-cycle XTEA. It is interesting to note that XTEA is more vulnerable to this kind of attacks than TEA, although the original improvement was aimed at the key-related attacks.

In the study of the block cipher RC6 [12], a candidate for the Advanced Encryption Standard (AES), authors noticed that block ciphers such as TEA and XTEA that use shifting tended to display some non-random distributions in the least significant bits of their output words. For a secure encryption algorithm, the bits patterns of the output are expected to be uniform, i.e., truly random. They employed the chi-square statistic x^2 to measure the deviation of the observed distributions in the least significant bits of the output from a uniform distribution. Their results showed that RC6 with 128-bit blocks could be distinguished from a random permutation with up to 15 rounds, and for some weak keys up to 17 rounds.

In [13] authors were the first to make use of a genetic algorithm (GA) with x^2 statistic and two customized fitness functions to study the same issue with TEA. More specifically, they studied the bit patterns of the least significant eight bits of the first output word of TEA, i.e., v[0] & 255. Their goal was to search bitmasks for the input, both the input data blocks and the input key, which produces a chi-square statistic value as far as possible from the expected ones. They were successful with one-cycle, two-cycle, and three cycle TEAs, but not with the four-cycle TEA, which is a much harder problem.

In [14] authors corrected one of the two fitness functions in [13] and used a meta-GA [15] to optimize the parameters in each GA, including population size and mutation rate, to improve the results in [13], but were unable to tackle the four-cycle TEA. Consequently, to find a means to attack TEA of greater than three-cycles remains challenging. Solving this problem calls for a different approach such as designing more effective fitness functions since the performance of GAs heavily depends on the structure of its fitness function or using other evolutionary computation techniques.

2. Quantum-Inspired Genetic Algorithms

2.1 Some Basic Concepts in Quantum Mechanics

In quantum mechanics, particles move from one point to another as if they are waves, reflecting the dual nature of both waves and particles. The shape of these waves depends on the particle's angular momentum and energy level. Particles are in a low energy state on one observation, and in a high energy state on the next. There is no transition at all. The location of quantum particles, such as electrons and photons, can be described by a quantum state vector $|\psi\rangle$, a weighted sum which in the case of two possible locations equals $\alpha|0\rangle+\beta|1\rangle$, where α and β are two weights influencing the particle being in locations $|0\rangle$ or $|1\rangle$, respectively. $|\psi\rangle$ represents a linear superposition of the particle given individual quantum state vectors. However, in the act of observing a quantum state, it collapses to a single state [16]. This fact will be important when we introduce the quantum-inspired genetic algorithms in Subsection 2.5.

2.2 Quantum Bit

The basic unit of information in quantum computing is not a traditional bit but a quantum system with two states such as a photon that has two polarized directions. This quantum system is called qubit. A qubit, quantum bit, is represented as

$$|\psi\rangle=\alpha|0\rangle+\beta|1\rangle$$

where α and β are complex numbers and $|\alpha|^2+|\beta|^2=1$. $|\alpha|^2$ defines the probability that the qubit will be found in state "0" and $|\beta|^2$ defines the probability that the qubit will be found in state "1". A qubit may be in the state "0", state "1", or a linear superposition of the two.

2.3 Quantum Chromosome

Like the other evolutionary algorithms, the quantum-inspired genetic algorithms have a representation of individual or chromosome. An m-qubit chromosome is defined as:

$$\begin{bmatrix}\alpha_1\ \alpha_2\ \dots\ \alpha_m\\ \beta_1\ \beta_2\ \cdots\ \beta_m\end{bmatrix}$$

where $|\alpha_i|^2+|\beta_i|^2=1, i=1,2,\dots,m$. This expression has the capability to represent a linear superposition of states, from which all possible combinations of different values can be derived. Let us look at one such example of 3-qubit chromosome:

$$\begin{bmatrix}\frac{1}{\sqrt{2}} & \frac{\sqrt{3}}{2} & \frac{-1}{\sqrt{2}}\\ \frac{1}{\sqrt{2}} & \frac{1}{2} & \frac{1}{\sqrt{2}}\end{bmatrix}$$

The states of this chromosome can be represented as

$$\frac{-\sqrt{3}}{4}|000\rangle+\frac{\sqrt{3}}{4}|001\rangle+\frac{-1}{4}|010\rangle+\frac{1}{4}|011\rangle+\frac{-\sqrt{3}}{4}|100\rangle+\frac{\sqrt{3}}{4}|101\rangle+\frac{-1}{4}|110\rangle+\frac{1}{4}|111\rangle$$

The above expression induces a probability distribution such that the probabilities that the chromosome is seen to be in the 8 states $|000\rangle$, $|000\rangle$, $|001\rangle$, $|010\rangle$, $|011\rangle$, $|100\rangle$, $|101\rangle$, $|110\rangle$ and $|111\rangle$ are the squares of the weights. This 3-qubit chromosome is capable of representing 8 states, and 8 3-bit classical binary chromosomes are required to represent 8 states (000), (001), (010), (011), (100), (101), (110), and (111). A qubit represents probabilities of being in state "0", "1", or a superposition of both, whereas a classical bit must be in either state "0" or "1". It is evident that a qubit contains more information than a classical bit.

2.4 Quantum Mutation and Crossover

The mutation operation of a bit in a binary chromosome is flipping that bit. We made use of two types of mutations, rotational mutation and point mutation, in the quantum genetic algorithms used in this work. The rotational mutation operation of a qubit proposed in [17] is defined by a quantum rotation matrix which satisfies $UU^* = U^*U = I$, where U^* is the Hermitian adjoint matrix of matrix U and I is an identity matrix. In this paper, we only used the following real-valued quantum rotation matrix:

$$U(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

where θ represents the angle of counterclockwise rotation.

The point mutation is to switch the values of α *and* β in a qubit, and the crossover operation of a quantum chromosome is defined similarly to that of a binary chromosome. The original version of quantum-inspired evolutionary algorithm proposed in [17] did not contain such operations. We observed in our experiments that our solutions tended to be trapped at local maxima, so we introduced these two operations to increase the diversity of our solution pool. Other similar definitions of mutation and crossover could be found in the literature.

2.5 Quantum Genetic Algorithms

Encouraged by the excellent performance of the quantum-inspired evolutionary algorithm in [17], we adapted the following quantum-inspired genetic algorithm (QGA) for our current study. The structure of QGA is described in the following pseudo code:

```
QGA:
Begin
t←0
  1) initialize quantum population Q(t) of N qubit
     chromosomes
  2) make binary population P(t) by observing the states
     of Q(t)
  3) evaluate P(t)
  4) store the best solutions among P(t) into b
  while( t < T)
  t←t+1
  1) evaluate P(t-1)
  2) select the top 50% of Q(t-1) to undergo rotational
     mutation, point mutation, and crossover to produce
     N/2 new qubit chromosomes
  3) Q(t)= (the top 50% of Q(t-1)) + (N/2 new qubit
     chromosomes)
  4) make P(t) by observing the states of Q(t)
  5) store the best solutions among P(t) into b
  end while
  End
```

In our implementation of QGA, we chose $\alpha = \beta = \frac{1}{\sqrt{2}}$ for all qubits in each chromosome when t = 0, so that each qubit had equal probability to be in state "0" or "1". The quantum rotation angle θ was chosen according to Table 1 as in [18], where $\varepsilon = 0.001\pi$.

3. Results

The input of TEA is 128 bits long, which is made of 64 bit blocks of data and a 128 bit key, and the output of TEA is the encrypted 64 bit data stored in v[0] and v[1], where v[0] and v[1] are defined in the C code introduced at the beginning of this paper.

We used a qubit chromosome to represent a bitmask. To evaluate each bitmask in a QGA, a logical AND operation between the bitmask and a randomly generated input pair, data-key, of 128 bits was performed. The resultant values were then passed to TEA to yield the output. There were 211 such randomly generated data-key pairs for each bitmask.

The focus of our work is studying the distribution of the bit patterns of v[0] & 255 in the output of TEA. We recorded the counts of different values of v[0] & 255 from the outputs of TEA. The important question is whether the observed counts were significantly different from the expected ones. There are a variety of ways to assess this difference including Pearson's chi-square, G test, and Fisher's exact test. We utilized the chi-square statistic in this work as in [13] and [14]. The Pearson's chi-square is

$$X^2 = \sum_{i=1}^{N} \frac{(O_i - E_i)}{E_i}$$

In this equation, N is the number of observations, O_i is the observed counts and E_i is the expected counts. In the current study, the expected counts follow a uniform distribution, which implies the bit patterns are truly random. There are 256 possible values from v[0] & 255, therefore the maximum value of the chi-square is 522,240 with 255 degrees of freedom and 2^{11} observations (See [14] for detailed calculation).

Table 1. Rotation angle θ updating rules

x_i	$best_i$	f(x)>f(best)	θ_i	$\alpha_i\beta_i>0$	$\alpha_i\beta_i<0$	$\alpha_i=0$	$\beta_i=0$
0	0	false	0	0	0	0	0
0	0	true	0	0	0	0	0
0	1	false	0	0	0	0	0
0	1	true	ε	-1	+1	±1	0
1	0	false	ε	-1	+1	±1	0
1	0	true	ε	+1	-1	0	±1
1	1	false	ε	+1	-1	0	±1
1	1	true	ε	+1	-1	0	±1

where f is the fitness function, x and best are a solution and the best solution respectively, x_i and $best_i$ are the i-th bit component of x and best.

In this section, we will compare different bitmasks found in each cycle of TEA using QGAs in our study and using GAs in [13] and [14].

3.1 One-Cycle TEA

We used the following fitness function as in [13] and [14],

$$fitness = \begin{cases} w^4, if\ x^2 = 522,240 \\ x^2,\ \ otherwise \end{cases} \tag{1}$$

where w represents the weight, the number of 1's, of the bitmask. This fitness function was first introduced in [6], but incorrectly used 522,480 in place of 522,240. This piece-wisely defined fitness function aims to find bitmasks that have maximal deviation from a uniform probability distribution.

For one-cycle TEA, we found bitmasks that had maximal deviation from the random distribution with x^2 = 522.240. In [13], the authors found their best solution at weight 153, and [14] found their best solutions to be at weights 154 and 155.

The bitmasks of higher weight are preferred since they permit a bigger set of inputs to be used for the test. In [13] authors used a GA with a population size of 100 to find the best bitmask of weight 153 and in [14] authors used a GA with a population size of 185 to find the best bitmasks of weights 154 and 155. To provide a baseline for comparison of different GA techniques, we ran our QGA with a population size of 100 to find the best bitmasks of weights ranging from 151 to 155, which are listed in Table 2. The two bitmasks of weights 151 and 152 in Table 2 were not reported in [13] and [14]. Because one-cycle TEA is relatively easier to break, all the bitmasks in Table 2 have their x^2= 522.240, which is the maximal value for this statistic.

3.2 Two-Cycle TEA

Since the two-cycle TEA is more difficult than one-cycle TEA, no bitmasks of heavy enough weights can produce the maximal deviation of 522,240. In [13] and [14], authors modified the fitness function in Equation (1) to create the following fitness function to break the two-cycle TEA,

$$fitness = \begin{cases} \frac{1}{w^3} + w^4, if\ x^2 \geq 403.4579 \\ \frac{1}{w^3},\ \ otherwise \end{cases} \tag{2}$$

The idea behind this fitness function is to divide thesearch process of GA into two steps. The first step is to find bitmasks with weights above the thresholdvalue 403.4579, which is about 0.5 percentile of all x^2 values and has a P-value of $5*10^{-9}$. The second is to increase the weights of those bitmasks.

Table 2. Our results of QGA on one-cycle TEA

Bitmask	x^2	Weight
{0xFFFFFF00,0xFFFFE000, 0xFFFFFF00,0xFFFFFF00, 0xFFFFFFFF, 0xFFFFFFFF}	522,240	155
{0xFFFFFF00,0xFFFFE000, 0xFFFFFF00,0xFFFEFF00, 0xFFFFFFFF, 0xFFFFFFFF}	522,240	154
{0xFFFFFF00,0xDFFFE000, 0xFFFFFF00,0xFFFEFF00, 0xFFFFFFFF, 0xFFFFFFFF}	522,240	153
{0xFFFFFF00,0xDFFFE000, 0xFFFFFF00,0xFFFEFF00, 0xFFFFFFFE, 0xFFFFFFFF}	522,240	152
{0xFFFFFF00,0xDFFFE000, 0xFFFFFF00,0xFFFEFF00, 0xFFFFFFFF, 0xFF7FFFDF}	522,240	151

Table 3. Results of GA on two-cycle TEA in [14]

Weight	x^2
157	459.6417
155	483.6
158	474.8167
145	486.2333
159	451.3917
158	415.8583
160	422.475
157	435.6833
162	488.3333
159	413.8417

Table 4. Our results of QGA on two-cycle TEA

Bitmask	x^2	Weight
{0xFFFFD7FF,0xFFDFFBFF,0xFFFCADF9, 0xFFFFFBCF, 0xFFFFFF5E, 0xFFFFFB8B}	611.925	170
{0xFFFFD7FF, xFFDFFBFF,0xFFFCAC75, 0xFFFFDFCF, 0xFFFFFF5E, 0xFFFFFFCB}	606.725	170
{0xFFFFD7F7, 0xFF9FFBFF, 0xFFFEAFF1, 0xFFFFFFCF, 0xFFFFFF5E, 0xFFFFFF8B}	588.875	171
{0xFFFFD7F7,0xFF9FFBFF,0xFFFCBD7B, 0xFFFFFFCF, 0xFFFFFF5E, 0xFFFFFB8F}	572.375	171
{0xFFFFD7FF,0xFF9FFBFF, 0xFFFCAD75, 0xFFFFFBEF, 0xFFFFFF5E, 0xFFFFFF9B}	538.5	171
{0xFFFFD7F7, 0xFF9FFBFF, 0xFFFFAC77, 0xFFFFDFCF, 0xFFFFFF7E, 0xFFFFFF8B}	578.4	171
{0xFFFFD7FF,0xFF9FFBFF, 0xFFFEBDF1, 0xFFFFFBCF, 0xFFFFFF5E, 0xFFFFFB9B}	662.075	171
{0xFFFFD7FF,0xFF9FFBFF, 0xFFFCAD75, 0xFFFFFBEF, 0xFFFFFF5E, 0xFFFFFF9B}	628.125	172
{0xFFFFF7FF,0xFF9FFBFF, 0xFFFCAD73, 0xFFFFFBCF, 0xFFFFF5DE, 0xFFFFFFBF}	635.725	172
{0xFFFFF7F7,0xFFDFFBFF,0xFFFCADF7, 0xFFFFFBCF,0xFFFFFFDF, 0xFFFFFB8B}	598.075	173

In [13], authors employed the fitness function defined in Equation (2) to find the following best bitmask with a weight of 155 and an average x^2 statistic of 508.15 on 30 random input-key datasets:

{0xBFFFF0FA, 0xFFFE7388, 0xFFFFF7F8,
0xFFFFF3F8, 0xFFFFEF85, 0xFFFFEF8C}

In [14] authors found ten bitmasks using the fitness function in Equation (2) and calculated the average x^2 statistic across 30 different random input-key datasets, each having 2^{11} input-key pairs. Their results are summarized in Table 3.

In [13] and [14], both authors used the same threshold in the fitness function as in Equation (2) for two-cycle, three-cycle, and four-cycle TEAs, and the bitmasks found for four-cycle TEA were not usable due to their low weights. We suspected that using a different threshold in the fitness function for each cycle might be more appropriate since the average x^2 values of various cycles are different. Based on this belief, we selected different thresholds in the fitness function for each different cycle.

We used the following fitness function for two-cycle TEA,

$$fitness = \begin{cases} w^4, if \ x^2 > 1100 \\ x^2, \ \ otherwise \end{cases} \tag{3}$$

The idea behind this fitness function is to ensure the minimum value for x^2 first, then find a bitmask of large weight.

Our QGA discovered ten bitmasks whose average x^2 statistic across 30 different random input-key datasets and weight are included in Table 4.

In Table 4, the average x^2 statistic was 602 and the average bitmask weight was 171, whereas the results from [14] in Table 3 had corresponding values of 453.3875 and 157 respectively.

Our results in Table 4 demonstrated a big improvement over those in [13] and [14]. As the cycles of TEA increase, our QGAs show their apparent advantage over GAs as illustrated in the following sections. In all the subsequent experiments below, we used a QGA with population size of 100, generation number of 200, and $\varepsilon = 0.001\pi$ in rotational mutation.

3.3 Three-Cycle TEA

For three-cycle TEA, authors in [14] used the same fitness function defined in Equation (2) as for the two-cycle TEA to find ten bitmasks. Their average x^2 statistic across 30 different random input-key datasets and weight are presented in Table 5.

In [13], authors used fitness function defined in Equation (2) to get the following best bitmask with a weight of 116 and an average x^2 statistic of 466.5 on 30 random input-key datasets:

{0xFFE1F040, 0x FCE70446, 0x FFEFF06E,
0x FFE7F42A, 0x FFBF1825, 0x FFFA0064}

We identified ten bitmasks using the following fitness function for three-cycle TEA,

$$fitness = \begin{cases} w^4, if \ x^2 > 900 \\ x^2, otherwise \end{cases} \tag{4}$$

The only difference between this function and that in Equation (3) is the threshold employed in the function definition. The information about these bitmasks is summarized in Table 6. The average X^2 statistic was 530.756 and the average bitmask weight was 117.8 in Table 6, while the results from [14] in Table 5 had corresponding values of 420.8242 and 100.2 respectively.

Table 5. Results of GA on three-cycle TEA in [14]

Weight	x^2
99	427.0
100	432.675
105	413.0417
109	437.7333
104	423.025
93	445.1333
100	396.1917
105	420.7333
79	437.2667
108	375.4417

For two-cycle and three-cycle TEAs, we obtained better bitmasks than those found in [13] and [14] in terms of both chi-square statistic and weight.

3.4 Four-Cycle TEA

The task of finding efficient bitmasks becomes more complicated as the cycles of TEA increase. The approaches in [13] and [14] were sufficient to find efficient bitmasks for TEA of cycles less than four, but failed to attack TEA of cycles greater than or equal to four.

In [13], using the fitness function in Equation (2) authors found bitmasks of relatively low weights, less than 47. They then took up a different approach. Instead of using chi-square statistic, they used Strict Avalanche Criterion (SAC), a more sensitive measure, to assess the deviation of the output of TEA from randomness. The best bitmask they found was

{0x96922A0C, 0x42C06402, 0x35B11001,
0x97000000, 0xF0000001, 0xBEB00001}

with a weight of 50 and an average x^2 statistic of 673.40 on 30 random input-key datasets. Since TEA takes input data of 64 bits, any bitmask of weight less than 64 cannot be useful for different cryptanalysis of TEA.

In [14], authors were unable to find any useful bitmasks for four-cycle TEA. They suspected that with more rounds of calculations in their GA, it might be possible to discover some adequate bitmasks.

Based on the principle that we should approach each cycle differently, the following fitness function was applied to four-cycle TEA,

$$fitness = \begin{cases} w^4, if \ x^2 > 800 \\ x^2, \ \ otherwise \end{cases} \tag{5}$$

Our QGA uncovered five bitmasks. For each of these bitmask, we computed the average x^2 statistic across 30 random input-key datasets. The results are listed in Table 7. All these x^2 statistic values have a P-value less than $5*10^{-9}$.

Table 6. Our results of QGA on three-cycle TEA

Bitmask	x^2	Weight
{0xF7D65CE6, 0x10FCA894,0xF2ABBFDD, 0xFF0557BB, 0xFF867C02, 0xFFD7E73D}	554.3	120
{0x77D65CE6, 0x10FCA894, 0xF2ABBFDD, 0xFF0557BB, 0xFF867C02, 0xFFD7E73D}	518.74	119
{0x77D65CE6, 0x10FCA894, 0xF2ABBFDD, 0xDF0557BB, 0xFF867C02, 0xFFD7E73D}	536.51	118
{0x77D65CE6, 0x10FCA894, 0xE2ABBFDD, 0xFF0557BB, 0xFF867C02, 0xFBD7E73D}	540.11	117
{0x77D65CE6, 0x10FCA894, 0xE2ABBFDD, 0xDF0557BB, 0xFF867C02, 0xFBD7E73D}	547.80	116
{0xF1729F86, 0x97B6EC6F, 0xFB5A1EE0, 0xFFD328F4, 0xFFE4408C, 0xFFB1FDEA}	542.23	117
{0xF1729F86, 0x97B6EC6F, 0xFB5A1EE0, 0xFFD328F4, 0xFDE4408C, 0xFFB1FDEA}	540.65	116
{0xF38FA5FB, 0xF7E44E4B, 0xF483FB22, 0xF23FE071, 0xFFE1C64D, 0xFFCF5074}	559.45	116
{0xF77C99E2, 0xD157C8BC, 0x7C79BF35, 0x9555D5F2, 0xFFFECA55, 0xCDDDABE5}	482.56	120
{0x773C98EF, 0xD92FCEBC, 0x5C79BF15, 0x955D55F2, 0xFFFECA45, 0xCDDDABC5}	485.21	119

Table 7. Our results of QGA on four-cycle TEA

Bitmask	x^2	Weight
{0x504007C7, 0xB03C5091, 0x84AE8212, 0x026029C7, 0x411BA198, 0xC81074B8}	740.22	69
{0xF407DC1C, 0x7A123211, 0x8F1042AE, 0x8040A0BE, 0x90017A89, 0x20C204C0}	749.12	69
{ 0x4520B630, 0x0A36E920, 0x0D051868, 0x0AEC3868, 0x2312C768, 0x2460F804}	721.33	69
{0x54D3A2C4, 0x901722EC, 0xE02B0591, 0x21D00283, 0x57848409, 0x49114082}	735.26	67
{0x3E2C642A, 0x80443210, 0xB446B064, 0x87250417, 0x0C93E181, 0x12040508}	767.23	64

The output v[0] & 255 of the first bitmask in Table 7, from two separate sample runs of TEA on one random input-key dataset of 2^{11} pairs, is displayed in the form of histograms in Figure 1. As illustrated in Figure 1, there is a clear peak or bias at the same position 152 for both runs although the frequencies at all other positions are relatively the same. The x^2 statistic values produced by these two sample runs of TEA were 927 and 941 respectively. The significance of these x^2 statistic values, which measure the deviation of TEA output from randomness, can be evaluated by their P-value. We thought it is more helpful if the plots like those in Figure 1 can be exhibited.

3.5 Five-Cycle TEA

In both [13] and [14], no results were reported for five-cycle TEA. We used the following fitness function in this case,

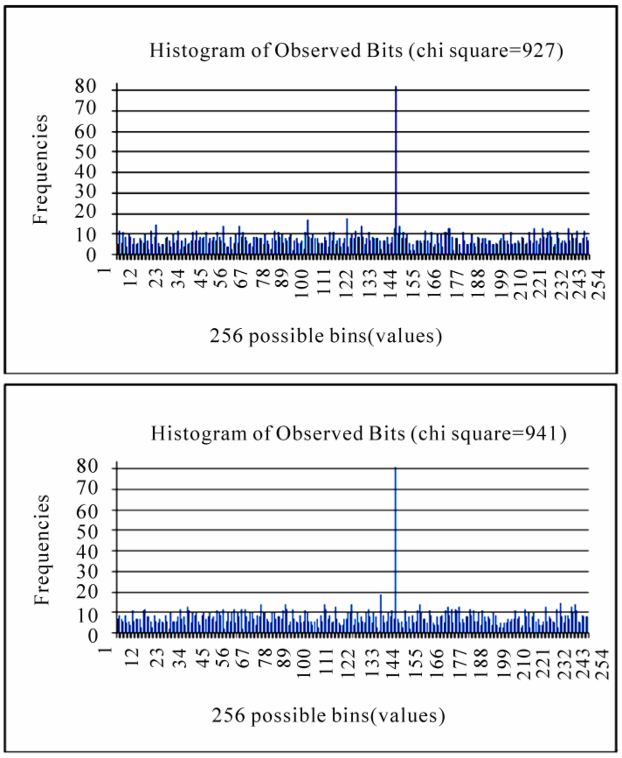

Figure 1. The two plots show the histograms of the output of the first bitmask in Table 7. The x-axis represents the possible 256 positions, and the y-axis represents the frequencies of the bit patterns of TEA output at various positions

$$fitness = \begin{cases} w^4, if\ x^2 > 700 \\ x^2,\ \ otherwise \end{cases} \qquad (6)$$

We found the following bitmask:

{0xE4822346, 0x830CA317, 0xCE9522DC, 0x3E13C130, 0x33C18B0A, 0x128A11A0}

This bitmask has a weight of 76, an average x^2 statistic of 631.74 on 30 random input-key datasets, and a P-value less than $5*10^{-9}$.

For five-cycle TEA, we only reported one bitmask that has a high chi-square statistic and a high weight. It was not the intent of our current study to conduct an exhaustive search of all bitmasks of interest, but rather to demonstrate the effectiveness of QGAs in the cryptanalysis of TEA.

4. Conclusions

In this paper, QGAs were utilized in the cryptanalysis of TEA. We not only significantly improved the results in [13] and [14] in terms of both bitmask chi-square statistic and weight, but also were able to break TEA of cycles greater than or equal to four, a challenge previous studies could not resolve. With these improved bitmasks, efficient distinguishers for TEA can be constructed. These distinguishers require few inputs to get high distinguishing probability [13]. Our success, we believed, was based on designing new fitness functions and the fact that the qubit chromosomes in QGAs are more informative than the bit chromosomes of same length in traditional GAs.

5. Acknowledgments

We thank Houghton College for its financial support and Dr. Aaron Garrett of Jacksonville State University for helpful discussion.

REFERENCES

[1] D. Wheeler, and R. Needham, “TEA, a tiny encryption algorithm,” Proceedings of the 1995 Fast Software Encryption Workshop, Springer-Verlag, pp. 97–110, 1995.

[2] S. J. Shepherd, “The tiny encryption algorithm,” Journal of Cryptologia, Vol. 31, No. 3, pp. 233–245, July 2007.

[3] D. Wagner, J. Kelsey, and B. Schneier, “Related-key cryptoanalysis of 3-WAY, Biham-DES, CAST, DES-X, NewDES, RC2 and TEA,” Proceedings of the IClCS’97 Conference, Springer-Verlag, pp. 233–246, 1997.

[4] F. Yang, J. Song, and H. Zhang, “Quantitative cryptanalysis of six-round DES using evolutionary algorithms,” Proceedings of the 3rd International Symposium on Advances in Computation and Intelligence, LNCS Vol. 5370, Springer-Verlag, pp. 134–141, 2008.

[5] E. Biham and A. Shamir, “Differential cryptanalysis of DES-like cryptosystems,” CRYPTO’90, LNCS 537, Springer-Verlag, pp. 2–21, 1991.

[6] J. Song, H. Zhang, Q. Meng, and Z. Wang, “Cryptanalysis of two-round DES using genetic algorithms,” ISICA’07: International Symposium on Intelligence Computation and Applications, LNCS, Springer-Verlag, Vol. 4683, pp. 583–590, 2007.

[7] J. Song, H. Zhang, Q. Meng, and Z. Wang, “Cryptanalysis of four-round DES based on genetic algorithms,” Proceedings of the International Conference on Wireless Communications, Networking and Mobile Computing, Springer-Verlag, LNCS, Vol. 4683, pp. 583–590, 2007.

[8] J. Holland, “Adaptation in natural and artificial systems,” Ann Arbor, MI: University of Michigan Press, 1975.

[9] S. Hong, D. Hong, Y. Ko, D. Chang, W. Lee, and S. Lee, “Differential cryptanalysis of TEA and XTEA,” ICISC 2003, LNCS 2971, Springer-Verlag, pp. 402–417, 2004.

[10] E. Biham, A. Biryukov, and A. Shamir, “Cryptanalysis of skipjack reduced to 31 rounds using impossible differentials,” Advances in Cryptology – EUROCRYT’99, LNCS, Springer-Verlag, Vol. 1592, pp. 12–23, 1994.

[11] D. Moon, K. Hwang, W. Lee, S. Lee, and J. Lim, “Impossible differential cryptanalysis of reduced round XTEA and TEA,” Fast Software Encryption, LNCS, Springer-Verlag, Vol. 2365. pp. 49–60, 2002.

[12] L. Knudsen and W. Meier, “Correlations in RC6 with a reduced number of rounds,” Proceedings of the Seventh Fast Software Encryption Workshop, Springer-Verlag, 2000.

[13] J. C. Hernandez and P. Isasi, “Finding efficient distinguishers for cryptographic mappings, with an application to the block cipher TEA,” Proceedings of the 2003 Congress on Evolutionary Computation CEC2003, pp. 341–348, IEEE Press, 2003.

[14] A. Garrett, J. Hamilton, and G. Dozier, “Genetic algorithm techniques for the cryptanalysis of TEA,” International Journal on Intelligent Control and Systems Special Session on Information Assurance. Vol. 12, pp. 325–330, 2007.

[15] J. J. Grefenstette, “Optimization of control parameters for genetic algorithms,” IEEE Transactions on Systems, Man, and Cybernetics, Vol. 16, No. 1, pp. 122–128, 1986.

[16] R. Penrose, “Shadows of the mind,” Oxford University Press, 1994.

[17] K. H. Han and J. H. Kim, “Introduction of quantum-inspired evolutionary algorithm,” in Proceedings of the 2002 FIRA Robot World Congress, pp. 243–248, May 2002.

[18] S. Y. Yang and L. C. Jiao, “The quantum evolutionary programming,” Proceedings of the 5th International Conference on Computational Intelligence and Multimedia Applications (ICCIMA’03), pp. 362–367, 2003.

[19] National Bureau of Standards, Data Encryption Standard, U.S. Department of Commerce, FIPS, Vol. 46, January 1977.