An Improved Automatic Search Method for Differential Trails in TEA Cipher

Kaihao Chen[1], Xiaoming Tang[1], Peng Xu[2], Man Guo[3], Weidong Qiu[1], Zheng Gong[4,5]
(Corresponding author: Peng Xu)

School of Information Security Engineering, Shanghai Jiao Tong University[1]
Dongchuan Road 800 Minhang, 200240 Shanghai, P.R. China
Institute of Information Technology, Police Training Institute[2]
250002 Shandong, P.R. China
(Email: 18053122000@189.cn)
Torch High Technology Industry Development Center, Ministry of Science and Technology[3]
School of Computer Science South China Normal University[4]
State Key Laboratory of Information Security, Institute of Information Engineering[5]
(Received Dec. 23, 2014; revised and accepted July 4 & Aug. 12, 2015)

Abstract

TEA (Tiny Encryption Algorithm) is a block cipher with simple ARX (addition, rotation, exclusive-OR) based Feistel network, designed for both hardware and software scenario. Inspired by the auto search algorithm for ARX cipher introduced by Biryukov and Velichkov in 2014, we proposed an improved version of auto search algorithm for ARX cipher and verified in block cipher TEA. By introducing a sorted partial difference distribution table (sorted pDDT), our algorithm can eliminate lots of branches in advance during differential trail search phase. As time of the search algorithm increase exponentially with the rounds increasing, our algorithm make a great improvement in the search performance.

Keywords: ARX, automatic search, differential trail, sorted pDDT, TEA

1 Introduction

In the past decades, lots of light weight primitives are published for the application of limited resource scenario. For the purpose of simplicity, some of those light weight block ciphers, such as TEA [12], and XTEA [8], use the ARX based design concept, which is combined by a small set of simple operations such as modular addition, bit shift and XOR (exclusive-OR). The ARX based design concept is simple and can be implemented efficiently both in software and hardware.

For the cryptanalysis of ARX based primitives, many techniques is used for the security evaluation such as rotational cryptanalysis [3], integral Zero-Correlation attack [11], etc. However Differential cryptanalysis is still a basic tools during the cryptanalysis of ARX cipher. In differential attack, attackers try to recover the secret key by exploiting the high differential probability pattern of the cipher. Finding a good differential pattern became a key step of the attack. Originally, the search procedure of a good differential pattern is a manual work, therefore the search of high probability differential alway fails for modern block cipher due to the high diffusion properties.

In [6], a branch-and-bound strategy is firstly applied to an automatic search approach of differential and linear trail proposed by Matsui. The author demonstrate a search algorithm for the differential and linear trail and practically apply to block cipher DES. Matsui's algorithm treated the the search procedure of differential and linear trail as a search over a tree of possible solution, in which every non-linear operation is a node with many possible difference or linear mask. By cutting off the paths which is not leading to an optimal solution, the author efficiently derive the best differential and linear trail of DES.

In [2], Biryukov and Velichkov proposed the first extension of Matsui's automatic search algorithm for differential trails in ARX based block cipher. Since the DDT (differential distribution table) for one round ARX operation requires $2^{3n} \times 4$ bytes of memory for n bit block size, it is unlikely to delivery an optimized result using the original Matsui's search algorithm on current computation power. To solve this memory and computation complexity issue, the author introduced a pDDT (partial differential distribution table) to search algorithm, which contains all the high probability differentials with respect to a fixed threshold. By using pDDT and high way & country road concept, the author successfully find the first full differential trails for block cipher TEA. The best result covers 18 round with one round advance comparing to the best differential attack on TEA(17 rounds). Also the author

present a lot of differential trails search result in XTEA, RAIDEN [9] and SPECK [1].

Contribution. In this paper, we proposed an improved version of threshold search algorithm based on Biryukov's differential search algorithm. By using a sorted pDDT approach, our improved version of search algorithm can cut off more unnecessary branches in advance. Comparing to the original search algorithm, our search algorithm is more efficiency and make a great improvement in the time complexity. We verify the search result practically in block cipher TEA, and we present the detail of comparison results with original search algorithm in the second part of our paper.

The contents of this paper are organized as follows. Part 2 gives a brief introduction of block cipher TEA and differential. Part 3 introduces automatic search for differential trail in ARX ciphers. In Part 4, we give a detailed description of our improved version of differential search algorithm and analyze the improvement it makes. We did the experiments in Part 5 and show the comparison with original algorithm. At last we give our conclusion in Part 4.

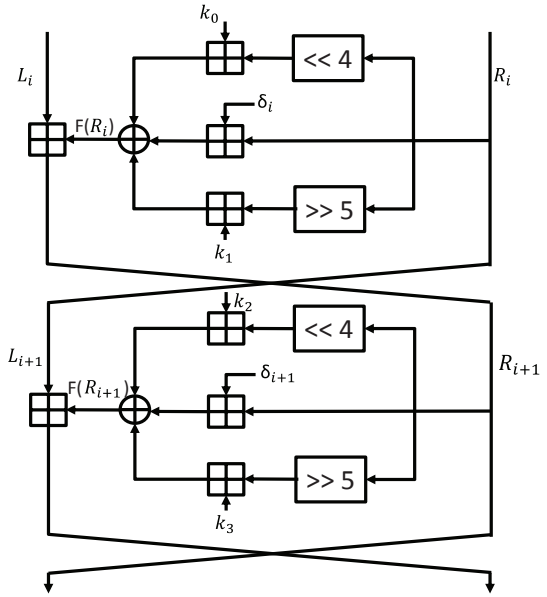

Figure 1: Two-round Feistel structure of TEA

2 Preliminaries

2.1 Notation

We use following notions in this paper:

L_i:	The left half of the i round input.
R_i:	The right half of the i round input.
K:	128-bit master key.
k_i:	The round key ($i = 0, 1, 2, 3$).
δ_i:	The round constant used in the i-th round.
$+$:	Addition modular 2^n
$\oplus$:	Exclusive-OR
$x\|y$:	Concatenation of two bit strings x and y.
$\#A$:	The number of elements in the set A
B_n:	Probability of the best n-round trail
$\widehat{B}_n$:	Probability of the best found n-round trail
$\bar{B}_n$:	An estimation for the best n-round probability

2.2 A Brief Introduction of Block Cipher TEA

Block cipher TEA was designed by Wheeler and Needham of the Cambridge Computer Lab and first presented at the FSE in 1994 [12]. It has a Feistel structure with a ARX based F function. Block cipher TEA takes a 64-bit plaintext as input and has a total of 64 iteration rounds. For the purpose of less memory usage, the round keys of block cipher TEA are derived from the 128-bit master key directly as $K = k_3|k_2|k_1|k_0$. A round constant δ_i is added in every round to prevent the slide attacks. We define the F-function as (for odd rounds):

$$F(x) = ((x \ll 4) + k_0) \oplus (x + \delta_r) \oplus ((x \gg 5) + k_1)$$

The encryption procedure are depicted more intuitively in Figure 1. Given the inputs L_i and R_i of round i, the output is calculated as: $L_{i+1} = R_i$, $R_{i+1} = L_i + F(R_i)$. The operations used in its Feistel structure is modular addition instead of XOR. More details of block cipher TEA, readers can refer to the paper [12].

3 Differential Analysis in ARX and Threshold Search

3.1 Differential Analysis in ARX Based Cipher

In traditional differential cryptanalysis, differential trail is considered using XOR differences. However, for lots of primitives, such as TEA, additive differences are more appropriate for the differential cryptanalysis since the round key and round constants are add-ed. In such primitives, the estimation of differential probability using ADD is more accurate than using XDP (XOR differential probability). Moreover, the number of ADD vs. XOR operations in TEA in one round is larger and then more components are linear in the round function, it is more suitable to use ADD difference instead of XOR difference in differential cryptanalysis in such primitives. The definition of ADP (addition differential probability) as below:

$$\mathrm{adp}^{\oplus}(\alpha, \beta \to \gamma) =$$
$$2^{-2n} \cdot \#\{(x, y) : ((x + \alpha) \oplus (y + \beta)) - (x \oplus y) = \gamma\}$$

In paper [4] and [5], the efficient computation method of probabilities $\mathrm{adp}^{\oplus}$and xdp^{+} have been demonstrated, and further generalized using S-function concept in [7] and [10]. We will not present a detailed description here due to the space limitation.

3.2 Partial Difference Distribution Table (pDDT)

Finding a good differential trail remains a great challenge in ARX based block cipher cryptanalysis. Comparing with S-box (substitution box) based block cipher, the ARX design concept uses the operations which is based on bit-level instead of word-level operations of S-box based design. In cryptanalysis of those S-box based block ciphers, the analysis of differential trails is usually considered at word level, and the upper bound of differential trails can be estimated easily through counting of active s-box. However, for the ARX based block cipher, it is inconvenient to estimate the upper bound of differential trails in traditional way since the size of differential table of ARX based block cipher can be considered as extremely large. As a sacrifice of completeness, Biryukov and Velichkov proposed a partial DDT with a pre-defined threshold probability $\mathbf{p}_{\text{thres}}$ in [2] to reduce the search space, which is defined as:

$$(\alpha, \beta, \gamma) \in \text{pDDT} \Leftrightarrow \text{DP}(\alpha, \beta \to \gamma) \geq \mathbf{p}_{\text{thres}}$$

In [2], Biryukov and Velichkov have proved that xdp^+ and $\text{adp}^{\oplus}$ are monotonously decreasing with the word size n bits:

$$p_n \leq p_{n-1} \leq \cdots \leq p_k \leq \cdots \leq p_1 \cdots p_0. \quad (1)$$

In Equation (1), $p_k = \text{DP}(\alpha_k, \beta_k \to \gamma_k)$, $1 \leq k \leq n$. With the property above, they proposed an algorithm using a recursive procedure rather than exponential computation to compute the pDDT. The time complexity of the algorithm depends on $\mathbf{p}_{\text{thres}}$.

3.3 Threshold Search

In [2], Biryukov and Velichkov proposed an extended algorithm to search for the best found differential trail in ARX ciphers named Threshold Search, which is similar to Matsui's branch-and-bound algorithm [6]. The probability of the best found trail so far for the first n rounds is notated as $\bar{B}_n$. Respectively, the probability of the best found trail is $\widehat{B}_n$. Besides, they came up with the concept of highways and country roads. Highways are differentials in pre-computed pDDT H with the probabilities above a pre-defined threshold $\mathbf{p}_{\text{thres}}$, while country roads are differentials with fixed input difference computed on-demand in an intervening round. The country roads are stored in table C with probabilities lower than $\mathbf{p}_{\text{thres}}$ and satisfy following two conditions:

1) Their probabilities are above a minimum value that may still improve $\widehat{B}_n$ i.e. in round r, the country road (α_r, β_r, p_r) (suffix r means round r instead of r-bit word) satisfies that $p_r \geq p_{min} = \bar{B}_n/(p_1 p_2 \cdots p_{r-1} \widehat{B}_{n-r})$;

2) Their output differences β_r ensure that the input difference for the next round is in H i.e. $\alpha_{r-1} + \beta_r = \alpha_{r+1} \in H$ such that it prevents the overload of the size of C.

In Threshold Search, the first two rounds search only explore differential in H as the input differences and the output differences can be freely chosen, therefore restricting the search range into H ensures the search for differential in these rounds are not overloaded. However, from the third round, the input difference is fixed due to the Feistel structure, the algorithm explores differential not only in H but also in C. If one trail search reaches the last round successfully, $\bar{B}_n$ will be updated. The final best found differential trail usually contains as many differentials in H as possible.

4 Improved Differential Search

4.1 Improved Threshold Search Using Sorted pDDT

Our improved Threshold Search is also a branch-and-bound algorithm based on Biryukov's algorithm. We introduced a sorted pDDT to search algorithm to improve elimination rate of unnecessary branches. One of key factors for time complexity of the threshold search is the branch numbers during search phase. If we can cut off the unnecessary branches in earlier phase, the time we used for differential trail search will be less. Therefore, we propose a sorted pDDT to cut off most unnecessary branches in advance. In sorted pDDT, the entries are sorted by input differences in ascending order, probabilities in descending order and output differences in ascending order in sequence. The pseudo-code is given in Algorithm 1.

Temporary table C is used as a pDDT with threshold p_{min} when $p_{min} < \mathbf{p}_{\text{thres}}$. We assume that if the largest probability in certain round r is chosen at the beginning stage, it has more chance to reach last round and p_{min} will be updated as well as $\bar{B}_n$. If p_{min} increased, less entries in H or C need to be traversed probably. It's easy to stop traversing efficiently if p_{min} isn't satisfied due to their sorting sequence. In original algorithm, C and H may have some entries in common. To solve this issue, we add maximum probability condition i.e. $\mathbf{p}_{\text{thres}}$ when building C to avoid traversing repeated entries.

We optimize the algorithm in the following routine:

1) Process in the first two and the last round is the same as Threshold Search;

2) From the third round onwards to round r, determine the required minimum probability, i.e. $p_{min} = \bar{B}_n/(p_1 p_2 \cdots p_{r-1} \widehat{B}_{n-r})$;

3) If $p_{min} \geq \mathbf{p}_{\text{thres}}$, it is unnecessary to calculate table C, and we only explore the entries in H with certain fixed input difference α_r and probabilities above p_{min} rather than all with certain fixed input difference, since H is a sorted pDDT;

Algorithm 1 Improved Threshold Search Based on Sorted pDDT

Input: **n**: number of rounds; **r**: current round; **H**: sorted pDDT; $\widehat{\mathbf{B}} = (\widehat{\mathbf{B}}_\mathbf{1}, \widehat{\mathbf{B}}_\mathbf{2}, \cdots, \widehat{\mathbf{B}}_{\mathbf{n-1}})$: probabilities of best found trails for the first $(n-1)$ rounds; $\widehat{\mathbf{B}}_\mathbf{n}$: initial estimate; $\bar{\mathbf{T}} = (\bar{\mathbf{T}}_\mathbf{1}, \bar{\mathbf{T}}_\mathbf{2}, \cdots, \bar{\mathbf{T}}_{\mathbf{n-1}})$: trail for n rounds with probability $\bar{B}_n$; $\mathbf{p}_{\text{thres}}$: probability threshold.
Output: $\widehat{\mathbf{B}}_\mathbf{n}, \widehat{\mathbf{T}} = (\widehat{\mathbf{T}}_\mathbf{1}, \widehat{\mathbf{T}}_\mathbf{2}, \cdots, \widehat{\mathbf{T}}_{\mathbf{n-1}})$: trail for n rounds with probability $\widehat{B}_n$.

function THRES_SEARCH_SORTED_PDDT($n, r, H, \widehat{B}, \bar{B}_n, \bar{T}$, p$_{\text{thres}}$)
  if $((r = 1) \vee (r = 2)) \wedge (r \neq n)$ **then**
    for all (α, β, p) in H **do**
      $p_r \leftarrow p, \widehat{B}_n \leftarrow p_1 p_2 \cdots p_r \widehat{B}_{n-r}$
      if $\widehat{B}_n \geq \bar{B}_n$ **then**
        $\alpha_r \leftarrow \alpha, \beta_r \leftarrow \beta$
        add $\widehat{T}_r \leftarrow (\alpha_r, \beta_r, p_r)$ to $\widehat{T}$
        THRES_SEARCH_SORTED_PDDT($n,\ r+1,\ H,\ \widehat{B},\ \bar{B}_n,\ \widehat{T}$, p$_{\text{thres}}$)
      end if
    end for
  end if
  if $(r > 2) \wedge (r \neq n)$ **then** $\alpha_r \leftarrow (\alpha_{r-2} + \beta_{r-1})$
    $P_{r-1} \leftarrow p_1 p_2 \cdots p_{r-1} \widehat{B}_{n-r}, p_{r,min} \leftarrow \bar{B}_n / P_{r-1}$
    if $p_{r,min} \leq 1.0$ **then**
      for all $\beta_r : (p_r(\alpha_r \rightarrow \beta_r) \geq p_{r,min}) \wedge ((\alpha_r, \beta_r, p_r) \in H)$ **do** add $\widehat{T}_r \leftarrow (\alpha_r, \beta_r, p_r)$ to $\widehat{T}$
        THRES_SEARCH_SORTED_PDDT($n,\ r+1,\ H,\ \widehat{B},\ \bar{B}_n,\ \widehat{T}$, p$_{\text{thres}}$), $p_{r,min} \leftarrow \bar{B}_n / P_{r-1}$
      end for
      if $p_{r,min} <$ p$_{\text{thres}}$ **then** $C \leftarrow \emptyset$
        for all $\beta_r : (p_r(\alpha_r \rightarrow \beta_r) \geq p_{r,min}) \wedge ((\alpha_{r+1} + \beta_r) = \gamma \in H)$ **do** add (α_r, β_r, p_r) to C
        end for
        if $(C = \emptyset) \wedge ((\beta_r, p_r) \leftarrow p_r = \max_\beta p(\alpha_r \rightarrow \beta) \geq p_{r,min})$ **then** add (α_r, β_r, p_r) to C
        end if
        if $(C \neq \emptyset)$ **then**
          for all $(\alpha, \beta, p) \in C \wedge (p_{r,min} <$ p$_{\text{thres}})$ **do** add $\widehat{T}_r \leftarrow (\alpha_r, \beta_r, p_r)$ to $\widehat{T}$
            THRES_SEARCH_SORTED_PDDT($n,\ r+1,\ H,\ \widehat{B},\ \bar{B}_n,\ \widehat{T}$, p$_{\text{thres}}$), $p_{r,min} \leftarrow \bar{B}_n / P_{r-1}$
          end for
        end if
      end if
    end if
  end if
  if $r = n$ **then** $\alpha_r \leftarrow (\alpha_{r-1} + \beta_{r-2})$
    if $(\alpha_r$ in $H)$ **then**
      $(\beta_r, p_r) \leftarrow p_r = \max_{\beta \in H} p(\alpha_r \rightarrow \beta)$
    else $(\beta_r, p_r) \leftarrow p_r = \max_\beta p(\alpha_r \rightarrow \beta)$
    end if
    $p_n \leftarrow p_r, \widehat{B}_n \leftarrow p_1 p_2 \cdots p_n$
    if $\widehat{B}_n \geq \bar{B}_n$ **then** $\alpha_n \leftarrow \alpha_r, \beta_n \leftarrow \beta$
      add $\widehat{T}_n \leftarrow (\alpha_n, \beta_n, p_n)$ to $\widehat{T}$
      $\bar{B}_n \leftarrow \widehat{B}_n, \bar{T} \leftarrow \widehat{T}$
    end if
  end if
  $\widehat{B}_n \leftarrow \bar{B}_n, \widehat{T} \leftarrow \bar{T}$
  return $\widehat{B}_n, \widehat{T}$
end function

4) If $p_{min} < \mathbf{p}_{\text{thres}}$, table C may be needed but we won't compute it immediately. We search for the entries with the fixed input difference in table H and choose the largest one to proceed to the next round first if we sucessfully find them. When the algorithm backtracks gradually to round r, the p_{min} may be updated and we choose the next entry in table H which still satisfies p_{min};

5) After exploring table H and if still $p_{min} < \mathbf{p}_{\text{thres}}$, we begin to compute a pDDT C under the conditions mentioned in Section 3.3;

6) Explore the entries in table C in pre-defined order, update p_{min} dynamically when the recursion backtracks, and terminate if the probability does not satisfy p_{min};

7) After exploring table H and C if needed, recursion returns and backtracks to the previous round i.e. $(r-1)$.

Compared with original algorithm, the sorted pDDT is required to be calculated instead of pDDT. Since it's easy to implement the sorted list with the underlying programming languages like C/C++ and inserting and finding operations in sorted list costs little extra time compared with ordinary list, the proposed idea have less extra computation costs.

In addition, we add P_{r-1} as the multiplication of constructed $(r-1)$-round trail probability and best found trail probability for the first $(n-r)$ rounds. Being similar as Biryukov's Threshold Search, Algorithm 1 operates by recursively extending a trail from i rounds to $(i+1)$ rounds, beginning with $i = 1$ and terminating at $i = n$. The recursion at level i is performed to level $(i+1)$ only if the multiplication of constructed i-round trail probability and best found trail probability for $(n-i)$ rounds is at least $\bar{B}_n$, i.e., $p_1 p_2 \cdots p_i \widehat{B}_{n-i} \geq \bar{B}_n$ or $p_i \geq \bar{B}_n / (p_1 p_2 \cdots p_{i-1} \widehat{B}_{n-i})$. For $i = n$ the last equation is equivalent to: $p_1 p_2 \cdots p_n = \widehat{B}_n \geq \bar{B}_n$. If the latter holds, the initial estimate is updated: $\bar{B}_n \leftarrow \widehat{B}_n$ and the corresponding trail is also updated accordingly: $\bar{T}_n \leftarrow \widehat{T}_n$.

As a result of our investigation, the sorted pDDT proposed above makes more contribution in table C than table H as the entries with certain fixed input difference in H are within a small amount or even none, thus it cuts off rare branches.

4.2 Evaluation of the Proposed Algorithm

Both original algorithm and Algorithm 1 will finish when the initial estimate $\bar{B}_n$ can't be improved any more. We found that the probability order of entries in C also have great influence on time complexity. If the entries with greater probability are more likely to update $\bar{B}_n$ in the last round, the updated p_{min} may reduce the explore of some remaining entries with smaller probability.

Table 1: Search time of the original and the proposed algorithm. Environment: Intel(R) Core(TM) i5-3470 CPU, 3.20GHz, 8GB RAM

Rounds	Time per round (original)	Time accumulation (original)	Time per round (Algorithm 1)	Time accumulation (Algorithm 1)
1	0.001 s	0.001 s	0.002 s	0.002 s
2	0.006 s	0.007 s	0.004 s	0.006 s
3	1.901 s	1.908 s	1.860 s	1.866 s
4	1 m 2 s	1 m 3 s	48.010 s	49.876 s
5	3 m 37 s	4 m 40 s	25.553 s	1 m 15 s
6	10 m 30 s	15 m 10 s	6 m 38 s	7 m 53 s
7	13 m 26 s	28 m 35 s	8 m 22 s	16 m 16 s
8	23 m 12 s	51 m 47 s	13 m 0 s	29 m 16 s
9	25 m 23 s	1 h 17 m 10 s	12 m 46 s	42 m 2 s
10	19 m 8 s	1 h 36 m 18 s	6 m 38 s	48 m 40 s
11	8 m 17 s	1 h 44 m 35 s	2 m 35 s	51 m 15 s
12	19 m 11 s	2 h 3 m 45 s	6 m 47 s	58 m 2 s
13	19 m 23 s	2 h 23 m 8 s	6 m 59 s	1 h 5 m 0 s
14	22 m 1 s	2 h 45 m 10 s	4 m 3 s	1 h 9 m 4 s
15	10 m 29 s	2 h 55 m 38 s	1 m 48 s	1 h 10 m 52 s
16	33 m 0 s	3 h 28 m 38 s	5 m 27 s	1 h 16 m 19 s

The experiment result of the proposed algorithm is presented in the next section.

5 Experiments of the Improved Threshold Search

We applied Algorithm 1 to TEA with additive difference. In our experiment, we ignore the influence of the round constants, the round keys dependence just for a comparison with the original algorithm.

The comparison results are presented in Table 1. Column (Time per round) lists the search time for the best found trail of n rounds, given the probabilities $\widehat{B}_1$, $\widehat{B}_2$, $\cdots$, $\widehat{B}_{n-1}$ of best found trails for the first 1, 2, $\cdots$, $n-1$ rounds. Column (Time accumulation) is the time summation for n rounds, if $\widehat{B}_1$, $\widehat{B}_1$, $\cdots$, $\widehat{B}_{n-1}$ not known. From the table it shows that the proposed algorithm improves the search performance effectively and speed up in an approximate range from 1 to 8 times without changing the best found trails. It is difficult to estimate and quantize the elimination rates of unnecessary branches during the search phase, however, we state that the improvement will be more effective for the ARX block cipher with more rounds.

6 Conclusions

In this paper, we proposed a improved version of automatic search algorithm with a sorted pDDT concept to ARX block cipher. By using sorted pDDT concept, which has input differences in ascending order, probabilities in descending order and output differences in ascending order in sequence, we can reduce the exploring amount of temporary table C to eliminate most of search branches. We verify the improvement experimentally on the block cipher TEA, and give the search results and timing costs compared with the original Biryukov's search algorithm. The result shows, using the our improved algorithm with sorted pDDT concept, search performance has remarkably improvement. The further improvement and analysis on the more types of ARX block cipher will be our further research direction.

Acknowledgments

The research work was supported by New Century Excellent Talents in University of Ministry of Education under Grant NCET-12-0358, Technology Innovation Research Program of Shanghai Municipal Education Commission under Grant 12ZZ019 and Supporting Program of the "Twelfth Five-year Plan" for Sci and Tech Research of China under Grant 2014BAK06B02. Zheng Gong is also supported by the National Natural Sciences Foundation of China under Grant No. 61572028, the Natural Science Foundation of Guangdong (No.2014A030313439), the Foundation for Distinguished Young Teachers in Higher Education of Guangdong under Grant No. Yq2013051, and the Project of Science and Technology New Star of Guangzhou Pearl River (2014J2200006).

References

[1] R. Beaulieu, D. Shors, J. Smith, S. Treatman-Clark, B. Weeks, and L. Wingers, "The simon and speck families of lightweight block ciphers," *IACR Cryptology ePrint Archive*, vol. 2013, pp. 404, 2013.

[2] A. Biryukov and V. Velichkov, "Automatic search for differential trails in arx ciphers," in *Topics in Cryptology (CT-RSA'14)*, pp. 227–250, Springer, 2014.

[3] D. Khovratovich, I. Nikolic, J. Pieprzyk, P. Sokolowski, and R. Steinfeld, "Rotational cryptanalysis of arx revisited," *IACR Cryptology ePrint Archive*, vol. 2015, pp. 95, 2015.

[4] H. Lipmaa and S. Moriai, "Efficient algorithms for computing differential properties of addition," in *Fast Software Encryption*, pp. 336–350, Springer, 2002.

[5] H. Lipmaa, J. Wallén, and P. Dumas, "On the additive differential probability of exclusive-or," in *Fast Software Encryption*, pp. 317–331, Springer, 2004.

[6] M. Matsui, "On correlation between the order of S-boxes and the strength of DES," in *Advances in Cryptology (EUROCRYPT'94)*, pp. 366–375, Springer, 1995.

[7] N. Mouha, V. Velichkov, C. De Cannière, and B. Preneel, "The differential analysis of s-functions," in *Selected Areas in Cryptography*, pp. 36–56, Springer, 2011.

[8] R. M. Needham and D. J. Wheeler, *TEA Extensions*, Oct. 1997. (`http://www.cix.co.uk/~klockstone/xtea.pdf`)

[9] J. Polimón, J. C. Hernández-Castro, J. M. Estévez-Tapiador, and A. Ribagorda, "Automated design of a lightweight block cipher with genetic programming," *KES Journal*, vol. 12, no. 1, pp. 3–14, 2008.

[10] V. Velichkov, N. Mouha, C. De Canniere, and B. Preneel, "The additive differential probability of arx," in *Fast Software Encryption*, pp. 342–358, Springer, 2011.

[11] L. Wen and M. Wang, "Integral zero-correlation distinguisher for ARX block cipher, with application to shacal-2," in *Information Security and Privacy*, pp. 454–461, Springer, 2014.

[12] D. J. Wheeler and R. M. Needham, "TEA, a tiny encryption algorithm," in *Fast Software Encryption*, pp. 363–366, Springer, 1995.

Kaihao Chen M.E., works in Lab of Cryptography and Computer Forensics, School of Information Security Engineering, Shanghai Jiao Tong University. He received B.E. in 2013 from Shanghai Jiao Tong University. His recent research directions are high performance parallel computing on GPU and analysis of block ciphers.

Xiaoming Tang Ph.D. candidate, School of Information Security Engineering Shanghai Jiao Tong University. His recent research directions are cryptography, including analysis of block ciphers and hash functions.

Peng Xu received the M.E. degree from the School of Information Science and Engineering, Shandong University, in 2008. His main research areas include prison informatization, information security, Internet of things and he has published more than 15 papers. Currently, he is the project leader of the Research and Demonstration of the Key Technology in Prison Intelligent Security System Construction of 2014 National Science and Technology Supporting Program, and is responsible for the study of the Research and Demonstration of the Key Technology in the Intelligent Analysis and Recognition System of Criminal Behavior. At the same time, he is appointed as the expert of the information construction of justice department and the member of the expert group of the Ministry of Science and Technology.

Man Guo got Master degree on Management Science and Engineering from Chinese Academy of Agricultural Science. Now she works as the Program Officer on Science & Technology Management at Torch High Technology Industry Development Center, Ministry of Science and Technology for more than 6 years. She also has 2-year work experiences as Program Officer in State Nuclear Power Technology Corporation. She has published more than ten academic papers on Mass entrepreneurship and innovation and S&T talent team construction.

Weidong Qiu received the M.S. degree in cryptography from Xidian University, Xi'an, China, in 1998, and Ph.D. degree in computer software theory from Shanghai Jiao Tong University, Shanghai, China, in 2001. Before he joined Shanghai Jiao Tong University in 2004, he was a Postdoctoral Fellow at Hagen FernUniversity, Hagen, Germany, from 2001 to 2003. He is currently a Professor in the School of Information Security and Engineering, Shanghai Jiao Tong University. He has published more than twenty academic papers on cryptology. His main research areas include cryptographic theory, technology of network security, and computer forensics.

Zheng Gong received Ph.D. in 2008 from Shanghai Jiao Tong University, China. From 2008 December to 2012 January, he was a postdoc in the DIES group of Twente University. Currently he is an associate professor of Computer Science at South China Normal University. His recent research directions are cryptography and provable security, including the design and analysis of block ciphers, hash functions and message authentication codes.