Security Analysis of the Whirlpool Hash Function in the Cloud of Things

Wei Li[1, 2, 3], Zhiyong Gao[1], Dawu Gu[4], Chenyu Ge[1], Linfeng Liao[1], Zhihong Zhou[5, 3], Ya Liu[6, 4], and Zhiqiang Liu[4]

[1]School of Computer Science and Technology, Donghua University
Shanghai, 201620, China
[2]State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences
Beijing, 100093, China
[3]Shanghai Key Laboratory of Integrate Administration Technologies for Information Security
Shanghai, 200240, China
[4]Department of Computer Science and Engineering, Shanghai Jiao Tong University
Shanghai, 200240, China
[5]College of Information Security, Shanghai Jiao Tong University
Shanghai, 200240, China
[6]Department of Computer Science and Engineering, University of Shanghai for Science and Technology
Shanghai, 200093, China
[e-mail: *zhouzhihong@sjtu.edu.cn*]
*Corresponding author: Zhihong Zhou

Received July 24, 2016; revised November 8, 2016; accepted November 15, 2016; published January 31, 2017

Abstract

With the advancement and deployment of leading-edge telecommunication technologies for sensing and collecting, computing related information, Cloud of Things (CoTs) has emerged as a typical application platform that is envisioned to revolutionize the daily activities of human society, such as intelligent transportation, modern logistics, food safety, environmental monitoring, etc. To avoid any possible malicious attack and resource abuse, employing hash functions is widely recognized as one of the most effective approaches for CoTs to achieve message integrity and data authentication. The Whirlpool hash function has served as part of the joint ISO/IEC 10118-3 International Standard by the International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC). In this paper, we propose an effective differential fault analysis on Whirlpool in the byte-oriented random fault model. The mathematical analysis and experimental results show that 8 random faults on average are required to obtain the current 512-bit message input of whirlpool and the secret

This work is supported by the National Natural Science Foundation of China under Grant No. 61472250, No. 61402288 and No. 61672347, Shanghai Natural Science Foundation under Grant No. 15ZR1400300, No. 16ZR1401100 and No. 15ZR1400900, Innovation Program of Shanghai Municipal Education Commission under Grant No. 14ZZ066, the Plan of Action for the Innovation of Science and Technology of Shanghai Municipal Science and Technology Commission under Grant No. 14511100300, Shanghai Engineering Research Center Project under Grant No. GCZX14014 and No. C14001, National Key Basic Research Program of China under Grant No. 2013CB338004, the open research fund of State Key Laboratory of Information Security, the Opening Project of Shanghai Key Laboratory of Integrate Administration Technologies for Information Security, and the Fundamental Research Funds for the Central Universities.

https://doi.org/10.3837/tiis.2017.01.028 **ISSN : 1976-7277**

key of HMAC-Whirlpool. Our work demonstrates that Whirlpool and HMAC-Whirlpool are both vulnerable to the single byte differential fault analysis. It provides a new reference for the security analysis of the same structure of the hash functions in the CoTs.

Keywords: Cloud of Things, Hash Function, Fault Analysis, Whirlpool

1. Introduction

With the adoption and integration of Internet of Things and cloud computing, Cloud of Things (CoTs) provides inherently us a perfect platform and service to allow intelligent usage of collection of applications, information and infrastructure in intelligent transportation, modern logistics, food safety, environmental monitoring, etc [1, 2]. However, security and privacy issues in CoTs would be much more challenging than what has been used in the conventional scenarios, especially for the cloud side. Large amounts of important personal information are stored in the cloud servers, which are under threat of various attacks like monitoring and eavesdropping, etc [3]. These threats demand assurance for message authentication and data integrity for which variety of cryptographic solutions can be sought. It poses a real challenge for cryptographic community and information security specialists to define a secure environment fulfilling the above mentioned security requirements.

Hash functions have been widely used in a variety of security applications in CoTs, such as digital signature, files transfer and authentication schemes, etc. As a popular hash function with the Merkle-Damgård structure, Whirlpool is proposed by Barreto and Rijmen in 2000 [4]. It has been adopted by the International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC) as part of the joint ISO/IEC 10118-3 International Standard, and implemented in many cryptographic software libraries. The Whirlpool hash fucntion processes variable-length input messages and generates the 512-bit output as the hash value. Due to the significance of this standard, its security has been evaluated and approved by the New European Schemes for Signatures, Integrity and Encryption (NESSIE) project [5]. Many researchers have investigated the classical cryptanalysis on Whirlpool, such as the collision attack, the second preimage attack, the preimage attack, etc. Mendel et al. published a collision attack on the 4-round Whirlpool hash function in 2009 [6]. And Lamberger et al. extended the collision attack to 5 rounds [7]. Then Sasaki proposed a (second) preimage attack on the 5-round Whirlpool hash function in 2011 [8], and the complexity of his attack was improved by Wu et al. in 2012 [9]. Later Sasaki et al. extended the preimage attack to 6 rounds [10]. Iwamoto presented a limited birthday distinguisher on 7-round whirlpool in 2013 [11]. And Ma et al. improved extended the attack to 7.5 and 9 rounds [12]. Furthermore, Guo et al. proposed a universal forgery and key recovery attacks on HMAC based on round-reduced Whirlpool, and a distinguishing-H attack on HMAC based on full Whirlpool [13, 14].

Different from the above classical cryptanalysis, fault analysis is one important type of side-channel attacks on cryptographic devices. The attacker could inject faults into these devices to derive the secret keys of cryptosystems. In 1996, Boneh et al. proposed the fault analysis on the RSA cryptosystem by exploiting the faulty calculations [15, 16]. Then the differential fault analysis (DFA) was first presented to break the DES cryptosystem in 1997

[17]. The DFA attack exploits accessible information like input-output behavior under malfunctions, amplifies and evaluates the leaked information with the help of statistical methods [18-23]. Since the security and privacy issues of cryptographic devices in the real cloud stroage would be much challenging, it is very critical to evaluate the security of the cryptosystems against the DFA attack.

To the best of our knowledge, little research has been devoted to the security of Whirlpool against the DFA attack. Although Whirlpool adopts the Miyaguchi-Preneel mode built from an AES-like block cipher, it is more difficult to do DFA attack on Whirlpool than the available block ciphers [24-28]. As for the DFA on the available block ciphers, the last subkey should be recovered at first by differential analysis. Then the attacker could decrypt the right ciphertext to obtain the input of the last round, which is the output of the penultimate round. They repeat the above procedure to recover more subkeys until the secret key is obtained by the key schedule. However, Whirlpool's Miyaguchi-Preneel mode combines the previous chaining value, the last message input and the output of the last round to get the hash value by the XOR operation. The XOR operation hinders completely deducing the relationship between the last subkey and the hash output difference. Furthermore, it has a 512-bit long key size and a complex round function. The diffusion layer propagates one single-byte fault to 8 bytes' difference in substitution layer of the last round and the attacking complexity is up to 2^{64}. It is not really practical to compute in the real scenario.

We thus propose a practical and effective DFA method to derive the message of Whirlpool, with application to breaking HMAC-Whirlpool. In the byte-oriented fault model, the attacker could induce a single byte error to the layer in the hash function. The location of this byte in this layer and the value of the error are both unknown. On this reasonable assumption, the DFA method could recover all subkeys and the original message of Whirlpool. The experiment shows that only 8 errors on average can be used to recover one 512-bit block of message input of Whirlpool and secret key of HMAC-Whirlpool. This is the first work that a differential fault attack on Whirlpool and HMAC-Whirlpool has been successfully put into practice. Compared with the classical cryptanalysis, the DFA on Whirlpool and HMAC-Whirlpool has a good performance in time complexity and memory complexity, as **Table 1** shows.

Table 1. The previous best results and our results on Whirlpool and HMAC-Whirlpool.

Method	# Rounds	Complexity		Reference
		Time	Memory	
Collision attack	5.5	2^{120}	2^{64}	[7]
Preimage attack	6	2^{481}	2^{256}	[10]
Limited-birthday distinguisher	9	2^{354}	2^{158}	[12]
Key recovery	7	$2^{482.3}$	2^{481}	[14]
Differential fault analysis	10	2^{33}	2^{15}	This paper

The rest of this paper is organized as follows. Section 2 briefly introduces the Whirlpool hash function. The next two sections propose our differential fault analysis to break Whirlpool and HMAC-Whirlpool. Section 5 summarizes the attacking complexity and section 6 shows the attacking experimental results. Finally the last section concludes the paper.

2. Description of Whirlpool

2.1 Structure

The Whirlpool hash function processes 512-bit message blocks and produces a 512-bit hash value as **Fig. 1** shows. An unambiguous padding method is applied to ensure that the message length is a multiple of 512 bits. Let $M = M^1 \| M^2 \| \cdots \| M^t$ be a t-block message after padding. The hash value $h = H(M)$ is computed as follows:

$$\begin{aligned} H^0 &= IV, \\ H^x &= W(H^{x-1}, M^x) \oplus H^{x-1} \oplus M^x, \\ h &= H^t, \end{aligned}$$

where IV is a predefined initial value, W is a 512-bit block cipher used in the Miyaguchi-Preneel mode, and $1 \le x \le t$.

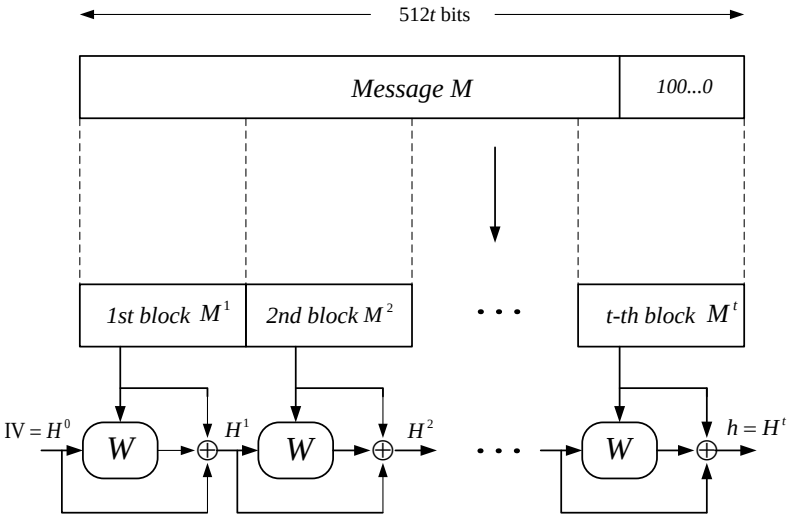


Fig. 1. Message digest generation using the Whirlpool hash function.

The W block cipher is designed on the basis of the wide trail strategy. Its block size and the key size are 512 bits, respectively. In its 10-round transformations, each round operates on a state of 8×8 bytes and updates the state by means of the sequence of the following layers:

$$ARK \circ MR \circ SC \circ SB.$$

The layers in each transformation are briefly described here:

- SubBytes(SB), where the only non-linear transformation layer applies an S-box to each byte of the state independently.
- ShiftColumns(SC), where the cyclical permutation layer rotates the byte of column o downwards by o-1 positions, with $1 \le 0 \le 8$.
- MixRows(MR), where the linear diffusion layer is a right-multiplication by an 8×8 row-oriented matrix, denoted by R.
- AddRoundKey(ARK), where the key addition layer adds the round key to the state.

The key schedule expands the 512-bit cipher key K into a sequence of round keys $K_{10}^x, \cdots, K_{10}^x$, with $1 \le x \le t$. It adopts the similar layers as follows:

$$AC \circ MR \circ SC \circ SB,$$

where the AddConstant(AC) adds the round constant to the state of the key schedule, and the other layers are used with $K_0^x = H^{x-1}$ with $1 \le x \le t$.

After the last round of the state update transformation, the previous chaining value H^{t-1} , the message block M^t , and the output value of the last round are combined by the XOR operation, resulting in the output of one iteration.

2.2 Notations

The notations of Whirlpool and its analysis are described as follows:

Table 2. Notations of Whirlpool.

Notations	Description
$A_i^x, B_i^x, C_i^x, D_i^x$	The right inputs of SubBytes, ShiftColumns, MixRows and AddRoundKey in the *i*-th round of the *x*-th 512-bit message with $1 \le x \le t$ and $1 \le i \le 10$
$\overline{A_i^x}, \overline{B_i^x}, \overline{C_i^x}, \overline{D_i^x}$	The faulty inputs of SubBytes, ShiftColumns, MixRows and AddRoundKey in the *i*-th round of the *x*-th 512-bit message with $1 \le x \le t$ and $1 \le i \le 10$
$\Delta A_i^x, \Delta B_i^x, \Delta C_i^x, \Delta D_i^x$	The input differences of SubBytes, ShiftColumns, MixRows, and AddRoundKey in the *i*-th round of the *x*-th 512-bit message with $1 \le x \le t$ and $1 \le i \le 10$
$SB^{-1}, SC^{-1}, MR^{-1}$	The inverse operations of SubBytes, ShiftColumns and MixRows
$a_{i,j}^x, b_{i,j}^x, d_{i,j}^x$	The *j*-th byte of A_i^x, B_i^x, D_i^x with $1 \le x \le t$, $1 \le i \le 10$ and $1 \le j \le 64$
$\Delta a_{i,j}^x, \Delta b_{i,j}^x, \Delta d_{i,j}^x$	The *j*-th byte difference of A_i^x, B_i^x, D_i^x with $1 \le x \le t$, $1 \le i \le 10$ and $1 \le j \le 64$

3. Differential Fault Analysis on Whirlpool

3.1 Basic assumption

The differential fault analysis exploits the difference between a normal and a faulty hash output after processing the same input. Our proposed fault model includes the following two assumptions:

- The attacker has the capability to choose one message to process and obtain the corresponding right and faulty hash values.
- The attacker could induce a single byte error to one transformation. However, both the location and the value of this byte in this round are unknown.

3.2 The propose attack

In this subsection, we apply the above basic idea and propose a novel differential fault analysis to recover the input message of Whirlpool. Our analysis is split into the following four successive phases for Whirlpool.

3.2.1 Phase 1: recovering $K_{10}^t \oplus H^{t-1} \oplus M^t$

The first phase of the attack aims at recovering the related value of K_{10}^t, that is, $K_{10}^t \oplus H^{t-1} \oplus M^t$ in the 10th round of the *t*-th block of message. The fault injection targets at the 8th round in this block as **Fig. 2** shows. The detailed location may be induced on either A_8^t, B_8^t, C_8^t whereas the approach is identical in either case. Note that any modification of one byte provokes the XOR-differences of ΔA_8^t on A_8^t, ΔB_8^t on B_8^t, ΔC_8^t on C_8^t, $\Delta \mathrm{D}_8^t$ on $\mathrm{D}_8^t, \cdots,$ ΔA_{10}^t on A_{10}^t, ΔB_{10}^t on B_{10}^t, ΔC_{10}^t on C_{10}^t, $\Delta \mathrm{D}_{10}^t$ on D_{10}^t. These alter the original right hash value H^t into the faulty hash value $\overline{H^t}$. We can observe that

$$\Delta D_{10}^t = (H^t \oplus M^t \oplus H^{t-1} \oplus K_{10}^t) \oplus (\overline{H^t} \oplus \overline{M^t} \oplus \overline{H^{t-1}} \oplus \overline{K_{10}^t}) = \Delta H^t,$$
$$\Delta C_{10}^t = MR^{-1}(\Delta D_{10}^t) = MR^{-1}(\Delta H^t),$$
$$\Delta B_{10}^t = SC^{-1}(\Delta C_{10}^t) = SC^{-1}(MR^{-1}(\Delta H^t))$$

holds, since $M^t = \overline{M^t}, H^{t-1} = \overline{H^{t-1}}, K^t = \overline{K_{10}^t}$, i.e. M^t, H^{t-1} and K_{10}^t are not affected by the faults.

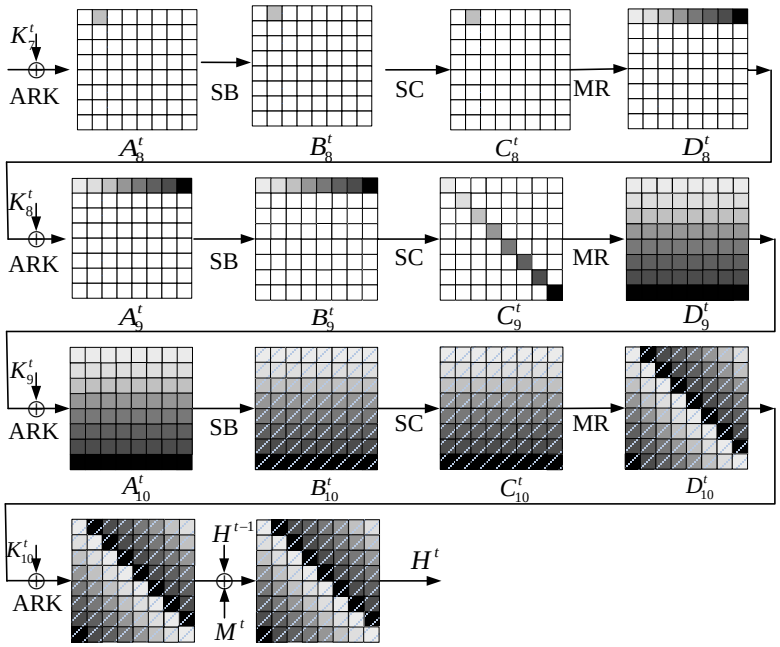


Fig. 2. One single-byte fault propagation path in Whirlpool.

The above equations, in conjunction with a pair of right and faulty hash values, allow to infer a relation between ΔA_{10}^t and ΔB_{10}^t. It is helpful to restrict a list of possible candidates for the value of $K_{10}^t \oplus H^{t-1} \oplus M^t$. The differential transformation of S-boxes in the last round of the message is

$$SS(\Delta a_{10,j}^t, \Delta b_{10,j}^t) = \left\{ a_{10,j}^t \middle| a_{10,j}^t \in \{0,1\}^8, S(a_{10,j}^t) \oplus (a_{10,j}^t \oplus \Delta a_{10,j}^t) = \Delta b_{10,j}^t \right\},$$

where $1 \le j \le 64$.

Thus, the output difference of the j-th S-box in the 10th round can be represented by

$$\Delta b_{10,j}^t = (\Delta B_{10}^t)_j = (SC^{-1}(MR^{-1}(\Delta H^t)))_j,$$

and the input difference of the j-th S-box in the 10th round is

$$\Delta a_{10,j}^t = (\Delta A_{10}^t)_j,$$

where $1 \le j \le 64$.

Given the pair of hash values $(H^t, \overline{H^t})$, the j-th byte of $K_{10}^t \oplus H^{t-1} \oplus M^t$ needs to satisfy

$$a_{10,j}^t \in SS(\triangle a_{10,j}^t, \triangle b_{10,j}^t),$$

where $1 \le j \le 64$. Therefore,

$$(SB^{-1}(SC^{-1}(MR^{-1}(H^t \oplus K_{10}^t \oplus H_{t-1} \oplus M^t))))_j \in SS(\Delta a_{10,j}^t, \Delta b_{10,j}^t),$$

where $1 \le j \le 64$.

That is,

$$(K_{10}^t \oplus H^{t-1} \oplus M^t) \in (H^t \oplus MR(SC(SB(SS(\Delta a_{10,1}^t, \Delta b_{10,1}^t), SS(\Delta a_{10,2}^t, \Delta b_{10,2}^t), \cdots, SS(\Delta a_{10,64}^t, \Delta b_{10,64}^t))))).$$

This procedure leads to a list of candidates $K_{10}^t \oplus H^{t-1} \oplus M^t$ which are compatible under the observed fault injections. We could apply the above procedure to multiple pairs of right and faulty hash values obtained from fault injections independently. Then the attacker could progressively further restrict this list of possible candidates to derive more values, since the correct candidate necessarily appears in the intersection of lists deduced from each pair of hash values. it is hence possible to identify the value of $K_{10}^t \oplus H^{t-1} \oplus M^t$ with only a few faults.

3.2.2 Phase 2: recovering K_9^t

The second phase of the attack aims at recovering K_9^t in the penultimate round. A fault may be induced on A_7^t, B_7^t or C_7^t in the 7th round, leading to the XOR-differences ΔA_7^t on A_7^t, ΔB_7^t on B_7^t, ΔC_7^t on C_7^t, ΔD_7^t on D_7^t, $\cdots$, ΔA_{10}^t on A_{10}^t, ΔB_{10}^t on B_{10}^t, ΔC_{10}^t on C_{10}^t and ΔD_{10}^t on D_{10}^t. We could compute the values in the t-th block of message as follows:

$$\begin{aligned}
\Delta D_9 &= (A_{10}^t \oplus K_{10}) \oplus (\overline{A_{10}^t} \oplus \overline{K_{10}}) = \Delta A_{10}^t, \\
\Delta C_9 &= MR^{-1}(\Delta D_9) = MR^{-1}(\Delta A_{10}^t), \\
\Delta B_9 &= SC^{-1}(\Delta D_9) = SC^{-1}(MR^{-1}(\Delta A_{10}^t)),
\end{aligned}$$

where

$$\Delta A_{10}^t = SB^{-1}(SC^{-1}(MR^{-1}(H^t \oplus K_{10}^t \oplus H^{t-1} \oplus M^t) \oplus SB^{-1}(SC^{-1}(MR^{-1}(\overline{H^t} \oplus K_{10}^t \oplus H^{t-1} \oplus M^t))).$$

Here, the value of $K_{10}^t \oplus H^{t-1} \oplus M^t$ has been derived in phase 1.

Given the above values, the output difference of the j-th S-box in the 9th round can be represented by

$$\Delta b_{9,j}^t = (\Delta B_9^t)_j = SC^{-1}(MR^{-1}(\Delta A_{10}^t))_j,$$

and the input difference of the j-th S-box in the 9th round is

$$\Delta a_{9,j}^t = (\Delta A_9^t)_j,$$

where $1 \le j \le 64$.

Thus, the j-th byte of $K_9^t \oplus K_{10}^t \oplus H_{t-1} \oplus M^t$ needs to satisfy

$$a_{9,j}^t \in SS(\Delta a_{9,j}^t, \Delta b_{9,j}^t),$$

where $1 \le j \le 64$ .Therefore,

$$(SB^{-1}(SC^{-1}(MR^{-1}(A_{10}^t \oplus K_9^t))))_j \in SS(\Delta a_{9,j}^t, \Delta b_{9,j}^t),$$

where $1 \le j \le 64$.

That is,

$$K_9^t \in SB^{-1}(SC^{-1}(MR^{-1}(H_t \oplus K_{10}^t \oplus H_{t-1} \oplus M^t))) \oplus MR(SC(SB(SS(\Delta a_{9,1}^t, \Delta b_{9,1}^t), SS(\Delta a_{9,2}^t, \Delta b_{9,2}^t), \cdots, SS(\Delta a_{9,64}^t, \Delta b_{9,64}^t))))),$$

where $1 \le j \le 64$.

This procedure leads to a list of candidates K_9^t which are compatible under the observed fault injections. The attacker applies the above procedure to multiple pairs of right and faulty hash values obtained from independently performed fault injections. Then the list of remaining candidates for K_9^t reduces rapidly when building the intersection. Finally, the correct value of K_9^t is retrieved.

3.2.3 Phase 3: recovering M^t

The third phase of the attack aims at deriving M^t by recovering two values K_{10}^t and H^{t-1}. We make advantage of $K_{10}^t \oplus H^{t-1} \oplus M^t$ and K_9^t derived in the previous two phases. On the basis of the key schedule, we could derive the following values:

$$\begin{aligned} K_{10}^t &= AC(MR(SC(SB(K_9^t)))), \\ H^{t-1} &= K_0^t \\ &= AC^{-1}(MR^{-1}(SC^{-1}(SB^{-1}(K_1^t)))) \\ &= AC^{-1}(MR^{-1}(SC^{-1}(SB^{-1}(AC^{-1}(MR^{-1}(SC^{-1}(SB^{-1}(K_2^t)))))))) \\ &= AC^{-1}(MR^{-1}(SC^{-1}(SB^{-1}(L(AC^{-1}(MR^{-1}(SC^{-1}(SB^{-1}(K_9^t)))))))))). \end{aligned}$$

The values of $K_{10}^t \oplus H^{t-1} \oplus M^t$, H^{t-1} and K_{10}^t are derived in the previous three phases, so the values of M^t could be obtained by the XOR operation.

3.2.4 Phase 4: recovering M

The last phase of the attack aims at recovering t values $M^{t-1}, \cdots, M^1$ and M . The previous three phases are available to derive the above values. The attacker could induce faults into the 8th and 7th rounds of the t-1-th blocks of message, and then take the similar attacking procedure to derive the values of M^{t-1}. By the similar attacking phases, the other blocks of the input input message $M^{t-2}, \cdots, M^1$ could be obtained. Therefore, the value of the message is computed as follows:

$$M = M^1 \| M^2 \| \cdots \| M^t .$$

3.3 Deriving the Input of the SubBytes

In the structure of Whirlpool, the MixRows layer propagates one single byte fault to eight-byte differences in the input of the SubBytes transformation. If we do brute force search

for the input of the SubBytes transformation, the complexity to recover one subkey is up to 2^{64}. This kind of brute force search is not really practical.

We propose an effective approach to select the input difference of the SubBytes transformation, so the input of the SubBytes transformation could be obtained with less complexity. Take the derivation of A_{10}^t as an example. One single byte error can lead to eight-byte differences independently after the computation of the diffusion layer in ΔA_9^t, and the differences could result in the 512-bit differences in ΔA_{10}^t. This important property helps to do brute force search on A_{10}^t. This approach simplifies the analysis and improves the attacking efficiency. On the basis of the byte-oriented fault model, there are eight types of ΔA_{10}^t in **Fig. 3**. There are proportional relationships among the values in different columns of the same row. On the basis of the relationships, we could compute the value of A_{10}^t.

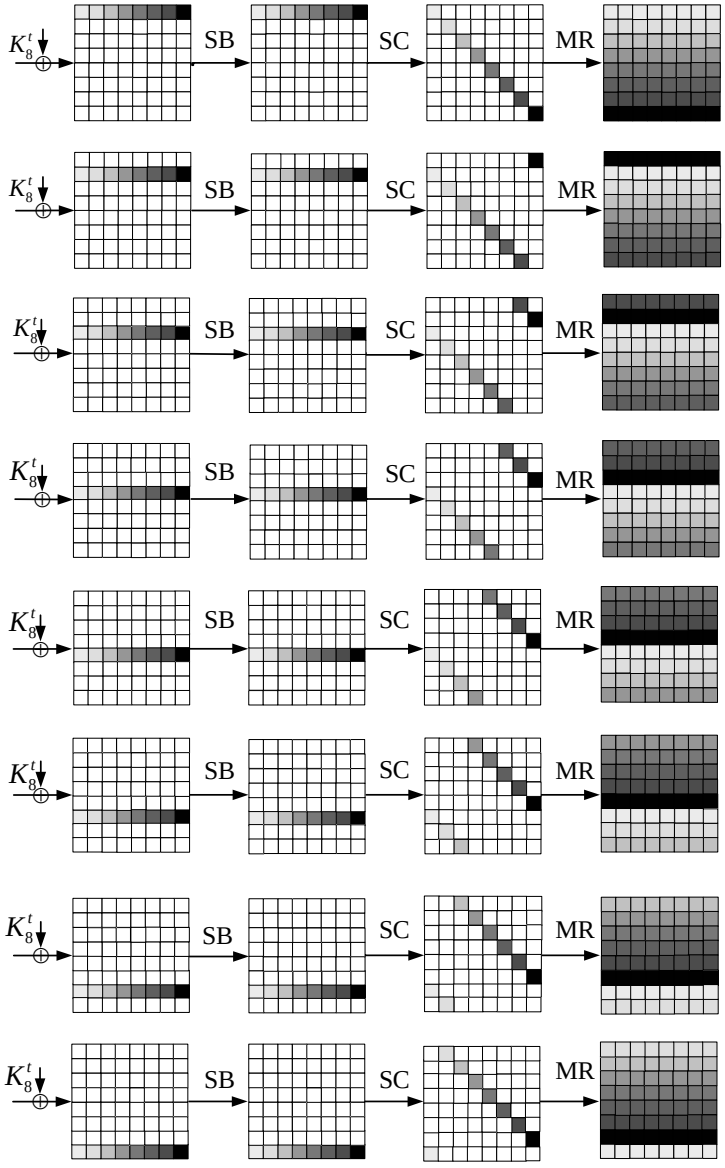


Fig. 3. The eight relationships among the fault propagation values.

The equations are derived as follows:

$$
\begin{aligned}
&(\Delta d^t_{10,1}, \Delta d^t_{10,10}, \Delta d^t_{10,19}, \Delta d^t_{10,28}, \Delta d^t_{10,37}, \Delta d^t_{10,46}, \Delta d^t_{10,55}, \Delta d^t_{10,64}) \\
&= (S(a^t_{10,1}) \oplus S(a^t_{10,1} \oplus \Delta a^t_{10,1}), S(a^t_{10,2}) \oplus S(a^t_{10,2} \oplus \Delta a^t_{10,2}), \cdots, S(a^t_{10,8}) \oplus S(a^t_{10,8} \oplus \Delta a^t_{10,8})) \cdot R, \\
&(\Delta d^t_{10,2}, \Delta d^t_{10,11}, \Delta d^t_{10,20}, \Delta d^t_{10,29}, \Delta d^t_{10,38}, \Delta d^t_{10,47}, \Delta d^t_{10,56}, \Delta d^t_{10,57}) \\
&= (S(a^t_{10,57}) \oplus S(a^t_{10,57} \oplus \Delta a^t_{10,57}), S(a^t_{10,58}) \oplus S(a^t_{10,58} \oplus \Delta a^t_{10,58}), \cdots, S(a^t_{10,64}) \oplus S(a^t_{10,64} \oplus \Delta a^t_{10,64})) \cdot R, \\
&(\Delta d^t_{10,3}, \Delta d^t_{10,12}, \Delta d^t_{10,21}, \Delta d^t_{10,30}, \Delta d^t_{10,39}, \Delta d^t_{10,48}, \Delta d^t_{10,49}, \Delta d^t_{10,58}) \\
&= (S(a^t_{10,49}) \oplus S(a^t_{10,49} \oplus \Delta a^t_{10,49}), S(a^t_{10,50}) \oplus S(a^t_{10,50} \oplus \Delta a^t_{10,50}), \cdots, S(a^t_{10,56}) \oplus S(a^t_{10,56} \oplus \Delta a^t_{10,56})) \cdot R, \\
&(\Delta d^t_{10,4}, \Delta d^t_{10,13}, \Delta d^t_{10,22}, \Delta d^t_{10,31}, \Delta d^t_{10,40}, \Delta d^t_{10,41}, \Delta d^t_{10,50}, \Delta d^t_{10,59}) \\
&= (S(a^t_{10,41}) \oplus S(a^t_{10,41} \oplus \Delta a^t_{10,41}), S(a^t_{10,42}) \oplus S(a^t_{10,42} \oplus \Delta a^t_{10,42}), \cdots, S(a^t_{10,48}) \oplus S(a^t_{10,48} \oplus \Delta a^t_{10,48})) \cdot R, \\
&(\Delta d^t_{10,5}, \Delta d^t_{10,14}, \Delta d^t_{10,23}, \Delta d^t_{10,32}, \Delta d^t_{10,33}, \Delta d^t_{10,42}, \Delta d^t_{10,51}, \Delta d^t_{10,60}) \\
&= (S(a^t_{10,33}) \oplus S(a^t_{10,33} \oplus_{\Delta} a^t_{10,33}), S(a^t_{10,34}) \oplus S(a^t_{10,34} \oplus \Delta a^t_{10,34}), \cdots, S(a^t_{10,40}) \oplus S(a^t_{10,40} \oplus \Delta a^t_{10,40})) \cdot R, \\
&(\Delta d^t_{10,6}, \Delta d^t_{10,15}, \Delta d^t_{10,24}, \Delta d^t_{10,25}, \Delta d^t_{10,34}, \Delta d^t_{10,43}, \Delta d^t_{10,52}, \Delta d^t_{10,61}) \\
&= (S(a^t_{10,25}) \oplus S(a^t_{10,25} \oplus \Delta a^t_{10,25}), S(a^t_{10,26}) \oplus S(a^t_{10,26} \oplus \Delta a^t_{10,26}), \cdots, S(a^t_{10,32}) \oplus S(a^t_{10,32} \oplus \Delta a^t_{10,32})) \cdot R, \\
&(\Delta d^t_{10,7}, \Delta d^t_{10,16}, \Delta d^t_{10,17}, \Delta d^t_{10,26}, \Delta d^t_{10,35}, \Delta d^t_{10,44}, \Delta d^t_{10,53}, \Delta d^t_{10,62}) \\
&= (S(a^t_{10,17}) \oplus S(a^t_{10,17} \oplus \Delta a^t_{10,17}), S(a^t_{10,18}) \oplus S(a^t_{10,18} \oplus \Delta a^t_{10,18}), \cdots, S(a^t_{10,24}) \oplus S(a^t_{10,24} \oplus \Delta a^t_{10,24})) \cdot R, \\
&(\Delta d^t_{10,8}, \Delta d^t_{10,9}, \Delta d^t_{10,18}, \Delta d^t_{10,27}, \Delta d^t_{10,36}, \Delta d^t_{10,45}, \Delta d^t_{10,54}, \Delta d^t_{10,63}) \\
&= (S(a^t_{10,9}) \oplus S(a^t_{10,9} \oplus \Delta a^t_{10,9}), S(a^t_{10,10}) \oplus S(a^t_{10,10} \oplus \Delta a^t_{10,10}), \cdots, S(a^t_{10,16}) \oplus S(a^t_{10,16} \oplus \Delta a^t_{10,16})) \cdot R,
\end{aligned}
$$

where R is a row-oriented matrix in the MixRows layer.

When the random fault influences every row of A^t_{10}, there are eight types of values of ΔA^t_{10} as **Fig. 3** shows. Thus, the values of ΔA^t_{10} could be represented by

$$\Delta A^t_{10} = \Delta C^t_9 \cdot R.$$

The attacker could continue inducing random faults and repeating the above operations until A^t_{10} has only one value.

4. Differential Fault Analysis on HMAC-Whirlpool

The keyed-Hash Message Authentication Code, abbreviated as HMAC, is a hash-based message authentication code proposed by Bellare et al. in CRYPTO [29]. It has been standardized by ANSI, IETF, ISO and NIST, and widely deployed in SSL, TLS, SSH and IPsec, etc. The HMAC is computed with a cryptographic hash function, such as Whirlpool. A secret key is involved and only the parties who know the secret key could generate a correct HMAC. The cryptographic strength of the HMAC depends on the secret key and the cryptographic strength of the underlying hash function. Let H represent the Whirlpool hash function. Its HMAC is defined by

$$HMAC(K, M) = H(K \oplus opad \,||\, H(K \oplus ipad \,||\, M)),$$

where M represents the message, K denotes the secret key, $||$ represents concatenation, the inner padding *opad* and the outer padding *ipad* denote two one-block different public constants. The attacker could derive the value of $K \oplus opad \,||\, H(K \oplus ipad \,||\, M)$ by the DFA. Thus, he could derive the value of $K \oplus opad$, and further derive the value of K by XORing the public constant *opad*.

5. Attacking Complexity

We summarize the attacking procedure to select 512-bit subkey candidates for one block of input message. The time complexity of brute-force search for one fault injection is

$$u = 2^{2s} \cdot \left(\left\lceil \frac{n}{s} \right\rceil \right)^2,$$

where n denotes the size of the SubBytes layer and s denotes the input size of one S-box.

In addition, an estimation of the number of faults necessary for the attack to be successful is vital. In the attacking procedure, the number of faults to recover a subkey depends on the fault location and the fault model.

We take the derivation of $K_{10}^t \oplus H^{t-1} \oplus M^t$ in phase 1 as an example. On the definition of the SubBytes layer, if $K_{10}^t \oplus H^{t-1} \oplus M^t$ is a candidate, then there may be another subkey candidate. In other words, if the input candidates set of S-boxes is not null, then the input A_{10}^t may have several candidates. It indicates that $K_{10}^t \oplus H^{t-1} \oplus M^t$ may have some possible elements.

In the fault model, a random error could be induced at any round of the hash function. If the fault occurs in the last round, only one single byte in the input of the SubBytes layer will change, which could recover at most one byte of the last subkey by the analysis. To recover the last subkey, it is necessary to induce many errors into different bytes.

If the fault is induced at an ideal location before the last round, then the input difference and output difference of the SubBytes layer in this round contain only one nonzero byte. However, the output difference of the MixRows layer has multibytes owing to the diffusion of linear transformation. Thus, the input difference of the SubBytes layer in the last round contains multibytes after the computation of the last several rounds. The above idea is applied in the attacking procedure to improve the efficiency of fault injection.

Since at least two errors can make one element in the intersection of $K_{10}^t \oplus H^{t-1} \oplus M^t$, we continue deriving intersection of subkey candidates sets until the intersection has only one element. Thus, at least two faults are required to derive multibytes of one subkey. The theoretical minimum number of faults to recover one subkey is defined as

$$v = \begin{cases} 0 & if \quad q = 0 \\ \left\lceil \frac{2n}{q} \right\rceil & if \quad 1 \le q < n \end{cases},$$

where n represents the size of the SubBytes layer, and q represents the maximum number of bits in a subkey derived by two faults. To derive the subkey, the value of q equals the number of bits in the nonzero output difference of the nonlinear transformation in this round. If q=0, then there is no bits of a subkey derived and thus v=0.

Thus, the overall time complexity to recover one block of input message is

$$u \cdot v \cdot g = \begin{cases} 0 & if \quad q = 0 \\ 2^{2s+1} \cdot \left\lceil \frac{n^3 \cdot g}{s^2 \cdot q} \right\rceil & if \quad 1 \le q < n \end{cases},$$

and the memory complexity to recover one block of input message is

$$g \cdot 2^6 \cdot 2^8,$$

where g denotes the number of subkeys to recover a secret key, n denotes the size of the SubBytes layer, s denotes the input size of one S-box, and q represents the maximum number of bits in a subkey derived by two faults.

Thus, we could make use of ideal faults to attack Whirlpool of HAMC-Whirlpool. The time complexity and data complexity in theory is about 2^{32} and 2^{15} for a 512-bit message block and the secret key, where $g=2$, $n=512$, $s=8$, $q=512$, and $v=2$.

6. Experimental Results

We implemented the attack on a PC using the Java language with 32GB memory. The fault induction was simulated by computer software. In this situation, we ran the attack algorithm to 1000 process units. We define accuracy, reliability and latency for evaluating the experimental results in detail.

Accuracy is a measure that defines how close the number of subkey candidates are to the true number of subkey candidates. Basically, the closer the experimental number of subkey candidates is to the true number, the more accurate the experiment is. Thus, we consider the Root Mean-Square Error(RMSE) to measure the accuracy, where RMSE is given by

$$RMSE = \sqrt{\frac{1}{N}\sum_{e=1}^{N}[h_e - h]},$$

where N is the number of experiments in a set, e denotes the index of the experiment, h_e represents the number of subkey candidates, and h denotes the number of true subkeys. As we know, there is only one true subkey. The closer the RMSE value is to 0, the more accurate the experiments are. We divide 1000 experiments as five groups on average, denoted as G_1, G_2, G_3, G_4 and G_5 .The RMSE values for every intersections of subkey candidates are shown in **Fig. 4** and **Table 3**, where N=200, h=1 and $e \in \{1,\cdots,1000\}$. Thus, 4 faulty ciphertexts are required to recover one subkey. Furthermore, the accuracy in every group for the same interaction is similar or equal.

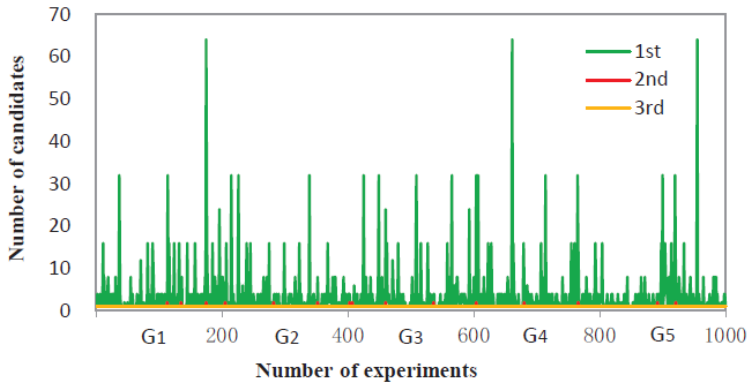


Fig. 4. Three intersections of subkey candidates in 1000 experiments.

Table 3. One subkey recovery on Accuracy by RMSE

Groups	2 faults	3 faults	4 faults
G_1	1.632	0.122	0
G_2	1.466	0.122	0
G_3	1.564	0.141	0
G_4	1.632	0.122	0
G_5	1.463	0.100	0

Reliability is the ratio of successful experiments out of all experiments made. If the attacker could derive only one subkey, the experiment is successful. Referring to **Table 4**, it is observed that the ratio of successful experiments by injecting 2, 3 and 4 faults are 52.2%, 98.5% and 100%, respectively. That is, the reliability is 100% if the attacker induces 4 random faults to break a subkey.

Table 4. One block of input message rececovery on Reliability

Groups	2 faults	3 faults	4 faults
G_1	50.5%	98.5%	100%
G_2	51.0%	98.5%	100%
G_3	55.5%	98.0%	100%
G_4	52.0%	98.5%	100%
G_5	52.0%	99.0%	100%

Latency is the time from the first fault injection to the recovery of the subkey in our software simulation. It is measured in minutes. **Fig. 5** shows that the latency of 1000 experiments. The time of 90.1% experiments is between 0.8 min and 1.6 min.

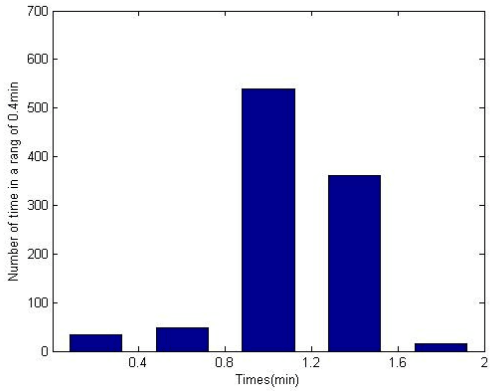


Fig. 5. One block of input message recovery on Latency.

In the byte-oriented fault model, we need about 8 faulty injections to retrieve the current block of 512-bit input message of Whirlpool and the secret key of HMAC-Whirlpool, respectively. The time complexity and data complexity in practice are 2^{33} and 2^{15} to break Whirlpool and HMAC-Whirlpool by the differential fault analysis.

6. Conclusion

This paper presents a DFA attack on Whirlpool and HMAC-Whirlpool in a byte-oriented fault model. The analysis could break the current message of Whirlpool and the secret key of HMAC-Whirlpool by only 8 faults on average. It shows that Whirlpool and HMAC-Whirlpool are both vulnerable to the single byte differential fault analysis. We expect that our research will provide deeper understanding of the security of other hash functions in the CoTs.

References

[1] M. Aazam, I. Khan, A. A. Alsaffar and E. Huh: "Cloud of Things: integrating Internet of Things and cloud computing and the issues involved," in *Proc. of Int. Bhurban Conf. on Applied Sciences and Technology*, pp. 414-419, January 14-18, 2014. Article (CrossRef Link).

[2] M. Aazam, E. Huh, M. St-Hilaire, C. Lung and I. Lambadaris: "Cloud of Things: integration of IoT with cloud computing," in *Proc. of Robots and Sensor Clouds*, vol. 36, pp. 77-94, August 18, 2016. Article (CrossRef Link).

[3] T. Bhattasali, R. Chaki and N. Chaki, "Secure and trusted Cloud of Things," in *Proc. of 2013 Annual IEEE India Conf.*, pp. 1-6, December 13-15, 2013. Article (CrossRef Link).

[4] P. Barreto and V. Rijmen: "The Whirlpool hashing function," in *Proc. of 1st open NESSIE Workshop*, pp. 543-553, November, 2000. Article (CrossRef Link).

[5] B. Preneel: "New European schemes for signature, integrity and encryption (NESSIE): a status report," in *Proc. of Int. Workshop on Practice and Theory in Public Key Cryptography*, pp. 297-309, February 12–14, 2002. Article (CrossRef Link).

[6] Y. Sasaki: "Meet-in-the-middle preimage attacks on AES hashing modes and an application to Whirlpool," in *Proc. of Int. Workshop on Fast Software Encryption*, pp. 378-396, February 13-16, 2011. Article (CrossRef Link).

[7] M. Lamberger, F. Mendel, C. Rechberger, V. Rijmen and M. Schlaffer: "Rebound distinguishers: results on the full Whirlpool compression function," in *Proc. of Int. Conf. on the Theory and Application of Cryptology and Information Security*, pp. 126-143, December 6-10, 2009. Article (CrossRef Link).

[8] F. Mendel, C. Rechberger, M. Schläffer and S. S. Thomsen: "The rebound attack: cryptanalysis of reduced Whirlpool and Grøstl," in *Proc. of Int. Conf. Fast Software Encryption*, pp. 260-276, February 22-25, 2009. Article (CrossRef Link).

[9] S. Wu, D. Feng, W. Wu, J. Guo, L. Dong and J. Zou: "(Pseudo) preimage attack on round-reduced Grøstl hash function and others," in *Proc. of Int. Conf. Fast Software Encryption*, pp. 127-145, March 19-21, 2012. Article (CrossRef Link).

[10] Y. Sasaki, L. Wang, S. Wu and W. Wu: "Investigating fundamental security requirements on Whirlpool: improved preimage and collision attacks," in *Proc. of Int. Conf. Theory and Application of Cryptology and Information Security*, pp. 562-579.December 2-6, 2012. Article (CrossRef Link).

[11] M. Iwamoto, T. Peyrin and Y. Sasaki: "Limited-birthday distinguishers for hash functions," in *Proc. of Int. Conf. Theory and Application of Cryptology and Information Security*, pp. 504-523, December 1-5, 2013. Article (CrossRef Link).

[12] M. Ma, B. Li, R. Hao and X. Li: "Improved cryptanalysis on reduced-round GOST and Whirlpool hash function," in *Proc. of Int. Conf. Applied Cryptography and Network Security*, pp. 289-307, June 10-13, 2014. Article (CrossRef Link).

[13] J. Guo, Y. Sasaki, L. Wang and S. Wu: "Cryptanalysis of HMAC/NMAC-Whirlpool," in *Proc. of Int. Conf. Theory and Application of Cryptology and Information Security*, pp. 21-40, December 1-5, 2013. Article (CrossRef Link).

[14] J. Guo, Y. Sasaki, L. Wang, M. Wu and L. Wen: "Equivalent key recovery attacks against HMAC and NMAC with Whirlpool reduced to 7 rounds," in *Proc. of Int. Conf. Fast Software Encryption*, pp. 571-590, March 3-5, 2014. Article (CrossRef Link).

[15] D. Boneh, R. A. DeMillo, R. J. Lipton and M. Yung: "On the importance of checking cryptographic protocols for faults," in *Proc. of Int. Conf. Theory Application Cryptographic Techniques*, pp. 37-51, May 11-15, 1997. Article (CrossRef Link).

[16] D. Boneh, R. A. DeMillo and R. J. Lipton: "On the importance of eliminating errors in cryptographic computations," *J. CRYPTOL.*, vol. 14, no. 2, pp. 101-119, March, 2001. Article (CrossRef Link).

[17] E. Biham and A. Shamir: "Differential fault analysis of secret key cryptosystems," in *Proc. of 17th Annual Int. Cryptology Conf.*, pp. 513-525, August 15-19, 1997. Article (CrossRef Link).

[18] M. Joye, J. J. Quisquater, Y. Sung-Ming and M. Yung, "Observability analysis-detecting when improved cryptosystems fail," in *Proc. of Cryptographer's Track RSA Conf.*, pp. 17-29, February 18-22, 2002. Article (CrossRef Link).
[19] I. C. Lin and C. C. Chang: "Security enhancement for digital signature schemes with fault tolerance in RSA," *Inform. Sciences*, vol. 177, no. 19, pp. 4031-4039, February 24-24, 2007. Article (CrossRef Link).
[20] L. Hemme and L. Hoffmann: "Differential fault analysis on the SHA1 compression function," in *Proc. of Fault Diagnosis and Tolerance in Cryptography*, pp. 54-62, September 28-28, 2011. Article (CrossRef Link).
[21] W. Fischer and A. C. Reuter: "Differential fault analysis on Grøstl," in *Proc. of Int. Workshop on Fault Diagnosis and Tolerance in Cryptography*, pp. 44-54, September 9-9, 2012. Article (CrossRef Link).
[22] R. AlTawy and A. M. Youssef: "Differential fault analysis of Streebog," in *Proc. of Int. Conf. on Information Security Practice and Experience*, pp. 35-49, May 5-8, 2015. Article (CrossRef Link).
[23] N. Bagheri, N. Ghaedi and K. S. Sanadhya: "Differential fault analysis of SHA-3," in *Proc. of Int. Conf. in Cryptology*, pp. 253-269, December 6-9, 2015. Article (CrossRef Link).
[24] G. Piret and J. J. Quisquater: "A differential fault attack technique against SPN structures, with application to the AES and KHAZAD," in *Proc. of Int. Workshop on Cryptographic Hardware and Embedded Systems*, pp. 77-88, September 8-10, 2003. Article (CrossRef Link).
[25] M. Amir, T. M. S. Mohammad and S. Mahmoud: "A generalized method of differential fault attack against AES cryptosystem," in *Proc. of Int. Workshop on Cryptographic Hardware and Embedded Systems*, pp. 91-100, October 10-13, 2006. Article (CrossRef Link).
[26] P. Dusart, G. Letourneux and O. Vivolo, "Differential fault analysis on A.E.S," in *Proc. of* 1st Int. Conf. Applied Cryptography and Network Security, pp. 293-306, October 16-19, 2003. Article (CrossRef Link).
[27] J. Blomer and J. P. Seifert: "Fault based cryptanalysis of the advanced encryption standard (AES)," in *Proc. of Int. Conf. on Financial Cryptography*, pp. 162-181, January 27-30, 2003. Article (CrossRef Link).
[28] M. Bellare, R. Canetti and H. Krawczyk: "Keying hash functions for message authentication," in *Proc. of Annual Int. Cryptology Conf.*, pp. 1-15, August 18–22, 1996. Article (CrossRef Link).
[29] M. Karpovsky, K. J. Kulikowski and A. Taubin: "Differential fault analysis attack resistant architectures for the Advanced Encryption Standard," in *Proc. of Int. Conf. Smart Card Research and Advanced Applications VI*, pp. 177-192, August 22–27, 2004. Article (CrossRef Link).

Wei Li is currently an associate professor in School of Computer Science and Technology, Donghua University. She was awarded as B.S. degree in engineering from Anhui University in 2002, and her M.S. degree and Ph.D. degree in engineering in 2006 and 2009, both from Shanghai Jiao Tong University. She serves as the member for CACR (China Association of Cryptologic Research), CCF (China Computer Federation) and ACM. Her research interests include the design and analysis of symmetric ciphers.

Zhiyong Gao is currently a Master candidate in School of Computer Science and Technology, Donghua University. His research interests include security analysis of hash functions.

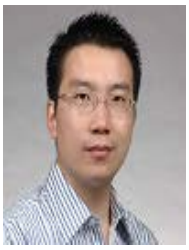
Dawu Gu is a professor at Shanghai Jiao Tong University in Computer Science and Engineering Department. He was awarded a B.S. degree in applied mathematics in 1992, and a Ph.D. degree in cryptograpgy in 1998, both from Xidian University of China. He serves as technical committee members for CACR (China Association of Cryptologic Research) and CCF (China Computer Federation), also as the members of ACM, IACR, IEICE. He was the winner of New Century Excellent Talent Program made by Ministry of Education of China in 2005. He has been invited as Chairs and TPC members for many international conferences like E-Forensics, ISPEC, ICIS, ACSA, CNCC, etc. His research interests cover cryptology and computer security. He has got over 100 scientific papers in academic journals and conferences.

Chenyu Ge is currently a Master candidate in School of Computer Science and Technology, Donghua University. Her research interests include security analysis of lightweight ciphers.

Linfeng Liao is currently a Master candidate in School of Computer Science and Technology, Donghua University. His research interests include security analysis of symmetric ciphers.

Zhihong Zhou is a lecturer in Shanghai Key Laboratory of Integrate Administration Technologies for Information Security, Shanghai Jiao Tong Univeristy. He was awarded her Ph.D. degree from Zhejiang Univeristy in 2008. His research interests include information security.

Ya Liu is currently a lecturer in Department of Computer Science and Engineering, University of Shanghai for Science and Technology. She was awarded her Ph.D. degree from Shanghai Jiao Tong University in 2013. Her research interests include the design and analysis of symmetric ciphers and computational number theory.



Zhiqiang Liu is now a lecturer in the department of Computer Science and Engineering, Shanghai Jiao Tong University. He received his B.S. degree and M.S. degree in Mathematics, and Ph.D. degree in Cryptography from Shanghai Jiao Tong University in 1998, 2001 and 2012 respectively. From 2001 to 2008, he worked in ZTE, Alcatel and VLI in the realm of NextGeneration Network (NGN)/IP Multimedia Subsystem (IMS). Currently, his research interests include cryptanalysis and design of block ciphers and hash functions.