Algebraic Differential Fault Analysis on SIMON block cipher

Duc-Phong Le, Sze Ling Yeo, and Khoongming Khoo

Abstract—Algebraic differential fault attack (ADFA) is an attack in which an attacker combines a differential fault attack and an algebraic technique to break a targeted cipher. In this paper, we present three attacks using three different algebraic techniques combined with a differential fault attack in the bit-flip fault model to break the SIMON block cipher. First, we introduce a new analytic method which is based on a differential trail between the correct and faulty ciphertexts. This method is able to recover the entire master key of any member of the SIMON family by injecting faults into a single round of the cipher. In our second attack, we present a simplified Gröbner basis algorithm to solve the faulty system. We show that this method could totally break SIMON ciphers with only 3 to 5 faults injected. Our third attack combines a fault attack with a modern SAT solver. By guessing some key bits and with only a single fault injected at the round $T-6$, where T is the number of rounds of a SIMON cipher, this combined attack could manage to recover a master key of the cipher. For the last two attacks, we perform experiments to demonstrate the effectiveness of our attacks. These experiments are implemented on personal computers and run in very reasonable timing.

Index Terms—Lightweight Block Ciphers, Fault Attacks, Algebraic Differential Fault Attacks, Algebraic Techniques, Gröbner basis, SAT solver.

1 Introduction

Lightweight block ciphers are increasingly implemented in resource constrained devices such as RFID or microchips. SIMON is a family of lightweight block ciphers, proposed by a team from the NSA [1]. This family, consisting of a variety of members with different block sizes and key sizes, is optimized for hardware performance.

Fault analysis is a very efficient attack on cryptographic implementations. Such an attack tries to influence the behavior of a device and determine sensitive information by examining the effects. There have been several mechanisms to inject faults into microprocessors. Examples include changes of the power supply, the clock frequency [2] or an intensive lighting of the circuit [3]. In most cases, this forces a change in the data in one of the registers.

The first fault attack against RSA-CRT implementation was reported in a Bellcore press in 1996 and subsequently analyzed by Boneh, DeMillo and Lipton in [4]. They showed that many implementations of RSA signatures and other public key algorithms are vulnerable to a certain transient fault occurring during the processing phase. In particular, the RSA-CRT implementation suffers from a high risk to be compromised using one erroneous result. Biham and Shamir then introduced Differential Fault Analysis (DFA) against symmetric cryptosystems such as the DES [5]. They assume that an attacker can disturb DES computations using the same - but *unknown* - plaintext at the last (three) DES round(s). The wrong ciphers provide a system of equations for the unknown last round key bits that finally reveal the correct key value.

Algebraic attack is a method of cryptanalysis against ciphers. The main idea of such an attack is to represent a cipher via algebraic equations, followed by an algebraic solving method to find solutions. Unlike statistical approaches such as differential cryptanalysis [6] or linear cryptanalysis [7], this attack is more feasible in the sense that it requires much fewer input data. Although the efficiency of this attack depends on the structure of the cipher, the recent advances in finding good representations of ciphers [8], and in algebraic solving methods [9]–[12] make this attack more feasible.

This work combines various algebraic techniques and the differential fault analysis to attack SIMON ciphers. It is assumed that our attacks are performed in the bit-flip fault model, i.e., a single injected fault produces exactly one bit flip. In the first attack, we construct differential trails for the SIMON ciphers. We demonstrate how these differential trails can be utilized to improve differential fault attacks against the SIMON ciphers. Our analysis shows that by injecting faults into only a *single round*, the full secret key of *any* SIMON ciphers can be found without guessing any key bit. The knowledge of plaintext is not required in this attack. Our second attack is a combination of a fault injection attack with Gröbner basis to retrieve the secret key. This attack will be implemented using Magma [13]. Our third attack combines a fault attack with a modern SAT solver. By guessing some key bits, this attack succeeds in recovering the secret key by using a single fault. In this attack, we use Magma to generate systems of equations and CryptoMiniSat 5.0.1 [10] as a SAT solver. Unlike the first attack, our last two attacks may require the knowledge of the plaintexts involved.

Among the three above attacks, the first attack, which

D-P. Le is with Canadian Institute for Cybersecurity, University of New Brunswick, Canada. E-mail: le.duc.phong@unb.ca
S. L. Yeo is with Data Security Unit, Cyber Security & Intelligence, Institute for Infocomm Research (I^2R), Singapore. E-mail: slyeo@i2r.a-star.edu.sg
KM. Khoo is with DSO National Laboratories, Singapore. Email: kkhoongm@dso.org.sg

is based on an analytic approach, is the most straightforward to implement. However, it requires at least $n/2$ fault injections. On the other hand, exploiting algebraic methods such as Gröbner basis and SAT solver reduce the number of fault injections needed at the cost of higher complexity. In our last attack, we performed experiments to show that one fault injection is sufficient to deduce the master key via SAT solvers, but this attack requires the attacker to guess a portion of the key bits.

To demonstrate our techniques, we performed tests on the SIMON block cipher. Nonetheless, the attacks described in this work are generic and can be applied to other lightweight block ciphers.

The paper is organized as follows. Section 2 briefly recalls differential fault attacks on block ciphers and SIMON ciphers. Section 3 describes our algebraic differential fault attack using a differential trail. Our attacks using a simplified Gröbner basis and SAT solvers are presented in Section 4 and Section 5, respectively. We also present our experiments in these sections. Finally, we conclude in Section 6.

2 PRELIMINARIES

2.1 Notations.

In the rest of the paper, we make use of the following notations. In general, we use capital letters for vectors or strings while small letters are used to represent individual bits.

T	: denotes the total number of rounds of the cipher.
(X^i, Y^i)	: denotes input of round i, $i = 0, 1, \ldots, T-1$.
(X^T, Y^T)	: denotes ciphertext.
(X'^i, Y'^i)	: denotes faulty input of round i, $i = 0, 1, \ldots, T-1$.
(X'^T, Y'^T)	: denotes faulty ciphertext.
$X^i + Y^i$	: denotes 'XOR' operation of X^i and Y^i.
$X^i Y^i$	: denotes 'AND' operation of X^i and Y^i.
(x_j^i, y_j^i)	: variables denoting bit j of the input of the round i, $i = 0, 1, \ldots, T, \quad j = 0, 1, \ldots, n-1$.
(x'^i_j, y'^i_j)	: variables denoting bit j of the faulty input of the round i, $i = 0, 1, \ldots, T, \quad j = 0, 1, \ldots, n-1$.
$x_j^i + y_j^i$	: denotes bitwise 'XOR' operation of x_j^i and y_j^i.
$x_j^i y_j^i$	: denotes bitwise 'AND' operation of x_j^i and y_j^i.
K^i	: denotes the round key in round i, $i = 0, 1, \ldots, T-1$.
k_j^i	: denotes $j^{th}, 0 \leq j \leq n-1$ bit round key of round i.
Δ^i	: denotes the difference between correct and faulty inputs of round i.
δ_j^i	: denotes the difference at bit j of δ^i, $i = 0, 1, \ldots, T$, $j = 0, 1, \ldots, n-1$.

2.2 Differential Fault Attacks on Block ciphers

Fault attacks aim at retrieving the secret information by disturbing the execution of a cryptographic implementation. The first fault attack was introduced on RSA-CRT implementation, an asymmetric cryptosystem. In this paper, we investigate fault attacks against symmetric cryptosystems, so-called *Differential Fault Analysis* or *DFA*. This attack was first proposed by Biham and Shamir in [5]. Subsequently, various DFAs on block ciphers were carried out, including attacks against AES [14], Triple DES [15], IDEA [16], SIMON [17].

In principle, the attacker will inject a fault in some rounds when executing a cryptographic implementation to obtain a faulty ciphertext. By analyzing the difference between the correct and faulty ciphertexts, the last round key can be revealed. With knowledge of the last round key, the attacker decrypts the correct ciphertext to obtain the input of the last round, which is the output of the second last round. Then, the attacker repeats this procedure to obtain more round keys until the secret key can be deduced by the key schedule. Readers are referred to [5] for more details.

2.2.1 Fault Attack Model

The most useful methods to generate faults were reported in [2], [3], [18]. These attacks allow a specific control of a single register, a bus or memory. As in [19], four different fault models have been defined:

- **Random fault model**: the bits are changed to a uniformly distributed random value;
- **Bit-flip fault model**: in this case, affected bits are flipped to their complementary value;
- **Stuck-at fault model**: the fault sets the bits to 0 or to 1, depending on the underlying hardware;
- **Unknown constant fault model**: the fault always sets the bits to the same unknown value.

2.2.2 Algebraic Differential Fault Attacks

The first combination of algebraic and fault attacks to break block ciphers was presented by Courtois et al. [20]. In principle, an algebraic differential fault attack works as follows:

- representing a cipher using Algebraic Normal Form (ANF) equations
- representing the fault using ANF equations
- solving the resulting ANF equations system by using Algebraic tools, e.g., SAT solvers [10] or Gröbner basis [11], [12].

Courtois et al. demonstrated such attacks against DES block cipher in [20]. The attacks first represent each DES S-Box as a system of cubic equations. Then, they convert the system of equations to a SAT problem, i.e., Conjunctive Normal Form (CNF) clauses. Finally, the attacks solve the SAT problem by using MiniSAT 2.0, the winner of SAT-Race 2006 competition [21].

Courtois et al. reported two different attacks. First, by injecting faults at the 14^{th} round and guessing 20 key bits, their attack could recover the key with 2 faulty ciphertexts with 2 bits flipped. This attack works 200 times as fast as the brute force attack. In order to use only 1 fault injected at this round, their attack requires to produce a precise fault on 10 bits. Next, by injecting faults at the 13^{th} round and guessing 24 key bits, their attack could recover the key with only 1 faulty ciphertext with 2 bits flipped. The attack is 10 times as fast as the brute force attack. This is due to the fact that, when one bit is flipped at the round 13, the diffusion of DES may affect all S-Boxes in the round 16.

2.3 SIMON block cipher

We will deploy our algebraic differential fault attacks on the SIMON cipher [1]. This section briefly reviews this cipher, as well as DFAs on it.

The SIMON family of lightweight block ciphers was designed by the National Security Agency and was optimized for hardware implementations. It is based on a typical Feistel design (see Figure 1) and comprises 3 simple operations, namely, the bitwise 'AND', 'rotation' and 'XOR' operations. Let n denote the word size. Then, as in [1], SIMON-$2n/mn$ denotes SIMON with block size and key size $2n$ and mn, respectively, where $m = 2, 3$ or 4, the number of n bit registers related to the key schedule process. The following table summarizes the variants of the SIMON family.

TABLE 1
Members of the SIMON family

Cipher	Block size $2n$	Key words m	Key size mn	Rounds T
SIMON-32/64	32	4	64	32
SIMON-48/72	48	3	72	36
SIMON-48/96	48	4	96	36
SIMON-64/96	64	3	96	42
SIMON-64/128	64	4	128	44
SIMON-96/96	96	2	96	52
SIMON-96/144	96	3	144	54
SIMON-128/128	128	2	128	68
SIMON-128/196	128	3	196	69
SIMON-128/256	128	4	256	72

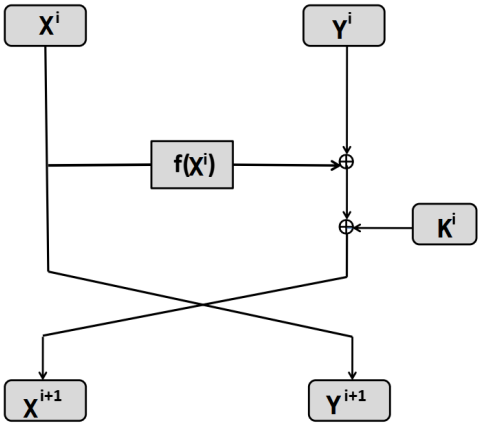

Fig. 1. One round of SIMON cipher

2.3.1 DFAs on SIMON cipher.

We briefly review existing differential fault attacks against SIMON cipher in the *bit-flip fault model*.

Attack at the second last round. Tupsamudre et al. [17] introduced the first differential fault analysis against the SIMON and SPECK ciphers. For the SIMON block cipher, they demonstrated two differential fault attacks in two different fault models. In the bit-flip fault model, in which a fault injected will flip a bit of the input, 2 bits of the last round key can be found with one successful fault. Eventually the n-bit key could be revealed by using $n/2$ faulty ciphertexts. In the random byte-flip fault model, in which a fault injected will modify a byte to a random value, Tupsamudre et al. showed that the last round key can be successfully recovered using $n/8$ faulty ciphertexts on average.

Basically, their attacks exploit the information leaked by the 'AND' operation which is the only non-linear function of SIMON. Specifically, the attacks injected a fault in the intermediate left half ciphertext X^{T-2}, where T is the number of rounds of SIMON. The value of the position of the fault could be deduced due to the following equation.

$$e = X^T + X'^t + f(Y^T) + f(Y'^t) \tag{1}$$

From the following equation, observe that one can deduce the last round key K^{T-1} if the value X^{T-2} is known.

$$K^{T-1} = X^{T-2} + f(Y^T) + X^T \tag{2}$$

where $f(X) = (rotr(X,1)\, rotr(X,8)) + rotr(X,2)$, and $rotr(X,i)$ denotes a right rotation of X by i bits.

Attack at the third last round. Vasquez et al. [22] extended the work in [17] by injecting a fault into the third round instead of the second round from the last. By doing so, they could deduce the secret key by injecting faults into either a single round if the number of registers $m = 2$, or 2 rounds if $m > 2$. The number of faults required to retrieve the secret key is about half of the attack in [17] (see [22, Table VIII] for more details).

3 Algebraic Differential Fault Attack using Differential Trails

Given a plaintext P, the SIMON encryption function outputs a corresponding ciphertext C. Let (X^i, Y^i) denote the intermediate inputs of P at round i, for $0 \leq i \leq T-1$, where T denotes the total number of rounds of the cipher, e.g., $T = 32$ for SIMON-32/64 (see Table 1).

Assume that a fault is injected into the input X^r of an intermediate round r and causes a bit-flip at the position l. Let X'^r be the fault value, so X^r and X'^r will be different at the l^{th} bit and identical everywhere else. In other words, $x^r_j = x'^r_j + 1$ for $j = l$ and $x^r_j = x'^r_j$ for $j \neq l$. Let $(x^r_{n-1}, \ldots, x^r_0)$, and $(y^r_{n-1}, \ldots, y^r_0)$ denote the input of round r. The following relations hold for any $j = 0, \ldots, n-1$:

$$\begin{aligned} x^{r+1}_j &= f(x^r_j) + y^r_j + k^r_j, \\ y^{r+1}_j &= x^r_j, \end{aligned} \tag{3}$$

where $f(x^r_j) = (x^r_{(j-1) \bmod n}\ x^r_{(j-8) \bmod n}) + x^r_{(j-2) \bmod n}$. As all indices are computed modulo n, in what follows, x^i_j will denote $x^i_{j \bmod n}$. We first consider the following lemma.

Lemma 3.1. Let $\delta^i_j = x^i_j + x'^i_j$ for $r \leq i \leq T$ be the differential representation of two correct and faulty bits x^i_j and x'^i_j. We have, $\delta^r_j = 0$ for $j \neq l$ and equal to 1 if $j = l$, and:

$$\delta^{i+1}_j = \delta^i_{j-1} x^i_{j-8} + \delta^i_{j-8} x^i_{j-1} + \delta^i_{j-1} \delta^i_{j-8} + \delta^i_{j-2} + \delta^{i-1}_j \tag{4}$$

Proof: From Equation (3), we have:

$$\begin{aligned} x^{i+1}_j &= x^i_{j-1} x^i_{j-8} + x^i_{j-2} + y^i_j + k^i_j, \text{ and} \\ x'^{(i+1)}_j &= x'^i_{j-1} x'^i_{j-8} + x'^i_{j-2} + y'^i_j + k^i_j \end{aligned}$$

Note that $y_j^i = x_j^{i-1}$. Summing up the two above equations gives:

$$\delta_j^{i+1} = x_{j-1}^i x_{j-8}^i + x_{j-1}'^i x_{j-8}'^i + \delta_{j-2}^i + \delta_j^{i-1}$$

We have:

$$\begin{aligned}
&x_{j-1}^i x_{j-8}^i + x_{j-1}'^i x_{j-8}'^i \\
&= (x_{j-1}^i + x_{j-1}'^i)(x_{j-8}^i + x_{j-8}'^i) + x_{j-1}'^i x_{j-8}^i + x_{j-1}^i x_{j-8}'^i \\
&= \delta_{j-1}^i \delta_{j-8}^i + (x_{j-1}'^i x_{j-8}^i + x_{j-1}^i x_{j-8}^i) + (x_{j-1}^i x_{j-8}'^i + x_{j-1}^i x_{j-8}^i) \\
&= \delta_{j-1}^i \delta_{j-8}^i + \delta_{j-1}^i x_{j-8}^i + \delta_{j-8}^i x_{j-1}^i
\end{aligned}$$

Thus,

$$\delta_j^{i+1} = \delta_{j-1}^i x_{j-8}^i + \delta_{j-8}^i x_{j-1}^i + \delta_{j-1}^i \delta_{j-8}^i + \delta_{j-2}^i + \delta_j^{i-1}.$$

□

Remark 1. From Equation (4), it can be seen that δ_j^{i+1} depends on 2 bits of the intermediate input X^i and 4 bits of the input difference Δ^i. Furthermore:

- if $\delta_{j-1}^i = \delta_{j-8}^i = 0$ and $\delta_{j-2}^i = \delta_j^{i-1}$, then $\delta_j^{i+1} = 0$, which is independent of the intermediate input X^i.
- if $\delta_{j-1}^i = 0$, $\delta_{j-8}^i = 0$, $\delta_j^{i-1} = 0$, and $\delta_{j-2}^i = 1$, then $\delta_j^{i+1} = 1$. This will give us a pattern of 1 in difference trails. For example, the position of 1 in the next round will be the position of 1 in the current round shifted to the left by two in Table 4 for SIMON-32/64. This pattern happens up to the fifth round from the round injected faults. Based on this pattern we may deduce the position of faults.

Lemma 3.1 says that each bit δ_j^i can be represented in terms of intermediate plaintext bits and difference bits in the previous rounds. The preceding remark indicates that these expressions can be simplified or deduced the difference bits in the earlier rounds are known. This motivates us to construct a differential trail table to record and trace the δ_j^i's using Lemma 3.1. In fact, this differential trail table is the main tool used in the subsequent attacks.

In the rest of this section, we will look at differential fault attack using Algorithm 1, where the key can be deduced directly by studying the differential trail. Then in Sections 4 and 5, we will look at algebraic fault attacks where fewer faults are injected at earlier rounds but the equations are more complex. Thus algebraic techniques like Gröbner basis and SAT solvers need to be employed to solve for the keys.

3.1 Attacking at the second last round

We first demonstrate our idea by injecting faults into the second last round $T-2$ of SIMON-32/64, and show that our attack is equivalent to the attack in [17]. Without loss of generality, suppose that we flip bit 15 at round $T-2$. Table 2 gives differential trails between the correct intermediate ciphertext and the faulty intermediate ciphertexts for the last 3 rounds (including the right half of round $T-2$ or equivalently, the left half of round $T-3$).

Since the difference in the ciphertexts (i.e., $C + C'$) is known, it follows that the differences Δ^{T-1} and Δ^T are known. Consequently, one can deduce the values of x_6^{T-2} and x_8^{T-2}. From Equation (2), we could retrieve two bits

ALGORITHM 1: Differential Fault Attacks using Differential Trails

- STEP 1: choose a random plaintext P, inputs to a SIMON encryption function, and get a ciphertext C.
- STEP 2: re-run the input P, choose an intermediate round r and inject a fault. Assume that one bit at the position l of the input X^r will be flipped. In other words, the correct and faulty intermediate ciphertexts X^r and X'^r will be different at the l^{th} bit and identical everywhere else, that is, $x_l^r = x_l'^r + 1$ and $x_j^r = x_j'^r$ for $j \neq l$.
- STEP 3: find output differences $\Delta^i = X^i + X'^i$, for $r \leq i \leq T$. For each round, these output differences could be represented by n values of $0, 1$ or algebraic equations of input variables, where n is the size of the left input part X^r. Construct the differential trail table to record these differences.
- STEP 4: from these output differences, deduce input bits of the round $T-2$, and earlier rounds, then retrieve corresponding bits of the last round key K^{T-1} due to Equation (2).
- STEP 5: repeat steps 2–4 to recover the full round key K^{T-1}.
- STEP 6: use K^{T-1}, decrypt the last round, and continue this process until sufficient round keys are obtained to recover the entire unknown master key. The required number of round keys needed depends on the key schedules of different variants of SIMON family. For example, we need to retrieve 4 round keys for SIMON-32/64, and 2 round keys for SIMON-128/128.

TABLE 2
Differential trail for the left half of the last 3 rounds of SIMON-32/64 when flipping bit 15. A value 1 (or 0) means that at the considered bit, the value of this bit in the correct output is different (or the same, respectively) from the faulty output. The notation $*$ denotes a non-linear expression that would not be used in our attack. As the attacker flipped the bit 15, $\delta_j^{T-2} = 1$, for $j = 15$, and $\delta_j^{T-2} = 0$, for $j \neq 15$.

Bit	15	14	13	12	11	10	9	8
Δ^{T-3}	0	0	0	0	0	0	0	0
Δ^{T-2}	1	0	0	0	0	0	0	0
Δ^{T-1}	0	0	0	0	0	0	0	0
Δ^T	$*$	0	0	0	0	0	$x_6^{T-2} + x_8^{T-1}$	$*$

Bit	7	6	5	4	3	2	1	0
Δ^{T-3}	0	0	0	0	0	0	0	0
Δ^{T-2}	0	0	0	0	0	0	0	0
Δ^{T-1}	x_6^{T-2}	0	0	0	0	0	1	x_8^{T-2}
Δ^T	0	0	0	0	1	$x_8^{T-2} + x_{10}^{T-1}$	$*$	0

of the last round key, namely, k_6^{T-1} and k_8^{T-1}. This is equivalent to Tupsamudre et al.'s attack [17]. If the attacker can control the position of the flipped bit, he needs 8 faults to obtain the last round key of SIMON-32/64.

Remark 2. The position of the fault could be easily deduced by using either Equation (1) or number pattern in Δ^{T-1} and observations in Remark 1.

3.2 Attacking at the third last round

This section extends our attack idea by injecting faults into the left input part of the third last round (that is, X^{T-3}) of SIMON-32/64. We first present the table of differential trails from round $T-3$ onwards. Likewise, without loss

of generality, we flip bit 15 at round $T-3$. Note that the position of a fault is easily deduced from patterns in Δ^{T-2} (that can be computed as shown below), and observations in Remark 1. Differential trails are shown in Table 3.

Consider the known ciphertext difference $C + C'$ at round T and trace backwards. Since the left half of round $T-1$ is the right half of round T, it follows that the left half difference at round $T-1$ is known. On the other hand, by Equation (4), we have:

$$\delta_j^{T-2} = x_{j-1}^{T-1}\delta_{j-8}^{T-1} + x_{j-8}^{T-1}\delta_{j-1}^{T-1} + \delta_{j-1}^{T-1}\delta_{j-8}^{T-1} + \delta_{j-2}^{T-1} + \delta_j^T.$$

Since the attacker knows Δ^T, Δ^{T-1} and X^{T-1}, she is able to compute Δ^{T-2}. Once Δ^{T-2} and Δ^{T-1} are known, the attacker will deduce the values of x_6^{T-3}, x_8^{T-3}, and then x_8^{T-2}, x_{10}^{T-2} from Table 3. Depending on the values of x_6^{T-3} and x_8^{T-3} (if they are not all zero), the attacker would be able to recover bits x_0^{T-2}, x_7^{T-2}, x_9^{T-2}, and x_{14}^{T-2} by analyzing the quadratic equations in the vector Δ_{T-1}. If bits of X^{T-2} are uniform and independent and identically distributed, that is the probability that each bit equals to 0 or 1 is the same so that $Pr(x_j^i = 0) = Pr(x_j^i = 1) = 1/2$, then the expected number of bits in X^{T-2} recovered by a single bit-flip in X^{T-3} is:

$$2 + \frac{1}{4} \times 0 + \frac{1}{2} \times 2 + \frac{1}{4} \times 2 = 3.5 \text{ bits.}$$

Indeed, both bits $x_6^{T-3} = x_8^{T-3} = 0$ with probability $1/4$, and in this case, we cannot recover any extra bit. If one or both of them equal to 1, we could recover 2 bits. On average, we could recover 3.5 bits for a single fault. This expected number is the same as the one given in [22, Section III.E].

Similar to the above analysis, for each bit of X^{T-2} found, the attacker can retrieve one key bit of the last round key. After recovering all bits of K^{T-1}, she decrypts the last round to get (X^{T-1}, Y^{T-1}) and (X'^{T-1}, Y'^{T-1}). Then with her knowledge of X^{T-3}, she is able to recover the round key K^{T-2}. Thus, if she can control the position of the faults, she is able to uncover the last two round keys of SIMON-32/64 by using only 8 faults.

3.3 Extending the attack to recover the full secret key

In the above sections, we demonstrated how to recover one and two round keys by injecting faults into a single round. Note that for members of SIMON family, where the key words m equals to 2 such as SIMON-96/96 or SIMON-128/128, two round keys are sufficient to uncover the entire master key. For other variants where $m = 3$ (and $m = 4$), the attacker needs to retrieve 3 (and 4 respectively) round keys to be able to fully extract the secret key.

In fact, the attacks in [17] require faults injected in $2, 3$, or 4 consecutive rounds to retrieve the corresponding number of round keys. The attack in [22] requires faults injected into 1 round (and 2 rounds) to retrieve 2 round keys (and 3 or 4 round keys, respectively). In this section, we describe how to extract the master key of variants of SIMON family where $m > 2$ by also injecting faults into a single round. In this attack, we inject a fault in round $T-5$. As before, we first present the table of differential trails for the last six rounds of SIMON-32/64.

As can be seen from Tables 2-3, for each difference $\delta_j^i = 1$, we could obtain 2 linear expressions in the next difference vector Δ^{i+1}. This could be easily verified from Lemma 3.1. When the number of rounds needed to find the differences increases, non-linear expressions become more complex (related to more variables at different rounds and at higher degrees). The variables in these expressions could not be deduced in a trivial way. Thus, we don't show them in Table 4. In the following sections, we will present other algebraic techniques, including Gröbner basis and SAT Solvers to solve these equations.

Deducing the position of a fault in this case is more challenging even when we know the values of Δ^T, Δ^{T-1} and Δ^{T-2}. However, Table 4 shows some patterns, for example, 4 consecutive 0s, $0, 1$ (bits 6, 5) in Δ^{T-2}. Using this fact and observations in Remark 1 could reduce the searching space of the fault position.

Similar to the preceding attacks, an attacker is able to get the value of Δ^{T-2} because she knows Δ^T, Δ^{T-1} and X^{T-1}. Once Δ^{T-2} and Δ^{T-1} are known, the attacker will deduce the values of x_{12}^{T-2} and x_{14}^{T-2} by summing up $\delta_{11}^{T-2} + \delta_{13}^{T-1}$, and $\delta_4^{T-2} + \delta_6^{T-1}$, respectively. She can continue to inject faults at different possible positions to recover all bits of X^{T-2}, and hence K^{T-1}. Then, she decrypts the last round with K^{T-1} to obtain (X^{T-1}, Y^{T-1}) and (X'^{T-1}, Y'^{T-1}) and continue this process to get more round keys. As such, analyzing up to the round $T-5$ will allow us to recover 4 round keys, which would be sufficient to obtain the secret master key of any member of the SIMON family.

Remark 3. Although we described in this section an attack against SIMON-32/64, it is straightforward to apply the same attack procedure to other members of SIMON family. The differential trails are easy to construct by using Equation (4). Appendix A lists differential trails of all members of SIMON family. In general, if one could control the position of faults, our attack needs at least $n/2$ faults to uncover the master key. For members with $m = 2$, analyzing quadratic equations could allow us to recover more than two bits (as described in Section 3.2). It is worth to note that, this attack requires no knowledge of the plaintext.

4 Algebraic Differential Fault Attacks using Gröbner basis

By examining the above attack using differential trails, observe that the following steps were carried out:

- Using the fact that Δ^{T-2} can be deduced from Δ^{T-1} and Δ^T, find some bits j from the differential trail table which are simple expressions of bits in X^{T-2} to determine some bits in X^{T-2}.
- Deduce the corresponding bits in K^{T-1} from the known bits in X^{T-2};
- Continue to deduce more plaintext bits from the relations in Δ^{T-2};
- Continue the above procedure with more fault injections.

TABLE 3
Differential trail for the left half of last 4 rounds when flipping bit 15. The notation $*$ in the last row denotes a non-linear expression that would not be used in our analysis attack.

Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Δ^{T-4}	0	0	0	0	0	0	0	0
Δ^{T-3}	1	0	0	0	0	0	0	0
Δ^{T-2}	0	0	0	0	0	0	0	0
Δ^{T-1}	$x_6^{T-3}x_{14}^{T-2}+1$	0	0	0	0	0	$x_6^{T-3}+x_8^{T-2}$	$x_6^{T-3}x_0^{T-2}+x_8^{T-3}x_7^{T-2}+x_6^{T-3}x_8^{T-3}$
Δ^{T}	0	0	0	0	$x_6^{T-3}+x_8^{T-2}+x_{10}^{T-1}$	$*$	$*$	0

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Δ^{T-4}	0	0	0	0	0	0	0	0
Δ^{T-3}	0	0	0	0	0	0	0	0
Δ^{T-2}	x_6^{T-3}	0	0	0	0	0	1	x_8^{T-3}
Δ^{T-1}	0	0	0	0	1	$x_8^{T-3}+x_{10}^{T-2}$	$x_8^{T-3}x_9^{T-2}$	0
Δ^{T}	$*$	0	1	$x_8^{T-3}+x_{10}^{T-2}+x_{12}^{T-1}$	$*$	$*$	$*$	$*$

TABLE 4
Differential trail for the left half of the last 6 rounds of SIMON-32/64 when flipping bit 15. The notation $*$ denotes a non-linear expression that would not be used in our attack. Δ_T could be represented by non-linear expressions, which are known and will not be used in our analysis attack.

Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Δ^{T-6}	0	0	0	0	0	0	0	0
Δ^{T-5}	1	0	0	0	0	0	0	0
Δ^{T-4}	0	0	0	0	0	0	0	0
Δ^{T-3}	$*$	0	0	0	0	0	$x_6^{T-5}+x_8^{T-4}$	$*$
Δ^{T-2}	0	0	0	0	$x_6^{T-5}+x_8^{T-4}+x_{10}^{T-3}$	$*$	$*$	0
Δ^{T-1}	$*$	0	$x_6^{T-5}+x_8^{T-4}+x_{10}^{T-3}+x_{12}^{T-2}$	$*$	$*$	$*$	$*$	$*$
Δ^{T}	Known values							

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Δ^{T-6}	0	0	0	0	0	0	0	0
Δ^{T-5}	0	0	0	0	0	0	0	0
Δ^{T-4}	x_6^{T-5}	0	0	0	0	0	1	x_8^{T-5}
Δ^{T-3}	0	0	0	0	1	$x_8^{T-5}+x_{10}^{T-4}$	$*$	0
Δ^{T-2}	$*$	0	1	$x_8^{T-5}+x_{10}^{T-4}+x_{12}^{T-3}$	$*$	$*$	$*$	$*$
Δ^{T-1}	1	$x_8^{T-5}+x_{10}^{T-4}+x_{12}^{T-3}+x_{14}^{T-2}$	$*$	$*$	$*$	$*$	$*$	0
Δ^{T}	Known values							

Essentially, one picks out simple relations from the differential trail tables corresponding to different fault injections to recover the key bits progressively. A natural way to extend the above approach is to consider all the possible known relations given the fault injections to construct an algebraic system and to solve this algebraic system via algebraic solving tools. In this section and the next, we will demonstrate this approach using both Gröbner basis and SAT solvers as our algebraic solving tools.

4.1 Constructing the algebraic system for the fault injections

Let $r \leq T$ be some positive integer. We consider a bit flip at the input of round r, say at bit $n-1$. As before, we obtain two pairs of plaintext/ciphertexts, namely, (P, C) and (P', C'), where C' is the faulty ciphertext. Let (X^i, Y^i) and (X'^i, Y'^i) denote the intermediate input values for the encryptions of P and P', respectively. By our construction, the following hold:

$$\begin{aligned} (X^i, Y^i) &= (X'^i, Y'^i), && \text{for } 0 \leq i < r \\ x_{n-1}^r &= x_{n-1}'^r + 1 \\ x_j^r &= x_j'^r, && \text{for } j \neq n-1 \end{aligned}$$

In this case, after round r, one will obtain a differential trail as in Table 6. One sees from the differential trail that up to round $r+4$, there remain some positions with known bit differences and up to round $r+5$, there exist positions where the bit differences are linear sums of some of the intermediate plaintext bits. In addition, each of the bits δ_j^i, where $i \geq r$, can be expressed in terms of difference bits as well as intermediate input bits given by Equation 4. We thus have the following basic algorithm (Algorithm 2) to carry out an algebraic differential fault attack.

Remark 4.

- We have described the algorithm in which the faults can be controlled. In the case where the faults are random, the algorithm can be easily modified.
- As seen below, r is typically taken to be in $\{T-5, T-6, T-7\}$.
- When the number of faults t is small, there may be more than one key satisfying all the polynomials in $\mathcal{A}$. As such, one can test each possible key with the original plaintext/ciphertext pair as in the last step of the algorithm.

After running Algorithm 2, the algebraic system $\mathcal{A}$ has around $(t+1)(r-2)n + nk$ variables. Specifically, the

ALGORITHM 2: Main algebraic differential fault attack using differential trails

1) Randomly pick a plaintext P and feed into a SIMON encryption function to obtain its corresponding ciphertext C.
2) Let t be a small positive integer and fix r to be some specific value.
3) For $i = 1$ to $i = t$ do
   a) Randomly inject a bit flip at bit j_i and obtain the faulty ciphertext C_i.
   b) Construct the differential trail table D_i.
4) Construct the set of SIMON cipher equations (refer to Equation 3) with X^r as the plaintext and C as the cipher text. Denote this set by $\mathcal{S}$.
5) Let $\mathcal{D}_i$ comprise all the equations of the form $\delta_l^j(i) = D_i(j,l)$ where $D_i(j,l)$ denotes the (j,l)-entry in the differential trail table D_i, $i = 1, 2, \ldots, t$, with the unknown $\delta_l^j(i)$'s as the variables.
6) Let $\mathcal{L}$ be the set of linear equations for the key schedule.
7) Let $\mathcal{A} = \mathcal{S} \cup \bigcup_{i=1}^{t} \mathcal{D}_i \cup \mathcal{L}$.
8) Solve $\mathcal{A}$ using an algebraic solving tool.
9) If the key bits are uniquely found, then return the solved key bits.
10) Otherwise, for all the possible solutions for the key bits, test with the plaintext/ciphertext pair (P, C) to determine the correct key.

variables include the intermediate input variables x_l^j, the difference variables $\delta_l^j(i)$, and the nk key variables for $i = 1, \ldots, t$, $j = r, \ldots, T-2$, $l = 0, 1, \ldots, n-1$. To solve an algebraic system comprising quadratic equations in so many variables is typically a challenging task. However, we note from the differential trail table that a number of the $\delta_l^j(i)$'s are either known or are linear expressions of some of the intermediate input variables. Moreover, the following lemma shows that one can obtain a number of equations involving only the key bits.

Lemma 4.1. For $j = 0, 1, \ldots, n-1$, we have

- $\delta_j^{T-3} = l_j(K^{T-1})$, where l_j is a function of the bits in K^{T-1} with degree smaller or equal to 1 (i.e., $\deg(l_j) \leq 1$).
- $\delta_j^{T-4} = q_j(K^{T-1}, K^{T-2})$, where q_j is a function of the bits in K^{T-1} and K^{T-2} with $\deg(q_j) \leq 2$.

Proof: In the following proof, we shall use the fact that $x^T, x^{T-1}, \Delta^T, \Delta^{T-1}, \Delta^{T-2}$ are known values that can be deduced from the ciphertexts and thus treated as constants.

- For all $j = 0, 1, \ldots, n-1$, it follows from Equation (3) that $x_j^{T-2} = x_{j-1}^{T-1} x_{j-8}^{T-1} + x_{j-2}^{T-1} + x_j^T + k_j^{T-1} = k_j^{T-1} + c_j$ where c_j is a constant. Substituting into Equation (4) then yields:

$$\begin{aligned}\delta_j^{T-3} =& \delta_{j-1}^{T-2}(k_{j-8}^{T-1} + c_{j-8}) + \delta_{j-8}^{T-2}(k_{j-1}^{T-1} + c_{j-1}) \\ &+ \delta_{j-1}^{T-2}\delta_{j-8}^{T-2} + \delta_{j-2}^{T-2} + \delta_j^{T-1} \\ =& \delta_{j-1}^{T-2} k_{j-8}^{T-1} + \delta_{j-8}^{T-2} k_{j-1}^{T-1} + b_j,\end{aligned}$$

where b_j is a known constant.

- Similarly, one can show that $x_j^{T-3} = k_{j-1}^{T-1} k_{j-8}^{T-1} + L_j(k_{j-1}^{T-1}, k_{j-2}^{T-1}, k_{j-8}^{T-1}, k_j^{T-2})$, where $\deg(L_j) \leq 1$. Hence,

$$\begin{aligned}\delta_j^{T-4} =& (k_{j-2}^{T-1} k_{j-9}^{T-1} + L_{j-1})(\delta_j^{T-2} k_{j-9}^{T-1} + \delta_{j-9}^{T-2} k_j^{T-1} + b_{j-8}) \\ &+ (k_{j-9}^{T-1} k_j^{T-1} + L_{j-8})(\delta_{j-2}^{T-2} k_{j-9}^{T-1} + \delta_{j-9}^{T-2} k_{j-2}^{T-1} + b_{j-1}) \\ &+ q_j \\ =& q_j',\end{aligned}$$

where both q_j and q_j' have degrees at most 2.

□

The following table summarizes the structure of Δ^{T-i} for $i = 0, 1, \ldots, 4$.

TABLE 5
Differences of the left half of the last five rounds of SIMON-32/64 deduced from the bottom

Difference	Values
Δ^T	Known
Δ^{T-1}	Known
Δ^{T-2}	Known
Δ^{T-3}	Linear relations of the last round key bits
Δ^{T-4}	Quadratic relations of the key bits in the last two rounds

Assume that we flip bit 15 at round $T-6$. We have the following differential trail table for the last seven rounds.

By combining the two trails in Table 6 and Table 5, we obtain algebraic equations involving only key bits as shown in Table 7. In this table, the δ_j^i is a variable, l_i is a linear function of the last round key bits and q_i is a quadratic combination of the key bits in the last two rounds. For example, at bit 14 of the round $T-4$, from Table 6, we have $\delta_{14}^{T-4} = 0$. Suppose that from Table 5, δ_{14}^{T-4} could be represented by a quadratic expression of the last two round key bits, $q_{14}(k)$. This results in a quadratic equation $q_{14}(k) = 0$ that involve only key bits in the last two rounds.

In Table 7, some of the l_i may be constant and some of the q_i may have degree less than 2. Furthermore, this table also shows that one can find linear and quadratic equations involving the intermediate input variables x_j^i's and key variables. Consequently, all these equations help to simplify the set $\mathcal{A}$.

4.2 Simplified Gröbner basis algorithm to solve for the master key

A common tool to solve algebraic systems is via Gröbner basis algorithms [11], [12]. In fact, efficient variants of the algorithms have been implemented in many computational software including Magma [13]. Typically, Gröbner basis algorithms progressively constructs matrices for multiples of the input polynomials and simplifying the system until the system is solved. One problem encountered by Gröbner basis algorithms is that the size of the matrices involved increase very rapidly, thereby blowing up the memory consumption. This is particularly true for dense systems with many variables. The set $\mathcal{A}$ constructed in Algorithm 2 is somewhat sparse in the sense that one can find many equations involving just the key variables.

Indeed, from Table 7, we see that one can obtain up to 7 linear equations in the last round key bits with one fault

TABLE 6
Differential trail for the left half of the cipher when a fault injected into round $r = T - 6$.

Bit	15	14	13	12	11	10	9	8
Δ^{T-7}	0	0	0	0	0	0	0	0
Δ^{T-6}	1	0	0	0	0	0	0	0
Δ^{T-5}	0	0	0	0	0	0	0	0
Δ^{T-4}	$*$	0	0	0	0	0	$x_6^{T-6} + x_8^{T-5}$	$*$
Δ^{T-3}	0	0	0	0	$x_6^{T-6} + x_8^{T-5} + x_{10}^{T-4}$	$*$	$*$	0
Δ^{T-2}	$*$	0	$x_6^{T-6} + x_8^{T-5} + x_{10}^{T-4} + x_{12}^{T-3}$	$*$	$*$	$*$	$*$	$*$
Δ^{T-1}	$x_6^{T-6} + x_8^{T-5} + x_{10}^{T-4} + x_{12}^{T-3} + x_{14}^{T-2}$	$*$	$*$	$*$	$*$	$*$	$*$	$x_8^{T-6} + x_{10}^{T-5} + x_{12}^{T-4} + x_{14}^{T-3} + x_0^{T-2}$
Δ^{T}	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$

Bit	7	6	5	4	3	2	1	0
Δ^{T-7}	0	0	0	0	0	0	0	0
Δ^{T-6}	0	0	0	0	0	0	0	0
Δ^{T-5}	x_6^{T-6}	0	0	0	0	0	1	x_8^{T-6}
Δ^{T-4}	0	0	0	0	1	$x_8^{T-6} + x_{10}^{T-5}$	$*$	0
Δ^{T-3}	$*$	0	1	$x_8^{T-6} + x_{10}^{T-5} + x_{12}^{T-4}$	$*$	$*$	$*$	$*$
Δ^{T-2}	1	$x_8^{T-6} + x_{10}^{T-5} + x_{12}^{T-4} + x_{14}^{T-3}$	$*$	$*$	$*$	$*$	$*$	0
Δ^{T-1}	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$
Δ^{T}	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$

TABLE 7
Algebraic equations obtained when bit 15 at round $T - 6$ is flipped. Values 0 and 1 are constant differences.

Bit	15	14	13	12	11	10	9	8
$T-4$	$*$	$q_{14}(k)$	$q_{13}(k)$	$q_{12}(k)$	$q_{11}(k)$	$q_{10}(k)$	$x_6^{T-6} + x_8^{T-5} + q_9(k)$	$*$
$T-3$	$l_{15}(k)$	$l_{14}(k)$	$l_{13}(k)$	$l_{12}(k)$	$x_6^{T-6} + x_8^{T-5} + x_{10}^{T-4} + l_{11}(k)$	$*$	$*$	$l_8(k)$
$T-2$	$*$	0	$x_6^{T-6} + x_8^{T-5} + x_{10}^{T-4} + x_{12}^{T-3} + \delta_{13}^{T-2}$	$*$	$*$	$*$	$*$	$*$
$T-1$	$x_6^{T-6} + x_8^{T-5} + x_{10}^{T-4} + x_{12}^{T-3} + x_{14}^{T-2} + \delta_{15}^{T-1}$	$*$	$*$	$*$	$*$	$*$	$*$	$x_8^{T-6} + x_{10}^{T-5} + x_{12}^{T-4} + x_{14}^{T-3} + x_0^{T-2} + \delta_8^{T-1}$
T	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$

Bit	7	6	5	4	3	2	1	0
$T-4$	$q_7(k)$	$q_6(k)$	$q_5(k)$	$q_4(k)$	$q_3(k)+1$	$x_8^{T-6} + x_{10}^{T-5} + q_2(k)$	$*$	$q_0(k)$
$T-3$	$*$	$l_6(k)$	$l_5(k)+1$	$x_8^{T-6} + x_{10}^{T-5} + x_{12}^{T-4} + l_4(k)$	$*$	$*$	$*$	$*$
$T-2$	1	$x_8^{T-6} + x_{10}^{T-5} + x_{12}^{T-4} + x_{14}^{T-3} + \delta_6^{T-2}$	$*$	$*$	$*$	$*$	$*$	0
$T-1$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$
T	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$

injection. From Lemma 4.1, it follows that one may expect to obtain $7 \times 3/4 \approx 5$ linear equations in the last round key bits. As such, with $t \geq 4$, there are likely to be sufficient linear equations to solve for the last round key. The last round key can then be substituted into the polynomials in $\mathcal{A}$ to solve for other round key bits. For $t < 4$, one can still utilize these equations in the round key bits to determine some of the key bits and to simplify the system with these key bits as well as other linear equations in $\mathcal{A}$. This motivates us to introduce a "simplified Gröbner basis" approach to recover the full key.

Our algorithm is described in Algorithm 3.

4.3 Experimental Results

We implemented our attacks with Magma [13] on three members of SIMON family, namely, SIMON-32/64, SIMON-48/96 and SIMON-64/128, and with different number of bits flipped. For each attack, we generated 50 random instances and tried to solve them using Algorithm 3.

The executed timings were obtained by running Magma Version 22-3 on a personal laptop. For each instance, we recorded the number of key bits that were solved by the algorithm. The average timings and average number of key bits solved over 50 instances are shown in Table 8.

As shown in Table 8, we perform fault injections at the rounds $T - 5$, $T - 6$, or $T - 7$. The number of key bits flipped (i.e., the number of faults injected) is only from 3 to 5 for different members of SIMON family. As shown by the experimental results, all the experiments have fast solving times (less than one minute), and we are able to recover (almost) all the key variables.

Remark 5. In case the position of the bit flip is random, one

ALGORITHM 3: Algebraic Differential Fault Attacks Using a Simplified Gröbner Basis

1) Randomly select a plaintext P and obtain its ciphertext C under a fixed key K.
2) Construct the set $\mathcal{A}$ using Algorithm 2 for some suitable t and r.
3) Construct the matrix for the coefficients of the polynomials in $\mathcal{A}$ with respect to a degree-respecting monomial ordering and perform Gaussian elimination on the system. Let $\mathcal{L}$ be the set of all the linear equations in the corresponding set of polynomials.
4) While new linear polynomials can be found, perform the following:
5) For each $L = x + L_0 \in \mathcal{L}$ for some variable x, substitute x for L_0 in the remaining polynomials in $\mathcal{A}$. Perform Gaussian elimination to find new linear polynomials.
6) Let $\mathcal{A}'$ be the set of remaining polynomials. Construct the matrix for the coefficients of the polynomials in $\mathcal{A}'$ with respect to a monomial ordering where all monomials involving only the key variables are placed to the right. Perform Gaussian elimination to obtain a set $\mathcal{K}$ of equations that involve only the key variables.
7) Compute the Gröbner basis G of $\mathcal{K}$ using any Gröbner basis algorithm.
8) If all the key variables are found, return the solved key.
9) Otherwise, compute the Gröbner basis G' of $\mathcal{A}' \cup G$.
10) Return G'.

will need to guess the possible bit flip positions. This involves at most $\binom{n}{t}$ different choices. Note that one can eliminate some choices by comparing the differences in round $T-2$. In particular, if bit i is flipped, then in round $T-2$, one has bit $(i-1) \bmod 16 = 0$, bit $(i-8) \bmod 16 = 1$ and bit $(i-15) \bmod 16 = 0$.

5 Algebraic Differential Fault Attacks using SAT solvers

Section 4 showed that given known plaintext/ciphertexts, SIMON ciphers could be broken with only 3 to 5 faults by using a simplified Gröbner basis. In this section, we consider the case where attackers have a very limited ability to inject faults during the encryption computation, that is, we assume that the attacker can only inject one fault ($t = 1$ in Algorithm 2). This section presents an algebraic fault attack with a single fault by exploiting SAT solvers.

The satisfiability problem (or SAT for short) is one of the classical NP complete problems that seeks to determine if a Boolean formula has a satisfiability assignment. With its wide and varied range of applications, such as in automated testing and artificial intelligence, many efficient SAT solvers have been implemented to solve SAT problems involving large numbers of variables and constraints. Some well-known modern SAT solvers include Minisat [21], CryptoMiniSat [10] and MapleSAT [9].[1]

1. For more updates on SAT solvers and their performance, the reader may check out the websites SAT Live *http: //www.satlive.org* and SATLIB *http: //www.satlib.org*.

5.1 SAT solver algorithm to solve for the master key

Observe that when $t = 1$, the set $\mathcal{A}$ constructed in Algorithm 2 reduces the problem of finding two possible plaintexts P and P' with input difference $(1, 0, \ldots, 0)$ and corresponding ciphertexts C and C' for $T - r$ rounds of the SIMON cipher. Since the key size is twice the block size, such a system is likely to admit 2^{2n} solutions on average. In order to obtain a unique solution, one can add in all the cipher equations (Equation 3) for the first r rounds, that is, starting from the plaintext P. We call this set $\mathcal{A}^*$. Thus, $\mathcal{A}^*$ will involve another $(r-1)n$ extra quadratic equations variables compared to $\mathcal{A}$. Typically, the number of variables will be too large for Gröbner basis algorithms to handle.

As such, we propose using SAT solvers to solve the system $\mathcal{A}^*$ as SAT solvers can better handle large numbers of variables. From our experiments, solving $\mathcal{A}^*$ directly does not yield any result within reasonable time. We thus propose to fix some key bits in order to solve the resulting system more quickly. This leads to the following algorithm.

ALGORITHM 4: Algebraic Differential Fault Attacks Using SAT Solvers

1) Randomly select a plaintext P and obtain its ciphertext C under a fixed key K.
2) Construct the set $\mathcal{A}$ using Algorithm 2 for $t = 1$ and for some suitable r.
3) Construct the set $\mathcal{A}^*$ as the union of $\mathcal{A}$ and all the cipher equations for the first r rounds.
4) Construct the matrix for the coefficients of the polynomials in $\mathcal{A}$ with respect to a degree-respecting monomial ordering and perform Gaussian elimination on the system. Let $\mathcal{L}$ be the set of all the linear equations in the corresponding set of polynomials.
5) While new linear polynomials can be found, perform the following:
6) For each $L = x + L_0 \in \mathcal{L}$ for some variable x, substitute x for L_0 in the remaining polynomials in $\mathcal{A}$. Perform Gaussian elimination to find new linear polynomials. Add the new polynomials to $\mathcal{L}$.
7) Choose k key bits that occur with the highest frequency in the set $\mathcal{A}^*$.
8) For each guess of the k key bits, simplify the system $\mathcal{A}^*$ with the values and let the resulting set be $\widehat{\mathcal{A}^*}$.
9) Convert the equations in $\widehat{\mathcal{A}^*} \cup \mathcal{L}$ into CNF and feed the CNF clauses into a modern SAT solver.
10) Return the key variables found.

Remark 6.

- In our algorithm, we include a step to find linear polynomials in order to find more relations for the SAT solver. Such a preprocessing step has been shown to be helpful.
- Here, we choose the key bits with the highest frequencies to try to simplify the system as much as possible.

5.2 Experimental Results

Based on the above algorithm, we performed experiments on SIMON-32/64, SIMON-48/72 and SIMON-48/96. After generating the ANF equations, we use SAGE to convert them into CNF clauses [23]. The timings were obtained by

TABLE 8
Number of key bits found and the corresponding timings. Each round key bit is represented by a variable. The total number of key variables equals to $T \times n$, where T is the total number of rounds and n is half of block size.

Cipher	Round	Total no of key variables	No of faults	Average No of key variables found	Timing (s)
SIMON-32/64	$T-5$	512	4	508.38	2.6
SIMON-32/64	$T-5$	512	5	511.46	0.7
SIMON-32/64	$T-6$	512	3	511.8	35.3
SIMON-32/64	$T-6$	512	4	511.9	2
SIMON-48/72	$T-6$	864	4	864	26
SIMON-48/72	$T-6$	864	5	864	8.5
SIMON-48/96	$T-6$	864	4	864	5.3
SIMON-48/96	$T-6$	864	5	864	4.1
SIMON-64/128	$T-6$	1048	5	1046	34.3
SIMON-64/128	$T-7$	1048	5	1048	28.8

running CryptoMiniSat 5.0.1 [10] on a 3.6GHz Intel Core i7 CPU with 8GB RAM. Note that the timings take account into the conversion of ANF equations into CNF clauses.

Table 9 presents the average timings in seconds obtained for different members of SIMON family and with different number of key bits fixed. For each experiment, we generated 50 random instances by injecting a fault into the round $T-6$ and recorded the time to solve the system. We fix the maximum time to solve an instance to be 10 minutes. The final column in the following table records the average timing for all the instances that can be solved within given maximum time.

TABLE 9
Number of instances solved out of 50 and corresponding executed timings.

Cipher	No of faults	No of key bits fixed	Instances solved	Timing (s)
SIMON-32/64	1	18	34	99.1
SIMON-32/64	1	20	42	69.4
SIMON-32/64	1	22	46	47.4
SIMON-48/72	1	22	36	41.2
SIMON-48/72	1	24	42	31.9
SIMON-48/72	1	26	45	37
SIMON-48/96	1	40	22	77.1
SIMON-48/96	1	42	30	103.7
SIMON-48/96	1	44	34	72.4

6 Conclusion

In this paper, we provided an improved differential trail for fault injection attack on the SIMON cipher. By analyzing this differential trail, we obtain three differential fault attacks on the SIMON cipher. The first one is analytic while the last two are algebraic, and they show a trade-off between the time complexity and the number of fault injections needed to launch the attack. In the first attack, we study the differential trail analytically and achieved an improved attack over the previous differential fault attacks [17], [22]. Specifically, our attack could manage to retrieve the entire master key of all members of SIMON family by injecting faults into a single round of the cipher. In the second attack, we apply Gröbner basis to the differential path to solve for SIMON-32/64, SIMON-48/72, SIMON-48/96, and SIMON-64/128. This attack reduces the number of fault injections needed to between 3 and 5 and also require negligible computation. The final attack applies SAT solver to the differential path to solve for the key of SIMON-32/64, SIMON-48/72, and SIMON-48/96 with just one single fault injection and some key bits guesses. In the last two algebraic attacks, we performed experiments to demonstrate our attack strategies.

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*, 2013. [Online]. Available: https://eprint.iacr.org/2013/404

[2] O. Kömmerling and M. G. Kuhn, "Design Principles for Tamper-resistant Smartcard Processors," in *Proceedings of the USENIX Workshop on Smartcard Technology on USENIX Workshop on Smartcard Technology*, ser. WOST'99, 1999, pp. 2–2.

[3] S. P. Skorobogatov and R. J. Anderson, "Optical fault induction attacks," in *Revised Papers from the 4th International Workshop on Cryptographic Hardware and Embedded Systems*, ser. CHES '02, 2003, pp. 2–12.

[4] D. Boneh, R. A. DeMillo, and R. J. Lipton, "On the Importance of Checking Cryptographic Protocols for Faults," in *Proceedings of the 16th Annual International Conference on Theory and Application of Cryptographic Techniques*, ser. EUROCRYPT'97, 1997, pp. 37–51.

[5] E. Biham and A. Shamir, "Differential Fault Analysis of Secret Key Cryptosystems," in *Proceedings of the 17th Annual International Cryptology Conference on Advances in Cryptology*, ser. CRYPTO '97, 1997, pp. 513–525.

[6] ——, "Differential Cryptanalysis of DES-like Cryptosystems," in *Proceedings of the 10th Annual International Cryptology Conference on Advances in Cryptology*, ser. CRYPTO '90, 1991, pp. 2–21.

[7] M. Matsui, "Linear Cryptanalysis Method for DES Cipher," in *Workshop on the Theory and Application of Cryptographic Techniques on Advances in Cryptology*, ser. EUROCRYPT '93, 1994, pp. 386–397.

[8] N. Courtois, A. Klimov, J. Patarin, and A. Shamir, "Efficient algorithms for solving overdefined systems of multivariate polynomial equations," in *Advances in Cryptology–EUROCRYPT 2000*. Springer, 2000, pp. 392–407.

[9] V. Ganesh and J. H. Liang, "MapleSAT," https://sites.google.com/a/gsd.uwaterloo.ca/maplesat/, accessed: 2017-11-28.

[10] M. Soos, K. Nohl, and C. Castelluccia, "Extending SAT Solvers to Cryptographic Problems," in *SAT*, ser. Lecture Notes in Computer Science, O. Kullmann, Ed., vol. 5584. Springer, 2009, pp. 244–257.

[11] J.-C. Faugere, "A new efficient algorithm for computing Gröbner bases (F 4)," *Journal of pure and applied algebra*, vol. 139, no. 1, pp. 61–88, 1999.

[12] ——, "A new efficient algorithm for computing Gröbner bases without reduction to zero (F 5)," in *Proceedings of ISSAC*. ACM, 2002, pp. 75–83.

[13] W. Bosma, J. Cannon, and C. Playoust, "The Magma algebra system. I. The user language," *J. Symbolic Comput.*, vol. 24, no. 3-4, pp. 235–265, 1997, computational algebra and number theory (London, 1993). [Online]. Available: http://dx.doi.org/10.1006/jsco.1996.0125

[14] C. Giraud, "DFA on AES," in *Proceedings of the 4th International Conference on Advanced Encryption Standard*, ser. AES'04, 2005, pp. 27–41.

[15] L. Hemme, "A Differential Fault Attack Against Early Rounds of (Triple-)DES," in *Cryptographic Hardware and Embedded Systems - CHES 2004*, M. Joye and J.-J. Quisquater, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 254–267.

[16] C. Clavier, B. Gierlichs, and I. Verbauwhede, "Fault Analysis Study of IDEA," in *Topics in Cryptology – CT-RSA 2008*, T. Malkin, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 274–287.

[17] H. Tupsamudre, S. Bisht, and D. Mukhopadhyay, "Differential Fault Analysis on the Families of SIMON and SPECK Ciphers," in *2014 Workshop on Fault Diagnosis and Tolerance in Cryptography*, Sept 2014, pp. 40–48.

[18] D. Samyde, R. Anderson, S. Skorobogatov, and J. Quisquater, "On a new way to read data from memory," in *Proceedings First International IEEE Security in Storage Workshop(SISW)*, vol. 00, 2002, p. 65.

[19] J. Blömer, M. Otto, and J.-P. Seifert, "A New CRT-RSA Algorithm Secure Against Bellcore Attacks," in *Proceedings of the 10th ACM Conference on Computer and Communications Security*, ser. CCS '03, 2003, pp. 311–320.

[20] N. Courtois, K. Jackson, and D. Ware, "Fault-Algebraic Attacks on Inner Rounds of DES," in *The eSmart 2010 European Smart Card Security Conference*, Sept 2010.

[21] N. Een and N. Sorensson, "An extensible sat-solver," in *SAT*, ser. Lecture Notes in Computer Science, E. Giunchiglia and A. Tacchella, Eds., vol. 2919. Springer, 2003, pp. 502–518.

[22] J. del Carmen Grados Vasquez, F. Borges, R. Portugal, and P. Lara, "An Efficient One-Bit Model for Differential Fault Analysis on SIMON Family," in *2015 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC)*, Sept 2015, pp. 61–70.

[23] Sage, "An ANF to CNF Converter using a Dense/Sparse Strategy," http://doc.sagemath.org/html/en/reference/sat/sage/sat/converters/polybori.html, accessed: 2017-11-28.

APPENDIX

Let $\Delta^i = X^i + X'^i = \{\delta^i_j\}_{0\le i\le T, 0\le j\le n-1}$ denote the difference of intermediate correct and faulty inputs at the round i. Without loss of generality, we suppose that a fault will be injected into the bit $n-1$. Since the differential trails depend only on block size n, but not on the key word m, members of SIMON family having the same block size (e.g., SIMON-48/72 and SIMON-48/96), have the same differential trail. We list below differential trails for different members of SIMON family. Likewise, the notation $*$ denotes for a complex non-linear expression of variables that is non-trivial to deduce their values.

SIMON-48/XX

Δ^{T-5} : $(1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)$

Δ^{T-4} : $(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, x_6^{T-5}, 0, 0, 0, 0, 0, 1, x_{16}^{T-5})$

Δ^{T-3} : $(1, 0, 0, 0, 0, 0, 0, 0, 0, x_6^{T-5} * x_{14}^{T-4}, 0, 0, 0, 0, 0, x_8^{T-4} + x_6^{T-5}, x_6^{T-5} * x_0^{T-4} + x_7^{T-5} * x_7^{T-4} + x_6^{T-5} * x_{16}^{T-5}, 0, 0, 0, 0, 1, x_{18}^{T-4} + x_{16}^{T-5}, x_{16}^{T-5} * x_{17}^{T-4}, 0)$

Δ^{T-2} : $(*, 0, 0, 0, 0, 0, *, *, 0, 0, 0, 0, x_{10}^{T-3} + x_8^{T-4} + x_6^{T-5}, *, *, 0, x_6^{T-3} + x_6^{T-5}, 0, 1, x_{20}^{T-3} + x_{18}^{T-4} + x_{16}^{T-5}, *, *, 0, x_{16}^{T-3} + x_{16}^{T-5})$

Δ^{T-1} : $(1, 0, 0, 0, *, *, *, 0, *, 0, x_{12}^{T-2} + x_{10}^{T-3} + x_8^{T-4} + x_6^{T-5}, *, *, *, x_6^{T-3} + x_8^{T-4}, *, *, x_{22}^{T-2} + x_{20}^{T-3} + x_{18}^{T-4} + x_{16}^{T-5}, *, *, *, x_{16}^{T-3} + x_{18}^{T-4}, *, *)$

Δ^{T} : $(*, 0, *, x_{16}^{T-1} + x_{16}^{T-3} + x_{16}^{T-5})$

SIMON-64/XX

Δ^{T-5} : $(1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)$

Δ^{T-4} : $(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, x_6^{T-5}, 0, 0, 0, 0, 0, 1, x_{24}^{T-5})$

Δ^{T-3} : $(1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, x_6^{T-5} * x_{14}^{T-4}, 0, 0, 0, 0, 0, x_8^{T-4} + x_6^{T-5}, x_6^{T-5} * x_0^{T-4} + x_{24}^{T-5} * x_7^{T-4} + x_6^{T-5} * x_{24}^{T-5}, 0, 0, 0, 0, 1, x_{26}^{T-4} + x_{24}^{T-5}, x_{24}^{T-5} * x_{25}^{T-4}, 0,)$

Δ^{T-2} : $(0, 0, 0, 0, 0, 0, 0, 0, *, 0, 0, 0, 0, 0, *, *, 0, 0, 0, 0, 0, x_{10}^{T-3} + x_8^{T-4} + x_6^{T-5}, *, *, 0, x_6^{T-3} + x_6^{T-5}, 0, 1, x_{28}^{T-3} + x_{26}^{T-4} + x_{24}^{T-5}, *, *, 0, x_{24}^{T-3} + x_{24}^{T-5})$

Δ^{T-1} : $(*, 0, 0, 0, 0, 0, *, *, 0, 0, 0, 0, *, *, *, 0, *, 0, x_{12}^{T-2} + x_{12}^{T-3} + x_{10}^{T-3} + x_8^{T-4} + x_6^{T-5}, *, *, *, x_6^{T-3} + x_8^{T-4}, *, *, 1, x_{30}^{T-2} + x_{28}^{T-3} + x_{26}^{T-4} + x_{24}^{T-5}, *, *, *, x_{24}^{T-3} + x_{26}^{T-4}, *, 0)$

Δ^{T} : $(0, 0, 0, 0, *, *, *, 0, *, 0, *, *, *, *, *, *, *, x_{14}^{T-1} + x_{12}^{T-2} + x_{12}^{T-3} + x_8^{T-4} + x_6^{T-5}, *, *, *, *, *, *, *, *, x_0^{T-1} + x_{30}^{T-2} + x_{28}^{T-3} + x_{26}^{T-4} + x_{24}^{T-5}, *, *, *, *, *, *, *, *, *)$

SIMON-96/XX

Δ^{T-5} : $(1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)$

Δ^{T-4} : $(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, x_6^{T-5}, 0, 0, 0, 0, 0, 1, x_{40}^{T-5})$

Δ^{T-3} : $(1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, x_6^{T-5} * x_{14}^{T-4}, 0, 0, 0, 0, 0, 0, x_8^{T-4} + x_6^{T-5}, x_6^{T-5} * x_0^{T-4} + x_{40}^{T-5} * x_7^{T-4} + x_6^{T-5} * x_{40}^{T-5}, 0, 0, 0, 0, 1, x_{42}^{T-4} + x_{40}^{T-5}, x_{40}^{T-5} * x_{41}^{T-4}, 0)$

Δ^{T-2} : $(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, *, 0, 0, 0, 0, 0, *, *, 0, 0, 0, 0, x_{10}^{T-3} + x_8^{T-4} + x_6^{T-5}, *, *, 0, x_6^{T-3} + x_6^{T-5}, 0, 1, x_{44}^{T-3} + x_{42}^{T-4} + x_{40}^{T-5}, *, *, 0, x_{40}^{T-3} + x_{40}^{T-5})$

Δ^{T-1} : $(1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, *, 0, 0, 0, 0, 0, *, *, 0, 0, 0, 0, *, *, *, 0, 0, x_{12}^{T-2} + x_{10}^{T-3} + x_8^{T-4} + x_6^{T-5}, *, *, *, x_6^{T-3} + x_8^{T-4}, *, 1, x_{46}^{T-2} + x_{44}^{T-3} + x_{42}^{T-4} + x_{40}^{T-5}, *, *, *, x_{40}^{T-3} + x_{42}^{T-4}, *, 0)$

Δ^{T} : $(0, 0, 0, 0, 0, 0, 0, 0, 0, *, 0, 0, 0, 0, 0, *, *, 0, 0, 0, 0, 0, *, *, *, 0, *, 0, *, *, *, *, *, *, *, x_{14}^{T-1} + x_{12}^{T-2} + x_{10}^{T-3} + x_8^{T-4} + x_6^{T-5}, *, *, *, *, *, *, *, x_0^{T-1} + x_{46}^{T-2} + x_{44}^{T-3} + x_{42}^{T-4} + x_{40}^{T-5}, *, *, *, *, *, *, *, 1, x_{40}^{T-1} + x_{40}^{T-3} + x_{40}^{T-5})$

Le Duc Phong received his PhD degree in Computer Science from University of Pau et des Pays de l'Adour, in 2009. Currently, he is working as a researcher at Canadian Institute for Cybersecurity (CIC), UNB, Canada. He used to work as a scientist at National University of Singapore (NUS) and Institute for Infocomm Research (I2R), Singapore. His research interests include the algorithmic aspects of cryptology, cybersecurity, and blockchain.

Sze Ling Yeo received her Ph.D in Mathematics from the National University of Singapore in 2006. She is currently a research scientist at Institute for Infocomm Research, A STAR and has a concurrent appointment as an adjunct Associate Professor at the Nanyang Technological University, Singapore. Her research interests are in the design and cryptanalysis of crypto primitives.

$\Delta^{T-5}:$ $(1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,$
$0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,$
$0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)$

$\Delta^{T-4}:$ $(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,$
$0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,$
$0, 0, 0, 0, 0, 0, x_6^{T-5}, 0, 0, 0, 0, 0, 1, x_{56}^{T-5})$

$\Delta^{T-3}:$ $(1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,$
$0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,$
$x_6^{T-5} * x_{14}^{T-4}, 0, 0, 0, 0, 0, x_8^{T-4} + x_6^{T-5}, x_6^{T-5} * x_0^{T-4} +$
$x_{56}^{T-5} * x_7^{T-4} + x_6^{T-5} * x_{56}^{T-5}, 0, 0, 0, 0, 1, x_{58}^{T-4} + x_{56}^{T-5},$
$x_{56}^{T-5} * x_{57}^{T-4}, 0)$

$\Delta^{T-2}:$ $(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,$
$0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, *, 0, 0, 0, 0, 0, *, *, 0,$
$0, 0, 0, x_{10}^{T-3} + x_8^{T-4} + x_6^{T-5}, *, *, 0, x_6^{T-3} + x_6^{T-5}, 0, 1,$
$x_{60}^{T-3} + x_{58}^{T-4} + x_{56}^{T-5}, *, *, 0, x_{56}^{T-3} + x_{56}^{T-5})$

$\Delta^{T-1}:$ $(1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,$
$0, 0, 0, 0, 0, 0, 0, *, 0, 0, 0, 0, 0, *, *, 0, 0, 0, 0, *, *, *, 0, *, 0,$
$x_{12}^{T-2} + x_{10}^{T-3} + x_8^{T-4} + x_6^{T-5}, *, *, *, x_6^{T-3} + x_8^{T-4}, *, 1,$
$x_{62}^{T-2} + x_{60}^{T-3} + x_{58}^{T-4} + x_{56}^{T-5}, *, *, *, x_{56}^{T-3} + x_{58}^{T-4}, *, 0)$

$\Delta^{T}:$ $(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,$
$0, *, 0, 0, 0, 0, 0, *, *, 0, 0, 0, 0, *, *, *, 0, *, 0, *, *, *, *, *,$
$*, x_{14}^{T-1} + x_{12}^{T-2} + x_{10}^{T-3} + x_8^{T-4} + x_6^{T-5}, *, *, *, *, *, *,$
$x_0^{T-1} + x_{62}^{T-2} + x_{60}^{T-3} + x_{58}^{T-4} + x_{56}^{T-5}, *, *, *, *, *, *, 1,$
$x_{56}^{T-1} + x_{56}^{T-3} + x_{56}^{T-5})$

Khoongming Khoo received his PhD in Combinatorics and Optimization from the University of Waterloo in 2003. He is currently a researcher at DSO National Laboratories and an Adjunct Associate Professor at Nanyang Technological University (NTU). His research interest is in the design and analysis of cryptographic primitives.