The Insecurity of SHA2 under the Differential Fault Characteristic of Boolean Functions

No Author Given

No Institute Given

Abstract. SHA2 has been widely adopted across various traditional public-key cryptosystems, post-quantum cryptography, personal identification, and network communication protocols, etc. Hence, ensuring the robust security of SHA2 is of critical importance. There have been several differential fault attacks based on random word faults targeting SHA1 and SHACAL-2. However, extending such random word-based fault attacks to SHA2 proves significantly more difficult due to the heightened complexity of the boolean functions in SHA2.
In this paper, assuming random word faults, we find some distinctive differential properties within the boolean functions in SHA2. Leveraging these findings, we propose a new differential fault attack methodology that can be effectively utilized to recover the final message block and its corresponding initial vector in SHA2, forge HMAC-SHA2 messages, extract the key of SHACAL-2, and extend our analysis to similar algorithm like SM3. We validate the effectiveness of these attacks through rigorous simulations and theoretical deductions, revealing that they indeed pose substantial threats to the security of SHA2. In our simulation-based experiments, our approach necessitates guessing T bits within a register, with T being no more than 5 at most, and having an approximate 95% (for SHA512) probability of guessing just 1 bit. Moreover, upon implementing a consecutive series of 15 fault injections, the success probability for recovering one register (excluding the guessed bits) approaches 100%. Ultimately, approximately 928 faulty outputs based on random word faults are required to carry out the attack successfully.

Keywords: SHA2, Differential Fault Attack, HMAC, SM3, Compression Function

1 Introduction

1.1 Background

SHA2 (Secure Hash Algorithm 2) family [1], serving as an advanced successor to SHA1 algorithm, was officially introduced by the National Institute of Standards and Technology (NIST) in the year 2001. This cryptographic suite encompasses four variants: SHA224, SHA256, SHA384, and SHA512, each sharing a consistent structural design in their compression function while differing in terms of data block length. The compression function of SHA2 is a core component within SHACAL block cipher algorithm [2] and HMAC (Hash-based Message Authentication Code) [3], wherein it integrates secret keys into the message content for enhanced security. SHA2 has been widely adopted across various public-key cryptosystems, personal identification and network communication protocols, etc. Notably, SHA256 and SHA512 have been prominently adopted as key derivation functions across both traditional public-key cryptography schemes and emerging post-quantum cryptography protocols. This application extends to well-known algorithms such as ECDSA, EdDSA, SM2, and to cutting-edge post-quantum algorithms like Dilithium and Kyber, etc. Furthermore, SHA2 algorithms have become a staple selection for authentication mechanisms within an array of secure communication protocols. These include but are not limited to TLS/SSL, IPSEC, SSH (Secure Shell), PGP, PIN verification, OTP (One-Time Password) systems, and others. This widespread adoption underscores the pervasive role that SHA2 plays in ensuring data integrity and authenticity across diverse digital domains.

The robustness of hash algorithms, which hinges on the inherent properties of their compression functions and resilience to a multitude of attack strategies including collision attacks [4] and differential attacks [5] is paramount for security. Despite the lack of publicly disclosed design intricacies, SHA2 is widely perceived as more impervious to various attacks such as birthday attacks [6], differential attacks [**?**], in comparison with MD5 and SHA1 algorithms. However, the computational complexity often renders many of these theoretical attacks impractical in real-world scenarios. As an alternative, over the past decade or so, a significant body of literature has shifted towards researching the implementation security of hash algorithm, primarily focusing on side channel attacks and fault attacks.

Side channel attacks on various implementations of hash algorithm, especially on HMAC, have been continuously discovered during the last dozen years. McEvoy et al. proposed a differential power analysis (DPA) on HMAC-SHA2 [8], which employs the leakage information of the calculation of the modular *ADDs* and *XORs* to reveal the secret initial state in HMAC-SHA2 and thereby can mount a forgery attack. Meanwhile, Fouque etc. proposed a template attack on HMAC [9], which can recover the secret key k by measuring a single execution of HMAC-SHA1 incorporating k. These attacks are realistic and have been demonstrated many times in real products [10,11,12]. Furthermore, as another category of physical attacks, fault attacks [13,14,15] have also been demonstrated to be effective against hash algorithms. Generally, adversary manages to disturb the registers and the corresponding boolean functions in the compression function (by means of voltage clitches, laser or electro-magnetic injection and so on) to output faulty results, and exploits them to do input recovery.

1.2 Previous Works

The known fault attacks against hash algorithms and their derivatives predominantly employ difference-based methodologies, which encompass differential fault attacks (DFA) and algebraic fault attacks (AFA). These attack strategies typically assume transient faults as the underlying fault models.

The first fault attack (i.e., DFA) based on 32-bit random word fault model, was introduced in FDTC 2009 [13], which utilizes the arithmetical differential relation between the faulty and correct boolean function $f(x, y, z) = x \oplus y \oplus z$ (with same message) to recover the key in SHACAL-1 manually. It is noteworthy that, due to the attack which relies on the adversary's knowledge of the final round output rH (which is distinct from the output H of SHA1) in SHACAL-1 where the final addition operation is absent, such an attack cannot be applied to the hash algorithms with the final addition step. Therefore, as an improvement, Hemme and Hoffmann have combined and extended the methodology proposed in FDTC 2011 [14], introducing a novel approach that specifically targets the cyclic shifts in the compression function of SHA1 to enable the recovery of the input of compression function in SHA1. However, in view of the more complex boolean functions, the DFA in [13,14] can not be expanded to SHA2. Similarly, there exist analogous DFAs based on random register word faults for SHACAL-2 as well [16,17]. Such attacks not only require knowledge of the correct and faulty output of the last round in SHACAL-2 (i.e., knowing every boolean differential value and lacking initial vector), but also fundamentally involve searching for the key through leveraging boolean and arithmetic difference pairs. These are unfeasible in SHA2. In addition, Bagheri et al. first introduced a DFA against SHA3 at INDOCRYPT 2015 [18], where with 80 faulty outputs, it was possible to recover 1592 bits (out of the total 1600 bits) of the internal state. Subsequently, in FDTC 2016, Luo et al. [19] improved this method by extending the random bit fault model to a random byte fault model. However, due to the fundamentally distinct message compression structure in SHA2 as opposed to the sponge construction in SHA3, the DFA techniques developed for SHA3 are not applicable to the analysis of SHA2. To the best of our knowledge, the fault attacks [15,20] based on round counter fault appear to be the only currently feasible fault attacks on SHA2. Such attacks attempt to solve equations in order to recover message blocks by reducing the number of rounds or bypassing the comparison between round counter and total number of rounds, effectively decreasing the number of rounds calculated in the compression function to fewer than 16. However, these require extremely high accuracy in the term of time and position for fault injection. Furthermore, a low-cost measure such as verifying the round counter can effectively hinder such attacks.

After that, scholars have turned their attention to researching the more practical and feasible fault attacks, i.e., AFA. AFA do not require manual analysis of differential paths within hash algorithms; instead, they derive algebraic equations based on both the faulty and correct results, feeding these into tools such as SAT solvers to reveal the message or initial state of the compression function. Thus, its outcomes are empirically derived rather than grounded in theoretical deduction. The AFAs presented in [21,22] are capable of forging the initial state of HMAC-SHA2 without being able to actually recover the key in HMAC-SHA2. Luo et al. in DATE 2017 [23] first proposed an AFA on SHA3, and later in [24] expanded upon the fault models employed. In general, since AFA lacks theoretical foundation for the differential relationships within the compression function, it often serves as an alternative to DFA, becoming advantageous when specific differential paths cannot be manually derived by DFA.

In summary, as of the current understanding, there is no reported DFA specifically tailored to exploit vulnerabilities within the compression function of SHA2, especially considering its final addition step. Consequently, the security of SHA2 under random word fault model remains an open question and necessitates further investigation.

1.3 Our Contributions

In this paper, we discovered and established several unique differential properties of the boolean functions ($Ch(x, y, z) = (x \wedge y) \oplus (x\prime \wedge y)$ and $Maj(x, y, z) = (a \wedge b) \oplus (a \wedge c) \oplus (c \wedge b)$) within the compression function of SHA2. By utilizing these properties, we proposed a novel differential fault attack strategy targeting the compression function of SHA2 that is predicated on the random word fault model. We successfully executed such attack on SHA2, HMAC-SHA2, SHACAL-2 and SM3 [25] (holding similar boolean functions), showcasing its efficacy.

The principal advantages of our proposed differential approach can be summarized as follows:

– The proposed approach takes advantage of previously undiscovered differential properties of boolean functions and has a lower computational complexity. Generally, the boolean functions $Ch(x, y, z)$ and $Maj(x, y, z)$, which are integral components of the SHA2 hash function, have historically been regarded as challenging for differential analysis due to their inherently complex operational behavior. Specifically, the presence of the final addition step in SHA2, which incorporates an unknown initial vector, has commonly been thought to directly hinder the effectiveness of differential analysis. However, this paper presents some novel transformations of the boolean functions, which mitigate the confused impact of the faulty registers serving as inputs to boolean functions. This innovative approach enables the discovery of boolean differential properties between fault-induced outputs and original inputs, facilitating the recovery of state registers within compression function with an overwhelming probability. A significant distinction from previous DFAs on SHA1 is the reduced uncertainty in our methodology: In contrast to the DFAs on SHA1 that might require guessing up to 16 bits, our innovative technique reduces the uncertainty to just T bits, where $1 \leq T \leq 5$, and with a 95% probability, T is equal to 1 when recovering both the final initial vector and message block. Comprehensive details are provided in Section 3.
– The developed methodology is highly versatile and can be broadly applied to various hash functions and cryptographic algorithms, not limited to SHA2 (including SHA224/256/384/512), HMAC-SHA2, SHACAL-2, SM3, etc. We have successfully managed to recover the final message block and its corresponding initial vector in SHA2. Furthermore, we have demonstrated the

capability to forge messages for HMAC-SHA2 and recover the secret key used in SHACAL-2. Particularly, we have proved that the boolean functions employed within SM3 [25] are indeed functionally equivalent to $Ch(x, y, z)$ and $Maj(x, y, z)$. These findings further underscore the broad applicability of our differential fault attack methodology across diverse cryptographic constructs. See Section 4 for more details.

– Such attacks exhibit an impressively high success probability, which holds considerable practical significance. We have thoroughly substantiated this through comprehensive theoretical analysis and experimental validation. The experimental results demonstrate that when 15 random word faults are introduced into a specific register, the success rate for correctly restoring the corresponding register in the right half of the last round input within SHA2 approaches nearly 100%. Correspondingly, targeting the left half of the last round input, the attack achieves a almost 100% probability to successfully recover $n-T$ bits of every register, where n is the bit length of word register. See Section 5 for more details.

The comparative analysis of our attacks against previous differential fault attacks [13,14,16,17,15,20] is depicted in Table 1, where "Number of fault injection" refers to the minimum number of fault injections required to achieve a success probability of at least 99% of attack (excluding the guessing step).

Table 1. Comparison with previous attack

Attacks	Fault model	Complexity of guessing	Number of fault injection	Algorithms of analysis
Our attack	Random word fault	$2^T (T \approx 1)$	928	SHA2, HMAC-SHA2, SHACAL-2 and SM3
[13]	Random word fault	2^{16}	120	SHACAL-1
[14]	Random word fault	2^{16}	1000	SHA1, SHACAL-1
[16,17]	Random word fault	–	128/240	SHACAL-2
[15,20]	Round counter fault	–	16	SHA2, SHACAL-2

The structure of the remainder of this paper is as follows: In Section 2, we present the fundamental theory of boolean operations and the architecture of both SHA2 and HMAC-SHA2. Section 3 delves into the core concepts of the differential theory applied to boolean functions along with the associated proofs. Section 4 provides concrete examples of differential attacks on SHA2, HMAC-SHA2, SHACAL-2, and SM3. Finally, in Section 5, we detail the experimental aspects that validate the effectiveness of the proposed key recovery method.

2 Preliminaries

2.1 Notations

Throughout the paper, we define the notations listed in Table 2.

Table 2. Notations

Symbols	Definition
n	The bit length of one word
N	The times of fault injection
T	The number of bits required to guess when recovering one register
$\mathbb{Z}_2^n$	The set of integers with bit length n
$\oplus$	Exclusive-OR (XOR)
$\cdot$ or &	Logical bitwise AND, i.e., boolean AND
$'$	Logical bitwise NOT
$+$	Addition modulo 2^t when the bit length of operators equals to n; Logical bitwise OR When the bit length of operators equals to 1
$\vee$	32-bit logical OR
xy	Abbreviation of $x \cdot y$. Higher priority than XOR and OR
x_i	The i-th bit value of n-bit x
$\sum\limits_{j=0}^{N-1} x^j$	$x^0 + \ldots + x^{N-1}$. When $x^j \in \mathbb{Z}_2$, equals to the sum of logical OR
$-$	Subtraction modulo 2^n
$\ggg t$	Circular right shift by t positions
$x \parallel y$	Concatenation of two operators x and y
$Ch(x, y, z)$	$(x\&y) \oplus (x'\&z)$, where $x, y, z \in \mathbb{Z}_2^n$.
$Maj(x, y, z)$	$(x\&y) \oplus (x\&z) \oplus (y\&z)$, where $x, y, z \in \mathbb{Z}_2^n$
$\sum_0(x)$, $\sum_1(x)$, $\sigma_0(x)$, $\sigma_1(x)$	$(x \ggg i) \oplus (x \ggg j) \oplus (x \ggg k)$, where $x \in \mathbb{Z}_2^n$ is n-bit operator, and i, j and k depend on n
$H(M, IV)$	Hash function with message M and initial vector IV
$F(V_i, M^i)$	The i-th compression function in $H(M, IV)$ with initial vector V_i and message block M^i
r	The number of message blocks
R	The round number of executing compression function
$f(V_{i,j}, W_j)$	The j-th round of transformation in the i-th group of F
$P(X)$	The probability of event X
$Bin(N, p)$	Binomial distribution, where N is the number of trials and p denotes the success probability in each trial
$X \sim Bin(N, p)$	Event X follows $Bin(N, p)$
P_i	The average probability of successfully recovering one bit of the right half of registers in the final round of $F(V_{r-1}, M^{r-1})$

2.2 SHA2 family

We recap briefly the SHA2 family hash algorithms(SHA224, SHA256, SHA384 and SHA512) below, which have the similar structure of compression function. In SHA224 / SHA256, the word size n is 32 and the round number R is 64, while n is 64 and R is 80 in SHA384/SHA512. The whole hash function $H(M, IV)$ with inputs the padded message $M = M^0||M^1||...||M^{r-1}$ and the initial vector IV is described in Figure 1, where $M^i(i \in \{0, ..., r-1\})$ is the i-th message block, F is the compression function of SHA2 and V_i is the input of i-th function $F(V_i, M^i)$(where $V_0 = IV$).

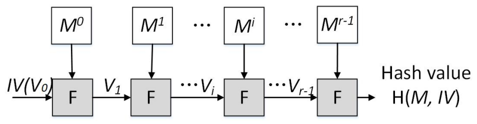


Fig. 1. Compression process of SHA2

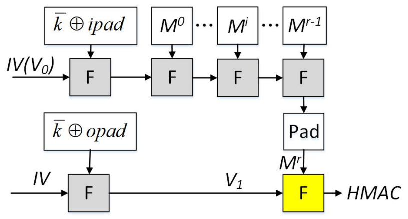


Fig. 2. Structure of HMAC

Message Extension. As shown in Figure 1, the message block M^i must be extended as R n-bits words $W_j (j = 0, ..., R-1)$. First, the message block M^i is split into sixteen n-bit words $W_0, ..., W_{15}$(i.e., $M^i = W_0||...||W_{15}$), and thereby is extended R words W_j by the following equation

$$W_j = \sigma_1 (W_{j-2}) + W_{j-7} + \sigma_0 (W_{j-15}) + W_{j-16} \, (16 \le j < R) \, . \tag{1}$$

Compression function. As depicted in Figure 3, the compression function F employs an R-round iterative transformation, denoted as $f(V_{i,j}, W_j)(j = 0, ..., R-1)$, which operates on the internal state for every round j from 0 to $R-1$. The internal state $V_{i,j} (i = 0, ..., r)$ is composed of eight concatenated n-bit word registers: $a_j, b_j, c_j, d_j, e_j, f_j, g_j$ and $h_j (j = 0, ..., R)$, such that $V_{i,j} = a_j||b_j||c_j||d_j||e_j||f_j||g_j||h_j$. This state is initially initialized with a constant initial vector V_0. i.e., $V_0 = V_{0,0} = a_0||b_0||c_0||d_0||e_0||f_0||g_0||h_0$. The transformation function f comprises several boolean functions namely $Maj(x, y, z)$, $Ch(x, y, z)$, $\sum_0 (x)$, $\sum_1 (x)$ and the additions modulo 2^n defined in Table 2, where K_j represents a predefined constant for each round. Following the execution of R rounds of transformation f, a final modulo 2^n addition takes place between the initial input $V_{i,0}$ and the output from the $(R-1)$-th round, $V_{i,R}$. This calculation produces the intermediate result V_{i+1}, which serves as the input for the subsequent iteration of the compression function F. Ultimately, after processing all groups, the final state V_r becomes the hash value $H(M, IV)$ when hashing the message M with the initialization vector IV.

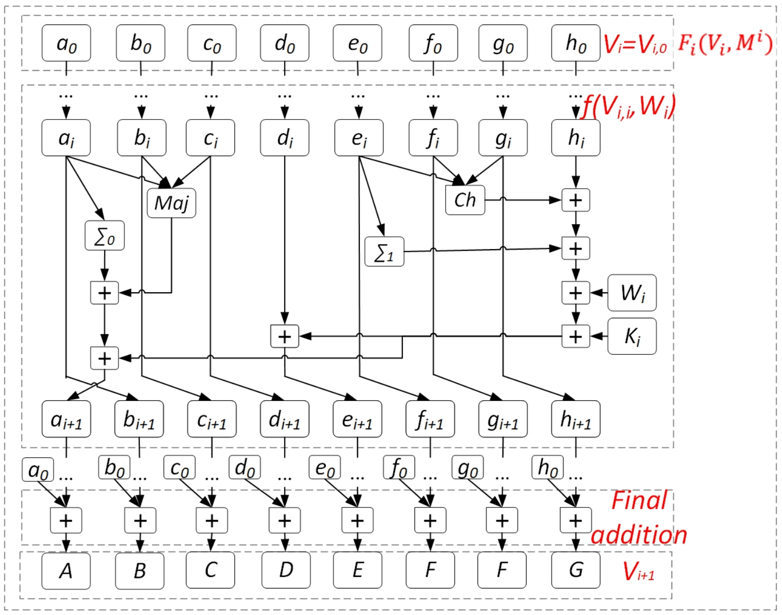
In our subsequent analysis, the spotlight is on two specific boolean functions within the round transformation $f(V_{i,j}, W_j)$: the $Ch(e_j, f_j, g_j)$ function and the $Maj(a_j, b_j, c_j)$ function.

2.3 HMAC of SHA2

HMAC is represented by

$$HMAC\,(M, k, IV) = H\left(\bar{k} \oplus opad \,\|\, H\left(\bar{k} \oplus ipad \,\|\, M, IV\right), IV\right),$$

where M represents the message being authenticated, k denotes the secret key, and $\bar{k}$ signifies the padded version of k that has been padded with zeroes to



a_0 b_0 c_0 d_0 e_0 f_0 g_0 h_0
$V_i = V_{i,0}$ $F_i(V_i, M^i)$
$f(V_{i,i}, W_i)$
a_i b_i c_i d_i e_i f_i g_i h_i
Maj
Ch
Σ_0
Σ_1
W_i
K_i
a_{i+1} b_{i+1} c_{i+1} d_{i+1} e_{i+1} f_{i+1} g_{i+1} h_{i+1}
Final addition
A B C D E F F G
V_{i+1}

Fig. 3. Compression function of SHA2

reach a specific length. *opad* and *ipad* are both $8n$-bit values that serve as unique padding for distinct stages of the HMAC process. The structure of HMAC can be visually illustrated in Figure 2. This figure would outline the dual hashing mechanism where the inner hash processes the concatenation of the XOR-ed padded key using *ipad* with the message M, along with an initialization vector IV. The result of this inner hash is then combined with the XOR-ed padded key using *opad* and rehashed, again incorporating the initial value IV. This double-hash approach ensures the integrity and authenticity of the message based on the security properties of the hash function H.

2.4 Boolean Theorems

The boolean operations, denoted by the symbols "$\oplus$", "$\cdot$" , "$\prime$" and "$+$" (as detailed in Table 2), when their operands are single-bit, exhibit the following distinctive properties.

Theorem 1 (operational rules).
$x, y, z \in \mathbb{Z}_2$ are 1-bit values and satisfy:

1. *distributive law.*
   (a) $(x \oplus y) \cdot z = (x \cdot z) \oplus (y \cdot z)$.
   (b) $(x + y) \cdot z = (x \cdot z) + (y \cdot z)$.
2. *absorption law.* $x + xy = x$, $x + x\prime y = x + y$.
3. $(x \cdot y)\prime = x\prime + y\prime$, $(x + y)\prime = x\prime \cdot y\prime$, $(x \oplus y)\prime = x\prime y + y\prime x$.

Theorem 2 (Transformation between boolean and arithmetic operations). *Given $x, y \in Z_2^n$, there exists the following transformation*

$$x + y = (x \oplus y) + ((x \cdot y) \ll 1)\,.$$

3 Methodology of Attack

In this section, we explore the specific differential characteristics of the boolean functions within SHA2 and formulate some deductions based on these properties. Subsequently, we introduce a differential fault attack strategy on SHA2 which can effectively recover the initial vector V_{r-1} and message block M^{r-1} in the final compression function.

3.1 Differential Fault Characteristics of Boolean Functions

Lemma 1. *If $x \oplus \sigma = x + \Delta (x, \sigma, \Delta \in \mathbb{Z}_2^n)$, then the bits of x, σ, Δ satisfy $\sigma_i\left(\Delta_i \oplus x_i\right) = \sigma_{i+1} \oplus \Delta_i \oplus \Delta_{i+1} (i = 0, \ldots, n-2)$ and $\sigma_0 = \Delta_0$.*

Proof. Let c is the carry of the addition modulo 2^n of x and σ. The bits of c satisfy

$$c_i = \begin{cases} 0 \quad i = 0 \\ \left(c_{i-1}\Delta_{i-1}\right) \oplus \left(c_{i-1} \oplus \Delta_{i-1}\right) x_{i-1} \quad 0 < i < n \end{cases}$$

and

$$x_i \oplus \sigma_i = x_i \oplus \Delta_i \oplus c_i \left(0 \le i < n\right).$$

Hence, $c_i = \sigma_i \oplus \Delta_i (\Delta_0 = \sigma_0)$ and $c_{i+1} = \Delta_{i+1} \oplus \sigma_{i+1} = c_i\Delta_i \oplus \left(c_i \oplus \Delta_i\right) x_i$.

Substituting c_i with $\sigma_i \oplus \Delta_i$, we have

$$\sigma_i\left(\Delta_i \oplus x_i\right) = \sigma_{i+1} \oplus \Delta_i \oplus \Delta_{i+1} \tag{2}$$

Lemma 2. *Given $\begin{cases} x \oplus \sigma^j = x + \Delta^j \\ x \oplus e\sigma^j = x + \Lambda^j \end{cases}$ for $j = 0, \ldots, N-1$, where $\sigma^j \in \mathbb{Z}_2^n$ is an unknown random number whose every bit follows $Bin(N, \frac{1}{2})$, $\Delta^j, \Lambda^j \in \mathbb{Z}_2^n$ are known values and $x, e \in \mathbb{Z}_2^n$ are unknown values, every bit e_i of e for $i = 0, \ldots, n-1$ can be determined with probability $1 - (\frac{1}{2})^N$.*

Proof. From *Lemma 1*, the following equations can be obtained.

$$\sigma_i^j\left(\Delta_i^j \oplus x_i\right) = \sigma_{i+1}^j \oplus A_i^j \tag{3}$$

$$\sigma_i^j e_i\left(\Lambda_i^j \oplus x_i\right) = \sigma_{i+1}^j e_{i+1} \oplus B_i^j \tag{4}$$

Hence, we can deduce

$$(\sigma_i^j)\prime\sigma_{i+1}^j = (\sigma_i^j)\prime A_i^j, \tag{5}$$

$$\sigma^j_{i+1} e_i' e_{i+1} = B^j_i e_i', \tag{6}$$

$$\sigma^j_{i+1} e_i e_{i+1}' = e_i \alpha^j_{i+1} \oplus e_i (\sigma^j_i)' \alpha^j_i, \tag{7}$$

where $A^j_i = \Delta^j_{i+1} \oplus \Delta^j_i$, $B^j_i = \Lambda^j_{i+1} \oplus \Lambda^j_i$, $\alpha^j_i = \Delta^j_i \oplus \Lambda^j_i$ and $\alpha^j_i \oplus \alpha^j_{i+1} = A^j_i \oplus B^j_i$. Equation (7) can be deduced by can deduce by multiplying e_i with equations (3) and (4) respectively and eliminating x_i. Equation (7) can be derived through a process that involves multiplying the equations (3) and (4) by e_i respectively, followed by algebraic elimination of x_i.

By equations (5), (6) and (7), the following relations hold.

1. **Case 1: obtain e_0.**
   From *Lemma 1*, we have $\sigma^j_0 = \Delta^j_0$ and $\sigma^j_0 e_0 = \Lambda^j_0$ for $j = 0, ..., N-1$. Hence, $\left(\sum\limits_{j=0}^{N-1} \sigma^j_0\right) e_0 = \sum\limits_{j=0}^{N-1} \Lambda^j_0$. By virtue of the randomness of σ^j_0, $e_0 = \sum\limits_{j=0}^{N-1} \Lambda^j_0$ holds with probability $1 - (\frac{1}{2})^N$ when $\sigma^j_0 = 1$ for a $j \in \{0, ..., N-1\}$.
2. **Case 2: obtain e_{i+1} when $e_i = 0$.**
   If $e_i = 0$, then $\sigma^j_{i+1} e_{i+1} = B^j_i$(by equation (6)). $\left(\sum\limits_{j=0}^{N-1} \sigma^j_{i+1}\right) e_{i+1} = \sum\limits_{j=0}^{N-1} B^j_i$. Similarly, $e_{i+1} = \sum\limits_{j=0}^{N-1} B^j_i$ holds with probability $1 - (\frac{1}{2})^N$ when $e_i = 0$.
3. **Case 3: obtain e_{i+1} when $e_i = 1$.**
   If $e_i = 1$, then $\sigma^j_{i+1} e_{i+1}' = \alpha^j_{i+1} \oplus \sigma^j_i{}' \alpha^j_i$(by equation (7)).
   When $i = 0$, $\sigma^j_0 = \Delta^j_0$ and $\sigma^j_1 e_1' = \alpha^j_1 \oplus \Delta^j_0{}' \alpha^j_0$.
   When $i > 0$, there are two cases:
   (a) If $e_{i-1} = 0$, then $\sigma^j_i = B^j_{i-1}$;
   (b) If $e_{i-1} = 1$, then $\alpha^j_i = \sigma^j_{i-1}{}' \alpha^j_{i-1}$. Hence, when $\alpha^j_i = 0$, $\sigma^j_{i+1} e_{i+1}' = \alpha^j_{i+1}$; when $\alpha^j_i = 1$, $\sigma^j_{i-1} = 0$ and $\sigma^j_i = A^j_{i-1}$(by equation (5)).

Consequently, from **Case 1-3**,

$$e_{i+1} = \begin{cases} \left(\sum\limits_{j=0}^{N-1} \left(\alpha^j_{i+1} \oplus B^j_{i-1}{}' \alpha^j_i\right)\right)' & e_{i-1} = 0, e_i = 1, i > 0 \\ \left(\sum\limits_{j=0, \alpha^j_i = 0}^{N-1} \left(\alpha^j_{i+1}\right) + \sum\limits_{j=0, \alpha^j_i = 1}^{N-1} \left(\alpha^j_{i+1} \oplus A^j_{i-1}{}' \alpha^j_i\right)\right)' & e_{i-1} = 1, e_i = 1, i > 0 \\ \left(\sum\limits_{j=0}^{N-1} \left(\alpha^j_1 \oplus \Delta^j_0{}' \alpha^j_0\right)\right)' & e_i = 1, i = 0 \end{cases}$$

holds with probability $1 - (\frac{1}{2})^N$ when $e_i = 1$.

Lemma 3. *Given* $\begin{cases} x \oplus e\sigma^j = x + \Lambda^j \\ x \oplus e'\varepsilon^j = x + \Psi^j \end{cases}$ *for* $j = 0, \ldots, N-1$, *where* $\sigma^j, \varepsilon^j \in \mathbb{Z}^n_2$ *are unknown random numbers whose every bit follows* $Bin(N, \frac{1}{2})$, $\Lambda^j, \Psi^j \in \mathbb{Z}^n_2$ *are known values and* $x, e \in \mathbb{Z}^n_2$ *are unknown values, every bit* e_i *of* e *can be determined with probability* $1 - (\frac{1}{2})^N$.

Proof. From *Lemma 1* and equation (6), we have

$$\begin{aligned} \sigma_{i+1}^j e_i{}' e_{i+1} &= B_i^j e_i{}' \\ \varepsilon_{i+1}^j e_i e_{i+1}{}' &= C_i^j e_i \end{aligned} \tag{8}$$

where $B_i^j = \Lambda_{i+1}^j \oplus \Lambda_i^j$ and $C_i^j = \Psi_{i+1}^j \oplus \Psi_i^j$.

Hence,

$$e_{i+1} = \begin{cases} \sum\limits_{j=0}^{N-1} B_i^j & e_i = 0 \\ \left(\sum\limits_{j=0}^{N-1} C_i^j\right)' & e_i = 1 \end{cases}$$

holds with probability $1-(\frac{1}{2})^N$, where e_0 equals to $\sum\limits_{j=0}^{N-1} \Lambda_0^j$ as defined in **Case 1** of Lemma 2.

Lemma 4. *Given that* $\begin{cases} x \oplus \sigma^j = x + \Delta^j \\ x \oplus e\sigma^j = x + \Lambda^j \end{cases}$ *for* $j = 0, \ldots, N-1$, *where* $\sigma^j \in \mathbb{Z}_2^n$ *is unknown random number whose every bit follows binomial distribution* $Bin(N, \frac{1}{2})$, $\Delta^j, \Lambda^j \in \mathbb{Z}_2^n$ *are known values,* $e \in \mathbb{Z}_2^n$ *is a known random number, and* $x \in \mathbb{Z}_2^n$ *are unknown constant, every bit* $x_i (0 \le i < n-1)$ *of* $x \in \mathbb{Z}_2^n$ *can be determined with an average probability* $\frac{1}{4}((1-(\frac{1}{2})^{E_{i+1}})(1-(\frac{1}{4})^{E_i})) + \frac{1}{2}(1-(\frac{1}{2})^{E_i})$, *where* E_i *is the expected value that* σ_i^j *is known for all the* $j \in \{0, ..., N-1\}$.

Proof. Let $\sigma_{i+1}^j = \mathcal{F}\left(\sigma_i^j, x_i\right)$, where $\mathcal{F}\left(\sigma_i^j, x_i\right)$ can be defined as

$$\mathcal{F}\left(\sigma_i^j, x_i\right) = \begin{cases} A_i^j & \sigma_i = 0 \\ \Delta_{i+1}^j \oplus x_i & \sigma_i = 1 \\ B_i^j & e_{i+1} = 1, e_i = 0 \\ \alpha_{i+1}^j \oplus \sigma_i^j{}' \alpha_i^j & e_{i+1} = 0, e_i = 1 \end{cases} \tag{9}$$

and $\sigma_0^j = \Delta_0^j$ for $j = 0, ..., N-1$.

We denote the expected value by E_i, which represents the average probability in N fault injections that the i-th bit of σ^j, i.e., σ_i^j is known for every j ranging from 0 to $N-1$ and random e following $Bin(N, 1/2)$. From equation(9) and the distribution of e, E_i satisfies the following equation when x_i is unknown, and E_i is larger than $\frac{N}{4}$ at least.

$$E_i = \begin{cases} N & i = 0 \\ \frac{E_{i-1}}{2} \times \frac{1}{2} + N \times \frac{1}{4} + E_{i-1} \times \frac{1}{4} = \frac{N}{2^i} + \frac{N}{4} & i \in \{1, \ldots, n-1\} \end{cases} \tag{10}$$

By the equations (3) and (4) in *Lemma 2*, we can deduce

$$\begin{aligned} \sigma_i^j \left(e_{i+1} \oplus e_i\right) x_i &= \sigma_i^j \left(\Delta_i^j e_{i+1} \oplus \Lambda_i^j e_i\right) \oplus A_i^j e_{i+1} \oplus B_i^j \\ \left(e_{i+1} \oplus e_i\right) \sigma_{i+1}^j &= B_i^j \oplus \left(A_i^j \oplus \sigma_i^j \alpha_i^j\right) e_i \end{aligned}, \tag{11}$$

where $A_i^j = \varDelta_{i+1}^j \oplus \varDelta_i^j$, $B_i^j = \varLambda_{i+1}^j \oplus \varLambda_i^j$, $\alpha_i^j = \varDelta_i^j \oplus \varLambda_i^j$ and $\alpha_i^j \oplus \alpha_{i+1}^j = A_i^j \oplus B_i^j$.

It is noted that, x_i just can be determined when $\sigma_i^j = 1$ and σ_{i+1}^j is known for some j (See equation (3)).

1. **Case 1:** $\{e_{i+1}, e_i\} = \{1, 0\}$
   (a) When $\{e_{i+1}, e_i\} = \{1, 0\}$, we have $\sigma_i^j \left(x_i \oplus \varDelta_i^j \right) = A_i^j \oplus B_i^j$ and $\sigma_{i+1}^j = B_i^j$ by equation (11).
   For $j = 0, ..., N-1$, if $\sigma_i^j = 1 (= \mathcal{F}\left(\sigma_{i-1}^j, x_{i-1}\right))$, then $x_i = \varDelta_{i+1}^j \oplus B_i^j$; Otherwise, if there exist $A_i^j \oplus B_i^j = 1$, then $x_i = \varDelta_i^j{}'$.
   (b) The average probability $P(\text{recovering } x_i | \{e_{i+1}, e_i\} = \{1, 0\})$ of recovering x_i when $\{e_{i+1}, e_i\} = \{1, 0\}$
   Since σ_{i+1}^j is known which equals to B_i^j for any j, the average probability of recovering x_i mainly depends on the condition that σ_i^j is known and equal to 1. From equation (10), we have $P(\text{recovering } x_i | \{e_{i+1}, e_i\} = \{1, 0\}) = 1 - (\frac{1}{2})^{E_i}$.
   Note that here the average probability rather than an exact probability is defined, due to the fact that the quantity of known σ_{i+1}^j is represented by the expected value E_i, which is contingent upon the random distribution of e_i. Consequently, the average probability might not exactly match the exact probability when e takes on specific real values(The experiments in Section 5 also can illustrate this case). Instead, it serves as an estimation that approximates the exact probability.
2. **Case 2:** $\{e_{i+1}, e_i\} = \{0, 1\}$
   (a) When $\{e_{i+1}, e_i\} = \{0, 1\}$, we have $\sigma_i^j \left(x_i \oplus \varLambda_i^j \right) = B_i^j$ and $\sigma_{i+1}^j = \alpha_{i+1}^j \oplus \sigma_i^j{}'\alpha_i^j$.
   For $j = 0, ..., N-1$, if $\sigma_i^j = 1 (= \mathcal{F}\left(\sigma_{i-1}^j, x_{i-1}\right))$, then $x_i = \varLambda_{i+1}^j$, Otherwise, if there exists $B_i^j = 1$, then $x_i = \varLambda_i^j{}'$.
   (b) The average probability $P(\text{recovering } x_i | \{e_{i+1}, e_i\} = \{0, 1\})$
   Similarly, the recovery of x_i depends on the known σ_i^js equaling to 1. Hence, $P(\text{recovering } x_i | \{e_{i+1}, e_i\} = \{1, 0\}) = 1 - (\frac{1}{2})^{E_i}$.
3. **Case 3:** $\{e_{i+1}, e_i\} = \{1, 1\}$. There are also two sub-phases:
   (a) If $\{e_{i+2}, e_{i+1}, e_i\} = \{1, 1, 1\}$, substituting $\{e_{i+2}, e_{i+1}, e_i\} = \{1, 1, 1\}$ into equation (11), we have $\sigma_i^j{}'\alpha_i^j = \alpha_{i+1}^j$ and $\sigma_{i+1}^j{}'\alpha_{i+1}^j = \alpha_{i+2}^j$. If $\sigma_i^j = 1$, then $\alpha_{i+1}^j = \alpha_{i+2}^j = 0$. Therefore, σ_{i+1}^j and x_i cannot be determined when $\sigma_i^j = 1$.
   (b) If $\{e_{i+2}, e_{i+1}, e_i\} = \{0, 1, 1\}$, we have $\sigma_{i+1}^j \left(x_{i+1} \oplus \varLambda_{i+1}^j \right) = B_{i+1}^j$ similarly.
   For $j = 0, ..., N-1$ and $i > 0$, if x_{i+1} has been known, $\sigma_i^j = 1 (= \mathcal{F}\left(\sigma_{i-1}^j, x_{i-1}\right))$ and $\left(x_{i+1} \oplus \varLambda_{i+1}^j \right) = 1$, then $\sigma_{i+1}^j = B_{i+1}^j$ and $x_i = \varDelta_{i+1}^j \oplus B_{i+1}^j$; Otherwise, if there exist $\sigma_i^j = 1$ and $B_{i+1}^j = 1$, then $\sigma_{i+1}^j = 1$, $x_{i+1} = \varLambda_{i+1}^j$ and $x_i = \varDelta_{i+1}^j{}'$.

(c) The average probability $P(\text{recovering } x_i | \{e_{i+2}, e_{i+1}, e_i\} = \{0,1,1\})$
The key conditions recovering x_i are as follow: 1) x_{i+1} is known whose average probability is $1-(\frac{1}{2})^{E_{i+1}}$ (see Case 2); 2) σ_i^j is known and equals to 1 and 3) $\left(x_{i+1} \oplus \Lambda_{i+1}^j\right) = 1$ depending on Λ_{i+1}^j with a probability $\frac{1}{2}$. Hence, $P(\text{recovering } x_i | \{e_{i+2}, e_{i+1}, e_i\} = \{0,1,1\}) = (1-(\frac{1}{2})^{E_{i+1}})(1-(\frac{1}{4})^{E_i})$.

4. **Case 4 :** $\{e_{i+1}, e_i\} = \{0,0\}$**.** There are two sub-phases:
(a) If $\{e_{i+2}, e_{i+1}, e_i\} = \{0,0,0\}$, we have $B_i^j = B_{i+1}^j = 0$. There is no available information for deducing x_i.
(b) If $\{e_{i+2}, e_{i+1}, e_i\} = \{1,0,0\}$, we have $\sigma_{i+1}^j \left(x_{i+1} \oplus \Delta_{i+1}^j\right) = A_{i+1}^j \oplus B_{i+1}^j$ and $\sigma_{i+2}^j = B_{i+1}^j$.
If x_{i+1} has been known, $\sigma_i^j = 1 (= \mathcal{F}\left(\sigma_{i-1}^j, x_{i-1}\right))$ and $x_{i+1} \oplus \Delta_{i+1}^j = 1$, then $\sigma_{i+1}^j = A_{i+1}^j \oplus B_{i+1}^j$ and $x_i = \Delta_{i+2}^j \oplus B_{i+1}^j$; Otherwise, for $j = 0, ..., N-1$, if there exist $\sigma_i^j = 1$ and $A_{i+1}^j \oplus B_{i+1}^j = 1$, then $\sigma_{i+1}^j = 1$ and $x_i = \Delta_{i+1}^j{\prime}$.
(c) The average probability $P(\text{recovering } x_i | \{e_{i+2}, e_{i+1}, e_i\} = \{1,0,0\})$
Similar to Case 3, $P(\text{recovering } x_i | \{e_{i+2}, e_{i+1}, e_i\} = \{1,0,0\}) = (1-(\frac{1}{2})^{E_{i+1}})(1-(\frac{1}{4})^{E_i})$.

In summary, the average probability of successfully recovering x_i, denoted henceforth as $P(\text{recovering } x_i)$ or more succinctly as P_i, satisfies the condition that

$$P_i = \begin{cases} \frac{\left(1-\left(\frac{1}{2}\right)^{E_{i+1}}\right)\left(1-\left(\frac{1}{4}\right)^{E_i}\right)}{4} + \frac{1-\left(\frac{1}{2}\right)^{E_i}}{2} & i \in \{0, \ldots, n-3\} \\ \frac{1-\left(\frac{1}{2}\right)^{E_i}}{2} & i = n-2 \end{cases} \tag{12}$$

When E_i(larger than $N/4$ at least) is enough large, P_i approximates to $\frac{3}{4}$ for $i \in \{0, \ldots, n-3\}$ and $\frac{1}{2}$ for $i = n-2$, respectively. Specially, when $\{e_{i+2}, e_{i+1}, e_i\} = \{0,0,0\}$ and $\{e_{i+2}, e_{i+1}, e_i\} = \{1,1,1\}$ with probability $\frac{1}{4}$, x_i can not be determined even if inducing more faults σ^j. Alternatively, the indetermination of the remain bits of x can be effectively offset by leveraging the randomness of the eight distinct instances of e within Algrithm 2.

In addition, it is noted that the most significant bit x_{n-1} can not be recovered due to no ${\sigma_n}^j$.

Lemma 5. *Given* $Maj\,(a,b,c) = ab \oplus bc \oplus ac$*,* $Maj\,(a, b\oplus\sigma, c) - Maj\,(a,b,c) = (b \oplus (a \oplus c)\,\sigma) - b$ *and* $Maj\,(a,b,c\oplus\sigma) - Maj\,(a,b,c) = (c \oplus (a\oplus b)\,\sigma) - c$ *hold, where* $a,b,c,e,f,g,\sigma \in \mathbb{Z}_2^n$*.*

Proof. From *Theorem 2*, we have

$$\begin{aligned} &Maj\,(a, b\oplus\sigma, c) = Maj\,(a,b,c) \oplus (a\oplus c)\,\sigma \\ &= Maj\,(a,b,c) + (a\oplus c)\,\sigma - (Maj\,(a,b,c)\,(a\oplus c)\,\sigma) \ll 1 \\ &= Maj\,(a,b,c) + (a\oplus c)\,\sigma - (b\,(a\oplus c)\,\sigma) \ll 1 \\ &= Maj\,(a,b,c) + (b \oplus (a\oplus c)\,\sigma) - b \end{aligned}.$$

Similarly, equation $Maj\,(a, b, c \oplus \sigma) - Maj\,(a, b, c) = (c \oplus (a \oplus b)\,\sigma) - c$ can also be proved.

Lemma 6. *Given* $Ch\,(e, f, g) = ef \oplus e'g$, $Ch\,(e, f \oplus \sigma, g) - Ch\,(e, f, g) = (f \oplus e\sigma) - f$ *and* $Ch\,(e, f, g \oplus \sigma) - Ch\,(e, f, g) = (f \oplus e'\sigma) - f$ *hold, where* $a, b, c, e, f, g, \sigma \in \mathbb{Z}_2^n$.

Proof. From *Theorem 2*, we have

$$\begin{aligned} &ch\,(e, f \oplus \sigma, g) = ch\,(e, f, g) \oplus e\sigma \\ &= ch\,(e, f, g) + e\sigma - (ch\,(e, f, g)\, e\sigma) \ll 1 \\ &= ch\,(e, f, g) + e\sigma - (ef\sigma \ll 1) \\ &= ch\,(e, f, g) + (f \oplus e\sigma) - f \end{aligned}$$

Similarly, equation $Ch\,(e, f, g \oplus \sigma) - Ch\,(e, f, g) = (f \oplus e'\sigma) - f$ can also be demonstrated to hold true using the same methodology.

3.2 Recovery of Initial Input and Message Block in Final Compression Function

By leveraging the preceding lemmas and adopting the random word fault model, we are able to recover the initial input and message block within the final compression function of SHA2 (as depicted in Figure 1). It is important to note that this random word fault model assumes a scenario where only one register, say x, is randomly perturbed during each fault injection. In such cases, the faulty version of the register after the j-th injection, denoted as x^j, can be represented as $x \oplus \sigma^j$. Here, σ^j is a random number for each $j = 0, ..., N-1$, with its individual bits following a binomial distribution $Bin(N, 1/2)$. Following the fault injection process, the subsequent conclusion can be derived.

Proposition 1. *Given both correct and faulty hash values denoted as* $\{A, ..., H\}$ *and* $\{A^j, ..., H^j\}$ *of SHA2 under* 14 *random word fault targets for any* $j \in \mathbb{Z}_N$ *(i.e.,* $14N$ *fault injections), it is possible to recover the* $(R-1)$*-th input registers* $\{a_{R-1}, b_{R-1}, ..., h_{R-1}\}$ *and the corresponding faulty values* $\{a^j_{R-1}, b^j_{R-1}, ..., h^j_{R-1}\}$ *with a probability* $P_r * P_l$*, where* $P_r = \left(1 - \left(\frac{1}{2}\right)^N\right)^{4n}$, $P_l = \left(1 - \left(\frac{1}{2}\right)^N\right)^{8n} \prod_{i=0}^{n-2}\left(1 - (1 - P_i)^8\right)$, *and the number* T *of bits required to be guessed is generally equal to 1 when* N *is sufficiently large.*

Proof. **Step.1 Recovering e_{R-1}, f_R, f_0 with faulty f_{R-1} in $R-1$-th round.**

As shown in Figure 4, if fault injection is induced into f_{R-1} for N times, then f_{R-1} is disturbed as $f^j_{R-1} = f_{R-1} \oplus \sigma^j$, and the final outputs A,E and G are changed as A^j, E^j and G^j for $j = 0, ..., N-1$.

From Lemma 6, we have

$$\begin{aligned} G^j - G &= \left(f_{R-1} \oplus \sigma^j\right) - f_{R-1} \\ A^j - A &= \left(f_{R-1} \oplus e_{R-1}\sigma^j\right) - f_{R-1} \end{aligned} \tag{13}$$

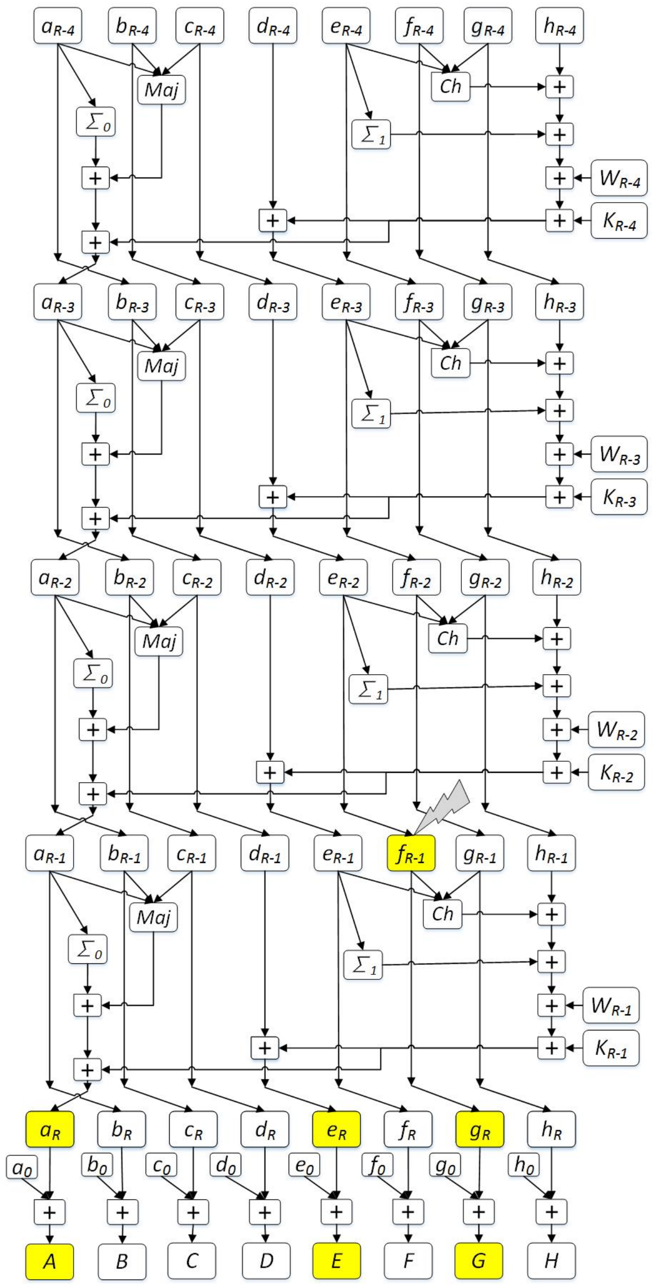


Fig. 4. Fault injection on f_{R-1}

Let $G^j - G = \Delta^j$ and $A^j - A = \Lambda^j$, then e_{R-1} can be determined from Lemma 2 with probability $\left(1 - (\frac{1}{2})^N\right)^n$. Thereby, f_R and f_0 can be recovered.

Step.2 Recovering f_{R-1}, g_R and g_0 with faulty f_{R-2} in $R-2$-th round.

By the same way, if fault injection is induced into f_{R-2}, then f_{R-2}, H, B are changed into $f^j_{R-2} = f_{R-2} \oplus \sigma^j$, H^j and B^j, repectively.

$$\begin{array}{l} H^j - H = \left(f_{R-2} \oplus \sigma^j\right) - f_{R-2} \\ B^j - B = \left(f_{R-2} \oplus e_{R-2}\sigma^j\right) - f_{R-2} \end{array} \tag{14}$$

From Lemma 2, e_{R-2}(i.e., g_R and g_0) can be recovered with probability $\left(1 - (\frac{1}{2})^N\right)^n$.

Step.3 Recovering $g_{R-1}(h_R)$ and h_0 with faulty f_{R-3} and g_{R-3} in $R-3$-th round.

If fault injection is induced into f_{R-3} for N times, then f_{R-3} is changed into $f^j_{R-3} = f_{R-3} \oplus \sigma^j$ and the output C is changed into C^j for $j = 0, ..., N-1$. Since the differential value of $f^j_{R-3} - f_{R-3}$ can not be obtained from the hash output directly, fault injection is still required to be induced into g_{R-3} for N times, then g_{R-3} is changed into $g^j_{R-3} = g_{R-3} \oplus \varepsilon^j$ and the output C is changed into C^{N+j} for $j = 0, ..., N-1$. In mathematical notation, ε^j represents a random variable that is akin to σ^j.

Hence, from Lemma 6, we have the following equations

$$\begin{array}{l} C^j - C = \left(f_{R-3} \oplus e_{R-3}\sigma^j\right) - f_{R-3} \\ C^{j+N} - C = \left(g_{R-3} \oplus e_{R-3}\prime\varepsilon^j\right) - g_{R-3} \end{array}. \tag{15}$$

From Lemma 3, let $C^j - C = \Lambda^j, C^{j+N} - C = \Psi^j$, then e_{R-3}, i.e., h_R and h_0 can be recovered with probability $\left(1 - (\frac{1}{2})^N\right)^n$.

Step.4 Recovering h_{R-1} with faulty f_{R-4} and g_{R-4} and in $R-4$-th round.

Consistently with the approach described above, fault injection is introduced into both registers f_{R-4} and g_{R-4} respectively, the faulty $f^j_{R-4}(= f^j_{R-4} \oplus \sigma^j)$, the corresponding faulty output D^j, $g^j_{R-4}(= g^j_{R-4} \oplus \varepsilon^j)$ and the corresponding faulty output D^{j+N} satisfy the following equations

$$\begin{array}{l} D^j - D = \left(f_{R-4} \oplus e_{R-4}\sigma^j\right) - f_{R-4} \\ D^{j+N} - D = \left(g_{R-4} \oplus e_{R-4}\prime\varepsilon^j\right) - g_{R-4} \end{array}. \tag{16}$$

From Lemma 3, let $D^j - D = \Lambda^j, D^{j+N} - D = \Psi^j$, then e_{R-4}, i.e., h_{R-1} can be recovered with probability $\left(1 - (\frac{1}{2})^N\right)^n$.

In summary, the four sequential steps outlined above enable the recovery of registers $\{e_{R-1}, f_{R-1}, g_{R-1}, h_{R-1}\}$, $\{f_R, g_R, h_R\}$ and $\{f_0, g_0, h_0\}$ with a probability P_r satisfying

$$P_r = \left(1 - \left(\frac{1}{2}\right)^N\right)^{4n}. \tag{17}$$

Meanwhile, the recovery needs conducting $6N$ fault injections.

It is worth noting that under circumstances where an attacker does not have access to G^j, G, H^j, and H, such as in the cases of SHA224 or SHA384, steps 1 and 2 can mount the same attacks as steps 3 and 4. Specially, N times of fault injections are induced to disturb f_{R-1} and g_{R-1} in step 1(f_{R-2} and g_{R-2} in step 2), and input the faulty A^j and A^{N+j}(B^j and B^{N+j} in step 2), respectively. As stated in step 3, $e_{R-1}(f_{R-1})$ can be recovered with Lemma 3.

Step.5 Recovering $a_{R-1} \oplus c_{R-1}$ and partial bits of b_{R-1}(denoted as $part(b_{R-1})$) with faulty b_{R-1} in $R-1$-th round.

As shown in Figure 4, if fault injection is induced into b_{R-1} for N times, then b_{R-1} is disturbed as $b^j_{R-1}(= b_{R-1} \oplus \sigma^j)$ randomly, and the final outputs A and C are changed as A^j and C^j for $j = 0, ..., N-1$, respectively.

From Lemma 5, we have

$$\begin{array}{l} C^j - C = \left(b_{R-1} \oplus \sigma^j\right) - b_{R-1} \\ A^j - A = \left(b_{R-1} \oplus (a_{R-1} \oplus c_{R-1})\sigma^j\right) - b_{R-1} \end{array}. \tag{18}$$

From Lemma 2, let $C^j - C = \Delta^j, A^j - A = \Lambda^j$, then $a_{R-1} \oplus c_{R-1}$ can be recovered with probability $\left(1 - (\frac{1}{2})^N\right)^n$.

Since $a_{R-1} \oplus c_{R-1}$ is known in equation (18), b_{R-1} can also be recovered from Lemma 4 with a probability P_i approximating to $\frac{3}{4}$, which depends on the value of $a_{R-1} \oplus c_{R-1}$. In view of the randomness of $a_{R-1} \oplus c_{R-1}$, about $\frac{1}{4}$ bits of b_{R-1} denoted by $part(b_{R-1})$ cannot be determined even when N is sufficiently large to the extent that $\left(1 - (\frac{1}{2})^N\right)^n \approx 1$.

Step.6 Recovering $a_{R-1} \oplus b_{R-1}$ and partial bits of c_{R-1}(denoted as $part(c_{R-1})$) with faulty c_{R-1} in $R-1$-th round.

By the same way with Step.5, c_{R-1}, A and D are disturbed as $c^j_{R-1}(= c_{R-1} \oplus \sigma^j)$, A^j and D^j by fault injection for $j = 0, ..., N-1$, respectively.

From Lemma 5, we have

$$\begin{array}{l} D^j - D = \left(c_{R-1} \oplus \sigma^j\right) - c_{R-1} \\ A^j - A = \left(c_{R-1} \oplus (a_{R-1} \oplus b_{R-1})\delta^j\right) - c_{R-1} \end{array}. \tag{19}$$

Naturally, $a_{R-1} \oplus b_{R-1}$ and $part(c_{R-1})$(depending on the randomness of $a_{R-1} \oplus b_{R-1}$) can be recovered by Lemmas 2 and 4.

Step.7 Recovering more bits of a_{R-1}, b_{R-1} and c_{R-1}.

Since knowing $a_{R-1} \oplus b_{R-1}$ and $a_{R-1} \oplus c_{R-1}$, i.e., knowing $b_{R-1} \oplus c_{R-1}$, $part(b_{R-1})$ and $part(c_{R-1})$ as mentioned above can be progressively refined or updated with more known bits by Algorithm 1. Consequently, the corresponding bits of a_{R-1}(denoted by $part(a_{R-1})$) with respect to $part(b_{R-1})$ and $part(c_{R-1})$ also can be updated. From Lemma 4, the probability of undetermined bits of $part(b_{R-1})$ and $part(c_{R-1})$ is $(1 - P_i)^2$ which approximates to $\frac{1}{2^4}$ when knowing $a_{R-1} \oplus b_{R-1}$ and $a_{R-1} \oplus c_{R-1}$.

Step.8 Recovering $b_{R-1} \oplus d_{R-1}$ and updating $part(c_{R-1})$ with faulty b_{R-2} in $R-2$-th round.

Algorithm 1 Algorithm of updating bits of target registers

```
Require: $u(= b \oplus c)$, $part(b)$ and $part(c)$
Ensure: updated $part(b)$ and $part(c)$
  For $i = 0$ to $n - 1$
      If $b_i$ is known in $part(b)$ and $c_i$ is unknown in $part(c)$.
          update $c_i = u_i \oplus b_i$ and set $c_i$ known in $part(c)$.
      Else If $c_i$ is known in $part(c)$ and $b_i$ is unknown in $part(b)$.
          update $b_i = u_i \oplus c_i$ and set $b_i$ known in $part(b)$.
  return $part(b)$ and $part(c)$
```

Analogously, if b_{R-2} is randomly disturbed by conducting N instances of fault injection, $a_{R-2} \oplus c_{R-2}$(i.e., $b_{R-1} \oplus d_{R-1}$) can be recovered by Lemma 2, and thereby $part(c_{R-1})$(i.e., b_{R-2}) can be updated with more known bits by Lemma 4. The probability of undetermined bits of $part(b_{R-1})$ and $part(c_{R-1})$ is $(1 - P_i)^3$ approximates to $\frac{1}{2^6}$ when knowing $b_{R-1} \oplus d_{R-1}$.

Step.9 Recovering more bits of a_{R-1}, b_{R-1}, c_{R-1} and d_{R-1}.

Since knowing $a_{R-1} \oplus c_{R-1}$, $b_{R-1} \oplus c_{R-1}$, $b_{R-1} \oplus d_{R-1}$ and the updated $part(c_{R-1})$, the corresponding $part(a_{R-1})$, $part(b_{R-1})$ and $part(d_{R-1})$ with respect to $part(c_{R-1})$ still can be updated by Algorithm 1.

When the number of undetermined bits reaches a predefined lower threshold T (where T is conventionally set to 1), this signifies that $n - T$ bits of the registers a_{R-1}, b_{R-1}, c_{R-1}, and d_{R-1} have been successfully recovered. However, if this condition is not met, the fault injection process can be iteratively continued.

In the $(R - 2)$-th round, faults are injected into register c_{R-2} to update $part(d_{R-1})$. In the subsequent $(R - 3)$-th round, fault injections target both b_{R-3} (enabling recovery of $c_{R-1} \oplus c_{R-3}$ and updating $part(d_{R-1})$) and c_{R-3} for updating $part(c_{R-1})$. Moving on to the $(R - 4)$-th round, faults are introduced into b_{R-4} to recover $d_{R-1} \oplus c_{R-4}$ and further update $part(d_{R-1})$, as well as into c_{R-4} to recover $d_{R-1} \oplus b_{R-4}$ and again update $part(d_{R-1})$. This iterative process allows for the gradual refinement and determination of $part(a_{R-1})$, $part(b_{R-1})$, $part(c_{R-1})$ and $part(d_{R-1})$ until all desired bits are resolved.

To sum up, $n - T$ bits of a_{R-1}, b_{R-1}, c_{R-1} and d_{R-1} can be recovered by the following Algorithm 2, in which T depends on the distributions of a_{R-1}, b_{R-1}, c_{R-1} and d_{R-1}. Except for the never undetermined most significant bit, the probability determining every one in the remaining bits of a_{R-1}, b_{R-1}, c_{R-1} and d_{R-1} is up to $1 - (1 - P_i)^8$ which approximates to 99.998%. Hence, the probability of successfully recovering all remaining bits is given by the following equation

$$P_l = \left(1 - \left(\frac{1}{2}\right)^N\right)^{8n} \prod_{i=0}^{n-2} (1 - (1 - P_i)^8), \tag{20}$$

which approximates to 99% when n assumes either 32 or 64(P_i referring to the individual bit recovery probability as defined in equation (12)) and N is enough big such as $N = 15$. Therefore, T is usually equal to 1. In addition, once the high T-bit values of any one in $\{a_{R-1}, b_{R-1}, c_{R-1}, d_{R-1}\}$ are determined, all the bits of $\{a_{R-1}, b_{R-1}, c_{R-1}, d_{R-1}\}$ can be recovered by the known values $a_{R-1} \oplus b_{R-1}, a_{R-1} \oplus c_{R-1}$ and $c_{R-1} \oplus d_{R-1}$. Therefore, only 1 bits is generally needed to guess in our analysis when recovering the left half registers. By Algorithm 2 above, we can recover $\{a_{R-1}, b_{R-1}, c_{R-1}, d_{R-1}\}$(with 1 guessed bit), $\{b_R, c_R, d_R\}$ and $\{b_R, c_R, d_R\}$. Moreover, $8N$ times of fault injections are needed for recovery.

Finally, $\{a_{R-1}, b_{R-1}, ..., h_{R-1}\}$, $\{b_R, c_R, d_R, f_R, g_R, h_R\}$ and $\{b_0, c_0, ..., h_0\}$ can be recovered, in which only one bit in $\{a_{R-1}, b_{R-1}, c_{R-1}, d_{R-1}\}$ is needed to guess with overwhelming probability $P_r * P_l$. Meanwhile, the recovery needs $14N$ times of fault injections.

Algorithm 2 Algorithm of recovering $n-T$ bits of a_{R-1}, b_{R-1}, c_{R-1} and d_{R-1}

```
Require: All the faulty hash results $\{S_0^j, S_1^j, S_2^j, S_3^j, S_4^j, S_5^j, S_6^j, S_7^j\}(=\{A^j, B^j, C^j, D^j, E^j, F^j, G^j, H^j\})$ for any $j = 0, ..., N-1$ and the correct hash results $\{S_0, S_1, S_2, S_3, S_4, S_5, S_6, S_7\}(=\{A, B, C, D, E, F, G, H\})$
Ensure: the last $n-T$ bits of $a_{R-1}$, $b_{R-1}$, $c_{R-1}$ and $d_{R-1}$ denoted by their respective segments $part(a_{R-1})$, $part(b_{R-1})$, $part(c_{R-1})$, and $part(d_{R-1})$
  1. Set the value of $T$ (such as $T = 1$), and let $ub$ to be the length of the unknown bits of $a_{R-1}$(or $b_{R-1}$, $c_{R-1}$, $d_{R-1}$).
  2. For $r = R-1$ to $R-4$
      2.1. If $ub > T$
          2.1.1. Fault injection is randomly induced into $b_r$ for $N$ times, and results in the faulty $b_r^j$ satisfying $b_r^j = b_r \oplus \sigma^j$ and the final outputs $\{S_0^j, S_1^j, S_2^j, S_3^j, S_4^j, S_5^j, S_6^j, S_7^j\}(=\{A^j, B^j, C^j, D^j, E^j, F^j, G^j, H^j\})$ for $j = 0, ..., N-1$.
          2.1.2. Let $\Delta^j = S_{R-r+1}^j - S_{R-r+1} = (b_r \oplus \sigma^j) - b_r$, $\Lambda^j = S_{R-r-1}^j - S_{R-r-1} = (b_r \oplus (a_r \oplus c_r)\sigma^j) - b_r$.
          2.1.3. Recover $a_r \oplus c_r$ by Lemma 2 and $part(b_r)$ by Lemma 4.
          2.1.4. Select $c_r$ as the target of fault injection and then sequentially replicate the analogous steps 2.1.1 through 2.1.3.
          2.1.5. Recover $a_r \oplus b_r$ by Lemma 2 and $part(c_r)$ by Lemma 4.
      2.2. Update $part(a_{R-1}), part(b_{R-1}), part(c_{R-1}), part(d_{R-1})$ by Algorithm 1.

  return $part(a_{R-1}), part(b_{R-1}), part(c_{R-1}), part(d_{R-1})$
```

Step.10 Recovering the faulty registers $\{a_{R-1}^j, ..., h_{R-1}^j\}$ for $j = 0, ..., N-1$

As shown in Figure 4, we have known $\{a_{R-1}, ..., h_{R-1}\}$, $\{b_0, c_0, d_0, f_0, g_0, h_0\}$, the correct hash values $\{A, ..., H\}$ and the faulty hash values $\{A^j, ..., H^j\}$ for any $j \in 0, ..., N-1$. Obviously, we have

$$a_{R-1}^j = B^j - b_0, b_{R-1}^j = C^j - c_0, c_{R-1}^j = D^j - d_0, and$$

$$e^j_{R-1} = F^j - f_0, f^j_{R-1} = G^j - g_0, g^j_{R-1} = H^j - h_0.$$

Let $T_i = Ch(e_i, f_i, g_i) + \sum_1 (e_i)$, $U_i = Maj(a_i, b_i, c_i) + \sum_0 (a_i)$, $T^j_i = Ch(e^j_i, f^j_i, g^j_i) + \sum_1 \left(e^j_i\right)$ and $U^j_i = Maj(a^j_i, b^j_i, c^j_i) + \sum_0 \left(a^j_i\right)(i \in 0, ..., R-1$ and $j \in 0, ..., N-1)$, then we have

$$h^j_{R-1} = A^j - A - (T^j_{R-1} + U^j_{R-1} - T_{R-1} - U_{R-1}) + h_{R-1},$$

where A^j, A, T^j_{R-1}, U^j_{R-1}, U_{R-1} and T_{R-1} are all known.

Hence, $d^j_{R-1} = E^j - E - (T^j_{R-1} + h^j_{R-1} - T_{R-1} - h_{R-1}) + d_{R-1}$.

Q.E.D

Proposition 2. *Given both correct and faulty $R-1$-th round input values denoted as $\{a_{R-1}, b_{R-1}, ..., h_{R-1}\}$ and $\{a^j_{R-1}, b^j_{R-1}, ..., h^j_{R-1}\}$ of SHA2 for any $j \in \mathbb{Z}_N$ under 48 random word fault targets, it is possible to recover the message block M^{r-1} and initial vector V_{r-1} of the final compression function $F(V_{r-1}, M^{r-1})$ with a probability $(1 - \frac{1}{2^N})^{32n}$.*

Proof. **Step 1. Recovering correct registers** $a_{R-2}, b_{R-2}, ..., h_{R-2}$

Obviously, $a_{R-2} = b_{R-1}$, $b_{R-2} = c_{R-1}$, $c_{R-2} = d_{R-1}$, $e_{R-2} = f_{R-1}$, $f_{R-2} = g_{R-1}$, $g_{R-2} = h_{R-1}$.

As proven in the Step 4 of proposition 1, if N times fault injections are induced in f_{R-5} and g_{R-5} respectively, then f_{R-5} and g_{R-5} are disturbed as $f^j_{R-5}(= f_{R-5} \oplus \sigma^j)$ and $g^j_{R-5}(= g_{R-5} \oplus \varepsilon^j)$, respectively. Hence, from Lemma 6,

$$\begin{array}{l} d^j_{R-1} - d_{R-1} = \left(f_{R-5} \oplus e_{R-5}\sigma^j\right) - f_{R-5} \\ d^{N+j}_{R-1} - d_{R-1} = \left(g_{R-5} \oplus e_{R-5}\prime\varepsilon^j\right) - g_{R-5} \end{array},$$

where d^j_{R-1} and d^{N+j}_{R-1} are the j-th faulty values of fourth register when mounting fault injection toward f_{R-5} and g_{R-5}, respectively.

From Lemma 3, let $d^j_{R-1} - d_{R-1} = \Lambda^j, d^{N+j}_{R-1} - d_{R-1} = \Psi^j$, then h_{R-5} can be recovered with probability $(1 - \frac{1}{2^N})^n$ and $2N$ fault injections.

As proven in the Step 5 of proposition 1, if N times fault injections are induced in b_{R-3} , then b_{R-3} are disturbed as $b^j_{R-3}(= b_{R-3} \oplus \sigma^j)$. Hence, from Lemma 5, we have

$$\begin{array}{l} d^j_{R-1} - d_{R-1} = \left(b_{R-3} \oplus \sigma^j\right) - b_{R-3} \\ b^j_{R-1} - b_{R-1} = \left(b_{R-3} \oplus (a_{R-3} \oplus c_{R-3})\sigma^j\right) - b_{R-3} \end{array}.$$

From Lemma 2, $a_{R-3} \oplus c_{R-3}$ can be recovered. Knowing $a_{R-3} = c_{R-1}$ and $c_{R-3} = d_{R-2}$, d_{R-2} can be recovered with probability $(1 - \frac{1}{2^N})^n$ and N fault injections.

Step 2. Recovering message W_{R-2}

Knowing T_{R-2}(defined in Step 10 of proposition 1, i.e., $Ch(e_{R-2}, f_{R-2}, g_{R-2}) + \sum_1 (e_{R-2})$), d_{R-2}, K_{R-2} and e_{R-1}, we have

$$W_{R-2} = e_{R-1} - T_{R-2} - K_{R-2} - d_{R-2}.$$

Step 3. Recovering faulty registers $\{a^j_{R-2}, ..., h^j_{R-2}\}$ for any $j \in \mathbb{Z}_N$

In $\{a^j_{R-2}, ..., h^j_{R-2}\}$, only d^j_{R-2} and h^j_{R-2} are unknown values. Moveover, T^j_{R-2} and U^j_{R-2} denoted in step 10 in proposition 1 are known. Hence, we have

$$h^j_{R-2} = a^j_{R-1} - T^j_{R-2} - W_{R-2} - K_{R-2} - U^j_{R-2},$$

and

$$d^j_{R-2} = e^j_{R-1} - T^j_{R-2} - W_{R-2} - K_{R-2} - h^j_{R-2}.$$

After the Steps 1-3 above, the $R-2$-th correct and faulty round input and message W_{R-2} can be recovered with probability $(1 - \frac{1}{2^N})^{2n}$ and $3N$ fault injections.

Step 4. Recovering the message block M^{r-1} and initial vector V_{r-1} in the compression function $F(V_{r-1}, M^{r-1})$.

Continue iterating through the Steps 1-3 until all 16 sets of intermediate round messages, specifically $\{W_{R-2}, ..., W_{R-17}\}$, have been successfully recovered. Subsequently, the original message block M can be inferred using equation (1). Furthermore, the initial input vector V_{r-1} can also be retroactively derived by leveraging the recovered message block M^{r-1}, along with the known inputs $\{a_{R-1}, ..., h_{R-1}\}$ from the final compression function stage.

To sum up , the last message block M^{r-1} and its corresponding initial vector V_{r-1} can be recovered with probability $(1 - \frac{1}{2^N})^{32n}$ and $48N$ fault injections.

Q.E.D

Finally, propositions 1 and 2 can be utilized to realize a practical differential fault attack. The more detailed procedure for accomplishing this is outlined in Algorithm 3. It is worth noting that Step 1 in Proposition 2 can be bypassed if the registers $\{a_{R-3}, ..., d_{R-3}\}$ have already been successfully retrieved during the execution of Algorithm 2 as prescribed in Proposition 1. As a result of this optimization, the cumulative number of fault injections diminishes to $62N - 2$ instances.

4 Cases of Study

Building upon the theoretical foundations laid out in the aforementioned propositions, we are poised to apply these concepts in the analysis of SHA2, SHACAL-2, and HMAC-SHA2. Furthermore, the scope of our attack methodology extends to encompass other algorithms employing SHA2-like boolean functions, including but not limited to SM3 and A5/1. The subsequent sections will delve into the procedure of our tailored attacks against various modes and algorithms.

4.1 Attack on SHA2 and its application

As stated in Figure 1, since attacker can not obtain the faulty output(such as the faulty values of $V_1, ..., V_{r-1}$) of compression function excepting the last hash function $F(V_{r-1}, M^{r-1})$, the target of our attack on SHA2 is the last compression

Algorithm 3 Recovery of initial vector V_{r-1} and message block M^{r-1} in compression function $F(V_{r-1}, M^{r-1})$

```
Require: The correct and faulty hash values denoted as $\{A, ..., H\}$ and
    $\{A^j, ..., H^j\}$ for any $j \in \mathbb{Z}_N$ and under 62 random word fault targets.
Ensure: $V_{r-1}(= \{a_0, ..., h_0\})$ and $M^{r-1}$
  1. According to proposition 1, recovering the last round of correct and faulty
  inputs $\{a_{R-1}, b_{R-1}, ..., h_{R-1}\}$ and $\{a^j_{R-1}, b^j_{R-1}, ..., h^j_{R-1}\}$ , where there exist
  $T$ indeterminate bits in any one of $a_{R-1}, ..., b_{R-1}$.
  2. guess the leftmost $T$ bits for any one of $a_{R-1}, ..., d_{R-1}$, and obtain $2^T$
  different $a_{R-1}, ..., d_{R-1}$.
  3. for each guessed $a_{R-1}, ..., d_{R-1}$
      3.1. According to proposition 2, recovering $V_{r-1}$ and $M^{r-1}$.
      3.2. if $F(V_{r-1}, M^{r-1}) + V_{r-1} = \{A, ..., H\}$
          return $M^{r-1}$ and $V_{r-1}$.
  return The recovery is failed.
```

function F. As illustrated in Figure 1, given that the attacker lacks access to the faulty outputs generated by the intermediate compression functionsexcept for the final compression function $F(V_{r-1}, M^{r-1})$, our attack strategy on the SHA2 algorithm focuses on $F(V_{r-1}, M^{r-1})$. This limitation necessitates focusing on this critical stage due to the unavailability of faulty information from preceding groups.

Therefore, with regard to SHA2 family including SHA224, SHA256, SHA384, and SHA512, only the final message block M^{r-1} and the corresponding initial state V_{r-1} can be recovered using Algorithm 3. Despite this constraint, several attack scenarios remain viable, as detailed below.

– In dynamic token systems, SHA2 is adopted to compress seed key and identifier, and finally input the left or right half of hash value. In this case, we can leverage Lemma 3 instead of Lemma 2 as in the context of SHA224 and SHA384. Consequently, when there exists a single message block for hash function, the seed key can indeed be recovered through our attack.
– For identity authentication mechanisms where a user's password, ID, and salt are concatenated and hashed to generate a verification hash, our attack method allows for the recovery of the ID, salt, and hash value. Under circumstances where the concatenated data fits into a single message block, the password can also be exposed via our fault attack.
– SHA2 plays a pivotal role in public key cryptography systems, often employed to hash messages or keys. For instance, in determining ECDSA and EdDSA signature schemes, the private key and message serve as inputs to a hash function or a key derivation function based on hash function. Should an attacker manage to acquire the output of this process, the security of these signatures is equally vulnerable to the threats posed by faults in SHA2 itself. Meanwhile, SHA2 is prominently recommended as a critical component within post-quantum cryptographic systems, serving roles such as a

pseudo-random number generator and for the compression of key information, among other applications. Its inherent security is intrinsically linked to the robustness of post-quantum cryptography, particularly in hash-based post-quantum cryptographic schemes where it plays an indispensable role. In essence, any compromise to SHA2's security directly impacts the security of these dependent cryptographic systems.

4.2 Forgery Attack on HMAC-SHA2

As stated in [21,22], an almost universal forgery attack can be conducted successfully under the results derived by our fault attacks.

As shown in Figure 2, there are two hash functions are calculated and the message block can be selected arbitrarily in the first hash function. The detailed process of the forgery attack is depicted in Figure 5, closely resembling the forgery attacks in [21,22]. First, input a single message block M^0 to calculate the correct HMAC value. Next, according to the proposed propositions, induce fault injections into the last compression function(highlighted in yellow) using the same message block M^0. Through the subsequent deductions in Algorithm 3, the attacker manages to recover the padded value resulting from $H(IV, \bar{k} \oplus ipad||M^0)$(i.e., M^r in Figure 2) as well as the initial value derived from $F(IV, \bar{k} \oplus opad)$(i.e., V_1 in Figure 2). Finally, after successfully recovering these intermediate values, the attacker gains the ability to forge the HMAC value for any message M whose first message block is M^0. This constitutes an almost universal forgery attack, demonstrating the vulnerability under such specific conditions.

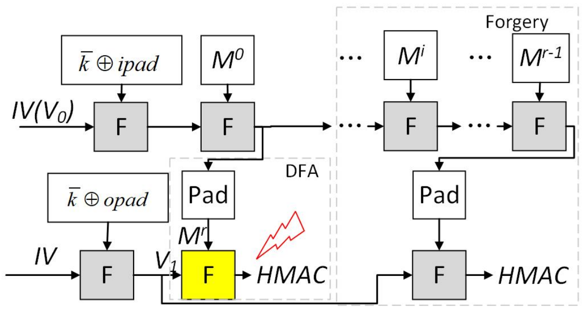


Fig. 5. Almost universal forgery attack on HMAC

4.3 Differential fault attack on SHACAL-2

SHACAL-2 is a block cipher algorithm derived from the compression function of SHA256, characterized by a block size of 256 bits and a key length of up to 512 bits. Developed under the auspices of the NESSIE program, it has been adopted for securing electronic systems. As depicted in Figure 6, with the exception of the absence of the "final addition" operation, the modified compression function F' of SHACAL-2 has the same transformation as in the F function of SHA256 (referenced in Figure 1), executing an identical sequence of 64 rounds denoted as f. In SHACAL-2, the message block in SHA256 serves as the encryption key, while the initial vector V_0 in SHA256 is treated akin to the plaintext, both being fed into the F' function as inputs.

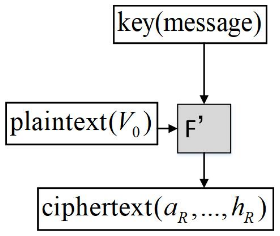


Fig. 6. Structure of SHACAL-2

Our proposed attack is not only viable against SHACAL-2 but also notably more accessible to execute. Initially, leveraging Proposition 1 and inducing random fault injections, we are able to recover the last input register values $\{a_{R-1}, ..., h_{R-1}\}$. Subsequently, according to Proposition 2, we can successfully retrieve both the plaintext (corresponding to the initial vector IV or V_0 in SHA2) and the key (which corresponds to the message block in SHA2).

Furthermore, due to the absence of a final addition step(as shown in Figure 3) before outputting the result, where the internal state registers $\{a_R, ..., h_R\}$ are directly exposed, every boolean differential value σ^j for any j defined in Lemma 4 becomes known. This means that the recovery of variable x in Lemma 4 does not rely on the randomness of e, but solely depends on the known σ^j via equation (2). As a consequence, the proposed method offers a notably increased probability of accurately reconstructing the left half of the registers $a_{R-1}, ..., h_{R-1}$ with a relatively lower number of fault injections required.

4.4 Extensions on Other Algorithms Having Similar Boolean Functions

To the best of our knowledge, several algorithms incorporate similar boolean functions, specifically Maj and Ch, such as SM3 [25], A5/1, and A2U2 [26], among others. Considering that our primary analytical scope here pertains to hash functions, we confine our study to the security resilience of SM3 against the proposed attack in this context. The examination and analysis of the boolean functions employed within stream cipher algorithms will be reserved as a topic for future, in-depth research.

As depicted in Figure 7, SM3 exhibits a structural resemblance to SHA256. It has boolean functions FF_j and GG_j which correspond to Maj and Ch respectively, where

$$FF_j(X,Y,Z) = (X\&Y) \vee (X\&Z) \vee (Y\&Z) \quad 16 \leq j \leq 63$$

and

$$GG_j(X,Y,Z) = (X\&Y) \vee (X'\&Z) \quad 16 \leq j \leq 63.$$

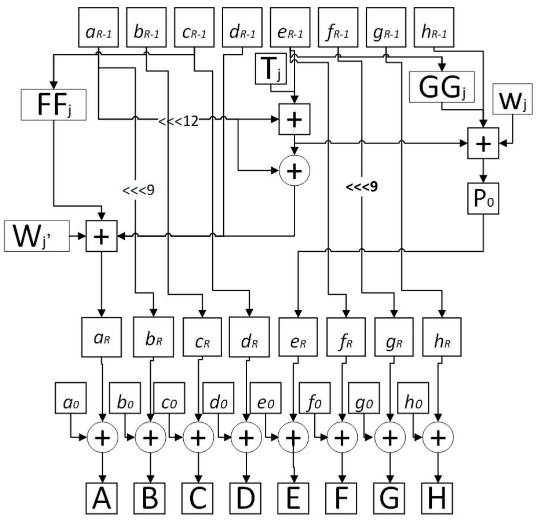


Fig. 7. Structure of SM3

The distinct between FF_j and $Maj(GG_j$ and $Ch)$ is using the operation $\vee$ not $\oplus$. The critical distinction between the function FF_j(or GG_j) and the function Maj(or Ch) is rooted in their employment of a logical OR operation ($\vee$) as opposed to the bitwise XOR operation ($\oplus$). However, upon closer deduction, we have ascertained that despite their apparent operational differences, these functions $FF_j(GG_j)$ and $Maj(Ch)$(or similar constructs) are indeed functionally equivalent. The detailed deduction is as follow.

$$\begin{aligned}
&(XY) \oplus (XZ) \oplus (YZ) \\
&= (XYZ' \vee XY'Z) \oplus (YZ) \\
&= (XYZ' \vee XY'Z)\,(YZ)' \vee (XYZ' \vee XY'Z)'\,(YZ) \\
&= (XYZ' \vee XY'Z)\,(Y' \vee Z') \vee (X' \vee (YZ' \vee Y'Z)')\,(YZ) \\
&= (XYZ' \vee XY'Z) \vee (X'YZ \vee YZ) \\
&= XYZ' \vee XY'Z \vee YZ \\
&= Y\,(XZ' \vee Z) \vee XY'Z \\
&= Y\,(Z \vee X) \vee XY'Z \\
&= XY \vee (Y \vee XY')\,Z \\
&= XY \vee XZ \vee YZ
\end{aligned} \tag{21}$$

$$\begin{aligned}
&(XY) \oplus (X'Z) \\
&= (XY)(X'Z)' \vee (XY)'(X'Z) \\
&= (XY \vee XYZ') \vee (X'Z \vee X'Y'Z) \\
&= XY \vee X'Z
\end{aligned} \tag{22}$$

By the deduction of equations (21) and (22), we can find that Maj is equal to FF_j and Ch is equal to GG_j. Therefore, akin to the attacks (differential fault analysis) instantiated on SHA2 in Sections 4.1 and 4.3, our proposed fault attack remains viable for both SM3 and its encryption counterpart. However, a significant difference arises: unlike SHA2, where an addition operation is performed between the initial input and the final round output registers before computing the hash value, SM3 utilizes XOR operations instead. This distinction becomes critical when considering scenarios like HMAC, where the initial input to the compression function is unknown. Consequently, it renders the execution of a almost forgery attack against SM3 impractical under such conditions.

In conclusion, our proposed attack methodology presents a multitude of potential attack instances, thereby posing a genuine threat to algorithms that incorporate the same boolean functions Maj and Ch. This highlights the significance of understanding and addressing these vulnerabilities in cryptographic systems utilizing similar functional constructs.

5 Experiments

In this section, we implemented a simulation of the random fault model and carried out the attack to empirically assess the feasibility and effectiveness of our proposed method. For clarity and brevity, we concentrated on validating Lemma 2 and Algorithm 2, which are central in recovering the right and left halves of the final round input for SHA256 and SHA512, respectively. It is important to emphasize that although not explicitly detailed here, other hash algorithms such as SHA224/384 and SM3 also adhere to a comparable operational structure, suggesting that similar attack strategies could be applicable to these instances as well.

Firstly, we systematically simulated fault injection experiments N times, where N ranged from 1 to 30, specifically targeting register f_{R-1} as depicted in Figure 4. For each unique value of N(corresponding to x axis), we executed 1000 attack instances based on Lemma 2 to attempt the recovery of e_{R-1}. Subsequently, we computed and contrasted both the empirical and theoretical success rates(corresponding to y axis) of recovering e_{R-1}. As presented in Figure 8, the red and blue lines depict the respective success rates of recovering e_{R-1} for SHA256 and SHA512. The dashed lines denote theoretical success rates, while solid lines represent experimental rates. The results indicate that the practical success rate slightly surpasses its theoretical counterpart. It is worth highlighting that the empirical success rate approaches an almost perfect 100% for both SHA256 and SHA512 when N reaches 15, which are achievable in genuine experimental settings. Moveover, it is worthy to note that all the attacks based on Lemma 2 such as the attacks recovering $a_{R-1} \oplus c_{R-1}$ and $a_{R-1} \oplus b_{R-1}$, are fundamentally identical. In addition, apart from the requirement of double fault targets, the attacks that rely on Lemma 3 exhibit an equivalent success rate to those based on Lemma 2. Therefore, we will not delve into the specifics here.

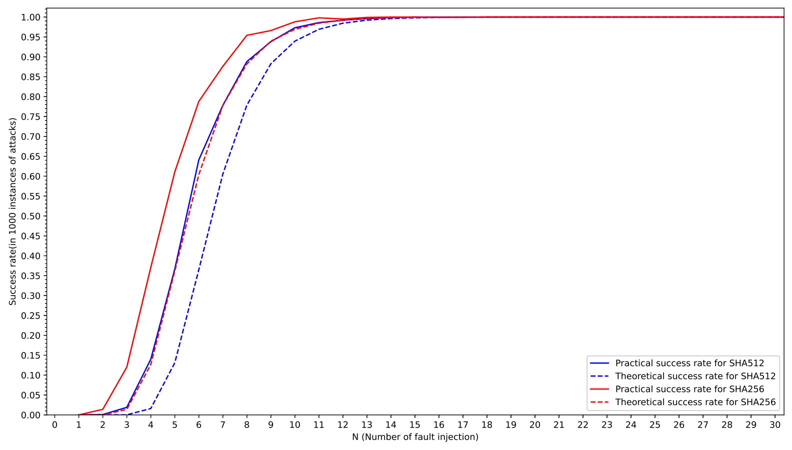


Fig. 8. Success rate of recovering e in Lemma 2 when performing N fault injections

Next, we proceeded to validate the practical feasibility and success rate of Algorithm 2, which fundamentally relies on Lemma 2 and Lemma 4 to recover the left half of the final round input. Similarly, we designated b_r and $c_r(r \in R-1, ..., R-4)$ as 8 targets sequentially and simulated $1 \sim 50(i.e., N = 1 \sim 50)$ random fault injections for each target. Subsequently, we executed 1000 repetitive instances of the attack in Algorithm 2 for each N. The experimental and theoretical results are shown in Figure 9. Similarly, the red and blue lines depict the respective success rates of SHA256 and SHA512. Dashed lines denote theoretical success rates, while solid lines represent experimental rates (experimental success rate for short) for successfully recovering the lower 31 bits of registers in Algorithm 2 for SHA256 or the lower 63 bits for SHA512, depending on N and e(see Lemma 4). Meanwhile, dash-dot lines indicate the experimental success rates (recovered success rate for short) of recovery being depending on N, i.e., the success rate of the attack when the consecutive bit triplet $\{e_{i+2}, e_{i+1}, e_i\}$ within the binary representation of e does not assume either the all-zero state ($\{0, 0, 0\}$) or the all-one state ($\{1, 1, 1\}$).

We can see that the experimental success rates of recovering all the lower 31/63 bits for SHA256 and SHA512 are jumping around 97% and 95% when N is lager than 9 respectively, while the theoretical average success rate is up to 99% when N is lager than 17/20 for SHA256/SHA512. The recovered success rates are nearly up to 100% when N is larger than 15, which are slightly larger than the theoretical average success rates. The above results reveal a slight divergence between the experimental and theoretical success rates. This discrepancy mainly stems from the fact that our theoretical calculations are based on an expectation of the number of known σ_i^j values, which assumes a random e following a binomial distribution with a probability of 1/2 over N fault injection trials ($Bin(1/2, N)$). However, in actual experiments, we utilized only eight unique randomly generated e values, which may account for the observed marginal difference. Moreover, when in the instances that the recovered bits are correct but their quantity falls short of $n - 1$ with probability 3% for SHA256(or 7% for SHA512), the average value of T(the average number of unrecoverable bits) consistently remains at 3, and its maximum value is 5. It implies that in the most disadvantageous experimental scenarios of our attacks, which have a relatively low likelihood (approximately 5% for SHA512 and 3% for SHA256), typically 3 bits would need to be deduced or guessed. On the other hand, under more common circumstances with a significantly higher probability of occurrence (around 95% for SHA512 and 97% for SHA256), it is predominantly the most significant bit alone that requires estimation.

In conclusion, leveraging the random fault model, we achieve a near-perfect success rate of approximately 100% in recovering every register within the right half of the $(R - 1)$-th input as the value of N approaches 15. Simultaneously, when the number N of fault injection instances reaches 15, and the number of bits needing to be guessed, denoted as T, generally amounts to 1(around 95% for SHA512 and 97% for SHA256) and rarely exceeds a maximum of 5 with an average value of 3(around 5% for SHA512 and 3% for SHA256), we achieve similarly impressive recovery success rates nearing 100% (refer to the recovered success rates depicted in Figure 9) for a register within the left half of the $(R - 1)$-th round input. Moreover, there are 928(i.e., $62N - 2$) fault injection instances required for recovering the message block and initial vector in the final compression function of SHA2.

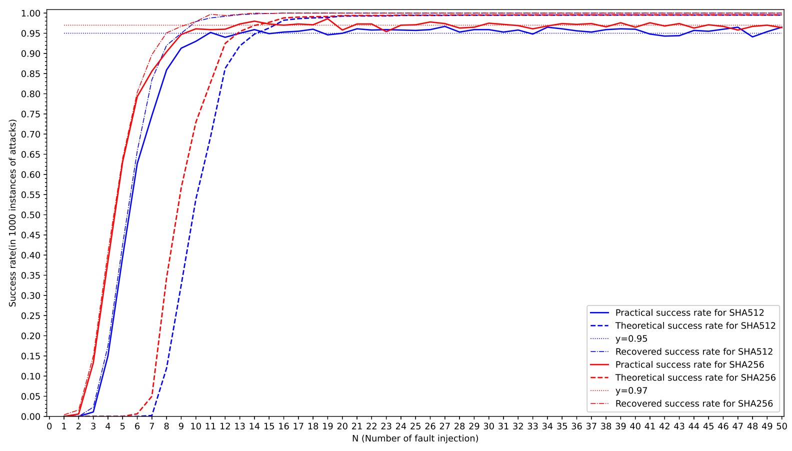


Fig. 9. The success rate of recovering the left half of the final round input following 8*N fault injections

6 Conclusion

In view of the complexity of boolean functions in the compression function of SHA2, the security of SHA2 remains ambiguous when facing differential fault attacks. This study has found some distinct differential fault properties of boolean function in SHA2, and thereby proposed a new differential fault attack which can be effectively applied on SHA2, HMAC-SHA2, SHACAL-2 and other algorithms like SM3. Through rigorous theoretical proof and experimental verification, we have demonstrated that our attacks posed real threat to SHA2 with overwhelming success probability. Finally, leveraging around 928 faulty outputs of SHA2 hash function, theoretically, facilitate an almost certain and comprehensive recovery of the last message block and its corresponding initial vector with an approximate probability approaching 100%.

The findings of our research reveal the essential characteristic of boolean functions in SHA2 and its threat on the security of SHA2, which not only extend the existing body of knowledge in the field of fault attack but also provide valuable insights for traditional differential analysis. We observed that the algorithms having similar boolean functions $Ch(x, y, z)$ and $Maj(x, y, z)$ still is vulnerable to our attack. Furthermore, in the compression function of SHA3, there are indeed multiple boolean functions. The question that we intend to delve into further is whether these boolean functions exhibit differential characteristics under random word(64 bits) fault model. This investigation aims to assess the resilience of SHA3 against potential attacks exploiting such differential properties. In addition, extending our analytical approach to perform a comprehensive examination of stream cipher algorithms such as A5/1 and A2U2 holding similar boolean functions, would be a valuable subject for further research.

However, it should be noted that despite these contributions, there are certain limitations to our attack, including lack of a practical verification targeting devices implementing SHA2. These areas open up opportunities for further investigation and refinement.

References

1. National Institute of Standards and Technology (NIST): Secure Hash Standard (SHS). Federal Information Processing Standards Publication 180-2, U.S. Department of Commerce, National Institute of Standards and Technology (2002) with Change Notice including SHA-224.
2. Biham, E., Joux, A.: SHACAL-2: A 256-Bit Block Cipher. In: Fast Software Encryption — FSE 2001. Volume 2040 of Lecture Notes in Computer Science., Springer (2001) 30–45
3. Bellare, M., Canetti, R., Krawczyk, H.: Keying Hash Functions for Message Authentication. Journal of Cryptology **9**(3) (07 1996) 185–211
4. Wang, X., Yu, H., Yin, Y., Dai, X.: Finding Collisions in the Full SHA-1. In: Advances in Cryptology - CRYPTO 2005. Volume 3621 of Lecture Notes in Computer Science., Springer (2005) 17C36
5. Stevens, M., Karpman, P., Peyrin, T.: Differential Collisions for SHA-1. In: Annual International Cryptology Conference (CRYPTO). Volume 8042 of Lecture Notes in Computer Science. (2013) 343–365
6. Stinson, D.: Some Birthday Attacks on Hash Functions. In: Advances in Cryptology CRYPTO '95. Volume 963 of Lecture Notes in Computer Science. (1995) 199–214
7. Bertoni, D., Daemen, J., Peeters, M., Assche, G.V.: Related-key differential attacks on reduced-round SHA-256. In: Fast Software Encryption (FSE) 2005. Volume 3557 of Lecture Notes in Computer Science. (2005) 154–167
8. Mcevoy, R.P., Tunstall, M., Murphy, C.C., Marnane, W.P.: Differential power analysis of hmac based on sha-2, and countermeasures. In: Information Security Applications, International Workshop, Wisa, Jeju Island, Korea, August, Revised Selected Papers. (2007)
9. Fouque, P.A., Leurent, G., Denis, R., Valette, F.: Practical electromagnetic template attack on hmac. In: International Workshop on Cryptographic Hardware and Embedded Systems (CHES), Springer-Verlag (2009)
10. Oswald, D.: Side-channel attacks on sha-1-based product authentication ics. In: Smart Card Research and Advanced Applications: 14th International Conference, CARDIS 2015, Bochum, Germany, November 4-6, 2015. Revised Selected Papers. Volume 9958 of Lecture Notes in Computer Science., Springer International Publishing (2016) 13–29
11. Gebotys, C.H., White, B.A., Mateos, E.: Preaveraging and carry propagate approaches to side-channel analysis of hmac-sha256. ACM Transactions on Embedded Computing Systems (TECS) **15**(1) (2016) 1–19
12. Samwel, N., Batina, L., Bertoni, G., Daemen, J., Susella, R.: Breaking ed25519 in wolfssl. In: Topics in CryptologyCCT-RSA 2018: The Cryptographers' Track at the RSA Conference 2018, San Francisco, CA, USA, April 16-20, 2018, Proceedings, Springer International Publishing (2018) 1–20

13. Li, R., Li, C., Gong, C.: Differential fault analysis on shacal-1. In: 2009 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC), IEEE (2009)
14. Hemme, L., Hoffmann, L.: Differential fault analysis on the sha1 compression function. In: 2011 Workshop on Fault Diagnosis and Tolerance in Cryptography, IEEE (2011) 54–62
15. Shoufan, A.: A fault attack on a hardware-based implementation of the secure hash algorithm sha-512. In: 2013 International Conference on Reconfigurable Computing and FPGAs (ReConFig), IEEE (2013) 1–7
16. Wei, Y., Li, L., Li, R., Li, C.: Differential fault analysis on shacal-2. Journal of Electronics & Information Technology **32**(2) (2010) 318–322
17. Shen, X.: Differential Fault Attacks on SHACAL-2 and the MD5 Encryption Modes. PhD thesis, National University of Defense Technology (2014)
18. Bagheri, N., Ghaedi, N., Sanadhya, S.K.: Differential fault analysis of sha-3. In: International Conference on Cryptology in India (INDOCRYPT), Springer, Cham (2015) 253–269
19. Luo, P., Fei, Y., Zhang, L., et al.: Differential fault analysis of sha3-224 and sha3-256. In: Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC), IEEE (2016) 4–15
20. Jeong, K., Lee, Y., Sung, J., Hong, S.: Security analysis of hmac/nmac by using fault injection. Journal of Applied Mathematics **2013** (2013)
21. Hao, R., Li, B., Ma, B., Song, L.: Algebraic fault attack on the sha-256 compression function. International Journal of Research in Computer Science **4**(2) (2014) 1–9
22. Nejati, S., Horáček, J., Gebotys, C., Ganesh, V.: Algebraic fault attack on sha hash functions using programmatic sat solvers. In: Principles and Practice of Constraint Programming - 24th International Conference, CP 2018, Lille, France, August 27-31, 2018, Proceedings. Volume 11015 of Lecture Notes in Computer Science., Springer International Publishing (2018) 737–754
23. Luo, P., Athanasiou, K., Fei, Y., Wahl, T.: Algebraic fault analysis of sha-3. In: Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017, IEEE (2017) 151–156
24. Luo, P., Athanasiou, K., Fei, Y., Wahl, T.: Algebraic fault analysis of sha-3 under relaxed fault models. IEEE Transactions on Information Forensics and Security **13**(7) (2018) 1752–1761
25. State Cryptography Administration of the People's Republic of China: Information Security Technology: SM3 Cryptographic Hash Algorithm. China National Standard GB/T 32907-2016, National Technical Committee for Information Security Standardization (2016) In Chinese. English Title translated by Author.
26. David, F., Mathieu, F., Ranasinghe, D.C., Larsen, T.: A2U2: a stream cipher for printed electronics RFID tags. In: 2011 IEEE International Conference on RFID, IEEE (2011)