On Corrective Patterns for the SHA-2 Family

Philip Hawkes[1], Michael Paddon[1], and Gregory G. Rose[1]

Qualcomm Australia, Level 3, 230 Victoria Rd, Gladesville, NSW 2111, Australia
{phawkes,mwp,ggr}@qualcomm.com

Abstract. The Secure Hash Standard (SHS) [3] includes hashing algorithms denoted SHA-n, $n \in 224, 256, 384, 512$ for producing message digests of length n. These algorithms are based on a common design, sometimes known as SHA-2, that consists of a message schedule and a register. The most successful attacks on the SHA algorithms are Chabaud-Joux differential collisions [1, 2, 4, 5, 7], which are based on finding a corrective pattern for the register. Previous analysis of the SHA-2 algoritms [4] indicated that, for all SHA-2 algorithms, the best corrective pattern has probability 2^{-66}. We find that the complexity of obtaining a collision is 2^{39} when the register state is unknown. Of this complexity, a factor of 2^9 corresponds to conditions on the internal state that must be satisfied, and a factor of 2^{30} corresponds to 30 bits of internal state that must be guessed correctly in order to generate a collision. When the register state is known (as is the case when generating a hash) then the guessed bits are known and the complexity is reduced to 2^9.
The simple analysis of the message schedule in [4] determines limits on the probability of collision for SHA-2, and was sufficient at that time to conclude that the algorithms resist the attacks. In [4] the claimed complexity is compared against the birthday attack bound of $2^{n/2}$. However, the corrective pattern can be converted into a second pre-image attack for which the complexity should be greater than 2^n. When accounting for the complexity of 2^9 per corrective pattern, the previous analysis of the message schedule yields lower bounds on the complexities 2^{27} for SHA-224/256 and 2^{45} for SHA-224/256. These complexities are significantly less than the 2^n bound. It is no longer certain that SHA-2 resists this attack. More detailed analysis of the message schedule is required.
Keywords: SHA-256, second pre-image attack.

1 Introduction

A hash function (or hash algorithm) is a cryptographic algorithms that is used to generate a message digest of fixed length n from a message of arbitrary length. The value of n is called the security parameter n, as it indicates the intended strength of the algorithm. A hash function with a security parameter n must provide resistance to three classes of attacks [6]:

Collision Attack In a collision attack, the attacker finds any two messages $M, M^* \neq M$ such that $h(M) = h(M^*)$. A secure hash function resists a collision attack if the complexity of the attack is $O(2^{n/2})$.

First Pre-image Attack In a first pre-image attack, the attacker is specified an output y, and the attacker must find a message M such that $h(M) = y$. A hash function resists a second pre-image attack if the complexity of the attack is approximately $O(2^n)$.

Second Pre-image Attack In a first pre-image attack, the attacker is specified message M, and the attacker must find another message $M^* \neq M$ such that $h(M) = h(M^*)$. A hash function resists a second pre-image attack if the complexity of the attack is $O(2^n)$.

The Secure Hash Standard (SHS) [3] defines five standard hash algorithms denoted SHA-1, SHA-224, SHA-256, SHA-384 and SHA-512 that produce message digests of length $n = 160$, 224, 256, 384 and 512 bits respectively. SHA-1 is the current, de-facto world-wide standard. However, advances in computing power have led to the need for stronger keys, and stronger hash functions. Consequently, the latter four algorithms are likely to become the de-facto world-wide standards of the future. These four algorithms are based on a common design that has become known as SHA-2.

The algorithms SHA-224 and SHA-256 differ from each other only in two aspects: the algorithms use distinct initial constants; and SHA-224 truncates the message digest to the 224 leftmost bits. The algorithms SHA-384 and SHA-512 algorithms are also identical, with the exception of distinct initial constants and the truncated output of SHA-384. This means that it is convenient to refer to SHA-224/256 and SHA-384/512. To save space when providing the details for SHA-224/256 and SHA-384/512, the details for SHA-384/512 are provide in parentheses. For example, writing "SHA-2 is based on 32-bit (resp. 64-bit) words" indicates that SHA-224/256 is based on 32-bit words and SHA-384/512 is based on 64-bit words.

The SHA algorithms have four phases; padding, parsing, message scheduling and register update. The updated register is called the *accumulating register* in this paper. The message schedule uses the padded and parsed message to generate a expanded sequence of inputs to the accumulating register; the expanded sequence of inputs is used during the register update.

Theus far, the most successful analysis of SHA algorithms resulted from the search for Chaboud-Joux differential collisions [1, 2, 4, 5, 7]. Of particular note are the collisions for SHA-0 (a pre-curser to SHA-1) found by Joux [5] and the collisions for MD4, MD5, HAVAL-128 and RIPEMD (pre-cursers to SHA-0) found by Wang et al [7].

This type of analysis first finds a high-probability corrective pattern for the accumulating register; that is, given one sequence of inputs to the accumulating register, the analysis finds another sequences of inputs such that the accumulating register results in equal values for both sequences of inputs. The analysis then analyzes the message schedule in an attempt to find two messages such that the message schedule will result in high-probability differential collisions being input to the accumulating register.

1.1 Results of this Paper

New Probabilities for Gilbert-Handschuh corrective patterns. Gilbert and Handschuh [4] looked for a high probability corrective patterns for SHA-2; where a corrective pattern is a sequence of XOR differences between two sequences of input words such that processing each sequence could result in equal values of the accumulation register. They claim that these corrective patterns are optimal in the sense that the probability of a collision is maximized for these corrective patterns. We agree with their conclusion that the differential is optimal. However, we have found that the probability of a collision is much higher than the probability 2^{-66} obtained by Gilbert and Handschuh. By transforming the XOR-differences into addition-differences, we find that the complexity of obtaining a collision is 2^{39} when the register state is unknown. Of this complexity, a factor of 2^9 corresponds to conditions on the internal state that must be satisfied, and a factor of 2^{30} corresponds to 30 bits of internal state that must be guessed correctly in order to generate a collision. When the register state is known (as is the case when generating a hash) then the guessed bits are known and the complexity is reduced to 2^9. Interestingly, this probability applies for all SHA-2 algorithms, independent of word size.

Our understanding is that the collisions on SHA-0 [5] and the collisions for MD4, MD5, HAVAL-128 and RIPEMD [7] were also obtained using addition-differences.

The corrective pattern yields a pre-image attack. The attack proposed in [4] is simple collision attack in which the attacker can specify all the details of the two messages that hash to equal message digests. However, the corrective pattern proposed herein can be used in a second pre-image attack, in which one of the messages is specified and the attacker generates a second message that hashes to an equal message digest. This is an important distinction, as the previous claims of resistance to the attack used the complexity bound of $2^{n/2}$ rather than the bound 2^n that applies for second pre-image attacks. In other words, the authors of [4] claimed resistance to the attack because the complexity exceeded the $2^{n/2}$ bound, but that was based on the asumption that the attack could only be performed as a collision attack. Now that that the attack is a second pre-image attack, the previous bounds must be increased to 2^n: that is, SHA-2 can only be considered secure if the complexity exceeds the 2^n bound.

Simple analysis of the Message Schedule is Insufficient. Gilbert and Handschuh computing a lower limit on the complexity of a Chabaud-Joux attack using the corrective pattern probability 2^{-66}. They claimed that at least three (resp. five) independent corrective patterns are required to perform a Chabaud-Joux attack when accounting for the message scheduling. When using the corrective pattern probability of 2^{-39}, the computed lower bound on the complexity becomes 2^{117} (resp. 2^{195});... assuming that the initial state is unknown. When the initial value of the register states are known, then the probability drops to 2^{-9} (resp. 2^{-45}). These complexities are significantly lower than the bounds required to prove resistance to second pre-image attacks, thus suggesting that SHA-2 may

not resist the attack. However, this lower bound is based on a simple analysis of the message schedule, and the bound is unrealistically optimistic. We cannot conclude (from these lower bound) whether SHA-2 is sufficiently secure or not. A more detailed analysis of the message schedule is required in order to obtained more accurate estimates for the complexity of a Chabaud-Joux attack. We have not yet attempted such an analysis, but recommend this as a worthwhile venture for any cryptologist.

Organization of this paper. The remainder of Section 1 introduces the necessary notation. Section 2 describes SHA-2. Section 3 discusses the corrective patterns of [4]. Section 4 discusses our goals in searching for corrective patterns, and Section 5 considers the relationship between XOR-differences and addition-differences. Section 6 then performs a detailed analysis of the corrective patterns. The complexity is determined in Section 7.

1.2 Notation

SHA-2 is based on 32-bit (resp. 64-bit) words. Within each word, the most significant bit(MSB) is the leftmost bit while the least significant bit (MSB) is the rightmost bit. For any word X, $\widehat{X}$ denotes the value of X with the MSB set to zero.

The i-th bit of a word a is denoted $a[i]$. SHA-2 uses two bit-wise operators: "$\wedge$" represents the bitwise AND operation with $(a \wedge b)[i] = a[i]b[i]$, $0 \leq i \leq n$; and "$\oplus$" represents the bitwise exclusive-OR operation with $(a \oplus b)[i] = a[i] \oplus b[i]$, $0 \leq i \leq w$, where w denotes the relevant word size.

The bit-wise complement of x (equal to $2^w - 1 - x$) is denoted x'. The function $ROTR^r(X)$ produces a word of the same size as X, but with the bits rotated cyclically to the right by r positions. That is, if $Y = ROTR^r(X)$ and the word size is w, then $Y[i] = X[i + r (\text{mod } w)]$, $0 \leq i \leq w$. The function $SHR^r(X)$ produces a word of the same size as X, but with the bits shifted (non-cyclically) to the right by r positions, with the remaining left-most bits filled with zeroes. That is, if $Y = SHR^r(X)$, then: $Y[i] = X[i + r]$ for $i < w - r$; and $Y[i] = 0$ for $w - r \leq i \leq w - 1$.

Finally, for our analysis we will denote the Hamming weight of x (that is, the number of one's in the binary representation of x) by $|x|$.

2 Description of SHA-2

SHA-2 consists of four phases: padding, parsing, message scheduling and register update.

Padding: Suppose that the length of the message, M, is l bits. Append the bit 1 to the end of the message, followed by k zero bits, where k is the smallest, non-negative solution to the equation $l + k + 1 \equiv 448$ (mod512) (resp. $l + k + 1 \equiv 896$ (mod1028)). Then append the 64-bit (resp. 128-bit) block that is equal to

the number l expressed using a binary representation. The length of the padded message should now be a multiple of 512 (resp. 1024) bits.

Parsing: The padded message is parsed into N 512-bit (resp. 1024-bit) blocks, $M^{(1)}, M^{(2)}, \ldots, M^{(N)}$. The 512 (resp. 1024) bits of the input block are expressed as sixteen 32-bit (resp. 64-bit) words. The first 32 bits (resp. 64 bits) of message block i are denoted $M_0^{(i)}$, the next 32 bits (resp. 64 bits) are $M_1^{(i)}$, and so on up to $M_{15}^{(i)}$.

Message Scheduling: The message schedule is applied to each message block individually. The message schedule first assigns the message words $M_0^{(i)}, \ldots, M_{15}^{(i)}$ to the values of the input words $W_0, \ldots, W_{15}$. The remainder of the input words $W_{16}, \ldots, W_{63}$ (resp. $W_{16}, \ldots, W_{79}$) are determined using the recurrence formula:

$$W_i = \sigma_1(W_{i-2}) + W_{i-7} + \sigma_0(W_{i-15}) + W_{i-16},$$

where σ_0, σ_1 are linear functions. For SHA-224/256, the functions σ_0, σ_1 act on 32-bit inputs and produce 32-bit outputs:

$$\sigma_0(X) = ROTR^7(X) \oplus ROTR^{18}(X) \oplus SHR^3(X),$$
$$\sigma_1(X) = ROTR^{17}(X) \oplus ROTR^{19}(X) \oplus SHR^{10}(X).$$

For SHA-384/512, the functions σ_0, σ_1 act on 64-bit inputs and produce 64-bit outputs:

$$\sigma_0(X) = ROTR^1(X) \oplus ROTR^8(X) \oplus SHR^7(X),$$
$$\sigma_1(X) = ROTR^{19}(X) \oplus ROTR^{61}(X) \oplus SHR^6(X).$$

Register Update: The accumulation register has 8 words of state A, B, C, D, E, F, G, H. For the first block of the message, these words are initialized to predetermined constants. For the remaining blocks of the message, the words are initialized to the intermediate hash value that results from from the preceding message block. Following initialization, 64 rounds (resp 80 rounds) of the compression function are applied to the expanded input sequence $\{W_i\}$. The t-th round of the compression function modifies the accumulation register using input word W_i and pre-determined constant K_i as input. The compression function uses addition modulo 2^{32} (resp. 2^{64}) and four non-linear functions:$CH, MJ, \Sigma_0, \Sigma_1$. SHA-224/256 uses functions with 32-bit inputs and 32-bit outputs, defined as:

$$CH(X,Y,Z) = (X \wedge Y) \oplus (X' \wedge Z);$$
$$MJ(X,Y,Z) = (X \wedge Y) \oplus (Y \wedge Z) \oplus (Z \wedge X);$$
$$\Sigma_0(X) = ROTR^2(X) \oplus ROTR^{13}(X) \oplus ROTR^{22}(X);$$
$$\Sigma_1(X) = ROTR^6(X) \oplus ROTR^{11}(X) \oplus ROTR^{25}(X).$$

SHA-384/512 uses functions with 64-bit inputs and 64-bit outputs, with CH and MJ defined as above and Σ_0, Σ_1 defined as:

$$\Sigma_0(X) = ROTR^{28}(X) \oplus ROTR^{34}(X) \oplus ROTR^{39}(X);$$
$$\Sigma_1(X) = ROTR^{14}(X) \oplus ROTR^{18}(X) \oplus ROTR^{41}(X).$$

The compression function modifies the accumulation register according to the following algorithm:

$$
\begin{aligned}
T1_i &= H +_i \Sigma_1(E_i) + CH(E_i, F_i, G_i) + K_i + W_i;\\
T2_i &= \Sigma_0(A_i) + MJ(A_i, B_i, C_i);\\
H_{i+1} = G_i;\quad G_{i+1} &= F_i;\quad F_{i+1} = E_i;\quad E_{i+1} = D_i + T1_i;\\
D_{i+1} = C_i;\quad C_{i+1} &= B_i;\quad B_{i+1} = A_i;\quad A_{i+1} = T1_i + T2_i.
\end{aligned}
$$

After all of the 64 (resp. 80) input words have been input to the accumulating register, the resulting values of the state are added modulo 2^{32} (resp. 2^{64}) to the initialized values of the state. These values become the new intermediate hash value. If this is the last message block, the new intermediate hash value is output as the resulting message digest. Otherwise, the algorithm proceeds to updating the register using the next message block.

3 Gilbert-Handschuh Corrective Patterns

The Chabaud-Joux approach to finding collisions divides the task into two separate analyses:

> "First one considers the injection in one of the words W_i of a ... difference, and one identifies the corresponding corrective patterns, i.e. sets of differences in the subsequent words W_{i+j} that cancel with high probability the resulting differences in the state registers after a few rounds.Then we search for low weight sequences of [corrective] patterns satisfying the linear recurrence of the message schedule." [4]

The corrective pattern represents two sequences of inputs $\{W_i\}$ and $\{W_i^*\}$ to the compression function. The *first run* is the sequence of internal state values and function outputs that result when inputting the sequence $\{W_i\}$ and the second run is the sequence of internal state values and function outputs that result when inputting the sequence $\{W_i^*\}$. For any input word, internal state value or function output, then we represent the value during the first run using X and the value during the second run by X^*. The *XOR-difference* in this value X is defined as $\Delta X = X \oplus X^*$.

Gilbert and Handschuh employed the following strategy for finding high probability corrective patterns.

> "Obviously, ... the best strategy is to inject a one bit difference in a given message word W_i, and for each consecutive round, to disable the propagation into the register A by appropriate corrective patterns of the next message words. Allowing for more than a single bit difference is not realistic as each Σ function automatically multiplies the Hamming weight of the difference by three in each step, and trying to match these bit locations using several initial difference bits implies a fatal decrease of the probability to obtain such differential collisions. Therefore

we believe that no other strategy can provide a sufficiently low weight perturbation pattern, hence an acceptable overall collision probability. The pattern has been obtained in a straightforward manner by setting the following equalities: let W_i be the word containing the perturbative one-bit difference. Then we define the next eight word differences by: $W_{i+1} = \Sigma_1(W_i) \oplus \Sigma_0(W_i)$; $W_{i+2} = \Sigma 1(\Sigma 0(W_i))$; $W_{i+3} = 0$; $W_{i+4} = W_i$; $W_{i+5} = \Sigma_1(W_i) \oplus \Sigma_0(W_i)$; $W_{i+6} = 0$; $W_{i+7} = 0$; $W_{i+8} = W_i$." [4]

The corrective pattern of Gilbert and Handschuh [4] is shown in table 1. In this table we have represented

$$\alpha 1 = \Delta W_i, \quad \beta 3 = \Sigma_0(\Delta W_i), \quad \gamma 3 = \Sigma_1(\Delta W_i)), \quad \epsilon 9 = \Sigma 1(\Sigma 0(\Delta W_i)).$$

The number after the Greek letter is there to remind us of the Hamming weight of the difference. For SHA-224/256

$$\begin{aligned}\epsilon 9 = & ROTR^1(\alpha 1) \oplus ROTR^6(\alpha 1) \oplus ROTR^8(\alpha 1) \\ & \oplus ROTR^{13}(\alpha 1) \oplus ROTR^{15}(\alpha 1) \oplus ROTR^{19}(\alpha 1) \\ & \oplus ROTR^{24}(\alpha 1) \oplus ROTR^{27}(\alpha 1) \oplus ROTR^{28}(\alpha 1),\end{aligned}$$

while for SHA-384/512

$$\begin{aligned}\epsilon 9 = & ROTR^5(\alpha 1) \oplus ROTR^{11}(\alpha 1) \oplus ROTR^{16}(\alpha 1) \\ & \oplus ROTR^{42}(\alpha 1) \oplus ROTR^{46}(\alpha 1) \oplus ROTR^{48}(\alpha 1) \\ & \oplus ROTR^{52}(\alpha 1) \oplus ROTR^{53}(\alpha 1) \oplus ROTR^{57}(\alpha 1).\end{aligned}$$

j	ΔH	ΔG	ΔF	ΔE	ΔD	ΔC	ΔB	ΔA	ΔW_{i+j}
0	-	-	-	-	-	-	-	-	$\alpha 1$
1	-	-	-	$\alpha 1$	-	-	-	$\alpha 1$	$\beta 3 \oplus \gamma 3$
2	-	-	$\alpha 1$	$\beta 3$	-	-	$\alpha 1$	-	$\epsilon 9$
3	-	$\alpha 1$	$\beta 3$	-	-	$\alpha 1$	-	-	-
4	$\alpha 1$	$\beta 3$	-	-	$\alpha 1$	-	-	-	$\alpha 1$
5	$\beta 3$	-	-	$\alpha 1$	-	-	-	-	$\beta 3 \oplus \gamma 3$
6	-	-	$\alpha 1$	-	-	-	-	-	-
7	-	$\alpha 1$	-	-	-	-	-	-	-
8	$\alpha 1$	-	-	-	-	-	-	-	$\alpha 1$

Table 1. The low-weight corrective pattern proposed by Gilbert and Handschuh [4]. The values in row i shows the differences in the state prior to applying the compression function to W_i.

The probability of the corrective pattern is the product of two probabilities: the probability of the XOR-differences proceeding correctly through the addition operations; and the probability of the XOR-differences proceeding correctly

through the CH and MJ operations. (The XOR-differences proceed through the Σ functions with probability one, since these functions are linear.)

Our results agree with [4] regarding the probability 2^{-18} to account for the CH and MJ operations. However, [4] claims a probability 2^{-48} to account for the addition operations, resulting in a total probability of 2^{-66}. In Section 7, we show that corrective patterns can be constructed that have probablity 2^{-39} when the initial state is unknown and $2-9$ when the intial state is known. This increase in probability is obtained by using addition-differences and the noting that any differences in the MSB always propagates correctly through the addition operation.

3.1 Analysis of the Message Schedule

Gilbert and Handschuh [4] analyzed the message schedule in order to obtain some lower bounds on the complexity of finding collision. Their goal was to show that the complexity of finding a collision exceeds the complexity of a birthday attack: $2^{n/2}$. In their analysis of SHA-224/256, they noted that

> "...at least 3 different patterns must be combined to follow the correct message schedule.
>
> If two of these patterns are applied, the probability to obtain a differential collision becomes lower than 2^{-132} whereas the complexity of a birthday attack on SHA-256 only represents 2^{128} computations on average." [4]

They concluded that SHA-256 resists the Chabaud-Joux attack. In their analysis of SHA-384/512, they noted that

> "...at least 5 different patterns must be combined to follow the correct message schedule.
>
> If four of these patterns are applied, the probability to obtain a differential collision becomes lower than 2^{-264} whereas the complexity of a birthday attack on SHA-512 only represents 2^{256} computations on average." [4]

They concluded that SHA-384/512 also resists the Chabaud-Joux attack.

A triple (IHV, M, M^*) denotes a intermediate hash value, a first run message block M, and a second run message block M^* that has been constructed from M^* so the input words $\{W_i\}$ and $\{W_i^*\}$ differ according to a specified corrective pattern.

The probabilities of [4] are computed for the average case. That is, those authors assume that the message block M is uniformly distributed and the intermediate hash value is uniformly distributed. Under these assumptions, [4] claims that the probability of a collision in the output is smaller than 2^{-132} (resp. 2^{-264}). Alternatively, one may read this as saying that more than 2^{132} (resp. 2^{264}) triples are required to find a triple that results in a collision. Quite reasonably, the authors of [4] compared this complexity to the complexity of

$2^{256/2=128}$(resp. $2^{512/2=256}$)) of a simple collision attack, and concluded that SHA-2 resists the attack.

Unfortunately, this does not tell the whole story. Let $1/N$ denote the probability of the collision. If, given a message, it is possible for an attacker to construct N triples for which M is a block of the original message, then then there is likely to be a collision in at least one of these triples. This means that the attacker will have has performed a second pre-image attack with complexity N. Moreover, if SHA-2 is to be considered secure, then the complexity of the attack should be greater than the second pre-image bound of 2^n which is significantly greater than the birthday attack bound of $2^{n/2}$.

For example, suppose that the probabilities obtained in [4] are accurate, and the probability of a collision is 2^{-132} (resp. 2^{-264}). This suggests that if an attacker can obtain 2^{132} (resp. 2^{264}) triples from a given message, then the attacker can perform a second pre-image attack with complexity 2^{132} (resp. 2^{264}). This complexity is far below the second pre-image bound of 2^{256} (resp. 2^{512}). This result would be a serious concern if not that the probability is an upper bound. That is, the analysis of [4] obtained unrealistic upper bounds on the probability by using an extremely simple analysis. We doubt that there are any differences in a message block that result in only 3 (resp. 5) occurrences of the highest probability corrective pattern. So the analysis of [4] only shows that the complexity of a second pre-image attack is greater than 2^{132} (resp. 2^{264}); and this does not reveal if SHA-2 is secure or otherwise.

It is interesting to note that SHA-2 will not accept messages of length great than 2^{64} (resp. 2^{128}) which may (in the few of some) nullify the previous comments.

Remember that the process complexities listed above correspond to the corrective pattern found by Gilbert and Handschuh. The complexity is reduced significantly when using the higher probability of our corrective patterns. The data complexity is also reduced. In short, the case for SHA-2 begins to look doubtful.

4 Goals in finding a Corrective Pattern

The Gilbert-Handschuh corrective pattern [4] causes a collision when certain conditions are satisfied. Those authors do not detail what these conditions are and it is unclear whether these conditions would be placed on the input words or the register states. Our corrective pattern examines the compression function in detail to specify how the attacker can cause a collision.

We begin by presuming that the attacker has been specified a multi-block message (the first-run message) and corresponding message digest, for which the attacker must find another message (the second-run message) that hashes to the same message digest. Note that the message schedule is deterministic, so an attacker will know the value of all first-run input words $\{W_i\}$.

There are three classes of actions the attacker performs in our attack:

1. The attacker *assumes* that certain bits in the register (at a particular round) satisfy a given condition when hashing the first-run message.
2. The attacker *guesses* the value of certain bits in the register (at a particular round) when hashing the first-run message.
3. Based on the assumed and guessed values of bits, and the known values of the first-run input words, the attacker *determines* the value of second-run input words $\{W_i^*\}$ that will cause a collision.

The corrective pattern results in equal register states when the assumed conditions are satisfied and when the values assigned to the guessed bits are correct. We will presume that corrective patterns may then combined to determine one or more second-run message blocks that may result in a collision (that is, the second-run message blocks may result in the same intermediate hash value as the first-run message blocks). The collision occurs when the set of assumed conditions are satisfied and when the values assigned to the set of guessed bits are correct. If n_g denote the number of bits that are guessed (in hashing the first-run message), then there are 2^{n_g} second-run message blocks that are determined from each first-run message block. *At this point in time, we have not conducted the analysis of the message schedule to determine how to find this second-run message block; we have only analyzed the compression function.*

Suppose the attacker has no information about the initial register states. The attacker attempts to find a collision at every message block until the attacker finds a message block for which the assumed conditions have been satisfied. Thus, if p_a denotes the probability that all the assumed conditions are satisfied (in hashing the first-run message) then the data complexity (the number of message blocks required) is $\frac{1}{p_a}$. At every one of these message blocks, the attacker attempts every possible combination of guesses in the off-chance that the assumed conditions are satisfied at that message block. The resulting amount of computation (the process complexity) is $\frac{1}{p_a} \cdot 2^{n_g}$ since both the assumed conditions and the guessed bits must be correct.

Note that the assumed conditions and guessed bits relate to the first-run message. When being used as hash function then the attacker knows the initial register states. Thus, once the first-run message is specified, then the attacker knows the value of every bit of every register state in every every of every message block. The implications are worrying.

- When the initial register states are unknown, then the attacker tests each message block under the assumption that the the conditions are satisfied and guesses bits at every message block. However, when the initial register states are known, then the attack can simply observe if the conditions are satisfied and only guesses bits at those message blocks where the conditions are satisfied. In other words, the attacker dos not need to guess the bits at every message block. This alone reduces the process complexity to $\frac{1}{p_a} + 2^{p_g}$.
- Further, the attacker now knows the value of the guessed bits, so the attacker only needs to trial one guess for the bits. Thus, once the attacker finds a first-run message block where the assumptions are satisfied, then the attacker

needs only construct one second-run message block. This reduces the process complexity to $\frac{1}{p_a}$.

Hence, the complexity is dominated by two factors: the complexity of determining the second-run message blocks that might cause a collision, and the probability of satisfying the corresponding assumed conditions. We begin by focussing on determining corrective patterns that minimize the number of assumptions. At this point in time, we have spent very little time studying how we determine the second-run message blocks that might cause a collision.

5 Addition-XOR Corrective Patterns

Our approach is somewhat different to [4]. Rather than considering only XOR-differences, we also consider addition-differences $\delta X = X^* - X$. Both differences are useful. An analysis of the Σ functions with respect to XOR differences is simple, while analysis of Σ functions with respect to addition differences is quite complex. However, the remainder of the compression function is best analyzed using addition differences. Lemma 1 describes what information is sufficient relate XOR-differences and addition-differences.

Lemma 1. If $\Delta X = \lambda$, then δX can be determined if $X[i]$ is known for every $i < 31$ (resp. $i < 63$) such that $\lambda[i] = 1$.

Proof. For every i such that $\lambda[i] = 1$, it is known that $X^*[i] = X[i] - 1$. For every i such that $\lambda[i] = 0$, it is known that $X^*[i] = X[i]$ Thus

$$\begin{aligned} X &= \sum_{i:\lambda[i]=1} X[i] \cdot 2^i + \sum_{i:\lambda[i]=0} X[i] \cdot 2^i, \\ X^* &= \sum_{i:\lambda[i]=1} X^*[i] \cdot 2^i + \sum_{i:\lambda[i]=0} X^*[i] \cdot 2^i, \\ &= \sum_{i:\lambda[i]=1} (1 - X[i]) \cdot 2^i + \sum_{i:\lambda[i]=0} X[i] \cdot 2^i, \\ \delta X = X^* - X &\equiv \sum_{i:\lambda[i]=1} (1 - 2X[i]) \cdot 2^i, \end{aligned}$$

which depends only on the bits $X[i]$ such that $\lambda[i] = 1$. For $i = 31$ the value of $(1 - 2X[i]) \cdot 2^i \equiv 2^{31} \pmod{2^{32}}$ is independent of $X[31]$. Similarly, for $i = 63$ the value of $(1 - 2X[i]) \cdot 2^i \equiv 2^{63} \pmod{2^{64}}$ is independent of $X[63]$. Thus δX can be determined if $X[i]$ is known for every $i < 31$ (resp. $i < 63$) such that $\lambda[i] = 1$. □

Note that the attacker can turn a "guess" into an "assumed condition" by only testing one guess for that value. In the following description I have minimized the sum of the number of assumed conditions and the number of guessed bits. Tables 3 and 4 describe the possible corrective patterns that result when some of the internal values have been assumed.

6 The Corrective Patterns

j		H	G	F	E	D	C	B	A	δW_{i+j}	Ass.	Gue.
0	Δ	-	-	-	-	-	-	-	-	$\delta E_{i+1} = \delta A_{i+1} = \delta_0$	$\widehat{\alpha 1}$	$\widehat{\alpha 1}$
	δ	-	-	-	-	-	-	-	-			
1	Δ	-	-	-	$\alpha 1$	-	-	-	$\alpha 1$	$-\delta\Sigma_1(E_{i+1}) - \delta\Sigma_0(A_{i+1})$	$\alpha 1$,$\alpha 1$	$\widehat{\beta 3}$
	δ	-	-	-	δ_0	-	-	-	δ_0	$= -\delta_{1,1} - \delta_{1,2}$	$\widehat{\beta 3}$	$\widehat{\gamma 3}$
2	Δ	-	-	$\alpha 1$	$\beta 3$	-	-	$\alpha 1$	-	$-\delta CH_{i+2} - \delta\Sigma_1(E_{i+2})$	$\alpha 1$	α_1
	δ	-	-	δ_0	$-\delta_{1,2}$	-	-	δ_0	-	$= -\delta_{2,1} - \delta_{2,2}$	β_3	$\widehat{\epsilon 9}$
3	Δ	-	$\alpha 1$	$\beta 3$	-	-	$\alpha 1$	-	-	$-\delta CH_{i+3} = -\delta_3$	$\alpha 1$	$\alpha 1$
	δ	-	δ_0	$-\delta_{1,2}$	-	-	δ_0	-	-			$\beta 3$
4	Δ	$\alpha 1$	$\beta 3$	-	-	$\alpha 1$	-	-	-	$-\delta H_{i+4} - \delta CH_{i+4}$	$\widehat{\alpha 1}$	$\beta 3$
	δ	δ_0	$-\delta_{1,2}$	-	-	δ_0	-	-	-	$= -\delta_0 - \delta_4$		
5	Δ	$\beta 3$	-	-	$\alpha 1$	-	-	-	-	$-\delta H_{i+5} - \delta CH_{i+5}$	-	$\alpha 1$
	δ	$-\delta_{1,2}$	-	-	δ_0	-	-	-	-	$-\delta\Sigma_1(E_{i+5})$		$\widehat{\gamma 3}$
										$= \delta_{1,2} - \delta_{5,1} - \delta_{5,2}$		
6	Δ	-	-	$\alpha 1$	-	-	-	-	-	$-\delta CH_{i+6} = -\delta_6$	-	$\alpha 1$
	δ	-	-	δ_0	-	-	-	-	-			
7	Δ	-	$\alpha 1$	-	-	-	-	-	-	$-\delta CH_{i+7} = -\delta_7$	-	$\alpha 1$
	δ	-	δ_0	-	-	-	-	-	-			
8	Δ	$\alpha 1$	-	-	-	-	-	-	-	$-\delta H_{i+8} = -\delta_0$	-	-
	δ	δ_0	-	-	-	-	-	-	-			

Total Assumed	$4\|\alpha 1\| + 2\|\widehat{\alpha 1}\| + 1\|\beta 3\| + 1\|\widehat{\beta 3}\| + 0\|\widehat{\gamma 3}\| + 0\|\widehat{\epsilon 9}\|$
Total Guessed	$5\|\alpha 1\| + 1\|\widehat{\alpha 1}\| + 2\|\beta 3\| + 1\|\widehat{\beta 3}\| + 2\|\widehat{\gamma 3}\| + 1\|\widehat{\epsilon 9}\|$

Table 2. A summary of the corrective pattern using addition based differences.

Table 2 summarizes the corrective pattern. Recall that $\alpha 1$ represents a single-bit difference, with $\beta 3 = \Sigma_0(\alpha 1)$, $\gamma 3 = \Sigma_1(\alpha 1)$ and $\epsilon 9 = \Sigma_1(\Sigma_0(\alpha 1) \wedge \alpha 1') = \Sigma_1(\beta 3 \wedge \alpha 1')$. For each round, the rows beginning with Δ and δ list the XOR differences and addition differences the respectively. The third last column (headed by "δW_{i+j}") lists the additive-differences δW_{i+j} required for the corrective pattern. The second last column (headed by "Ass.") lists the bit positions where the attacker must assume conditions on register states, while the last column (headed by "Gue.") lists the bit positions where the attacker must guess bits. The last two columns indicate the bits positions at which a condition has been applied and bit positions where bits have been guessed.

Tables 3, 4 and 5 describe the corrective pattern in detail. The following notation is used:

- For each round, the first two rows (the rows beginning with Δ and δ) list the XOR differences and addition differences respectively.
- Each difference ($\alpha 1$, $\beta 3$, $\gamma 3$, $\epsilon 9$) is allocated a row for showing the values of the register states at the bits where the bit differences is one. The first set

of rows (with M@...) show the assigned values during the first run and the second set of rows (with M^*@...) show the assigned values during the second run.
  - The values **J**, **L**, **Q**, $\mathbf{q}_3$, $\mathbf{q}_4$ **R** and **S** correspond to guesses of bits in the first run that change in the second run. These values must be correctly guessed by the attacker for the corrective pattern to be successful.
  - The values ξ_i, $i \in \{2, 3, 5, 6, 7\}$ and ψ_i, $i \in \{3, 4\}$ correspond to guesses of bits that are equal in both the first and second run. These values must be correctly guessed by the attacker for the corrective pattern to be successful.
  - The values X, Y, Z do not need to be guessed: they are shown for the purposes of indicating bits of register state values that are equal.
- The remaining rows explain what conditions are required to be placed on the register values and function outputs. At each such row
  - The first column indicates if this condition places an assumption on the register values (denoted "Ass".), or if an state value must be guessed (denoted "Gue".) or if the internal values determine the value(s) for W_i (denoted "Det.").
  - The next column(s) indicates what values are assigned. We use the notation $X = U;\lambda$ to indicate that the attacker assigns either an internal condition(s) $X[i] = U[i]$, at the bit(s) where $\lambda[i] = 1$; or to indicate that the attacker assigns the guessed value(s) $U[i]$ to the value(s) $X[i]$ at the bit(s) where $\lambda[i] = 1$. Any relevant consequences of assigning this values are also noted.
  - The last two columns show the bit positions for which assumed condition are made (headed by "Ass.") and the bit positions for which bits which are guessed (headed by "Gue."). These are the important factors in determining the complexity
- The table is ended with a summary of the number of bits assumed and guessed for the rounds in that table.

The principle idea is to guess sufficient bits of the internal state during the first run, so that we can determine the addition-differences that would generate or cancel the XOR differences. While the table refers to input words $W_0, \ldots, W_8$ and the associated register states, the reader should note that the difference pattern can be applied to any sequence of input words $W_i, \ldots, W_{i+8}$. The notation $W_0, \ldots, W_8$ is used to keep the table at a reasonable width.

6.1 Detailed Explanation of Table 3: Rounds 0 to 2

Round 0

Guess $E_1 = \mathbf{J};\widehat{\alpha 1}$. The attacker wants to inject an XOR-difference $\Delta E_1 = \alpha 1$. Guessing the value of the bits of E_1 at the bit where $\widehat{\alpha 1}[i] = 1$ gives the attacker sufficient information to determine the corresponding addition difference δE_1 . We use the value $\mathbf{J}$ to denote the guess for these bits. *This step requires the attacker to guess bit position(s) indicated by* $\widehat{\alpha 1}$.

Determine Corresponding δE_1. Note that $E_1^* = \mathbf{J} \oplus \alpha 1 = \mathbf{J}';\widehat{\alpha 1}$. The addition difference δE_1, which we denote δ_0, is computed as $\delta_0 = E_1^* - E_1 = (\mathbf{J} \oplus \alpha 1) - \mathbf{J}$.

Determine required δW_0. Note that injecting any addition-difference χ between W_0 and W_0^* will cause the addition-difference χ to occur between between E_1 and E_1^*. Thus, to cause the addition difference δ_0 to occur between the pair E_1 and E_1^*, the attacker injects the difference $\delta W_0 = \delta_0$ between W_0 and W_0^*. Note that this will also have the effect of causing the addition-difference δ_0 between between A_1 and A_1^*.

Assume that the condition $A_1 = E_1;\widehat{\alpha 1}$ holds. If this condition holds, then

$$\Delta A_1 = A_1^* \oplus A_1 \;=\; (E_1 + \delta_0) \oplus E_1 = (E_1 \oplus \alpha 1) \oplus E_1 = \alpha 1.$$

If this condition does not hold, then the injected difference will result in more than one bit of difference in A_1, and the complexity of causing a collision will increase. *This step requires the attacker to make assumptions on bit position(s) indicated by* $\widehat{\alpha 1}$.

For the remaining rounds, the attacker tries to cancel out the internal differences in the register states by ensuring that $\delta A_{i+1} = \delta T1_i + \delta T2_i = 0$.

Round 1

Assume that internal conditions result in $\Delta CH_1 = 0$. Since $E_1^* = E_1 \oplus \alpha 1$, the CHOOSE function will choose the value of $F_1 = F_1^*;\alpha 1$ at one run and the value of $G_1 = G_1^*;\alpha 1$, at the other run. The attacker could guess that $G_1 = F_1;\alpha 1$, or the attacker could guess that $G_1 = F_1';\alpha 1$. In the first case, where the attacker guesses that $G_1 = F_1;\alpha 1$, then $\Delta CH_1 = \delta CH_1 = 0$, and the attacker does not need to guess the values of G_1 and F_1. In the second case, where the attacker guesses that $G_1 = F_1';\alpha 1$, then the attacker knows that $\Delta CH_1 = \alpha 1$, but the attacker cannot determine δCH_1 without also guessing either G_1 or F_1. So the second case requires the attacker to guess an additional bit of information. Consequently, we choose the first case in order to minimize the total amount of bits guessed and conditions assumed. Thus, in this step, the attacker assumes that $G_1 = F_1;\alpha$, (which also means that $G_1^* = F_1^*;\alpha 1$), which gives $\Delta CH_1 = 0$. We will denote the value of $G_1;\alpha$ by X: the attacker does not need to guess this value; we have just shown this value in the table to indicate that the values are equal. *This step requires the attacker to make assumptions on bit position(s) indicated by* $\alpha 1$.

		H_i	G_i	F_i	E_i	D_i	C_i	B_i	A_i	CH	Σ_1	$T1$	MJ	Σ_0	$T2$	E_{i+1}	A_{i+1}	Ass.	Gue.
0	Δ	-	-	-	-	-	-	-	-	-	-	?	-	-	-	$\alpha 1$	$\alpha 1$		
	δ	-	-	-	-	-	-	-	-	-	-	δ_0	-	-	-	δ_0	δ_0		
	$M@\alpha 1$			X	X			Y	Y			$\mathbf{J}$				$\mathbf{J}$	$\mathbf{J}$		
	$M^*@\alpha 1$			X	X			Y	Y			$\mathbf{J}'$				$\mathbf{J}'$	$\mathbf{J}'$		
	Gue.	$E_1 = \mathbf{J}; \widehat{\alpha 1}$								$\delta E_1 \stackrel{\text{def}}{=} \delta_0 = (\mathbf{J} \oplus \alpha) - \mathbf{J}$									$\widehat{\alpha 1}$
	Det.	$\delta W_0 = \delta E_1 = a$								$\Rightarrow \delta E_1 = \delta_0 \Rightarrow \delta A_1 = \delta_0$									
	Ass.	$A_1 = E_1 = \mathbf{J}; \widehat{\alpha 1}$								$\Rightarrow \Delta A_1 = \Delta E_1 = \alpha 1$								$\widehat{\alpha 1}$	
	Round 0:	$\delta W_0 = \delta E_1$											Ass. $\widehat{\alpha 1}$				Gue. $\widehat{\alpha 1}$		

		H_i	G_i	F_i	E_i	D_i	C_i	B_i	A_i	CH	Σ_1	$T1$	MJ	Σ_0	$T2$	E_{i+1}	A_{i+1}	Ass.	Gue.
1	Δ	-	-	-	$\alpha 1$	-	-	-	$\alpha 1$	-	$\gamma 3$	?	-	$\beta 3$	$\beta 3$	$\beta 3$	-		
	δ	-	-	-	δ_0	-	-	-	δ_0	-	$\delta_{1,1}$	$-\delta_{1,2}$	-	$\delta_{1,2}$	$\delta_{1,2}$	$-\delta_{1,2}$	-		
	$M@\alpha 1$		X	X	$\mathbf{J}$		Y	Y	$\mathbf{J}$							ξ_2	Y		
	$M@\beta 3$			Z	Z									$\mathbf{Q}$		$\mathbf{Q}'$			
	$M@\gamma 3$										$\mathbf{L}$								
	$M^*@\alpha 1$		X	X	$\mathbf{J}'$		Y	Y	$\mathbf{J}'$							ξ_2	Y		
	$M^*@\beta 3$			Z	Z									$\mathbf{Q}'$		$\mathbf{Q}$			
	$M^*@\gamma 3$										$\mathbf{L}'$								
	Ass.	$\Delta CH_1 = 0$:								$G_1 = F_1;\alpha 1 \Leftrightarrow F_0 = E_0 \stackrel{\text{def}}{=} X; \alpha 1$								$\alpha 1$	
		$\Delta MJ_1 = 0$:								$C_1 = B_1;\alpha 1 \Leftrightarrow B_0 = A_0 \stackrel{\text{def}}{=} Y; \alpha 1$								$\alpha 1$	
	Gue.	$\Sigma_1(E_1) = \mathbf{L}; \widehat{\gamma 3}$								$\Rightarrow \delta\Sigma_1(E_1) \stackrel{\text{def}}{=} \delta_{1,1} = (\mathbf{L} \oplus \gamma 3) - \mathbf{L}$									$\widehat{\gamma 3}$
		$\Sigma_0(A_1) = \mathbf{Q}; \widehat{\beta 3}$								$\Rightarrow \delta\Sigma_0(A_1^*) \stackrel{\text{def}}{=} \delta_{1,1} = (\mathbf{Q} \oplus \beta 3) - \mathbf{Q}$									$\widehat{\beta 3}$
	Det.	$\delta W_1 = -\delta\Sigma_1(E_1) - \delta\Sigma_0(A_1^*)$								$\Rightarrow \quad \delta W_1 = -\delta_{1,1} - \delta_{1,2}$ $\Rightarrow \ \delta E_2 = -\delta_{1,2},\ \delta A_2 = 0$									
	Ass.	$E_2 = \Sigma_0(A_1)' = \mathbf{Q}'; \widehat{\beta 3}$								$\Rightarrow E_2^* = \mathbf{Q},\ \Delta E_2 = \beta 3$								$\widehat{\beta 3}$	
	Round 1:	$\delta W_1 = -\delta\Sigma_1(E_1) - \delta\Sigma_0(A_1)$											Ass. $\alpha 1, \alpha 1, \widehat{\beta 3}$				Gue. $\widehat{\beta 3}, \widehat{\gamma 3}$		

		H_i	G_i	F_i	E_i	D_i	C_i	B_i	A_i	CH	Σ_1	$T1$	MJ	Σ_0	$T2$	E_{i+1}	A_{i+1}	Ass.	Gue.
2	Δ	-	-	$\alpha 1$	$\beta 3$	-	-	$\alpha 1$	-	ξ_2	$\epsilon 9$	-	-	-	-	-	-		
	δ	-	-	δ_0	$-\delta_{1,2}$	-	-	δ_0	-	$\delta_{2,1}$	$\delta_{2,2}$	-	-	-	-	-	-		
	$M@\alpha 1$	X	X	$\mathbf{J}$	ξ_2	Y	Y	$\mathbf{J}$	Y	$\mathbf{J};\xi_2$						ξ_3	Y		
	$M@\beta 3$		Z	Z	$\mathbf{Q}'$											ψ_3			
	$M@\epsilon 9$										$\mathbf{R}$								
	$M^*@\alpha 1$	X	X	$\mathbf{J}'$	ξ_2	Y	Y	$\mathbf{J}'$	Y	$\mathbf{J}';\xi_2$						ξ_3	Y		
	$M^*@\beta 3$		Z	Z	$\mathbf{Q}$											ψ_3			
	$M^*@\epsilon 9$										$\mathbf{R}'$								
	Ass.	$\Delta CH_2 = 0;\ \beta 3$								$G_2 = F_2;\beta 3 \Leftrightarrow E_1 = E_0 \stackrel{\text{def}}{=} Z;\ \beta 3$								$\beta 3$	
	Gue.	$E_2(= E_2^*) = \xi_2;\ \alpha 1$								$\Rightarrow CH_2 = \mathbf{J};\xi_2,\ CH_2^* = \mathbf{J}';\ \xi_2$ $\Rightarrow \delta CH_2 \stackrel{\text{def}}{=} \delta_{2,1} = (\mathbf{J} \oplus \xi_2) - \mathbf{J}.$									$\alpha 1$
	Ass.	$\Delta MJ_2 = 0$:								$A_2 = C_2 = Y;\ \alpha 1$								$\alpha 1$	
	Gue.	$\Sigma_1(E_2) = \mathbf{R};\ \epsilon 9$								$\Rightarrow \delta\Sigma_1(E_2) \stackrel{\text{def}}{=} \delta_{2,2} = (\mathbf{R} \oplus \epsilon 9) - \mathbf{R}$									$\widehat{\epsilon 9}$
	Det.	$\delta W_2 = -\delta CH_2 - \delta\Sigma_1(E_2)$								$\Rightarrow \delta W_2 = -\delta_{2,1} - \delta_{2,2}$									
	Round 2:	$\delta W_2 = -\delta CH_2 - \delta\Sigma_1(E_2)$											Ass. $\alpha 1, \beta 3$				Gue. $\alpha 1, \widehat{\epsilon 9}$		

Rounds 0 to 2: Assumed Conditions	$3\lvert\alpha 1\rvert + 1\lvert\widehat{\alpha 1}\rvert + 1\lvert\beta 3\rvert + 1\lvert\widehat{\beta 3}\rvert + 0\lvert\widehat{\gamma 3}\rvert + 0\lvert\widehat{\epsilon 9}\rvert$
Rounds 0 to 2: Bits Guessed	$1\lvert\alpha 1\rvert + 1\lvert\widehat{\alpha 1}\rvert + 0\lvert\beta 3\rvert + 1\lvert\widehat{\beta 3}\rvert + 1\lvert\widehat{\gamma 3}\rvert + 1\lvert\widehat{\epsilon 9}\rvert$

Table 3. Details of the corrective pattern for rounds 0 to 2.

Note 1. *If $\alpha 1 = 2^{w-1}$, then the attacker does not have to guess $G_2[i]$ or $F_2[i]$ in the second case in order to determine the corresponding additive difference. This means that, if $\alpha 1 = 2^{w-1}$, then the attacker may guess the value of $(G_2[i] \oplus F_2[i])$ (rather than assuming that $G_2[i] \oplus F_2[i] = 0$) as the attacker can detemine the addition-difference to cancel the resulting difference. Thus, if $\alpha 1 = 2^{w-1}$, then one of the assume conditions becomes a guessed bit.*

Assume that internal conditions result in $\Delta MJ_1 = 0$. Observe that $B_1 = B_1^*$, and $C_1 = C_1^*$. The attacker could guess that $B_1 = C_1;\alpha$, or the attacker could guess that $B_1 = C_1';\alpha$. In the first case, then the output of the MAJORITY function is the value (B_1) in both runs, and thus $\Delta MJ = \delta MJ = 0$; the attacker does not have to input a difference in W_1 in order to cancel this difference. In the second case, then the output of the MAJORITY function has $MJ_1 = \mathbf{J};\alpha$, and $MJ_1^* = \mathbf{J'};\alpha$ so $\Delta MJ_1 = \alpha$, and $\delta MJ_1 = \delta_0 = \delta T2_1$. In this case, the attacker will want a difference $\delta T1_1 = -\delta_0$, in order to have $\Delta A_1 = \delta A_1 = 0$. This will inject a difference between E_2 and E_2^*. Cancelling this difference in further rounds would add significant complexity. This would result in the attacker guessing further bits of state. Consequently, we choose the first case in order to minimize the total amount of bits guessed and conditions assumed. Thus, in this step, the attacker assumes that $B_1 = C_1;\alpha$, (which also means that $B_1^* = C_1^*;\alpha$), which gives $\Delta MJ_1 = 0$. We will denote the value of $B_1;\alpha$ by Y: the attacker does not need to guess this value; we have just shown this value in the table to indicate that the values are equal. *This step requires the attacker to make assumptions on bit position(s) indicated by $\alpha 1$.*

Guess $\Sigma_1(E_1) = \mathbf{L};\widehat{\gamma 3}$. Assuming that the assigned values are correct, then the attacker knows that $\Delta E_1 = \alpha 1$, and thus $\Delta\Sigma_1(E_1) = \Sigma_1(\Delta E_1) = \gamma 3$. The attacker needs to know the addition-difference $\delta\Sigma_1(E_1)$ so that the attacker can insert the necessary difference in δW_1 to cancel this difference out in the compression function. That is, the attacker needs to know the value of $\Sigma_1(E_1)$ at the bits where $\widehat{\gamma 3}[i] = 1$. Each bit of $\Sigma_1(E_1)$ has inputs from three bits, and the attacker has only assigned values to one of these bits. Consequently, the attacker needs to guess the value of the bits of $\Sigma_1(E_1)$ at the bits where $\widehat{\gamma 3}[i] = 1$. This gives the attacker sufficient information to determine the corresponding addition difference $\delta\Sigma_1(E_1)$. We use the value $\mathbf{L}$ to denote the guess for these bits. *This step requires the attacker to guess bit position(s) indicated by $\widehat{\gamma 3}$.*

Guess $\Sigma_0(A_1) = \mathbf{Q};\widehat{\beta 3}$. Assuming that the assigned values are correct, then the attacker knows that $\Delta A_1 = \alpha 1$, and thus $\Delta\Sigma_0(A_1) = \Sigma_0(\Delta A_1) = \beta 3$. The attacker needs to know the addition-difference $\delta\Sigma_0(A_1)$ so that the attacker can insert the necessary difference in δW_1 to cancel this difference out in the compression function. That is, the attacker needs to know the value of $\Sigma_0(A_1)$ at the bits where $\widehat{\beta 3}[i] = 1$ Each bit of $\Sigma_0(A_1)$ has inputs from three bits, and the attacker has only assigned values to one of these bits. Consequently, the attacker needs to guess the value of the bits of $\Sigma_0(A_1)$

at the bits where $\widehat{\beta 3}[i] = 1$. This gives the attacker sufficient information to determine the corresponding addition difference $\delta\Sigma_0(A_1)$. We use the value **Q** to denote the guess for these bits. *This step requires the attacker to guess bit position(s) indicated by* $\widehat{\beta 3}$.

Determine required δW_1. The attacker injects an addition-difference between W_1 and W_1^* in order to cancel the differences so that $\delta A_2 = 0$. Thus $\delta W_1 = -\delta\Sigma_1(E_1) - \delta\Sigma_0(A_1) = -\delta_{1,1} - \delta_{1,2}$. A by-product of injecting this difference is that $\delta T1_1 = -\delta_{1,2}$ and $\delta E_2 = -\delta_{1,2}$.

Assume that the condition $E_2 = \Sigma_0(A_1);\widehat{\beta 3}$ holds. If this condition holds, then

$$\Delta E_2 = E_2^* \oplus E_2 \;=\; (E_2 - \delta_{1,2}) \oplus E_2 = (\Sigma_0(A_1) \oplus \beta 3) \oplus \Sigma_0(A_1) = \beta 3.$$

If this condition does not hold, then the injected difference will result in more than one bit of difference in A_1, and the complexity of causing a collision will increase. *This step requires the attacker to make assumptions on bit position(s) indicated by* $\widehat{\beta 3}$.

Round 2

Assume the internal conditions give $\Delta CH_3 = 0;\beta 3$. That is, the attacker wishes to get $\Delta CH_2[i] = 0$, at the bits where $\beta 3[i] = 1$. The attacker knows that the CHOOSE function will choose the value of $F_2 = F_2^*;\beta 3$, at one run and the value of $G_2 = G_2^*;\beta 3$, at the other run. For the three bits where $\beta 3[i] = 1$, the attacker could guess that $G_2[i] = F_2[i]$, or the attacker could guess that $G_2[i] = F_2'[i]$. In the first case, where the attacker guesses that $G_2[i] = F_2[i]$, then $\Delta CH_2[i]$ and the attacker does not need to guess the values of $G_2[i]$ and $F_2[i]$ as there is no difference to cancel. In the second case ,where the attacker guesses that $G_2[i] = F_2'[i]$, then the attacker knows that $\Delta CH_2[i] = 1$, but the attacker cannot determine the corresponding addition-difference δCH_2 without also guessing either $G_2[i]$ or $F_2[i]$. So the second case requires the attacker to guess an additional bit of information. Consequently, for each bit where $\beta 3[i] = 1$, the we choose the first case in order to minimize the total amount of bits guessed and conditions assumed. Thus, in this step, the attacker assumes that $G_2 = F_2;\beta 3$, (which also means that $G_2^* = F_2^*;\beta 3$), which gives $\Delta CH_2 = 0;\beta 3$. We will denote the value of $G_2;\beta_3$ by Z: the attacker does not need to guess this value; we have just shown this value in the table to indicate that the values are equal. *This step requires the attacker to make assumptions on bit position(s) indicated by* $\widehat{\beta 3}$.

Guess $E_2 = \xi_2;\alpha 1$, and infer ΔCH_2 and δCH_2. Observe that $E_2 = E_2^*;\alpha$. The value of E_2 affects where the CHOOSE function outputs F_2 and F_2^* or G_2 and G_2^* in the first and second run respectively. Let i denote the bits position where $\alpha 1 = 1$.

- If $E_2[i] = 1$, then the CHOOSE function outputs $F_2[i] = \mathbf{J}[i]$, and $F_2^*[i] = \mathbf{J}'[i]$, in the first and second run respectively. Thus $\Delta CH_2 = \alpha 1$, and since the value of $CH_2[i]$ is known, the attacker also knows $\delta CH_2 = \delta_0$.

– If $E_2[i] = 0$, then the CHOOSE function outputs $G_2[i]$, and $G_2^*[i] = G_2[i]$, in the first and second run respectively. Thus $\Delta CH_2 = \delta CH_2 = 0$, and the attacker does not have to input a difference in W_2 in order to cancel this difference.

Thus, if $E_2 = E_2^* = \xi_2;\alpha 1$, then $\Delta CH_2 = \xi_2$, and $\delta CH_2 = 0$, when $xi_2 = 0$ and $\delta CH_2 = \alpha 1$ when $\xi_2 = \delta_0$. *This step requires the attacker to guess bit position(s) indicated by* $\alpha 1$.

Assume that internal conditions result in $\Delta MJ_2 = 0$. Observe that $A_2 = A_2^*$, and $C_2 = C_2^* = Y$. The attacker could guess that $A_2 = C_2;\alpha$, or the attacker could guess that $A_2 = C_2';\alpha$. In the first case, then the output of the MAJORITY function is the value (A_2) in both runs, and thus $\Delta MJ = \delta MJ = 0$; the attacker does not have to input a difference in W_2 in order to cancel this difference. In the second case, then the output of the MAJORITY function has $MJ_2 = \mathbf{J};\alpha$, and $MJ_2^* = \mathbf{J'};\alpha$, so $\Delta MJ_2 = \alpha$, and $\delta MJ_2 = \delta_0 = \delta T2_2$. In this case, the attacker will want a difference $\delta T1_2 = -\delta_0$, in order to have $\Delta A_2 = \delta A_2 = 0$. This will inject a difference between E_3 and E_3^*. Cancelling this difference in further rounds would add significant complexity. This would result in the attacker guessing further bits of state. Consequently, we choose the first case in order to minimize the total amount of bits guessed and conditions assumed. Thus, in this step, the attacker assumes that $A_2 = C_2 = Y;\alpha$, (which also means that $A_2^* = C_2^* = Y;\alpha$), which gives $\Delta MJ_2 = 0$. *This step requires the attacker to make assumptions on bit position(s) indicated by* $\alpha 1$.

Guess $\Sigma_1(E_2) = \mathbf{R};\widehat{\epsilon 9}$. Assuming that the assigned values are correct, then the attacker knows that $\Delta E_2 = \beta 3$ and thus $\Delta\Sigma_1(E_2) = \Sigma_1(\Delta E_2) = \epsilon 9$. The attacker needs to know the addition-difference $\delta\Sigma_1(E_2)$ so that the attacker can insert the necessary difference in δW_2 to cancel this difference out in the compression function. That is, the attacker needs to know the value of $\Sigma_1(E_2)$ at the bits where $\widehat{\epsilon 9}[i] = 1$, Each bit of $\Sigma_1(E_2)$ has inputs from three bits, and the attacker has only assigned values to one of these bits. Consequently, the attacker needs to guess the value of the bits of $\Sigma_1(E_2)$ at the bits where $\widehat{\epsilon 9}[i] = 1$. This gives the attacker sufficient information to determine the corresponding addition difference $\delta\Sigma_1(E_2)$. We use the value $\mathbf{R}$ to denote the guess for these bits. *This step requires the attacker to guess bit position(s) indicated by* $\widehat{\epsilon 9}$.

Determine required δW_2. The attacker injects an addition-difference between W_1 and W_1^* in order to cancel the differences so that $\delta A_3 = 0$. Thus $\delta W_2 = -\delta CH_2 - \delta\Sigma_1(E_2) = -\delta_{2,1} - \delta_{2,2}$. A by-product of injecting this difference is that $\delta T1_2 = 0$, and $\delta E_3 = 0$.

6.2 Detailed Explanation of Table 4: Rounds 3 to 5

Round 3

Guess the value of $E_3 = E_3^* = \xi_3;\alpha 1$. Let i denote the bits position where $\alpha 1 = 1$.

- If $E_3[i] = 1$, then the CHOOSE function outputs $F_3[i]$ and $F_3^*[i] = F_3[i]$ in the first and second run respectively. Thus $\Delta CH_3[i] = 0$, and the attacker does not have to input a difference in W_3 in order to cancel a difference.
- If $E_3[i] = 0$, then the CHOOSE function outputs $G_3[i] = \mathbf{J}[i]$ and $G_3^*[i] = \mathbf{J}'[i]$ in the first and second run respectively. Thus $\Delta CH_3[i] = \alpha 1[i]$, and since the value of $CH_3[i]$ is known, the attacker also knows that the corresponding addition-difference is δ_0.

Thus, if $E_3[i] = E_3^*[i] = \xi_3[i]$, then $\Delta CH_3 = \tau_3[i] = \xi_3'[i] \wedge \alpha 1$. Let δ_3 denote the corresponding addition-difference. We will use this information to compute δCH_3 at a later step *This step requires the attacker to guess bit position(s) indicated by* $\alpha 1$.

Guess the value of $E_3 = E_3^* = \psi_3;\beta 3$. Let i denote one of the three bit positions where $\beta 3[i] = 1$.

- If $E_3[i] = 1$, then the CHOOSE function outputs $F_3[i] = \mathbf{Q}[i]$, and $F_3^*[i] = \mathbf{Q}'[i]$, in the first and second run respectively. Thus $\Delta CH_3[i] = 1$, and since the value of $CH_3[i]$ is known, the attacker also knows the corresponding additive difference.
- If $E_3[i] = 0$, then the CHOOSE function outputs $G_3[i]$ and $G_3^*[i] = G_3[i]$, in the first and second run respectively. Thus $\Delta CH_3[i] = 0$, and the attacker does not have to input a difference in W_3 in order to cancel a difference.

Thus, if $E_3 = E_3^* = \psi_3;\beta 3$, then $\Delta CH_3 = \psi_3$. *This step requires the attacker to guess bit position(s) indicated by* $\beta 3$.

Compute δCH_3. After guessing $E_3 = E_3^* = \xi_3;\alpha 1$, and $E_3 = E_3^* = \psi_3;\beta 3$, the attacker has determined ΔCH_3 and knows the values of $CH[i]$ ad the bit positions where $\Delta CH_3 = 1$. Hence the attacker has enough information to determine $\delta CH_3 \stackrel{\text{def}}{=} \delta_3$, where

$$\delta CH_3 = (\mathbf{J}_3 \oplus \tau_3) - \mathbf{J}_3' + (\mathbf{Q} \wedge \psi_3) - ((\mathbf{Q} \wedge \psi_3) \oplus \psi_3).$$

Assume that the internal conditions result in $\Delta MJ_3 = 0$. Observer that $A_3 = A_3^*$, and $B_3 = B_3^*$. The attacker could guess that $A_3 = B_3;\alpha$, or the attacker could guess that $A_3 = B_3';\alpha$. In the first case, then the output of the MAJORITY function is the value (A_3) in both runs, and thus $\Delta MJ = \delta MJ = 0$; the attacker does not have to input a difference in W_3 in order to cancel this difference. In the second case, then the output of the MAJORITY function has $MJ_3 = \mathbf{J};\alpha$, and $MJ_3^* = \mathbf{J}';\alpha$, so $\Delta MJ_3 = \alpha$, and $\delta MJ_3 = \delta_0 = \delta T2_3$. In this case, the attacker will want a difference $\delta T1_3 = -\delta_0$, in order to have $\Delta A_3 = \delta A_3 = 0$. This will inject a difference between E_3

		H_i	G_i	F_i	E_i	D_i	C_i	B_i	A_i	CH	Σ_1	MJ	Σ_0	E_{i+1}	Ass.	Gue.
3	Δ	-	$\alpha 1$	$\beta 3$	-	-	$\alpha 1$	-	-	$\tau_3 \oplus \psi_3$	-	-	-	-		
	δ	-	δ_0	$-\delta_{1,2}$	-	-	δ_0	-	-	δ_3	-	-	-	-		
	$M@\alpha 1$	X	$\mathbf{J}$	ξ_2	ξ'_3	Y	$\mathbf{J}$	Y	Y	$\mathbf{J};\widehat{\xi_3}$				ξ_5		
	$M@\beta 3$	Z	Z	$\mathbf{Q}'$	ψ_3					$\mathbf{Q}';\psi'_4$				ψ'_4		
	$M^*@\alpha 1$	X	$\mathbf{J}'$	ξ_2	ξ'_3	Y	$\mathbf{J}'$	Y	Y	$\mathbf{J}';\widehat{\xi_3}$				ξ_5		
	$M^*@\beta 3$	Z	Z	$\mathbf{Q}$	ψ_3					$\mathbf{Q};\psi'_4$				ψ_4		
	Gue.	$E_3 = \xi_3;\alpha 1 \Rightarrow CH_3 = \mathbf{J};\tau_3,\ CH_3^* = \mathbf{J}';\tau_3$ where $\tau_3 = \xi'_3 \wedge \alpha 1$														$\alpha 1$
		$E_3 = \psi_3;\beta 3 \Rightarrow CH_3 = \mathbf{Q};\psi_3$ and $CH_3 = \mathbf{Q};\psi_3$														$\beta 3$
	Det.	$\delta CH_3 \stackrel{\text{def}}{=} \delta_3 = (\mathbf{J}_3 \oplus \tau_3) - \mathbf{J}'_3 + (\mathbf{Q} \wedge \psi_3) - ((\mathbf{Q} \wedge \psi_3) \oplus \psi_3)$														
	Ass.	$\Delta MJ_3 = 0 \Rightarrow A_3 = B_3 = Y;\alpha 1$													$\alpha 1$	
	Det.	$\delta W_3 = -\delta CH_3 = -\delta_3$														

	Round 3: $\delta W_3 = -\delta CH_3$	Ass. $\alpha 1$	Gue. $\alpha 1, \beta 3$

		H_i	G_i	F_i	E_i	D_i	C_i	B_i	A_i	CH	Σ_1	MJ	Σ_0	E_{i+1}	Ass.	Gue.
4	Δ	$\alpha 1$	$\beta 3$	-	-	$\alpha 1$	-	-	-	μ_4	-	-	-	$\alpha 1$		
	δ	δ_0	$-\delta_{1,2}$	-	-	δ_0	-	-	-	δ_4	-	-	-	δ_0		
	$M@\alpha 1$	$\mathbf{J}$	ξ_2	ξ_3	ξ_5	$\mathbf{J}$	Y	Y		μ_4		Y		$\mathbf{J}$		
	$M@\beta 3$	Z	$\mathbf{Q}'$	ψ_3	ψ_4					$\mathbf{Q}';\widehat{\mu_4}$						
	$M^*@\alpha 1$	$\mathbf{J}'$	ξ_2	ξ_3	ξ_5	$\mathbf{J}'$	Y	Y		μ_4		Y		$\mathbf{J}'$		
	$M^*@\beta 3$	Z	$\mathbf{Q}$	ψ_3	ψ_4					$\mathbf{Q};\widehat{\mu_4}$						
	Gue.	$E_4 = \psi_4;\beta 3 \Rightarrow CH_4 = \mathbf{Q}';\mu_4$ and $CH_4 = \mathbf{Q};\mu_4$ where $\mu_4 = \psi'_4 \wedge \beta 3$														$\beta 3$
	Det.	$\delta CH_4 \stackrel{\text{def}}{=} \delta_4 = (\mathbf{Q} \wedge \mu_4) - ((\mathbf{Q} \wedge \mu_4) \oplus \mu_4)$														
	Det.	$\delta W_4 = -\delta H_4 - \delta CH_4 \Rightarrow \delta W_4 = -\delta_0 - \delta_4$														
	Ass.	$E_5 = D_4 = \mathbf{J};\alpha 1 \Rightarrow \delta E_5 = \delta D_4 = \delta_0$													$\widehat{\alpha 1}$	

	Round 4: $\delta W_4 = -\delta H_4 - \delta CH_4$	Ass. $\widehat{\alpha 1}$	Gue. $\beta 3$

		H_i	G_i	F_i	E_i	D_i	C_i	B_i	A_i	CH	Σ_1	MJ	Σ_0	E_{i+1}	Ass.	Gue.
5	Δ	$\beta 3$	-	-	$\alpha 1$	-	-	-	-	τ_5	$\gamma 3$	-	-	-		
	δ	$-\delta_{1,2}$	-	-	δ_0	-	-	-	-	$\delta_{5,1}$	$\delta_{5,2}$	-	-	-		
	$M@\alpha 1$	ξ_2	ξ_3	ξ_5	$\mathbf{J}$	Y	Y							ξ_6		
	$M@\beta 3$	$\mathbf{Q}'$	ψ_3	ψ_4												
	$M@\gamma 3$										$\mathbf{S}$					
	$M^*@\alpha 1$	ξ_2	ξ_3	ξ_5	$\mathbf{J}'$	Y	Y							ξ_6		
	$M^*@\beta 3$	$\mathbf{Q}$	ψ_3	ψ_4												
	$M^*@\gamma 3$										$\mathbf{S}$					
	Det.	$\delta H_5 = -\delta_{1,2}$														
	Gue.	$F_5 = \xi_5;\alpha 1$. Attacker infers $CH_5;\alpha 1$ and $CH_5^*;\alpha 1$														$\alpha 1$
	Det.	$\Rightarrow\ \Delta CH_5 \stackrel{\text{def}}{=} \tau_5 = (\xi_5 \oplus \xi_3) \Rightarrow \delta CH_5 \stackrel{\text{def}}{=} \delta_{5,1} = CH_5^* - CH_5$														
	Gue. $\mathbf{S}$	$\Sigma_1(E_5) = \mathbf{S};\gamma 3 \Rightarrow \delta\Sigma_1(E_5) \stackrel{\text{def}}{=} \delta_{5,2} = (\mathbf{S} \oplus \gamma 3) - \mathbf{S}.$														$\widehat{\gamma 3}$
	Det.	$\delta W_5 = -\delta H_5 - \delta CH_5 - \delta\Sigma_1(E_5) = \delta_{1,2} - \delta_{5,1} - \delta_{5,2}$														

	Round 5: $\delta W_5 = -\delta H_5 - \delta CH_5 - \delta\Sigma_1(E_5)$	Ass.	Gue. $\alpha 1, \gamma 3$

	Rounds 3 to 5: Assumed Conditions	$1\|\alpha 1\| + 1\|\widehat{\alpha 1}\| + 0\|\beta 3\| + 0\|\widehat{\beta 3}\| + 0\|\widehat{\gamma 3}\| + 0\|\widehat{\epsilon 9}\|$
	Rounds 3 to 5: Bits Guessed	$2\|\alpha 1\| + 0\|\widehat{\alpha 1}\| + 2\|\beta 3\| + 0\|\widehat{\beta 3}\| + 1\|\widehat{\gamma 3}\| + 0\|\widehat{\epsilon 9}\|$

Table 4. Details of the corrective pattern for rounds 3 to 5.

and E_3^*. Cancelling this difference in further rounds would add significant complexity. This would result in the attacker guessing further bits of state. Consequently, we choose the first case in order to minimize the total amount of bits guessed and conditions assumed. Thus, in this step, the attacker assumes that $A_3 = B_3 = Y;\alpha$, (which also means that $A_3^* = B_3^* = Y;\alpha$), which gives $\Delta MJ_3 = 0$. *This step requires the attacker to make assumptions on bit position(s) indicated by* $\alpha 1$.

Determine required δW_3. The attacker injects an addition-difference between W_3 and W_3^* in order to cancel the differences so that $\delta A_4 = 0$. Thus $\delta W_3 = -\delta CH_3 = -\delta_3$. A by-product of injecting this difference is that $\delta T1_3 = 0$, and $\delta E_4 = 0$.

Round 4

Determine δH_4. Note that the difference in E_1 has now propagated to the top of the register so $\delta H_4 = \delta E_1 = \delta_0$.

Guess the value of $E_4 = E_4^* = \psi_4;\beta 3$. Let i denote the bits position where $\beta 3 = 1$.

- If $E_4[i] = 1$, then the CHOOSE function outputs $F_4[i]$ and $F_4^*[i] = F_4[i]$, in the first and second run respectively. Thus $\Delta CH_4[i] = 0$, and the attacker does not have to input a difference in W_4 in order to cancel a difference.
- If $E_4[i] = 0$, then the CHOOSE function outputs $G_4[i] = \mathbf{Q}'[i]$, and $G_4^*[i] = \mathbf{Q}[i]$, in the first and second run respectively. Thus $\Delta CH_4[i] = \beta 3[i]$, and since the value of $CH_4[i]$ is known, the attacker also knows that the corresponding addition-difference is δ_0.

Thus, if $E_4 = E_4^* = \psi_4$, then $\Delta CH_4 = \mu_4 = \psi_4' \wedge \beta 3$, with $CH_4 = \mathbf{Q}' \wedge \mu_4$, and $CH_4 = \mathbf{Q} \wedge \mu_4$ *This step requires the attacker to guess bit position(s) indicated by* $\beta 3$.

Determine δCH_4. Since $CH_4[i]$ is known at the bit positions where $\Delta CH[i] = 0$, the attacker can determine δCH_4, which we denote by δ_4:

$$\delta CH_4 = (\mathbf{Q} \wedge \mu_4) - (\mathbf{Q}' \wedge \mu_4).$$

Determine required δW_4. The attacker injects an addition-difference between W_4 and W_4^* in order to cancel the differences so that $\delta A_5 = 0$. Thus $\delta W_4 = -\delta H_4 - \delta CH_4 = -\delta_0 - \delta_4$. A by-product of injecting this difference is that $\delta T1_4 = 0$, and since $\delta D_4 = \delta_0$, this implies that $\delta E_5 = \delta_0$.

Assume that the condition $E_5 = D_4;\widehat{\alpha 1}$ holds. If this condition holds, then

$$\Delta E_5 = E_5^* \oplus A_5 \;=\; (E_5 + \delta_0) \oplus E_5 = (D_4 \oplus \alpha 1) \oplus D_4 = \alpha 1.$$

If this condition does not hold, then the injected difference will result in more than one bit of difference in E_5, and the complexity of causing a collision will increase. *This step requires the attacker to make assumptions on bit position(s) indicated by* $\widehat{\alpha 1}$.

Round 5

Determine δH_5. Note that the difference in E_2 has now propagated to the top of the register so $\delta H_5 = \delta E_2 = -\delta_{1,2}$.

Guess $F_5 = \xi_5;\alpha 1$, and infer δCH_5. Observe that $F_5 = F_5^*$. The CHOOSE function will choose the value of $F_5 = F_5^* = \xi_5;\alpha 1$, at one run and the value of $G_5 = G_5^* = \xi_3;\alpha 1$, at the other run. This will give $\Delta CH_5 = (\xi_3 \oplus \xi_5)$. If the $\Delta CH_5 = 0$, then the attacker does not need to cancel out the difference. If $\Delta CH_5 = \alpha 1$, then the attacker knows the value of CH_5 at this bit, so the attacker is able to determine δCH_5, which we denote by $\delta_{5,1}$.

Guess $\Sigma_1(E_5) = \mathbf{S};\widehat{\gamma 3}$. Assuming that the assigned values are correct, then the attacker knows that $\Delta E_5 = \alpha 1$ and thus $\Delta\Sigma_1(E_5) = \Sigma_1(\Delta E_5) = \gamma 3$. The attacker needs to know the addition-difference $\delta\Sigma_1(E_5)$ so that the attacker can insert the necessary difference in δW_1 to cancel this difference out in the compression function. That is, the attacker needs to know the value of $\Sigma_1(E_5)$ at the bits where $\widehat{\gamma 3}[i] = 1$. Each bit of $\Sigma_1(E_5)$ has inputs from three bits, and the attacker has only assigned values to one of these bits. Consequently, the attacker needs to guess the value of the bits of $\Sigma_1(E_5)$ at the bits where $\widehat{\gamma 3}[i] = 1$. This gives the attacker sufficient information to determine the corresponding addition difference $\delta\Sigma_1(E_5)$ which we denote by $\delta_{5,2}$. We use the value $\mathbf{S}$ to denote the guess for these bits. *This step requires the attacker to guess bit position(s) indicated by $\widehat{\gamma 3}$.*

Determine required δW_5. The attacker injects an addition-difference between W_5 and W_5^* in order to cancel the differences so that $\delta A_6 = 0$. Thus $\delta W_5 = -\delta H_5 - \delta\Sigma_1(E_5) - \delta CH_5 = -\delta_{1,2} - \delta_{5,1} - \delta_{5,2}$. A by-product of injecting this difference is that $\delta T1_5 = 0$, and $\delta E_6 = 0$.

6.3 Detailed Explanation of Table 5: Rounds 6 to 8

		H_i	G_i	F_i	E_i	D_i	C_i	B_i	A_i	W_i	CH	Σ_1	$T1$	MJ	Σ_0	$T2$	E_{i+1}	A_{i+1}	Ass.	Gue.
6	Δ	-	-	$\alpha 1$	-	-	-	-	-		ξ_6	-	-	-	-	-	-	-		
	δ	-	-	δ_0	-	-	-	-	-		δ_6	-	-	-	-	-	-	-		
	$M@\alpha 1$	ξ_3'	ξ_5	$\mathbf{J}$	ξ_6	Y					$\mathbf{J}$;ξ_6						ξ_7			
	$M^*@\alpha 1$	ξ_3'	ξ_5	$\mathbf{J'}$	ξ_6	Y					$\mathbf{J'}$;ξ_6						ξ_7			
	Gue.	$E_6 = \xi_6;\ \alpha 1$									$\Rightarrow CH_6 = \mathbf{J};\xi_6,\ CH_6^* = \mathbf{J'};\xi_6$									$\alpha 1$
	Det.	$\delta CH_6 \stackrel{\text{def}}{=} \delta_6$									$\delta_6 = (\mathbf{J} \oplus \xi_6) - \mathbf{J}$									
	Det.	$\delta W_6 = -\delta_6$																		
	Summary of Round 6:	$\delta W_6 = -\delta CH_6 = -\delta_6$																		$\alpha 1$
		H_i	G_i	F_i	E_i	D_i	C_i	B_i	A_i	W_i	CH	Σ_1	$T1$	MJ	Σ_0	$T2$	E_{i+1}	A_{i+1}	Ass.	Gue.
7	Δ	-	$\alpha 1$	-	-	-	-	-	-		τ_7	-	-	-	-	-	-	-		
	δ	-	δ_0	-	-	-	-	-	-		δ_7	-	-	-	-	-	-	-		
	$M@\alpha 1$	ξ_5	$\mathbf{J}$	ξ_6	ξ_7						$\mathbf{J}$;τ_7									
	$M^*@\alpha 1$	ξ_5	$\mathbf{J'}$	ξ_6	ξ_7						$\mathbf{J'}$;τ_7									
	Gue.	$E_7 = \xi_7;\ \alpha 1$									$\Rightarrow CH_7 = \mathbf{J};\tau_7,\ CH_7^* = \mathbf{J'};\tau_7$ where $\tau_7 = \xi_7' \wedge \alpha 1$.									$\alpha 1$
	Det.	$\delta CH_7 \stackrel{\text{def}}{=} \delta_7$									$\delta_7 = (\mathbf{J} \oplus \tau_7) - \mathbf{J}$									
	Det.	$\delta W_7 = -\delta CH_7 = -\delta_7$																		
	Summary of Round 7:	$\delta W_7 = -\delta_7$																		$\alpha 1$
		H_i	G_i	F_i	E_i	D_i	C_i	B_i	A_i	W_i	CH	Σ_1	$T1$	MJ	Σ_0	$T2$	E_{i+1}	A_{i+1}	Ass.	Gue.
8	Δ	$\alpha 1$	-	-	-	-	-	-	-		-	-	-	-	-	-	-	-		
	δ	δ_0	-	-	-	-	-	-	-		-	-	-	-	-	-	-	-		
	$M@\alpha 1$	$\mathbf{J}$																		
	$M^*@\alpha 1$	$\mathbf{J'}$																		
	Det.	$\delta H_8 = \delta_0$																		
	Det.	$\delta W_8 = -\delta H_8 = -\delta_0$																		
	Summary of Round 8:	$\delta W_8 = -\delta_0$																		
	Rounds 6 to 8: Bits Guessed	$2\lvert\alpha 1\rvert + 0\lvert\widehat{\alpha 1}\rvert + 0\lvert\beta 3\rvert + 0\lvert\widehat{\beta 3}\rvert + 0\lvert\widehat{\gamma 3}\rvert + 0\lvert\widehat{\epsilon 9}\rvert$																		

Table 5. Details of the corrective pattern for rounds 6 to 8.

Round 6

Guess $E_6 = E_6^* = \xi_6$;$\alpha 1$, and infer ΔCH_6 and δCH_6. Observe $E_6 = E_6^*$;$\alpha 1$. The value of E_6 affects where the CHOOSE function outputs F_6 and F_6^* or G_6 and G_6^* in the first and second run respectively. Let i denote the bits position where $\alpha 1 = 1$.

- If $E_6[i] = 1$, then the CHOOSE function outputs $F_6[i] = \mathbf{J}[i]$, and $F_6^*[i] = \mathbf{J'}[i]$, in the first and second run respectively. Thus $\Delta CH_6 = \alpha 1$, and since the value of $CH_6[i]$ is known, the attacker also knows $\delta CH_6 = \delta_0$.
- If $E_6[i] = 0$, then the CHOOSE function outputs $G_6[i]$ and $G_6^*[i] = G_6[i]$, in the first and second run respectively. Thus $\Delta CH_6 = \delta CH_6 = 0$, and

the attacker does not have to input a difference in W_6 in order to cancel this difference.

Thus, if $E_6 = E_6^* = \xi_6;\alpha 1$, then $\Delta CH_6 = \xi_6$, and $\delta CH_6 = 0$, when $xi_6 = 0$ and $\delta CH_6 = \alpha 1$, when $\xi_6 = \delta_0$. Let δ_6 denote the value of δCH_6. *This step requires the attacker to guess bit position(s) indicated by* $\alpha 1$.

Determine required δW_6. The attacker injects an addition-difference between W_6 and W_6^* in order to cancel the differences so that $\delta A_7 = 0$. Thus $\delta W_6 = -\delta CH_6 = -\delta_6$. A by-product of injecting this difference is that $\delta T1_6 = 0$, and $\delta E_7 = 0$.

Round 7

Guess the value of $E_7 = E_7^* = \xi_7;\alpha 1$. Let i denote the bits position where $\alpha 1 = 1$.

- If $E_7[i] = 1$, then the CHOOSE function outputs $F_7[i]$ and $F_7^*[i] = F_7[i]$, in the first and second run respectively. Thus $\Delta CH_7[i] = 0$ and the attacker does not have to input a difference in W_7 in order to cancel a difference.
- If $E_7[i] = 0$, then the CHOOSE function outputs $G_7[i] = \mathbf{J}[i]$, and $G_7^*[i] = \mathbf{J'}[i]$, in the first and second run respectively. Thus $\Delta CH_7[i] = \alpha 1[i]$, and since the value of $CH_7[i]$ is known, the attacker also knows that the corresponding addition-difference is δ_0.

Thus, if $E_7[i] = E_7^*[i] = \xi_7[i]$, then $\Delta CH_7 = \tau_7[i] = \xi_7'[i] \wedge \alpha 1$. Let δ_7 denote the corresponding addition-difference. *This step requires the attacker to guess bit position(s) indicated by* $\alpha 1$.

Determine required δW_7. The attacker injects an addition-difference between W_7 and W_7^* in order to cancel the differences so that $\delta A_8 = 0$. Thus $\delta W_7 = -\delta CH_7 = -\delta_7$. A by-product of injecting this difference is that $\delta T1_7 = 0$, and $\delta E_8 = 0$.

Round 8

Determine δH_8. Note that the difference in A_1 has now propagated to the top of the register so $\delta H_8 = \delta A_1 = \delta_0$.

Determine required δW_8. The attacker injects an addition-difference between W_8 and W_8^* in order to cancel the differences so that $\delta A_9 = 0$. Thus $\delta W_8 = -\delta CH_8 = -\delta_8$. After injecting this differece, all the internal conditions have been canceled and the register states in the first and second runs are equal.

7 Complexity

The probability that the assumptions hold is is equal to 2^{-u} where u is the total number of bits that have assumptions placed on them. Table 6 indicates that $u = 4|\alpha 1| + 2|\widehat{\alpha 1}| + 1|\beta 3| + 1|\widehat{\beta 3}|$. An exception occurs when $\alpha 1 = 2^{w-1}$ (See

Rounds 0 to 2: Assumed Conditions	$3\|\alpha 1\| + 1\|\widehat{\alpha 1}\| + 1\|\beta 3\| + 1\|\widehat{\beta 3}\| + 0\|\widehat{\gamma 3}\| + 0\|\widehat{\epsilon 9}\|$
Rounds 3 to 5: Assumed Conditions	$1\|\alpha 1\| + 1\|\widehat{\alpha 1}\| + 0\|\beta 3\| + 0\|\widehat{\beta 3}\| + 0\|\widehat{\gamma 3}\| + 0\|\widehat{\epsilon 9}\|$
Total Assumed Conditions	$4\|\alpha 1\| + 2\|\widehat{\alpha 1}\| + 1\|\beta 3\| + 1\|\widehat{\beta 3}\| + 0\|\widehat{\gamma 3}\| + 0\|\widehat{\epsilon 9}\|$
Rounds 0 to 2: Bits Guessed	$1\|\alpha 1\| + 1\|\widehat{\alpha 1}\| + 0\|\beta 3\| + 1\|\widehat{\beta 3}\| + 1\|\widehat{\gamma 3}\| + 1\|\widehat{\epsilon 9}\|$
Rounds 3 to 5: Bits Guessed	$2\|\alpha 1\| + 0\|\widehat{\alpha 1}\| + 2\|\beta 3\| + 0\|\widehat{\beta 3}\| + 1\|\widehat{\gamma 3}\| + 0\|\widehat{\epsilon 9}\|$
Rounds 6 to 8: Bits Guessed	$2\|\alpha 1\| + 0\|\widehat{\alpha 1}\| + 0\|\beta 3\| + 0\|\widehat{\beta 3}\| + 0\|\widehat{\gamma 3}\| + 0\|\widehat{\epsilon 9}\|$
Total Bits Guessed	$5\|\alpha 1\| + 1\|\widehat{\alpha 1}\| + 2\|\beta 3\| + 1\|\widehat{\beta 3}\| + 2\|\widehat{\gamma 3}\| + 1\|\widehat{\epsilon 9}\|$
Total Number of Bits	$9\|\alpha 1\| + 3\|\widehat{\alpha 1}\| + 3\|\beta 3\| + 2\|\widehat{\beta 3}\| + 2\|\widehat{\gamma 3}\| + 1\|\widehat{\epsilon 9}\|$

Table 6. A summary of number of assumed conditions and guessed bits in the corrective pattern.

Note 1 in analysis of Round 1), in which case $u = 3|\alpha 1| + 2|\widehat{\alpha 1}| + 1|\beta 3| + 1|\widehat{\beta 3}|$. For each message block, where the conditions are assumed to hold, the attacker attempts 2^v guesses for bits of the internal state, v is the total number of bits that are guessed. Table 6 indicates that $v = 5|\alpha 1| + 2|\widehat{\alpha 1}| + 1|\beta 3| + 2|\widehat{\beta 3}| + 1|\widehat{\epsilon 9}|$. An exception occurs when $\alpha 1 = 2^{w-1}$ (See Note 1 in analysis of Round 1), in which case $v = 6|\alpha 1| + 2|\widehat{\alpha 1}| + 1|\beta 3| + 2|\widehat{\beta 3}| + 1|\widehat{\epsilon 9}|$. For most single-bit differences $\alpha 1$, these values are:

$$|\widehat{\alpha 1}| = |\alpha 1| = 1;\ |\widehat{\beta 3}| = |\beta 3| = 3;\ |\widehat{\gamma 3}| = 3;\ |\widehat{\epsilon 9}| = 9;\ \text{so}$$
$$u = 4 \cdot 1 + 2 \cdot 1 + 1 \cdot 3 + 1 \cdot 3 + 0 \cdot 3 + 0 \cdot 9 = 12,$$
$$v = 5 \cdot 1 + 1 \cdot 1 + 2 \cdot 3 + 1 \cdot 3 + 2 \cdot 3 + 1 \cdot 9 = 30.$$

Thus, the probability of the assumed conditions being true is 2^{-12} and for each block where the conditions are assumed to hold, the attacker attempts 2^{30} guesses for bits of the internal state. This indiates a total complexity of 2^{42}: assuming that the initial register states are unknown.

The complexity can be deceased by choosing $\alpha 1$ such that one or more of the differences $\alpha 1$, $\beta 3$, $\gamma 3$,$\epsilon 9$ has the MSB equal to 1. The effect of this is to reduce the corresponding weight(s) of $\widehat{\alpha 1}$ $\widehat{\beta 3}$, $\widehat{\gamma 3}$, or $\widehat{\epsilon 9}$. If the difference $\alpha 1 = 0x80000000$ is chosen, then $\widehat{\alpha 1} = 0x00000000$ and the probability is increased from 2^{42} to 2^{39}: assuming that the initial register states are unknown. However, we note that that one of the assumed conditions has also become guess, so the probability of the assumed conditions being true is 2^{-9} and for each block where the conditions are assumed to hold, the attacker attempts 2^{30} guesses for bits of the internal state.

In the case of SHA-224/256, the probability is also increased to 2^{-39} when $\alpha 1 = 0x00001000$, for which

$$\beta 3 = 0x80400400,\ \gamma 3 = 0x00080042,\ \epsilon 9 = 0xa2130850,$$
$$\widehat{\beta 3} = 0x00400400,\ \widehat{\gamma 3} = 0x00080042,\ \widehat{\epsilon 9} = 0x22130850,$$

so the Hamming weights of both $\widehat{\beta 3}$ and $\widehat{\epsilon 9}$ have decreased by one.

These two corrective patters appears to have the minimum probability for corrective patterns based on single-bit differences.

7.1 Known Initial States

When computing an un-keyed hash of a message, then the initial register states are known to the attacker. As noted in Section 4, this reduces the complexity significantly for two reasons: the attacker can test the assumptions independently of guessing bits; and the attacker no longer has to guess bits since the values can be computed by the attacker. This reduces, to 2^9, the complexity of finding a corrective pattern that results in equal register states.

8 Conclusion

New corrective patterns for the SHA-2 register are presented. The probabbility of the corrective pattern is 2^{-39} when the intial register states are unknown, and 2^{-9} when the initial register states are known. This probability voids the previous analysis that concluded that the SHA-2 algorithms resist Chaboud-Joux attacks. However, we cannot conclude (from our results) whether the SHA-2 algorithms are sufficiently secure or not. A more detailed analysis of the message schedule is required. We have not yet attempted such an analysis, but recommend this as an worthwhile venture for any cryptologist.

References

1. Eli Biham, Rafi Chen *New results on SHA-0 and SHA-1* Short talk presented at CRYPTO 2004 Rump Session, 2004.
2. F. Chabaud and A. Joux, *Differential Collisions in SHA-0,* Advances in Cryptology-CRYPTO'98, Lecture Notes in Computer Science, vol.1462, pp.56-71, Springer-Verlag, 1998.
3. National Institute of Standards and Technology, *Federal Information Processing Standards (FIPS) Publication 180-2, Secure Hash Standard (SHS),* February, 2004.
4. H. Gilbert and H. Hanschuh, *Security Analysis of SHA-256 and sisters,* Selected Areas in Cryptography, SAC 2003, Ottawa, Canada, Lecture Notes in Computer Science, vol. 3006, M. Matsui and R. Zuccheratopp (Eds), pp. 175-193, Springer, 2004.
5. A. Joux, *Collisions in SHA-0* Short talk presented at CRYPTO 2004 Rump Session, 2004.
6. A. Menezes, P van Oorschot and S. Vanstone, *Handbook of Applied Cryptography,* CRC Press LLC, 1997.
7. X. Wang, D. Feng, X. Lai and H. Yu, *Collisions for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD,* Cryptology ePrint Archive, Report 2004/199, see `http://eprint.iacr.org/`