Collision attacks on reduced-round SHA256

A Project Report Submitted
in Partial Fulfillment of Requirements
for the Degree of

Bachelor of Technology

by

Divjot Singh Rawla (2016csb1039)

Department of Computer Science & Engineering
Indian Institute of Technology Ropar
Rupnagar 140001, India
June 2020

Abstract

The work presents a new method of producing collisions on reduced rounds of SHA-256. Using the same, we provide collisions on SHA-256 reduced to 22 rounds and semi-free start collisions on SHA-256 reduced to 23 rounds. Additionally, we provide analysis over small components of SHA-256 (Σ and σ functions).

Honor Code

We certify that we have properly cited any material taken from other sources and have obtained permission for any copyrighted material included in this report. We take full responsibility for any code submitted as part of this project and the contents of this report.

Divjot Singh Rawla (2016csb1039)

Certificate

It is certified that the B. Tech. project "Title of the Project" has been done by Divjot Singh Rawla (2016csb1039) under my supervision. This report has been submitted towards partial fulfillment of B. Tech. project requirements.

Dr. Somitra Sanadhya
Project Supervisor
Department of Computer Science & Engineering
Indian Institute of Technology Ropar
Rupnagar-140001

Contents

Contents v

List of Figures vii

List of Tables viii

Nomenclature viii

1 Introduction 1

1.1 SHA-256 Details . . . 2

1.2 Collisions in SHA-256 . . . 4

2 Methodology 6

2.1 Definitions & Notations . . . 6

2.1.1 Basic Notation . . . 6

2.1.2 Differential Table . . . 6

2.1.3 Types of collisions . . . 8

2.2 Approach . . . 8

2.2.1 First Phase . . . 8

2.2.2 Second Phase . . . 14

2.2.2.1 Manual Analysis To Reduce Variables . . . 14

2.2.2.2 Solving Equations . . . 15

2.2.2.3 Searching a complete differential . . . 16

2.2.3 Third Phase . . . 19

3 Collisions on 22 Steps **20**
3.1 Description 20
3.2 Solving Additional Constraint 20

4 Analysis of Σ Functions **23**
4.1 Motivation 23
4.2 Differentials Given Additive Difference 23
4.3 Analysis of Σ_0 24
4.4 Experiment 25

5 Future Work **28**
5.1 Extension of Method 28
5.2 Alternate Method 33
5.3 Better Message Search 34

References **35**

List of Figures

1.1 Cryptographic Hash Function (H) with output size 256 bits 1
1.2 SHA-2 step update function, image from [Sanadhya and Sarkar, 2008a] 3

List of Tables

1.1 Previous Results for Collisions on Reduced Rounds of SHA-256 . 5

2.1 Representation of relations in differential tables 7
2.2 Incomplete differential table for 17 step collision 10
2.3 Incomplete Differential table for 17 step collision after analysis . . 13
2.4 Conditions on causing bits for fixed output of Σ_1 16
2.5 Complete Differential Table for 17 step collision 18
2.6 Message Pair for 17 step collision 19

3.1 Complete Differential Table for 22 step collision 22
3.2 Collision for 22 step collision 22

4.1 Results of Algorithm 1 27

5.1 Incomplete Differential Table for 23 step collision 30
5.2 Complete Differential Table for 23 step collision 32
5.3 Semi-Free Start Collision for 23 steps: Message Pairs 32
5.4 Semi-Free Start Collision for 23 steps: Initial Registers 33

Chapter 1

Introduction

Cryptographic hash functions (referred to as hash functions everywhere in this report) are the building blocks of various cryptographic protocols and primitives. A hash function operates on an arbitrary sized input and produces a fixed sized digest.

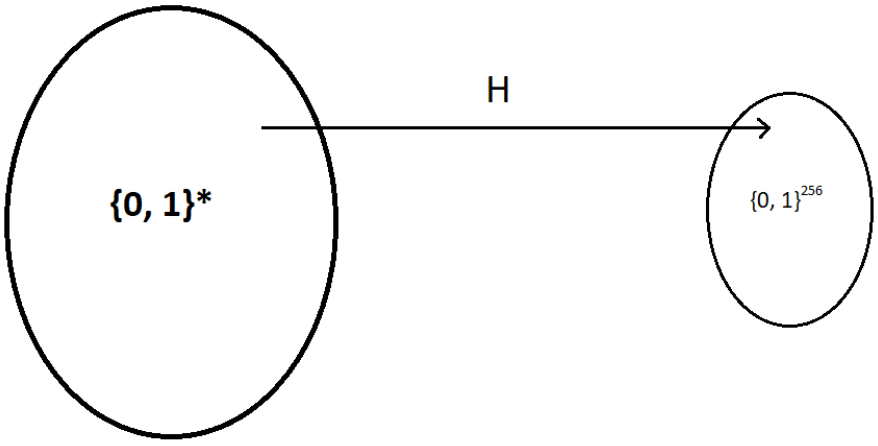

Figure 1.1: Cryptographic Hash Function (H) with output size 256 bits

A secure cryptographic hash function is required to satisfy some security properties, such as preimage resistance, second preimage resistance, collision resistance etc. In this work, we focus on the hash function SHA-256 and attempt to analyze

the differential properties of this hash function in order to attack its collision resistance property.

1.1 SHA-256 Details

For SHA256, the function is of the form $\{0,1\}^* \rightarrow \{0,1\}^{256}$. An input message is first padded and divided into chunks of 512 bits each; and then each chunk of 512 bits is processed in sequence.

An input message chunk of 512 bits is divided into 16 words of 32 bits each, denoted as $M_0, M_1, \ldots, M_{15}$. The SHA-256 processing has 64 steps, each step requiring a 32-bit input. These inputs are produced by the "message expansion" function, which is defined via a recursion as follows:

$$W_i = \begin{cases} M_i, & \text{if } i < 16 \\ \sigma_1(W_{i-2}) + W_{i-7} + \sigma_0(W_{i-15}) + W_{i-16}, & \text{otherwise} \end{cases}$$

$$\begin{aligned} \text{where,} \quad \sigma_0(x) &= (x \ggg 7) \oplus (x \ggg 18) \oplus (x \gg 3), \\ \sigma_1(x) &= (x \ggg 17) \oplus (x \ggg 19) \oplus (x \gg 10). \end{aligned}$$

SHA-256 also specifies a set of 8 registers (a, b $\ldots$ h). Every register is a 32 bit unsigned integer. The produced 64 words from message expansion are then applied sequentially to update this set of registers with the function illustrated in 1.2 and given as follows:

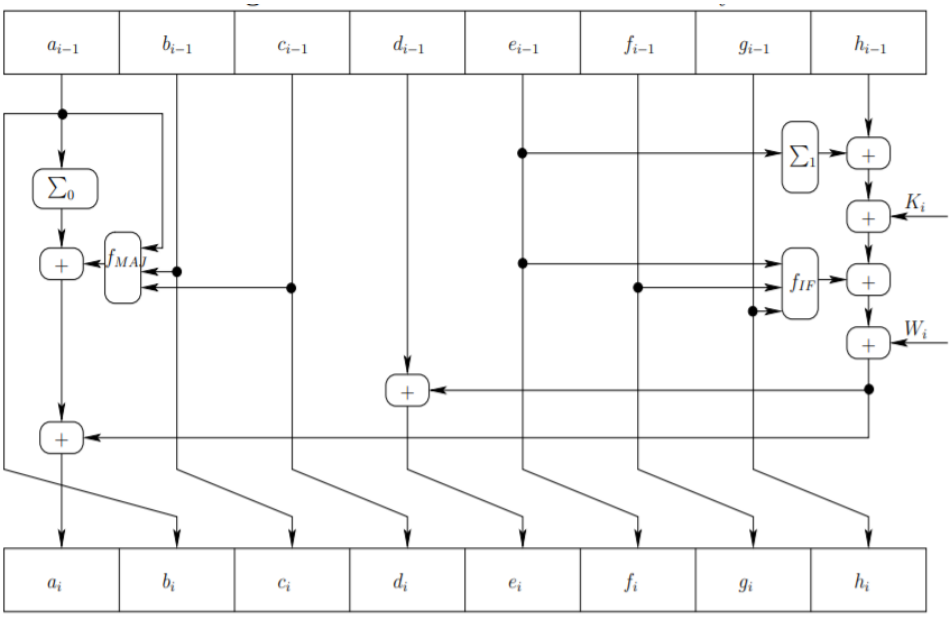

Figure 1.2: SHA-2 step update function, image from [Sanadhya and Sarkar, 2008a]

$$
\begin{aligned}
T_0 &= \Sigma_0(A_{i-1}) + f_{MAJ}(A_{i-1}, B_{i-1}, C_{i-1}) \\
T_1 &= \Sigma_1(E_{i-1}) + f_{IF}(E_{i-1}.F_{i-1}, G_{i-1}) + H_{i-1} + K_i + W_i \\
A_i &= T_0 + T_1 \\
B_i &= A_{i-1} \\
C_i &= B_{i-1} \\
D_i &= C_{i-1} \\
E_i &= D_{i-1} + T_1 \\
F_i &= E_{i-1} \\
G_i &= F_{i-1} \\
H_i &= G_{i-1}
\end{aligned}
\tag{1.1}
$$

The boolean functions f_{MAJ} and f_{IF} are defined as follows:

$$
\begin{aligned}
f_{MAJ}(x, y, z) &= (x \wedge y) \oplus (y \wedge z) \oplus (z \wedge x) \\
f_{IF}(x, y, z) &= (x \wedge y) \oplus (\neg x \wedge z)
\end{aligned}
$$

The two functions Σ_0 and Σ_1 are defined as:

$$\Sigma_0(x) = (x \ggg 2) \oplus (x \ggg 13) \oplus (x \ggg 22)$$
$$\Sigma_1(x) = (x \ggg 6) \oplus (x \ggg 11) \oplus (x \ggg 25)$$

The output for the current block of message is defined as the xor of the final value of registers (a_{63}, b_{63} ... h_{63}) with their initial value (a_{-1}, b_{-1} ... h_{-1}). And this output is used as the initial registers for the next block of the message. The value for the hash of the message is the output of the last block of this message.

Alternatively, we can also describe the step update using only A, E registers in the following way:

$$\begin{aligned}
A_i &= \Sigma_0(A_{i-1}) + f_{MAJ}(A_{i-1}, A_{i-2}, A_{i-3}) + E_i - A_{i-1} \\
B_i &= A_{i-1} \\
C_i &= A_{i-2} \\
D_i &= A_{i-3} \\
E_i &= A_{i-4} + \Sigma_1(E_{i-1}) + f_{IF}(E_{i-1}.E_{i-2}, E_{i-3}) + E_{i-4} + K_i + W_i + E_{i-4} \\
F_i &= E_{i-1} \\
G_i &= E_{i-2} \\
H_i &= E_{i-3}
\end{aligned} \tag{1.2}$$

The alternate definition of SHA-256 is used for analysis in this work. For more details of the design, refer to the NIST design document for SHA-2.

1.2 Collisions in SHA-256

Two messages are said to produce a collision if they have the same hash value. It is clear that collisions will exist for cryptographic hash functions because the size of domain is larger than the size of the range. The collision resistance property says that it is computationally inefficient to find such collisions. The aim of collision attacks is to find such messages with more efficiency than what is guaranteed by the design.

The structure of SHA-256 is such that it is enough to find two unequal blocks that produce the same output to find a collision. Following table summarizes the results of the collision attacks on SHA-256:

Publication	Rounds
[Sanadhya and Sarkar, 2008a]	21
[Indesteege et al., 2008]	24
[Sanadhya and Sarkar, 2008b]	24 (More Efficient)
[Mendel et al., 2011]	27
[Mendel et al., 2015]	28

Table 1.1: Previous Results for Collisions on Reduced Rounds of SHA-256

Chapter 2

Methodology

2.1 Definitions & Notations

2.1.1 Basic Notation

Notation 1. *$x_0, x_1, x_2 \dots x_{31}$ represents the 32-bit unsigned binary value of x*

Notation 2. *$+$ denotes addition modulo 2^{32}*

Notation 3. *$\oplus$ denotes bit-wise xor operation*

Notation 4. *δx denotes the value x' - x*

Notation 5. *$\delta\Sigma(x)$ denotes the value $\Sigma(x')$ - $\Sigma(x)$*

Notation 6. *$\delta f_{MAJ}(\delta x, \delta y, \delta z)$ denotes the value $f_{MAJ}(x', y', z')$ - $f_{MAJ}(x, y, z)$*

Notation 7. *$\delta f_{IF}(\delta x, \delta y, \delta z)$ denotes the value $f_{IF}(x', y', z')$ - $f_{IF}(x, y, z)$*

2.1.2 Differential Table

A differential table is used to describe the relation between bits at the same position of two variables. The following table states the representation of relations between two bits X and X': Now that we have defined the notation for a single bit, we can describe variables with more number of bits as well. The notation $\nabla K = [\,\mathrm{u}\,\mathrm{n}\,0\,\text{-}\,]$ represents two variables $K \in \{0,1\}^4$ and $K' \in \{0,1\}^4$ such that

(X, X')	(0, 0)	(1, 0)	(0, 1)	(1, 1)
0	✓	-	-	-
1	-	-	-	✓
-	✓	-	-	✓
n	-	-	✓	-
u	-	✓	-	-
x	-	✓	✓	-
?	✓	✓	✓	✓

Table 2.1: Representation of relations in differential tables

the $K_0 = 1, K'_0 = 0$ and the other bit positions also satisfy the bit constraints. Here is the formal definition:

Notation 8. *$\nabla D = [d_0, d_1, d_2 \ldots d_{n-1}]$ denotes the set $\{(D, D')$: $(D_i, D'_i) \in \nabla D_i$ $\forall\ i<n\}$*

Notation 9. *Incomplete Differential (∇_0) denotes a differential that only has '?' or '-' relationship between bits*

Notation 10. *Partially Complete Differential (∇_1) denotes a differential that only has 'x' or '-' relationship between bits*

Notation 11. *Complete Differential (∇_2) denotes a differential that only has '-', '0', '1', 'u' or 'n' relationship between bits. A differential table stores differentials for all A, E, and W step values.*

Notation 12. *nzbits(∇) denotes the set of bit positions of ∇ that are 'x', 'u' or 'n'.*

Notation 13. *fully-cancels ($\nabla_1 A, \nabla_1 B$) is true $\iff$ there exists some $\nabla_2 A$ and $\nabla_2 B$ such that $\delta A + \delta B = 0$*

Notation 14. *subset-cancels ($\nabla_1 A, \nabla_1 B$) is true $\iff$ there exists a $\nabla_2 A_{sub}$ and $\nabla_2 B$ such that $\delta A_{sub} + \delta B = 0$, where $nzbits(\nabla_1 A_{sub}) \subseteq nzbits(\nabla_1 A)$*

2.1.3 Types of collisions

Definition 1. *Reduced Rounds Collision* is a collision on a variation of SHA-256 that has less than 64 rounds.

Definition 2. *Reduced Rounds Semi-Free Start Collision* is a collision on a variation of SHA-256 that has less than 64 rounds with a different set of initial registers but same for the two messages that collide.

Definition 3. *Reduced Rounds Free Start Collision* is a collision on a variation of SHA-256 that has less than 64 rounds. In this case, the set of initial for the two messages is not the same. A free start collision is valuable only if there is very little difference between the initial registers.

2.2 Approach

The strategy is broken into three phases(inspiration from work in [Mendel et al., 2011]. A brief description of all the parts follows:

1. The first phase is used to come up with an incomplete differential table that is satisfiable with respect to the additive difference equations imposed by SHA-256.

2. The second phase deals with the conversion of the differential table produced in the first phase to a complete differential table while still keeping the additive difference equations satisfiable.

3. The third phase produces the message pairs that follow this differential table using a slightly modified version of DFS.

In the following sub-sections we describe these phases in detail. An example of a 17 step collision is kept running during the entire explanation.

2.2.1 First Phase

The following table is used as a starting step (this is the same table as used in [Mendel et al., 2011]):

	∇A
-4	- -
-3	- -
-2	- -
-1	- -
0	- -
1	- -
2	- -
3	- -
4	- -
5	- -
6	- -
7	? ?
8	? ?
9	? ?
10	- -
11	- -
12	- -
13	- -
14	- -
15	- -
16	- -
17	- -

	∇E
-4	- -
-3	- -
-2	- -
-1	- -
0	- -
1	- -
2	- -
3	- -
4	- -
5	- -
6	- -
7	? ?
8	? ?
9	? ?
10	? ?
11	? ?
12	? ?
13	? ?
14	- -
15	- -
16	- -
17	- -

	∇W
0	- -
1	- -
2	- -
3	- -
4	- -
5	- -
6	- -
7	? ?
8	? ?
9	- -
10	? ?
11	- -
12	- -
13	- -
14	- -
15	? ?
16	- -
17	? ?

Table 2.2: Incomplete differential table for 17 step collision

Now, the set of additive equations caused by the differential tables above can be obtained by the using the alternative description (1.2). If all the variables of input component($f_{IF} or f_{MAJ}$) have 0 additive difference, then the whole component has to have 0 additive difference. Using this fact, we can generate the following equations:

$$\begin{aligned}
\delta A_7 &= \delta E_7 \\
\delta\Sigma_0(A_7) + \delta f_{MAJ}(\delta A_7, \delta A_6, \delta A_5) + \delta E_8 &= \delta A_8 \\
\delta\Sigma_0(A_8) + \delta f_{MAJ}(\delta A_8, \delta A_7, \delta A_6) + \delta E_9 &= 0 \\
\delta\Sigma_0(A_9) + \delta f_{MAJ}(\delta A_9, \delta A_8, \delta A_7) + \delta E_{10} &= 0 \\
\delta f_{MAJ}(\delta A_{10}, \delta A_9, \delta A_8) + \delta E_{11} &= \delta A_7 \\
\delta f_{MAJ}(\delta A_{11}, \delta A_{10}, \delta A_9) + \delta E_{12} &= \delta A_8 \\
\delta E_{13} &= \delta A_9, \text{and}
\end{aligned} \tag{2.1}$$

$$
\begin{aligned}
\delta W_7 &= \delta E_7 \\
\delta\Sigma_1(E_7) + \delta f_{IF}(\delta E_7, \delta E_6, \delta E_5) + \delta W_8 &= \delta E_8 \\
\delta\Sigma_1(E_8) + \delta f_{IF}(\delta E_8, \delta E_7, \delta E_6) &= \delta E_9 \\
\delta\Sigma_1(E_9) + \delta f_{IF}(\delta E_9, \delta E_8, \delta E_7) + \delta W_{10} &= \delta E_{10} \\
\delta A_7 + \delta\Sigma_1(E_{10}) + \delta f_{IF}(\delta E_{10}, \delta E_9, \delta E_8) + \delta E_7 &= \delta E_{11} \\
\delta A_8 + \delta\Sigma_1(E_{11}) + \delta f_{IF}(\delta E_{11}, \delta E_{10}, \delta E_9) + \delta E_8 &= \delta E_{12} \\
\delta A_9 + \delta\Sigma_1(E_{12}) + \delta f_{IF}(\delta E_{12}, \delta E_{11}, \delta E_{10}) + \delta E_9 &= \delta E_{13} \\
\delta\Sigma_1(E_{13}) + \delta f_{IF}(\delta E_{13}, \delta E_{12}, \delta E_{11}) + \delta E_{10} &= 0 \\
\delta f_{IF}(\delta E_{14}, \delta E_{13}, \delta E_{12}) + \delta E_{11} + \delta W_{15} &= 0 \\
\delta f_{IF}(\delta E_{15}, \delta E_{14}, \delta E_{13}) + \delta E_{12} &= 0 \\
\delta E_{13} + \delta W_{17} &= 0
\end{aligned}
\tag{2.2}
$$

It is clear that this differential table satisfies all the conditions we desired from the first phase. However, in this phase we also try to simplify our differential to have only few message words that are different. This makes further analysis easier.

We decide to make $\delta\ W_{17} = 0$. This causes $\delta\ E_{13}$, $\delta\ E_{12}$, $\delta\ A_9$, $\delta\ A_8$ all equal to 0 and thus, makes the additive difference equations much less complicated.

These are the final equations:

$$
\begin{aligned}
\delta A_7 &= \delta E_7 \\
\delta\Sigma_0(A_7) + \delta f_{MAJ}(\delta A_7, \delta A_6, \delta A_5) + \delta E_8 &= 0 \\
\delta f_{MAJ}(\delta A_8, \delta A_7, \delta A_6) + \delta E_9 &= 0 \\
\delta f_{MAJ}(\delta A_9, \delta A_8, \delta A_7) + \delta E_{10} &= 0 \\
\delta E_{11} - \delta A_7 &= 0, \text{and}
\end{aligned}
\tag{2.3}
$$

$$
\begin{aligned}
\delta W_7 &= \delta E_7 \\
\delta\Sigma_1(E_7) + \delta f_{IF}(\delta E_7, \delta E_6, \delta E_5) + \delta W_8 &= \delta E_8 \\
\delta\Sigma_1(E_8) + \delta f_{IF}(\delta E_8, \delta E_7, \delta E_6) &= \delta E_9 \\
\delta\Sigma_1(E_9) + \delta f_{IF}(\delta E_9, \delta E_8, \delta E_7) + \delta W_{10} &= \delta E_{10} \\
\delta\Sigma_1(E_{10}) + \delta f_{IF}(\delta E_{10}, \delta E_9, \delta E_8) + \delta E_7 &= 0 \\
\delta\Sigma_1(E_{11}) + \delta f_{IF}(\delta E_{11}, \delta E_{10}, \delta E_9) + \delta E_8 &= 0 \\
\delta f_{IF}(\delta E_{12}, \delta E_{11}, \delta E_{10}) + \delta E_9 &= 0 \\
\delta f_{IF}(\delta E_{13}, \delta E_{12}, \delta E_{11}) + \delta E_{10} &= 0 \\
\delta E_{11} + \delta W_{15} &= 0
\end{aligned}
\tag{2.4}
$$

This choice also has an additional benefit that if along with the above equations, we can also have

$$\delta\sigma(W_{15}) + \delta W_{10} = 0$$

we get a 22 step collision. Later, we use this property to produce a 22 step collision.

These are the tables we get after making imposing $\delta\ W_{17} = 0$:

	∇W
0	- -
1	- -
2	- -
3	- -
4	- -
5	- -
6	- -
7	? ?
8	? ?
9	- -
10	? ?
11	- -
12	- -
13	- -
14	- -
15	? ?

	∇A
-4	- -
-3	- -
-2	- -
-1	- -
0	- -
1	- -
2	- -
3	- -
4	- -
5	- -
6	- -
7	? ?
8	- -
9	- -
10	- -
11	- -
12	- -
13	- -
14	- -
15	- -

	∇E
-4	- -
-3	- -
-2	- -
-1	- -
0	- -
1	- -
2	- -
3	- -
4	- -
5	- -
6	- -
7	? ?
8	? ?
9	? ?
10	? ?
11	? ?
12	- -
13	- -
14	- -
15	- -

Table 2.3: Incomplete Differential table for 17 step collision after analysis

This marks the end of the first phase. We have come up with a differential that is solvable and has only a few variables.

2.2.2 Second Phase

The goal in this phase is to convert the incomplete differential created in the first phase to a complete one.

2.2.2.1 Manual Analysis To Reduce Variables

We first decrease the number of variables in our equations by adding more constraints. This ensures that the search for a complete differential is easier. In the example above, we are add the following constraints:

- $nzbits(\nabla E_i) = 0$ or a for a constant value of a. This makes sure that we are dealing with a single variable all the time which makes the search easier. Note that we can not make $\delta E_7 = 0$, since it make the equations unsatisfiable.
- $subset-cancels(\Sigma_0(\delta E_7), \delta E_7)$ and $fully-cancels(\Sigma_1(\delta E_7), \delta E_7)$. We provide reasons for this choice later.

After these choices, we tried certain choices for setting $\delta E_i = 0$ and were successful for $i = 9$ and $i = 10$, i.e it was possible for us to have a set of satisfiable equations if we put $\delta E_9 = \delta E_{10} = 0$. Now, we have $nzbits(\nabla E_7) = nzbits(\nabla A_7) = nzbits(\nabla E_8) = nzbits(\nabla E_{11}) \neq \phi$ and $nzbits(\nabla E_9) = nzbits(\nabla E_{10}) = \phi$.

The application of all the constraints stated above converts the set of equations to ($\overline{A}$ indicates $\delta A = 0$):

$$\begin{aligned} \delta A_7 &= \delta E_7 \\ \delta\Sigma_0(A_7) + \delta f_{MAJ}(\delta A_7, \delta\overline{A_6}, \delta\overline{A_5}) + \delta E_8 &= 0 \\ \delta f_{MAJ}(\delta\overline{A_8}, \delta A_7, \delta\overline{A_6}) &= 0 \\ \delta f_{MAJ}(\delta\overline{A_9}, \delta\overline{A_8}, \delta A_7) &= 0 \\ \delta E_{11} - \delta A_7 &= 0, \text{and} \end{aligned} \tag{2.5}$$

$$
\begin{aligned}
\delta W_7 &= \delta E_7 \\
\delta\Sigma_1(E_7) + \delta f_{IF}(\delta E_7, \delta\overline{E_6}, \delta\overline{E_5}) + \delta W_8 &= \delta E_8 \\
\delta\Sigma_1(E_8) + \delta f_{IF}(\delta E_8, \delta E_7, \overline{\delta E_6}) &= 0 \\
\delta f_{IF}(\delta\overline{E_9}, \delta E_8, \delta E_7) + \delta W_{10} &= 0 \\
\delta f_{IF}(\delta\overline{E_{10}}, \delta\overline{E_9}, \delta E_8) + \delta E_7 &= 0 \\
\delta\Sigma_1(E_{11}) + \delta f_{IF}(\delta E_{11}, \delta\overline{E_{10}}, \delta\overline{E_9}) + \delta E_8 &= 0 \\
\delta f_{IF}(\delta\overline{E_{12}}, \delta E_{11}, \delta\overline{E_{10}}) &= 0 \\
\delta f_{IF}(\delta\overline{E_{13}}, \delta\overline{E_{12}}, \delta E_{11}) &= 0 \\
\delta E_{11} + \delta W_{15} &= 0
\end{aligned}
\tag{2.6}
$$

2.2.2.2 Solving Equations

For the first two equations in (2.6), we have two free variables of W, so these are anyway satisfiable.

The 3^{rd} equation in (2.5) can be solved by setting $\nabla A_8[i] = \nabla A_6[i]\ \forall\ i \in nzbits(\nabla A_7)$. Same can be done for the 4^{th} equation.

The 7^{th} equation in (2.6) can be solved by setting $\nabla E_{12}[i] = 0\ \forall\ i \in nzbits(\nabla E_8)$. Same can be done for the 9^{th} equation.

It is clear from the 5^{th} equation in (2.6), that $\delta E_7 + \delta E_8 = 0$ and $\nabla E_{10}[i] = 0$ $\forall$ bit positions $i \in nzbits(\nabla E_8)$. With this choice, the 2^{nd} equation from (2.5) is satisfiable. This is the reason for choosing $subset - cancels(\Sigma_0(\delta E_7), \delta E_7)$ constraint. At places where we choose bits from δE_7 for $subset - cancels$ relation to hold, we make f_{MAJ} difference 0. Whereas, at other places, we make f_{MAJ} difference non-zero. This ensures that the 2^{nd} equation is satisfied.

The 3^{rd} equation in (2.6) can be solved by setting $\nabla E_6[i] = 1\ \forall\ i \in nzbits(\nabla E_8)$. This ensures that $\delta f_{MAJ}(\delta E_8, \delta E_7, \overline{\delta E_6}) = \delta E_7$ and because of the $fully - cancels$ constraint, this is also solvable. The 4^{th} and 6^{th} equations can also be solved in a similar fashion.

It is not guaranteed that a solution is going to exist, however we do not have any contradictions yet. That is to say, it is possible to have a solution that is based on our equations and tables.

2.2.2.3 Searching a complete differential

The search for a partially complete differential now starts. We look at different choices for a partially complete differential for E_7 uniformly at random and test whether the "cancels" constraints are satisfied.

In case they are, we go on to find a complete differential for E_7. Here, we look at every possible complete differential for E_7 without '0' and '1'. Once a complete differential is fixed, a complete differential for $\delta\Sigma_1(E_7)$ and $\delta\Sigma_0(E_7)$ is also fixed. The next step is to check whether it is possible for us to fix '-' bits of ∇E_7 so that satisfy our conditions. We demonstrate this method using the example of satisfying $\delta\Sigma_1(E_{11}) + \delta E_8 = 0$:

The next step is to check if '-' bits of ∇E_{11} can be changed to '0' or '1' such that $\nabla_2\Sigma_1(E_{11})$ attains the desired value. By definition, $\nabla\Sigma_1(E_{11}[i])$ is caused by $\nabla E_{11}[(i+6)\%32]$, $\nabla E_{11}[(i+11)\%32]$, $\nabla E_{11}[(i+25)\%32]$. Let x, y, z be the causing bits and a be the output, then the following table can be generated:

x	y	z	a	conditions
u	-	-	u	$y = z$
n	-	-	u	$y \neq z$
u	u	-	u	not possible
u	n	-	u	not possible
n	u	-	u	not possible
n	n	-	u	not possible
u	u	u	u	always
u	n	u	u	not possible
n	u	u	u	not possible
n	n	u	u	always
u	u	n	u	not possible
u	n	n	u	always
n	u	n	u	always
n	n	n	u	not possible

Table 2.4: Conditions on causing bits for fixed output of Σ_1

A similar table can be created for a = 'n'. x, y, z can be substituted with the causing bits of $\nabla\Sigma_1(E_{11}[i])$ and thus the conditions necessary for $\delta\Sigma_1(E_{11})$ + $\delta E_8 = 0$ can be obtained. To check whether a solution exists, we convert this problem into a 2-SAT which can be then solved in polynomial time.

$$y = z \equiv ((\neg y \vee z) \wedge (\neg z \vee y))$$

Although, once the value of ∇E_7 is fixed, the exact value of δW_7 and δW_{10} is fixed. However, we need not add constraints related to ∇W_7, ∇W_8, and ∇W_{10} since we can decide them using the A and E values. After success in all stages mentioned above, a complete differential table can be achieved. This is one of the tables we came up with:

	∇W
0	- -
1	- -
2	- -
3	- -
4	- -
5	- -
6	- -
7	? ?
8	? ?
9	- -
10	? ?
11	- -
12	- -
13	- -
14	- -
15	u u - - - - u - n u - u - n - u - - n - - u - - - n - n - - - -

	∇A
-4	- -
-3	- -
-2	- -
-1	- -
0	- -
1	- -
2	- -
3	- -
4	- -
5	0 0 - - - - 0 - 0 0 - 0 - 0 - 1 - - 0 - - 0 - - - 0 - 0 - - - -
6	0 0 - - - - 0 - 0 0 - 0 - 0 - 0 - - 0 - - 0 - - - 0 - 0 - - - -
7	n n 0 0 1 1 n - u n 1 n 1 u 1 n 1 0 u 0 0 n 0 0 0 u 0 u 0 0 0 1
8	0 0 - - - - 0 - 0 0 - 0 - 0 - 0 - - 0 - - 0 - - - 0 - 0 - - - -
9	0 0 - - - - 0 - 0 0 - 0 - 0 - 0 - - 0 - - 0 - - - 0 - 0 - - - -
10	- -
11	- -
12	- -
13	- -
14	- -
15	- -

	∇E
-4	- -
-3	- -
-2	- -
-1	- -
0	- -
1	- -
2	- -
3	- -
4	- -
5	- -
6	1 1 - - - - 1 - 1 1 - 1 - 1 - 1 - - 1 - - 1 - - - 1 - 1 - - - -
7	n n - - - - n - u n - n - u - n - - u - - n - - - u - u - - - -
8	u u 0 1 0 0 u 0 n u 1 u 0 n 1 u 0 - n 0 1 u 0 1 0 n 0 n 0 0 1 1
9	0 0 - - - - 0 - 0 0 - 0 - 0 - 0 - - 0 - - 0 - - - 0 - 0 - - - -
10	0 0 - - - - 0 - 0 0 - 0 - 0 - 0 - - 0 - - 0 - - - 0 - 0 - - - -
11	n n 0 1 0 0 n 0 u n 1 n 0 u 1 n 0 - u 0 1 n 0 1 0 u 0 u 0 0 1 1
12	0 0 - - - - 0 - 0 0 - 0 - 0 - 0 - - 0 - - 0 - - - 0 - 0 - - - -
13	1 1 - - - - 1 - 1 1 - 1 - 1 - 1 - - 1 - - 1 - - - 1 - 1 - - - -
14	- -
15	- -

Table 2.5: Complete Differential Table for 17 step collision

2.2.3 Third Phase

The third phase takes in a complete differential table and produces the message pair. This phase uses a little tweaked version of DFS where wise choices for which children need to be visited are used. For example, in the differential above ∇A_7 is denser than ∇E_7, so we use a valid ∇A_7 and compute ∇W_7 and ∇E_7 with respect to this. We look at all possibilities of ∇A_7 since they are very less.

Another choice we make is to look at very few children for steps ¡ 6. At 6, we increase the number of children based on whether we were able to reach step 8 or not. These tweaks make the search more efficient.

Here is the message pair that follows the differential:

29dbe312	c2f1bb0f	4e107dfd	7b291f24	8bfb39ca	437f3e09	a4e9dd0a	98652468
c78f8a62	36b2f9db	767efe99	5c82322a	2f7e3830	383cb8a9	2f36112a	f5fbdeff
29dbe312	c2f1bb0f	4e107dfd	7b291f24	8bfb39ca	437f3e09	a4e9dd0a	8e818dab
656b8779	36b2f9db	80629556	5c82322a	2f7e3830	383cb8a9	2f36112a	ffdf75bc

Table 2.6: Message Pair for 17 step collision

Chapter 3

Collisions on 22 Steps

3.1 Description

The collision attack on 22 steps is an extension of the 17 step collision. Table 2.3 as the differential table. The constraints and way to solve the equations is also the same. Only the following additional constraint is added:

$$\delta\sigma_1(W_{15}) + \delta W_{10} = 0$$

The reason that the 17 step collision extends to 22 steps if the above condition is met is that this causes $\delta W_{17} = 0$ which in turn causes $\delta W_i = 0 \ \forall \ i \in \{18, 19, 20, 21\}$.

3.2 Solving Additional Constraint

The idea is to find a solution for all the other equations in the way described before and then check if $\delta\sigma_1(W_{15}) + \delta W_{10} = 0$ can be satisfied.

Once we have come up with a solution for other equations both δW_{15} and δW_{10} are fixed. And in fact with our methodology, $\delta W_{15} = \delta W_{10}$. So, we need to find a $D \in \nabla_2$ such that $\delta D = -\delta\sigma_1(D) = \delta W_{15}$.

To solve for this, we randomly fix a D such that $\delta D = \delta W_{15}$ and check if we $-\delta\sigma_1(D) = \delta W_{15}$. This check is done using a table similar to Table 2.4. This step is repeated a threshold number of times and the threshold is in the range of

10^4. Using this strategy, we were able to come up with many differentials, one of which is described below:

	∇W
0	- -
1	- -
2	- -
3	- -
4	- -
5	- -
6	- -
7	? ?
8	? ?
9	- -
10	? ?
11	- -
12	- -
13	- -
14	- -
15	n 0 n 0 0 0 n 0 - u 0 0 n 1 u n 0 n u 1 u 0 u u 1 1 1 1 n u n n

	∇A
-4	- -
-3	- -
-2	- -
-1	- -
0	- -
1	- -
2	- -
3	- -
4	- -
5	1 - 1 - - - 0 - - 1 0 - 1 - 1 - - 1 - - 1 - 0 0 - - - - 0 1 - 0
6	0 - 0 - - - 0 - - 0 0 - 0 - 0 - - 0 - - 0 - 0 0 - - - - 0 0 - 0
7	u 1 u 1 - 0 u 0 1 u n - u - u 0 1 n 0 0 n 1 n n 1 - 0 0 u u 0 u
8	0 - 0 - - - 0 - - 0 0 - 0 - 0 - - 0 - - 0 - 0 0 - - - - 0 0 - 0
9	0 - 0 - - - 0 - - 0 0 - 0 - 0 - - 0 - - 0 - 0 0 - - - - 0 0 - 0
10	- -
11	- -
12	- -
13	- -
14	- -
15	- -

	∇E
-4	- -
-3	- -
-2	- -
-1	- -
0	- -
1	- -
2	- -
3	- -
4	- -
5	- -
6	1 - 1 - - - 1 - - 1 1 - 1 - 1 - - 1 - - 1 - 1 1 - - - - 1 1 - 1
7	u - u - - - u - - u n - u - u - - n - - n - n n - - - - u u - u
8	n 0 n 0 0 0 n 1 1 n u 1 n 1 n 0 - u 0 1 u 1 u u 1 0 0 - n n 1 n
9	0 - 0 - - - 0 - - 0 0 - 0 - 0 - - 0 - - 0 - 0 0 - - - - 0 0 - 0
10	0 - 0 - - - 0 - - 0 0 - 0 - 0 - - 0 - - 0 - 0 0 - - - - 0 0 - 0
11	u 0 u 0 0 0 u 1 1 u n 1 u 1 u 0 - n 0 1 n 1 n n 1 0 0 - u u 1 u
12	0 - 0 - - - 0 - - 0 0 - 0 - 0 - - 0 - - 0 - 0 0 - - - - 0 0 - 0
13	1 - 1 - - - 1 - - 1 1 - 1 - 1 - - 1 - - 1 - 1 1 - - - - 1 1 - 1
14	- -
15	- -

Table 3.1: Complete Differential Table for 22 step collision

Using the above differential and the same technique as used for 17 steps, here is one of the collisions we came up with:

3f9b7677 7a40715a 1bdfe40d fff8fbc7 627c930c d907e336 e4b47c7d e4461b97
fb31413d 1f3f9137 e46768f4 afd36254 3cfb6d32 22ee03ad afa4ff34 2fdc6300
3f9b7677 7a40715a 1bdfe40d fff8fbc7 627c930c d907e336 e4b47c7d 3517cd52
afbb8f11 1f3f9137 9395b739 afd36254 3cfb6d32 22ee03ad afa4ff34 df0ab145

Table 3.2: Collision for 22 step collision

Chapter 4

Analysis of Σ Functions

4.1 Motivation

Often while solving equations for collisions on SHA-256, we get equations like

$$\begin{aligned} \delta\Sigma(x) &= \alpha, \\ \delta x &= \beta \end{aligned} \tag{4.1}$$

Therefore, it is interesting to know whether these equations are solvable and if yes, then how? This is the problem we answer to some extent in this chapter.

4.2 Differentials Given Additive Difference

In this section, we focus on the problem of finding differentials for variables given the additive difference between them, i.e. given

$$\delta x = \beta$$

we want to find complete differentials for x.

Theorem 1. If we have one solution for complete differential given additive difference, we can generate many solutions in most cases.

This can be done by replacing a "u-" structure in the differential to "nu" recursively, this transformation does not alter the additive difference because a

'u' at the i^{th} position contributes -2^i to δx and 'n' at the i^{th} position contributes 2^i. For example, all these complete differentials have the same additive difference:

u n - - - u - - - - - n - - - u

u n - - - n u - - - - n - - - u

u n - - - n n u - - - n - - - u

u n - - - n n u - - - u n - - u

u n - - - n n n u - - u n - - u

...

So, if we can come up with one differential, we can come up with many. To come up with one differential, we write δx in binary. If the i^{th} position has 1 in binary, we write 'n' in the differential and '-' otherwise. In case, the difference is more than 2^{31}, we get the value $2^{32} - \delta x$ and similarly build a differential by using both 'u' and 'n'.

4.3 Analysis of Σ_0

In this section, we want to check whether it is possible for us to come up with differentials for x which satisfy

$$\begin{aligned} \delta\Sigma_0(x) &= \alpha, \\ \delta x &= \beta \end{aligned}$$

Theorem 2. Given an incomplete differential for x, an incomplete differential for $\Sigma_0(x)$ can be obtained.

The Σ_0 function can be treated as a transformation which takes in 3 causing bits and outputs a single caused bit. Let us say that the causing bits are 'u', 'u', '-'. Then, the caused bit:

$$\begin{aligned} a &= 0 \oplus 0 \oplus r \\ a' &= 1 \oplus 1 \oplus r \end{aligned} \tag{4.2}$$

In case the causing bits are 'u', '-', '-':

$$a = 0 \oplus r \oplus t = 0 \oplus p$$
$$a' = 1 \oplus r \oplus t = 1 \oplus p$$

We can see that the equations (4.2) result in the caused bit(a) to be '-' and the equations (4.3) result in the caused bit(a) to be 'x'. Using a similar analysis over all possibilities, it can be seen that if an odd number of causing bits are 'x', then the caused bit is 'x' and otherwise the caused bit is '-'. Table 2.4 shows the conditions we get to convert this 'x' to a 'u' or 'n'.

The first step in search for a solution to (4.1) is to fix a random value for $\nabla_2 x$. This gives $\nabla_1 \Sigma_0(x)$. Next it needs to be checked if the bits of $\nabla_1 \Sigma_0(x)$ can be set such that the difference is α. The next step is to check whether the assignment can be generated using the Σ_0 function(This is done using table 2.4).

Since, there may not always exist a solution, the search can be done for some threshold number of times.

4.4 Experiment

The experiment was to determine the probability of finding a differential for x that satisfies (4.1) using the approach described above. This is the algorithm:

Algorithm 1: Experiment on Σ

```
Result: Average number of iterations for success
totalExp = 100;
currentExp = 0;
totalIt = 0;
THRES = 1000;
while currentExp < totalExp do
    found=false;
    iterations=0;
    while !found do
        iterations++;
        δx = rand();
        δsig = rand();
        testDiff = getRandDiff(δsig);
        testNum = 0;
        while testNum < THRES do
            testNum++;
            ∇2x = getRandDiff(δx);
            ∇1sig = getSigmaDiff(∇2x);
            if fully-cancels(∇1sig, testDiff)) then
                ∇2sig = getCompleteDiff(∇1sig, testDiff);
                if conditionsStatisfiable(∇2sig, Σ) then
                    found=true;
                    break;
                end
            end
        end
    end
    totalIt = totalIt + iterations;
end
return totalIt/totalExp;
```

Function	Output
Σ_0	31732
Σ_1	27246
σ_0	10
σ_1	19358

Table 4.1: Results of Algorithm 1

Chapter 5

Future Work

5.1 Extension of Method

A natural extension of the method is to find a different differential that works for more number of steps. While coming up with a differential for which the current approach would work the first step is to look at message expansion and find a differential for W which is satisfiable and has a few words which are different. It can be initially assumed that $\sigma(W_i)$ can take whatever value that is required. The experiments written in the 4th Chapter ensure that it is not very difficult for these conditions to work out.

The next step is to check whether the conditions caused because of the SHA-256 requirements have a solution or not. If it is possible to solve these, the starting point should lead to a collision.

One of the initial differentials we found was by moving the differential created for the 17 step collision by the end of first phase one step down:

	∇W
0	- -
1	- -
2	- -
3	- -
4	- -
5	- -
6	- -
7	- -
8	? ?
9	? ?
10	- -
11	? ?
12	- -
13	- -
14	- -
15	- -
16	? ?

	∇A
-4	- -
-3	- -
-2	- -
-1	- -
0	- -
1	- -
2	- -
3	- -
4	- -
5	- -
6	- -
7	- -
8	? ?
9	- -
10	- -
11	- -
12	- -
13	- -
14	- -
15	- -

	∇E
-4	- -
-3	- -
-2	- -
-1	- -
0	- -
1	- -
2	- -
3	- -
4	- -
5	- -
6	- -
7	- -
8	? ?
9	? ?
10	? ?
11	? ?
12	? ?
13	- -
14	- -
15	- -

Table 5.1: Incomplete Differential Table for 23 step collision

Putting similar constraints we put for 17 steps and two equations for message expansion, and solving the equations, we get the following differential: (The equations we got and how we solved them will be clear after looking at the complete differential)

	∇W
0	- -
1	- -
2	- -
3	- -
4	- -
5	- -
6	- -
7	- -
8	u - n - n - - n - nn - - uu - - - - - u - - n - uu - - - nu
9	n - u - u - - u - uu - - nn - - - - - n - - u - nn - - - un
10	- -
11	u - n - n - - n - nn - - uu - - - - - u - - n - uu - - - nu
12	- -
13	- -
14	- -
15	- -
16	n - nunnuu1unu1nuun000uunnunn111un

	∇A
-4	- -
-3	- -
-2	- -
-1	- -
0	- -
1	- -
2	- -
3	- -
4	- -
5	- -
6	1 - 0 - 0 - - 1 - 00 - - 00 - - - - - 0 - - 0 - 10 - - - 11
7	0 - 0 - 0 - - 0 - 00 - - 00 - - - - - 0 - - 0 - 00 - - - 00
8	u1n0n01n - nn0 - uu0110 - u - 0n1uu00 - nu
9	1 - 1 - 1 - - 1 - 11 - - 11 - - - - - 1 - - 1 - 11 - - - 11
10	1 - 1 - 1 - - 1 - 11 - - 11 - - - - - 1 - - 1 - 11 - - - 11
11	- -
12	- -
13	- -
14	- -
15	- -

	∇E
-4	- -
-3	- -
-2	- -
-1	- -
0	- -
1	- -
2	- -
3	- -
4	- -
5	- -
6	1 - 1 - 1 - - 1 - 1 1 - - 1 1 - - - - - 1 - - 1 - 1 1 - - - 1 1
7	0 - 0 - 0 - - 0 - 0 0 - - 0 0 - - - - - 0 - - 0 - 0 0 - - - 0 0
8	u 1 n 1 n 0 1 n 1 n n 1 0 u u 0 0 0 1 1 u 0 1 n 1 u u 0 - 1 n u
9	n 1 u 1 u 0 1 u 1 u u 1 0 n n 0 0 0 1 1 n 0 1 u 1 n n 0 - 1 u n
10	n 1 u 1 u 0 1 u 1 u u 1 0 n n 0 0 0 1 1 n 0 1 u 1 n n 0 - 1 u n
11	1 - 1 - 1 - - 1 - 1 1 - - 1 1 - - - - - 1 - - 1 - 1 1 - - - 1 1
12	u 1 n 1 n 0 1 n 1 n n 1 0 u u 0 0 0 1 1 u 0 1 n 1 u u 0 - 1 n u
13	1 - 1 - 1 - - 1 - 1 1 - - 1 1 - - - - - 1 - - 1 - 1 1 - - - 1 1
14	1 - 1 - 1 - - 1 - 1 1 - - 1 1 - - - - - 1 - - 1 - 1 1 - - - 1 1
15	- -

Table 5.2: Complete Differential Table for 23 step collision

After finding the differential, we could not find a collision for this. However, finding a semi-free start collision was way more efficient. To find a semi-free start collision, we randomly fix A, E registers values at step 7, 8, 9 and 10. We can now move backwards and forwards to solve for a semi-free start collision. In case, we get stuck at some point, choose a different set A, E values and start again. Using this strategy, this is the collision we got:

3f9b7677 7a40715a 1bdfe40d fff8fbc7 627c930c d907e336 e4b47c7d e4461b97
fb31413d 1f3f9137 e46768f4 afd36254 3cfb6d32 22ee03ad afa4ff34 2fdc6300
3f9b7677 7a40715a 1bdfe40d fff8fbc7 627c930c d907e336 e4b47c7d 3517cd52
afbb8f11 1f3f9137 9395b739 afd36254 3cfb6d32 22ee03ad afa4ff34 df0ab145

Table 5.3: Semi-Free Start Collision for 23 steps: Message Pairs

a935e5c2	991ea8ca	ac40863c	446de64a	dc34c355	e1f1393b	980fe90f	ed3b0325

Table 5.4: Semi-Free Start Collision for 23 steps: Initial Registers

So, we propose two problems for future work after this discussion:

- Find a differential for more than 23 steps.
- Develop a better way of searching for message pairs given a differential.

5.2 Alternate Method

This method is for the production of differentials fully automatically, i.e. without any manual analysis. Here, we once we are done with the first phase, we feed the equations to an algorithm and it outputs a complete differential. Let us look at the equations we had after the first phase for 17 step differential:

$$\begin{aligned}
\delta A_7 &= \delta E_7 \\
\delta\Sigma_0(A_7) + \delta f_{MAJ}(\delta A_7, \delta A_6, \delta A_5) + \delta E_8 &= 0 \\
\delta f_{MAJ}(\delta A_8, \delta A_7, \delta A_6) + \delta E_9 &= 0 \\
\delta f_{MAJ}(\delta A_9, \delta A_8, \delta A_7) + \delta E_{10} &= 0 \\
\delta E_{11} - \delta A_7 &= 0, \text{and}
\end{aligned} \tag{5.1}$$

$$\begin{aligned}
\delta W_7 &= \delta E_7 \\
\delta\Sigma_1(E_7) + \delta f_{IF}(\delta E_7, \delta E_6, \delta E_5) + \delta W_8 &= \delta E_8 \\
\delta\Sigma_1(E_8) + \delta f_{IF}(\delta E_8, \delta E_7, \delta E_6) &= \delta E_9 \\
\delta\Sigma_1(E_9) + \delta f_{IF}(\delta E_9, \delta E_8, \delta E_7) + \delta W_{10} &= \delta E_{10} \\
\delta\Sigma_1(E_{10}) + \delta f_{IF}(\delta E_{10}, \delta E_9, \delta E_8) + \delta E_7 &= 0 \\
\delta\Sigma_1(E_{11}) + \delta f_{IF}(\delta E_{11}, \delta E_{10}, \delta E_9) + \delta E_8 &= 0 \\
\delta f_{IF}(\delta E_{12}, \delta E_{11}, \delta E_{10}) + \delta E_9 &= 0 \\
\delta f_{IF}(\delta E_{13}, \delta E_{12}, \delta E_{11}) + \delta E_{10} &= 0 \\
\delta E_{11} + \delta W_{15} &= 0
\end{aligned} \tag{5.2}$$

What we have thought so far is a kind of a DFS over the equations (5.2). So, we assign a random valid value whenever we need to. After we find satisfying values for these, we test whether this is also a solution for equations (5.1). One run of this algorithm looks like this:

- Fix a value for δW_7. This fixes δE_7.
- Fix values for $\delta\Sigma_1(E_7)$ and $\delta f_{IF}(\delta E_7, \delta E_6, \delta E_5)$ and this fixes the value for δE_8. Ensure that while fixing the values, these are possible with whatever choices we have made above. For example, we should consider only those choices for $\delta\Sigma_1(E_7)$ which can be created using δE_7 chosen in the first step.
- Fix values for $\delta\Sigma_1(E_8)$ and $\delta f_{IF}(\delta E_8, \delta E_7, \delta E_6)$ and this fixes the values for δE_9
- ....
- Fix values for $\delta\Sigma_1(E_{10})$, $\delta f_{IF}(\delta E_{10}, \delta E_9, \delta E_8)$ and check if the 5^{th} equation is true. In case it is, we proceed and otherwise, we make a few more guesses at this step and move back if it still does not work.
- ....
- We have now found a satisfying assignment for (5.2), now we have fixed values of δA_i. Check if these can solve (5.1).

We have not yet tried out this approach. So we can say nothing about the results.

5.3 Better Message Search

Using the tweaked version of a DFS (described previously) for message search, we were not able to find messages for the 23 step differential table we built. Hence, this method needs improvements as well.

References

Sebastiaan Indesteege, Florian Mendel, Bart Preneel, and Christian Rechberger. Collisions and other non-random properties for step-reduced sha-256. Cryptology ePrint Archive, Report 2008/131, 2008. https://eprint.iacr.org/2008/131. 5

Florian Mendel, Tomislav Nad, and Martin Schläffer. Finding sha-2 characteristics: Searching through a minefield of contradictions, 2011. URL https://www.iacr.org/archive/asiacrypt2011/70730287/70730287.pdf. 5, 8

Florian Mendel, Tomislav Nad, and Martin Schläffer. Improving local collisions: New attacks on reduced sha-256. Cryptology ePrint Archive, Report 2015/350, 2015. https://eprint.iacr.org/2015/350. 5

Somitra Kumar Sanadhya and Palash Sarkar. Non-linear reduced round attacks against sha-2 hash family. Cryptology ePrint Archive, Report 2008/174, 2008a. https://eprint.iacr.org/2008/174. vii, 3, 5

Somitra Kumar Sanadhya and Palash Sarkar. New collision attacks against up to 24-step sha-2. Cryptology ePrint Archive, Report 2008/270, 2008b. https://eprint.iacr.org/2008/270. 5