Impact of Rotations in SHA-1 and Related Hash Functions*

Norbert Pramstaller, Christian Rechberger, and Vincent Rijmen

Institute for Applied Information Processing and Communications (IAIK),
Graz University of Technology, Austria
{Norbert.Pramstaller, Christian.Rechberger,
Vincent.Rijmen}@iaik.tugraz.at

Abstract. SHA-1 uses a single set of rotation constants within the compression function. However, most other members of the MD4 family of hash functions use multiple sets of rotation constants, *i. e.* the rotation amounts change with the step being processed.

To our knowledge, no design rationales on the choice of rotation constants are given on any of these hash functions. This is the first paper that analyzes rotations in iterated hash functions. We focus on SHA-1-like hash functions and use recent developments in the analysis of these hash functions to evaluate the security implications of using multiple sets of rotation constants in the compression function instead of a single set. Additionally, we give some observations on the set of constants used in SHA-0 and SHA-1.

1 Introduction

SHA-0 was introduced in 1993 and SHA-1 was introduced in 1995 without giving any rationales on the design. Both are based on the MD4 design strategy, however the used message expansions are more complex. Additionally, a *single* set of rotation constants instead of *multiple* sets are used during state update, *i. e.* the rotation constants remain the same for all steps. Later on, in 1998, the hash function HAS-160 was specified for use in South Korea's Digital Signature Algorithm. The structure of HAS-160 is very similar to SHA-1. However, one distinct feature is the use of multiple sets of rotation constants (as in MD4 and MD5) instead of a single set. Several questions are open so far:

1. Why were the rotation constants for SHA-1 chosen as they are?
2. Would there be better choices for these constants from a security point of view?
3. Is there a security advantage of using multiple sets of rotation constants instead of a single set?

We attempt to give some answers to these questions. To our knowledge this is the first article which deals with the issue of rotation constants in iterated hash

* The work in this paper has been supported by the Austrian Science Fund (FWF), project P18138.

B. Preneel and S. Tavares (Eds.): SAC 2005, LNCS 3897, pp. 261–275, 2006.

functions. The outline and main contributions of this article are as follows. In Section 2, we give a short description of SHA-1 and HAS-160. Afterwards, in Section 3, we review and comment on currently known analysis strategies for SHA-1. This review serves as an important starting point for some comparisons done later in this article. Looking at HAS-160, we see that due to its non-recursive message expansion the basic building block for most of these strategies (elementary collisions as introduced by [3]) can not be directly applied. However, the Rijmen-Oswald extension [15] can be used to overcome these problems.

Afterwards we turn to the influence of multiple sets of rotation constants. We analyze the effect of multiple sets of rotation constants in simplified models in Section 4. We show that these multiple sets improve the avalanche effect in the first steps of our simplified, linearized model.

Section 5 contains the main contribution. We compare single and multiple sets of rotation constants in hash functions like SHA-1, HAS-160 and variations thereof. We identify two reasons why the complexity of an attack increases when multiple sets are used. Firstly, we show that the weight of collision-producing differences increases and secondly, we show that more conditions are needed due to an optimization trick which is less effective with multiple sets. Here, we also give a first observation on the design of SHA-1. For 80 or more steps, the benefits of multiple sets over a single set of rotation constants is negligible (in terms of Hamming weight of a collision-producing difference). Additionally, we analyze the attack complexity for variants of SHA-1 having different single sets of rotation constants. We show that in the case of full SHA-1 (80 steps), rotating the chaining variable A by *5* to the left and chaining variable B by *2* to the right are the smallest possible values which do not impair security. Finally, we discuss advantages of having these small constants.

1.1 Used Notation and Terminology

Table 1 contains a description of symbols used throughout this article. Note that if negative integers are used as rotation constants, the rotation direction is reversed from left to right. Whenever we talk about Hamming weight of differential

Table 1. Used notation

notation	description
$A \oplus B$	addition of A and B modulo 2 (XOR)
$A + B$	addition of A and B modulo 2^{32}
$A \vee B$	logical OR of two bit-strings A and B
M_t	input message word t (32-bits), index t starts with 0
W_t	expanded input message word t (32-bits), index t starts with 0
$A \lll n$	bit-rotation of A by n positions to the left, $0 \leq n \leq 31$
step	the SHA-1 compression function consists of 80 steps
round	the SHA-1 compression function consists of 4 rounds $= 4 \times 20$ steps
N	number of steps of the compression function

patterns we refer to the smallest Hamming weight we found using an adapted version [13] of Leon's algorithm [7] for finding low-weight words in linear codes.

2 Description of Used Hash Functions

In this section, we shortly describe SHA-1 and the differences of HAS-160 compared to SHA-1.

2.1 SHA-0 and SHA-1

The SHA family of hash functions is described in [11]. Briefly, their compression function consists of two phases: a message expansion and a state update transformation. These phases are explained in more detail in the following. SHA-0 and SHA-1 share the same state update, but SHA-0 has a simpler message expansion. Both SHA-0 and SHA-1 consist of 80 steps. Since we will study variable-step versions in this article, we denote the number of steps by N.

Message Expansion. In SHA-1, the message expansion is defined as follows. The input is a 512-bit message, denoted by a row vector m. The message is also represented by 16 32-bit words, denoted by M_t, with $t = 0, 1, \ldots, 15$.

In the message expansion, this input is expanded linearly into N 32-bit words W_t, also denoted as the $32N$-bit expanded message word w. The words W_t are defined as follows.

$$W_t = M_t, \quad t = 0, \ldots, 15 \tag{1}$$

$$W_t = (W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16}) \lll 1, \quad t > 15 \tag{2}$$

The message expansion of SHA-0 is very similar, but uses:

$$W_t = W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16}, \quad t > 15\ . \tag{3}$$

Consequently, a bit at a certain position i in one of the words of w only depends on the bits at corresponding positions in the words of m.

State Update Transformation. The state update transformation starts from a (fixed) initial state for 5 32-bit registers and updates them in N steps, using one word W_t in every step. Figure 1 illustrates one step of the state update transformation. The function f depends on the step number: steps 0 to 19 (round 1) use the *IF-function* and steps 40 to 59 (round 3) use the *MAJ-function*.

$$f_{\mathrm{if}}(B, C, D) = BC \oplus \overline{B}D \tag{4}$$

$$f_{\mathrm{maj}}(B, C, D) = BC \oplus BD \oplus CD \tag{5}$$

The remaining rounds use a 3-input XOR. A round constant K_t is added in every step. There are four different constants; one for each round. After the last application of the state update transformation, the initial register values are

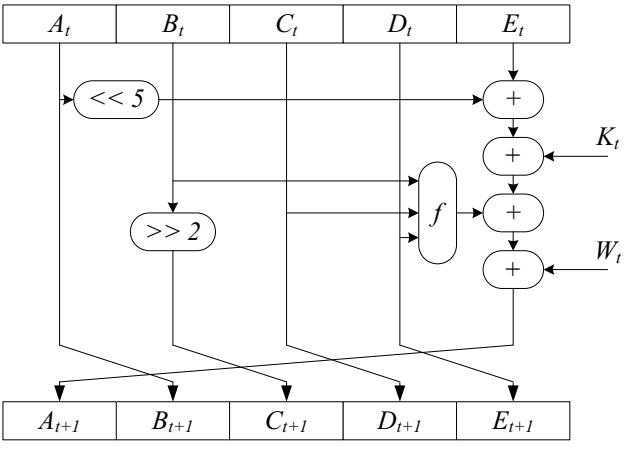

Fig. 1. One step of the state update transformation of SHA-1

added to the final values, and the result is either the input to the next iteration function or the final digest.

2.2 HAS-160

HAS-160 [17] is designed for use with the South Korean KCDSA digital signature algorithm [16]. The output length is 160 bits. A security evaluation of KCDSA by Lim *et al.* can be found in [9, 5]. An independent English description of HAS-160 is available [10, 8]. HAS-160 can be seen as a predecessor of the HAS-V family of hash functions proposed in [12]. The design is based on SHA-1, however some features are distinct. Subsequently, only the differences to SHA-1 are described.

Round Constants. HAS-160 uses a different set of round constants. We do not need their actual values in this article.

Message Expansion. In SHA-0 and SHA-1, 16 input message words M_t are expanded into 80 expanded message words W_t using a recursive definition. In HAS-160, the 16 input words are expanded into 20 words (differently for each round) and permuted for each of the four rounds. For actual permutation tables and expansion tables, refer to [10, 8].

Boolean Functions in the State Update. The only difference to SHA-1 is the 3-input Boolean function used for steps 40-59. We denote this function f_{has3}.

$$f_{\text{has3}}(B, C, D) = C \oplus (B \vee \overline{D}) \tag{6}$$

The impact of this difference with respect to collision-search attacks is analyzed in Section 5.2.

Rotations in the State Update. In SHA-0 and SHA-1, the chaining variable A_t is rotated by 5 bit-positions to the left before it is input to a modular addition.

In HAS-160, this single rotation constant is replaced by multiple constants, *i. e.* each step within a round rotates A_t differently. The actual values are

$$S_1(t \bmod 20) = \{5, 11, 7, 15, 6, 13, 8, 14, 7, 12, 9, 11, 8, 15, 6, 12, 9, 14, 5, 13\} , \quad 0 \leqslant t \leqslant 79 . \tag{7}$$

In SHA-0 and SHA-1, the chaining variable B_t is rotated by 30 bit-positions to the left before it becomes variable C_{t+1}. In HAS-160, this single rotation constant is replaced by multiple rotation constants for each round. The actual values are

$$\begin{aligned} S_2(t) &= 10, \quad 0 \leqslant t \leqslant 19 , \\ S_2(t) &= 17, \quad 20 \leqslant t \leqslant 39 , \\ S_2(t) &= 25, \quad 40 \leqslant t \leqslant 59 , \\ S_2(t) &= 30, \quad 60 \leqslant t \leqslant 79 . \end{aligned} \tag{8}$$

Note that this concept of having multiple sets of rotation constants is different to what is referred to as data dependent rotations (DDR).

3 Outline of Recent Attacks on SHA-0 and SHA-1

In this section we give an overview and comment on all the analysis techniques that were used in recent years to analyze SHA-0 or SHA-1. The content of this section is the basis for our approach to compare variants of SHA-1 later in this article.

3.1 Differential Characteristics

Most recent collision attacks use the following strategy. Firstly, a differential characteristic through the compression function of the hash function is constructed. Secondly, messages are constructed, which follow the characteristic.

3.2 Original Chabaud-Joux Approach

In the original approach of Chabaud and Joux [3], the differential characteristic is determined by constructing a linear approximation for all the nonlinear elements of SHA. Subsequently, Chabaud and Joux look for a differential characteristic through this linear approximation. Since a differential characteristic propagates in a deterministic way through a linear function, the characteristic is determined completely by the choice of input difference. Hence, there are 2^{512} different characteristics. A fraction of 2^{-160} of these, results in a zero output difference (a collision).

Chabaud and Joux use the same linear approximation in every step. Consequently, every local collision contains the same number of corrections, *i. e.* 5. They impose the additional constraint that the pattern of perturbations is a valid expanded message, which accounts for another reduction factor 2^{-160}. Hence, there remain 192 "free" bits.

3.3 Rijmen-Oswald Extension

In [15], it is proposed to drop the condition that the perturbation pattern should be a valid expanded message. Any *collision-producing difference*, *i. e.* input difference that produces output difference zero in the linearized model, can be used. This approach increases the number of free bits to 352. The approach still results in collisions that are linear combinations of local collisions, each consisting of a perturbation and 5 corrections, but there are now less restrictions on the perturbation pattern.

Until now, it hasn't been demonstrated that this extension can result in better differential characteristics for SHA-1. However, for other hash functions, the improvement could be significant.

3.4 Multi-block

In multi-block collisions, we can also use differentials that don't result in a zero output. For instance, in a two-block collision, all we require is that the output difference in both blocks is equal, because then, final feed-forward will result in cancelation of the differences (with a certain probability). For a z-block collision, we get $512z-160$ free bits ($512z-320$ if we require that the perturbation pattern is a valid expanded message).

3.5 Exploiting Non-linearity—Improvements by Wang *et al.*

If we study the differentials used by Wang *et al.* [21, 19], then we see that they create even more freedom by allowing differential characteristics that don't follow the linear approximation in the first steps. The propagation of differences through nonlinear functions is non-deterministic and this can be exploited. The possibility to exploit non-linear behavior is also observed in [2]. The Rijmen-Oswald extension can be adapted to exploit this additional freedom.

3.6 Removing Conditions

For the second step of the attack, constructing a pair of messages that follows this characteristic, a number of conditions on message words and intermediate chaining variables need to be fulfilled. As already observed in [3], conditions on the first steps can be pre-fulfilled. Using the fact that their exist neutral bits in the compression function, this approach was extended to cover the first 20-22 steps of SHA-0 [1]. Wang *et al.* employ a different technique called message modification in their collision-search attacks on SHA-0 [21] and SHA-1 [19] to pre-fulfill the conditions in more than 20 steps. Note that a variant of this technique is also used in the analysis of MD4 and MD5 [18, 20, 6].

3.7 Decrease Final Search Complexity

There are still several ways to speed up the attack. Firstly, it is advantageous to choose the bit position of message differences to be in the MSB, since there a possible change in the carry has no effect. The reason for this is that in this case

no condition on message words and chaining variables are necessary to prevent this carry. A simple optimization is therefore to rotate each word of the difference such that the number of MSBs of value 1 is maximized.

A second trick deals with the implementation of the final search. After pre-fulfilling some conditions, the remaining conditions can only be fulfilled using random trials. There are two natural ways to talk about the time-complexity of this final search. One is to use the numbers of message pairs needed. In this case, the time complexity can be estimated by 2^c, where c corresponds to the number of conditions that cannot be pre-fulfilled. Another way of looking at it is to use the number of steps as a means to express time complexity. It seems natural to define the time-complexity 1 to be the N steps of the compression function.

A "good" pre-computed 13-word or 14-word message-pair can be used as a starting point. Depending on the conditions on the message words m_{14}, m_{15} and m_{16} we get up to 96 degrees of freedom for our final search. If all degrees of freedom are used without finding a collision, a new pre-computed message pair is needed. However, we can stop our step-computations after the first condition with probability $p_1 = 1/2$, after the second condition $p_2 = 1/4$ and so on. Since we assume random trials for conditions we cannot pre-fulfill, we always estimate the probability for fulfilling the conditions to be 0.5.

Starting from step 13 or 14, on average 10 steps are enough. Therefore we estimate the time-complexity of our final search to be 2^{c-3} (for $N = 80$). Joux *et al.* [4] use a similar reasoning and arrive at 2^{c-2}. The difference is that there the same amount of computation is assumed for the second message pair. However, the steps for the second message pair are only needed if all conditions are fulfilled in order to check if the messages really collide.

3.8 Application of Attacks to HAS-160

Due to the non-recursive structure of the message expansion of HAS-160, a direct application of the Chabaud-Joux approach is not possible. However, using the approach described in [15], generating a differential characteristic for HAS-160 is straightforward. Some details on constructing messages that follow this characteristic are given in Section 5.2.

4 Rotations During the Step Update—Analysis of Simplified Models

The original step update function of HAS-160 is defined as follows:

$$A_{t+5} = (A_{t+4} \lll S_0) + f(A_{t+3}, A_{t+2} \lll S_1, A_{t+1} \lll S_1) + A_t \lll S_1 + W_t + K_t, \tag{9}$$

whereas S_0 has different values for each step of a round and S_1 has different values for different rounds as defined in Section 2.2. Note that the SHA-1 state update has the same structure, but S_0 has always a value of 5 and S_1 has always a value of 30.

In order to analyze the impact of the multiple constants in S_0, we use a simplified model of the state update. We replace the modular addition by an XOR and the function f by a 3-input XOR. In a first approximation, we consider only two chaining variables, and therefore only one bit-rotation.

$$A_{t+2} = (A_{t+1} \lll S_0) \oplus A_t \tag{10}$$

If we introduce a single-bit difference in one of the chaining variables of Equation 10, we observe an increase of the Hamming weight of this difference with increased number of steps. Our first observation is that whenever S_0 is constant and a multiple of 2, the number of affected bits decreases. Summing over all 80 steps, we get a maximum of 283 affected bits in the chaining variables when we apply a single-bit difference in the first chaining variable. When we do the same computations for multiple rotation constants we get a total of 1077 affected bits. Note that if half of the bits would be affected in each step, we would arrive at $80 \times 16 = 1280$ affected bits.

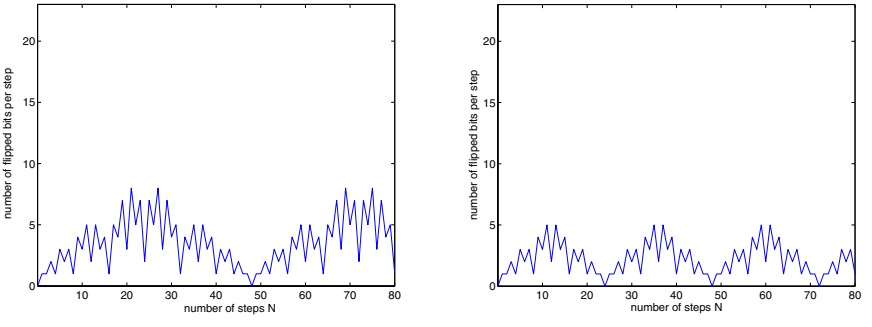

Fig. 2. Number of affected bits per step for constant bit-rotations. The constant is not a multiple of 2 in the left figure. In the right figure, the used constant is a multiple of 2.

Figure 2 gives another point of view: the number of affected bits per step for a single rotation constant. Here we distinguish between cases where the value for the bit-rotations is a multiple of 2, and where this is not the case. The symmetry in Figure 2 can be explained by our simplified and linearized model. If we apply the same method to compute the number of affected bits for the case of multiple rotation constants, we get the result shown in Figure 3.

We observe a much steeper increase in the first rounds. Due to the multiple rotation constants, differences do not cancel out early. Later on, the ideal 16 affected bits per round are reached.

Next, we extend our model to three chaining variables to contain the second bit-rotation of variable B. The resulting equation is as follows:

$$A_{t+3} = (A_{t+2} \lll S_0) \oplus A_{t+1} \oplus A_t \lll S_1 \ . \tag{11}$$

Our simulation results and conclusions are pretty similar to the case for two chaining variables. Therefore we omit them. However, we found an example

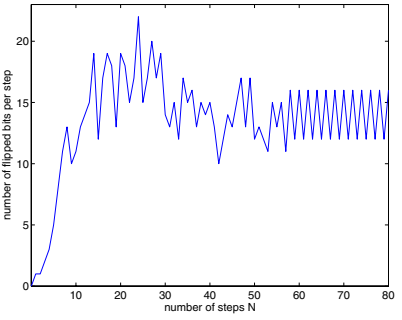

Fig. 3. Number of affected bits per step for multiple rotation constants

were we "outperformed" the ideal case: we used a constant 24-bit rotation for A and "pseudo-random" rotations for B. Using this setting, we arrived at 1352 bit-flips after 80 steps.

5 Impact of Multiple Rotation Constants on the Attack Complexity

In this section, we are comparing several variants of SHA-1. We use the approach described in [15] to find low-weight input differences, which in turn can be used to analyze the complexity of a collision-search attack. Even if we consider the recent results by Wang *et al.* , comparing Hamming weights using this method is sound since the underlying principle is the same.

Quote from [12]: "*The variable shift amount seems to provide better immunity against attacks such as differential collision in SHA-0 [3]. The generalization of inner collisions to a full compression function seemed to be harder with variable shift amounts.*"

The method of [3] assumes a message expansion defined by a recursion, which is a reason for the difficulties of applying this approach to HAS-160. However, these problems are overcome if the Rijmen-Oswald extension is applied.

Multiple rotation constants account for a slightly increased Hamming weight of collision-producing differences, which in turn slightly increases the number of conditions that have to be fulfilled in the final search for a collision. Later on, we will show that this increase is negligible after 80 or more steps. There are two reasons why multiple rotation constants result in higher collision-search complexity:

1. Higher Hamming weight of the collision-producing difference in the linearized model
2. It is less likely to take advantage of some condition-reducing effects.

5.1 Higher Hamming Weight of the Collision-Producing Difference in the Linearized Model

We consider the first point now. In order to study the effect of different rotation constants in an actual design, we searched for low-weight collision-producing

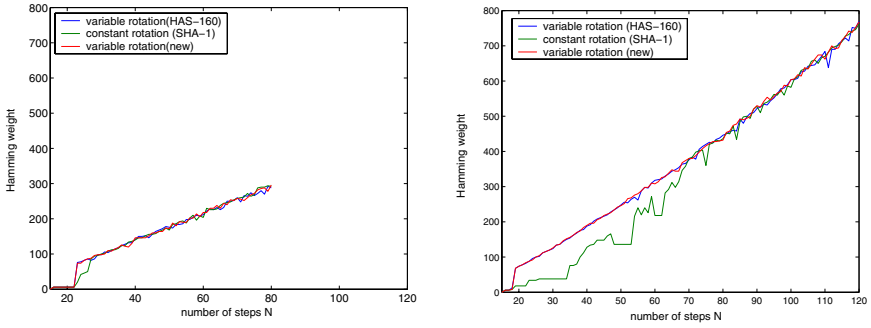

Fig. 4. Weight of collision-producing differences for single and multiple sets of rotation constants. On the left, the HAS-160 message expansion is computed for up to 80 steps. On the right, the SHA-1 message expansion is computed for up to 120 steps.

differences in variants of SHA-1, where we slightly changed the state update transformation.

Firstly, we compare the state update transformations used by SHA-1 and HAS-160. The result is depicted in Figure 4.

We consider three different state update variations. The original HAS-160 state update having multiple sets of rotation constants, the original SHA-1 state update having a single set of rotation constants and a new state update having different multiple sets of rotation constants. These variations of the state update are combined with both the HAS-160 message expansion (depicted on the left) and the SHA-1 message expansion (depicted on the right).

When looking at the values, we observe that using the HAS-160 message expansion instead of the SHA-1 message expansion actually decreases the best found Hamming weight. We also see that the lower Hamming weights for versions using a single set of rotation constants catch up on the Hamming weights of the variants with multiple sets with increased number of steps. In the case of the HAS-160 message expansion, this happens after 30 steps. In the case of SHA-1 the difference between single and multiple sets of rotation constants vanishes after 80 steps. This gives us a first hint on the choice made by the designers of SHA.

Observation 1. *The difference between a single set rotation constants and multiple sets of rotation constants vanishes with increased number of steps. In contrast to the HAS-160 message expansion, the SHA-1 message expansion delays this process until step 80.*

Secondly, we evaluate the effect of different single sets of rotation constants for SHA-1. Instead of rotating variable A by 5 positions to the left, we evaluated the attack complexity for all possibilities from 0-31. The results are depicted in Figure 5. In the step-reduced version, we see considerable differences for the chosen rotation constants of A. The constant 5, which was chosen for SHA-1, is in this setting favorable for the attacker. However, with increased number

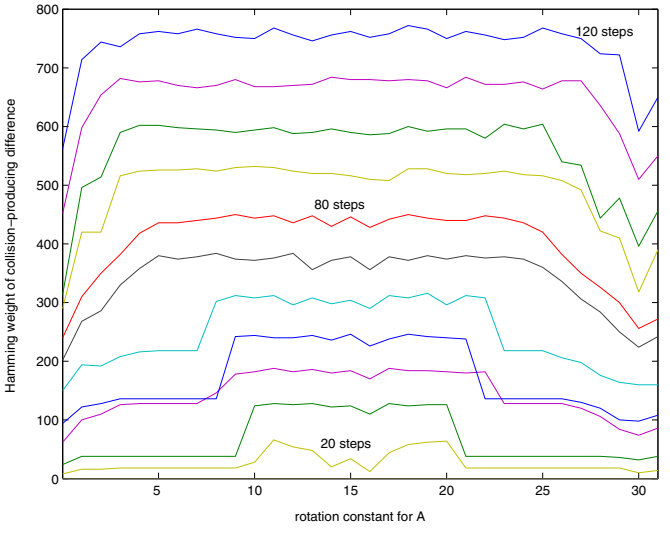

Fig. 5. Hamming weight of collision-producing differences for all possible bit-rotations of A and 20 to 120 steps of SHA-1

of steps, this advantage vanishes. After 80 steps, five bit-rotations are already enough to arrive at the plateau of Hamming weights found.

We apply the same technique for chaining variable B. Instead of rotating variable B by 30 positions to the left, we again evaluated the attack complexity for all possibilities from 0-31. The results are depicted in Figure 6.

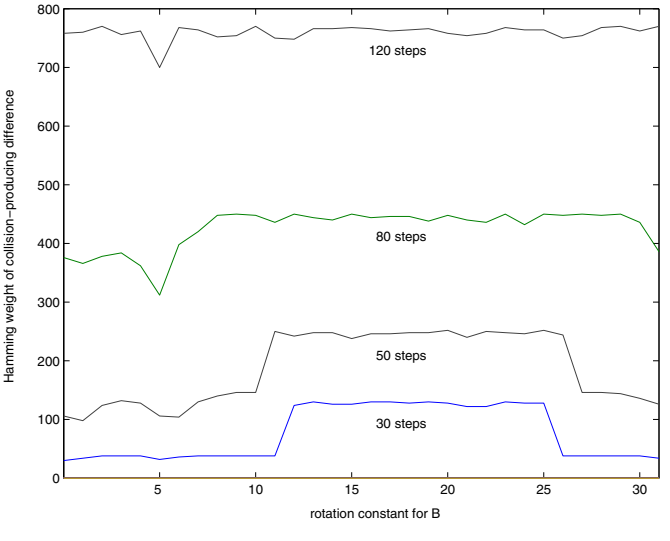

Fig. 6. Hamming weight of collision-producing differences for all possible bit-rotations of B and 30 to 120 steps of SHA-1

In the step-reduced version, we see considerable differences for the chosen rotation constants of B. The constant 30, which was chosen for SHA-1, is in this setting again favorable for the attacker. However, with increased number of steps, this advantage vanishes. After 80 steps, the value 30 (or -2) is already enough to arrive at the plateau of Hamming weights found.

Observation 2. *In the case of full SHA-1 (80 steps),* 5 *is the lowest possible value for rotating A and* 30 *is the highest possible value for rotating B to result in comparatively high Hamming weights for collision-producing differences.*

The advantage of having these constants is as follows: Let's consider platforms where constant-time shifters or rotators (see *e. g.* [14]) are neither implemented in hardware nor as microcode. There, rotating B by *e. g.* 2 positions to the right instead of more is faster. Note that these observations cannot be seen as a design criterium for the SHA family since they do not apply to SHA-0. Refer to Appendix A for details.

5.2 Impact of Multiple Rotation Constants on the Condition Generating Phase

Let us now consider the second point mentioned above: the assumption, that it is less likely to take advantage of some condition-reducing effects due to multiple sets of rotation constants. This refers to the second step of our analysis: deriving conditions on chaining variables and input message words to make the real hash function behave like the linearized model.

Before looking at the effect of multiple sets of rotation constants, the effect of the new non-linear Boolean function introduced in HAS-160 is analyzed: the f_{MAJ} function used in SHA-1 has the nice property (for an attacker) that whatever (non-zero) input difference is applied, it is always possible to find conditions on the inputs which modifies the output difference towards an XOR-like behavior. This ensures that every possible collision-producing message difference in the linearized model can lead to a real collision, assuming a number of conditions is fulfilled.

Table 2. Conditions that need to be fulfilled in order to have a differential behavior identical to that of an XOR

input differences	f_{xor}	f_{if}	f_{maj}	f_{has3}
000	0	*always*	*always*	*always*
001	1	$B = 0$	$B \oplus C = 1$	$B = 0$
010	1	$B = 1$	$B \oplus D = 1$	*always*
011	0	*never*	$C \oplus D = 1$	$B = 0$
100	1	$C \oplus D = 1$	$C \oplus D = 1$	$D = 1$
101	0	$B \oplus C \oplus D = 0$	$B \oplus D = 1$	$B \oplus D = 0$
110	0	$B \oplus C \oplus D = 0$	$B \oplus C = 1$	$D = 1$
111	1	$C \oplus D = 0$	*always*	$B \oplus D = 0$

As illustrated in Table 2, the new Boolean function does not increase the difficulty for an attacker to find conditions. As opposed to f_{if} (input difference 011), we can always find conditions on the inputs of f_{has3} to make it behave like an XOR for all input differences.

The new Boolean function does not put additional hurdles for an attack. Due to multiple sets of rotation constants aligning differences to optimize the carry-drop effect (see Section 3.7) is less effective. At this point, it is difficult to estimate the influence on the attack complexity compared to SHA-1, since the bit-rotation of the SHA-1 message expansion has a similar effect.

6 Conclusion

We have analyzed the effect of multiple sets of rotation constants in HAS-160 and compared them to the single set of rotation constants used in SHA-1. The bottom line is that multiple sets increase the attack complexity, the difference to a single set however vanishes for increased number of steps. In our comparisons, the Hamming weight of collision-producing differences in a linearized model was used as a means to compare attack complexities on a relative scale. We also gave some observations on the design of the compression function of SHA-1. For 80 or more steps of SHA-1, the benefits of having multiple sets of rotation constants instead of a single set are negligible. We finally observe that the chosen values for rotations used in the state update of SHA-1 are *on the edge* as far as the provided security level is concerned. Without impairing security, rotating the chaining variable A by *5* to the left and chaining variable B by *2* to the right are the smallest possible values. Platforms without constant-time shifters benefit from this choice.

References

1. Eli Biham and Rafi Chen. Near-Collisions of SHA-0. In Matthew K. Franklin, editor, *Advances in Cryptology - CRYPTO 2004, 24th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 2004, Proceedings*, volume 3152 of *LNCS*, pages 290–305. Springer, 2004.
2. Eli Biham, Rafi Chen, Antoine Joux, Patrick Carribault, Christophe Lemuet, and William Jalby. Collisions of SHA-0 and Reduced SHA-1. In Ronald Cramer, editor, *Advances in Cryptology - EUROCRYPT 2005: 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005. Proceedings*, volume 3494 of *LNCS*, pages 36–57. Springer, 2005.
3. Florent Chabaud and Antoine Joux. Differential Collisions in SHA-0. In Hugo Krawczyk, editor, *Advances in Cryptology - CRYPTO '98, 18th Annual International Cryptology Conference, Santa Barbara, California, USA, August 23-27, 1998, Proceedings*, volume 1462, pages 56–71. Springer, 1998.
4. Antoine Joux, Patrick Carribault, William Jalby, and Christophe Lemuet. Full iterative differential collisions in SHA-0, 2004. Preprint.

5. KCDSA Task Force Team. The Korean Certificate-based Digital Signature Algorithm, 1998. Available at `http://grouper.ieee.org/groups/1363/P1363a/contributions/kcdsa1363.pdf`.
6. Vlastimil Klima. Finding MD5 Collisions on a Notebook PC Using Multi-message Modifications, 2005. Preprint, available at `http://eprint.iacr.org/2005/102`.
7. Jeffrey S. Leon. A probabilistic algorithm for computing minimum weights of large error-correcting codes. *IEEE Transactions on Information Theory*, 34(5):1354–1359, 1988.
8. Chae Hoon Lim. The revised version of KCDSA, 2000. Unpublished Manuscript, available at `http://dasan.sejong.ac.kr/~chlim/pub/kcdsa1.ps`.
9. Chae Hoon Lim and Pil Joong Lee. A Study on the Proposed Korean Digital Signature Algorithm. In Kazuo Ohta and Dingyi Pei, editors, *Advances in Cryptology - ASIACRYPT '98, International Conference on the Theory and Applications of Cryptology and Information Security, Beijing, China, October 18-22, 1998, Proceedings*, volume 1514 of *Lecture Notes in Computer Science*, pages 175–186. Springer, 1998.
10. Jack Lloyd. A Description of HAS-160, 2003. Available at `www.randombit.net/papers/has160.html`.
11. National Institute of Standards and Technology (NIST). FIPS-180-2: Secure Hash Standard, August 2002. Available online at `http://www.itl.nist.gov/fipspubs/`.
12. Nan Kyoung Park, Joon Ho Hwang, and Pil Joong Lee: HAS-V: A New Hash Function with Variable Output Length. In Douglas R. Stinson and Stafford E. Tavares, editors, *Advances in Cryptology - CRYPTO '98, 18th Annual International Cryptology Conference, Santa Barbara, California, USA, August 23-27, 1998, Proceedings*, volume 2012, pages 202–216. Springer, 2001.
13. Norbert Pramstaller, Christian Rechberger, and Vincent Rijmen. Exploiting Coding Theory for Collision Attacks on SHA-1. In *Cryptography and Coding, 10th IMA International Conference, Cirencester, UK, December 19-21, 2005, Proceedings to appear*, LNCS. Springer, 2005.
14. Jan M. Rabaey. *Digital Integrated Circuits*. Prentice Hall, 1996.
15. Vincent Rijmen and Elisabeth Oswald. Update on SHA-1. In Alfred Menezes, editor, *Topics in Cryptology - CT-RSA 2005, The Cryptographers' Track at the RSA Conference 2005, San Francisco, CA, USA, February 14-18, 2005, Proceedings*, volume 3376 of *LNCS*, pages 58–71. Springer, 2005.
16. TTA. Digital Signature Mechanism with Appendix - Part 2 : Certificate-based Digital Signature Algorithm, TTAS.KO-12.0011/R1, 2000.
17. TTA. Hash Function Standard - Part 2: Hash Function Algorithm Standard (HAS-160), TTAS.KO-12.0011/R1, 2000.
18. Xiaoyun Wang, Xuejia Lai, Dengguo Feng, Hui Chen, and Xiuyuan Yu. Cryptanalysis of the Hash Functions MD4 and RIPEMD. In Ronald Cramer, editor, *Advances in Cryptology - EUROCRYPT 2005: 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005. Proceedings*, volume 3494 of *LNCS*, pages 1–18. Springer, 2005.
19. Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu. Finding Collisions in the Full SHA-1. In Victor Shoup, editor, *Advances in Cryptology - CRYPTO 2005, 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings*, volume 3621 of *LNCS*, pages 17–36. Springer, 2005.

20. Xiaoyun Wang and Hongbo Yu. How to Break MD5 and Other Hash Functions. In Ronald Cramer, editor, *Advances in Cryptology - EUROCRYPT 2005: 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005. Proceedings*, volume 3494 of *LNCS*, pages 19–35. Springer, 2005.
21. Xiaoyun Wang, Hongbo Yu, and Yiqun Lisa Yin. Efficient Collision Search Attacks on SHA-0. In Victor Shoup, editor, *Advances in Cryptology - CRYPTO 2005, 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings*, volume 3621 of *LNCS*, pages 1–16. Springer, 2005.

A Single Sets of Rotation Constants for SHA-0

We evaluate the effect of different single sets of rotation constants for SHA-0. Instead of rotating variable A by 5 positions to the left, we evaluated the attack complexity for all possibilities from 0-31. The results are depicted in Figure 7.

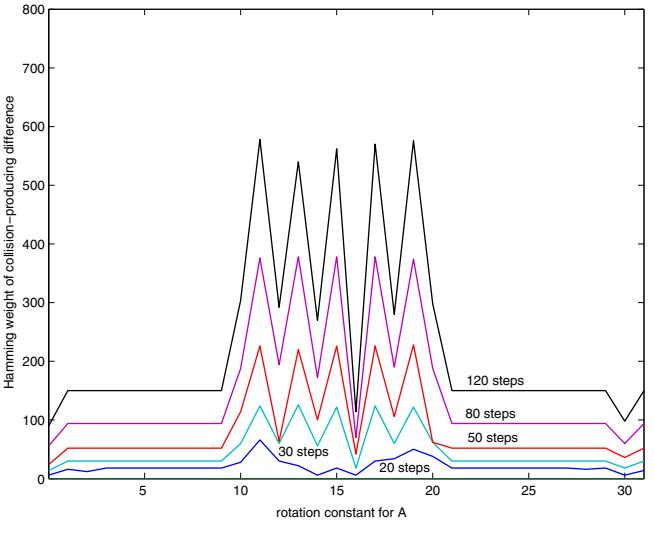

Fig. 7. Hamming weight of collision-producing differences for all possible bit-rotations of A and 20 to 120 steps of SHA-0

Using the Hamming weight for the rotation constant 5 as a starting point, we see that higher as well as lower Hamming weights for collision-producing differences are possible when choosing different rotation constants. This holds for all considered variants from 20 to 120 steps.