

Security Evaluation of SHA-224, SHA-512/224, and SHA-512/256

Christoph Dobraunig Maria Eichlseder Florian Mendel

Institute for Applied Information Processing and Communications
Graz University of Technology
Inffeldgasse 16a, 8010 Graz, Austria
krypto@iaik.tugraz.at

February 2015

Executive Summary

The truncated variants of the SHA-2 family (SHA-224, SHA-512/224, SHA-512/256, SHA-384) have received significantly less public cryptanalysis compared to the untruncated variants, SHA-256 and SHA-512. This is particularly true for the two newest members of the SHA-2 family, SHA-512/224 and SHA-512/256. However, the two new members (added in FIPS PUB 180-4 [22] in 2012) offer a number of very attractive features for applications: Their wide-pipe construction offers additional security compared to SHA-224 and SHA-256, while their 64-bit-oriented design even performs better on modern architectures. In this report, we review published cryptanalysis on SHA-2 for its applicability to the three evaluation candidates and present new results regarding the collision resistance of SHA-224, SHA-512/224, and SHA-512/256. Below, we give a summary of our findings.

Generic attacks on iterated hash functions. Many generic attacks that have been applied successfully to members of the SHA-2 family are based on its underlying Merkle-Damgård construction with its narrow internal state. Due to the final truncation applied by SHA-512/224 and SHA-512/256, their internal state size is at least twice the output length (wide pipe construction), so many of these generic attacks offer no advantage. In particular, neither Joux’ multicollision attack [10], nor Kelsey and Kohno’s herding and Nostradamus attacks [11], nor Kelsey and Schneier’s second preimages for long messages [12] are faster than brute force.

For SHA-224, on the other hand, the attacks are similarly successful as for SHA-256: Joux’ multicollision attack is applicable; the complexity of 2^{172} for the Nostradamus attack is the same as for SHA-256, thus significantly faster than brute force; and second preimages for long messages can be found faster than brute force for messages of $\geq 2^{32}$ blocks. The latter is also reflected by NIST SP 800-107 [6], NIST’s recommendations for the application of the SHA-2 family, which lists SHA-224’s second preimage resistance as 201–224 bits, depending on the target message length (security $\leq \min\{224, 256 - M\}$ for messages of 2^M blocks).

Preimage attacks. The preimage attacks covering the most steps of the SHA-2 family are variants of meet-in-the-middle attacks. Most of the published attacks target SHA-256 and SHA-512. However, Aoki et al. [1] published pseudo-preimage attacks on SHA-256 and SHA-512, which they extended to 43-step SHA-224 with a complexity of $2^{219.9}$. To

the best of our knowledge, this is the pseudo-preimage attack covering the most steps for SHA-224 published so far. Unfortunately, no cryptanalysis regarding the (pseudo) preimage resistance of SHA-512/224 and SHA-512/256 has been published yet. Aoki et al. do extend their analysis to SHA-384, however, and it seems that a similar advantage can also be gained when applying the attack to SHA-512/224 and SHA-512/256, but further investigation is required for detailed complexity estimations.

When looking at other members of the SHA-2 family, more results are available. For SHA-256 and SHA-512, Khovratovich et al. [13] published preimage attacks for 45-step SHA-256 and 50-step SHA-512 with complexity of $2^{255.5}$ and $2^{511.5}$, respectively. Regarding pseudo-preimage attacks, 52-step SHA-256 and 57-step SHA-512 can be reached with complexity of 2^{255} and 2^{511} , respectively. Evaluating how those results for SHA-256 and SHA-512 apply to SHA-224, SHA-512/224, and SHA-512/256, is not a trivial task. So it is part of further research how the attacks by Khovratovich et al. [13] can be carried over to SHA-224, SHA-512/224, and SHA-512/256. However, it is unlikely that preimages for an equal number of steps as for SHA-512 can be achieved for SHA-512/224 or SHA-512/256 due to the large internal state of these constructions.

Collision attacks. For creating collisions, two attack strategies have turned out to be fruitful: the meet-in-the-middle attack, and attacks based on heuristic differential cryptanalysis. However, the results obtained by these two attacks differ drastically. So far, meet-in-the-middle attacks achieve free-start collisions with complexities close to the birthday bound. In contrast, approaches based on differential cryptanalysis have been able to create semi-free-start collisions and collisions for (step-reduced) hash functions with practical complexity.

In 2012, Li et al. [15] showed how to create free-start collisions out of meet-in-the-middle attacks. By using this approach, Li et al. developed free-start collision attacks for 40-step SHA-224, 52-step SHA-256, and 57-step SHA-512 with complexity of 2^{110} , $2^{127.5}$, and $2^{255.5}$.

With the help of differential cryptanalysis, Sanadhya and Sarkar [24] as well as Indestege et al. [9] gave practical collisions for 24-step SHA-256 and SHA-512. Subsequently, their results were partially improved by Mendel et al. [16, 17] for SHA-256 leading to a collision attack for 31-step SHA-256 with complexity $2^{65.5}$. Furthermore, they have been able to create a practical collision for 28-step SHA-256 and a 38-step semi-free-start collision. Regarding SHA-512, Eichlseder et al. [8] present a method to create practical 38-step semi-free-start collisions. All these results also trivially apply to truncated variants of the SHA-2 family. In this work, we extend these results to more steps. We have been able to create practical collisions for 27 steps of SHA-512/224 and SHA-512/256, as well as practical semi-free-start collisions for 39-step SHA-512 and all truncated variants. By hiding differences in the truncated words of the final output, the strategy to create practical semi-free-start collisions for 39-step SHA-512 can be applied to create practical free-start collisions for 43-step SHA-512/256 and 44-step SHA-512/224. A similar strategy applies to SHA-224 and hence, we are able to present free-start collisions for 39 steps of SHA-224 with practical complexity.

Practical distinguishers. Non-random properties for the SHA-2 family with practical complexity were first demonstrated by Indestege et al. [9] for up to 31 steps. Later Yu and Wang shown non-randomness for SHA-256 reduced to 39 steps [29] with complexity $2^{184.5}$ and provided a practical example for 33 steps. Biryukov et al. [3] gave a second-order differential collision for the compression function of 47-step SHA-256 obtained via a boomerang attack. Their example quartet is also valid for SHA-224. The generic complexity for this attack would be $> 2^{74}$ for SHA-224 (and $> 2^{85}$ for SHA-256). Yu and Bai [28] recently presented a similar boomerang result for the 48-step SHA-512 compression function. Their example quartet also applies for SHA-512/224 and SHA-512/256.

Contents

1	The SHA-2 Family	6
1.1	SHA-512	6
1.2	SHA-384, SHA-512/256, and SHA-512/224	8
1.3	SHA-256 and SHA-224	8
2	Security Goals	9
2.1	Basic security requirements	9
2.2	Other security requirements	10
3	Generic Attacks	11
3.1	Length extension property	11
3.2	Multicollision attack	12
3.3	Herding and the Nostradamus attack	12
3.4	Second preimages for long messages	13
3.5	Conclusion	13
4	Preimage Attacks	14
4.1	Meet-in-the-Middle Attacks	14
4.2	Pseudo-preimages	15
4.3	Preimages	16
4.4	Conclusion	17
5	Collision Attacks	18
5.1	Published results	19
5.2	General strategy for practical collision search	20
5.3	Practical attacks for truncated SHA-512 variants	24
5.4	Conclusion	29
6	Summary and Discussion	30
A	Practical examples	35
A.1	Preimage examples	35
A.2	Collision examples	36
A.3	Collision characteristics	39

Chapter 1

The SHA-2 Family

The SHA-2 family of hash functions is specified by NIST as part of the Secure Hash Standard (SHS) [22]. The standard defines two main algorithms, SHA-256 and SHA-512, with truncated variants SHA-224 (based on SHA-256) and SHA-512/224, SHA-512/256, and SHA-384 (based on SHA-512). In addition, NIST defines a general truncation procedure for arbitrary output lengths up to 512 bits. Below, we first describe SHA-512, followed by its truncated variants SHA-512/224, SHA-512/256 and SHA-384. Then, we describe the variants SHA-256 and SHA-224 with smaller state sizes.

1.1 SHA-512

SHA-512 is an iterated hash function that pads and processes the input message using t 1024-bit message blocks m_j . The 512-bit hash value is computed using the compression function f in a Merkle-Damgård construction, as in Figure 1.1. The hash output is the final 512-bit chaining value h_t .

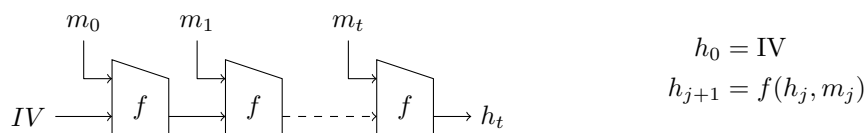


Figure 1.1: Merkle-Damgård construction of the SHA-2 family

In the following, we briefly describe the compression function f of SHA-512. It basically consists of two parts: the message expansion and the state update transformation. A more detailed description of SHA-512 is given by NIST [22].

Padding and message expansion

The message expansion of SHA-512 splits each 1024-bit message block into 16 64-bit words M_i , $i = 0, \dots, 15$, and expands these into 80 expanded message words W_i as

follows:

$$W_i = \begin{cases} M_i & 0 \leq i < 16, \\ \sigma_1(W_{i-2}) + W_{i-7} + \sigma_0(W_{i-15}) + W_{i-16} & 16 \leq i < 80. \end{cases} \quad (1.1)$$

The functions $\sigma_0(x)$ and $\sigma_1(x)$ are given by

$$\begin{aligned} \sigma_0(x) &= (x \ggg 1) \oplus (x \ggg 8) \oplus (x \gg 7), \\ \sigma_1(x) &= (x \ggg 19) \oplus (x \ggg 61) \oplus (x \gg 6). \end{aligned}$$

State update transformation

We use the alternative description of the SHA-512 state update by Mendel et al. [16], which is illustrated in Figure 1.2.

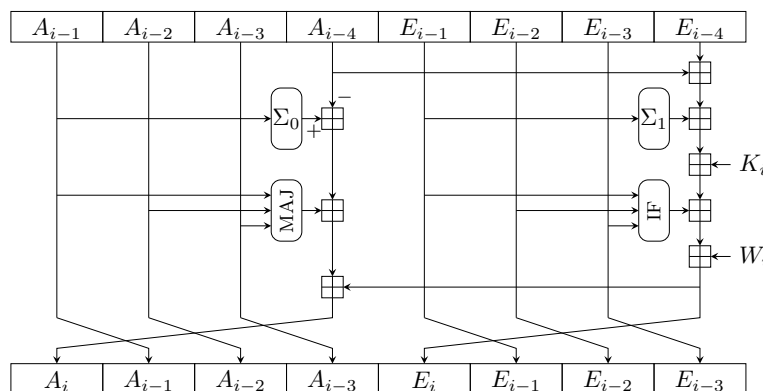


Figure 1.2: The state update transformation of SHA-512.

The state update transformation starts from the previous 512-bit chaining value $h_j = (A_{-1}, \dots, A_{-4}, E_{-1}, \dots, E_{-4})$ and updates it by applying the step functions 80 times. In each step $i = 0, \dots, 79$, one 64-bit expanded message word W_i is used to compute the two state variables E_i and A_i as follows:

$$E_i = A_{i-4} + E_{i-4} + \Sigma_1(E_{i-1}) + \text{IF}(E_{i-1}, E_{i-2}, E_{i-3}) + K_i + W_i, \quad (1.2)$$

$$A_i = E_i - A_{i-4} + \Sigma_0(A_{i-1}) + \text{MAJ}(A_{i-1}, A_{i-2}, A_{i-3}). \quad (1.3)$$

The step constants K_i are chosen as nothing-up-my-sleeve numbers (fractional parts of the cube roots of the first prime numbers); for details, we refer to the standard document [22]. The bitwise Boolean functions IF and MAJ used in each step are defined by

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

and the linear functions Σ_0 and Σ_1 are defined as follows:

$$\begin{aligned}\Sigma_0(x) &= (x \ggg 28) \oplus (x \ggg 34) \oplus (x \ggg 39), \\ \Sigma_1(x) &= (x \ggg 14) \oplus (x \ggg 18) \oplus (x \ggg 41).\end{aligned}$$

After the last step of the state update transformation, the previous chaining value is added to the output of the state update (Davies-Meyer construction). The result of this feed-forward sum is the chaining value h_{j+1} for the next message block m_{j+1} (or the final hash value h_t):

$$h_{j+1} = (A_{79} + A_{-1}, \dots, A_{76} + A_{-4}, E_{79} + E_{-1}, \dots, E_{76} + E_{-4}). \quad (1.4)$$

1.2 SHA-384, SHA-512/256, and SHA-512/224

These truncated variants of SHA-512 differ only in their initial values and a final truncation to 224, 256, or 384 bits, respectively. The rest of the algorithmic description remains exactly the same. The message digest of SHA-512/256 is obtained by omitting the output words $E_{79} + E_{-1}$, $E_{78} + E_{-2}$, $E_{77} + E_{-3}$, and $E_{76} + E_{-4}$ of the last compression function call. SHA-512/224 additionally omits the 32 least significant bits of $A_{76} + A_{-4}$. SHA-384 omits the two words $E_{77} + E_{-3}$ and $E_{76} + E_{-4}$.

1.3 SHA-256 and SHA-224

SHA-256 and SHA-512 are closely related. Thus, we only point out properties of SHA-256 which differ from SHA-512:

- The wordsize is 32 instead of 64 bits, so SHA-256 processes 512-bit message blocks.
- IV and K_i are the 32 most significant bits of the respective SHA-512 value.
- The step function is applied 64 instead of 80 times, so only the first 64 expanded message words are used.
- The linear functions $\sigma_0, \sigma_1, \Sigma_0$ and Σ_1 use different rotation values: The functions $\sigma_0(x)$ and $\sigma_1(x)$ for SHA-256 are given by

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

and the linear functions Σ_0 and Σ_1 are defined as follows:

$$\begin{aligned}\Sigma_0(x) &= (x \ggg 2) \oplus (x \ggg 13) \oplus (x \ggg 22) \\ \Sigma_1(x) &= (x \ggg 6) \oplus (x \ggg 11) \oplus (x \ggg 25).\end{aligned}$$

SHA-224 is a truncated variant of SHA-256 with different IV , in which the output word $E_{60} + E_{-4}$ is omitted.

Chapter 2

Security Goals

In this section, we define the security goals that hash functions should fulfill. First, we define the three classical security notions of collision resistance, preimage resistance, and second preimage resistance. Then, we discuss attacks on hash functions in weaker scenarios such as free-start collisions, semi-free-start collisions, near-collisions and pseudo-preimages.

2.1 Basic security requirements

Since cryptographic hash functions are used in many applications with different requirements, it is difficult to state all the properties that are expected from a hash function. However, we list here the most common requirements. Informally, a cryptographic hash function H has to fulfill the following three basic security requirements.

- *Collision resistance*: it should be hard to find two messages M and M^* , with $M^* \neq M$, such that $H(M) = H(M^*)$.
- *Second preimage resistance*: for a given message M , it should be hard to find a second message $M^* \neq M$ such that $H(M) = H(M^*)$.
- *Preimage resistance*: for a given hash value h , it should be hard to find a message M such that $H(M) = h$.

The resistance of a hash function to collision and (second) preimage attacks depends in the first place on the bitlength n of the hash value h . Regardless of how a hash function is designed, an adversary will always be able to find preimages or second preimages after trying out about 2^n different messages. Finding collisions requires a much smaller number of trials: about $2^{n/2}$ due to the birthday paradox. A function is said to achieve *ideal security* if these bounds are guaranteed. If the internal structure of the hash function allows to construct collisions or (second) preimage faster than expected based on its hash size, then the hash function is considered to be broken. For a formal treatment of these three security properties of cryptographic hash functions we refer to [23, 25].

2.2 Other security requirements

Along with the well known security notions of collision resistance and (second) preimage resistance, it is often required that a cryptographic hash function should fulfill several other properties. For instance in [21], NIST requires for SHA-3 candidates that any r -bit hash function specified by selecting a fixed r -bit subset of the n output bits should adhere to the same security requirements as the original function. From a practical application point of view, this requirement makes a lot of sense when we want to guarantee security in cases where the output space of the hash function is reduced by means of a simple truncation. In particular, this results in the security requirements of partial preimage and near-collision resistance.

- *Partial preimage resistance:* Given r bits of a hash value h , it should be hard to find a message M with $h^* = H(M)$ such that h^* matches h in the specified r bits.
- *Near-collision resistance:* It should be hard to find any two messages M, M^* such that $H(M)$ and $H(M^*)$ differ in only a small number of bits.

Ideally, a cryptographic hash function should behave like a random oracle. Hash functions following the Merkle-Damgård construction (like SHA-256 and SHA-512) are vulnerable to some generic attacks (see Chapter 3) and thus clearly do not behave like a random oracle. However, Coron et al. [4] have shown that the hash function obtained by truncating a non-trivial number of bits from the output of a Merkle-Damgård construction is indistinguishable from a random oracle.

In addition to considering the complexities of finding collisions and (second) preimages, it is common to examine the feasibility of attacks on slightly modified versions of the hash function. One approach is to investigate the difficulty to find collisions or preimages when the initial value h_0 can be freely chosen or changed, since this gives a good view on the security of the hash function. In [14], Lai and Massey introduce the notion of semi-free-start collisions and free-start collisions.

- *Semi-free-start collision attack:* Find two messages M and M^* as well as an initial value h_0 , with $M^* \neq M$, such that $H(h_0, M) = H(h_0, M^*)$.
- *Free-start collision attack:* Find two messages M and M^* as well as two initial values h_0 and h_0^* , with $(M^*, h_0^*) \neq (M, h_0)$, such that $H(h_0, M) = H(h_0^*, M^*)$.

Note that the notion of free-start collisions can be adopted to (second) preimages as well, leading to the following definition of pseudo-preimage attacks.

- *Pseudo-preimage attack:* Given a hash value h , find a message M and an initial value h_0 , such that $h = H(h_0, M)$.

Since the initial value is an integral part of the hash function specification, free-start collisions and pseudo-preimages are of limited practical interest. However, they are still considered as certification weaknesses and cast suspicion about the security of the hash function.

Chapter 3

Generic Attacks

In this section, we discuss previously published generic attacks on iterated hash functions based on the Merkle-Damgård construction [5, 19], such as SHA-256 and SHA-512. This includes length extension, multicollisions by Joux [10], the herding and Nostradamus attacks by Kelsey and Kohno [11], and second preimages for long messages by Kelsey and Schneier [12]. As already observed in NIST Special Publication 800-107 [6], the attack of Kelsey and Schneier affects the classical security notion of second preimage resistance. The complexity to create second preimages depends on the actual length of the message. We summarize the mentioned attacks and discuss their impact on SHA-224, SHA-512/224, and SHA-512/256, which truncate the final output of the Merkle-Damgård construction.

3.1 Length extension property

The length extension property is a well-known weakness of iterated hash functions following the Merkle-Damgård design principle, already mentioned by Damgård and Merkle in [5, 19]. Assume we are given two messages M and M^* of the same length that result in a collision. Then it is easy to construct many suffixes S such that $M\|S$ and $M^*\|S$ also collide. Hence, an arbitrary number of collisions can be constructed with negligible extra effort, once a single collision has been found. Another related weakness is the fact that given $H(M)$ and the length of the message, but not M itself, one can compute $H(M\|S)$ for any suffix S using the same property as above.

Depending on the actual use of the hash function, this behavior can be a serious threat. For instance, hash functions vulnerable to the length extension property cannot be used to construct a secure message authentication code (MAC) by prepending the key as a secret message prefix, $\text{MAC}_K(x) = H(K\|x)$. Obviously, the length extension property would result in trivial forgery attacks on the MAC.

For SHA-512/224 and SHA-512/256, at least half of the bits of the internal state are discarded to produce the final hash value. The complexity of recovering the unknown bits is too high to lead to a meaningful attack. For this reason, length extension attacks pose no threat for SHA-512/224 and SHA-512/256.

In the case of SHA-224, however, only 32 bits of the 256-bit internal state are truncated to obtain the final hash digest. Consequently, each hash collision is a collision of the compression function output (and thus eligible for length extensions) with a probability of 2^{-32} . Similarly, given $H(M)$ but not M , there are only 2^{32} candidates for $H(M||S)$ for any suffix S . This is clearly a non-random property of the hash function.

3.2 Multicollision attack

The multicollision attack is another generic attack on iterated hash functions based on the Merkle-Damgård construction. Ideally, finding r different messages that hash to the same n -bit hash value has a complexity of about $2^{n \cdot (r-1)/r}$. However, Joux showed in [10] that finding multicollisions for iterated hash functions has roughly the same complexity as finding a single collision. More specifically, finding an r -collision has a complexity of about $\log_2 r \cdot 2^{n/2}$ compression function evaluations. In the case of SHA-256 and SHA-512, an r -collision can be constructed with a complexity of $\log_2 r \cdot 2^{128}$ and $\log_2 r \cdot 2^{256}$, respectively.

The attack relies on the fact that internal collision can be created relatively cheaply. Since the internal state for SHA-512/224 and SHA-512/256 is $\geq 2 \cdot n$, the complexity of creating an internal collision is $\geq 2^n$. In the case of SHA-512/256, the cost for creating a r -multicollision in this way is $\log_2 r \cdot 2^n$. If we just hash messages and search for a r -multicollision at the hash value directly, we have a complexity of about $2^{n \cdot (r-1)/r}$, which is not worse than 2^n . Thus, Joux' attack does not bring any advantage.

For SHA-224, on the other hand, the attack allows to construct r -collisions with a complexity of $\log_2 r \cdot 2^{128}$. This is faster than the generic $2^{224 \cdot (r-1)/r}$ for $r \geq 3$.

3.3 Herding and the Nostradamus attack

Kelsey and Kohno [11] introduced an interesting property for hash functions called chosen-target forced-prefix (CTFP) preimage resistance. This basically states that it should be hard to find many partial messages (suffixes) that result in the same hash value and afterwards, find a chosen prefix that can be connected to one of these suffixes. Kelsey and Kohno demonstrated that the complexity for creating a CTFP is significantly below 2^n for hash functions based on the Merkle-Damgård construction.

Their so-called herding attack uses the fact that one can construct multicollisions starting from different chaining values with a relatively low complexity. This is used to build a tree-like structure, the “diamond structure”, where blockwise internal collisions are constructed until a common chaining value is reached. The complexity to create such a structure is roughly $2^{k/2+n/2+2}$, where k is the width of the diamond structure. The complexity to link a chosen prefix to this structure is 2^{n-k} .

This technique can be used in the so-called Nostradamus attack scenario: Nostradamus predicts some future event, which he does not reveal yet. Instead, he gives the hash value h of his prediction as commitment. After the event takes place, he can

choose the prefix of the message almost freely by using the technique describe above, instead of constructing a preimage.

The complexity of the attack is given by $2^{n-k} + 2^{k/2+n/2+2}$. The choice of $k = 84$ and $k = 169$ for SHA-256 and SHA-512 results in an attack complexity of 2^{172} and 2^{343} , respectively. Again, like for multicollisions, the necessity of finding internal collisions means that the herding attack provides no advantage for SHA-512/224 and SHA-512/256. For SHA-224, the attack complexity of 2^{172} is the same as for SHA-256, which is still faster than a brute force search for preimages.

3.4 Second preimages for long messages

When considering the Merkle-Damgård construction without length padding, it is easy to see that long messages increase the number of intermediate chaining values to hit when searching for a second preimage. Thus, for the Merkle-Damgård construction without length padding, a second preimage for a 2^k block message can be constructed with a complexity of 2^{n-k} . Since it is not trivial to control the length of the second preimage, this simple form of the attack is prohibited by the length padding of the hash function.

However, Kelsey and Schneier [12] proposed a way to produce second preimages for long messages by using the concept of expandable messages based on Joux' multicollisions. They show how to construct second preimages for long messages with 2^k blocks with a complexity of about $k \cdot 2^{n/2+1} + 2^{n-k+1}$.

For SHA-224, this is faster than brute force for messages of at least 2^{32} blocks, and reduces second preimage resistance to 201–224 bits, depending on the message length. For SHA-512/224 and SHA-512/256, the method provides no advantage for the attacker.

3.5 Conclusion

For SHA-512/224 and SHA-512/256, the size of the internal state (chaining variables) is at least twice the size of the hash value. This has a significant impact on the applicability of the investigated generic attacks on iterated hash functions. The truncation effectively precludes length extensions, multicollisions, herding attacks, and second preimages for long messages. None of the attacks provides an advantage over generic brute force methods.

In the case of SHA-224, the internal state is only slightly larger than the hash value itself. This leads to the fact that the investigated attacks are still applicable and show non-random properties of SHA-224. However, the gain over brute force attacks is not as big as for standard Merkle-Damgård constructions like SHA-256. More specifically, length extensions are possible for SHA-224 (with an extra effort of 2^{32}). Joux' multicollision attack finds r -multicollisions faster than the generic method for $r \geq 3$. The herding attack for finding chosen-target forced-prefix preimages has a complexity of about 2^{172} (compared to 2^{224} for straightforward preimages). Second preimages can be found faster than 2^{224} for messages of $> 2^{32}$ blocks, which reduces second preimage resistance to 2^{201} for the maximum allowed message length of 2^{64} bits (or 2^{55} message blocks).

Chapter 4

Preimage Attacks

This chapter discusses preimage attacks on the SHA-2 family. Over the past years, the following strategy has turned out to be fruitful:

1. Create one-block pseudo-preimages using a meet-in-the-middle attack strategy.
2. Convert one of the one-block pseudo-preimages into a two-block preimage using an unbalanced meet-in-the-middle attack.

We first give an overview about the general concepts and ideas of performing a meet-in-the-middle attack on a compression function. Afterwards, we discuss concrete attack techniques to create pseudo-preimages for SHA-2. At last, we show how to convert pseudo-preimages into preimages. A summary of published pseudo-preimage and preimage attacks covering the most rounds based on this strategy is given in Table 4.1.

Table 4.1: Summary of the best published preimage and pseudo-preimage attacks.

Attack	Target	Steps	Complexity	Reference
preimage	SHA-512	50	$2^{511.5}$	[13]
	SHA-256	45	$2^{255.5}$	[13]
pseudo-preimage	SHA-512	57	2^{511}	[13]
	SHA-384	43	2^{366}	[1]
	SHA-256	52	2^{255}	[13]
	SHA-224	43	2^{220}	[1]

4.1 Meet-in-the-Middle Attacks

In this section, we give the basic idea of a meet-in-the-middle attack to create preimages under idealized assumptions. We want to find a one-block preimage of the hash value h . That is, we search for suitable message block m_0 that fulfills $f(\text{IV}, m_0) = h$ for the compression function f . By randomly trying various messages, the complexity to find a solution is 2^n .

The compression function f of the SHA-2 family with its final feed-forward addition can be seen as a Davies-Meyer construction with a dedicated block cipher E , that is, $f(h_j, m_j) = E(h_j, m_j) + h_j$. Assume now the block cipher $E(h_j, m_j)$ could be split into functions (chunks) f_a and f_b , where f_a only depends on one part of the message m_a and f_b only depends on the other part of the message m_b and the output of f_a , that is, $m_j = (m_a, m_b)$ and $E(h_j, m_j) = f_b(f_a(h_j, m_a), m_b)$ as in Figure 4.1. Then, the search for a preimage can be solved by the search for a collision (match) at the central matching point (the output of f_a). This can be done by trying random message parts m_a and m_b , and calculating $x_a = f_a(\text{IV}, m_a)$ and $x_b = f_b^{-1}(h - \text{IV}, m_b)$ until a match $x_a = x_b$ is found. If m_a and m_b are large enough, the complexity of this meet-in-the-middle attack is about $2^{w/2}$ in general, where w is the size of the internal state.

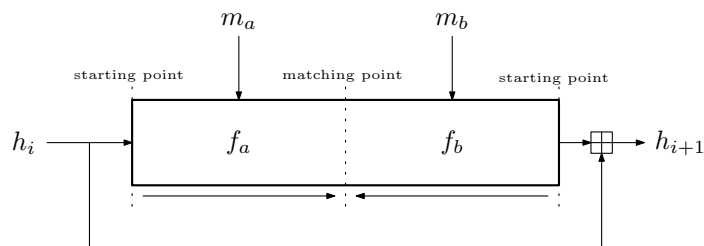


Figure 4.1: Concept of a meet-in-the-middle preimage attack.

4.2 Pseudo-preimages

For a higher number of steps of SHA-2, such a simple separation of the message and block cipher into two parts becomes impossible. To attack a higher number of steps, more sophisticated techniques are necessary. The most successful attacks of Aoki et al. [1] and Khovratovich et al. [13] are based on the splice-and-cut technique [2], which produces pseudo-preimages instead of preimages. We give a high-level overview of the attack techniques below and refer to [1, 13] for details.

In the pseudo-preimage attacks on SHA-2 [1, 13], a clean separation in two functions each only depending on independent message parts is not possible due to the high number of steps attacked. However, as indicated in Figure 4.2, it is still possible to split the cipher in two chunks. Although both chunks depend on a large, common part of the message, there may be some message words (called neutral words) or bits (neutral bits) which influence only one chunk.

An additional technique is the use of an initial structure rather than a simple starting point, as illustrated in Figure 4.2. This allows to use some words as neutral words that would otherwise influence both chunks. For SHA-2, the initial structure can be based on the properties of the nonlinear function IF, which allows to move some of the input words W_i around between 4 consecutive steps [1]. In a similar vein, Khovratovich et al. [13] use the concept of bicliques, which resembles the idea of the initial structure.

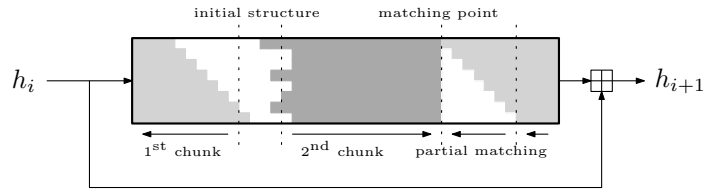


Figure 4.2: Concept of a pseudo-preimage attack on SHA-2 (see [1]).

Moreover, SHA-2 has an unbalanced Feistel structure. Thus, message words W_i do not influence the whole state at the time of their injection, and parts of the state are unaffected over several steps. This behavior can be exploited with the partial matching technique, such that at the matching point, no match for the complete state is required, but just for a fraction of the state words.

As indicated in Figure 4.2, the starting point of the attack is somewhere in the middle and not at the endings. Therefore, the actual value of h_i is determined during the attack and cannot be predefined. So only a pseudo-preimage of h is created. The reasons to do so are manifold, for instance it enables the freedom to use the best suitable position for the initial structure. For chop-MD constructions, it is helpful, or even necessary (if $w \geq n/2$) to put the matching point at (or close to) the end (the feed-forward). This way, only the shorter hash value has to be matched and not the whole internal state.

Aoki et al. [1] published pseudo-preimage attacks on 43-step SHA-224 and 43-step SHA-256, with a complexity of $2^{219.9}$ and $2^{251.9}$. The result for SHA-256 has been improved by Khovratovich et al. [13], finally reaching 52 steps with a complexity of 2^{255} .

The best published pseudo-preimage result for SHA-384 is a 43-step attack with a complexity of 2^{366} by Aoki et al. [1], and for SHA-512 a 57-step attack with a complexity of 2^{511} by Khovratovich et al. [13]. So far, no results exist regarding pseudo-preimage attacks on the other truncated SHA-512 variants, SHA-512/224 and SHA-512/256. However, we estimate that pseudo-preimages covering a similar number of steps as for SHA-384 are feasible for these variants, with comparable gains over the generic complexities (since the number of output words to match is reduced accordingly).

4.3 Preimages

Under certain conditions, the pseudo-preimages created with the techniques in the previous section can be converted into preimages. This is done by using a two-block approach and performing an unbalanced meet-in-the-middle attack on the chaining value h_1 , as illustrated in Figure 4.3. This approach is possible if pseudo-preimages can be created with a complexity of 2^y , where $y < w - 2$, with w the size of the chaining value. The resulting complexity of the attack is then $2^{\frac{w+y}{2}+1}$ [18, Fact 9.99]. Thus, this attack gives no advantage over generic preimage attacks for chop-MD constructions with $w \geq 2n$.

The preimage attack covering the most steps of SHA-256 converts a 45-step pseudo-preimage and has a complexity of $2^{255.5}$. Although such preimage attacks are conceptually possible on step-reduced variants of SHA-224, no such attack is known to us.

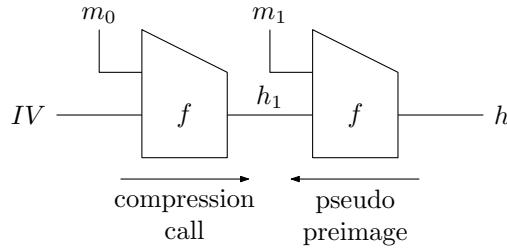


Figure 4.3: Converting a pseudo-preimage into a preimage.

Khovratovich et al. [13] published a preimage attack on 50 steps of SHA-512 with a complexity of $2^{511.5}$. For SHA-384, the situation is similar to SHA-224: The attack seems conceptually possible, but no results are known to us. SHA-512/224 and SHA-512/256 are truncated variants of SHA-512 with $w \geq 2n$. As observed in § 4.2, the matching point of a meet-in-the-middle attack has to be at (or close to) the end of the compression function. So, only pseudo-preimages can be created by using this approach. However, these pseudo-preimages cannot be converted to preimages with the above approach. Thus, the straight-forward application of the preimage attacks presented in this chapter is no threat to SHA-512/224 and SHA-512/256, and other ways have to be explored to create preimages.

For this reason, we considered practical attacks based on equation solving. Our approach is based on SAT solving, an idea previously applied successfully to MD5 [20] and other hash functions. By fixing some of the message words and leaving enough free variables for matching, we searched for preimages of the all-zero output hash value in the step-reduced hash function. We were able to find preimages for up to 18 steps of all SHA-2 variants in a relatively short time (including correct padding), but found no results for 19 or more steps. Example preimages are given in Appendix A.1 in Table A.1. To the best of our knowledge, these are the only published practical results on SHA-2 preimages. Clearly, the gap between theoretical and practical attacks is significantly larger for preimages than for collisions.

4.4 Conclusion

The only published result on pseudo-preimage attacks on the truncated variants SHA-224, SHA-512/224 or SHA-512/256 is due to Aoki et al. [1]. They present a 43-step pseudo-preimage attack for SHA-224 with a complexity of 2^{220} . We estimate that the same strategy can be applied to SHA-512/224 and SHA-512/256 (similar to Aoki et al.’s results on SHA-384), with a comparable advantage over generic attacks.

There are no results on preimages, and the generic method of turning pseudo-preimages into preimages is not applicable for SHA-512/224 and SHA-512/256. In our practical experiments with equation solving, we were able to construct preimages for up to 18 steps of SHA-224, SHA-512/224, and SHA-512/256.

Chapter 5

Collision Attacks

For creating collisions, two attack strategies have turned out to be fruitful: the meet-in-the-middle attack, and attacks based on heuristic differential cryptanalysis. However, the results obtained by these two attacks differ drastically. So far, meet-in-the-middle attacks achieve free-start collisions with complexities close to the birthday bound. In contrast, approaches based on differential cryptanalysis have been able to create semi-free-start collisions and collisions for (step-reduced) hash functions with practical complexity, but for fewer steps.

Below, we first describe the theoretic results based on pseudo-preimages, as well as a number of results on practical collisions, semi-free-start collisions, and non-randomness properties based on boomerang attacks in § 5.1. A summary of the best previously published collision attacks is given in Table 5.1. Afterwards, we present new practical collision, semi-free-start collision, and free-start collision results for all investigated SHA-2 variants. We first describe the general approach and search strategy in § 5.2, followed by the application to the individual hash functions in § 5.3. An updated table including the new results of this work is in Table 6.1.

Table 5.1: Summary of best published collision attacks.

Attack	Target	Steps	Complexity	Reference
collision	all	24	practical	[9, 24]
	SHA-256	28 31	practical $2^{65.5}$	[17]
	SHA-224	28 31	practical $2^{65.5}$	[17]
semi-free-start collision	all	38	practical	[8, 17]
free-start collision	SHA-512	57	$2^{255.5}$	[15]
	SHA-384	40	2^{183}	
	SHA-256	52	$2^{127.5}$	
	SHA-224	40	2^{110}	

5.1 Published results

5.1.1 Free-start collisions from preimage attacks

In this section, we recapitulate the work of Li et al. [15] to show how to convert pseudo-preimage attacks into free-start collision attacks. First, we have to turn the pseudo-preimage attacks of § 4.2 into r -bit partial preimage attacks on the compression function. As illustrated in Figure 5.1, this can be done by moving the matching point of a meet-in-the-middle attack to the end of the compression function.

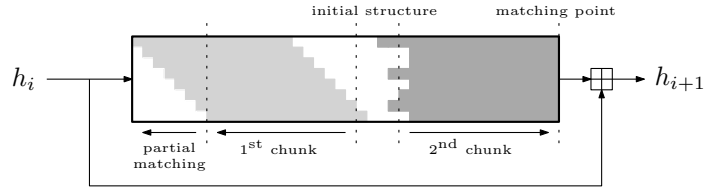


Figure 5.1: Concept of a r -bit partial pseudo-preimage attack (inspired by [1]).

Now we assume that we have an attack to create a r -bit partial pseudo-preimage where all other $n - r$ bit of the chaining value h_{i+1} are random. If we repeatedly create r -bit partial pseudo-preimages of the same r -bit value, the other $n - r$ bit match with birthday complexity. Thus, after creating about $2^{(n-r)/2}$ r -bit partial pseudo-preimages, we expect to have a free-start-collision. To be more efficient than a generic collision attack, the complexity for creating a r -bit partial pseudo-preimage must be significantly smaller than $2^{r/2}$.

By using this approach, Li et al. showed free-start collision attacks for 40-step SHA-224, 52-step SHA-256, and 57-step SHA-512 with complexity of 2^{110} , $2^{127.5}$, and $2^{255.5}$ (with correct padding) [15]. Note that attacks on up to 43-step SHA-224 might be possible when ignoring the padding. The application of this attack on SHA-512/244 and SHA-512/256 correlates with the prospects to create pseudo-preimages for these truncated variants. Hence, the statements made in Chapter 4 apply.

5.1.2 Practical collision attacks

With the help of differential cryptanalysis, Sanadhya and Sarkar [24] and Indesteege et al. [9] gave practical collisions for 24-step SHA-256 and SHA-512. Subsequently, their results were partially improved by Mendel et al. [16, 17] for SHA-256, leading to a collision attack for 31-step SHA-256 with complexity $2^{65.5}$. Furthermore, they have been able to create a practical collision for 28-step SHA-256 and a 38-step semi-free-start collision. Regarding SHA-512, Eichseder et al. [8] present a method to create practical 38-step semi-free-start collisions. All these results also trivially apply to truncated variants of the SHA-2 family. A summary of the best previously published collision attacks is given in Table 5.1.

5.1.3 Practical distinguishers

Non-random properties for the SHA-2 family with practical complexity were first demonstrated by Indestege et al. [9] for up to 31 steps. Later Yu and Wang showed non-randomness for SHA-256 reduced to 39 steps [29] with complexity $2^{184.5}$ and provided a practical example for 33 steps. Biryukov et al. [3] gave a second-order differential collision for the compression function of 47-step SHA-256 obtained via a boomerang attack. Their example quartet is also valid for SHA-224. The generic complexity for this attack would be $> 2^{74}$ for SHA-224 (and $> 2^{85}$ for SHA-256). Yu and Bai [28] recently presented a similar boomerang result for the 48-step SHA-512 compression function. Their example quartet also applies for SHA-512/224 and SHA-512/256.

5.2 General strategy for practical collision search

Starting from the ground-breaking results of Wang et al. [26, 27], the search techniques used for practical collisions have been significantly improved, hitting their current peak in the attacks on SHA-256 [3, 17] and SHA-512 [8, 28]. In spite of all achieved improvements, the top-level attack strategy has remained essentially the same. At first, a suitable starting point for the search must be determined to define the search space and hopefully make the ensuing search process feasible. The search itself usually involves two phases: The search for a suitable differential characteristic, and the message modification phase to determine a collision-producing message pair for this characteristic. The search for this characteristic and message pair can either be done by hand or, for more complex functions like SHA-2, using an automatic search tool. We use a heuristic search tool based on a guess-and-determine strategy, which we briefly describe in § 5.2.1. Afterwards, we discuss the choice of suitable starting points in § 5.2.2.

5.2.1 Guess-and-determine search tool

To search for differential characteristics and colliding message pairs, we use an automatic search tool, which implements a configurable heuristic guess-and-determine search strategy. Roughly, the tool is partitioned in two separate, but closely interacting parts: The representation of the analyzed cryptographic primitive and the search procedure.

Representation

The tool internally represents differences at bit level, allowing to store all possible stages from a completely unrestricted bit over signed differences down to exact values. Thus, the same tool can be used in the search for a characteristic and in the search for a message pair. The conditions are grouped in words representing the internal variables of the cipher. These words can then be connected with any operations (typically bitwise functions or modular additions) to define the cipher.

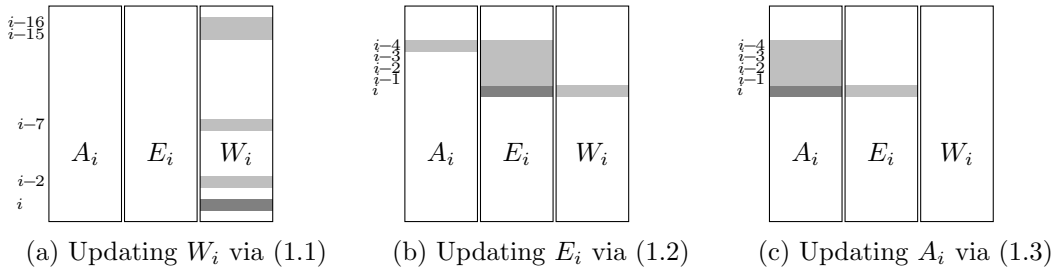


Figure 5.2: Update rules to compute $A_i, E_i,$ and W_i (■) from other state words (■).

Search

The search procedure uses the bitwise conditions as variables, and attempts to find a solving assignment with the help of a heuristic guess-and-determine strategy [7], similar to SAT solvers. The following steps are repeated until a solution is found:

- **Guess:** Pick a bit and guess its value (e.g., no difference, or a specific assignment).
- **Determine:** The previous guess influences other connected bit conditions. Determine these effects, which might result in further refinement of other bit conditions, or a contradiction.
- **Backtrack:** If a contradiction is detected, resolve this conflict by undoing previous guesses and replacing them with other choices.

This simple approach alone is not sufficient to go through the whole search space, so numerous refinements have been proposed to fine-tune this method. These include the detection of two-bit conditions [16], backtracking strategies, and a look-ahead approach to guide the search [8]. Additionally, SHA-2-specific heuristics and strategies [16, 17] have been proposed, deciding which parts of the state to guess with higher priority.

5.2.2 Finding starting points for SHA-2

To model SHA-2 as a satisfiability problem for the search tool, we need to introduce suitable intermediate variables. Based on the alternative description from § 1, we only use the words A_i and E_i of the state, plus the words W_i of the message expansion. Figure 5.2 illustrates the update rules for A, E and W by highlighting the input words for updating each word: Each row represents one of the 80 step iterations, with its three state words $A_i, E_i,$ and W_i .

Local collisions

All our results are based on “local collisions” in the message expansion: carefully selecting (expanded) message words in the middle steps so that the differences can cancel out in as many consecutive steps as possible in the forward and backward expansion, i.e., the first and last few expanded message words contain no differences. The t middle steps

with differences can induce differences in the A_i and E_i words. However, the W_i words can be used to achieve zero difference in the last 4 of the t words E_i , and in the last 8 of the t words A_i . This is necessary to obtain words with zero difference in the very last 4 steps of the state update and thus in the output chaining value.

As an example, the starting point for the 27-step collisions for SHA-256 [16] allows differences in expanded message words W_7, W_8, W_{12}, W_{15} , and W_{17} , as well as state words E_7, \dots, E_{13} and A_7, \dots, A_{10} . The exact bitwise signed differences are chosen during the search such that any potential differences in $W_{19}, W_{22}, W_{23}, W_{24}$, as well as E_{14}, \dots, E_{17} and A_{10}, \dots, A_{13} cancel out. The resulting starting point is illustrated in Figure 5.3a. We show in § 5.3.3 how the same starting point can be used for SHA-512. In addition, we use the closely related starting point for 28-step SHA-256 collisions [17], illustrated in Figure 5.3b, to also find collisions for SHA-224.

The semi-free-start collision starting point covering the most steps so far is for 38 steps of SHA-256 [17] and SHA-512 [8], with a local collision spanning $t = 18$ steps, as illustrated in Figure 5.3c. Considering the large number of steps, the number of expanded message words with differences and cancellations is remarkably low: only 6 words with differences, and 6 words imposing cancellation conditions.

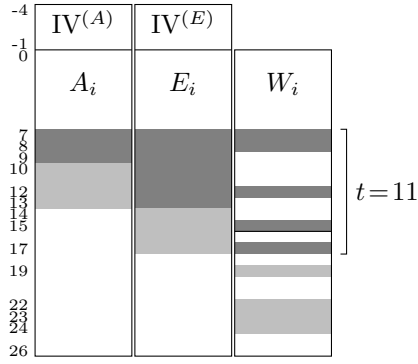
To find candidates for a higher number of steps, we enumerated all possible selections of active message words (more precisely, of some $t \leq 20$ intermediate expanded message words, the “core words” of the local collision) and investigated the forward and backward expansion under certain assumptions: the t core words are chosen freely, according to the message expansion rule; in the forward and backward expansion, if at least 2 of the input words have differences, they are assumed to cancel out, while a single input word with difference never cancels out. Criteria for selecting suitable candidates then include a low number t of spanned steps and a low number of required cancellation constraints. The best (consistent) result for 39 steps, spanning $t = 19$ steps with 9 cancellations, is given in Figure 5.3d.

Semi-free-start collisions and collisions

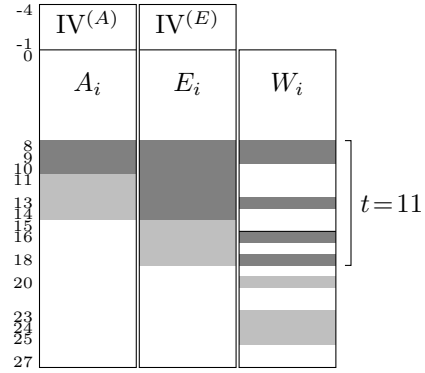
The discussed starting points are targeted to find semi-free-start collisions, that is, different messages m, m' and an IV h_0 such that $f(h_0, m) = f(h_0, m')$. However, they can also be used for hash function collisions with the original IV h_0 by trading the freedom of the IV for freedom in the message words.

In order to find hash function collisions, the first few message words W_i must retain sufficient freedom (i.e., they should not be constrained by conditions from the message expansion for cancelling differences) to allow to match the correct IV value. Ideally, this means that the first 8 message words W_0, \dots, W_7 are free of any conditions (no differences, but also not constrained by conditions from other message words connected via the message expansion). If the W_i differences are sparse enough overall, it can also be sufficient to have at least 5 words W_0, \dots, W_4 free of conditions by providing the remaining freedom with a two-block approach [17].

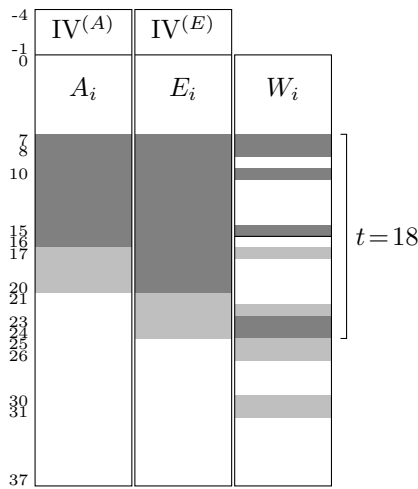
The starting points of Figure 5.3a and Figure 5.3d both have at least 7 message words free of differences in the beginning. However, the local collision shown in Fig-



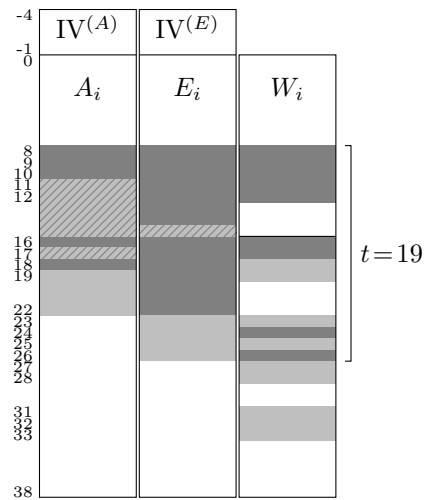
(a) 27-step collision of SHA-256 [16] and SHA-512 (§ 5.3.3).



(b) 28-step collision of SHA-256 [17] and SHA-224 (§ 5.3.3).



(c) 38-step semi-free-start collision of SHA-256 [17] and SHA-512 [8].



(d) 39-step semi-free-start collision of SHA-512 (§ 5.3.1).

Figure 5.3: SHA-2 starting points: Words with differences ■ and cancellations ■, ▨.

ure 5.3d spans over $t = 19$ steps. Thus, the first message words are constrained by many conditions, leaving not enough freedom to match the correct IV. The starting point with $t = 18$ in Figure 5.3c is similarly unsuitable for IV matching. In contrast, the 11-step local collision shown in Figure 5.3a provides enough freedom in the first 7 message words to be used in a single-block collision attack [16].

5.3 Practical attacks for truncated SHA-512 variants

The hash functions SHA-512/224 and SHA-512/256 differ from SHA-512 in their IV and a final processing step, which truncates the 512-bit state to 224 or 256 bits, respectively. Consequently, the semi-free-start collisions demonstrated for SHA-512 [8] are also valid for these truncated versions (since the IV is non-standard anyway in this attack scenario). In this section, we first improve these results by providing 39-step semi-free-start collisions for SHA-512 and its variants. We then extend this result to free-start collisions for 43-step SHA-512/256 and 44-step SHA-512/224. By free-start collisions, we mean two messages m, m' and two IVs h_0, h'_0 such that the hash values of m (under IV h_0) and m' (under IV h'_0) collide. Note that free-start collisions are not equivalent to collisions of the compression function for truncated SHA-2 versions, since the truncated output bits of the last compression function call may contain differences. Additionally, we present collisions for 27 steps of SHA-512, SHA-512/224, and SHA-512/256.

5.3.1 Semi-free-start collisions

We use the 39-step starting point from Figure 5.3d. Previous work showed that sparse differences particularly in the A_i words are essential for the success probability of the message modification phase. For this reason, we additionally require that in 6 words between A_8 and A_{18} , namely $A_{11}, A_{12}, A_{13}, A_{14}, A_{15}$, and A_{17} , differences also cancel out. The five consecutive zero-difference words in A_i also force E_{15} to zero difference. These additional requirements are already marked in Figure 5.3d (hatched area).

The first task for the search procedure with the solving tool is to fix a suitable signed characteristic. Compared to the previously published 38-step SHA-512 semi-free-start collision [8], the local collision for our starting point spans 19 steps (compared to previously 18) and has 9 (previously 6) active expanded message words. Cancellations are also required in 9 (previously 6) expanded message words. This increases the necessity for very sparse differences in A_i and W_i in steps 16–26. For this reason, we require a single-bit difference in W_{26}, W_{17} and A_{18} , and very low Hamming weights for the other words. We finally found a characteristic with at most two active bits in almost all words of A_i and W_i (except $A_9, A_{10}, W_{11}, W_{12}$), given in Appendix A in Table A.9.

After the characteristic is fixed, we need to find a complying message pair. We start by guessing the dense parts in A_i and E_i , hoping that the sparser conditions in the later steps are fulfilled probabilistically. Since the dense parts are already almost fully determined by the characteristics and the sparse parts pose only so few conditions, a

message pair is easily found. The result is a semi-free-start collision valid for all SHA-512 variants.

5.3.2 Free-start collisions

Free-start collisions are a generalization of semi-free-start collisions, so the 39-step results obtained in the previous section give a first result for SHA-512/224 and SHA-512/256. However, we can take advantage of the truncated output bits to add several more steps. If we add another step in the beginning or in the end, the existing difference pattern remains unchanged, but there will be differences in the word W_0 (computable via backward expansion, which includes $W_{i+9} = W_9$, the previous W_8 from Figure 5.3d) or in the new word W_{39} (via the normal forward expansion, which includes $W_{39-15} = W_{24}$), respectively. These, in turn, can imply differences in E_0 or in A_{38} and E_{38} , which translates to differences in the IV (turning semi-free-start into free-start results, and included in the hash value via the feed-forward) or directly in the compression function output, respectively.

The advantage of adding steps in the beginning is that it is possible to limit the additional differences in the state update words to E , and keep A free of new differences. Any differences in E_{-1}, \dots, E_{-4} will be added to the compression function output with the final feed-forward, but the corresponding words of the result are truncated, so the hash outputs still collide. Note that we have ignored the padding for the following free-start collisions.

Free-start collisions for 43-step SHA-512/256

Since SHA-512/256 truncates the last 4 output words of the compression function call ($E_{79} + E_{-1}$, $E_{78} + E_{-2}$, $E_{77} + E_{-3}$, and $E_{76} + E_{-4}$), differences in E_{-1}, \dots, E_{-4} are acceptable for a free-start collision. This observation allows us to add 4 additional steps in the beginning of the 39-step starting point from Figure 5.3d. Shifting the characteristic “downwards” by 4 steps causes the previous message words W_{12}, \dots, W_{15} to turn into new expanded message words W_{16}, \dots, W_{19} ; in particular, this affects the difference in the previous word W_{12} . To determine a compatible difference pattern for the new first 4 words, the message expansion can be computed backwards from the new words W_4, \dots, W_{19} via

$$W_i = W_{i+16} - \sigma_1(W_{i+14}) - W_{i+9} - \sigma_0(W_{i+1}).$$

It turns out that all 4 new words will contain differences (W_3 from $W_{3+9} = W_{12}$; W_2 from $W_{2+1} = W_3$ and $W_{2+14} = W_{16}$; W_1 from $W_{1+1} = W_2$ and $W_{1+14} = W_{15}$; and W_0 from $W_{0+1} = W_1$, $W_{0+14} = W_{14}$ and $W_{0+16} = W_{16}$). However, similar to steps 27–30, the state words A_i and E_i can be kept free of differences for 4 steps. To achieve this, the search tool needs to find differences in the IV words E_{-4}, \dots, E_{-1} to cancel out those in W_0, \dots, W_3 when computing E_0, \dots, E_3 . The resulting starting point is given in Figure 5.4a.

For the search procedure with the solving tool, we fixed the signed differences of steps 12–30 to the same values as the 39-step SHA-512 semi-free-start collision of § 5.3.1. Then,

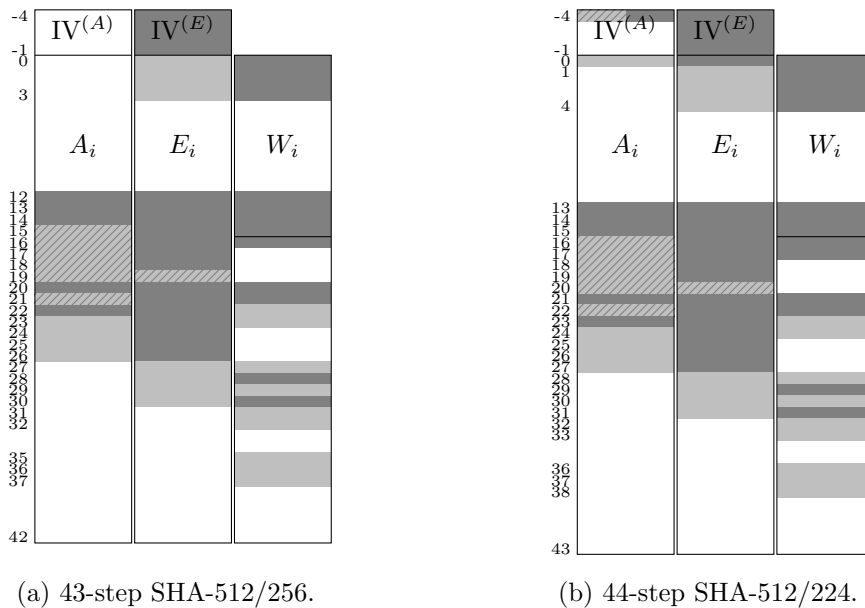


Figure 5.4: Potential free-start starting points (differences \blacksquare and cancellations \blacksquare , \hbar).

to complete the characteristic, we first search for a valid solution for the dense part of the middle steps (A_i and E_i in steps 13–16, and E_i in steps 17–27), and finally fix the corresponding message words W_i in steps 13–17, which determines the complete state, including the dense differences in the prepended steps and IV.

The search only takes seconds on a standard computer; an example for a free-start collision is given in Appendix A in Table A.4b. We also give a free-start collision for 39-step SHA-224 obtained with a similar method (with 1 additional step added to the 38-step SHA-256 characteristic [17]) in Table A.2b.

Free-start collisions for 44-step SHA-512/224

A very similar strategy can be employed to extend the previous 43-step free-start collision by another step for SHA-512/224. Prepending an additional step shifts the difference of previous word E_{-1} to E_0 , which in turn requires a cancellation in A_0 and a difference in A_{-4} , as illustrated in Figure 5.4b. However, only the least significant 32 bits of the corresponding compression function output word are truncated. Furthermore, this output word is computed from A_{-4} via modular addition, so even differences only in the lower 32 bits can possibly cause differences in the untruncated output bits.

Fortunately, the underlying characteristic of signed differences as used for the 39-step SHA-512 semi-free-start collision is well compatible with our constraints: The difference in A_{-4} needs to cancel that in W_4 in a modular addition (via E_0 , by equations (1.3) and (1.2) or Figure 5.2, since all other involved words have zero difference). This difference of W_4 , in turn, is dictated by that in W_{13} (by the update rule for W_{20} , where again all other involved words have zero difference). None of these equalities involves any of

the bitwise functions $\sigma, \Sigma, \text{MAJ}$ or IF . Thus, the modular difference in A_{-4} must be the same as that in W_{13} , which is already fixed by the underlying characteristic to a modular difference of $+32$. Written as bitwise differences, this will translate to a single-bit difference (in the sixth least significant bit) with probability $\frac{1}{2}$ (which does not carry over to the untruncated bits of the final output with overwhelming probability). Indeed, the example for a free-start collision given in Appendix A in Table A.3b only displays this single-bit difference in A_{-4} (and no carries in the output bits).

5.3.3 Collisions

So far, the best practical collisions found for SHA-512 are those for 24 steps, proposed independently by Sanadhya and Sarkar [24] and Indesteege et al. [9], together with 24-step collisions for SHA-256. While the results for SHA-256 have since been improved to 27 [16], 28 [17] (both practical), and finally 31 steps [17] (theoretical attack with almost practical complexity), no such improvements have been proposed for SHA-512 so far. The main reason for this seems to be the doubling in state size from SHA-256 to SHA-512; this larger search space increases the difficulty of the problem for the solving tools.

Starting point for SHA-512

Since the message expansion is essentially the same for all SHA-2 variants (except for different word sizes and rotation values, of course), the SHA-256 starting points can theoretically also be used for SHA-512. However, the resulting search complexity is different. For our results, we used the 27-step starting point (based on a local collision over the $t = 11$ steps 7–17), as illustrated in Figure 5.3a. Just as the 39-step semi-free-start starting point (Figure 5.3d), it requires that differences cancel in E in 4 of the t steps (E_{14}, \dots, E_{17}) and in A in the 4 previous steps (A_{10}, \dots, A_{13}), as well as in several steps of the message expansion.

Finding a solution from this starting point requires significantly more effort than for SHA-256. Of course, we also tried to expand our search to the closely related 28-step starting point, which adds an additional step in the beginning of the 27-step version. However, the additional constraints imposed on the message expansion by this added step prevented us from finding any suitable (reasonably sparse) characteristics.

In contrast to the results from § 5.3.2, since the IV needs to exactly match the original IV, we were not able to take advantage of the final truncation to simplify the search process, or add additional steps. We first search a characteristic for SHA-512, and then try to use it to match the different IVs for SHA-512/224 and SHA-512/256.

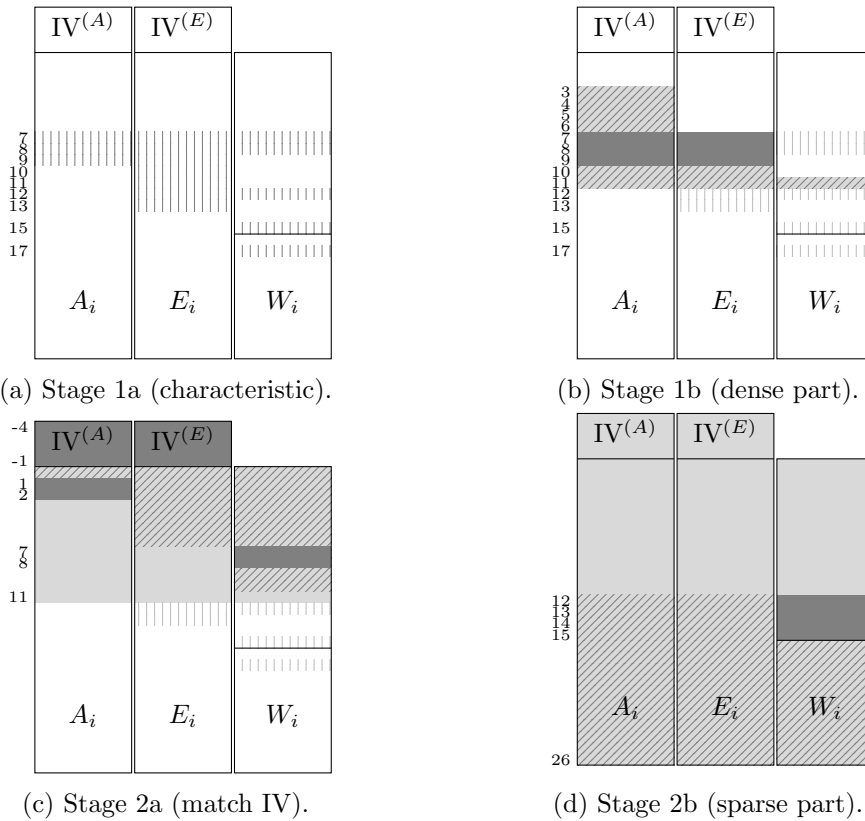


Figure 5.5: Stages of the 27-step collision search (guessed values \blacksquare and differences \square , derived values $\text{\textbackslash}\text{\textbackslash}$, and previously fixed values \blacksquare and differences \square).

Search strategy

The search progresses in several stages, as illustrated in Figure 5.5:

1. Fix signed characteristic:

- (a) **Find candidate characteristic** (Figure 5.5a): First fix the signed differences of the message expansion W (5 words) and state update A (3 words). Since the word W_{17} poses conditions on the first few message words, whose freedom we will later need to match the IV, we focus on keeping its signed difference as sparse as possible, with only few difference bits. With much lower priority, also determine the differences in the state update words E (7 words) to complete the signed characteristic. The characteristic is very dense in E , but this only has limited influence on the success of the IV matching phase.
- (b) **Verify dense parts** (Figure 5.5b): Fully determine the values of A and E in the densest steps 7–9 to verify the validity of the candidate characteristic. If necessary, fix any remaining free bits of A and E in steps 10–11. This fully determines A_3, \dots, A_{11} , E_7, \dots, E_{11} and W_{11} .

To maneuver the search process in the large search space and detect contradictions as soon as possible, we need to apply the look-ahead strategies previously employed for semi-free-start collisions on SHA-512 [8] in this stage (with 16 look-ahead candidates per guess).

2. **Message modification to match IV:** Starting from the best signed characteristics of the previous stage, with the correct IV inserted, find a solution message pair step by step:
 - (a) **Match IV** (Figure 5.5c): Fix the values in the more difficult, heavily constrained words first (W_{10}, W_9, W_8, W_7). Choosing W_{10} and W_9 also determines A_2 and A_1 (via E_6 and E_5). Together with W_7, W_8 , and the IV, this determines all values in steps 0–11.
 - (b) **Finalize message for sparse parts** (Figure 5.5d): choosing the 4 remaining message words W_{12}, \dots, W_{15} allows to satisfy the remaining, sparse parts of the characteristic in steps 12–26 with high probability.

Unlike the other stages, guesses are not made randomly here, but systematically word-by-word. Since most conditions are from modular additions, we always start from the least significant bits and proceed towards the more significant bits. This last stage needs to be repeated for each IV separately, which takes some hours on a single CPU per target IV.

Our results for collisions for 27-step SHA-512/224 and SHA-512/256 are given in Appendix A in Tables A.3a and A.4a, respectively. We also include a 28-step collision for SHA-224, based on the SHA-256 characteristic [17], in Table A.2a.

5.4 Conclusion

The best practical collision attacks for SHA-224 are 28-step collisions, 38-step semi-free-start collisions [17], and 39-step free-start collisions (without padding). An almost practical attack is the 31-step collision with a complexity of $2^{65.5}$ [17]. Free-start collisions (with padding) for up to 40 steps are possible with a complexity of 2^{110} [15]. Additionally, non-random properties were demonstrated via practical second-order collisions for up to 47 steps [3].

For SHA-512/224 and SHA-512/256, the best practical results are 27-step collisions, 39-step semi-free-start collisions, and 43-step (SHA-512/256) or 44-step (SHA-512/224) free-start collisions (without padding). Second-order collision attacks cover up to 48 steps [28]. No attacks on more steps with theoretical complexity are known to us.

Chapter 6

Summary and Discussion

In this report, we presented a detailed overview of the known results on the cryptanalysis of the SHA-2 family. While several results have been published for SHA-256 and SHA-512 in the past, the truncated variants of the SHA-2 family have received significantly less public cryptanalysis. This is particularly true for the two newest members of the SHA-2 family, SHA-512/224 and SHA-512/256.

In Chapter 3, we have given an overview about generic attacks on the Merkle-Damgård construction and discussed the applicability to the truncated variants. Since only 32 bits are truncated for SHA-224, the presented generic attacks are still applicable, but the gain is not as big as for the untruncated variants (SHA-256 and SHA-512). In contrast, the internal state of SHA-512/224 and SHA-512/256 is at least twice as big as the hash value. This precludes most generic attacks.

As discussed in Chapter 4, little work has been published so far about (second) preimage attacks on the truncated variants of the SHA-2 family. Although not published, we believe that pseudo-preimage for a similar number of steps are possible for the truncated variants as for the untruncated variants by using similar attack strategies. However, the situation changes if we take a look at preimages. As discussed in Chapter 4, we think that the currently used meet-in-the-middle approaches cannot be used to attack an equally high number of steps anymore. Due to the larger internal state in the truncated variants of the SHA-2 family, pseudo-preimages cannot be converted into preimage attacks so easily. For this reason, we considered practical attacks based on equation solving. Our approach is based on SAT solving, an idea previously applied successfully to MD5 and other hash functions. The results are practical preimage attacks for 18 steps.

In contrast to preimage attacks, most published collision attacks for the SHA-2 family are of practical complexity and could easily be extended to the truncated variants. Furthermore, as shown in Chapter 5, the truncation allows us to mount practical free-start collision attacks on several more steps of the truncated variants, compared to SHA-256 and SHA-512.

All the results of our analysis are summarized in Table 6.1. Based on this analysis, we do not expect new spectacular breakthroughs related to collision or preimage attacks on the truncated variants of the SHA-2 family in the near future.

Table 6.1: Summary of attacks on SHA-224, SHA-512/224, and SHA-512/256.

Target	Attack	Steps	Complexity	Reference
SHA-512/256	preimage	18	practical	this work
	collision	27	practical	this work
	semi-free-start collision	39	practical	this work
	free-start collision*	43	practical	this work
	distinguisher	48	practical	[28]
SHA-512/224	preimage	18	practical	this work
	collision	27	practical	this work
	semi-free-start collision	39	practical	this work
	free-start collision*	44	practical	this work
	distinguisher	48	practical	[28]
SHA-224	preimage	18	practical	this work
	pseudo-preimage	43	2^{220}	[1]
	collision	28	practical	[17]
		31	$2^{65.5}$	[17]
	semi-free-start collision	38	practical	[17]
	free-start collision*	39	practical	this work
	free-start collision	40	2^{110}	[15]
distinguisher	47	practical	[3]	

(*) without padding

Bibliography

- [1] Aoki, K., Guo, J., Matusiewicz, K., Sasaki, Y., Wang, L.: Preimages for step-reduced SHA-2. In: Matsui, M. (ed.) *Advances in Cryptology – ASIACRYPT 2009*. LNCS, vol. 5912, pp. 578–597. Springer (2009)
- [2] Aoki, K., Sasaki, Y.: Preimage attacks on one-block MD4, 63-step MD5 and more. In: Avanzi, R.M., Keliher, L., Sica, F. (eds.) *Selected Areas in Cryptography – SAC 2008*. LNCS, vol. 5381, pp. 103–119. Springer (2008)
- [3] Biryukov, A., Lamberger, M., Mendel, F., Nikolic, I.: Second-order differential collisions for reduced SHA-256. In: Lee, D.H., Wang, X. (eds.) *Advances in Cryptology – ASIACRYPT 2011*. LNCS, vol. 7073, pp. 270–287. Springer (2011)
- [4] Coron, J., Dodis, Y., Malinaud, C., Puniya, P.: Merkle-Damgård revisited: How to construct a hash function. In: Shoup, V. (ed.) *Advances in Cryptology – CRYPTO 2005*. LNCS, vol. 3621, pp. 430–448. Springer (2005)
- [5] Damgård, I.: A design principle for hash functions. In: Brassard, G. (ed.) *Advances in Cryptology – CRYPTO ’89*. LNCS, vol. 435, pp. 416–427. Springer (1989)
- [6] Dang, Q.: NIST SP 800-107 revision 1. Recommendation for applications using approved hash algorithms (August 2012), <http://csrc.nist.gov/publications/nistpubs/800-107-rev1/sp800-107-rev1.pdf>
- [7] De Cannière, C., Rechberger, C.: Finding SHA-1 characteristics: General results and applications. In: Lai, X., Chen, K. (eds.) *Advances in Cryptology – ASIACRYPT 2006*. LNCS, vol. 4284, pp. 1–20. Springer (2006)
- [8] Eichlseder, M., Mendel, F., Schläffer, M.: Branching heuristics in differential collision search with applications to SHA-512. In: Cid, C., Rechberger, C. (eds.) *Fast Software Encryption – FSE 2014*. LNCS, Springer (2014), in press
- [9] Indestege, S., Mendel, F., Preneel, B., Rechberger, C.: Collisions and other non-random properties for step-reduced SHA-256. In: Avanzi, R.M., Keliher, L., Sica, F. (eds.) *Selected Areas in Cryptography – SAC 2008*. LNCS, vol. 5381, pp. 276–293. Springer (2009)

- [10] Joux, A.: Multicollisions in iterated hash functions. Application to cascaded constructions. In: Franklin, M.K. (ed.) *Advances in Cryptology – CRYPTO 2004*. LNCS, vol. 3152, pp. 306–316. Springer (2004)
- [11] Kelsey, J., Kohno, T.: Herding hash functions and the Nostradamus attack. In: Vaudenay, S. (ed.) *Advances in Cryptology – EUROCRYPT 2006*. LNCS, vol. 4004, pp. 183–200. Springer (2006)
- [12] Kelsey, J., Schneier, B.: Second preimages on n -bit hash functions for much less than 2^n work. In: Cramer, R. (ed.) *Advances in Cryptology – EUROCRYPT 2005*. LNCS, vol. 3494, pp. 474–490. Springer (2005)
- [13] Khovratovich, D., Rechberger, C., Savelieva, A.: Bicliques for preimages: Attacks on Skein-512 and the SHA-2 family. In: Canteaut, A. (ed.) *Fast Software Encryption – FSE 2012*. LNCS, vol. 7549, pp. 244–263. Springer (2012)
- [14] Lai, X., Massey, J.L.: Hash function based on block ciphers. In: Rueppel, R.A. (ed.) *Advances in Cryptology – EUROCRYPT '92*. LNCS, vol. 658, pp. 55–70. Springer (1992)
- [15] Li, J., Isobe, T., Shibutani, K.: Converting meet-in-the-middle preimage attack into pseudo collision attack: Application to SHA-2. In: Canteaut, A. (ed.) *Fast Software Encryption – FSE 2012*. LNCS, vol. 7549, pp. 264–286. Springer (2012)
- [16] Mendel, F., Nad, T., Schl affer, M.: Finding SHA-2 characteristics: Searching through a minefield of contradictions. In: Lee, D.H., Wang, X. (eds.) *Advances in Cryptology – ASIACRYPT 2011*. LNCS, vol. 7073, pp. 288–307. Springer (2011)
- [17] Mendel, F., Nad, T., Schl affer, M.: Improving local collisions: New attacks on reduced SHA-256. In: Johansson, T., Nguyen, P.Q. (eds.) *Advances in Cryptology – EUROCRYPT 2013*. LNCS, vol. 7881, pp. 262–278. Springer (2013)
- [18] Menezes, A., van Oorschot, P.C., Vanstone, S.A.: *Handbook of Applied Cryptography*. CRC Press (1996)
- [19] Merkle, R.C.: One way hash functions and DES. In: Brassard, G. (ed.) *Advances in Cryptology – CRYPTO '89*. LNCS, vol. 435, pp. 428–446. Springer (1989)
- [20] Mironov, I., Zhang, L.: Applications of SAT solvers to cryptanalysis of hash functions. In: Biere, A., Gomes, C.P. (eds.) *Theory and Applications of Satisfiability Testing – SAT 2006*. LNCS, vol. 4121, pp. 102–115. Springer (2006)
- [21] National Institute of Standards and Technology: Announcing request for candidate algorithm nominations for a new cryptographic hash algorithm (SHA-3) family. Federal Register 27(212), 62212–62220 (November 2007), http://csrc.nist.gov/groups/ST/hash/documents/FR_Notice_Nov07.pdf

- [22] National Institute of Standards and Technology: FIPS PUB 180-4: Secure Hash Standard. Federal Information Processing Standards Publication 180-4, U.S. Department of Commerce (March 2012), <http://csrc.nist.gov/publications/fips/fips180-4/fips-180-4.pdf>
- [23] Rogaway, P., Shrimpton, T.: Cryptographic hash-function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance. In: Roy, B.K., Meier, W. (eds.) *Fast Software Encryption – FSE 2004*. LNCS, vol. 3017, pp. 371–388. Springer (2004)
- [24] Sanadhya, S.K., Sarkar, P.: New collision attacks against up to 24-step SHA-2. In: Chowdhury, D.R., Rijmen, V., Das, A. (eds.) *Progress in Cryptology – INDOCRYPT 2008*. LNCS, vol. 5365, pp. 91–103. Springer (2008)
- [25] Stinson, D.R.: Some observations on the theory of cryptographic hash functions. *Designs, Codes and Cryptography* 38(2), 259–277 (2006)
- [26] Wang, X., Yin, Y.L., Yu, H.: Finding collisions in the full SHA-1. In: Shoup, V. (ed.) *Advances in Cryptology – CRYPTO 2005*. LNCS, vol. 3621, pp. 17–36. Springer (2005)
- [27] Wang, X., Yu, H.: How to break MD5 and other hash functions. In: Cramer, R. (ed.) *Advances in Cryptology – EUROCRYPT 2005*. LNCS, vol. 3494, pp. 19–35. Springer (2005)
- [28] Yu, H., Bai, D.: Boomerang attack on step-reduced SHA-512. *IACR Cryptology ePrint Archive, Report 2014/945* (2014), <http://eprint.iacr.org/2014/945>
- [29] Yu, H., Wang, X.: Distinguishing attack on the secret-prefix MAC based on the 39-step SHA-256. In: Boyd, C., Nieto, J.M.G. (eds.) *Information Security and Privacy – ACISP 2009*. LNCS, vol. 5594, pp. 185–201. Springer (2009)

Appendix A

Practical examples

A.1 Preimage examples

Table A.1 gives examples for preimages of the all-zero hash value for 18-step SHA-2 variants. The given messages already include correct padding.

Table A.1: Preimage examples for 18-step SHA-2.

(a) Example for 18 steps of SHA-224.

m	5591f19b 1e174173 ae2a6d79 aeadacae 1fe7fe67 e2976b66 68e32a5a 37a51fc5 64f0d2e6 8a21612f c6e9be2b 8b63908c 9e178ef1 0229fee1 00000000 000001bf
h_1	00000000 00000000 00000000 00000000 00000000 00000000 00000000

(b) Example for 18 steps of SHA-512/224.

m	e6843e38c707e63c d6792d88ffcba4c 6f50d86c5fd13705 13ebd8cc02f50f21 d56123648cab6609 872159f6675f9037 636543b42e076221 28395f2079f84e49 59e0958d2f06d8a1 73e433284ba40d4a d4b60ad9e368a54a 16db11505b6012ce d22b26bafef07639f 363301def9e35b2b 0000000000000000 000000000000037f
h_1	0000000000000000 0000000000000000 0000000000000000 00000000

(c) Example for 18 steps of SHA-512/256.

m	ca3d05aae841ea94 523f1613943d27c0 e3fa8f925d513a70 f818bf091828fb2d 875731c87d8b2e1f 911671a61869fcef f0bb93da2de861e1 f02550dd12bc482d fb0cf0420ade87e4 7e1434b7c2f9d009 8170ee8fcf4967d0 680f90f56e3ccd14 362883b52d17533a d6c00e91677bbf8b 0000000000000000 000000000000037f
h_1	0000000000000000 0000000000000000 0000000000000000 0000000000000000

A.2 Collision examples

Results for the free-start collisions of § 5.3.2 are given in Tables A.2b, A.3b, and A.4b, and for the collisions of § 5.3.3 in Tables A.2a, A.3a, and A.4a.

Table A.2: Results for SHA-224.

(a) Example of a collision for 28 steps of SHA-224.

m	a3e2972c 73ba31f5 e9f85a00 c2cb8cbc 4dd15d27 d240b6f6 0c1243ec 07504bfd 134d0e53 fb839dcc e14f2cbc 53c3c1c6 6ac516a7 a19b112f d844f621 939e7e53
m^*	a3e2972c 73ba31f5 e9f85a00 c2cb8cbc 4dd15d27 d240b6f6 0c1243ec 07504bfd e3f85ef3 f75b9934 e14f2cbc 53c3c1c6 6ac516a7 a3db19bf d844f621 939e7e53
Δm	00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 f0b550a0 0cd804f8 00000000 00000000 00000000 02400890 00000000 00000000
h_1	869fd0b5 fb96d20b 28177d1e c7af4885 06ecb162 88332da4 5022b6c7

(b) Example of a free-start collision for 39 steps of SHA-224.

h_0	5594e74a 2234bcbd 635966b5 97a9e488 4a6a7d63 7c28ca05 f1384c84 d2c9753a
h_0^*	5594e74a 2234bcbd 635966b5 97a9e488 4a6a7d63 7c28ca05 f1384c84 d2c97532
Δh_0	00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000008
m	949722de 48501dcc 7fe48849 ba821c7a 5b5a1e6f 30c487c8 401134e4 2d9eacc7 dd15d12f 9079b000 37c75e69 a47f0dde 8baaf91a d348cc06 2b64ef59 011f91be
m^*	949722e6 48501dcc 7fe48849 ba821c7a 5b5a1e6f 30c487c8 401134e4 2d9eacc7 e373cfef 9079affc 37c75e69 a4808dde 8baaf91a d348cc06 2b64ef59 011f91be
Δm	00000038 00000000 00000000 00000000 00000000 00000000 00000000 00000000 3e661ec0 00001ffc 00000000 00ff8000 00000000 00000000 00000000 00000000
h_1	1d6d980a 2aa5f9c0 9843296b da4f8baa 09c36608 7e2bdad9 cb0f1654

Table A.3: Results for SHA-512/224.

(a) Example of a collision for 27 steps of SHA-512/224.

m	20dbf13a352116a9 295506e205afd435 abfe4826742c1a1a 279f07c7813dd9be 47da77c701a98858 25aec1349d486501 37a992a15616ea31 e2b122ecf19e90d3 2fff6025dc03dd67 032c261d740f459e 2e2599bd6e7e74df d490bd22815eb494 72fedf1f607df6e3 87fc91fcfb7397fd e647b1b499eee17f 2dff8e493cbc8a4c
m^*	20dbf13a352116a9 295506e205afd435 abfe4826742c1a1a 279f07c7813dd9be 47da77c701a98858 25aec1349d486501 37a992a15616ea31 5cc1250cb19e90d3 203fdfe5dc03dd66 032c261d740f459e 2e2599bd6e7e74df d490bd22815eb494 f0bc01167075f6eb 87fc91fcfb7397fd e647b1b499eee17f d3f8fe713d7c8a4c
Δm	0000000000000000 0000000000000000 0000000000000000 0000000000000000 0000000000000000 0000000000000000 0000000000000000 be7007e040000000 0fc0bfc000000001 0000000000000000 0000000000000000 0000000000000000 8242de0910080008 0000000000000000 0000000000000000 fe07703801c00000
h_1	65b11e66e48da563 1b70d12da92e2dba 8f338768bb95601b 60b995bb

(b) Example of a free-start collision for 44 steps of SHA-512/224.

h_0	fef65b64d3694995 959fbfb82ed84eb1 1d9e855642e62ef2 335cc6d027695d91 921d197e5cfa2803 e26c6eb26163a692 9ff3cf4d26f1de78 5323942861d9139a
h_0^*	fef65b64d3694995 959fbfb82ed84eb1 1d9e855642e62ef2 335cc6d027695db1 a712860cdcfa1ff8 470749bbf7628f44 20cdfd694df67216 8e07b5fa2c7fedf0
Δh_0	0000000000000000 0000000000000000 0000000000000000 0000000000000020 350f9f72800037fb a56b2709960129d6 bf3e32246b07ac6e dd2421d24da6fe6a
m	7a19df6089d00684 03ed2a0d0c29e00e 36c91e35f681fbb8 bb2b47428aef294 dce94ccc981d39a3 44230f73cf56d9ef e9d46b26b44950c8 550bed4b9419741c 58a98894206e00de f3448a6f761d384d 9ae59f3a3bcc5bba ece85d5c77be431b 6e3cf817e9376cc7 b74a2a43c0b96c93 7c5b51d6fe2a0c26 5a9868e5bf2e422d
m^*	5e031bbe28b2d027 ded424ef85255cc3 ad2f514be0830c1f a635dab40aef294 dce94ccc981d3983 44230f73cf56d9ef e9d46b26b44950c8 550bed4b9419741c 58a98894206e00de f3448a6f761d384d 9ae59f3a3bcc5bba ece85d5c77be431b 6e3cf817e9376cc7 b74a2a43c0b96cb3 5c5b51d6fe2a0c36 5a9868e5bf2e420d
Δm	241ac4dea162d6a3 dd390ee2890cbccd 9be64f7e1602f7a7 1d1e9df68000080b 0000000000000020 0000000000000000 0000000000000000 0000000000000000 0000000000000000 0000000000000000 0000000000000000 0000000000000000 0000000000000000 0000000000000020 2000000000000010 0000000000000020
h_1	e309edf68f4d89b8 5c356e0359eb0dab 76b4a45ec3c2cd25 8bd0955d

Table A.4: Results for SHA-512/256.

(a) Example of a collision for 27 steps of SHA-512/256.

m	306b0c2ebe7c1341 c8b55d4df1c5f4fe b91a173aeceb818a 33b5977f9b46e58b 6c6d5a4f87f1364f 1b7e33249d4acf4f b7f784ecdcaefc1f a33edafe7afc0452 dfc0200932c2b9df faec7d05e3518e56 ec2e19a7ee867396 d490bd22815eb494 72fedf1f887df303 f95891f08483da25 c327d0afa2c4f902 2c5f0c0806a4e298
m^*	306b0c2ebe7c1341 c8b55d4df1c5f4fe b91a173aeceb818a 33b5977f9b46e58b 6c6d5a4f87f1364f 1b7e33249d4acf4f b7f784ecdcaefc1f 1d4edd1e3afc0452 d0009fc932c2b9de faec7d05e3518e56 ec2e19a7ee867396 d490bd22815eb494 f0bc01169875f30b f95891f08483da25 c327d0afa2c4f902 d2587c300764e298
Δm	0000000000000000 0000000000000000 0000000000000000 0000000000000000 0000000000000000 0000000000000000 0000000000000000 be7007e040000000 0fc0bfc000000001 0000000000000000 0000000000000000 0000000000000000 8242de0910080008 0000000000000000 0000000000000000 fe07703801c00000
h_1	fcb5c8faf05fd68 c676b8f17b5daae3 6233801174b7fd01 0ff72ab4a869c54f

(b) Example of a free-start collision for 43 steps of SHA-512/256.

h_0	159b52516f10f30d 546b2042f240afee f25339b24c441edf d62c698666558242 e5a9e39861fbd81d d2138eacc20d5224 a332c16df23609fb 73f78341dfd7a4e5
h_0^*	159b52516f10f30d 546b2042f240afee f25339b24c441edf d62c698666558242 e5a9e39861fbd83d 72e259ce420d5a0f 4db37906cc361264 ae579d9e0275b446
Δh_0	0000000000000000 0000000000000000 0000000000000000 0000000000000000 0000000000000020 a0f1d7628000082b ee81b86b3e001b9f dda01edfdda210a3
m	cfbec86f1cf6821e dd3343c25aad835a 2a08612b753f3d6b b328d40d2c624ef7 b3e51f8a3a63bd6f 4abdf96375bbf609 a8c5c1f784672e86 a78e2aa625830d4b 169dcb5039bf3d9f fbcc43ffe8bd8ae47 1b3eaeffc5c6a46 f668a2a728851b4e 374601ea44422bdb 2ca290d26a23a02f 6685babbfdbc5e22 e000111457201fd4
m^*	ee37d77210586a56 b2a4122800ad72cf 89399609f53f3560 b328d40d2c624ed7 b3e51f8a3a63bd6f 4abdf96375bbf609 a8c5c1f784672e86 a78e2aa625830d4b 169dcb5039bf3d9f fbcc43ffe8bd8ae47 1b3eaeffc5c6a46 f668a2a728851b4e 374601ea44422bfb 0ca290d26a23a03f 6685babbfdbc5e02 e07e151457202055
Δm	21891f1d0caee848 6f9751ea5a00f195 a331f7228000080b 0000000000000020 0000000000000000 0000000000000000 0000000000000000 0000000000000000 0000000000000000 0000000000000000 0000000000000000 0000000000000000 0000000000000020 2000000000000010 0000000000000020 007e040000003f81
h_1	1d7041bbbf6a676a 03d8c440d9246b9d 20ce2d17c5b0b2c4 7e6e4d33a7f54afd

A.3 Collision characteristics

Characteristics for the semi-free-start collisions of § 5.3.1 are given in Table A.9; for the free-start collisions of § 5.3.2 in Tables A.7, A.10, and A.11; and for the collisions of § 5.3.3 in Tables A.6 and A.8. All characteristics show the signed differences (**u**, **n**) and their immediately necessary conditions (0, 1), all given using the bitwise notation of generalized conditions [7], see Table A.5. Additionally, all bits with two-bit conditions [16] are highlighted (◻).

Table A.5: Notation for all generalized conditions on a pair of bits [7].

(X_i, X_i^*)	(0, 0)	(1, 0)	(0, 1)	(1, 1)	(X_i, X_i^*)	(0, 0)	(1, 0)	(0, 1)	(1, 1)
?	✓	✓	✓	✓	3	✓	✓	-	-
-	✓	-	-	✓	5	✓	-	✓	-
x	-	✓	✓	-	7	✓	✓	✓	-
0	✓	-	-	-	A	-	✓	-	✓
u	-	✓	-	-	B	✓	✓	-	✓
n	-	-	✓	-	C	-	-	✓	✓
1	-	-	-	✓	D	✓	-	✓	✓
#	-	-	-	-	E	-	✓	✓	✓

Table A.6: Characteristic for a collision for 28 steps of SHA-224.

i	A_i	E_i	W_i
-4			
-3			
-2			
-1			
0			
1			
2			
3			
4			
5			
6			
7			
8	0uu-nuu-u-uu-mnn-uu-nu-n	0uu-1nu-uun-u-0uu-n0--nnu11	nnu-n-nn-u-u-n-n-n-n
9	-n-u-n1	1-1011-u1u0-0nn-0-u-n0010uu-0	11-un1-un0nn-110-11u01uunnu
10	nnnnnn-u-u-u-u-uu	0n010n1011101011n011u1110n0nu01u	
11	-1	n-1un1-n11000n10n0110n10001-u0	
12	-1	0011n11n00u0n11u0uu10110uu10-00	
13		000100010n011nuuuuuuu1n11011101	-n-n-n-n-1n-u
14		11-00u-0un0u000=00-u0nn-annu-0	-1
15		1-10-11001011-00-0001	
16		1-01-1-0-00-1111	1-u-0uun-10un01uun-n
17			
18			n-n-n-n-1-n01n-n-u
19			
20			
21			
22			
23			
24			
25			
26			
27			

Table A.7: Characteristic for a free-start collision for 39 steps of SHA-224.

i	A_i	E_i	W_i
-4			
-3			
-2			
-1			
0			nuu
1			
2			
3			
4			
5			
6		0-0-00	
7		1-1-11	
8	n-u-nn-u	nu-u-nn-u-nnnnnnnn-nn	nuuuu-nn-un-unn-nn
9	n-uuuuu-n-un-uuunn-u	un-00-nnn-n1-n-n0-nnn-0-n-u	1001000001111-1101unnnnnnnnn00
10	uuu-u-n-u-u-n	u0nn0-u10-n01-10111uu0un1-u0uu	
11		0110n0u-n10nu0-u-10u0nn-001-	nuuuuuuuu
12		nnnn1-11-111101000111-0100	
13		u010u1uu10-n0un111-1u010u1u0-1	
14		n01n11-111u-n001-u1=0u0-0100n-1	
15		010-11-0-0010-0-10-10-01-	
16	-u-n	0100u1-000-nuuu=1000-11-100110	-n-
17		00-0nu-1100000u00110-000n-1u00	
18		100-n01-unn-01un-nuu1-10-1-01nu	
19		111-000-11-000-1-1-100	
20		0-uuuuuuuuuu-111-0-0-00	
21		1-00010111-1nuuuuuuu10	
22		11111111111-000000000-	
23		1111111111-	
24			n-un-u-
25			
26			
27			
28			
29			
30			
31			
32			
33			
34			
35			
36			
37			
38			

Table A.8: Characteristic for a collision for 27 steps of SHA-512 and SHA-512/t.

z	A_i	E_i	W_i
-4			
-3			
-2			
-1			
0			
1			
2			
3			
4			
5			
6			
7			
8			
9			
10			
11			
12			
13			
14			
15			
16			
17			
18			
19			
20			
21			
22			
23			
24			
25			
26			

Table A.9: Characteristic for a semi-free-start collision for 39 steps of SHA-512 and SHA-512/t

i	A_i	E_i	W_i
-4			
-3			
-2			
-1			
1			
2			
3			
4			
5			
6			
7			
8			
9			
10			
11			
12			
13			
14			
15			
16			
17			
18			
19			
20			
21			
22			
23			
24			
25			
26			
27			
28			
29			
30			
31			
32			
33			
34			
35			
36			
37			
38			

Table A.10: Characteristic for a free-start collision for 44 steps of SHA-512/224.

	A_i	E_i	W_i
1			
2			
3			
4			
5			
6			
7			
8			
9			
10			
11			
12			
13			
14			
15			
16			
17			
18			
19			
20			
21			
22			
23			
24			
25			
26			
27			
28			
29			
30			
31			
32			
33			
34			
35			
36			
37			
38			
39			
40			
41			
42			
43			

Table A.11: Characteristic for a free-start collision for 43 steps of SHA-512/256.

i	A_i	E_i	W_i
4			
3			
2			
1			
0			
1			
2			
3			
4			
5			
6			
7			
8			
9			
10			
11			
12			
13			
14			
15			
16			
17			
18			
19			
20			
21			
22			
23			
24			
25			
26			
27			
28			
29			
30			
31			
32			
33			
34			
35			
36			
37			
38			
39			
40			
41			
42			