Current Status of SHA-1

Prof. dr. ir. Vincent Rijmen

DI Florian Mendel
DI Norbert Pramstaller
DI Christian Rechberger

February 21, 2007

Contents

1 Introduction

This report was produced for the Rundfunk und Telekom Regulierungs-GmbH, Austria.

1.1 Scope of this report

This report contains:

1. An overview of the recently published results on the collision resistance of hash functions of the MD4 family, in particular SHA, SHA-1, SHA-224, SHA-256 and RIPEMD-160.
2. A discussion of the impact of these results on cryptographic applications.
3. A discussion of the possibility to continue the use of SHA-1 in cryptographic applications after collision attacks have been demonstrated.

In particular for the second and third point, this report presents only crypto-technical arguments. Legal requirements, for instance as specified by technical annexes to the European Directive on electronic signatures, cf. [11], and the Austrian signature law, are outside the scope of this report. Also not considered are the possible discrepancies between technical results and the public perception of these results (press coverage).

1.2 Unit of operation

As is customary in cryptographic literature, we express the (expected) computational complexity of an attack using as unit operation one iteration of the hash function. A software implementation of SHA-1 processing 128 Mbytes/s, executes 2^{21} iterations of the hash function per second. Such a performance figure is easily achievable on a modern PC with a clock speed of 2 GHz [6].

However, inventors of attacks are often slightly optimistic in estimating the workload of their attacks and the performance achievable with optimized implementations of the hashing algorithms in 'real' applications. This cannot always be duplicated in attacks.

2 The hash functions SHA, SHA-1, SHA-224, SHA-256, and RIPEMD-160

In this section, we will shortly describe the hash functions SHA, SHA-1, SHA-224, SHA-256, and RIPEMD-160, which are based on the design structure of MD4 [31]. This description will illustrate that the designers of the SHA family of hash functions (the NSA) have tried to remedy weaknesses in earlier designs by increasing the complexity in follow-up designs.

2.1 Iterated hash functions

All hash functions of the SHA family and RIPEMD-160 are *iterated* constructions. The variable-length input message is *padded* and subsequently divided into *blocks* of fixed length.

For the hash functions discussed in this report the block length is 512 bits. The hash functions consist of the repeated application of a *compression function* denoted by f_c, which takes as inputs one message block m_i and a chaining variable h_i. The output is the updated value of the chaining variable:

$$h_{i+1} \leftarrow f_c(h_i, m_i). \tag{1}$$

The initial value of the chaining variable is called IV and fixed by the designer. The final value of the chaining variable is the hash function output, referred to as message digest or fingerprint. The compression function consists of three parts:

Message expansion: The 512-bit input message block is expanded into a number of 32-bit words. The number of expanded message words equals the number of steps in the state update transformation.

State update transformation: The state update transformation consists of a number of relatively simple steps. For the SHA family of hash functions, RIPEMD-160, and most other hash functions in use today, the structure of the transformation is very similar to an unbalanced Feistel network [34]. If we compare the state update with a Feistel-like block cipher such as DES [25], then the chaining variable is used as 'message input' (plaintext) and the blocks of the expanded message are used as 'round keys'.

Feed forward: The input of the state update transformation is added to the output value after processing one message block. This is also called the Davies-Meyer construction and ensures that if the input message block is fixed then the compression function is non-invertible in the chaining variable.

2.2 SHA

The original design of the hash function SHA was published by NIST in 1993. It was withdrawn in 1995 and replaced by SHA-1. Both SHA and SHA-1 produce a hash value of 160 bits.

Let a single 512-bit message block be denoted by a vector m, consisting of 16 32-bit words M_i, with $0 \leq i \leq 15$. These 32-bit words are then linearly expanded into 80 32-bit words W_i:

$$W_i = \begin{cases} M_i, & \text{for } \ 0 \leq i \leq 15, \\ W_{i-3} \oplus W_{i-8} \oplus W_{i-14} \oplus W_{i-16} & \text{for } 16 \leq i \leq 79 \ . \end{cases} \tag{2}$$

Besides the extension of the output length from 128 to 160 bits, this message expansion forms the main difference between MD5 [32] and SHA.

The state update transformation operates on 5 32-bit registers, which are initialized with the current value of the chaining variable (IV for processing the first message block). It consists of 80 steps, divided into 4 rounds of 20 steps each. A single step of the state update transformation is shown in Figure 1. In each step the function f is applied to the state variables B_i, C_i, and D_i. The function f depends on the step number: steps 0 to 19 (round 1) use f_{IF} and steps 40 to 59 (round 3) use f_{MAJ}. f_{XOR} is applied in the remaining steps (round 2 and 4). The functions are defined as:

$$\begin{aligned} f_{IF}(B,C,D) &= B \wedge C \oplus \neg B \wedge D \\ f_{MAJ}(B,C,D) &= B \wedge C \oplus B \wedge D \oplus C \wedge D \\ f_{XOR}(B,C,D) &= B \oplus C \oplus D \ , \end{aligned} \tag{3}$$

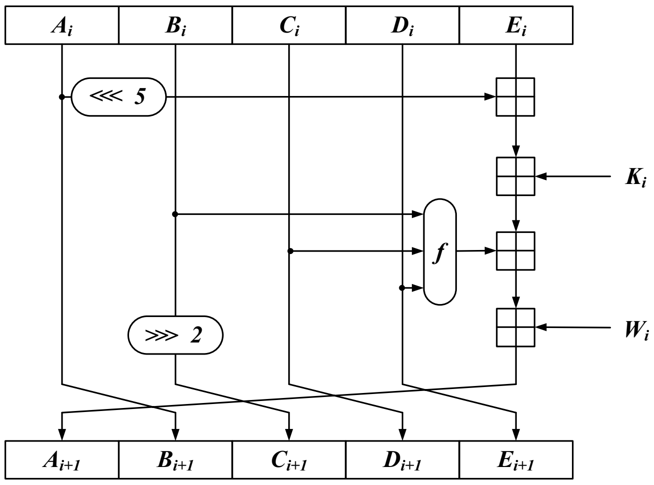

Figure 1: One step of the state update transformation of SHA and SHA-1

where $\wedge$ denotes the logical AND operation, $\oplus$ corresponds to addition modulo 2, and $\neg B$ is the bitwise complement of B. The state update transformation also uses step constants K_i.

2.3 SHA-1

The only difference between SHA and SHA-1 [26] is a one-bit rotation in the message expansion, namely (2) is replaced by:

$$W_i = \begin{cases} M_i, & \text{for } \ 0 \le i \le 15, \\ (W_{i-3} \oplus W_{i-8} \oplus W_{i-14} \oplus W_{i-16}) \lll 1 & \text{for } 16 \le i \le 79 \ . \end{cases} \tag{4}$$

2.4 SHA-224 and SHA-256

SHA-224 and SHA-256 [27] operate on chaining variables of 256 bits. Both hash functions use the same compression function. The differences are the use of a different IV and the fact that the output of SHA-224 is produced by truncating the final value of the 256-bit chaining variable.

The message expansion takes as input a vector m with 16 words M_i and outputs 64 32-bit words W_i, generated according to the following formula:

$$W_i = \begin{cases} M_i & \text{for } 0 \le i < 15 \\ \sigma_1(W_{i-2}) + W_{i-7} + \sigma_0(W_{i-15}) + W_{i-16} & \text{for } 15 \le i < 64 \end{cases} . \tag{5}$$

The functions $\sigma_0(x)$ and $\sigma_1(x)$ are defined as follows:

$$\begin{aligned} \sigma_0(x) &= ROTR^7(x) \oplus ROTR^{18}(x) \oplus SHR^3(x) \\ \sigma_1(x) &= ROTR^{17}(x) \oplus ROTR^{19}(x) \oplus SHR^{10}(x) \ , \end{aligned} \tag{6}$$

where $ROTR^a$ denotes cyclic rotation by a positions to the right, and SHR^a denotes a logical shift by a positions to the right.

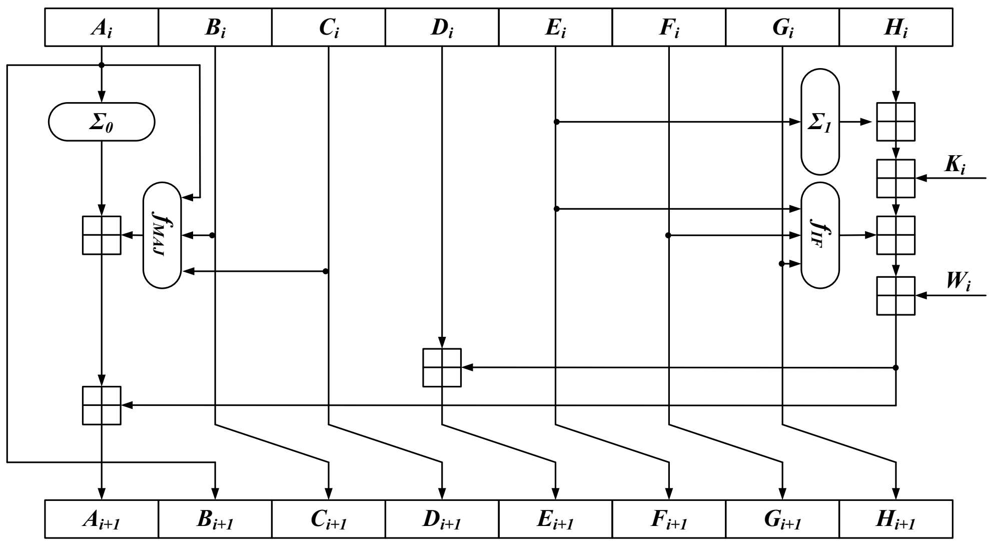

Figure 2: One step of the state update transformation of SHA-224 and SHA-256.

The compression function consists of 64 identical steps. One step is depicted in Figure 2. The step transformation employs the bitwise Boolean functions f_{MAJ} and f_{IF}, and two GF(2)-linear functions $\Sigma_0(x)$ and $\Sigma_1(x)$:

$$\begin{aligned}\Sigma_0(x) &= ROTR^2(x) \oplus ROTR^{13}(x) \oplus ROTR^{22}(x) && (7)\\ \Sigma_1(x) &= ROTR^6(x) \oplus ROTR^{11}(x) \oplus ROTR^{25}(x)\ . && (8)\end{aligned}$$

The i-th step uses a fixed constant K_i and the i-th word W_i of the expanded message.

2.5 RIPEMD-160

The hash function RIPEMD-160 was proposed by Hans Dobbertin, Antoon Bosselaers and Bart Preneel [10]. It produces a 160-bit hash value. Like its predecessor RIPEMD, it consists of two parallel streams. While RIPEMD consists of two parallel streams of MD4, the two streams are designed differently in the case of RIPEMD-160.

The message expansion of RIPEMD-160 is a permutation of the 16 message words in each round, where different permutations are used in each round of the left and the right stream.

In each stream 5 rounds of 16 steps each are used to update the 5 32-bit registers. Figure 3 shows one step transformation. The step transformation employs 5 bitwise Boolean functions $f_1, \ldots, f_5$ in each stream:

$$\begin{aligned}f_1(B,C,D) &= B \oplus C \oplus D\\ f_2(B,C,D) &= (B \wedge C) \oplus (\neg B \wedge D)\\ f_3(B,C,D) &= (B \vee \neg C) \oplus D \qquad (9)\\ f_4(B,C,D) &= (B \wedge D) \oplus (C \wedge \neg D)\\ f_5(B,C,D) &= B \oplus (C \vee \neg D)\ ,\end{aligned}$$

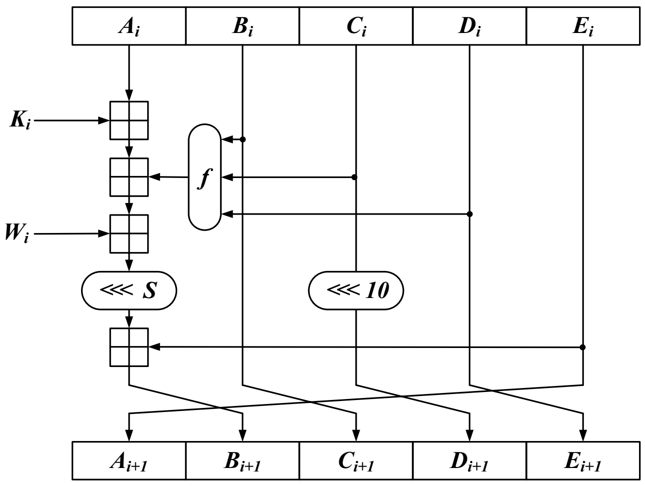

Figure 3: One step of the state update transformation of RIPEMD-160.

where $\vee$ denotes the bitwise OR operation. The order of the Boolean function is different in each stream. A step constant K_i is added in every step; the constant is different for each round and for each stream. Different rotation values S are used in each step and in both streams. After the last step, the initial value and the values of the right and the left stream are combined (feed forward), resulting in the output of one iteration.

2.6 Reduced variants

When analyzing a hash function, cryptographers usually start by looking at variants of the compression function, using a number of steps lower than specified by the designers. While attacks against a very small number of steps say nothing about the strength of the full design, it is commonly believed that attacks against only slightly reduced variants indicate problems with the design.

When reading results on reduced variants, it should be kept in mind that the workload of attacks usually increases exponentially in the number of steps. This implies that a hash function for which there is an attack on a variant with only half the number of steps, is by no means ‘half-broken’.

3 Recent Results

3.1 SHA

The first public analysis of a hash function similar to SHA-1, was the attack on SHA presented by Chabaud and Joux at the CRYPTO 1998 conference in Santa Barbara (CA) [4]. The attack used so-called *perturbations* and *corrections*, based on differential cryptanalysis. It had an expected workload of 2^{61}. It was not implemented in practice, but since the expected workload was below 2^{80}, which is the complexity of the square-root attack, this result constituted an academical attack.

The results on SHA were subsequently improved. In 2004, Biham and Chen presented an algorithm to produce near-collisions [2]. In 2005 Biham *et al.* presented optimizations to the attack [3], but the main improvement came from Wang *et al.*, who presented an attack with a complexity of 2^{39} operations [38]. In 2006, Naito *et al.* improved this even further to 2^{36} operations [24].

3.2 SHA-1

Early 2005, Rijmen and Oswald presented an academical attack on SHA-1 reduced to 53 steps (instead of the 80 steps specified in the standard) [30]. In August 2005, Wang *et al.* presented the first collision attack on full SHA-1, with an estimated complexity of 2^{69} [37]. This attack was implemented on a reduced variant with only 58 out of 80 steps. For this variant, the attack had a complexity of 2^{33}. Related results as well as an improved estimate of the complexity of the attack were presented in [21, 28].

In October 2005, Wang presented a further improvement upon her original attack, reducing the estimated complexity of the attack to 2^{63} [36].

In December 2006, De Cannière and Rechberger presented the best known collision example to date: a colliding message pair for 64 out of 80 steps of SHA-1 [8]. The result was made possible by generalizing differential cryptanalysis and by using an advanced searching tool. The attack needs about 2^{35} operations to find the colliding message pair. Estimating the complexity of this attack when applied to the full SHA-1 is currently not possible.

3.3 SHA-224 and SHA-256

Several teams of researchers have applied the recently developed attack methods to reduced or simplified variants of SHA-224 and SHA-256 [12, 14, 15, 19, 22]. These algorithms have a much more complicated structure than SHA-1, both in the message expansion and in the state update transformation, which makes the analysis tedious and involved.

The shift operations in the message expansion (6) severely limit the usefulness of the perturbation-correction approach. It is still possible to find low-weight difference patterns that may result in a collision, but the search space increases dramatically. The presence of two nonlinear Boolean functions in the step update transformation is a very efficient means to counter methods based on differential cryptanalysis. This implies again more work to perform the analysis and to find the weakest spots in the algorithm.

Currently the best result is a method to find collisions for SHA-224 reduced to 19 out of 64 steps [22]. These collisions would also be near-collisions for SHA-256 reduced to 19 out of 64 steps. In the same article, a method to find pseudo-collisions is given, illustrated by an example for SHA-256 reduced to 22 steps. Note, that the mentioned results do not imply any weaknesses for SHA-224 and SHA-256 since the analysis has been performed for a very small number of steps (cf. Section 2.6). Nevertheless, further research is required to make accurate statements on the security margins of these hash functions.

3.4 RIPEMD-160

Since the design of RIPEMD-160 is similar to the design of its predecessor RIPEMD, and also MD5 and SHA-1, one could fear that it will succumb to the same attacks. However, a recently published study indicates that RIPEMD-160 is not vulnerable to the recently developed attacks [20].

RIPEMD-160 has a much more complicated structure than its predecessor RIPEMD, which makes the analysis more difficult. Especially, the increased number of rounds and the different design of the two streams make it much harder to find characteristics with high probability in RIPEMD-160.

Currently the best result on RIPEMD-160 is an academical attack on a simplified variant of the hash function, where the rotation of state variable C in the step function is removed and where the number of steps is reduced from 80 to 48. However, this does not imply any weaknesses for RIPEMD-160 since the analysis has been performed for a simplified variant of the hash function. Nevertheless, further analysis is required to get a good view on the security margins of RIPEMD-160.

3.5 Conclusions

We may expect that the first collision for SHA-1 will be found sometime between the summer of 2007 and the end of 2008.

4 Impact of collisions on digital signatures

4.1 The attack

Digital signatures are currently the only technical process that can result in advanced electronic signatures. All currently known digital signature schemes use hash functions in order to preprocess the input before the ‘raw’ signing primitive is applied.

A collision attack on a hash function can hence be used to mount attacks like the following. Firstly, the attacker constructs two messages x and y such that $h(x) = h(y)$. A digital signature on document x will now also be a correct signature for y and vice versa. Subsequently, there are two scenarios:

Forgery: The attacker convinces the victim to sign message x. Then she can forge a signature on y by simply copying the signature from the message x to the message y.

Repudiation: The attacker signs x and sends it to the victim. Later on, the attacker can deny the signature on x by showing the colliding message y and claiming that she is the victim of a forgery attack.

4.2 Limitations

It is important to realize that the attack only works if the attacker can control both messages x and y. For existing signatures, one of the messages has already been fixed. Finding a suitable y for a given x is called a *second preimage attack*, which is believed to be much more difficult than the collision attack. There are currently no second preimage attacks on the hash functions of the SHA family and RIPEMD-160. This is a first limitation, which applies to all collision attacks.

There are two more limitations, which follow from the particular way in which the current attacks operate (differential cryptanalysis). Firstly, the difference between the two messages influences the workload of the attack. In the basic form of the attack, the two messages x and y have to differ in exactly two blocks at consecutive positions. Furthermore, the lowest workload can be achieved only if the difference between the message blocks takes one specific value or a very limited set of specific values. Secondly, the attack fixes large parts of the two differing message blocks to 'meaningless values.' These limitations have sometimes led to the false belief that the impact of the current results on practical applications is small.

4.3 Circumventing the limitations

Shortly after the first results on the collision attacks were published, it was already demonstrated how to exploit them in highly redundant file formats like PostScript documents [7] and executable files [16, 23, 33]. Basically, the constrained parts of the message can be hidden in areas of high redundancy. In [18] Lenstra and de Weger explain how to produce forged X.509 certificates (based on the hash function MD5) by crafting public keys that result in a collision.

Stevens *et al.* improve on these results and present a method to produce what they call *target collisions* [35]. In brief, given arbitrary messages x and y, the method allows to construct two message 'tails' s, t such that the concatenation of x and s collides with the concatenation of y and t:

$$h(x\|s) = h(y\|t). \tag{10}$$

They illustrate the improved method by presenting two X.509 certificates with different subject common name and different public key moduli. Roughly speaking, x and y contain meaningful content and s and t are valid RSA public key moduli. For this example, they use a tail of only 8 message blocks (4 kbyte).

Finally, the methods explained in [9] can be used to significantly reduce the number of 'meaningless bits' in the colliding messages.

4.4 Conclusion

Practical applications are endangered by collisions only if some additional constraints are satisfied. These additional constraints increase the workload to construct exploitable collisions. However, this increase in workload is getting smaller and the difference in workload may quickly become insignificant.

Collisions will always remain less useful to attackers than second preimages. Collision attacks don't endanger the integrity of signatures existing prior to the development of the attacks.

5 Impact of collisions on HMAC

Hash functions are used in MAC (Message Authentication Code) constructions such as HMAC [1]. This construction is provably secure under certain assumptions on the security of the underlying hash function. It is therefore natural to wonder whether the recent collision attacks on hash functions impact the security of HMAC based on these hash functions.

Recent work shows that HMAC constructions based on SHA or reduced variants of SHA-1 (61 out of 80 steps) have at least theoretical weaknesses [5, 17, 29]. For full HMAC-SHA-1 no weaknesses have been reported to date.

6 Possibility to continue the use of SHA-1

It is clear from the previous sections that collisions for SHA-1 will be found in the near future. This will make SHA-1 unfit for use in the current methods to create advanced electronic signatures. In this section, we first discuss a method that has been proposed recently and that may allow the continued use of SHA-1 in the creation of advanced electronic signatures. Next, we discuss other scenarios where SHA-1 still can be used. We conclude with a classical argument against the further use of SHA-1.

6.1 Randomized hashing

In order to allow continued use of hash functions for which collision attacks have been demonstrated, Halevi and Krawczyk propose a strengthened *mode of operation* for hash functions [13]. By appending a (pseudo-)random value to the message right before the hashing operation is performed, the probability that an attacker can use a collision to create a forgery, is greatly reduced. Halevi and Krawczyk argue that this strategy also restores the non-repudiation property. Adapting an application to this new mode of operation seems however more complicated than replacing the broken hash function, if a replacement hash function is available.

6.2 When collisions don't matter

Collisions are less of a concern in applications that don't have the requirement to use advanced electronic signatures. For instance, communication protocols like SSL can still use SHA-1 if the legitimate parties trust one another.

Since the collision attack requires control of the document that will be signed electronically (hashed), the party creating the document doesn't need to worry about collision attacks. Possible danger exists only for parties who:

1. sign a document created by or under control of another party, or
2. rely on a document created and signed by another party.

For example, assume an e-Government application where the citizens produce a document, sign it electronically and submit the signed document to some authority. A collision attack would allow the citizens to deny having signed the document as received by the authority. Hence, the risk is on the side of the authority.

6.3 Security margin arguments

Some researchers believe in a theory of graceful degradation of hash functions: hash functions would evolve over time from 'very secure' to 'completely insecure' over several intermediate stages. Under this assumption, a hash function for which collisions can be found, is more likely to succumb to a second preimage attack in the near future.

References

[1] Mihir Bellare, Ran Canetti, Hugo Krawczyk, "Keying hash functions for message authentication," *Advances in Cryptology - CRYPTO '96*, *LNCS 1109*, N. Koblitz, Ed., Springer-Verlag, 1996, pp. 1–15.

[2] Eli Biham, Rafi Chen, "Near-Collisions of SHA-0," *Advances in Cryptology - CRYPTO 2004*, *LNCS 3152*, M. K. Franklin, Ed., Springer-Verlag, 2004, pp. 290–305.

[3] Eli Biham, Rafi Chen, Antoine Joux, Patrick Carribault, Christophe Lemuet, William Jalby, "Collisions of SHA-0 and Reduced SHA-1," *Advances in Cryptology - EUROCRYPT 2005*, *LNCS 3494*, R. Cramer, Ed., Springer-Verlag, 2005, pp. 36–57.

[4] Florent Chabaud, Antoine Joux, "Differential Collisions in SHA-0," *Advances in Cryptology - CRYPTO '98*, *LNCS 1462*, H. Krawczyk, Ed., Springer-Verlag, 1998, pp. 56–71.

[5] Scott Contini, Yiqun Lisa Yin, "Forgery and Partial Key-Recovery Attacks on HMAC and NMAC Using Hash Collisions,“ *Advances in Cryptology - ASIACRYPT 2006*, *LNCS 4284*, X. Lai, K. Chen, Eds., Springer-Verlag, 2006, pp. 37–53.

[6] Wei Dai, "Crypto++ 5.2.1 benchmarks," `www.cryptopp.com/amd64-benchmarks.html`.

[7] Magnus Daum, Stefan Lucks, "Attacking hash hunctions by poisoned messages, The story of Alice and her boss." `http://www.cits.rub.de/MD5Collisions/`.

[8] Christophe De Cannière, Christian Rechberger, "Finding SHA-1 Characteristics: General Results and Applications," *Advances in Cryptology - ASIACRYPT 2006*, *LNCS 4248*, X. Lai, K. Chen, Eds., Springer-Verlag, 2006, pp. 1–20.

[9] Christophe De Cannière, Christian Rechberger, "SHA-1 collisions: Partial meaningful at no extra cost?," August 2006. Presented at rump session of CRYPTO 2006.

[10] Hans Dobbertin, Antoon Bosselaers, Bart Preneel, "RIPEMD-160: A Strengthened Version of RIPEMD," *Fast Software Encryption - FSE '96*, *LNCS 1039*, D. Gollmann, Ed., Springer-Verlag, 1996, pp. 71–82.

[11] European Telecommunications Standards Institute (ETSI), *ETSI TS 102 176-1 V1.2.1 (2005-07): Electronic Signatures and Infrastructures (ESI); Algorithms and Parameters for Secure Electronic Signatures; Part 1: Hash functions and asymmetric algorithms*

[12] Henri Gilbert, Helena Handschuh, "Security analysis of SHA-256 and sisters," *Selected Areas in Cryptography - SAC 2003*, *LNCS 3006*, M. Matsui, R. Zuccherato, Eds., Springer-Verlag, 2003, pp. 175–193.

[13] Shai Halevi, Hugo Krawczyk, "Strengthening digital signatures via randomized hashinh," Presented at the Cryptographic Hash Workshop hosted by NIST, October 2005. `http://www.ee.technion.ac.il/~hugo/rhash.pdf`.

[14] Philip Hawkes, Michael Paddon, Gregory G. Rose, "On corrective patterns for the SHA-2 family," Cryptology ePrint Archive, Report 2004/207, August 2004. `http://eprint.iacr.org/`.

[15] Hirotaka Yoshida, Alex Biryukov, "Analysis of a SHA-256 Variant," *Selected Areas in Cryptography - SAC 2005*, *LNCS 3897*, B. Preneel, S. E. Tavares, Eds., Springer-Verlag, 2006, pp. 245–260.

[16] Dan Kaminski, "MD5 considered to be harmful someday," Cryptology ePrint Archive, Report 2004/357, 2004, `http://eprint.iacr.org/`.

[17] Jongsung Kim, Alex Biryukov, Bart Preneel, Seokhie Hong, "On the Security of HMAC and NMAC Based on HAVAL, MD4, MD5, SHA-0 and SHA-1 (extended abstract)," *Security in Information Networks - SCN 2006*, *LNCS 4116*, R. De Prisco, M. Yung, Eds., Springer-Verlag, 2006, pp. 242–256.

[18] Arjen K. Lenstra, Benne de Weger, "On the possibility of constructing meaningful hash collisions for public keys," `http://www.win.tue.nl/~bdeweger/CollidingCertificates/ddl-full.pdf`.

[19] Krystian Matusiewicz, Josef Pieprzyk, Norbert Pramstaller, Christian Rechberger, Vincent Rijmen, "Analysis of simplified variants of SHA-256," *WEWoRC 2005*, *LNI P-74*, C. Wolf, S. Lucks, P. W. Yau, Eds., Gesellschaft fü Informatik, 2005, pp. 123-134.

[20] Florian Mendel, Norbert Pramstaller, Christian Rechberger, Vincent Rijmen, "On the Collision Resistance of RIPEMD-160," *Information Security - ISC 2006*, *LNCS 4176*, S. K. Katsikas, J. Lopez, M. Backes, S. Gritzalis, B. Preneel, Eds., Springer-Verlag, 2006, pp. 101–116.

[21] Florian Mendel, Norbert Pramstaller, Christian Rechberger, Vincent Rijmen, "The Impact of Carries on the Complexity of Collision Attacks on SHA-1," *Fast Software Encryption - FSE 2006*, *LNCS 4047*, M. Robshaw, Ed., Springer-Verlag, 2006, pp. 278–292.

[22] Florian Mendel, Norbert Pramstaller, Christian Rechberger, Vincent Rijmen, "Analysis of Step-Reduced SHA-256," *Fast Software Encryption - FSE 2006*, *LNCS 4047*, M. Robshaw, Ed., Springer-Verlag, 2006, pp. 126–143.

[23] Ondrej Mikle, "Practical attacks on digital signatures using MD5 message digest," Cryptology ePrint Archive, Report 2004/356, 2004, `http://eprint.iacr.org/`.

[24] Yusuke Naito, Yu Sasaki, Takeshi Shimoyama, Jun Yajima, Noboru Kunihiro, Kazuo Ohta, "Improved collision search for SHA-0," *Advance in Cryptology - ASIACRYPT 2006*, *LNCS 4284*, X. Lai, K. Chen, Eds., Springer-Verlag, 2006, pp. 21–36.

[25] National Institute of Standards and Technology (NIST). "FIPS-46-3: Data Encryption Standard," October 1999. `http://www.itl.nist.gov/fipspubs/`.

[26] National Institute of Standards and Technology (NIST), "FIPS Publication 180-1: Secure Hash Standard," April 17, 1995. Available via `www.itl.nist.gov/fipspubs/fip180-1.htm`

[27] National Institute of Standards and Technology (NIST), "FIPS Publication 180-2: Secure Hash Standard," August 1, 2002. `csrc.nist.gov/publications/fips/fips180-2/fips180-2withchangenotice.pdf`

[28] Norbert Pramstaller, Christian Rechberger, Vincent Rijmen, "Exploiting Coding Theory for Collision Attacks on SHA-1," *Cryptography and Coding*, *LNCS 3796*, N. P. Smart, Ed., Springer-Verlag, 2005, pp. 78–95.

[29] Christian Rechberger, Vincent Rijmen, "On Authentication Using HMAC and Non-Random Properties," *Financial Cryptography 2007*, *LNCS*, Springer, to appear.

[30] Vincent Rijmen, Elisabeth Oswald, "Update on SHA-1," *Topics in Cryptology - CT-RSA 2005*, *LNCS 3376*, A. Menezes, Ed., Springer-Verlag, 2005, pp. 58–71.

[31] Ronald L. Rivest, "The MD4 Message Digest Algorithm. RFC 1320", `http://www.faqs.org/rfcs/rfc1320.html`.

[32] Ronald L. Rivest, "The MD5 Message Digest Algorithm. RFC 1321," `http://www.faqs.org/rfcs/rfc1321.html`.

[33] Peter Selinger, "MD collision demo", `http://www.mscs.dal.ca/~selinger/md5collision/`.

[34] Bruce Schneier, John Kelsey, "Unbalanced Feistel Networks and Block Cipher Design," *Fast Software Encryption - FSE '96*, *LNCS 1039*, D. Gollmann, Ed., Springer-Verlag, 1996, pp. 121–144.

[35] Marc Stevens, Arjen K. Lenstra, Benne de Weger, "Target Collisions for MD5 and Colliding X.509 Certificates for Different Identities," *Advances in Cryptology - EUROCRYPT 2007*, *LNCS*, Springer, 2007, to appear.

[36] Xiaoyun Wang, Andrew Yao, Frances Yao, "Cryptanalysis of SHA-1," Presented at the Cryptographic Hash Workshop hosted by NIST, October 2005.

[37] Xiaoyun Wang, Yiqun Lisa Yin, Hongbo Yu, "Finding Collisions in the Full SHA-1," *Advances in Cryptology - CRYPTO 2005*, *LNCS 3621*, V. Shoup, Ed., Springer-Verlag, 2005, pp. 17–36.

[38] Xiaoyun Wang, Hongbo Yu, Yiqun Lisa Yin, "Efficient Collision Search Attacks on SHA-0," *Advances in Cryptology - CRYPTO 2005*, *LNCS 3621*, V. Shoup, Ed., Springer-Verlag, 2005, pp. 1–16.