# A Framework for Iterative Hash Functions — HAIFA*

Eli Biham† Orr Dunkelman†
Computer Science Department, Technion.
Haifa 32000, Israel.
{biham,orrd}@cs.technion.ac.il

**Abstract**

For years hash functions were built from compression functions using the Merkle-Damgård construction. Recently, several flaws in this construction were identified, allowing for pre-image attacks and second pre-image attacks on such hash functions even when the underlying compression functions are secure.

In this paper we propose the HAsh Iterative FrAmework (HAIFA). Our framework can fix many of the found flaws while supporting several additional properties such as randomized hashing and variable hash size. HAIFA allows for an online computation of the hash function in one pass with a fixed amount of memory independent of the number of blocks in the message.

## 1 Introduction

Recall the three requirements from a cryptographic compression function or a hash function $f(\cdot)$:

1. Pre-image resistance: Given $y = f(x)$ it is hard to find $x'$ s.t. $f(x') = y$.
2. Second pre-image resistance: Given $x$ it is hard to find $x'$ s.t. $f(x) = f(x')$.
3. Collision resistance: It is hard to find $x, x'$ s.t. $f(x) = f'(x)$.

The Merkle-Damgård construction is the most widely used transformation of cryptographic secure compression functions into cryptographic hash functions [4, 13]. The Merkle-Damgård construction suggests a simple transformation that maintains the collision resistance property of the underlying compression function. For years it was widely believed that the transformation maintains the pre-image resistance of the underlying compression function as well as the second pre-image resistance.

However, in recent years several counter examples for these beliefs were suggested. The first evidence for this was the works of Dean [5]. Dean showed that if fix-points of the compression function are easily found, then second pre-image attacks on Merkle-Damgård hash functions can be mounted using $O(m \cdot 2^{m/2})$ time and $O(m \cdot 2^{m/2})$ memory. This attack is achieved using expandable messages, i.e., messages that can be expanded without changing the chaining value. Later, Kelsey and Schneier have proposed the same ideas, while removing the assumption that fix-points can be easily found [10]. This improvement was achieved using Joux's ideas for efficiently finding multi-collisions in an iterative hash functions [8]. It is worth mentioning that the multi-collision attack shows that the strength of $h(x) = h_1(x)||h_2(x)$ for two independent iterative hash functions $h_1(\cdot)$ and $h_2(\cdot)$ is only as secure as the more secure of the two functions (up to some small factor).

*An initial version of this work was presented in the hash function workshop in Krakow, June 2005.

†This work was supported in part by the Israel MOD Research and Technology Unit.

The main pitfall of these attacks is the fact that the messages that collide are relatively long. In [9], Kelsey and Kohno showed that using a simple precomputation, it is possible to reduce the time requirements of pre-image attacks (in some sense) of relatively short messages, while keeping the time complexity of the attack to be much below $O(2^m)$.

In this work we suggest the HAsh Iterative FrAmework (HAIFA) to replace the Merkle-Damgård construction. HAIFA maintains the good properties of the Merkle-Damgård construction while adding to the security of the transformation, as well as to the scalability of the transformation.

HAIFA has several attractive properties: simplicity, maintaining the collision resistance of the compressions function, increasing the security of iterative hash functions against (second) pre-image attacks, and the prevention of easy-to-use fix-points of the compression function. HAIFA also supports variable hash size and randomized hashing [6] as part of the framework. HAIFA also posses the online hashing property of the Merkle-Damgård construction. The computation of an HAIFA hash function requires one pass on the message, without keeping the entire message in memory, and while using a fixed amount of memory for the hashing of each block.

This paper is organized as follows: In Section 2 we describe the Merkle-Damgård construction and various results against the construction. In Section 3 we propose HAIFA. We discuss the security aspects of HAIFA in Section 4. In Section 5 we briefly discuss the suggestion to iterate and expand the message before applying the hash function as a remedy to the Merkle-Damgård construction. Finally, Section 6 summarizes the paper.

# 2 The Merkle-Damgård Constructions and its Pitfalls

The Merkle-Damgård construction is a simple and elegant way to transform a compression function $C_{MD} : \{0,1\}^{m_c} \times \{0,1\}^n \rightarrow \{0,1\}^{m_c}$ into a hash function [4, 13]. Throughout this paper $m_c$ denotes the size of the chaining value, and $n$ denotes the block size for the compression function. We also denote by $m$ the hash output length (in many cases $m = m_c$).

The message $M$ is padded with its length (after additional padding to make the message a multiple of the block size $n$ after the final padding), and the message is divided into blocks of $n$ bits each, $M = M_1 M_2 M_3 \dots M_k$. An initial chaining value $h_0 = IV \in \{0,1\}^{m_c}$ is set for the hash function (also called initialization vector) and the following process is repeated $k$ times:

$$h_i = C_{MD}(h_{i-1}, M_i)$$

The final $h_k$ is outputed as the hash value.

It is easy to prove that once a collision of $h$ is found, then a collision of the compression function $C_{MD}$ is found as well [4, 13]. Thus, the Merkle-Damgård construction retains the collision resistance of the compression function.

When from $h_i = C_{MD}(h_{i-1}, M_i)$ and $M_i$ the value of $h_{i-1}$ can be easily computed, a pre-image attack on $h$ can be mounted using a birthday attack [22]. However, the opposite statement is not true. Even if an inversion attack on $C_{MD}$ requires $O(2^m)$ operations, the security claims for $h$ cannot offer security better than $O(2^{m/2})$. This surprising property was first noted by Dean [5], which went unnoticed until rediscovered (and expanded) by Kelsey and Schneier [10].

## 2.1 Fixed Points, Expandable Messages, and Finding Second Pre-Images

It was widely believed by the cryptographic community that the security proof of the Merkle-Damgård construction applies also to second pre-image attacks. However, Dean [5] noticed that this is not true for long messages if the compression function has an easy to find fix-points. His observations were later generalized by Kelsey and Schneier that used the multi-collision technique to replace the need of easily found fix-points [10].

Let us consider a long message $M = M_1 M_2 \dots M_l$ that is processed using $h$, a Merkle-Damgård hash function, when the message length is not padded (the Merkle-Damgård strengthening). An attacker

that wishes to construct a message $M^*$ such that $h(M) = h(M^*)$ can randomly select messages $M'$ until $h(M')$ equals to any of the $l$ chaining values found during the computation of $h(M)$. Once such an instance is found, the attacker can concatenate to $M'$ the message blocks of $M$ that are hashed starting from the given chaining value, resulting in $M^*$ s.t. $h(M) = h(M^*)$. This attack is foiled by the Merkle-Damgård strengthing, as the message length is padded, changing the last block that enters the compression function.

Assume that the compression function $C_{MD}$ is such that finding fix-points is easy, i.e., it is easy to find $h, M$ s.t. $h = C_{MD}(h, M)$. This is the case for the Davies-Meyer construction that takes a block cipher $E$ that accept $m_c$-bit plaintexts and $n$-bit keys and sets

$$h_i = C_{MD}(h_{i-1}, M) = E_M(h_{i-1}) \oplus h_{i-1}.$$

For such a compression function it is easy to find fix-points by computing $h = E_M^{-1}(0)$ for randomly selected messages $M$.

Dean uses these fix-points to bypass the Merkle-Damgård strengthing. His attack has three main steps:

1. Finding $O(2^{m_c/2})$ fix-points denoted by $A = (h, m)$.
2. Selecting $O(2^{m_c/2})$ one message block, and computing their chaining value denoted by $B = C_{MD}(IV, \tilde{m})$.
3. Once a collision between a chaining value and a fixed point, i.e., between chaining values in $A$ and in $B$, is found, the previous attack is repeated (i.e., trying to add blocks that cause the same chaining values as the original message).

Once such a message is found, it is easy to expand the number of blocks in the message to the appropriate length by repeating the fix-points any times as needed.

Kelsey and Schneier transformed the attack to the case were fix-points are not easily found. While Dean's expandable message could be extended by a concatenation of one message block, in their attack they use the multi-collision technique to produce an expandable message. They replace the first two steps in Dean's attack in the following procedure. In each iteration $1 \leq i \leq t$ of the procedure a collision between a one block message and a $2^{i-1}+1$ block message is found. This procedure finds a chaining value that can be reached by messages of lengths between $t$ and $2^{t+1} + t - 1$ blocks. Then, from this chaining value the third step of Dean's attack is executed, and the length of the found message is controlled by the expandable prefix.

## 2.2 Multi-Collisions in Iterative Hash Functions

Joux identified the fact that when iterative hash functions are used, finding multi-collisions, i.e., a set of messages with the same hash value, is almost as easy as finding a single collision [8]. His main observation is the fact that in iterative hashing schemes, such as the Merkle-Damgård, it is possible to find a collision for each block, e.g., for any $h_{i-1}$ finding $M_i$ and $M_i^*$ such that $C_{MD}(h_{i-1}, M_i) = C_{MD}(h_{i-1}, M_i^*)$. Finding $t$ such one block collision (each starting from the chaining value produced by the previous block collision) it is possible to construct $2^t$ messages with the same hash value by selecting for $i$th block of the message either $M_i$ or $M_i^*$.

Joux's observation leads to the conclusion that concatenation of two hash functions, i.e., $h(x) = h_1(x)||h_2(x)$, is secure against collision attacks just like the stronger of the two underlying hash functions. Moreover, concatenation of several iterative hash functions is as secure as the stronger of the hash functions (up to some a factor of $m^{k-1}$, where $k$ is the number of hash functions).

It is worth mentioning that using fix-points of several blocks, Joux proved that the concatenation of hash functions is as secure against pre-image attacks as the strongest of all the hash functions. These results have disproved several widely believed assumptions on the behavior of hash functions.

### 2.3 Herding Iterative Hash Functions

Kelsey and Kohno have observed that it is possible to perform a time-memory tradeoff for several instances of pre-image attacks [9]. In their attack, an attacker commits to a public digest value that corresponds to some meaningful message, e.g., prediction of the results of the best paper of the EUROCRYPT 2006 conference. After the announcement of the result, the attacker publishes a message that has the pre-published digest value and contains the correct information along with some suffix.

The attack is based on selecting the digest value carefully, helping the attacker to perform a pre-image attack on this value. The main idea behind this attack is to start with $2^t$ possible chaining values $h_i$. From each such chaining value, $O(2^{m_c/2})$ one block message are hashed, resulting in collisions between pairs of chaining values and messages $(h_i, m_j)$. The attacker continues iteratively to find collisions between the new chaining values, until the attacker finds the final collision. The attacker then publishes the last chaining value as a target. When trying to find a pre-image to this target, the attacker needs to perform only $O(2^{m_c-t})$ operations until finding a message that has a chaining value among the $2^t$ original values.

We note that unlike the previous attacks that require long messages, this attack appends relatively short suffix (about $t$ blocks) to the "real" message. We also note, that the total time complexity of the attack is about $O(2^{m_c/2+t})$ offline operations for the first step of the attack and $O(2^{m_c-t})$ online operations for the second step. For $t = m_c/4$ the overall time complexity of this attack is $O(2^{3m_c/4})$ for finding a pre-image.

## 3 The HAsh Iterative FrAmework

We propose the HAsh Iterative FrAmework to solve many of the pitfalls of the Merkle-Damgård construction. The main ideas behind HAIFA are the introduction of number of bits that were hashed so far and a salt value into the compression functions. Formally, instead of using a compression function $C_{MD} : \{0,1\}^{m_c} \times \{0,1\}^n \rightarrow \{0,1\}^{m_c}$, we propose to use $C : \{0,1\}^{m_c} \times \{0,1\}^n \times \{0,1\}^b \times \{0,1\}^s \rightarrow \{0,1\}^{m_c}$, i.e., in HAIFA the chaining value $h_i$ is computed as

$$h_i = C(h_{i-1}, M_i, \#bits, salt),$$

where $\#bits$ is the number of bits hashed so far and $salt$ is a salt value.

### 3.1 Number of Bits Hashed so Far

The inclusion of the number of bits hashed so far was suggested (with small variants) in order to prevent the easy exploitation of fix-points. The attacker is forced to work harder in order to find fix-points. While for $C_{MD}$, once a fix-point is found, i.e., $(h, M)$ such that $h = C_{MD}(h, M)$, it can be used as many times as the attacker sees fit [5, 10]. Even if the compression function does not mix the $\#bits$ parameter well, once an attacker finds a fix-point of the form $(h, M, bits, salt)$ such that $h = C(h, M, bits, salt)$, she cannot concatenate it to itself as many times as she wishes because the number of bits hashed so far has changed.

We note that it is possible to add the number of blocks that were treated so far. However, current schemes keep track of the number of bits hashed so far (for the padding), and not the number of blocks. Thus, it is easier for implementations to consider only one parameter (number of bits) rather two (somewhat related) parameters (number of bits and number of blocks).

It is interesting to consider message authentication codes based on the following HAIFA hash function $h(\cdot)$: $MAC_k(M) = h(k, M)$. While for a Merkle-Damgård construction or suggestions that use the number of blocks hashed so far, this suggestion is clearly not secure against message expansion techniques, for HAIFA this construction is secure. The reason for that, is that the last block is compressed with the number of bits that were processed so far. If this value is not a multiple of a block, then the resulting digest does not equal the chaining value that is needed to the expansion of the message. If the message is a multiple of a block, then an additional block

is hashed (with the padding) with the same number of bits that were hashed so far. Thus, the chaining value required for the extension remains obscure to the attacker.

### 3.2 Salt

The *salt* parameter can be considered as defining a family of hash functions as needed by the formal definitions of [16] in order to ensure the security of the family of hash functions. This parameter can be viewed as an instance of the randomized hashing concept, thus, inheriting all the "goodies" such concept provides:

- Ability to define the security of the hash function.
- Transformation of all attacks on the hash function that can use precomputation from an off-line part and an on-line part to only on-line parts (as the exact *salt* is not known in advance).
- Increasing the security of digital signatures, as the signer chooses the *salt* value, and thus, any attack aiming at finding two messages with the same hash value has to take the *salt* into consideration.

### 3.3 Variable Hash Size

Different digest sizes are needed for different applications. This fact has motivated NIST to publish SHA-224 and SHA-384 as truncated variants of the SHA-256 and SHA-512, respectively. We note that these truncated hash functions use the same construction, but with different initial values.

Our framework supports truncation that allows arbitrary digest sizes (up to the output size of the compression function), while securing the construction against attacks that try to find two messages that have similar digest values. This problem eliminates the easy solution of just taking the number of output bits from the output of the compression function.

Let $IV$ be an initial value chosen by the designer of the compression function, and let $m$ be the required length of the output. For producing hash values of $m$ bits the following initial value is computed

$$IV_m = C(m, IV, 0, 0).$$

Then, the hash value is computed starting from the initial value $IV_m$. After the final block is processed, $m$ bits are taken from the output of the compression function as the hash value. This approach is somewhat equivalent to adding the length $m$ of the digest length at the beginning of the message and then truncating the output to the required size.

The padding scheme used in HAIFA is very similar to the one used in the Merkle-Damgård construction, i.e., the message is padded with 1, as many needed 0's, the length of the message encoded in a fixed number of bits, and the digest size.

This suggestion allows for an easy support in various digest lengths. An implementation of an HAIFA-based hash function requires only the value of $IV$ for the ability to produce any hash length, while in implementations where only one output length $l$ is needed, $IV_l$ can be precomputed and hardcoded into the implementation.

This second padding ensures that even if two messages $M_1$ and $M_2$ are found, such that under $IV_{l_1}$ and $IV_{l_2}$ ($M_1$ hashed to obtain $l_1$ bits and $M_2$ hashed to a digest of $l_2$ bits) their chaining values collide, then the last block changes this behavior. This approach is similar to the one used in the original strengthing, even though it deals with a scenario of two different output sizes.

## 4 The Security of HAIFA Hash Functions

We first note that proving the HAIFA hash function is collision resistant if the underlying compression function is collision resistant is quite easy. The same arguments that are used to prove that the Merkle-Damgård construction retains the collision resistance of the underlying compression function, can be used to prove that HAIFA does so as well.

We note that any iterative construction can be attacked by some of the attacks described in Section 2.

However, as noted before, using our ideas, it is possible to reduce the applicability of these attacks, by preventing an efficient pre-computation that reduces the online computational phase of these attacks.

Let us consider the multi-collision attack. This attack works against all iterative hashing schemes, independent of their structure. While the time complexity for finding collisions for each block is not different in our framework than in the Merkle-Damgård construction, an attacker cannot pre-compute these multi-collisions before the choosing of the salt value.

As noted before, our framework prevents Dean's attack, as it is highly unlikely that some fix-point of the compression function can be found. When considering the constructions proposed in [6] it is evident that this attack still applies to them. Dean's attack can be easily applied to the randomized Merkle-Damgård constructions $H_r(M) = H(M \oplus (r|r|r \ldots |r))$ and $\tilde{H}_r(M) = H_r(0|M)$, as once a fix-point is found, it remains a fix-point for these constructions. We note that the first two steps of Dean's attack can be mounted off-line just as in Dean's attack on regular Merkle-Damgård hash functions for these two constructions.

As for Kelsey and Schneier's attack, our framework again prevents the pre-computation involved in the attack, and transforms the entire attack into an online attack. This follows from the fact, that any iterative construction is susceptible to the multi-collision attack.

Let us consider the herding attack. It can be easily shown that $H_r$ and $\tilde{H}_r$ of [6] are susceptible to these attacks, as the mixing of the random strings into the messages is relatively weak. Thus, an attacker can choose the chaining values at random and find collisions in the underlying compression function. Then, when $r$ is given to the attacker, she can easily compute the messages that produce the desired hash value. Our observations explain why the usage of the Merkle-Damgård construction with randomization cannot solve the problems of the Merkle-Damgård construction.

Under HAIFA such a precomputation is infeasible, as the salt is mixed into the chaining value in a much stronger way. This prevents the herding attack, as the attacker cannot find the exact digest value (unless the salt is known in advance). We note that if a security of $O(2^m)$ against pre-image attacks such as the herding attack is requested, then the size of the salt must be at least $m_c/2$ bits long, in order to prevent the herding attack.

We note that if the length of the salt is short, then an attacker can still use precomputation to overcome the security proposed by HAIFA. It is therefore recommended that the salt length would be of 64 bits at least, or at least $m_c/2$ bits when possible.

It is possible to treat the two parameters *salt* and *#bits* as additional fields in the chaining value and removing them in the last block. The approach of increasing the chaining value was promoted in [11] and it may seem that our suggestion follows this approach. However, the analysis in [11] assumes that the hash function is a "good" hash function for all the bits of the chaining value, while our approach shows that the *salt* and *#bits* parameters can be treated under a relatively weak transformation, without affecting the security of the construction. Thus, it is expected the HAIFA hash functions will be faster than wide hash constructions. We also note that the additional memory required by our suggestion is the size of the salt (which is fixed for all blocks, i.e., can be cached), unlike the larger memory required for keeping a larger internal value.

We conclude that the security of a Merkle-Damgård hash function against a pre-image attacks is equivalent to its security against collision attacks. For HAIFA this is not the case, as we have showed earlier. We give the security level of an ideal hash function and of the Merkle-Damgård and HAIFA constructions (under two cases — with a variable salt, and with a fixed salt) in Table 1.

# 5 A Short Note on Iteration and Expansion of the Message

The recent results on hash functions has motivated a lot of suggestions aiming to fix the flaws that were found. Among the common solutions to the problems of the Merkle-Damgård solution, are solutions

| Type of Attack | Ideal Hash Function | MD | HAIFA fixed salt | HAIFA with (distinct) salts |
|---|---|---|---|---|
| | = | ≥ | ≥ | ≥ |
| Preimage | $2^{m_c}$ | $2^{m_c}$ | $2^{m_c}$ | $2^{m_c}$ |
| One-of-many pre-image ($k'$ targets) | $2^{m_c}/k'$ | $2^{m_c}/k'$ | $2^{m_c}/k'$ | $2^{m_c}$ |
| Second-pre-image ($k$ blocks) | $2^{m_c}$ | $2^{m_c}/k$ | $2^{m_c}$ | $2^{m_c}$ |
| One-of-many second pre-image($k$ blocks in total, $k'$ messages) | $2^{m_c}/k'$ | $2^{m_c}/k$ | $2^{m_c}/k'$ | $2^{m_c}$ |
| Collision | $2^{m_c/2}$ | $2^{m_c/2}$ | $2^{m_c/2}$ | $2^{m_c/2}$ |
| Multi-collision ($k$-collision) | $2^{m_c(k-1)/k}$ | $\lceil \log_2 k \rceil 2^{m_c/2}$ | $\lceil \log_2 k \rceil 2^{m_c/2}$ | $\lceil \log_2 k \rceil 2^{m_c/2}$ |
| Herding [9] | – | Offline: $2^{m_c/2+t}$ Online: $2^{m_c-t}$ | Offline: $2^{m_c/2+t}$ Online: $2^{m_c-t}$ | Offline: $2^{m_c/2+t+s}$ Online: $2^{m_c-t}$ |

The figures are given for MD and HAIFA hash functions that use an ideal compression function.

Table 1: Complexities of Attacks on Merkle-Damgård and HAIFA Hash Functions with Comparison for an Ideal Hash Function

that suggest iterating the message several times during the hash process. For example, instead of computing $h(M)$ it was suggested to compute $h(MM)$ or even $h(M\check{M})$, where $\check{M}$ is $M$ written backwards, i.e., $0\check{1}1 = 110$.

Such suggestions aim at solving both the problems caused by the multi-collision attack, Dean's attack (and its improved version), and the herding attack, as well as the problems identified by the attacks on specific hash functions [18, 19, 20, 21]. Once a message is iterated more than once, it prevents the "natural" flow of all of these attacks.

However, as first noted in [14], and later expanded in [7, 15], any expansion and iteration function, that has a constant rate, can be attacked by variants of the multi-collision attack. Thus, despite their much slower application, and the additional memory requirements, iteration and expansion of the message does not offer a much better security.

# 6 Summary

In this paper we have presented HAIFA as a replacement for the Merkle-Damgård construction. The main differences are the addition of the number of bits hashed so far to the compression function along with a salt value. In cases where there is no need to add salt (e.g., message authentication codes) it is possible to set its value to 0.

We note that even today's compression functions can be used in HAIFA hash functions by changing the API of such compression functions. For example, by setting in SHA-1 64 bits (out of the 512 bits of each block) to represent the number of bits hashed so far, and 64 bits to represent the salt, the new compression function would hash 384 bits per call of the compression function. This increases the computational effort of hashing long messages by a factor of about 4/3, but at the same time provides security against various attacks. New hash functions are expected to mix the salt and the number of bits much more efficiently.

We conclude that new hash functions should be designed under the HAsh Iterative FrAmework.

# References


[1] Eli Biham, Rafi Chen, *Near-Collisions of SHA-0*, Advances in Cryptology, proceedings of CRYPTO 2004, Lecture Notes in Computer Science 3152, pp. 290–305, Springer-Verlag, 2004.

[2] Eli Biham, Rafi Chen, Antoine Joux, Patrick Carribault, Christophe Lemuet, William Jalby, *Collisions of SHA-0 and Reduced SHA-1*, Advances in Cryptology, proceedings of EURO-

CRYPT 2005, Lecture Notes in Computer Science 3621, pp. 36–57, 2005.

[3] Florent Chabaud, Antoine Joux, *Differential Collisions in SHA-0*, Advances in Cryptology, proceedings of CRYPTO 1998, Lecture Notes in Computer Science 1462, pp. 56–71, Springer-Verlag, 1998.

[4] Ivan Damgård, *A Design Principle for Hash Functions*, Advances in Cryptology, proceedings of CRYPTO 1989, Lecture Notes in Computer Science 435, pp. 416–427, Springer-Verlag, 1990.

[5] Richared D. Dean, *Formal Aspects of Mobile Code Security.*, Ph.D. dissertation, Princeton University, 1999.

[6] Shai Halevei, Hugo Krawczyk, *Strengthening Digital Signatures via Randomized Hashing*, preproceedings of Cryptographic Hash Workshop, held in NIST, Gaithersburg, Maryland, 2005. Available on-line at *http://www.ee.technion.ac.il/∼hugo/rhash.pdf*.

[7] Jonathan J. Hoch, Adi Shamir, *Breaking the ICE — Finding Multicollisions in Iterated Concatenated and Expanded (ICE) Hash Functions*, preproceedings of Fast Software Encryption 2006, pp. 199–214, 2006.

[8] Antoine Joux, *Multicollisions in Iterated Hash Functions*, Advances in Cryptology, proceedings of CRYPTO 2004, Lecture Notes in Computer Science 3152, pp. 306–316, Springer-Verlag, 2004.

[9] John Kelsey, Tadayoshi Kohno, *Herding Hash Functions and the Nostradamus Attack*, preproceedings of Cryptographic Hash Workshop, held in NIST, Gaithersburg, Maryland, 2005.

[10] John Kelsey, Bruce Schneier, *Second Preimages on n-Bit Hash Functions for Much Less than* $2^n$, Advances in Cryptology, proceedings of EUROCRYPT 2005, Lecture Notes in Computer Science 3494, pp. 474–490, Springer-Verlag, 2005.

[11] Stefan Lucks, *A Failure-Friendly Design Principle for Hash Functions*, Advances in Cryptology, proceedings of ASIACRYPT 2005, Lecture Notes in Computer Science 3788, pp. 474–494, Springer-Verlag, 2005.

[12] Ralph C. Merkle, *Secrecy, Authentication, and Public Key Systems*, UMI Research press, 1982.

[13] Ralph C. Merkle, *One Way Hash Functions and DES*, Advances in Cryptology, proceedings of CRYPTO 1989, Lecture Notes in Computer Science 435, pp. 428–446, Springer-Verlag, 1990.

[14] Mridul Nandi, Douglas R. Stinson, *Multicollision Attacks on Generalized Hash Functions*, IACR eprint report 2004/.

[15] Mridul Nandi, Douglas R. Stinson, *Multicollision Attacks on some Generalized Sequential Hash Functions*, IACR eprint report 2006/055.

[16] Phillip Rogaway, Thomas Shrimpton, *Cryptographic Hash-Function Basics: Definitions, Implications, and Separations for Preimage Resistance, Second-Preimage Resistance, and Collision Resistance*, proceedings of Fast Software Encryption 2004, Lecture Notes in Computer Science 3017, pp. 371–388, Springer-Verlag, 2004.

[17] US National Bureau of Standards, *Secure Hash Standard*, Federal Information Processing Standards Publications No. 180-2, 2002.

[18] Xiaoyun Wang, Xuejia Lai, Dengguo Feng, Hui Chen, Xiuyuan Yu, *Cryptanalysis of the Hash Functions MD4 and RIPEMD*, Advances in Cryptology, proceedings of EUROCRYPT 2005, Lecture Notes in Computer Science 3494, pp. 1–18, 2005.

[19] Xiaoyun Wang, Yiqun Lisa Yin, Hongbo Yu, *Finding Collisions in the Full SHA-1*, Advances in Cryptology, proceedings of CRYPTO 2005, Lecture Notes in Computer Science 3621, pp. 17–36, 2005.

[20] Xiaoyun Wang, Hongbo Yu, *How to Break MD5 and Other Hash Functions*, Advances in Cryptology, proceedings of EUROCRYPT 2005, Lecture Notes in Computer Science 3494, pp. 19–35, 2005.

[21] Xiaoyun Wang, Hongbo Yu, Yiqun Lisa Yin, *Efficient Collision Search Attacks on SHA-0*, Advances in Cryptology, proceedings of CRYPTO 2005, Lecture Notes in Computer Science 3621, pp. 1–16, 2005.

[22] Gideon Yuval, *How to Swindle Rabin*, Cryptologia, Vol. 3, pp. 187–190, 1979.