NIST Hash Workshop, 10/2005

Enhancing the MD-Strengthening & Designing Scalable 1-Way Hash Families

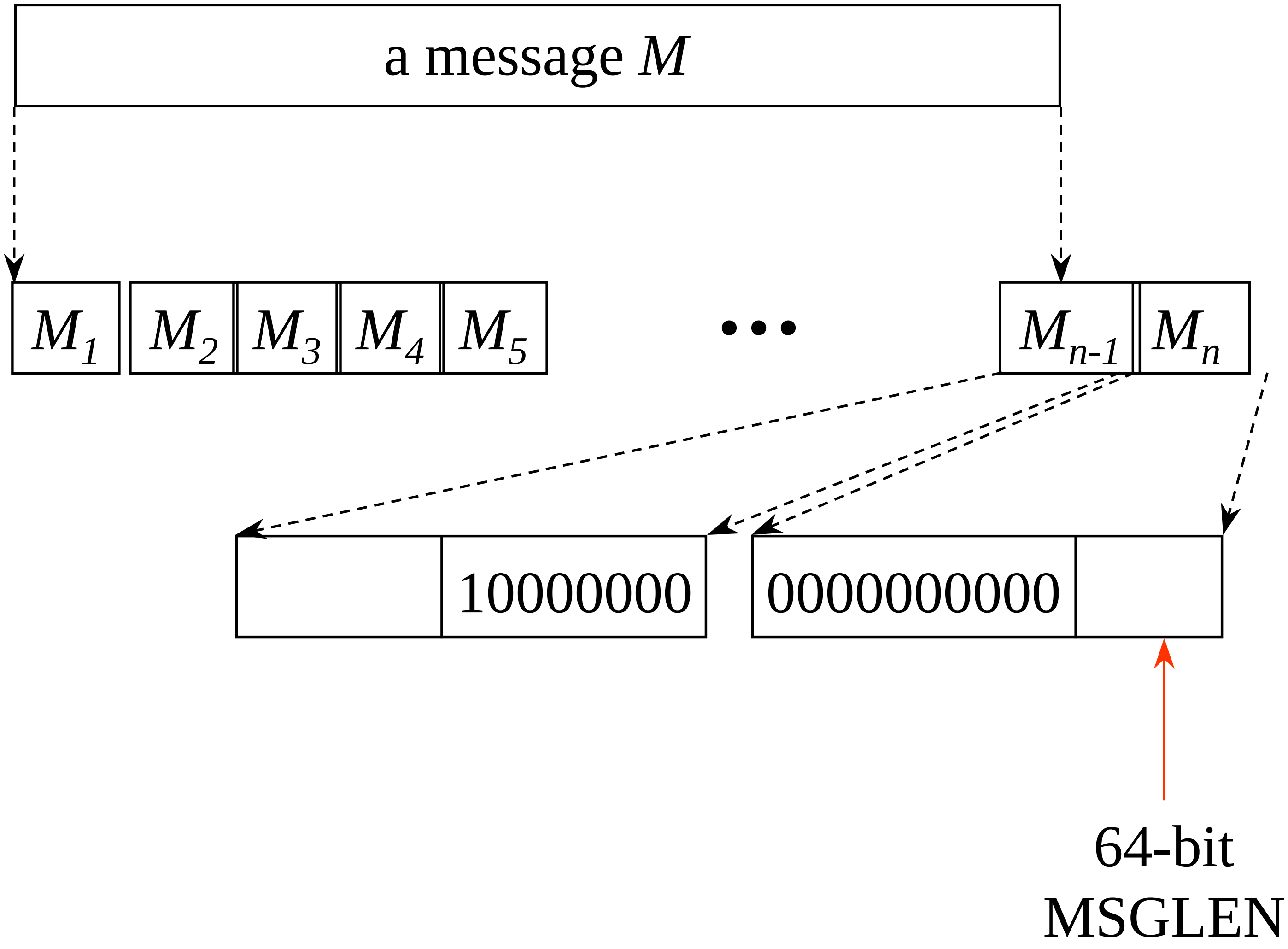
Neil Kauer,	Wachovia Bank neil.kauer@wachovia.com
Tony Suarez,	Wachovia Bank neil.kauer@wachovia.com
Yuliang Zheng,	UNC Charlotte, yzheng@uncc.edu

Outline

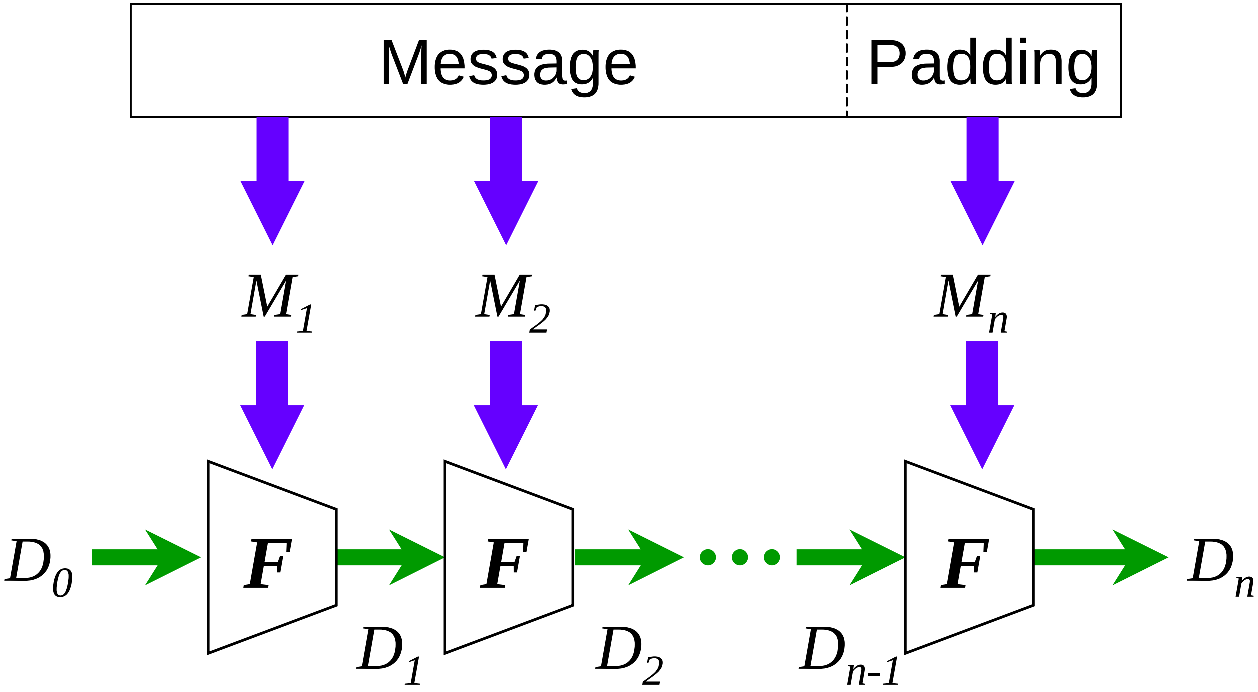
- Tagging messages prior to Damgaard-Merkle hashing
- Risk mitigation centered strategies for designing 1-way hashes

Part 1
Message tagging

MD5/SHA: Breaking up & padding a message *M*



Damgaard-Merkle Strengthening



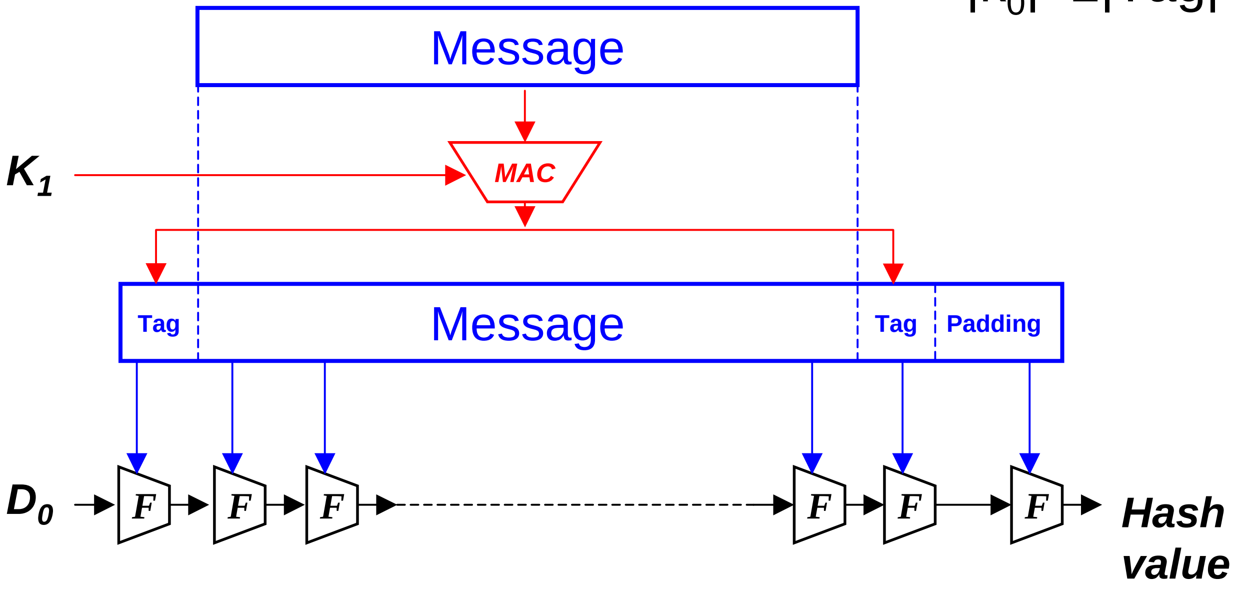
Segments and Blocks

- A message is the concatenation of message segments
  - segments may have the same or different sizes.
- A message segment is composed of message blocks
  - each block representing a basic processing unit of the block compression function of a MD type one-way hash algorithm.

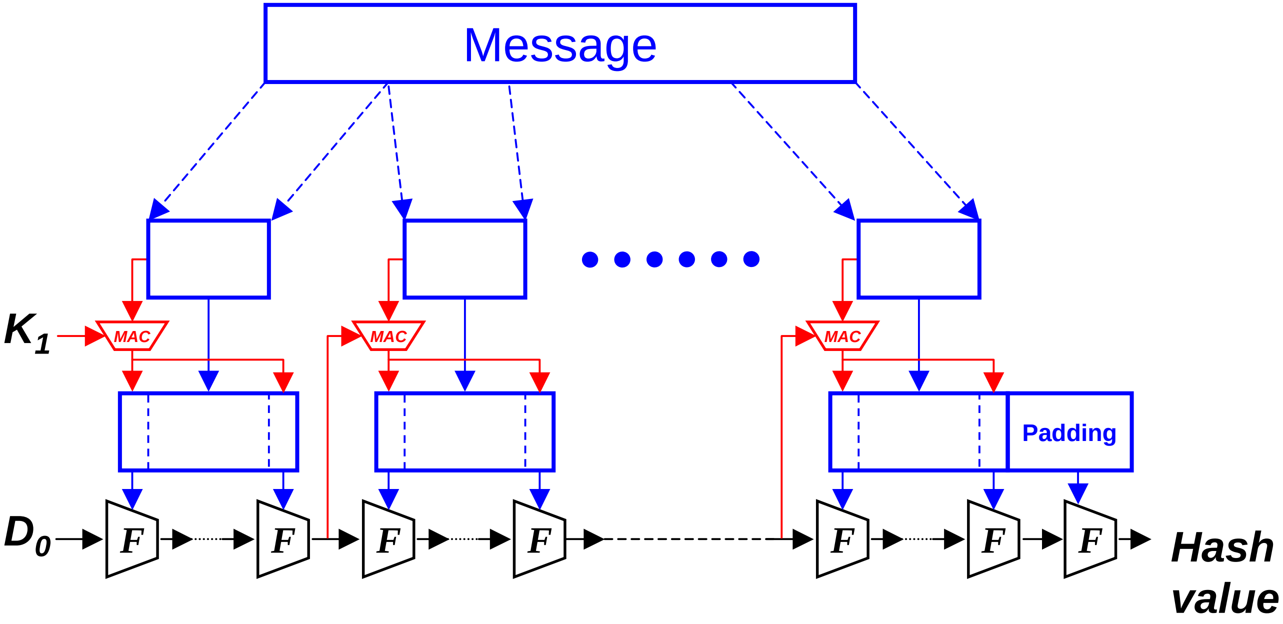
Global Tagging

|Tag| >= 160 bits

$|k_0|=2|\text{Tag}| >= 320$ bits



Local Tagging



How to handle the key K_1

- Does not have to be kept secret
- Can be treated in the same way as constants used in 1-way hashing
  - Pick the first 256 bits of (Sqrt(7) – 2) as the fixed key,
  - Pick it at random and fix it for all applications, or
  - Pick it at random and fix it for certain types of applications/period of time

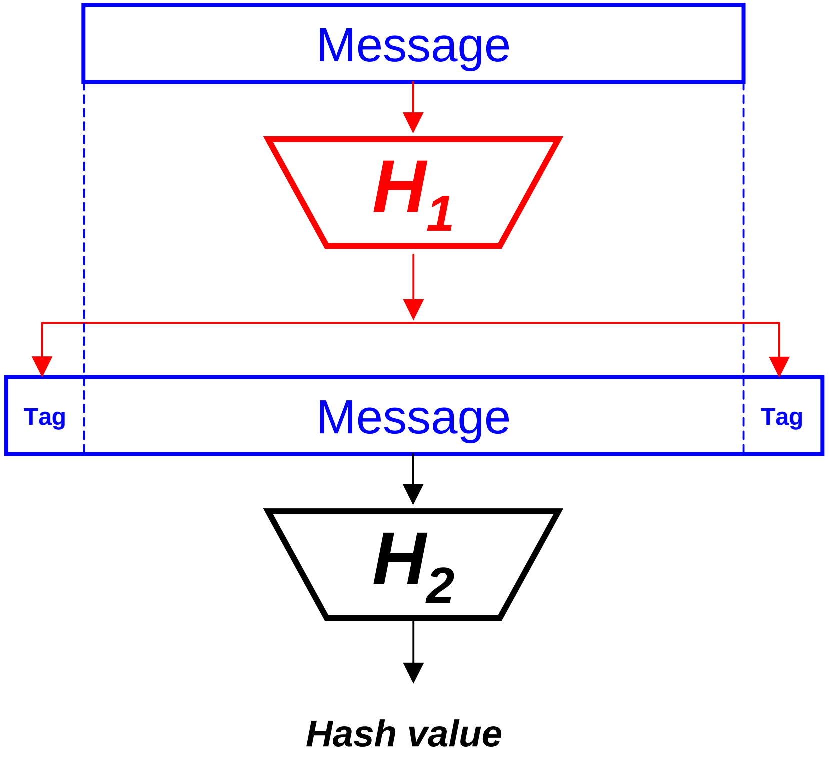
Comparison

- Global tagging
  - Whole message scanned twice
  - Best for
    - Relatively short messages
    - Long messages that allow double scanning/pre-processing
- Local tagging
  - Message scanned twice locally only
  - Best for long (&short) messages

Variants

- Front and back use different tags
  - E.g., Tag & Flipped Tag
- With local tagging
  - Tags are added to every other segment only
    - Improve efficiency
- Sequential composition of hashing algorithms (see next slide)

Sequential Composition of Hashes



Universal Hash ➔ MAC

- Given an “almost strongly universal (ASU)” or “almost XOR universal (AXU)” hash family *F*, then *u(x)* defined by

$$u(x) = \beta + f(x)$$

  is a MAC, where
  - *f* is chosen at random from *F* and,
  - *β* is chosen at random from the set of all possible outcomes of *F*,
  - + is XOR.

Fast MAC from Evaluation Hash

- Let $M = m_1 m_2 \dots m_n$

  $m(x) = m_1 + m_2 x + \dots + m_n x^{n-1}$

  – each m_i is viewed as being in $GF(2^t)$.

- Then the following is a MAC

$$u(M) = \beta + \alpha * m(\alpha)$$

where α and β are from $GF(2^t)$.

Collision Probability of the MAC

- For unknown α and β chosen at random, the probability of finding a second message that collides a known message is approx.

$$n\ 2^{-t}$$

- For $n=2^{32}$ and $t=128$, we have

$$2^{-96}$$

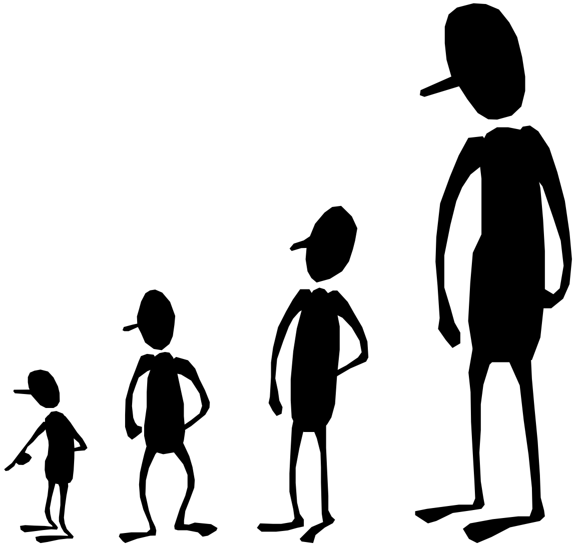
Part 2
Risk mitigation centered design

Things to consider when designing

- In addition to technical issues
  - Security strength
  - Algorithmic efficiency
  - Code size
- Business issues
  - 1-way hash is one of many components
  - Managing risks is the real issue
  - Cost of
    - Deployment,
    - maintenance,
    - upgrading &
    - replacement

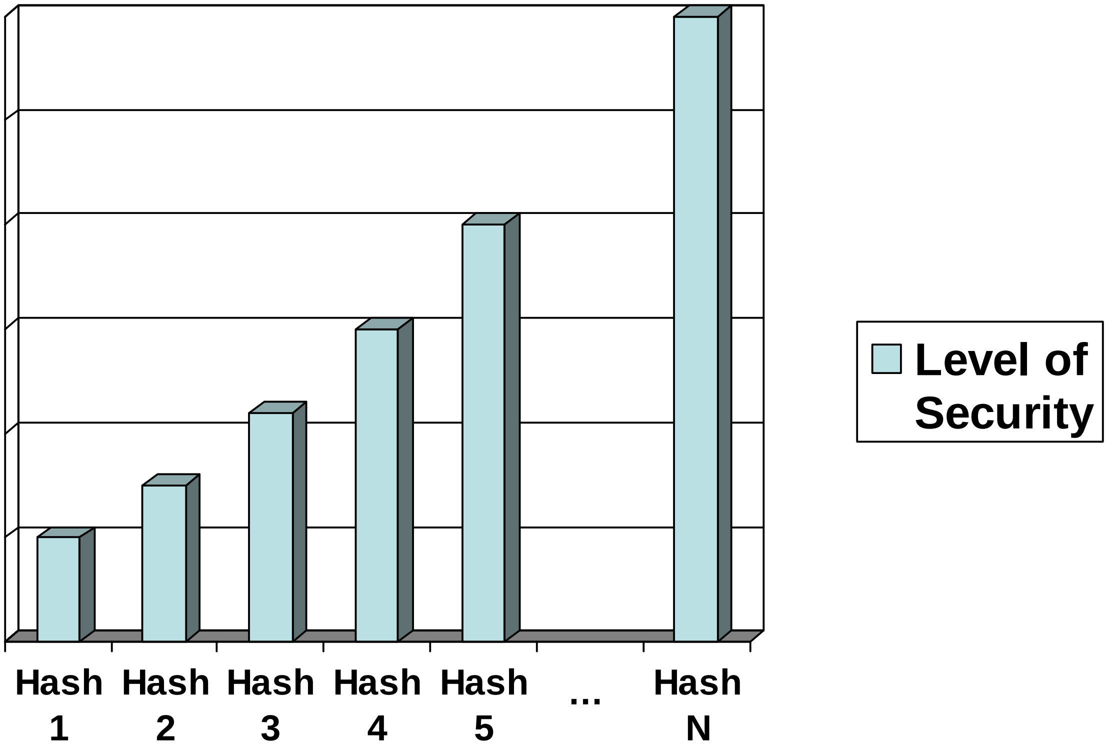
Specific suggestion #1

- Design families of parameterized 1-way hash algorithms
  - Past examples
    - HAVAL (15 siblings)
    - SHA (SHA1, SHA224, SHA256, SHA384, SHA512)



Specific suggestion #2

- Sibling algorithms have varying security strengths



Specific suggestions #3-#6

- Diverse internal structures
  - “genetic heterogeneity”
- Same/similar code size for siblings
- Support for same API
- Comparable computational performance

Specific suggestion #7

- Build an on-line repository of approved 1-way hash algorithms
  - Allow “hot switching”

Future work

- To verify in a formal way the effectiveness of techniques for enhancing MD hashing
  - Local message tagging
  - Global message tagging

Enhancing the MD-Strengthening and Designing Scalable Families of One-Way Hash Algorithms

Neil Kauer [*] Tony Suarez [*] Yuliang Zheng [†]

November 1, 2005

Abstract

One-way hash algorithms are an indispensable tool in data security. Over the last decade or so a number of one-way hash algorithms have been designed and many of them have been used in numerous applications. Recent progress in cryptanalytic attacks on one-way hash algorithms by Wang and co-workers, however, has brought up the urgency of research into new and more secure algorithms. The goal of this paper is two-folded. On one hand we propose a simple technique to affix authentication tags to messages prior to being hashed by an iterative one-way hash algorithm with the aim of increasing the overall security of the algorithm against cryptanalytic attacks. One the other hand we advocate the importance of a system oriented approach towards the design and deployment of new families of one-way hash algorithms that support greater scalability and facilitate migration to newer member algorithms upon the compromise of deployed ones. We base our observations on a common sense premise that there is no specific one-way hash algorithm can remain secure forever and it will eventually be broken by a cryptanalytic attack faster than exhaustive research.

Keywords

One-Way Hash Algorithm, Replaceable One-Way Hash Algorithm, One-Way Hash Family, Long Term Security, Risk Mitigation

1 Introduction

The most notable technique for designing one-way hash algorithms is an iterative method proposed by Damgaard and Merkle in their papers presented at CRYPTO'89 [2, 8]. Damgaard and Merkle's technique is also called the MD-strengthening. The core of the Damgaard-Merkle design is a block compressor that takes as input a message block of fixed size and outputs a new block that too has a pre-specified size. To hash a message of arbitrary size, one adds a padding to the end of the message and views the padded message as a concatenation of message blocks each of which has the same size required by the block compressor. This iterative approach is illustrated in Figure 1 where the block compressor is denoted by F and the message blocks by M_1, M_2, . . ., M_n, all of which contain the same number of bits.

The iterative approach advocated by Damgaard and Merkle has greatly influenced the design of a number of one-way hash algorithms, including but not limited to SHA1 and its siblings [9], the MD family [6, 12, 13], HAVAL [22], and the RIPEMD family [11, 3].

Since the publication of these algorithms, we have witnessed the steady accumulation of novel cryptanalytic techniques to attack these algorithms. Perhaps the most significant de-

[*] Corporate Information Security, Wachovia Bank, 1525 West T. Harris Blvd, Charlotte, NC 28288, USA. {neil.kauer, tony.suarez}@wachovia.com

[†] Information Security and Assurance Center, UNC Charlotte, 9201 University City Blvd, Charlotte, NC 28223, USA. yzheng@uncc.edu

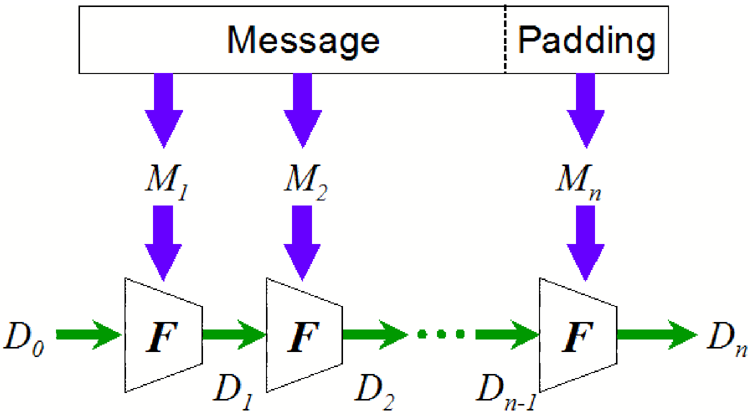


Figure 1: Damgaard-Merkle Iterative One-Way Hash

velopment so far is a sequence of successful attacks discovered by Xiaoyun Wang and co-workers [16, 20, 17, 19, 21, 18]. The colliding MD5 based RSA public key certificates successfully created by Lenstra, Wang and de Weger [7] demonstrate quite convincingly the potential effectiveness and practicality of the new crypt-analytic attacks. A few other relevant papers include [1, 5, 4].

These latest advancements in breaking iterative one-way hash algorithms remind us of the importance of developing new ideas on the design of stronger one-way hash algorithms. One obvious approach would be to research into completely new design methods that are different from the iterative method of Damgaard and Merkle. A more conservative approach would be to investigate techniques that may be used to enhance the security of iterative algorithms based the MD-strengthening.

In addition to technical approaches to more secure one-way hash algorithms, we believe that other issues such as deployment, upgrading, scalability and choices for users and security risk mitigation should also be taken into account when designing new algorithms for future applications.

This paper contributes to the endeavor of searching for new one-way hash algorithms for future applications in two different ways. The first is technical. Specifically we propose to use authentication tags to strengthen the security of an iterative one-way hash algorithm. And the second is presented with a view to minimize costs and inconvenience accompanying upgrading an existing one-way hash algorithm to newer one.

2 Technical Approaches

With the Damgaard and Merkle technique, a message is always padded to the end to ensure that the resultant message can be viewed as the concatenation of a multiple number of equally sized blocks required by the underlying block compressor. The padding is generated on-the-fly according to pre-determined rules. The length in bits of the original message is typically included in the padding.

Padding a message before applying an iterative hashing to blocks of the message can be viewed as a technique that maps the original, less structured message to one that is structurally richer. When resultant messages are highly structured, a collision attacker is forced to look for colliding messages from the structured messages rather than from the original unstructured ones, whereby significantly increasing the hurdle for the attacker to launch a successful attack. We note that converting to structured messages is in a way reminiscent of error correcting codes which are widely used in communication systems.

We examine two related techniques for adding structural information to messages to be hashed.

The first technique is more suitable for relatively short messages or long messages that can be scanned twice, while the second technique is designed for data streams or long messages that only allow to be scanned twice locally.

In [15], Szydlo and Yin discuss different approaches to enhancing MD5 and SHA-1 that are built using the Damgaard and Merkle technique.

2.1 Global Tagging

We observe that in most applications, messages to be hashed are relatively short. These applications include authenticated key establishment (SSL, IPsec), authenticated message exchange, authenticated financial transactions, one-way hash based pseudo-random number generation, public key certificates, and many others. Some applications may involve long messages that are stored in a secondary storage, and hence may be scanned twice or more during the process of creating an authentication tag. Affixing an integrity check sum to a large software and data package or the entire contents on a large capacity storage medium like a DVD and a magnetic or optical tape falls into this type of applications.

This motivates us to look into ways to add structured information to an entire message prior to being hashed by an iterative one-way hash algorithm. Two crucial requirements for adding information to a message are that the operation is fast and the resultant message corresponds to an element from a large set of highly structured mathematical objects.

One method is to compute a tag using a fast message authentication code (MAC) and affix the tag to both the start and the end of the original message. The key required by the message authentication code may be a public value and fixed across all applications. Alternatively, the key may act as a tweakable parameter that is chosen at random for a specific set of applications and may or may not be made public. The resultant message that has the message authentication tag affixed to both ends would then be padded in a way similar to MD5 or SHA1 and hashed iteratively using an block compressor. This approach is indicated in Figure 2, and can be described in a slightly formal way as follows:

Let M be a message, MAC be a message authentication code, D_0 be an initial hash value required by the underlying iterative one-way hash algorithm, and K_1 be a key for a message authentication code which is either fixed across all applications or agreed upon by relevant parties for a specific application. Also let $padding$ denote information that is padded to the end of a message, and MD be an iterative hash operation such as one based on the MD-strengthening. MD takes as input an initial hash value and a message whose length is a multiple of the basic block required by the underlying block compressor. And finally let D denote the hash value of the entire message. Then the proposed approach can be summarized as follows:

1. Computing MAC tag: $T = \text{MAC}(K_1, M)$;
2. Affixing MAC tag: $M' = (T, M, T)$;
3. Padding: $Padding = \text{PAD}(M')$, $M'' = (M', Padding)$;
4. Iterative Hashing: $D = \text{MD}(D_0, M'')$

For a very short message, say one that has fewer than a couple of thousand bits, additional padding may be required prior to computing a MAC tag to ensure that the resultant message contains an adequate number of (say at least 16) basic blocks for the iterative one-way hash.

2.2 Local Tagging

In some applications it may be impractical to affix a message authentication tag of an entire message to both ends of the message. An example of such applications include those that involve stream data such as digitized voice and video images. Another example is hashing with resource constrained devices such as a smart card which may not have adequate buffer memory for the storage of a message in its entirety which may be required in order to scan the message twice or more.

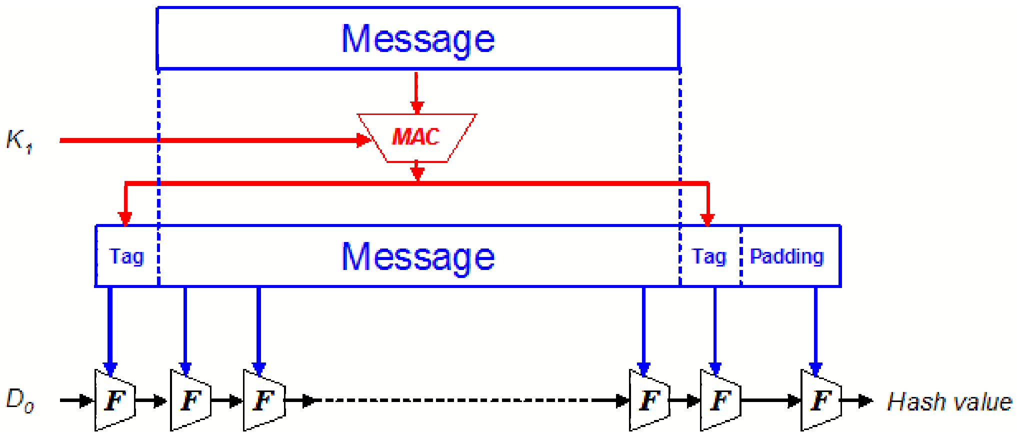


Figure 2: Global Tagging

For these applications, a possible approach is to compute an authentication tag on part of a message and affix the tag to both ends of the part involved. Specifically, we can view the original message as the concatenation of a sequence of message segments, each of which is in turn the conjunction of a number of basic message blocks required by an iterative block compressor. The length of a segment can be as long as practical, only restricted by such factors as buffer size and permissible processing delay.

With this method, the cryptographic key required for computing the message authentication tag of the first segment can be either fixed or selected for specific applications. For a subsequent segment, the intermediate hash value obtained just before arriving at the message segment can be used to derive a key for the message authentication code. Figure 3 illustrates this approach. Likewise, the approach can also be described in a slightly formal way.

Let M be a message, MAC be a message authentication code, D_0 be an initial hash value required by the underlying iterative one-way hash algorithm, and K_1 be a public value acting as a key for the message authentication code. K_1 can be either fixed across all applications or agreed upon for a specific set of applications. View the message M as the concatenation of segments of equal size, namely, $M = S_1 S_2 \cdots S_\ell$, where all S_i's have an equal number of (say at least 16) message blocks. Also let MD be a hash operation that, given an initial hash value and a message segment, iteratively applies to the message blocks a block compressor such as one based on the MD-strengthening. Further, let GetMACkey be a simple function that creates a key for a message authentication code from an intermediate hash value, and PAD an operation to create a padding of an input message. And finally let D denote the hash value of the message. The proposed approach can be summarized as follows:

1. Let $Y_0 = D_0$;
2. For $i = 1, 2, \ldots, \ell$ do the following:
   (a) Computing MAC tag: $T_i = \text{MAC}(K_i, S_i)$;
   (b) Affixing MAC tag: $S_i^* = (T_i, S_i, T_i)$;
   (c) Iterative Hashing: $Y_i = \text{MD}(Y_{i-1}, S_i^*)$;
   (d) Setting MAC Key: $K_{i+1} = \text{GetMACkey}(Y_i)$;
3. Padding: $Padding = \text{PAD}(S_1^* S_2^* \cdots S_\ell^*)$;
4. Finalizing: $D = \text{MD}(Y_\ell, Padding)$.

Note that the last segment S_ℓ may need to be padded prior to the computation of the MAC tag in order for it to have an adequate number of basic message blocks.

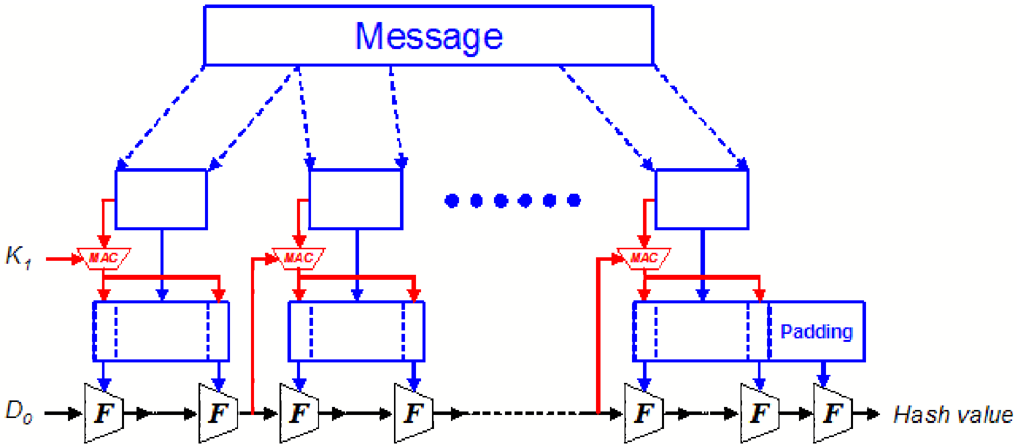


Figure 3: Local Tagging

2.3 Variants

A number of variants of the above method can be considered to trade the level of security for efficiency. In one of these variants message authentication tags are computed and affixed to every second segment. The number of segments can be made sure to be an odd number of at least three (3) so that the last segment is always affixed with tags to both ends. In another variant, the size (number of blocks) of a segment may be a variable from a specified range, determined by a previous intermediate hash value.

We also note that an authentication tag may be slightly modified, say by flipping all of its bits, when it is attached to the end of a message segment.

In addition, the idea of affixing authentication tags may be adapted to the composition of one-way hash algorithms. Specifically we consider the sequential composition of two different one-way hash functions H_1 and H_2. As an example, consider a message M that can be scanned twice. One may compute its hash value as follows:

$$D = H_2(H_1(M), M, H_1(M))$$

See also Figure 4. It is hoped that the computational effort for breaking the sequentially composed hash algorithms is the sum of efforts for breaking the two individual hash algorithms separately, although a rigorous analysis yet needs to be carried out.

2.4 A Candidate Message Authentication Code

While in theory any MAC can be used in the calculation of authentication tags of segments, a good candidate MAC should fulfill a couple of important requirements. The first requirement is that the MAC must admit fast computation. The second requirement is the MAC can handle messages of variable lengthes, especially those that are long. And the third requirement is that the probability for one to find a second message that collides a known message is diminishingly small for all messages one might encounter in practice.

A careful examination of known message authentication codes against these requirements shows that very good candidates can be selected from authentication codes constructed from universal hash functions. Technically such an authentication code can be obtained from an "almost strongly universal (ASU)" or "almost exclusive-OR universal (AXU)" hash family by exclusive-ORing the output of a function from the universal hash family with an additional key chosen at random. Of particular interest is a universal hash family based on polynomial evaluation in a finite field called the evaluation hash [14].

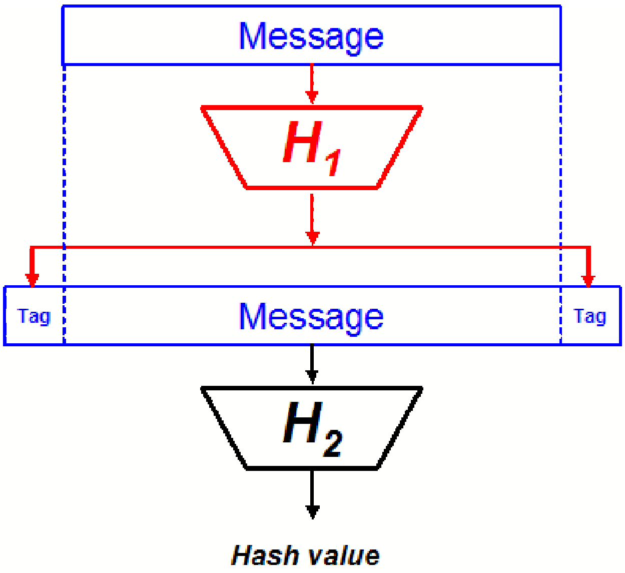


Figure 4: Sequential Composition

An analysis and comparison of performance of a few notable universal hash families including the evaluation hash can be found in [10].

Each instance of the evaluation hash is specified by two elements α and β of the finite field $GF(2^t)$, where t serves as a security parameter that determines the overall level of security of the message authentication code. A message M is viewed as the conjunction of n blocks $m_1, m_2, \cdots, m_n$, each of which has t bits and hence can be viewed as an element of $GF(2^t)$. Note that padding to the last block m_n may be needed to ensure that it too has t bits. As a result the message M can be viewed as a polynomial $m(x)$ of degree $n-1$ in $GF(2^t)$, namely $m(x) = m_1 + m_2 x + \cdots + m_n x^{n-1}$. The MAC tag of the message is then defined as follows:

$$\begin{aligned} tag &= \beta + \alpha \cdot m(\alpha) \\ &= \beta + m_1\alpha + m_2\alpha^2 + \cdots + m_n\alpha^n \end{aligned}$$

where all the operations are in $GF(2^t)$.

With the evaluation hash based message authentication code, the probability of finding a second message that collides a known message is approximately $n2^{-t}$, when α and β are chosen at random and unknown to a collision finder. Assuming that $t = 128$ and $n <= 2^{32}$, then the probability to successfully find a collision is at most 2^{-96}, a vanishingly small value. Note that the negligible probability of success still holds even if the collision finder has knowledge on α and β but does not use it in finding a collision. In particular, the probability for a new message obtained by flipping a few bits of a known message to collide the known message is bounded by the same vanishingly small value, if the flipping is done in a way that is not correlated to the values of α and β.

2.5 Advantages and Disadvantages

A major advantage of the two technical approaches, namely global tagging and local tagging, is that they allow the separation of the design of basic block compressors from the transformation of messages to be hashed. As a result, it has the potential to harden existing widely deployed one-way hash algorithms such as SHA1 by converting a message into a structured one prior to the use of the one-way hash algorithm. Further research on the effectiveness of this technique is required prior to its use in practice.

The converting operation can be done by employing a plug-and-play software routine or an additional piece of hardware handling the computation and affixation of authentication tags. As a result if the technique is indeed effective in hardening an existing one-way hash algorithm, it would help prolong the usability and life-span of widely deployed one-way hash algorithms and create more time for the smooth transition or migration to newer and hopefully more secure one-way hash algorithms.

A possible disadvantage is that additional computation is required to computer message authentication tags for message segments, resulting in a loss in the speed or performance of a one-way hash algorithm.

3 Risk Mitigation Friendly One-Way Hash Family

In the second part of this paper we turn our attention to a different aspect concerning the design, deployment and upgrading of one-way hash algorithms. It is important to note that one-way hash algorithms, and in fact almost all cryptographic techniques, are only some of the numerous tools used by an organization in achieving their organizational objectives, and for many organizations security is typically not considered as their *direct* goals, but rather intermediate steps on the way to achieve their real business objectives.

Further, in many organizations the deployment of cryptographic solutions is considered as part of a broader set of measures to mitigate risks that accompany the business operations of these organizations. Within this context, we feel that it is important for the designer of cryptographic algorithms to keep in mind the issue of minimizing costs associated with deployment, maintenance, upgrading and replacement of a cryptographic tool including one-way hash algorithms.

In the remaining part of this section we focus on a number of ideas on the design and deployment of future one-way hash algorithms that we hope will facilitate the process of mitigating risks in an unavoidable event when a one-way hash algorithm is eventually broken and rendered useless.

We emphasize that requirements we are going to present are not intended to be formal, and some of the requirements may not be necessarily consistent with others.

3.1 Parameterized One-Way Hash Algorithms

Instead of focusing on a single candidate one-way hash algorithm and hoping that it is secure an extended period of time, we propose to design a family of one-way hash algorithms. Each member of the family would have a well-understood level of security and performance, and all the member algorithms can be identified with a simple parameter or index. The parameter should correspond to a security level and allows all the members to be arranged or sorted more or less according to the security level.

A well designed parameterized family of one-way hash algorithms would have a considerably large number of members (say over 20). Such a one-way hash family would not only provide different applications with one-way hash algorithms that have most appropriate levels of security and performance, but also make it easier scale up, that is, to switch to a new member algorithm in an event when an algorithm used currently turns out be to vulnerable to attacks.

In a loose sense, HAVAL can be considered as a 15-member one-way hash family. Each member algorithm in HAVAL can be specified by a combination of rounds and sizes of output hash values. Likewise, SHA-0, SHA-1 and their newer siblings SHA-224, SHA-256, SHA-384 and SHA-512 can be viewed to form a family of one-way hash algorithms. And similarly, MD2, MD4 and MD5 can be regarded as a family. Further RIPEMD, RIPEMD128 and RIPEMD160 can be considered to be a family. In fact, one may also view all these algorithms as members of a larger family that is designed according to the MD-strengthening.

3.2 Diverse Internal Structures

One problem with the SHA, MD, HAVAL, and RIPEMD one-way hash families is that members in a family share a similar or identical structure. An undesirable consequence of this "genetic homogeneity" is that they may also share the same types of weaknesses. Once a particular member is found to have certain weaknesses, the level of confidence on other members of the same family would drop too, even though specific weaknesses are yet to be found with the other members.

It is therefore desirable for each member of the same family to be designed with structures that are as diverse as practical. While diversity in structure may not necessarily render the algorithms stronger in the long run, it nevertheless may increase significantly the amount of effort for "copycat attacks", namely adapting techniques for attacking one member to another from the same family.

3.3 Same Code Size

One-way hash algorithms that can be compiled into binary code of identical length/size would facilitate switching from one algorithm to another, especially when a algorithm is embedded in firmware or in a complex system that does not allow easy recompilation. The firmware for a device is typically stored in non-volatile memory whose size may not be easily changed once the device is shipped to a user. Therefore upgrading the firmware to a newer version with a different one-way hash algorithm can be made easier if the algorithms have the same code size. For some large, complex systems, the current trend is to update parts of the system without the need to completely stop the whole system. Such a system should benefit from the use of one-way hash algorithms with the same size.

We note that this requirement may conflict that of having diverse structures.

3.4 Comparable Computational Performance

When a practitioner decides to adopt a specific cryptographic algorithm, one of the many factors that he would have to consider is computational delay the algorithm introduces. The algorithm would be used only if the delay is within the acceptable range for the specific applications where the algorithm is intended to be used. It is therefore desirable that computational performance of member algorithms that are within the same vicinity when ordered according to a parameter is comparable. Satisfying this property will allow switching from one algorithm to another to be carried out without impacting significantly the overall performance of applications which use the algorithm.

3.5 Supporting the Same API

When members of a one-way hash family are implemented either in software or hardware, it would be preferable that all member algorithms can be implemented in such a way that they can be called or invoked by the same application programming interface (API). To meet this requirement, all member algorithms should have the same number and types of parameters in the API.

This requirement may not be as easy to meet as one would think at the first glance, if one also wishes to meet the requirement of diverse structures for member algorithms, especially in resource constraint environments such as embedded and mobile computing devices.

3.6 Building an Online Repository

A final aspect we consider is related to a repository of approved or standardized one-way hash algorithms and implementations. The idea is to build an online repository in such a way that applications can query the repository with a set of requirements and receive without too much delay from the repository (the executable, authenticated code of) the best candidate that fulfill the requirements. This query and feedback process may need to be done on the fly and in real time to achieve the goal of "hot switching" or "just-in-time switching".

Even assuming that such a approach is feasible, there is likely a large cost associated with the establishment of such a repository as well as the development of applications that are made aware of the repository. In addition, success of this type of repositories would rely heavily on the availability of an authentication infrastructure such as a public key infrastructure or PKI. Since one-way hash algorithms are also used in a PKI, it appears that we may have to address two circular problems which rely on each other for proper functions. Nevertheless, if the repository can be built, it is envisioned that many applications may be freed from updates, recompiling and redeployment.

4 Future Research

The speculation on the potential of using authentication tags to strengthen an iterative one-way hash algorithm needs to be further examined in a careful manner before it is put in practice. More research needs to be done with respect to one-way hash families that meet some or all of the requirements outlined in Section 3. We hope this paper serves as an opener for new approaches and ideas that may lead to better, more secure and more usable one-way hash algorithms.

References

[1] Eli Biham and Rafi Chen. Near-collisions of SHA-0. In *Advances in Cryptology - CRYPTO'04*, volume 3152 of *Lecture Notes in Computer Science*, pages 290–305, Berlin, New York, Tokyo, 2004. Springer-Verlag.

[2] Ivan Damgård. A design principle for hash functions. In *Advances in Cryptology - CRYPTO'89*, volume 435 of *Lecture Notes in Computer Science*, pages 416–427, Berlin, New York, Tokyo, 1990. Springer-Verlag.

[3] Hans Dobbertin, Antoon Bosselaers, and Bart Preneel. RIPEMD-160, a strengthened version of RIPEMD. In *Fast Software Encryption - FSE'96*, volume 1039 of *Lecture Notes in Computer Science*, pages 71–82, Berlin, New York, Tokyo, 1996. Springer-Verlag.

[4] Antoine Joux. Collisions for SHA0. Technical report, Presented at Crypto'04 Rump Session, 2004.

[5] Antoine Joux. Multicollisions in iterated hash functions: Application to cascaded constructions. In *Advances in Cryptology - CRYPTO'04*, volume 3152 of *Lecture Notes in Computer Science*, pages 306–316, Berlin, New York, Tokyo, 2004. Springer-Verlag.

[6] Burt Kaliski. The MD2 message digest algorithm. Request for Comments RFC 1319, IETF, 1992.

[7] Arjen Lenstra, Xiaoyun Wang, and Benne de Weger. Colliding X.509 certificates. ePrint Archive http://eprint.iacr.org/2005/067, March 2005.

[8] Ralph C. Merkle. One way hash functions and DES. In *Advances in Cryptology - CRYPTO'89*, volume 435 of *Lecture Notes in Computer Science*, pages 428–446, Berlin, New York, Tokyo, 1990. Springer-Verlag.

[9] National Institute of Standards and Technology. Secure hash standard. Federal Information Processing Standards Publication FIPS PUB 180-2, U.S. Department of Commerce, August 2002.

[10] Wim Nevelsteen and Bart Preneel. Software performance of universal hash functions. In *Advances in Cryptology - EUROCRYPT'99*, volume 1592 of *Lecture Notes in Computer Science*, pages 24–41, Berlin, New York, Tokyo, 1999. Springer-Verlag.

[11] Research and Development in Advanced Communication Technologies in Europe. RIPEM integrity primitives: Final report of RACE integriy primitives evaluation

(R1040). Technical report, RACE, June 1992.

[12] Ronald L. Rivest. The MD4 message digest algorithm. Request for Comments RFC 1320, IETF, 1992. (Also presented at Crypto'90, 1990).

[13] Ronald L. Rivest. The MD5 message digest algorithm. Request for Comments RFC 1321, IETF, 1992.

[14] Victor Shoup. On fast and provably secure message authentication based on universal hashing. In *Advances in Cryptology - CRYPTO'96*, volume 1109 of *Lecture Notes in Computer Science*, pages 313–328, Berlin, New York, Tokyo, 1996. Springer-Verlag.

[15] Michael Szydlo and Yiqun Lisa Yin. Collision-resistant usage of MD5 and SHA-1 via message preprocessing. ePrint Archive http://eprint.iacr.org/2005/248, 2005.

[16] Xiaoyun Wang, Dengguo Feng, Xuejia Lai, and Hongbo Yu. Collisions for hash functions MD4, MD5, HAVAL-128 and RIPEMD. ePrint Archive http://eprint.iacr.org/2004/199, Presented at Crypto'04 Rump Session, 2004.

[17] Xiaoyun Wang, Xuejia Lai, Dengguo Feng, Hui Chen, and Xiuyuan Yu. Cryptanalysis for hash functions MD4 and RIPEMD. In Ronald Cramer, editor, *Advances in Cryptology - EUROCRYPT'05*, volume 3494 of *Lecture Notes in Computer Science*, pages 1–18, Berlin, New York, Tokyo, 2005. Springer-Verlag.

[18] Xiaoyun Wang, Andrew Yao, and Frances Yao. New collision search for SHA-1. Technical report, Presented at Crypto'05 Rump Session by Adi Shamir, August 2005.

[19] Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu. Finding collisions in the full SHA-1. In Victor Shoup, editor, *Advances in Cryptology - CRYPTO'05*, volume 3621 of *Lecture Notes in Computer Science*, pages 18–36, Berlin, New York, Tokyo, 2005. Springer-Verlag.

[20] Xiaoyun Wang and Hongbo Yu. How to break MD5 and other hash functions. In Ronald Cramer, editor, *Advances in Cryptology - EUROCRYPT'05*, volume 3494 of *Lecture Notes in Computer Science*, pages 19–35, Berlin, New York, Tokyo, 2005. Springer-Verlag.

[21] Xiaoyun Wang, Hongbo Yu, and Yiqun Lisa Yin. Efficient collision search attacks on SHA-0. In Victor Shoup, editor, *Advances in Cryptology - CRYPTO'05*, volume 3621 of *Lecture Notes in Computer Science*, pages 1–16, Berlin, New York, Tokyo, 2005. Springer-Verlag.

[22] Yuliang Zheng, Josef Pieprzyk, and Jennifer Seberry. HAVAL - a one-way hashing algorithm with varialbe length of output. In Jennifer Seberry and Yuliang Zheng, editors, *Advances in Cryptology - AUSCRYPT'92*, volume 718 of *Lecture Notes in Computer Science*, pages 83–104, Berlin, New York, Tokyo, 1993. Springer-Verlag.