Some thoughts on Collision Attacks in the Hash Functions MD5, SHA-0 and SHA-1 *

Praveen Gauravaram **,William Millan and Juanma Gonzalez Neito

Information Security Institute (ISI), QUT, Australia.
{p.gauravaram,b.millan,j.gonzaleznieto}@isi.qut.edu.au

Abstract. The design principle of Merkle-Damgård construction is collision resistance of the compression function implies collision resistance of the hash function. Recently multi-block collisions have been found on the hash functions MD5, SHA-0 and SHA-1 using differential cryptanalysis. These multi-block collisions raise several questions on some definitions and properties used in the hash function literature. In this report, we take a closer look at some of the literature in cryptographic hash functions and give our insights on them. We bring out some important differences between the 1989's Damgård's hash function and the hash functions that followed it. We conclude that these hash functions did not consider the pseudo-collision attack in their design criteria. We also doubt whether these hash functions achieve the design principle of Merkle-Damgård's construction. We formalise some definitions on the properties of hash functions in the literature.

Keywords: Hash function, collision attacks, MD5, SHA-0 and SHA-1

1 Introduction

In 1989, Damgård [5] and Merkle [11] independently proposed a similar iterative structure to construct collision resistant cryptographic hash functions. Since then, this iterated design has been called as Merkle-Damgård iterative structure. The design principle or motivation of this construction is:

> "If there exists a computationally collision free function f from m bits to t bits where $m > t$, then there exists a computationally collision free function h mapping messages of arbitrary polynomial lengths to t-bit strings." [5]

Since then, it has been believed that the Merkle-Damgård [5, 11] iterated construction reduces the problem of designing a collision resistant hash function on inputs of arbitrary length to the design of a collision resistant compression function on inputs of fixed size. The current dedicated hash functions such as MD4, MD5, SHA-0, SHA-1 and SHA-256 are constructed following this design criterion. As far as we know the first practical hash function that inherited the above motivation is MD4 [14]. Since MD4 and the hash functions that followed it were influenced by Damgård's work, we stick to referring only to Damgård's work in this paper. For example, one can see a sample of the following statements from the literature that emphasize the above belief.

1. "A collision resistant compression function implies a collision resistant iterated hash function." [9]

* This is a draft version of the work in progress. Any comments, suggestions and corrections via email are very much appreciated and acknowledged in the final draft.

** The author(http://www.isrc.qut.edu.au/people/subramap/) has been supported by the QUT post-graduate research award and ISI top-up funding.

2. "If one properly iterates a collision-resistant function with a fixed domain, then one can construct a collision resistant hash-function with an enlarged domain." [15]
3. "Weaknesses in the compression function will generally result in weaknesses of hash function, but the converse does not hold in general." [12]
4. "The motivation for this iterative structure arises from the observation that if the compression function is collision-resistant then so is the resultant iterated hash function.(The converse is not necessarily true)." [2]
5. "It is easy to see that if compression function is collision-resistant then so is iterated hash function". [3]

The recent multi-block differential collision attacks on the hash functions MD5, SHA-0 and SHA-1 [4, 18, 19] cast serious doubt on this *well established belief* due to the 1989's Damgård construction and the way this belief is used in constructing cryptographic hash functions starting from MD4.

In this report, we step back from the flurry of recent attacks on the popular and widely used hash functions to give our insights on "what is fundamentally wrong?". We will examine what actually the above list of statements actually mean. In the process, different sections of some of the literature in the hash functions are related to develop the understanding of the fundamental concepts in hash functions. The main motivation of this paper is to relate as closely as possible some fundamentals on hash functions to the latest attacks on hash functions and to explore some narrow gaps between fundamental concepts and recent collision attacks on hash functions. The paper starts with an informal discussion proceeding toward formalizing some definitions.

The paper is organised as follows: Section 2 of the paper gives some background information on the specification of hash functions, different informal definitions on collision attacks and the recent differential multi-block collision attacks on hash functions. Section 3 gives new observations on multi-block collision attacks on hash functions and answers several questions. Section 4 points out the differences between Damgård's original hash function construction and current dedicated hash functions to see whether these hash functions follow the design objectives of Damgård's construction. Section 5 gives some formal definitions related to various collision attacks on hash functions. Section 6 concludes the report.

2 Background

Specification of the hash functions:
For implementation efficiency of a hash function, we require some kind of iterated structure (similar to the product cipher concept in block ciphers). Therefore a compression function must be defined along with a style of iteration. The Merkle-Damgård style is the absolute simplest possible one can achieve.
The specification of a hash function following Merkle-Damgård iterative structure requires a description of the compression function, fixing an initial value (IV), a padding procedure and a pre-processing stage [10,12]. Every hash function fixes the IV (such an IV is called as fixed IV [10]) or a small set of IVs that can be used with the hash function, together *with the motivation of the choice* to avoid the accusation that it is a deliberately inserted weakness [12]. If IV is generated pseudorandomly, the algorithm that generates the IV should be described. The specification includes an upper bound on the size of the input.

If the IV of the hash function is not specified, it should be hard to find collisions for any IV. In such a case, to avoid some trivial attacks, the length of the information that gets hashed is added in a separate block at the end [12]. The current practical dedicated hash functions (MD4, SHA-0, SHA-1,SHA-256) fix the IV of the hash function and also add the length of the information at the end of the message. Also, these hash functions fixed only one IV not a small set of IVs in their design specification.

Interestingly, we observed that there is neither a motivation nor an algorithm given that generates the IV for the hash functions SHA-1 [1] and MD5 [13]. Though there is an algorithm that generates the IV for the hash functions SHA-256, SHA-224, SHA-384 and SHA-512 [1], there is no motivation for the choice of the IVs.

Collision attacks on hash functions:

Informally, a hash function is said to be collision resistant if it is hard to find any two inputs that map to the same output or digest for a given specification of the hash function. A hash function is said to be near-collision resistant if it is hard to find any two inputs such that their digests differ in only a few bits for a given specification of the hash function. Formal definitions are given in Section 4 of the paper. Based on the IV used in finding collisions, collision attacks in hash functions are classified as follows [10]:

1. Collision attack: collisions using a fixed IV for two distinct message inputs(*Type 1* collisions).
2. Semi-free-start collision attack: collisions using the same random (or arbitrary) IV for two distinct message inputs(*Type 2* collisions) .
3. Pseudo-collision attack: free-start collision attack using two different IVs for two distinct message inputs (note this is the basis of most published attacks [4, 19]) (*Type 3*collisions).

A collision attack is considered to be a stronger attack compared to the other two as the other two attacks deviate from the strict definition of a collision which is to find two distinct inputs that map to the same output for a given specification of the hash function. Semi-free-start collision attack is not so dangerous if the IV used in the attack is an outcome of a pseudorandom process as the probability of hitting the right IV is negligible [12, p.33]. Obviously, pseudo-collision attacks deviate significantly from the strict definition of a collision. In addition, many hash functions might not have considered this attack in their design criteria. This was also pointed by Preneel [12]. We will see this in Section 3.

Multi-block collision attacks on hash functions

The collisions attacks on the hash functions MD5, SHA-0 and SHA-1 are multi-block collision attacks [4, 18, 19].

Collisions have been found in these hash functions without ever getting a *Type 1*-collision on the first application of the CF. The attacks use near-collisions obtained after processing the first distinct message blocks (x_1, x_1') as a tool to get *Type 3* collisions for the second distinct message blocks (x_2, x_2') as shown in Fig 1. For example, 2-block collisions were found in MD5 and SHA-1 [18, 19] and 4-block collisions were found in SHA-0 [4] (This was later improved in [17] to the collision format $H(M, M') = H(M, M'')$). One can append extra blocks after the collided blocks to extend the collisions and this is possible due to the length extension property of the Merkle-Damgård structure. These collisions are basically a chain of trivial *Type 2* collisions.

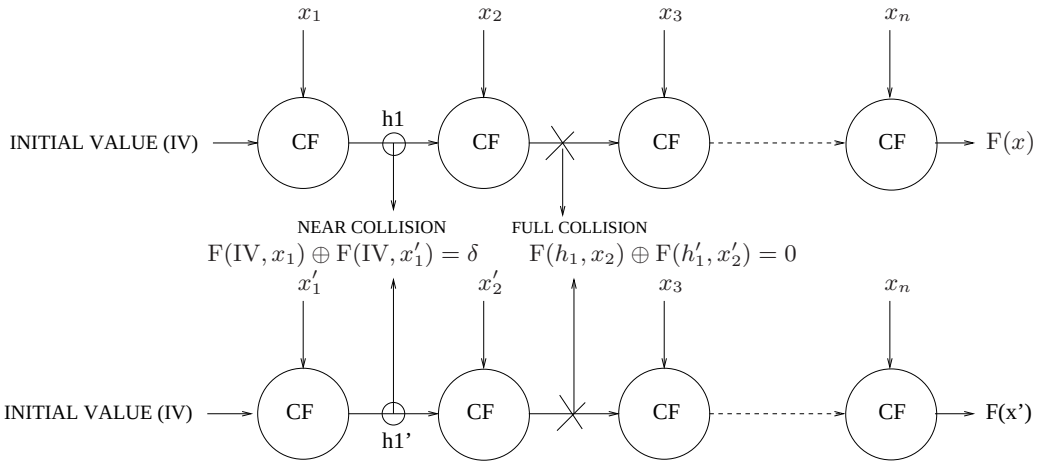

Fig. 1. 2-Block collision in a hash function F (F, is for example, MD5)

3 New observations on multi-block collisions

Now let us observe following statement from [10].

> "A compression function secure against fixed IV collision attacks is necessary and sometimes, but not always, sufficient for a secure iterated hash; and security against pseudo collision attacks is desirable, but not always necessary for a secure hash function in practice" [10, p.373].

If we compare the multi-block collision attacks on MD5, SHA-0 and SHA-1 with the above statement several observations can be made. We can think of the following:

1. No *Type 1* collisions have been found in the compression functions of MD5, SHA-0 and SHA-1 using the fixed IV. Nevertheless these are not secure hash functions (see [4, 18, 19]). The cryptanalysis of these hash functions concurs with the first part of the above statement.
2. Multi-block collisions were found after processing more than one application of the compression function. For example, after two blocks in MD5 [19] and SHA-1 [4] functions and after 4 blocks for SHA-0 [18]. Consider, for example, a 2-block collision as shown in Fig 1. The two chaining variable inputs ($h1$ and $h1'$ in Fig 1) to the second compression function are the outputs of the first compression function and they are not same as the IV of the hash function. If we assume these collisions on second blocks to be *Type 3* collisions and assume the second part of the above statement to be true then it turns out to be that MD5 and SHA-1 are somehow secure hash functions.
3. From points 1 and 2, we may say that MD5 and SHA-1 are not secure iterated hash functions in "some way" and not secure in practice.

From the informal definitions of near-collisions and pseudo collisions in Section 2, we can say that the *multi-block collision is a combination of a near-collision and a pseudo-collision in a hash function*. In principle, there are other ways of generating multi-block collisions instead of using near-collisions (see Appendix A). Establishing collisions in hash functions through near-collisions is easy for humans using differential cryptanalysis. Any two different chaining variables can work with

the fixed IV of the hash function in creating a full hash collision after processing some subsequent message blocks. If this is the case, then *clearly the hash functions MD5, SHA-0 and SHA-1 that are weak against multi-block collisions have not considered security against pseudo-collisions in their design criteria.* Preneel has pointed out in 1993 [12] itself that most hash functions are not designed to meet this criteria. Note that SHA-1 did not exist then. This might have been the case with the hash functions SHA-256 and SHA-512.

Now consider the second part of the above quotation which says that the security against pseudo collision attacks is desirable, but not always necessary for a secure hash function in practice. In the wake of multi-block collision attacks, we will make this statement a bit more clear. It is the combination of near-collisions produced by the fixed IV and pseudo collisions produced by the nearly collided values after the first message block was processed that produced multi-block collisions. Pseudo-collisions are not always necessary for a secure hash function in practice when only pseudo IVs (IVs different from the fixed IV or a set of fixed IVs of the hash function) are used on the compression function in generating collisions as done in [7].

Now the question is: ***"Can we give equal importance to Type 3 collisions and Type 1 collisions?"***

From the above discussion, the tools involved in finding multi-block collisions are fixed IV of the hash function and nearly collided chaining values. In practice, collisions to the hash function are considered to be significant when they are obtained using the fixed IV of the hash function but it does not matter whether the fixed IV collisions are obtained for the first application of the compression function of the hash function or not.

Hence, we define a new security requirement for cryptographic hash functions by rephrasing the statement introduced at the beginning of this Section as follows:

> *A compression function secure against fixed IV collision attacks is necessary and sometimes, but not always, sufficient for a secure iterated hash. The security of the compression function against pseudo collision attacks is desirable and necessary for a secure hash function in practice.*

The above security requirement may be difficult to achieve in practice. This requirement may also admit to an arbitrary number of formalizations. We attempt to formalize different types of collisions in Section 4 of the paper.

4 Do these hash functions follow the design principle of MD?

Recall that the design criterion of Merkle-Damgård iterated construction is collision resistance of the compression function implies collision resistance of the hash function. But the recent cryptanalysis shows that this is not true. *Collision resistance of the compression function is necessary but not sufficient for an iterated hash function.*

In the multi-block collision attacks on MD5, SHA-0 and SHA-1, it was found to be hard to find full collisions for the first iteration of the compression function using the fixed IV of the hash function than to do so for the second iteration. For example, the complexity of finding near-collisions or generating two closely related pseudo IVs for the SHA-1 compression function using fixed IV is 2^{68} computations whereas the complexity of finding full collision on 2 blocks is 2^{69} [18] (this

was recently improved to 2^{63} [20]). In addition, it is obvious as shown by the cryptanalysis of hash functions that a search for one-block near-collisions for the compression function with fixed IV is easier than the search for the full collisions for the compression function with fixed IV [4].

Comparing what has actually happened from the recent attacks on the hash functions to the list of quotes we have given in Section 1, a question that one would ask is, *Do the hash functions MD5, SHA-0 and SHA-1 follow the design principle of Damgård construction* [1].

One may think from these multi-block collision attacks that the *collision resistance property of the compression function is not preserved after the second use of the compression function, thus contradicting the design principle of Damgård construction.* The attacks on the hash functions MD5, SHA-0 and SHA-1 may show that these hash functions no longer follow any of the list of the statements given in Section 1 as collision resistance property of the compression function using fixed IV is not extended to the hash function.

As said earlier, the first practical hash function that was influenced by Damgård's design is MD4 [14]. MD5 and hash functions from the SHA family followed MD4 in their design approach. These practical hash functions that inherited Damgård's original design differ from Damgård's design in some ways. We repeat Damgård's construction [5,12] here to explain the differences between it and the hash functions (starting from MD4) that used it in their design.

Fact: Let f be a fixed size CRHF family mapping n bits to l bits. Then there exists a CRHF family F mapping strings of arbitrary length to l bit strings.
Damgård has given a proof for the above statement using two constructions considering two different cases. The two constructions are as follows:

1. $n - 1 > 1$: The message is split into t blocks of size $n - l - 1$ bits with the application of a padding rule. The sequence F_i is then defined by:

$$F_1 = f(0^{l+1}||x_1)$$
$$F_i = f(F_{i-1}||1||x_i) \text{ for } i = 2, 3, \ldots, t$$

   The digest $F(x)$ is equal to F_t.
2. $n - 1 = 1$: In this case message is processed bit by bit. An l-bit string F_0 is selected uniformly. The sequence F_i is then defined by:
   $F_i = f(F_{i-1}||x_i)$ for $i = 1, 2, \ldots, t$

Here we list some differences (that we spotted) between Damgård's construction and the hash functions that inherited its design philosophy:

1. The first application of the compression function in Damgård's design contains 0's instead of a non-zero fixed IV used in the dedicated hash functions. That is, the initial state in Damgård's design is chosen to be 0^{l+1}.
2. The chosen initial state of 0^{l+1} in Damgård's construction avoids adversaries finding a collision for two messages of the form $x'||x$ and x. This case cannot occur *if the attacker cannot find a preimage under the compression function of the initial state, which is chosen to be* 0^{l+1} *in the first Damgård's construction.*

[1] The hash functions in MDx and SHAx family, resemble Damgård construction [5] of hashing a message block after block. So we just stick to referring to Damgård's design [5]

3. One can also choose the initial state arbitrarily and fix it as was done in the practical hash functions, then *under certain conditions on the compression function f, hardness of finding a preimage under the compression function of the initial state (fixed IV) follows from difficulty in finding a collision for the compression function of the initial state.* [6]
4. *Most hash functions in practice do not have this extra bit marking the start of the message. They implicitly assume that an adversary cannot find collisions to the compression function of the initial state (IV) implying hardness in finding a pre-image for the compression function using fixed IV.*
5. While Damgård's construction appends the message with a separate block containing 0's, the practical hash functions add the length of the information that gets hashed in the last block placing an upper bound on the amount of information that has to be hashed.

From above, we can say that a collision to the Damgård's construction using a non-zero fixed initial state will give a collision for the compression function f using the initial state. All practical hash functions have a non-zero initial state which is fixed. Even if an application uses, for example, MD5 hash function it means that the application uses MD5 hash function as a black-box with its fixed IV. So if the compression function is collision resistant with it's fixed IV then so is it's iterated version. It should be noted that Damgård's construction uses 0's instead of a fixed IV in it's design.

In Damgård's proof of his first construction [5], one can see that if there is a collision to the hash function there should have been a collision to one of the compression functions in the iteration which is valid only when the initial state is chosen to be 0^{l+1}. This is different when the first compression function uses a non-zero initial state as one of its inputs. In such a case, a collision for the hash function F would either give a collision for f under the initial state or a preimage under the initial state. Under certain conditions on the compression function f with that non-zero initial state, the collision intractability of the hash function follows from the hardness of finding a preimage using that non-zero initial state. This follows from collision intractability of the compression function with the fixed IV.

See [10] where it defines collision resistance as computational infeasibility in finding two distinct inputs x and x' such that $F(x) = F(x')$. One can assume that this might have implicitly considered the IV of the hash function in its collision resistance definition. Also look at the list of statements given in Section 1 that do not explicitly say whether the term "collision resistance" considers the initial state (fixed IV) of the hash function. They assume implicitly; the reason being as said before that the specification of a hash function includes the IV of the hash function.

If we look at the formal definition for collision resistance of the hash function family [15](see Section 4), it speaks of the difficulty in finding two distinct inputs that map to the same output for a key chosen at random from a set of keys and comments that strengthening the definition for all keys in the set does not make any sense. This formal definition may be indirectly saying that one has to make sure to fix a set of initial values in the design criteria for a hash function.

From this discussion, one can easily see that setting an IV or a set of IVs should be part of the design criteria for the hash function as pointed out by Preneel [12]. In addition, finding *Type 1* collisions for the hash function is a stronger attack than finding collisions by other means. To be specific and accurate, finding collisions for the compression function using the initial state (fixed IV) is the most powerful attack (for example, see collisions on MD4, RIPEMD [16]). The design criteria of the hash function should also take into account the protection against pseudo collisions as explained in Section 3. We are unsure how to see this formally at the moment as one cannot give a formal definition for a collision based on fixed IV [15].

Now, we can say that collision resistance of the compression function with the initial state of the hash function implies collision resistance of the hash function. For this statement to be valid, a collision to the hash function should have given a collision to the compression function with the initial state (fixed IV). But this is not what has happened with the broken hash functions MD5, SHA-0 and SHA-1 as the compression functions of these hash functions are still collision resistant with respect to their respective initial states. Hence, we doubt whether these hash functions satisfy the motivation of Damgård's design.

5 Formal Notion

In this section we revisit the formal definition given for collision resistance property of the hash function [15].

Notation:

For simplicity, the notation of [15] is followed in this section. We define a compression function F as a family of functions, $F : \mathcal{K} \times \mathcal{B} \rightarrow \{0,1\}^n$. $\mathcal{B} = \{0,1\}^b$ is the fixed size input to any compression function in the family F.

Definition 1. *Let $H : \mathcal{K} \times \mathcal{M} \rightarrow \mathcal{Y}$ be a hash function family and let A be the adversary. Then collision resistance is defined as:*

$$\mathrm{Adv}_H^{\mathrm{Coll}}(A) = \Pr[\, K \xleftarrow{\$} \mathcal{K}; (M_1, M_2) \xleftarrow{\$} A(K) : (M_1 \neq M_1) \wedge (H_K(M_1) = H_K(M_2))]. \qquad \square$$

As pointed out [15], it does not make sense to strengthen this definition for all $K \in \mathcal{K}$ as for any fixed function there exists an efficient program that finds a collision. But on the other hand, the design specification of the hash function should include an IV or a set of IVs along with the motivation of the choice. Let $\mathcal{K}'$ be the set of IVs or keys assigned for the hash function in the specification. Obviously, $\mathcal{K}' \subset \mathcal{K}$.

What we are trying to say here is there is a slight gap between the way the collision resistance has been defined and requirements of the specification of the hash function. The reason is collision resistance definition cannot be strengthened to all keys in the set $\mathcal{K}$ and at the same time it is a necessary requirement to fix a set of keys in the design specification of the hash function [12]. In addition, there are three different types of collision attacks as listed in Section 2. Which collision attack does the formal definition of collision resistance is considering? It seems that collision resistance property has taken into account the resistance against fixed IV and random IV collision attacks. This leads to a question that is it possible to give formal definitions for resistance against the three different types of collision attacks? One may ask if it is of any worth. Certainly, there is a value in doing so as there is a certain degree of importance for each of the three collision attacks as disclosed in Section 2 of the paper.

In our analysis, we differentiate the compression function from the hash function. Assuming that a hash function is designed with the specification containing a small set of keys in $\mathcal{K}' \subset \mathcal{K}$, everyone knows the keys in the set $\mathcal{K}'$. The keys that are not in the set $\mathcal{K}'$ can be considered as pseudo keys (leading to pseudo collision attack) or random keys (leading to semi-free-start collision attacks). Now we give formal definition for the collision resistance of the compression function followed by formal definition for other properties listed in Section 2.

Definition 2. *Let $F : \mathcal{K} \times \mathcal{B} \rightarrow \mathcal{Y}$ be a compression function family and let A be the adversary. Then collision resistance of the compression function is defined as:*

$\mathrm{Adv}_F^{\mathrm{Coll}}(A) = \Pr[\, K \xleftarrow{\$} \mathcal{K}; (M_1, M_2) \xleftarrow{\$} A(K) : (M_1 \neq M_2) \wedge (F_K(M_1) = F_K(M_2))]$ *where* $M_1, M_2 \in \mathcal{B}$ □

Now we give formal definition for resistance against semi-free start collision attack. We call this property as semi-free start collision resistance. One can think of many ways to give a formal definition for this property. They are defined both for the compression function and hash function families. Let k be the size of each key (in bits) in the key space $\mathcal{K}$.

Definition 3. *Let $H : \mathcal{K} \times \mathcal{M} \rightarrow \mathcal{Y}$ be a hash function family. Let A be the adversary. Then semi-free start collision resistance is defined as:*

$\mathrm{Adv}_H^{\mathrm{semi-coll}}(A) = \Pr[\, (K, M_1, M_2) \leftarrow A(1^k) : (M_1 \neq M_2) \wedge (H_K(M_1) = H_K(M_2))]$. □

Definition 4. *Let $F : \mathcal{K} \times \mathcal{B} \rightarrow \mathcal{Y}$ be a compression function family. Let A be the adversary. Then semi-free start collision resistance is defined as:*

$\mathrm{Adv}_F^{\mathrm{semi-coll}}(A) = \Pr[\, (K, M_1, M_2) \leftarrow A(1^k) : (M_1 \neq M_2) \wedge (F_K(M_1) = F_K(M_2))]$ *where* $M_1, M_2 \in \mathcal{B}$ □

Now we give formal definition for resistance against pseudo collision attacks. We call this property as pseudo collision resistance. One can think of many ways to give a formal definition for this property. They are defined both for the compression function and hash function families. Obviously this definition forms the strongest definition of collision resistance property.

Definition 5. *Let $H : \mathcal{K} \times \mathcal{M} \rightarrow \mathcal{Y}$ be a hash function family. Let A be the adversary. Then pseudo collision resistance is defined as:*

$\mathrm{Adv}_H^{\mathrm{pseudo-coll}}(A) = \Pr[\, (K_1, K_2, M_1, M_2) \leftarrow A(1^k) : (M_1 \neq M_2) \wedge (K_1 \neq K_2) \wedge (H_{K_1}(M_1) = H_{K_2}(M_2))]$. □

Definition 6. *Let $F : \mathcal{K} \times \mathcal{B} \rightarrow \mathcal{Y}$ be a compression function family. Let A be the adversary. Then pseudo collision resistance is defined as:*

$\mathrm{Adv}_F^{\mathrm{pseudo-coll}}(A) = \Pr[\, (K_1, K_2, M_1, M_2) \leftarrow A(1^k) : (M_1 \neq M_2) \wedge (K_1 \neq K_2) \wedge (F_{K_1}(M_1) = F_{K_2}(M_2))]$ *where* $M_1, M_2 \in \mathcal{B}$ □

In Section 2, near-collision was defined informally. One can formalize near-collision resistance in many ways that have nothing to do with the hamming distance. One can ask how near are the two colliding chaining values to create a collision. Since, no one knows the exact requirement for the near-collisions, we use the term "related-collisions" instead of near-collisions.

Definition 7. *Let $F : \mathcal{K} \times \mathcal{B} \rightarrow \mathcal{Y}$ be a compression function family. Let T be a non-trivial function and A be the adversary. Then related-collision resistance for F is defined as:*

$\mathrm{Adv}_F^{\mathrm{related-coll}}(A) = \Pr[\, K \xleftarrow{\$} \mathcal{K}; (M_1, M_2) \xleftarrow{\$} A(K) : (M_1 \neq M_2) \wedge (F_K(M_1) = T(F_K(M_2)))]$ *where* $M_1, M_2 \in \mathcal{B}$ □

As said in Section 3 of the report, the new security requirement may lead to arbitrary number of formalizations which will be treated as our future work.

6 Conclusion

In this report, we related several sections of the literature in hash functions in the light of recent attacks on hash functions to develop our understanding on the subject. We have given a new security requirement for a cryptographic hash function which may be hard to achieve in practice. We have provided reasons for the claim that the hash functions MD5, SHA-0 and SHA-1 do not consider the protection against pseudo collisions in their design criteria. Further, in this report we doubt whether these hash functions followed the design principle Damgård iterative structure. To support our doubt, we have given the differences between Damgård's original hash function design and the designs that followed it. Finally, formal definitions for resistance against different types of collision attacks are given.

References

1. National Institute of Standards and Technology (NIST) , Computer Systems Laboratory. Secure Hash Standard. Federal Information Processing Standards Publication (FIPS PUB) 180-2, August 2002.
2. Mihir Bellare, Ran Canetti, and Hugo Krawczyk. Keying hash functions for message authentication. In Neal Koblitz, editor, *Advances in Cryptology—CRYPTO '96*, volume 1109 of *Lecture Notes in Computer Science*, pages 1–15. Springer-Verlag, 18–22 August 1996. Full version of the paper is available at `"http://www-cse.ucsd.edu/users/mihir/papers/hmac.html"`.
3. Mihir Bellare, Ran Canetti, and Hugo Krawczyk. Pseudorandom functions revisited: The cascade construction and its concrete security. In *Proceedings of the 37th Annual IEEE Symposium on Foundations of Computer Science, FOCS'96 (Burlington, VT, October 14-16, 1996)*, pages 514–523. IEEE Computer Society, IEEE Computer Society Press, 1996.
4. Eli Biham, Rafi Chen, Antoine Joux, Patrick Carribault, Christophe Lemuet, and William Jalby. Collisions of SHA-0 and Reduced SHA-1. In Ronald Cramer, editor, *Advances in Cryptology - EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 36–57. Springer, 2005.
5. Ivan Damgard. A design principle for hash functions. In Gilles Brassard, editor, *Advances in Cryptology: CRYPTO 89*, volume 435 of *Lecture Notes in Computer Science*, pages 416–427. Springer-Verlag, 1989.
6. Ivan Damgard. Personal communication, 2005.
7. Bert den Boer and Antoon Bosselaers. Collisions for the compression function of MD5. In *Advances in Cryptology – EUROCRYPT '93*, volume 765 of *Lecture Notes in Computer Science*, pages 293–304, 1994.
8. Donald E. Knuth. *The Art of Computer Programming*. Addison-Wesley in Computer Science and Information Processing. Addison-Wesley, 1973.
9. Stefan Lucks. Design principles for iterated hash functions. Cryptology ePrint Archive, Report 2004/253, 2004. `http://eprint.iacr.org/`.
10. Alfred J. Menezes, Paul C. Van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*, chapter Hash Functions and Data Integrity, pages 321–383. The CRC Press series on discrete mathematics and its applications. CRC Press, 1997.
11. Ralph Merkle. One way hash functions and DES. In Gilles Brassard, editor, *Advances in Cryptology: CRYPTO 89*, volume 435 of *Lecture Notes in Computer Science*, pages 428–446. Springer-Verlag, 1989.
12. Bart Preneel. *Analysis and design of Cryptographic Hash Functions*. PhD thesis, Katholieke Universiteit Leuven, 1993.
13. Ron Rivest. The MD5 message-digest algorithm. Internet Request for Comment RFC 1321, Internet Engineering Task Force, April 1992.
14. Ronald Rivest. The MD4 message digest algorithm. In *Advances in Cryptology – CRYPTO '90*, volume 537 of *Lecture Notes in Computer Science*, pages 303–311, 1991.
15. Phillip Rogaway and Thomas Shrimpton. Cryptographic hash-function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance. In Bimal K. Roy and Willi Meier, editors, *Fast Software Encryption (FSE)*, volume 3017 of *Lecture Notes in Computer Science*, pages 371–388. Springer-Verlag, 2004.
16. Xiaoyun Wang, Xuejia Lai, Dengguo Feng, Hui Chen, and Xiuyuan Yu. Cryptanalysis of the Hash Functions MD4 and RIPEMD. In Ronald Cramer, editor, *Advances in Cryptology - EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2005.

17. Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu. 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. Springer, 2005, 14–18 August 2005.
18. 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 17–36. Springer, 2005, 14–18 August 2005.
19. Xiaoyun Wang and Hongbo Yu. How to Break MD5 and Other Hash Functions. In Ronald Cramer, editor, *Advances in Cryptology - EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 19–35. Springer, 2005.
20. Andrew Yao Xiaoyun Wang and Frances Yao. New collision search for SHA-1. Presented at Crypto'05 Rump Session by Adi Shamir, 2005. There is no official paper yet that explains this new result.

A Other way of generating multi-block collisions instead of near-collisions

A special case is the near-collision after first block, but in principle any two different chaining variables could be used to assist in creating a full hash collision after some subsequent message blocks are processed.

So one attack is: Begin with an off-line computation and do steps (1) and (2) They could be done in parallel or in any order, or partly interleaved, etc.... 1) with the known fixed IV as chaining input, collect many pairs of message block input and chaining value output. Sort this list by chaining output and call it "A". 2) collect many triples (input chaining value, message block, output chaining value), sort this list by chaining input and call it "B". Then search for a match in these lists: find "a" from A and "b" from B such that output of element "a" is exactly the input in element "b". Sort the lists by these important data, so the list searching can be done in logarithmic time using well-known list search algorithms (see [8]). If a match is found then (by taking note of the message blocks used in the matching elements) we can instantly construct a 2-(message)-block collision for the two iterated CFs and by extension attack many other real hash collisions.

Given that the CFs have n-bit chaining variables, then if both list A and list B have size $2^{n/2}$, by Birthday Paradox we should expect to find a match.

By using clever algorithms for generating list B (for example distinguished points method) we can have a virtual list size much larger than we can store, at cost of extra operations to search the list.