Constructing Secure Hash Functions by Enhancing Merkle-Damgård Construction

Praveen Gauravaram[1], William Millan[1], Ed Dawson[1] and Kapali Viswanathan[2]

[1] Information Security Institute (ISI)
Queensland University of Technology (QUT)
2 George Street, GPO Box 2434, Brisbane QLD 4001, Australia.
p.gauravaram@isi.qut.edu.au,{b.millan,e.dawson}@qut.edu.au
[2] Technology Development Department, ABB Corporate Research Centre
ABB Global Services Limited, 49, Race Course Road, Bangalore - 560 001, India.
kapaleeswaran.v@in.abb.com

Abstract. Recently multi-block collision attacks (MBCA) were found on the Merkle-Damgård (**MD**)-structure based hash functions MD5, SHA-0 and SHA-1. In this paper, we introduce a new cryptographic construction called **3C** devised by enhancing the **MD** construction. We show that the **3C** construction is at least as secure as the **MD** construction against single-block and multi-block collision attacks. This is the first result of this kind showing a generic construction which is at least as resistant as **MD** against MBCA. To further improve the resistance of the design against MBCA, we propose the **3C+** design as an enhancement of **3C**. Both these constructions are very simple adjustments to the **MD** construction and are immune to the straight forward extension attacks that apply to the **MD** hash function. We also show that **3C** resists some known generic attacks that work on the **MD** construction. Finally, we compare the security and efficiency features of **3C** with other **MD** based proposals.
Keywords: Merkle-Damgård construction, MBCA, 3C, 3C+.

1 Introduction

In 1989, Damgård [2] and Merkle [13] independently proposed a similar iterative structure to construct a collision resistant cryptographic hash function $H : \{0,1\}^* \rightarrow \{0,1\}^t$ using a fixed length input collision resistant compression function $f : \{0,1\}^b \times \{0,1\}^t \rightarrow \{0,1\}^t$. Since then, this iterated design has been called Merkle-Damgård (**MD**) construction which influenced the designs of popular dedicated hash functions such as MD5, SHA-0 and SHA-1. The design motivation of the **MD** construction is that if the compression function f is collision resistant then so is the resultant iterated hash function H.

It is known that, a compression function f secure against the fixed initial value (IV) collisions is necessary but not sufficient to generate a secure hash function H [12, p.373]. The latest multi-block collision attacks (MBCA) on the hash functions MD5, SHA-0 and SHA-1 [1,17–19] prove this insufficiency. These

attacks clearly show that these iterated hash functions do *not properly preserve* the collision resistance property of their respective compression functions with the fixed IV. The MBCA on hash functions leave open the questions; Is it possible to design collision resistant hash functions relying on the collision resistance of the compression function with fixed IV?, Is it possible to design a simple and efficient structures that offer more resistance to MBCA than the **MD** structure?

In this paper, we attempt to answer these questions. Our motivation is to show that while consecutive iterations of the compression function is necessary for the implementation efficiency of a hash function, **the way** the compression function is iterated or used is quite important for the security of the hash function. In this paper, we propose a new mode of operation for the **MD** construction called **3C**. The **3C** hash function processes the intermediate chaining values of the **MD** construction by maintaining a second internal chaining variable. The **3C** construction is the simplest modification of the **MD** construction that one can obtain to improve its security against MBCA.

We show that the **3C** construction is at least as secure as **MD** construction against collision attacks, in particular against MBCA. That is, if there exists an adversary that finds a multi-block collision on **3C** then that adversary would have also found a multi-block collision on the **MD** hash function. In addition, we show that if there exists an adversary that can perform an MBCA on a t-bit **MD** hash function based on a given compression function then the security of **3C** against MBCA instantiated with the same compression function could be as much as 2^t times the security of **MD** against MBCA, depending on the subtle properties of the compression function. We conjecture that this security multiplier of **3C** against MBCA is close to 2^t for any compression function which is secure against single-block collision attacks. We note that multiplier of at least $2^{t/2}$ is sufficient to provide immunity to MBCA. Next, extra memory is added to the **3C** construction and call this variant **3C+** and analyse the difficulty in implementing an MBCA on it compared to **3C**. Analysis for **3C** against known generic attacks [3, 8, 9] is given which applies to **3C+** as well. We found that while Joux's generic attacks [8] work on **3C**, the known generic second preimage attacks [3, 9] found on the **MD** hash function do not work.

In Section 2, we describe **MD** hashing and collision attacks on it. In Section 3, new observations on the MBCA are discussed. In section 4, **3C** is introduced and its analysis against MBCA is covered in Section 5. In Section 6, analysis of **3C** against generic attacks is given and **3C** is compared with some other similar hash function proposals in Section 7. In Section 8, **3C+** is introduced and is analysed against MBCA. The paper is concluded in Section 9.

2 MD hashing and collision attacks

A collision resistant cryptographic hash function H following **MD** structure is a function that hashes a message $M \in \{0,1\}^*$ to outputs of fixed length $\{0,1\}^t$. The specification of H includes the description of the compression function f, initial state value (IV) and a padding procedure [12, 14]. Every hash function

fixes the IV (fixed IV) with an upper bound on the size $|M|$ of the input M. The message M is split into blocks $M_1, \ldots, M_{L-1}$ of equal length b where a block M_L containing the length $|M|$ (**MD** strengthening) [12] is added. Each block M_i is iterated using a fixed length input compression function f computing $H_i = f(H_{i-1}, M_i)$ where $i = 1$ to L and finally outputting $H_{IV}(M) = H_L$ as shown in Fig 1.

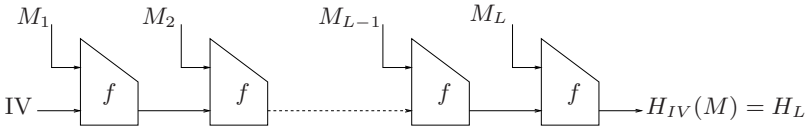

Fig. 1. The Merkle-Damgård (**MD**) construction

Collision attacks on the compression functions:
A hash function H is said to be collision resistant if it is hard to find any two distinct inputs M and N such that $H(M) = H(N)$. For a formal definition see [15]. A hash function H is said to be near-collision resistant if it is hard to find any two distinct inputs M and N such that $H(M) \oplus H(N) = \Delta$ has some small weight. Based on the IV used in finding collisions, collision attacks on the compression functions are classified as follows [12, p.372]:

1. Collision attack: collisions using a fixed IV for two distinct messages (e.g. [16]). We call them *Type 1* collisions.
2. Semi-free-start collision attack: collisions using the same random (or arbitrary) IV for two distinct message inputs(e.g. [5]). We call them *Type 2* collisions.
3. Pseudo-collision attack: free-start collision attack using two different IVs for two distinct message inputs(e.g. [4]). We call them *Type 3* collisions.

Multi-block collision attacks on hash functions:
A multi-block collision attack (MBCA) technique on an iterated hash function finds two colliding messages each of at least two blocks in length. The recent collision attacks on MD5 [19], SHA-0 [1,17] and SHA-1 [18] are multi-block collision attacks where collisions were found by processing more than one message block. Since, by far, most of the possible messages are more than a single block and collisions are distributed randomly, it is fair to say that most collisions that could exist are in fact multi-block collisions. Hence, any result improving the resistance against MBCA is very significant.

3 New Observations on multi-block collision attacks

This section aims at developing the understanding of the multi-block collision attacks on hash functions by linking several parts of the literature. We observed

that multi-block collision attacks on hash functions can further be classified into three categories based on the manner in which the compression functions are attacked. This leads to choosing particular message block formats that result in an MBCA. For example, 2-block collision attacks can be classified into three types as shown in Table 1 based on the message formats chosen.

Table 1. Classification for 2-block collision attacks

MBCA Type	Message formats
MBCA-1	(M_1, M_2) and (M_1, N_2)
MBCA-2	(M_1, M_2) and (N_1, M_2)
MBCA-3	(M_1, M_2) and (N_1, N_2)

While the 2-block collision attacks on MD5 [19] and SHA-1 [18] belong to MBCA-3 category, the 2-block collision attack on SHA-0 [17] belongs to an MBCA-1 category[1]. In an MBCA-3, near-collisions found after processing first message blocks were converted to full collisions as was demonstrated on MD5 and SHA-1. The *Type 1* collisions were (reportedly) hard to find for the single compression functions of these hash algorithms. For example, the attacks on MD5 and SHA-1 use near-collisions obtained after processing the first distinct message blocks (M_1, N_1) as a tool to find collisions for the second distinct message blocks (M_2, N_2). This technique can be generalized to more than two blocks as the 4-block collision attack on SHA-0 [1]. Similarly, 2-block MBCA-1 and MBCA-2 attack techniques can be generalized to more than two blocks. For example, in an MBCA-1 technique [17], a few initial message blocks to be processed can be chosen to be the same to satisfy certain conditions required in the attack followed by the processing of two different message blocks that give a collision.

We note that in a 2-block MBCA, collisions found on the second blocks are basically a *special case* of *Type-3* collisions for the compression function as these collisions require processing of two equal (MBCA-1) or different blocks (MBCA-2 and MBCA-3) using the fixed IV of the hash function. That is, a 2-block MBCA-3 on the **MD** hash function is a combination of a near-collision and a *special Type-3* on the compression function. The collision on the second compression function is a *special Type-3* as it requires a particular nearly collided value obtained based on certain conditions essential for the attack as inputs for the second block messages. In addition, near-collisions do not need to begin after processing message blocks based on the fixed IV of the hash function. They can also be due to an arbitrary chaining value when the attacker chooses the same blocks initially and starts an MBCA-3 after processing those same initial blocks.

[1] The first full 4-block collision attack on SHA-0 [1] also belongs to an MBCA-3 category except that differences in the message blocks span over more than two blocks.

Hence multi-block difference collisions, whether they start from the fixed IV or arbitrary chaining values are clearly a chain of *special Type 3* collisions.

From these observations on MBCA, it is clear that the designers of MD5, SHA-0 and SHA-1 *have not considered security of the compression functions of these hash functions against special Type-3 collisions in their design criteria.* Preneel pointed out more than a decade back [14] that most hash functions are not designed to meet this criteria. Note that SHA-1 did not exist then. Even Damgård's [2] proof implicitly notes that the necessity of *special Type-3* collision resistance for the compression functions. In addition, to attain *Type-3* collisions, the two IVs do not have to be significantly different as suggested in [12, p.372]. For example, the two IVs in the *Type-3* collision attack on the compression function of MD5 [4] differ in *only* 6 bits. From the known attacks on hash functions, we derived Table 2 assuming that if the compression function is not *Type-1* collision resistant then it is neither *Type-2* nor *Type-3* collision resistant. The sign "-" in the Table 2 shows does not apply.

Table 2. Resistances of some compression functions

Compression function	*Type-1*	*Type-2*	*Type-3*	*Special Type-3*
MD4	NO [16]	NO	NO	-
MD5	YES	NO [5]	NO [4]	NO [19]
SHA-0	YES	NO [19]	YES	NO [1]
SHA-1	YES	YES	YES	NO [18]
RIPEMD	NO [16]	NO	NO	-
HAVAL-128	NO [16]	NO	NO	-

4 The 3C construction: An Enhanced MD construction

The **3C** construction is shown in Fig. 2 and 3. This structure has an *accumulator* XOR function iterated in the *accumulation chain* (whose chaining value is denoted by u_i in Fig. 3) and a compression function f (f, for example, is the compression function of MD5 or SHA-1) iterated in the *cascade chain* (whose chaining value is denoted by w_i in Fig. 3) exactly as in the **MD** construction. Clearly, **3C** is a very simple and efficient modification to the **MD** construction. One economic benefit of our proposal is that any software currently implementing an **MD**-style hash function can be very simply altered to adopt the **3C** structure, without altering the underlying compression function.

3C hashing process: For $i = 1$ to L, let w_i and u_i be the chaining values in the *cascade chain* and *accumulation chain* respectively. Then, as in the **MD** hash, for $i = 1$ to L, $w_i = f(w_{i-1}, M_i)$ where $w_0 = IV$ and $u_1 = w_1$. In the *accumulation chain*, for $i = 2$ to L, $u_i = u_{i-1} \oplus w_i$. The result u_L in the *accumulation chain* is denoted with Z. An extra compression function f, denoted by g, is added at

the end and the hash result of **3C** is $g(\overline{Z}, w_L)$. To process one block data, the compression function is executed three times; first to process the data block, next to process the padded block (**MD** strengthening) and finally the block $\overline{Z}$ formed in the *accumulation chain* as shown in Fig 3. If the size of data is less than block size b of f then zeros are appended to the data to make a b-bit data block.

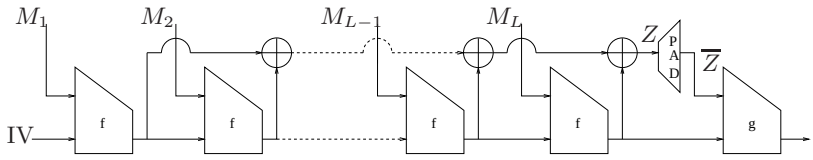

Fig. 2. The **3C**-hash function

5 Security analysis of the 3C hash function

In this section, we investigate the security of **3C** against single-block and multi-block collision attacks. We conclude that the security of **3C** against single-block collision attacks is upper bounded by the collision security of the compression function and its security against MBCA is not less than that on **MD**. Fig 3 is used to explain the analysis.

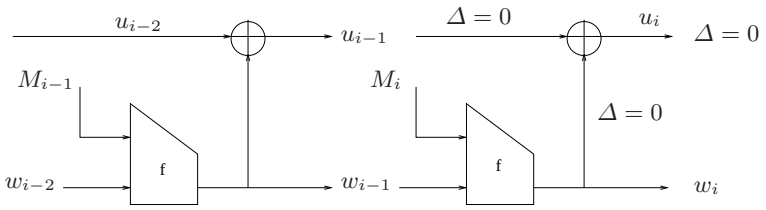

Fig. 3. Creating an internal collision for **3C**

Consider a **3C** hash function H. Consider two distinct messages $M \neq N$ of same length L (including padding) such that $H(M) = H(N)$ is the result of a collision on **3C**. The messages M and N are expanded to sequences $(M_1, \ldots, M_L) \neq (N_1, \ldots, N_L)$ where the last data blocks are the padded blocks containing the length L of the messages. We denote by (H_i^M, H_i^N) and (u_i, v_i) (for $i = 1$ to L), the internal hash values obtained on the *cascade chain* and *accumulation chain* while computing $H(M)$ and $H(N)$ respectively. We denote (u_L, v_L) by (Z_M, Z_N) and $\overline{Z}_M = \mathrm{PAD}(Z_M)$, $\overline{Z}_N = \mathrm{PAD}(Z_N)$. All possible types of collisions on H are given in Definition 1.

Definition 1. *Every collision on H takes one of the following forms:*

1. *Terminal/Final collisions on H: They involve one of the following cases:*
   - $H_L^M \neq H_L^N$ *and* $\overline{Z}_M \neq \overline{Z}_N$ *with* $g(H_L^M, \overline{Z}_M) = g(H_L^N, \overline{Z}_N)$
   - $H_L^M = H_L^N$ *and* $\overline{Z}_M \neq \overline{Z}_N$ *with* $g(H_L^M, \overline{Z}_M) = g(H_L^N, \overline{Z}_N)$
   - $H_L^M \neq H_L^N$ *and* $\overline{Z}_M = \overline{Z}_N$ *with* $g(H_L^M, \overline{Z}_M) = g(H_L^N, \overline{Z}_N)$
2. *Internal collisions on H: $H_L^M = H_L^N$ and $\overline{Z}_M = \overline{Z}_N$ implies $g(H_L^M, \overline{Z}_M) = g(H_L^N, \overline{Z}_N)$.* □

Definition 2. *A compression function $f : \{0,1\}^b \rightarrow \{0,1\}^t$ is Type-1* (resp. *Type-2, Type-3*) *collision resistant if the best possible collision attack on it using fixed IV* (resp. *arbitrary IV, different IVs*) *is the birthday attack which takes about $2^{t/2}$ operations of f. For sufficiently large t, it is computationally infeasible to perform this attack.*

Lemma 1. *Against single block collision attacks, the security of* **3C** *is exactly equal to the security of* **MD** *when both the constructions are instantiated with the same f.*

Proof: By inspection of **3C** structure in Fig 3, it is clear that it contains **MD** construction in it. Hence, an adversary that is able to find a single block collision for the f-function is able to construct a collision for the hash function with virtually no additional effort. It follows that the collision security of **3C** is upper bounded by the collision security of the f-function. □

Apart from single block collision attacks, the only other approach to find collisions for **3C** is to use multi-block messages. This invites the opportunity to use messages with different lengths. For two messages with the same length, an internal collision for **3C** gives an actual collision for **3C**. However, for two messages of different lengths, in general, this is not the case due to the different padding strings used as a virtual message block in the second last iteration of the compression function. Thus the security analysis of **3C** can be restricted to considering internal collisions generated by pairs of messages with the same length. We now examine the nature of internal collisions for **3C**.

Lemma 2. *To get an internal collision on* **3C** *at iteration i, it is required that a collision in the accumulation chain exists at iteration $i-1$.*

Proof: An internal collision in **3C** at iteration i is a simultaneous collision in the *accumulation chain* and *cascade chain* at iteration i. For messages M and N to collide on the *cascade chain* at iteration i, the condition that $H_i^M \oplus H_i^N = 0$ must be satisfied. Now for an internal collision on **3C**, the condition that $(H_i^M \oplus H_i^N) \oplus (u_{i-1} \oplus v_{i-1}) = 0$ must be satisfied. This condition will occur only when $u_{i-1} \oplus v_{i-1} = 0$, which is basically a collision in the *accumulation chain* at iteration $i-1$. □

Remark: A collision in the accumulation chain at iteration $i-1$ is achieved by creating a sequence of **MD** chain differences where the chaining difference in the **MD** chain at iteration $i-1$ is the XOR sum of all the previous differences in the **MD** chain until the iteration $i-2$.

Lemma 3. *Assuming the existence of a collision in the accumulation chain at iteration $i-1$, it requires a single-block special Type-3 collision attack on f to create an internal collision in* **3C** *at iteration i.*

Proof: By inspection of **3C** structure in Fig 3, a special Type-3 collision attack must be performed on the f-function at iteration i. It is a *special Type-3* collision attack as the attacker must use the internal chaining values of the *cascade chain* at iteration $i-1$ that created a collision in the *accumulation chain* as inputs to get a collision at iteration i. This is equivalent to performing a single-block *special Type-3* collision attack on f at iteration i. □

Now we can consider the above process in two ways: as two separate single block collision attacks, or as a multi-block collision attack on the MD-chain with an extra t-bit requirement. We ignore the first option as the single block collision security of the f-function is already an upper bound for the collision security of 3C from Lemma 1. The second case can be achieved in either of the following two ways:

1. Assume the existence of an MBCA on the **MD**-chain when it is instantiated with some given compression function. Then the MBCA on the **MD**-chain has to be repeated until the internal chaining differences on the **MD**-chain happen to produce the required collision on both the accumulation and cascade chains. This option requires repeating the attack at most 2^t times under an assumption that the internal chaining differences are uniformly distributed.
2. Devise an entirely new MBCA for the **3C** when instantiated with some given compression function satisfying some conditions on the differences.

Now, the above two cases result in the following theorems:

Theorem 1 *If there is an MBCA on the* **MD** *construction instantiated with a given f then the security of* **3C** *instantiated with the same f against an MBCA is at most 2^t times the security of* **MD** *against MBCA.*

Proof: To obtain a collision in the *accumulation chain* required by Lemma 2, an MBCA on the **MD** chain must be repeated, where each attempt succeeds with probability 2^{-t}. That is, the security of **3C** against MBCA is some multiple ($[1, 2^t]$) of the security against MBCA for the **MD**. □

The difficulty of providing a tight quantitative analysis for **3C** against MBCA prevents a more precise formal proof for the practical collision security of **3C** at this stage leading to the following conjecture.

Conjecture: From the above analysis, we conjecture that the improvement in the security of **3C** against MBCA is close to 2^t over the security of **MD** against MBCA.

Theorem 2 *The security of* **3C** *against an MBCA is not less than the security of* **MD** *against MBCA.*

Proof: Every internal collision for **3C** contains within it a collision for **MD**. There exist collisions for **MD** that are not internal collisions for **3C**. Thus the security of **3C** against MBCA is lower bounded by the security of **MD** against MBCA. □

Remarks: While at least two blocks must be processed to find a multi-block collision on **MD**, at least three blocks must be processed to create a multi-block collision on **3C**. For example, the difference pattern $(0, \Delta, \Delta, 0)$ which creates a collision on **MD** based on a given f, will also create a collision on **3C** based on the same f. But note that its reduced pattern $(0, \Delta)$ would create a collision for **MD** but not for **3C**. In Section 8, we propose a construction called **3C+** with similar properties to **3C** as an improvement of **3C** for more protection against MBCA.

6 Security analysis of 3C against known generic attacks:

Analysis against Joux attacks:

Joux [8] described a generic multicollision attack on the **MD** hash where constructing 2^d-collisions costs d times as much as building ordinary 2-collisions. This attack can be used as a tool to find multi (2^{nd}) preimages very effectively on the **MD** hash. We note that these attacks work on **3C** as effectively as they are on the **MD** hash. Following [10], our adversaries are probabilistic algorithms and we focus on the expected running time. Running time is described asymptotically. We use the symbol O for the "expected running time is asymptotically at most".

In a multicollision attack on **3C**, the attacker finds collisions on every function f in the *cascade chain* (for example using the birthday attack) that would result in a collision at the subsequent point of the XOR operation in the *accumulation chain*. If the function f in the *cascade chain* of **3C** is modeled as a random oracle, as an upper bound, the total complexity to find 2^d-collisions on **3C** is $O(d * 2^{t/2})$.

We note that the attack technique used to find D-way (2^{nd}) preimages on the **MD** hash for a given hash value works on **3C** as well. For example on **3C**, the attacker first finds D-collisions on d-block messages $M^1, \ldots, M^{2^d}$ with $H_d = H(M^1) = \ldots = H(M^{2^d})$ with a complexity of $O(d * 2^{t/2})$. Then she finds the block M_{d+1} such that the execution of the last two compression functions would result in the given digest Y. The later task takes time $O(2^t)$ as the last two compression functions are treated as a single component. Hence the total cost of finding D-preimages for **3C** is $O(d * 2^{t/2} + 2^t)$. To find D-2^{nd} preimages for a given message M, the attacker first computes the hash $H(M)$ of the message M and then finds D-preimages as explained above that all collide to $H(M)$.

Analysis against second-preimage attacks:

Dean [3] has demonstrated that for hash functions with fixed point compression functions, it would cost less than 2^t effort to find second preimages. Kelsey and Schneier [9] have expanded this result using Joux multicollision finding a technique to find second preimages for hash functions based on any compression

function for an effort less than 2^t. Both these attacks use the notion of *expandable messages*- patterns of messages of different lengths that all process to internal hash values without considering **MD** strengthening. Following [9], an (a, b)-expandable message will take on any length between a and b message blocks.

For a compression function $H_i = f(H_{i-1}, M_i)$, a fixed point is a pair (H_{i-1}, M_i) such that $H_{i-1} = f(H_{i-1}, M_i)$. The compression functions of many hash functions such as MD5 and SHA-1 are Davies-Meyer designs with a block cipher operating in a feed-forward mode. For these compression functions, there exists one and only one fixed point for every message block. For a t-bit hash function with a maximum of 2^d blocks in its messages, using fixed points it costs about $2^{t/2+1}$ compression function computations to find $(1, 2^d)$-expandable messages [9]. In the **3C** design, since the chaining state is twice as large as the hash value, a fixed point is defined for both the chains and this is obtained for any message block M_i, *only when* $f(0, M[_i) = 0$ and this occurs with a probability of 2^{-t}. Hence having fixed points for the compression functions will not assist in finding second preimages for less than 2^t work on the **3C** design.

It was demonstrated in [9] that finding a $(d, d + 2^d - 1)$ expandable message for any compression function with t-bit state takes only $d \times 2^{t/2+1}$ effort. The procedure involves first finding colliding pair of messages, one of one block and the other of $2^{d-1} + 1$ blocks starting from the initial state of the hash function. Then using the collided state as the starting state, collision pair of length either 1 or $2^{d-2}+1$ is found and this process is continued until a collision pair of length 1 or 2 is reached. It was shown in [9] that applying this generic expandable message finding algorithm to find the second preimage for a message of $2^d + d + 1$ -block length message costs $d \times 2^{t/2+1} + 2^{n-d+1}$ compression function computations. When this attack technique is applied on **3C**, a collision at both the chains is required and this costs an effort of 2^t at every stage as the size of the internal state is twice that of the hash size.But if different parts of the internal state of **3C** are attacked separately, **3C** might not resist the second preimage attack.

7 Comparison of 3C with other hash function proposals

Ferguson and Schneier [6] proposed double-hashing scheme $H_{IV}(H_{IV}(x))$ to prevent straight-forward length extension attacks. It is obvious that multi block collision attacks work on this nested construction as effectively as they are on **MD**. As on the **MD** hash, 2^d-collisions can be found on their scheme with a complexity of $O(d.2^{t/2})$ and finding 2^d-(2^{nd}) preimages would take time $O(d.2^{t/2} + 2^t)$. Gauravaram *et al.* [7] proposed CRUSH hash function based on iterated length halving technique as an alternative for **MD** hash function well before the invention of MBCA on MD5 anticipating the single point of failure of hash functions in the MD family. CRUSH is immune against extension attacks and resists known MBCA techniques.

Lucks [10] proposed wide-pipe and its special case double-pipe hash designs as failure-tolerant designs showing that they provide more resistance against generic attacks [8] than the **MD** hash. While wide-pipe hash maintains more

internal state than the hash size t using larger compression functions, double-pipe hash maintains twice the hash size as the internal state size by employing one single t-bit compression function used twice in parallel for each message block. In contrast, one could see **3C** structure as special cases of wide-pipe hash and is optimally efficient as no new large compression function needs to be designed for its execution. The wide-pipe, **3C** and double-hashing proposals resist the straight-forward length extension attacks which is a well-known weakness of the **MD** hash function. Informally, given the digest H of the message M, it is straight forward to compute N and H' such that $H' = H(M||N)$ even for unknown M but for known $|M|$. The attack uses $H(M)$ as the internal hash value to compute $H(M||N)$. All these hash functions provide $t/2$-bit level of security against straight forward extension attacks as long as their design criteria is satisfied; for example, wide-pipe hash requires processing of the compression function with an internal state at least twice the size of the hash value, **3C** requires at least three calls to the compression function. Note that **3C** prevents extension attacks with out using large compression function as in the wide-pipe hash.

While the wide-pipe and double-pipe hash functions are designed to provide more resistance against generic attacks, **3C** and **3C+** are enhancements of **MD** resisting recent multi-block collision attacks on the **MD** based hash functions. In addition, one can combine the wide-pipe hash and the **3C** construction to attain a hybrid construction called 3CWP (see Fig 4) attaining additional protection against both the generic attacks and MBCA.

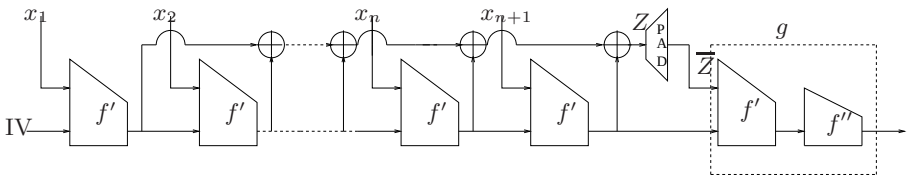

Fig. 4. The **3CWP** hybrid construction

From the performance point of view, **3C** is slightly more expensive than **MD** especially when it is used to process short messages as the former requires at least three iterations of the compression function to process an arbitrary length message. To process 1-block (resp. 2-block) message, the running time of the **3C** is twice (resp. 1.5 times) that of **MD**. On an Intel Pentium 4 3.2GHz processor, **3C** based on the compression of MD5, incurs about 0.36% overhead and **3C** with the compression function of SHA-1 incurs about 0.27% overhead when these functions are used to process long messages. **3C** requires an extra iteration of the compression function similar to the double hashing proposal [6] and is as efficient as this scheme for the processing of long messages and unlike the double hashing scheme, **3C** is a single hashing scheme.

8 The 3C+ construction: An enhanced 3C construction

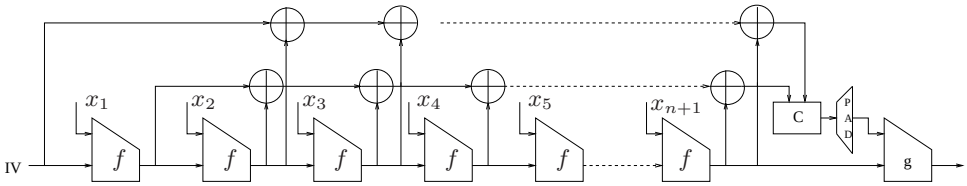

Fig. 5. The **3C+** hash construction

Fig 5 shows the **3C+** construction where a third internal chain called *final chain* has been added on top of the *cascade* and *accumulation chains* of **3C**. In **3C+**, we call *accumulation chain* as the *middle chain*. The *final chain* in **3C+** slightly differs in the way it accumulates data from the *accumulation chain* of **3C** as it accumulates data from the *cascade chain* but the accumulation starts after processing the second message block. The final compression function f (denoted by g in Fig 5) takes as "message" the concatenation of the accumulated data from the *middle* and *final* chains, appropriately padded.

To find a multi-block collision on **3C+** the attacker has to get collisions simultaneously on all the three chains. To create a simultaneous collision on all the three chains at iteration i, the chaining difference on the *cascade chain* at iteration $i-1$ (say Δ_{i-1}) must cancel the differences accumulated in the *middle* and *final* chains till the iteration $i-2$. That is, the *middle* and *final* chains must maintain an equal difference Δ_{i-1} until the iteration $i-2$ of the function f for a cancellation at iteration $i-1$. This is impossible if *middle* and *final* chains do not start with the same difference before iteration $i-2$. That is, if an attacker finds two colliding messages that have identical first message blocks, then these two chains begin with a difference zero. This implies that a minimum of four message blocks need to be processed to find an MBCA on **3C+**. For example, the pattern $(0, 0, \Delta, \Delta, 0)$ creates a simultaneous collision on all the chains after processing four blocks. From this discussion, it is clear that **3C+** structure demands crafting and maintaining the same difference in both the *final* and *middle* chains until a multi-block collision is found.

Finally, we note that if the input to the *final chain* is taken from the *middle chain* rather than from the *cascade chain*, the difference pattern $(0, \Delta, \Delta, 0)$ that creates a collision on **MD** and **3C** will also create a collision on this modified construction of **3C+** and the MBCA security of this construction is lower bounded by **3C** and **MD**. Note this pattern does not create a collision on **3C+**. This justifies our statement that the security of a hash function depends on the **way** the compression function is used in constructing a hash function.

We note that one can construct many variants for our **3C** and **3C+** designs by replacing XOR functions with any function in such a way that the new construction is at least as secure as **MD**. Here we provide two examples. If the XOR

function in **3C** is replaced with the function f then this modified construction resembles double pipe hash in some way and is less efficient than **3C** against MBCA. The chaining values on the *cascade chain* after the second iteration of f will be used as data block (by appending 0's to chaining values) inputs for the compression functions in the accumulation chain. Clearly the amount of control that an attacker can have on these blocks to create an MBCA is less than **3C**. A slight variant of **3C+** whose cost relative to **3C** bound to be nearly as small as XOR can be designed by interpreting the t-bit chaining value in the *final chain* as an element of GF(2^t) and multiplying it by 2 at each step. If the *final chain* accumulation process starts after the first iteration of f unlike in **3C+** then this structure results in a *final chain* accumulation equation that resembles Galois-Carter-Wegman structure of GHASH in [11].

9 Conclusion

The recent cryptanalysis of hash functions MD5, SHA-0, SHA-1 exploited the **MD** iterative structure of these hash functions using multi-block collision search techniques. The proposed **3C** and **3C+** variants to the **MD** construction are at least as resistant as **MD** against MBCA. The constructions can be implemented by simple adjustments to the existing **MD**-style implementations. This paper is the first paper to introduce solutions to MBCA on **MD** based hash functions since they were first identified by Wang *et al* on MD5 [16]. This paper will not and should not be taken as the last step but should be regarded by the Crypto community as the first step to improve the general design of hash functions.

Acknowledgments:
Thanks to anonymous reviewers of ACISP 2006 for many useful comments on several aspects of the paper and their valuable insights. Many thanks to Suganya Annadurai, Paulo Barreto, Matt Henricksen, John Kelsey, Lars Knudsen, Adrian McCullagh, David McGrew, Juanma González Nieto, Vincent Rijmen and Søren Thomsen for their encouragement and comments on the analysis, design and performance aspects presented in the earlier drafts.

References

1. 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.
2. 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.
3. Richard Drews Dean. *Formal Aspects of Mobile Code Security.* PhD thesis, Princeton University, 1999.
4. Bert denBoer and Antoon Bosselaers. Collisions for the compression function of MD5. In T. Helleseth, editor, *Advances in Cryptology — Eurocrypt '93*, volume 765 of *Lecture Notes in Computer Science*, pages 293–304, Berlin, 1994. Springer-Verlag.

5. Hans Dobbertin. Cryptanalysis of MD5 compress. Presented at the rump session of Euro Crypto'96 Rump Session, 1996.
6. Niels Ferguson and Bruce Schneier. *Practical Cryptography*, chapter Hash Functions, pages 83–96. John Wiley & Sons, 2003.
7. Praveen Gauravaram, William Millan, and Lauren May. CRUSH: A New Cryptographic Hash Function using Iterated Halving Technique. In *Proceedings of the workshop on Cryptographic Algorithms and their uses*, pages 28–39, Goldcoast, Australia, July 4–5 2004.
8. Antoine Joux. Multicollisions in Iterated Hash Functions. Application to Cascaded Constructions. In Matt Franklin, editor, *Advances in Cryptology-CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 306–316, Santa Barbara, California, USA, August 15–19 2004. Springer.
9. John Kelsey and Bruce Schneier. Second Preimages on n-bit Hash Functions for Much Less than 2n̂ Work. In Ronald Cramer, editor, *Advances in Cryptology - EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 474–490. Springer, 2005.
10. Stefan Lucks. A Failure-Friendly Design Principle for Hash Functions. In Bimal Roy, editor, *Advances in Cryptology - ASIACRYPT 2005*, volume 3788 of *Lecture Notes in Computer Science*, pages 474–494. Springer-Verlag, 2005.
11. David McGrew and John Viega. The Galois/Counter Mode of Operation (gcm). NIST special publication, National Institute for Standards and Technology.
12. 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.
13. 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.
14. Bart Preneel. *Analysis and design of Cryptographic Hash Functions*. PhD thesis, Katholieke Universiteit Leuven, 1993.
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, Dengguo Feng, Xuejia Lai, and Hongbo Yu. Collisions for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD. Cryptology ePrint Archive, Report 2004/199, 2004. `http://eprint.iacr.org/`.
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.