Characterizing Padding Rules of MD Hash Functions Preserving Collision Security

Mridul Nandi

National Institute of Standards and Technology
`mridul.nandi@gmail.com`

Abstract. This paper characterizes collision preserving padding rules and provides variants of Merkle-Damgård (MD) which are having less or no overhead costs due to length. We first show that suffix-free property of padding rule is necessary as well as sufficient to preserve the collision security of MD hash function for an arbitrary domain $\{0,1\}^*$. Knowing this, we propose a simple suffix-free padding rule padding only $\log|M|$ bits for a message M, which is less than that of Damgård's and Sarkar's padding rules. We also prove that the length-padding is not absolutely necessary. We show that a simple variant of MD with 10^d-padding (or any injective padding) is collision resistant provided that the underlying compression function is collision resistant after chopping the last-bit. Finally, we design another variant of MD hash function preserving all three basic security notions of hash functions, namely collision and (2nd) preimage. This is an improvement over a recently designed (SAC-08) three-property preserving hash function in terms of both salt size and efficiency.

Keywords : MD hash function, padding rule, suffix-free, collision resistant.

1 Introduction

Hash function has become an essential object in many cryptographic protocols [4] particularly in signature schemes [2, 6, 11]. It takes an input from a message space $\mathcal{M}$ (usually $\{0,1\}^*$ or $\{0,1\}^{\leq 2^s-1}$ for some s) and it outputs a t-bit string for a fixed t. The hash function plays role of preprocessor in many applications so that one can work with t-bit $H(M)$ instead of an arbitrary sized M, which essentially helps us to keep design of a protocol simple and efficient. In most cases, securities of these protocols rely on the *collision resistance* property (it is hard to find two different messages with same hash value) of the hash function. The most popular design of a hash function is Merkle-Damgård [5, 10] or MD hash function where a compression function $f : \{0,1\}^{b+t} \to \{0,1\}^t$ is designed first. Given a message, some additional bits may be padded to it so that it can be partitioned into several blocks of size b. The compression function is then sequentially applied to an initial value and to all blocks of the padded message. It seems difficult to design a hash function from scratch, based on only simple logical or/and arithmetical operations, which can provide absolute collision security (or provable collision security like discrete log based hash function [7]). However, it is well known that the MD hash functions with length strengthening padding which includes length of the message, preserves collision security i.e. the hash function is collision resistant if so is the compression function. So we are able to at least transfer the infeasibility assumption of a hash function to a smaller domain compression function and hence only task remains to design a good compression function.

Padding rule is essential for MD hash function (proposed by both Damgård [5] and Merkle [10] independently in crypto-1989). However they used different padding rules. Merkle's padding rule can not handle arbitrary length messages. Sarkar [16] recently introduced a padding rule which can handle arbitrary messages and the number of padding bits is $\mathrm{O}(\log |M| \log^* |M|)$ for some slowly growing function $\log^*$ defined in [16]. This is asymptotically less that that of Damgård's padding rule where $\mathrm{O}(|M|)$ bits are padded. Note, if the size of padded bits is more, it may cost more invocations of the underlying compression function. So, in terms of efficiency of hash function, one should try to keep size of pad as small as possible. Any arbitrary padding may not be good as the hash function is desired to preserve collision security. Clearly, injectivity is a basic requirement of a padding rule. A padding rule is said to preserve collision security for MD if the MD hash function with this padding rule preserves collision security. So, it is worthwhile to characterize all padding rules preserving collision security.

Our Contribution. In this paper, we first show that *suffix-free property is both necessary and sufficient to preserve collision security for MD hash function.* Damgård in [5] mentioned prefix-free padding rules. Stinson [17], Bellare and Rogaway [3] mentioned suffix-free property while proving collision preserving of particular padding rules. Even though sufficiency of suffix-free padding rule seems intuitive, we do not know any paper proving it. Some observations on Merkle's padding rule can be found in [8]. On the other hand, the necessity of the suffix-free property is non-trivial. We propose *a simple efficient suffix-free padding rule, padding* $\mathrm{O}(\log(|M|))$ *bits, which can handle arbitrary messages.* We see a comparison of new padding rules with known padding rules in Table 1.

Let a t-bit compression function f is collision resistant in the first $(t-1)$-bits (i.e. collision resistant after chopping the last bit). We show that a simple variant of MD hash function (converting 0^t chaining value (if any) into $0^{t-1}1$) without any length-padding (any injective padding such as 10^d-padding works) is collision resistant. We actually prove a stronger statement which says that any collision of the new hash function reduces to either a collision of f or a collision of the first $(t-1)$ bits of f **with the collision value** 0^{t-1}. Thus, we are able to remove overhead costs due to length.

We also provide *an improved three property (collision, (2nd) preimage) preserving salted hash function which is a variant of MD hash function and is more efficient than recently proposed hash function* [1] in terms of salt size.

Organization of the paper. We first give an overview of the security notions of a hash function and padding rules of MD hash functions in section 2. In section 3, we characterize the collision preserving padding rules for any fixed initial value. We also have provided simple examples of padding rule in the same section. In the following section, we prove a simple variant can completely avoid length-padding and still have collision security under a reasonable additional assumption. In section 5, we study an improved variant of BCM (backward chaining mode) hash function which preserves all basic three security notions of a hash function.

2 Overview of MD Hash Function, Padding Rule

A hash function $H : \mathcal{M} \to \{0,1\}^t$ is called a *collision resistant* [15, 18] hash functions if *it is "hard" to find a collision pair* (M, M') *i.e.,* $M \neq M'$ *such that* $H(M) = H(M')$. We define collision-advantage of an algorithm A as

$$\mathbf{Adv}_H^{\mathrm{coll}}(A) := \mathsf{Pr}[A \to (M, M') : H(M) = H(M'), M \neq M']$$

where probability is calculated over the random coins of A. Informally, a hash function is called collision resistant if, for any efficient algorithm A, the collision advantage of A for H is negligible. Unfortunately, we can not rule out the existence of an efficient collision finding algorithm A

outputting (M, M') which is eventually a collision pair of the hash function H. But nobody may know or write down this algorithm based on our current knowledge. Therefore, we can say that a hash function is *collision resistant* if no efficient collision finding algorithm is known for it. Rogaway formalized this approach by introducing human ignorance model [14]. Keeping this in mind, we use the following definition of preserving properties of hash securities.

Definition 1. *A hash family $H := \{H_{\mathrm{IV}}\}_{\mathrm{IV} \in \{0,1\}^t}$ based on a compression function f is said to preserve (ϵ, ϵ')-collision security if given an efficient algorithm A with at least ϵ collision advantage for H, we can construct (write down its code modulo the subroutine A) an efficient algorithm A' with at least ϵ' collision advantage for f.*

Merkle-Damgård Hash function. Markle-Damgård or MD hash function has three basic components namely, (1) an underlying compression function $f : \{0,1\}^{b+t} \to \{0,1\}^t$ for some $b > 0$, (2) an initial value $\mathrm{IV} \in \{0,1\}^t$ and (3) an easily computable padding rule $\mathsf{pad} : \mathcal{M} \to (\{0,1\}^b)^+$ for some message space $\mathcal{M}$. We define the classical iterated function $f^+_{\mathrm{IV}} : (\{0,1\}^b)^+ \to \{0,1\}^t$ as $f^+_{\mathrm{IV}}(M_1, \cdots, M_\ell) = f(f(\cdots f(\mathrm{IV}, M_1), \cdots), M_\ell)$, $M_1, \cdots, M_\ell \in \{0,1\}^b$. The MD hash function $\mathsf{MD}^f_{\mathrm{IV},\mathsf{pad}}$ is defined as the composition of the following maps:

$$\mathcal{M} \xrightarrow{\mathsf{pad}} (\{0,1\}^b)^+ \xrightarrow{f^+_{\mathrm{IV}}} \{0,1\}^t \quad \text{(mapping as a function)}$$

Thus, for all $M \in \mathcal{M}$., $\mathsf{MD}^f_{\mathrm{IV},\mathsf{pad}}(M) = f^+_{\mathrm{IV}}(\mathsf{pad}(M))$. An illustration is given in figure 1. Padding rule is essential to make the message size compatible with the domain of f^+_{IV} as well as to keep the hash function collision preserving. In this paper, we will mainly study padding rules of MD and its different variants.

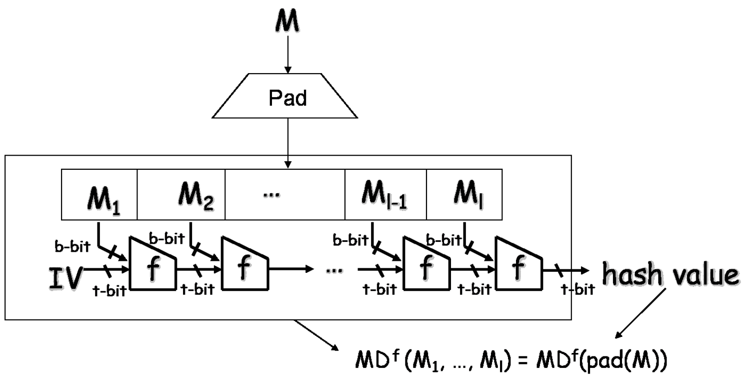

Fig. 1. The classical sequential iteration of a compression function.

Padding Rules. The simplest possible (must be injective) padding rule is $\mathsf{pad}_0(M) = M \| 10^d$ where d is the smallest nonnegative integer such that $|M| + 1 + d$ is a multiple of b. However, pad_0 may not be sufficient to show the collision resistance property of MD under the only assumption that f is collision resistant. We show that there is a collision resistant compression function f with a collision pair for $\mathsf{MD}^{f}_{\mathrm{IV},\mathsf{pad}_0}$, the MD hash based on f using pad_0 padding rule. Moreover, the same result is true for any other "simply defined" padding rule which is not suffix-free (see Theorem 1 in section 3.3).

Definition 2. *Let $X, Y \in \{0,1\}^*$. We call X a suffix of Y if there exists a binary string Z such that $Y = Z\|X$. A padding rule* pad *is called* suffix-free *if, for any $M \neq M'$,* $\mathsf{pad}(M)$ *is not a suffix of* $\mathsf{pad}(M')$.

Here, we list some known padding rules. In crypto-1989, Damgård and Merkle independently proposed the classical MD iteration, but with different padding rules. Besides Merkle's and Damgård's padding rule, Sarkar defined a generalized version of Merkle's padding rule which has message space $\{0,1\}^*$ unlike Merkle's padding rule where message space is $\{0,1\}^{2^s-1}$ for some fixed s.

1. **Merkle's padding rule :** The message space $\mathcal{M} = \{0,1\}^{2^s-1}$ for some fixed s (we usually choose $s = 64$ or 128) and the padding rule is defined as

$$\mathsf{pad}_{\mathrm{merk}}(M) = M \parallel 10^d \parallel \mathrm{len}_s(M)$$

   where d is the smallest nonnegative integer such that $d + (|M| + 1 + s)$ is a multiple of b and $\mathrm{len}_s(M)$ represents the s bit binary representation of $|M|$. The classical MD hash function uses Merkle's padding rule. For example, SHA-2 hash function is nothing but MD hash function with Merkle's padding rule for $s = 64$. Note that Merkle's padding rule can hash messages of maximum possible size $2^s - 1$.

2. **Damgård's padding rule :** The message space for Damgård's padding rule is $\{0,1\}^*$. He used different padding rule $\mathsf{pad}_{\mathrm{damg}}$ which does not need to pad length of the message but it pads every message block by a single bit 0 or 1 depending on whether it is first block or not and finally, it pads one complete block representing the amount of 0-padding. More precisely, let $M\|0^d = x_1\|\cdots\|x_k$ where $|x_i| = b-1$ and d is the smallest nonnegative integer such that $|M| + d$ is multiple of $b-1$. Now,

$$\mathsf{pad}_{\mathrm{damg}}(M) = 0 \parallel x_1 \parallel 1 \parallel x_2 \parallel \cdots \parallel 1 \parallel x_k \parallel \mathrm{len}_b(d)$$

   One disadvantage of using Damgård's padding is that for large messages it is not as efficient as Merkle's length padding as it needs more number of padded bits. However, unlike $\mathsf{pad}_{\mathrm{merk}}$, it can be applied to any arbitrary messages.

3. **Sarkar's padding rule :** Sarkar defined a new padding rule which can be applied to any tree based iteration which includes the classical sequential iteration. The padding rule is defined as

$$\mathsf{pad}_{\mathrm{sarkar}}(M) = 0 \parallel X_0 \parallel 0^{d_0} \parallel 1 \parallel X_1 \parallel 0^{d_1} \parallel \cdots \parallel 1 \parallel X_k \parallel 0^{d_k}$$

   where $X_0 = M$, $X_i = \chi(X_{i-1})$, $1 \leq i \leq k$ and $\chi(x)$ denotes the smallest binary representation of $|x|$. Let k be the least positive integer such that $|X_k| \leq b$. The d_i's are smallest nonnegative integer so that $|X_i| + d_i$ is a multiple of $b-1$. Note that, it can handle arbitrary messages and needs $\mathrm{O}(\log(|M|) \times \log^*(|M|))$ many padded bits where $\log^*$ is much slower function compare to log (see [16] for more details).

Table 1. A comparison table for different padding rules with an underlying compression function $f : \{0,1\}^{b+t} \to \{0,1\}^t$. pad_0 represents the 10^d padding rule defined in this section and $\mathsf{pad}_{\text{length}}$ represents the suffix padding rule defined in the section 3.2. Padding length is given in terms of order. The last column represents when the hash function is collision secure.

padding rule	message space	padding length	assumption on f
$\mathsf{pad}_{\text{merk}}$ [10]	$\{0,1\}^{\leq 2^s-1}$	b	collision-secure
$\mathsf{pad}_{\text{damg}}$ [5]	$\{0,1\}^*$	$\|M\|$	collision-secure
$\mathsf{pad}_{\text{sarkar}}$ [16]	$\{0,1\}^*$	$\log \|M\| \log^* \|M\|$	collision-secure
$\mathsf{pad}_{\text{length}}$	$\{0,1\}^*$	$\log \|M\|$	collision-secure
pad_0	$\{0,1\}^*$	b	$(t-1)$-bit collision-secure

It is straightforward to see that all these padding rules are suffix-free (we leave it for readers to verify). We provide a simple example of suffix-free padding rule $\mathsf{pad}_{\text{length}}$ which needs $\mathrm{O}(\log(|M|))$ many bits. Basically, we apply Damgård's padding rule to the length of the message instead of applying it to the whole message. The precise definition of the padding rule can be found in section 3.2 and it is parameterized by a parameter s. Note that for any choice of s, the message space of this padding rule is $\{0,1\}^*$. This padding is so far the most efficient padding rule (in terms of the number of padding bits) and if the message size is less than 2^{63} then this padding rule with $s = 64$ is same as Merkle's padding rule. Therefore in practice, we can apply MD hash function as long as message size is less than 2^{63}. In table 1, we make a comparison for all these padding rules with respect to message space $\mathcal{M}$, length of the padded bits for a message M and how it preserves the collision security.

3 Characterization of Collision Preserving Padding Rules

In this section, we characterize all padding rules applied to MD hash function preserving the collision resistant property. We first show that suffix-free padding rule is sufficient to preserve collision resistant and then we provide some simple examples of suffix-free padding rules which are better than the known padding rules in terms of the padding size and the message space, in which padding rule can be applied. Finally, we show that suffix-free property is also necessary to preserve collision resistant property.

3.1 Sufficient condition of collision-preserving padding

For any $f : \{0,1\}^{b+t} \to \{0,1\}^t$ and $h \in \{0,1\}^t$ we have defined the iterated function $f_h^+ : (\{0,1\}^b)^+ \to \{0,1\}^t$. We can extend the definition to the domain $(\{0,1\}^b)^*$ by defining $f_h^+(\lambda) = h$ where λ is the empty bit string. It is easy to see that if $X_1 \in (\{0,1\}^b)^{k_1}$ and $X_2 \in (\{0,1\}^b)^{k_2}$ then

$$f_h^+(X_1 \| X_2) = f_{h'}^+(X_2) \text{ where } h' = f_h^+(X_1).$$

Now we provide basic intuitive lemma whose immediate corollary is that the suffix-free padding rule preserves collision resistant for MD hash function. The lemma says that if we have free-start collision for iterated hash f^+ (i.e., $f_h^+(X) = f_{h'}^+(X')$) with same length then there must be an intermediate collision during computations of $f_h^+(X)$ and $f_{h'}^+(X')$. A computation of $f_h^+(x_1, \cdots, x_k)$ means that the sequence of computations of $h_i = f(h_{i-1}, x_i)$, $1 \leq i \leq k$, where $h_0 = h$.

Lemma 1. (basic lemma)
Let $f:\{0,1\}^{b+t}\rightarrow\{0,1\}^t$ *and* $(h,x_1,\cdots,x_k)\neq(h',x'_1,\cdots,x'_k)$ *where* $h,h'\in\{0,1\}^t$ *and* x_1,x'_1,$\cdots$, $x_k,x'_k\in\{0,1\}^b$*. Then,*

$$f_h^+(x_1,\cdots,x_k)=f_{h'}^+(x'_1,\cdots,x'_k)\Rightarrow f(z,x_i)=f(z',x'_i),\ (z,x_i)\neq(z',x'_i)\ \textit{for some}\ 1\le i\le k$$

where $z=f_h^+(x_1,\cdots,x_{i-1})$ *and* $z'=f_{h'}^+(x'_1,\cdots,x'_{i-1})$.

Proof. Define $h_0=h$, $h'_0=h'$, $h_i=f_h^+(x_1,\cdots,x_i)$ and $h'_i=f_{h'}^+(x'_1,\cdots,x'_i)$, $1\le i\le k$. Now we restate the statement of the lemma as follows.

> Given that $h_k=h'_k$ and $(h_0,x_1,\cdots,x_k)\neq(h'_0,x'_1,\cdots,x'_k)$ there must exist $0\le i<k$ such that $(h_i,x_{i+1})\neq(h'_i,x'_{i+1})$ but $h_{i+1}=h'_{i+1}$.

Thus, we have a collision $f(h_i,x_{i+1})=h_{i+1}=h'_{i+1}=f(h'_i,x'_{i+1})$. To prove that there exists above such i, we use the contradiction method. So assume that, for all i, it is not true. Therefore, for all $1\le i<k$, $h_{i+1}=h'_{i+1}$ implies that $(h_i,x_{i+1})=(h'_i,x'_{i+1})$. Starting from $h_k=h'_k$ we have $(h_{k-1},x_k)=(h'_{k-1},x'_k)$. Since $h'_{k-1}=h_{k-1}$ we also have $(h_{k-2},x_{k-1})=(h'_{k-2},x'_{k-1})$ and so on. Thus we must have $(h_0,x_1,\cdots,x_k)=(h'_0,x'_1,\cdots,x'_k)$ which is not true. Hence the claim is proved by contradiction. □

Recall that, $H^f_{\mathrm{IV},\mathsf{pad}}(M)=f^+_{\mathrm{IV}}(\mathsf{pad}(M))$ for a padding rule $\mathsf{pad}:\mathcal{M}\rightarrow(\{0,1\}^b)^+$. Here we fix an initial value IV and hence we are only interested in a single hash function and we write $H:=\mathsf{MD}^f_{\mathsf{pad}}$. The computation of padding rule must be injective. Otherwise we will be able to find a collision of the hash function easily for any choice of underlying compression function. More precisely, if we have two messages $M\neq M'$ such that $\mathsf{pad}(M)=\mathsf{pad}(M')$ then clearly $H^f_{\mathrm{IV},\mathsf{pad}}(M)=H^f_{\mathrm{IV},\mathsf{pad}}(M')$ for any f. Since we usually choose a simply defined padding rule we can assume that we will be able to find efficiently a collision pair (M,M') of the padding rule if it is not injective. Now we want to classify all padding rules such that MD hash functions based on these padding rule preserves collision security for any choice of initial value. Here we show that this class is nothing but the set of all suffix-free padding rules.

Theorem 1. Sufficient Condition for collision-preserving padding
If pad *is suffix-free padding rule then given any collision pair* (M,M') *for* $\mathsf{MD}^f_{\mathsf{pad}}$ *we can construct a collision pair of* f *efficiently. Thus,* $\mathsf{MD}^f_{\mathsf{pad}}$ *is preserving* (ϵ,ϵ)*-collision security for any* $\epsilon>0$.

Proof. Let $\mathsf{pad}(M)=X$ and $\mathsf{pad}(M')=X'$. Without loss of generality we assume that $|X|\le|X'|$. Let $X\in(\{0,1\}^b)^k$ and $X'=(Z,Y)$ where $Y\in(\{0,1\}^b)^k$. Define $h'=f_h^+(Z)$. As (M,M') is a collision pair, $f_h^+(X)=f_h^+(X')$ and hence $f^h(X)=f_{h'}(Y)$ where both X and Y are members of $(\{0,1\}^b)^k$ and $X\neq Y$ (since pad is suffix-free and hence X is not a suffix of X'). Thus, by using the above basic lemma 1 we must have a collision for f in the computation of $f_h^+(X)$ and $f_{h'}^+(Y)$. Since the computation of $f_h^+(X')$ includes the computation of $f_{h'}^+(Y)$, we are done. The collision advantage for f is at least the collision advantage of MD hash function. Proving the efficiency of the collision finding for f is simple as we need to compute at most $(|M|+|M'|)/b$ computations of f where (M,M') is a collision pair generated from a collision finding algorithm for MD hash function. □

3.2 Simple Examples of suffix-free Padding Rules

We would like to note that known padding rules such as Damgård's padding rule, Merkle's padding rule, Sarkar's padding rule are all suffix-free.

Proposition 1. *The padding rules* $\mathsf{pad}_{\mathrm{merk}}$*,* $\mathsf{pad}_{\mathrm{damg}}$*, and* $\mathsf{pad}_{\mathrm{sarkar}}$ *are suffix-free. Hence the Merkle-Damgård hash function based on these padding rules preserves the collision security.*

It is straightforward to verify and so we leave it for readers to verify. Recall that Damgård and Sarkar's padding rules have domain $\{0,1\}^*$ whereas the Merkle's padding rule has domain $\{0,1\}^{2^d-1}$ for some fixed d. Damgård padding rule pads $\mathrm{O}(|M|)$ bits to the message M whereas Sarkar's padding rule needs $\mathrm{O}(\log^* |M| \log |M|)$ many bits pad where $\log^*(|M|)$ is much slower function compared to $\log(|M|)$. Now we propose a much simpler padding rule which is suffix-free and which takes $\mathrm{O}(\log(|M|))$ many padded bits.

Let $\chi_s(|M|)$ denote the smallest multiple of s-bit binary representation of $|M|$. One can fix a suitable integer s, such as 8,32,64 etc. In other words, we first have a binary represent of $|M|$ and then add a sequence of 0's in the beginning of the length representation so that the size becomes multiple of s. Now we write $\chi_{s-1}(|M|) = p_1 \| \cdots \| p_{\ell-1} \| p_\ell$ where $|p_i| = s-1$ for $1 \le i \le \ell$. Let d be the smallest non-negative integer such that $(d + |M| + \ell s)$ is a multiple of b. Now we define

$$\mathsf{pad}_{\mathrm{length}}(M) = M \parallel 0^d \parallel 0 \parallel p_1 \parallel 1 \parallel p_2 \parallel 1 \parallel \cdots \parallel p_\ell.$$

By definition of d, the size of $\mathsf{pad}_{\mathrm{length}}$ becomes multiple of b. It is easy to see that this padding rule pads $d + \ell s$ many bits which is at most $b + \lceil \log(|M|) \rceil + \lceil \frac{\log(|M|)}{s-1} \rceil$. So the number of padding bits for this padding rule is $\mathrm{O}(\log |M|)$. Moreover, it can also be applied to arbitrary messages. Now we prove that it is a suffix-free padding rule. By using theorem 1, the MD hash function with this padding rule preserves collision security.

Lemma 2. *The padding rule* $\mathsf{pad}_{\mathrm{length}}$ *is suffix-free.*

Proof. Suppose $\mathsf{pad}_{\mathrm{length}}(M') = (X, \mathsf{pad}_{\mathrm{length}}(M))$ for some X, M and M'. Let us denote

$$\mathsf{pad}_{\mathrm{length}}(M) = M0^d 0 p_1 1 p_2 \cdots 1 p_\ell, \quad \mathsf{pad}_{\mathrm{length}}(M) = M' 0^{d'} 0 p'_1 1 p'_2 \cdots 1 p'_{\ell'}.$$

Since $\mathsf{pad}_{\mathrm{length}}(M') = (X, \mathsf{pad}_{\mathrm{length}}(M))$ we must have $\ell' = \ell$ and $p'_i = p_i$, $1 \le i \le \ell$ by comparing the positions of 1 and 0 bits. So, $|M| = |M'|$ and hence $d = d'$. Thus, $|\mathsf{pad}_{\mathrm{length}}(M')| = |\mathsf{pad}_{\mathrm{length}}(M)|$. So, $X = \lambda$ and $\mathsf{pad}_{\mathrm{length}}(M') = \mathsf{pad}_{\mathrm{length}}(M)$ and hence $M = M'$. This proves that $\mathsf{pad}_{\mathrm{length}}$ is suffix-free. □

Remark 1. Note that for any messages of size up to 2^{63} the padding rule $\mathsf{pad}_{\mathrm{length}}$ with $s = 64$ is same as $\mathsf{pad}_{\mathrm{merk}}$. So we can use Merkle's padding rule and for any message longer than 2^{63} one can extend the definition by using $\mathsf{pad}_{\mathrm{length}}$ with $s = 64$. One may argue that the Merkle's padding rule is sufficient enough for all practical messages and the new padding rule only have of theoretical interest. However, in some applications (such as smart card) short messages appear more frequently (message sizes are usually less than 2^{15}). In those application, we can choose small value of s (such as 8 or 16). With this padding rule, we save at least 16 bit padding and as a result the hash function may be faster (since we can save sometimes one compression function invocation which is significant for short messages) than usual Merkle's padding rule. So, there are some applications where this padding rules are practically useful. Note that with $s = 16$, we can pad any message of arbitrary length. If we know beforehand that the message size can be large in some application, then $s = 32$ would be a reasonable choice.

Remark 2. **A variant of $\mathsf{pad}_{\text{length}}$**
Instead of padding length of the message, one can pad the binary representation of the number of message blocks. The padding rule capturing this notion is defined as follows:

$$\mathsf{pad}'_{\text{length}}(M) = M \parallel 10^d \parallel 0 \parallel p_1 \parallel 1 \parallel p_2 \parallel 1 \parallel \cdots \parallel p_\ell$$

where $\chi_{s-1}(\lceil \frac{|M|}{b} \rceil) = p_1 \| \cdots \| p_{\ell-1} \| p_\ell$ and d is the smallest non-negative integer such that $(d + 1 + |M| + \ell s)$ is multiple of b. This variant actually pads the number of blocks of the message M instead of length of the message. In terms of order, both padding rule needs $\mathrm{O}(\log |M|)$ many bits. However, in most cases of message sizes, the later needs less number of padding bits.

3.3 A Necessary Condition for Collision-preserving Padding Rule

We have shown that any suffix-free property is good for collision-preserving. Now we show that it is also a necessary condition. To show this let us fix a padding rule pad which is not suffix-free. Therefore, we also fix $M \neq M'$ such that $\mathsf{pad}(M)$ is a suffix of $\mathsf{pad}(M')$. We can do so since we assume that padding rule is simply defined function and hence it must be easy to find such a pair. Now we first construct a compression function f given a collision secure compression function f' (if there is no such then the question is moot) such that (M, M') is a collision pair of H^f_{pad}. Then we prove that finding collision of f given M and M' and the oracle f is as hard as finding collision of f'. This proves that suffix-free padding rule is necessary to have a collision-preserving MD hash function for any compression function and for any initial value.

Theorem 2. Suffix-free is necessary condition
Let pad *be a fixed padding rule which is not suffix-free. Now, given a collision resistant compression function f' there is a collision resistant compression function f such that the Merkle-Damgård hash function based on f with the padding rule* pad *is not collision resistant (by providing a collision pair of it).*

Proof. Let us assume that there is a $(t-1)$-bit collision resistant compression function $f' : \{0,1\}^{b+t} \to \{0,1\}^{t-1}$, otherwise the question is moot. Without loss of generality, let the last bit of IV is 1. We have fixed a pair (M, M') such that $\mathsf{pad}(M)$ is a suffix of $\mathsf{pad}(M')$. Let $\mathsf{pad}(M') = (X, m) \| \mathsf{pad}(M)$, $m \in \{0,1\}^b$ and $X \in (\{0,1\}^b)^*$. For simplicity we first assume that $X = \lambda$. Define

$$f(x) = \begin{cases} f'(x) \parallel 0 & \text{if } x \neq \mathrm{IV} \| m \\ \mathrm{IV} & \text{if } x = \mathrm{IV} \| m \end{cases}$$

Thus, $f(\mathrm{IV}, m) = \mathrm{IV}$, a fixed point for f. Now it is easy to see that $H^f_{\mathsf{pad}}(M) = H^f_{\mathsf{pad}}(M')$. Now we have to show that f is collision resistant. Suppose there exists a collision finding algorithm A which finds collision for f. Let $x \neq x'$ such that $f(x) = f(x')$ where A returns the collision pair (x, x'). Now, it is easy to check that (x, x') is a collision pair of f' too. Since we do not know any collision algorithm for f', f must be collision resistant.

Now consider the case where $X \in (\{0,1\}^b)^+$. We first define $\widetilde{f}(x) = f'(x) \| 0$ and compute $\mathrm{IV}' = \widetilde{f}^+_{\mathrm{IV}}(X)$. Note that the last bit of IV' is 0 and hence it is different from IV. Since f' is a collision resistant compression function, IV' should be different from all other intermediate values in the computation of $\widetilde{f}^+_{\mathrm{IV}}(X)$. Otherwise, we will be able to find collision of f' easily. We define

$$f(x) = \begin{cases} f'(x) \parallel 0 & \text{if } x \neq \mathrm{IV}' \| m \\ \mathrm{IV} & \text{if } x = \mathrm{IV}' \| m \end{cases}$$

So, $f(x)$ and $\widetilde{f}(x)$ are same whenever $x \neq \mathrm{IV}' \| m$. Since IV' is different from all other intermediate values in the computation of $\widetilde{f}^{+}_{\mathrm{IV}}(X)$, we must have $f^{+}_{\mathrm{IV}}(X) = \widetilde{f}^{+}_{\mathrm{IV}}(X) = \mathrm{IV}$. Now we can show that (M, M') is a collision pair of H^f_{pad}. Note that, $H^f_{\mathsf{pad}}(M') = f^{+}_{\mathrm{IV}}(X, m, \mathsf{pad}(M)) = f^{+}_{\mathrm{IV}'}(m, \mathsf{pad}(M)) = f^{+}_{\mathrm{IV}}(\mathsf{pad}(M)) = H^f_{\mathsf{pad}}(M)$. Hence $H^f_{\mathsf{pad}}(M) = H^f_{\mathsf{pad}}(M')$. Now we show that the compression function f is also collision resistant. Suppose not, $x \neq x'$ and $f(x) = f(x')$ then clearly $f'(x) = f'(x')$ if $x, x' \neq \mathrm{IV}' \| m$. Moreover x or x' can not be $\mathrm{IV}' \| m$ otherwise the last bits of $f(x)$ and $f(x')$ will be different. Hence f is collision resistant as long as f' is collision resistant. The collision pair (M, M') also does not help to find a collision since M and M' are efficiently computable and we can use M and M' for any collision finding algorithm for f'. □

3.4 Is $\mathsf{pad}_{\text{length}}$ optimum?

We have already shown that MD hash with a padding rule pad must be suffix-free to preserve the collision security property. We also have provided an example of suffix-free padding rule $\mathsf{pad}_{\text{length}}$ which requires $O(\log |M|)$ bits. A natural question is to ask whether it is the optimum in terms of the padding bits. Let pad be any suffix-free padding rule and N_i denotes the number of all messages which has i bits after applying the padding rule pad. In other words, $N_i = |\mathsf{pad}^{-1}(\{0,1\}^i)|$. By using the property that pad is suffix-free one can find the following relation:

$$\sum_{j=1}^{k} N_j 2^{k-j} \leq 2^k.$$

If pad requires asymptotically $d(n)$ bits for n-bit messages then $\sum_{j=1}^{n+d(n)} N_j \geq 2^n$. Using these two relations one may be able to show that $d(n) \geq \log(n)$. However, we have no strong evident to claim that. We postulate this as a conjecture.

Conjecture: For any suffix-free padding rule $\mathsf{pad} : \{0,1\}^* \to (\{0,1\}^b)^+$, $|\mathsf{pad}(M)| \geq |M| + \log |M|$ for sufficiently large $|M|$.

4 Length padding is redundant for a variant of MD

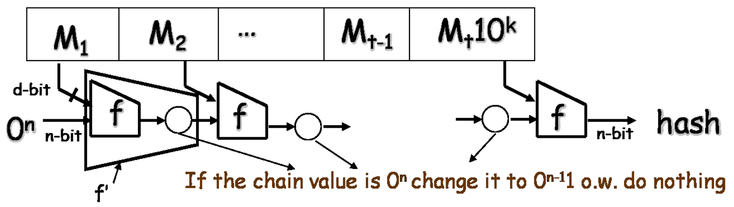

Fig. 2. A variant of MD without length padding.

In this section, we prove that the length padding is unnecessary if we use a small variant of Merkle-Damgård hash function and the underlying compression function is little more secure than collision resistant. We define the variant of the Merkle-Damgård hash function as follows.

Algorithm 1 A Variant of Merkle-Damgård Hash Function

```
Require: $f : \{0,1\}^t \times \{0,1\}^b \to \{0,1\}^t$, $M \in \{0,1\}^*$.
 1: $d$ is the remainder when we divide $t - |M| - 1$ by $t$.
 2: partition $M10^d = M_1 \| \cdots \| M_\ell$, $M_1, \cdots, M_\ell \in \{0,1\}^b$
 3: $h_0 = 0^t$
 4: for $i = 1$ to $\ell$ do
 5:     $h_i = f(h_{i-1}, M_i)$
 6:     if $h_i = 0^t$ then
 7:         $h_i = 0^{t-1}1$
 8:     end if
 9: end for
10: return $h_\ell$.
```

We simply apply MD hash function with pad_0 padding and with a checking in internal chaining value. If we ever come up with 0^t as a chaining value then we simply change the chaining value into $0^{t-1}1$. So we can think that it is MD hash function based on a compression function f' defined below.

$$f'(x) = \begin{cases} f(x) & \text{if } f(x) \neq 0^t \\ 0^{t-1}1 & \text{if } f(x) = 0^t \end{cases}$$

Note that $\mathbf{0} = 0^t$ is also the initial value of the Merkle-Damgård hash function. Suppose $\mathsf{MD}_{\mathbf{0}}^{f'}(M) = \mathsf{MD}_{\mathbf{0}}^{f'}(M)$ with $M \neq M'$. Then by using simple backward induction on the padded messages, we can prove that either there is a collision of f' or there is a message $x \in \{0,1\}^{b+t}$ such that $f'(x) = \mathbf{0}$. By definition of f', there does not exist any such x. If (x, x') is a collision pair of f' then either it is also a collision for f or $f(x) = 0^t$ and $f(x') = 0^{t-1}1$. So either we have a collision of f or we have collision on the first $(t-1)$ bits of f with the collision value 0^{t-1}. Clearly, satisfying the second condition seems much harder than finding collision for any reasonable practical construction of a compression function. Thus, we have proved the following theorem.

Theorem 3. *Suppose it is hard to find collision of f or it is hard to find X_0 and X_1 such that $f(X_0) = 0^t$ and $f(X_1) = 0^{t-1}1$ then the hash function based on f defined in Algorithm 5 is collision resistant.*

Corollary 1. *Suppose it is hard to find collision on the first $(t-1)$-bits of a compression function f then the hash function based on f defined in Algorithm 5 is collision resistant.*

Remark 3. In the last section, we have proved that the suffix-free padding is necessary to preserve collision security for MD hash function. To do so, we provide a counter example of the underlying compression function (easy to get the initial value 0^t even though it is collision secure). On the contrary, in this section, we have shown that any injective padding rule, not necessarily suffix-free, is sufficient to have collision security. Actually, we make sure that the counterexample illustrated in the section 3.3, does not appear by imposing the if condition (step-6 of the Algorithm 5). This is the main reason, we do not contradict. Moreover, the variant actually do not preserve collision security according to the definition 1. But, it says that if we have little more security of the underlying compression function than collision security, then the hash function is collision secure. The additional security assumption seems to hold for any known practical secure hash function.

5 Three-property preserving Hash Function

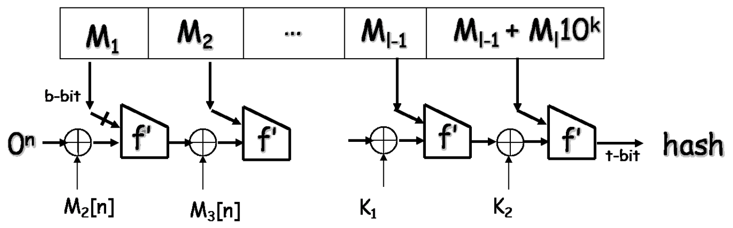

Fig. 3. A variant of BCM preserving 2nd preimage.

Now we consider salted hash function which outputs the hash value with a salt $K \in \{0,1\}^s$ which is chosen randomly and keep it public (unlike message authentication code, hash function should be publicly computable). We denote the salted hash function as $H_K(\cdot)$ where K is the salt. We define three basic securities of a hash function for a salted hash function. These are collision, preimage and second-preimage and its corresponding advantages of adversaries is defined as

$$\mathbf{Adv}_H^{\mathrm{coll}}(A_1) = \mathsf{Pr}[A_1(K) \to (M, M') : H_K(M) = H_K(M'), M \neq M']$$
$$\mathbf{Adv}_H^{\mathrm{2PI}}(A_3) = \mathsf{Pr}[A_3(K, M) \to M' : H_K(M) = H_K(M'), M \neq M']$$
$$\mathbf{Adv}_H^{\mathrm{PI}}(A_2) = \mathsf{Pr}[A(K, z) \to M : H_K(M) = z]$$

where probabilities are computed over the internal randomness of the adversaries, uniform distribution of K, z and M chosen from $\{0,1\}^s$, $\{0,1\}^t$, and $\{0,1\}^\ell$ for some ℓ, respectively. Now we define a variant of $\mathcal{BCM}$ [1] (backward chaining mode) which is simpler and needs less salt size. Recall that $\mathcal{BCM}$ uses the padding rule $\mathsf{pad}_{\mathrm{merk}}$ and its salt size is $b + 2t$. In this paper, we use salt of size $2t$ only. Here we would like to make a note that in the previous version [12] we have proposed a design with t-bit salt which unfortunately does not preserve the 2nd preimage for all messages. So the proposed claim (Theorem 4 in [12]) is actually wrong. However, we have corrected the design at the cost of extra t-bit salt. We denote the hash function as $\mathsf{BCM}^f_{\mathsf{pad}_0}$ since it uses the simplest injective padding rule pad_0. One may use some other padding rules.

A hash family $H := \{H_K\}_{K \in \{0,1\}^s}$ based on a compression function f is said to almost preserve (ϵ, ϵ')-collision security if given an efficient algorithm A with at least ϵ collision advantage for H_K, we can construct (write down its code modulo the subroutine A) an efficient algorithm A' with at least ϵ' collision advantage for f' i.e. either we have collision of f or we have two preimages of 0^t and $0^{t-1}1$ both. We say that H preserve (ϵ, ϵ')-second preimage security if given an efficient algorithm A with at least ϵ second preimage advantage for H_K, we can construct (write down its code modulo the subroutine A) an efficient algorithm A' with at least ϵ' second preimage advantage for f.

Algorithm 2 A Variant of Merkle-Damgård Hash Function $\mathsf{BCM}^f_{\mathsf{pad}_0}$

```
Require: $f : \{0,1\}^t \times \{0,1\}^b \to \{0,1\}^t$ and $M \in \{0,1\}^*$.
 1: $d$ is the remainder when we divide $t - |M| - 1$ by $t$.
 2: partition $M10^d = M_1\| \cdots \|M_\ell$, $M_1, \cdots, M_\ell \in \{0,1\}^b$
 3: $h_0 = 0^t$
 4: for $i = 1$ to $\ell$ do
 5:     if $i = \ell - 1$ then
 6:         $h_\ell = f(h_{\ell-1} \oplus K_1, M_{\ell-1})$
 7:     else if $i = \ell$ then
 8:         $h_\ell = f(h_{\ell-1} \oplus K_2, M_\ell + M_\ell)$
 9:     else
10:         $h_i = f(h_{i-1} \oplus M_{i+1}[t], M_i)$ \\ $X[t]$ represents the first $t$-bits of $X$.
11:     end if
12:     if $h_i = 0^t$ then
13:         $h_i = 0^{t-1}1$
14:     end if
15: end for
16: return $h_\ell$.
```

For MD construction or BCM constructions, the preimage-preserving property is easy to show and we skip the details. Note that in our definition of preimage we choose the target message randomly and then we want to find a message whose hash value matches the target. Inverting $\mathsf{BCM}_{\mathsf{pad}_0}$ or Merkle-Damgård reduces to inverting the compression function by looking at the last invocation of the compression function.

Theorem 4. *$\mathsf{BCM}_{\mathsf{pad}_0}$ almost preserves (ϵ, ϵ)-collision security. Moreover, if $\mathsf{Pr}[f(X) \in \{0^t, 0^{t-1}1\} : X \xleftarrow{*} \{0,1\}^{b+t}] \le \nu$ (ideally it should be close to $\frac{1}{2^{t-1}}$) then $\mathsf{BCM}_{\mathsf{pad}_0}$ preserves $(\epsilon, \frac{(\epsilon-\nu)}{\ell+1})$-second preimage security for any messages with at least ℓ complete blocks.*

Proof. We first note that for any $M \neq M'$, $\mathsf{BCM}_{\mathsf{pad}_0}(M) = \mathsf{BCM}_{\mathsf{pad}_0}(M')$ then there must be a collision in f'. This is due to the same reason we have for the modified Merkle-Damgård construction. Note that f' does not output 0^t and 0^t is the initial value. Thus, it almost preserves (ϵ, ϵ)-collision security. To prove the second preimage-preserving property, let A be a second preimage attacker of $\mathsf{BCM}_{\mathsf{pad}_0}$ for any randomly chosen message of size $s \ge b$. For the time being let us assume $s = \ell b$. Now we write a second preimage attacker A' for f. Given a random $X := (h, m) \in \{0,1\}^{b+t}$, A' works as follows: Let $h_j = f^+(M_1, \ldots, M_j)$, $\forall j$.

```
1: Choose uniformly $i$ from $\{1, 2, \ldots, \ell+1\}$, $M = (M_1, \ldots, M_\ell)$ from $\{0,1\}^s$ and $(K_1, K_2)$ from
   $\{0,1\}^{2t}$.
2: if $h_j = 0^t$ or $0^{t-1}1$ for some $j$ then
3:     Abort
4: end if
5: if $i \le \ell - 1$ then
6:     write $M_{i+1} = f'^+_{IV}(M_1, \ldots, M_{i-1}) \oplus h$, $M_i = m$
7: else if $i = \ell$ then
8:     write $K_1 = f'^+_{IV}(M_1, \ldots, M_{i-1}) \oplus h$, $M_i = m$
9: else
```

```
10:     write $K_2 = f'^{+}_{IV}(M_1, \ldots, M_{i-1}) \oplus h$, $M_\ell = m \oplus 0^{t-1}1$
11: end if
12: run $A(M_1 \ldots, M_\ell, K_1, K_2)$ and obtain a message $M'$.
13: if $\mathsf{BCM}_{\mathsf{pad}_0}(M) \neq \mathsf{BCM}_{\mathsf{pad}_0}(M')$ then
14:     Abort
15: else
16:     If $f(h, m) = f(h', m')$ for some $(h', m') \neq (h, m)$ obtained from the computation $\mathsf{BCM}_{\mathsf{pad}_0}(M')$
17:     return $(h', m')$.
18: end if
```

In line 16 we are always to able to find a collision pair of f since during the computation of $\mathsf{BCM}_{\mathsf{pad}_0}(M)$ no chaining value is 0^t or $0^{t-1}1$ and hence we should have a collision of f (as mentioned for collision-preserving property). The collision pair eventually contain (h, m) has probability $\frac{1}{\ell+1}$ as i is chosen randomly. It is also easy to see that the inputs for A is uniformly distributed. □

6 Conclusion

We study padding rules in different aspects which is very essential for designing hash function. Appending 1 followed by a suitable size sequence of 0 and binary representation of length is a very standard way to have padding for hash function. But, this padding rule can only be applied for messages of size at most some specified large value. Both for theoretical and practical point of views, we can look for padding rules which can handle arbitrary messages. Padding rules by Damgård and Sarkar are some known examples for this. Here we have shown that suffix-free padding rule is sufficient to preserve collision resistant and as a result we construct a simple padding rule which is more efficient than padding rules both by Damgård and Sarkar. We have also proved that suffix is a necessary condition preserving collision security. Thus, it would be interesting to see that whether padding rule is optimum or not. So we propose the following open problem.

Open Problem : Try to find suffix-free padding rule which needs less than logarithm number of bits. On the other hand, we can try to prove that asymptotically any suffix-free padding rule needs at least logarithm number of bits.

We believe that the second option is more feasible. We also have shown that the simplest padding such as padding 10^d only can be sufficient for collision preserving property if we restrict collision resistant assumption of the underlying compression function for the first $(t-1)$ bits. Thus, the simplest Merkle-Damgård hash function becomes collision resistant which does not have any overhead costs due to length of the message. We also study a simple variant of MD hash function which preserves collision resistance, second preimage as well as preimage resistance. Thus we believe that padding length is not needed if we choose initial value properly.

References

1. E. Andreeva and B. Preneel *A Three-Property-Preserving Hash Function.* To appear in Selected Areas in Cryptography, 2008.
2. M. Bellare and P. Rogaway. *Collision-Resistant Hashing: Towards Making UOWHFs Practical.* Advances in Cryptology - Crypto'97, Lecture Notes in Computer Science, vol 1294, Springer-Verlag, 1997, pp. 470-484.

3. M. Bellare and P. Rogaway. *Introduction to Modern Cryptography.* Available at http://www-cse.ucsd.edu/ mihir/cse207/classnotes.html
4. R. Cramer and V. Shoup. *Using Hash Functions as a Hedge against Chosen Ciphertext Attack.* Advances in Cryptology - Eurocrypt'00, Lecture Notes in Computer Science, vol 2442, Springer-Verlag, pp. 275-288, 2000.
5. I. B. Damgård. *A Design Principle for Hash Functions.* Advances in Cryptology - Crypto'89, Lecture Notes in Computer Sciences, vol 435, Springer-Verlag, pp. 416-427, 1989.
6. I. B. Damgård. *Collision Free Hash Functions and Public Key Signature Schemes.* Advances in Cryptology - Eurocrypt'87, Lecture Notes in Computer Sciences, Vol. 304, Springer-Verlag, pp. 203-216, 1987.
7. J. K. Gibson. *Discrete logarithm hash function that is collision free and one-way.* IEE Proceedings-E 138, pp. 407-410, 1991.
8. Don. B Johnson. *Improving Hash Function Padding.* NIST hash workshop 2005. Available at http://csrc.nist.gov/groups/ST/hash/documents/Johnson_Padding.pdf
9. J. Kelsey and B. Schneier. *Second Preimages on n-bit Hash Functions for Much Less than 2^n Work.* Advances in Cryptology - Eurocrypt 2004, Lecture Notes in Computer Scince, Springer-Verlag, vol 3494, pp 474-490. .
10. R. Merkle. *One Way Hash Functions and DES.* Advances in Cryptology - Crypto'89, Lecture Notes in Computer Sciences, Vol. 435, Springer-Verlag, pp. 428-446, 1989.
11. M. Naor and M. Yung. *Universal one-way hash functions and their cryptographic applications*, Proceedings of the Twenty First Annual ACM Symposium on Theory of Computing, ACM Press, pp 33-43, 1989.
12. M. Nandi *Characterizing Padding Rules of MD Hash Functions Preserving Collision Security*, Proceeding of Information Security and Privacy, Lecture Notes in Computer Science, Springer Volume 5594/2009, pp 171-184, 2009.
13. NIST/NSA. *FIPS 180-2 Secure Hash Standard.* August, 2002. Online available at http://csrc.nist.gov/publications/fips/fips180-2/fips180-2.pdf.
14. P. Rogaway, *Formalizing Human Ignorance: Collision-Resistant Hashing without the Keys.* Progress in Cryptology - VIETCRYPT 2006, Vol 4341/2006, pp 211-228, 2007
15. P. Rogaway, T. Shrimpton. *Cryptographic Hash Function Basics: Definitions, Implications and separations for Pre-image resistance, Second Pre-image Resistance and Collision Resistance*, Fast Software Encryption'04, Lecture Notes in Computer Scince, Springer-Verlag, vol 3017. 2004.
16. P. Sarkar. *Domain Extender for Collision Resistant Hash Functions: Improving Upon Merkle-Damgard Iteration.* Discrete Applied Mathematics Volume 157, Issue 5, 6 March 2009, pp 1086-1097.
17. D. R. Stinson. *Cryptography : Theory and Practice*, Second Edition, CRC Press, Inc.
18. D. R. Stinson. *Some observations on the theory of cryptographic hash functions.* Designs, Codes and Cryptography, Vol 38, 2006, pp 259-277.