Enhanced Target Collision Resistant Hash Functions Revisited*

Mohammad Reza Reyhanitabar, Willy Susilo, and Yi Mu

Centre for Computer and Information Security Research,
School of Computer Science and Software Engineering
University of Wollongong, Australia
{rezar, wsusilo, ymu}@uow.edu.au

Abstract. Enhanced Target Collision Resistance (eTCR) property for a hash function was put forth by Halevi and Krawczyk in Crypto 2006, in conjunction with the randomized hashing mode that is used to realize such a hash function family. eTCR is a strengthened variant of the well-known TCR (or UOWHF) property for a hash function family (*i.e.* a dedicated-key hash function). The contributions of this paper are twofold. First, we compare the new eTCR property with the well-known collision resistance (CR) property, where both properties are considered for a dedicated-key hash function. We show there is a *separation* between the two notions, that is *in general*, eTCR property cannot be claimed to be weaker (or stronger) than CR property for any arbitrary dedicated-key hash function. Second, we consider the problem of eTCR property preserving domain extension. We study several domain extension methods for this purpose, including (Plain, Strengthened, and Prefix-free) Merkle-Damgård, Randomized Hashing (considered in dedicated-key hash setting), Shoup, Enveloped Shoup, XOR Linear Hash (XLH), and Linear Hash (LH) methods. Interestingly, we show that the only eTCR preserving method is a *nested variant* of LH which has a drawback of having high key expansion factor. Therefore, it is interesting to design a new and *efficient* eTCR preserving domain extension in the *standard model*.

Keywords: Hash Functions, CR, TCR, eTCR, Domain Extension.

1 Introduction

Cryptographic hash functions are widely used in many cryptographic schemes, most importantly as building blocks for digital signature schemes and message authentication codes (MACs). Their application in signature schemes following hash-and-sign paradigm, like DSA, requires the collision resistance (CR) property. Contini and Yin [5] showed that breaking the CR property of a hash function can also endanger security of the MAC schemes, which are based on the hash function, such as HMAC. Despite being a very essential and widely-desirable security property of a hash function, CR has been shown to be a very

* The full version of this paper is available from [18].

O. Dunkelman (Ed.): FSE 2009, LNCS 5665, pp. 327–344, 2009.

strong and demanding property for hash functions from theoretical viewpoint [22, 4, 17] as well as being a practically endangered property by the recent advances in cryptanalysis of widely-used standard hash functions like MD5 and SHA-1 [25, 24]. In response to these observations in regard to the strong CR property for hash functions and its implication on the security of many applications, recently several ways out of this uneasy situation have been proposed.

The first approach is to avoid relying on the CR property in the design of new applications and instead, just base the security on other weaker than CR properties like Target Collision Resistance ("Ask less of a hash function and it is less likely to disappoint! " [4]). This is an attractive and wise methodology in the design of new applications using hash functions, but unfortunately it might be of limited use to secure an already implemented and in-use application, if the required modifications are significant and hence prohibitive (and not cost effective) in practice.

The second approach is to design new hash functions to replace current endangered hash function standards like SHA-1. For achieving this goal, NIST has started a public competition for selecting a new secure hash standard SHA-3 to replace the current SHA-1 standard [15]. It is hoped that new hash standard will be able to resist against all known cryptanalysis methods, especially powerful statistical methods like differential cryptanalysis which have been successfully used to attack MD5, SHA-1 and other hash functions [25, 24, 23].

Another methodology has also recently been considered as an intermediate step between the aforementioned two approaches in [10, 9]. This approach aims at providing a "safety net" by fixing the current complete reliance on endangered CR property without having to change the internals of an already implemented hash function like SHA-1 and instead, just by using the hash function in some black-box modes of operation. Based on this idea, Randomized Hashing mode was proposed in [10] and announced by NIST as Draft SP 800-106 [16]. In a nutshell, Randomized Hashing construction converts a keyless hash function H (e.g. SHA-1) to a dedicated-key hash function $\tilde{H}$ defined as $\tilde{H}_K(M) = H(K||(M_1 \oplus K)|| \cdots ||(M_L \oplus K))$, where H is an iterated Merkle-Damgård hash function based on a compression function h. ($M_1|| \cdots ||M_L$ is the padded message after applying strengthening padding.) Note that Randomized Hashing keys the entire iterated hash function H at once, by using it as a black-box function and just preprocessing the input message M with the key K (*i.e.* the *random salt*).

Although the main motivation for the design of a randomized hashing mode in [10] was to free reliance on collision resistance assumption on the underlying hash function (by making off-line attacks ineffective by using a random key), in parallel to this aim, a new security property was also introduced and defined for hash functions, namely *enhanced* Target Collision Resistance (eTCR) property. Having $\tilde{H}$ as the first example of a construction for eTCR hash functions in hand, we also note that an eTCR hash function is an interesting and useful new primitive. In [10], the security of the specific example function $\tilde{H}$ in eTCR sense is based on some new assumptions (called c-SPR and e-SPR) about keyless

compression function h. However, this example function $\tilde{H}$, may be threatened as a result of future cryptanalysis results, but the notion of eTCR hashing will still remain useful independently from this specific function. By using an eTCR hash function family $\{H_K\}$ in a hash-and-sign digital signature scheme, one does not need to sign the key K used for the hashing. It is only required to sign $H_K(M)$ and the key K is sent in public to the verifier as part of the signed message [10]. This is an improvement compared to using a TCR (UOWHF) hash function family where one needs to sign $H_K(M)||K$ [4].

Our Contributions

Our aim in this paper is to investigate the eTCR hashing as a new and interesting notion. Following the previous background on the CR notion, the first natural question that arises is whether eTCR is weaker than CR in general. It is known that both CR and eTCR imply TCR property (*i.e.* are stronger notions than TCR) [14, 20, 10], but the relation between CR and eTCR has not been considered yet. As our first contribution in this paper, we compare the eTCR property with the CR property, where both properties are considered formally for a dedicated-key hash function. We show that there is a separation between eTCR and CR notions, that is *in general*, eTCR property cannot be claimed to be weaker (or stronger) than CR property for any arbitrary dedicated-key hash function. Although our separation result does not rule out the possibility of designing specific dedicated-key hash functions in which eTCR might be easier to achieve compared to CR, it emphasizes the point that any such a construction should explicitly show that this is indeed the case.

As our second contribution, we consider the problem of eTCR preserving domain extension. Assuming that one has been able to design a dedicated-key compression function which possesses eTCR property, the next step will be how to extend its domain to obtain a full-fledged hash function which also provably possesses eTCR property and is capable of hashing any variable length message. In the case of CR property the seminal works of Merkle [12] and Damgård [7] show that Merkle-Damgård (MD) iteration with strengthening (length indicating) padding is a CR preserving domain extender. Analysis and design of (multi-)property preserving domain extenders for hash function has been recently attracted new attention in several works considering several different security properties, such as [4, 3, 2, 1]. We investigate eight domain extension transforms for this purpose; namely Plain MD [12, 7], Strengthened MD [12, 7], Prefix-free MD [6, 11], Randomized Hashing [10] (considered in dedicated-key hash setting), Shoup [21], Enveloped Shoup [2], XOR Linear Hash (XLH) [4], and a variant of Linear Hash (LH) [4] methods. Interestingly, we show that the only eTCR preserving method among these methods is a *nested variant* of LH (defined based on a variant proposed in [4]) which has the drawback of having high key expansion factor. The overview of constructions and the properties they preserve are shown in Table 1. The symbol "✓" means that the notion is provably preserved by the construction; "×" means that it is not preserved. Underlined entries related to eTCR property are the results shown in this paper.

Table 1. Overview of constructions and the properties they preserve

Scheme	CR	TCR	eTCR
Plain MD	× [12, 7]	× [4]	×
Strengthened MD	✓[12, 7]	× [4]	×
Prefix-free MD	× [2]	× [2]	×
Randomized Hashing	✓[1]	× [1]	×
Shoup	✓[21]	✓[21]	×
Enveloped Shoup	✓[2]	✓[2]	×
XOR Linear Hash (XLH)	✓[1]	✓[4]	×
Nested Linear Hash (LH)	✓[4]	✓[4]	✓

2 Preliminaries

2.1 Notations

If A is a probabilistic algorithm then by $y \xleftarrow{\$} A(x_1, \cdots, x_n)$ it is meant that y is a random variable which is defined from the experiment of running A with inputs $x_1, \cdots, x_n$ and assigning the output to y. To show that an algorithm A is run without any input (*i.e.* when the input is an empty string) we use the notation $y \xleftarrow{\$} A()$. By time complexity of an algorithm we mean the running time, relative to some fixed model of computation (e.g. RAM) plus the size of the description of the algorithm using some fixed encoding method. If X is a finite set, by $x \xleftarrow{\$} X$ it is meant that x is chosen from X uniformly at random. Let $x||y$ denote the string obtained from concatenating string y to string x. Let 1^m and 0^m, respectively, denote a string of m consecutive 1 and 0 bits. For a binary string M, let $M_{1\ldots n}$ denote the first n bits of M, $|M|$ denote its length in bits and $|M|_b \triangleq \lceil |M|/b \rceil$ denote its length in b-bit blocks. For a positive integer m, let $\langle m \rangle_b$ denotes binary representation of m by a string of length exactly b bits. If S is a finite set we denote size of S by $|S|$. The set of all binary strings of length n bits (for some positive integer n) is denoted as $\{0,1\}^n$, the set of all binary strings whose lengths are variable but upper-bounded by N is denoted by $\{0,1\}^{\leq N}$ and the set of all binary strings of arbitrary length is denoted by $\{0,1\}^*$.

2.2 Two Settings for Hash Functions

In a formal study of cryptographic hash functions and their security notions, two different but related settings can be considered. The first setting is the traditional keyless hash function setting where a hash function refers to a single function H (e.g. H=SHA-1) that maps variable length messages to fixed length output hash value. In the second setting, by a hash function it is meant a family of hash functions $H : \mathcal{K} \times \mathcal{M} \rightarrow \{0,1\}^n$, also called a dedicated-key hash function [2], which is indexed by a key space $\mathcal{K}$. A key $K \in \mathcal{K}$ acts as an index to select a specific member function from the family and often the key argument is denoted as a subscript, that is $H_K(M) = H(K, M)$, for all $M \in \mathcal{M}$. In a

formal treatment of hash functions and the study of relationships between different security properties, one should clarify the target setting, namely whether keyless or dedicated-key setting is considered. This is worth emphasizing as some security properties like TCR and eTCR are inherently defined and make sense for a dedicated-key hash function [20, 10]. Regarding CR property there is a well-known foundational dilemma, namely CR can only be *formally* defined for a dedicated-key hash function, but it has also been used widely as a security *assumption* in the case of keyless hash functions like SHA-1. We will briefly review this formalization issue for CR in Subsection 2.3 and for a detailed discussion we refer to [19].

2.3 Definition of Security Notions: CR, TCR and eTCR

In this section, we recall three security notions directly relevant to our discussions in the rest of the paper; namely, CR, TCR, and eTCR, where these properties are formally defined for a dedicated-key hash function. We also recall the well-known definitional dilemma regarding CR assumption for a keyless hash function.

A dedicated-key hash function $H : \mathcal{K} \times \mathcal{M} \to \{0,1\}^n$ is called (t, ϵ)-x secure, where $\mathsf{x} \in \{\text{CR, TCR, eTCR}\}$ if the advantage of *any* adversary, having time complexity at most t, is less than ϵ, where the advantage of an adversary A, denoted by $\mathrm{Adv}^{\mathsf{x}}_H(A)$, is defined as the probability that a specific winning condition is satisfied by A upon finishing the game (experiment) defining the property x. The probability is taken over all randomness used in the defining game as well as that of the adversary itself. The advantage functions for an adversary A against the CR, TCR and eTCR properties of the hash function H are defined as follows, where in the case of TCR and eTCR, adversary is denoted by a two-stage algorithm $A = (A_1, A_2)$:

$$\mathrm{Adv}^{CR}_H(A) = \Pr\left\{K \xleftarrow{\$} \mathcal{K}; (M, M') \xleftarrow{\$} A(K) : M \neq M' \ \wedge \ H_K(M) = H_K(M')\right\}$$

$$\mathrm{Adv}^{TCR}_H(A) = \Pr\left\{\begin{array}{l}(M, State) \xleftarrow{\$} A_1(); \\ K \xleftarrow{\$} \mathcal{K}; \\ M' \xleftarrow{\$} A_2(K, State);\end{array} : \quad M \neq M' \ \wedge \ H_K(M) = H_K(M')\right\}$$

$$\mathrm{Adv}^{eTCR}_H(A) = \Pr\left\{\begin{array}{l}(M, State) \xleftarrow{\$} A_1(); \\ K \xleftarrow{\$} \mathcal{K}; \\ (K', M') \xleftarrow{\$} A_2(K, State);\end{array} : \begin{array}{c}(K, M) \neq (K', M') \\ \wedge \\ H_K(M) = H_{K'}(M')\end{array}\right\}$$

CR for a Keyless Hash Function. Collision resistance as a security property cannot be formally defined for a keyless hash function $H : \mathcal{M} \to \{0,1\}^n$. *Informally*, one would say that it is "*infeasible*" to find two distinct messages M and M' such that $H(M) = H(M')$. But it is easy to see that if $|\mathcal{M}| > 2^n$ (*i.e.* if the function is compressing) then there are many colliding pairs and hence, trivially there *exists* an *efficient* program that can always output a colliding pair M and M', namely a simple one with M and M' included in its code. That is, *infeasibility* cannot be formalized by an statement like "there exists no efficient adversary with non-negligible advantage" as clearly there are many such

adversaries as mentioned before. The point is that *no human being knows* such a program [19], but the latter concept cannot be formalized mathematically. Therefore, in the context of keyless hash functions, CR can only be treated as a strong *assumption* to be used in a constructive security reduction following human-ignorance framework of [19]. We will call such a CR assumption about a keyless hash function as **keyless-CR assumption** to distinguish it from formally definable CR notion for a dedicated-key hash function. We note that as a result of recent collision finding attacks, it is shown that keyless-CR assumption is completely invalid for MD5 [25] and theoretically endangered assumption for SHA-1 [24].

3 eTCR Property *vs.* CR Property

In this Section, we show that there is a separation between CR and eTCR, that is none of these two properties can be claimed to be weaker or stronger than the other *in general* in dedicated-key hash function setting. We emphasize that we consider relation between CR and eTCR as formally defined properties for a dedicated-key hash function following the methodology of [20]. The CR property considered in this section should not be mixed with the strong keyless-CR assumption for a keyless hash function.

3.1 CR $\nRightarrow$ eTCR

We want to show that the CR property does not imply the eTCR property. That is, eTCR as a security notion for a dedicated-key hash function is not weaker than the CR property. This is done by showing as a counterexample, a dedicated-key hash function which is secure in CR sense but completely insecure in eTCR sense.

Lemma 1 (CR does not imply eTCR). *Assume that there exists a dedicated-key hash function* $H : \{0,1\}^k \times \{0,1\}^m \rightarrow \{0,1\}^n$ *which is* $(t,\epsilon)-CR$. *Select (and* fix*) an arbitrary message* $M^* \in \{0,1\}^m$ *and an arbitrary key* $K^* \in \{0,1\}^k$ *(e.g.* $M^* = 1^m$ *and* $K^* = 1^k$*). The dedicated-key hash function* $G : \{0,1\}^k \times \{0,1\}^m \rightarrow \{0,1\}^n$ *shown in this lemma is* $(t',\epsilon')-CR$, *where* $t' = t - cT_H$ *and* $\epsilon' = \epsilon + 2^{-k}$, *but it is completely insecure in eTCR sense.* T_H *denotes the time for one computation of* H *and* c *is a small constant.*

$$G_K(M) = \begin{cases} M^*_{1\cdots n} & \text{if } M = M^* \bigvee K = K^* \quad (1) \\ H_K(M^*) & \text{if } M \neq M^* \bigwedge K \neq K^* \bigwedge H_K(M) = M^*_{1\cdots n} \quad (2) \\ H_K(M) & \text{otherwise} \quad (3) \end{cases}$$

The proof is valid for any arbitrary selection of parameters $M^* \in \{0,1\}^m$ and $K^* \in \{0,1\}^k$, and hence, this construction actually shows 2^{m+k} such counterexample functions, which are CR but not eTCR.

Proof. Let's first demonstrate that G as a dedicated-key hash function is not secure in eTCR sense. This can be easily shown by the following simple adversary $A = (A_1, A_2)$ playing eTCR game against G. In the first stage of eTCR attack, A_1 outputs the target message as $M = M^*$. In the second stage of the attack, A_2, after receiving the first randomly selected key K (where $K \xleftarrow{\$} \{0,1\}^k$), outputs a different message $M' \neq M^*$ and selects the second key as $K' = K^*$. It can be seen easily that the adversary $A = (A_1, A_2)$ wins the eTCR game, as $M' \neq M^*$ implies that $(M^*, K) \neq (M', K^*)$ and by the construction of G we have $G_K(M^*) = G_{K^*}(M') = M^*_{1\cdots n}$; that is both of the conditions for winning eTCR game are satisfied. Therefore, the hash function family G is completely insecure in eTCR sense.

To complete the proof, we need to show that the hash function family G inherits the CR property of H. This is done by reducing CR security of G to that of H. Let A be an adversary that can win CR game against G with probability ϵ' using time complexity t'. We construct an adversary B against CR property of H with success probability of at least $\epsilon = \epsilon' - 2^{-k}$ ($\approx \epsilon'$, for large k) and time $t = t' + cT_H$ as stated in the lemma. The construction of B and the complete analysis can be found in the full version of this paper in [18]. □

3.2 eTCR $\not\Rightarrow$ CR

We want to demonstrate that the eTCR property does not imply the CR property. That is, the CR property as a security notion for a dedicated-key hash function is not a weaker than the eTCR property. This is done by showing as a counterexample, a dedicated-key hash function which is secure in eTCR sense but completely insecure in CR sense.

Lemma 2 (eTCR does not imply CR). *Assume that there exists a dedicated-key hash function $H : \{0,1\}^k \times \{0,1\}^m \to \{0,1\}^n$, where $m > k \geq n$, which is $(t, \epsilon) - eTCR$. The dedicated-key hash function $G : \{0,1\}^k \times \{0,1\}^m \to \{0,1\}^n$ shown in this lemma is $(t', \epsilon') - eTCR$, where $t' = t - c$, $\epsilon' = \epsilon + 2^{-k+1}$, but it is completely insecure in CR sense. (c is a small constant.)*

$$G_K(M) = \begin{cases} H_K(0^{m-k}||K) & \text{if } M = 1^{m-k}||K \\ H_K(M) & \text{otherwise} \end{cases}$$

Proof. We firstly demonstrate that G as a dedicated-key hash function is not secure in CR sense. This can be easily shown by the following simple adversary A that plays CR game against G. On receiving the key K, the adversary A outputs two different messages as $M = 1^{m-k}||K$ and $M' = 0^{m-k}||K$ and wins the CR game as we have $G_K(1^{m-k}||K) = H_K(0^{m-k}||K) = G_K(0^{m-k}||K)$.

It remains to show that that G indeed is an eTCR secure hash function family. Let $A = (A_1, A_2)$ be an adversary which wins the eTCR game against G with probability ϵ' and using time complexity t'. We construct an adversary $B = (B_1, B_2)$ which uses A as a subroutine and wins eTCR game against H with success probability of at least $\epsilon = \epsilon' - 2^{-k+1}(\approx \epsilon'$, for large $k)$ and having

time complexity $t = t' + c$ where small constant c can be determined from the description of algorithm B. The description of the algorithm B and the complete analysis can be found in the full version of this paper in [18]. □

3.3 The Case for Randomized Hashing

Randomized Hashing method is a simple method to obtain a dedicated-key hash function $\tilde{H} : \mathcal{K} \times M \rightarrow \{0,1\}^n$ from an iterated (keyless) hash function H as $\tilde{H}(K, M) \triangleq H\big(K||(M_1 \oplus K)|| \cdots ||(M_L \oplus K)\big)$, where $\mathcal{K} = \{0,1\}^b$ and H itself is constructed by iterating a keyless compression function $h : \{0,1\}^{n+b} \rightarrow \{0,1\}^n$ and using a *fixed* initial chaining value IV. The analysis in [10] reduces the security of $\tilde{H}$ in eTCR sense to some assumptions, called c-SPR and e-SPR, on the keyless compression function h which are weaker than the keyless-CR assumption on h.

Here, we are interested in a somewhat different question, namely whether (formally definable) CR for this specific design of dedicated-key hash function $\tilde{H}$ implies that it is eTCR or not. Interestingly, we can gather a strong evidence that CR for $\tilde{H}$ implies that it is also eTCR, by the following argument. First, from the construction of $\tilde{H}$ it can be seen that CR for $\tilde{H}$ implies keyless-CR for a hash function H^* which is identical to the H except that its initial chaining value is a random and *known* value $IV^* = h(IV||K)$ instead of the prefixed IV (Note that K is selected at random and is provided to the adversary at the start of CR game). This is easily proved, as any adversary that can find collisions for H^* (*i.e.* breaks it in keyless-CR sense) can be used to construct an adversary that can break $\tilde{H}$ in CR sense. Second, from recent cryptanalysis methods which use differential attacks to find collisions [25, 24], we have a strong evidence that finding collisions for H^* under known IV^* would not be harder than finding collisions for H under IV, for a practical hash function like MD5 or SHA-1. That is, we argue that if H^* is keyless-CR then H is also keyless-CR. Finally, we note that keyless-CR assumption on H in turn implies that $\tilde{H}$ is eTCR as follows. Consider a successful eTCR attack against $\tilde{H}$ where on finishing the attack we will have $(K, M) \neq (K', M')$ and $\tilde{H}(K, M) = \tilde{H}(K', M')$, where $M = M_1|| \cdots ||M_L$ and $M' = M'_1|| \cdots ||M'_L$. Referring to the construction of $\tilde{H}$ this is translated to $H\big(K||(M_1 \oplus K)|| \cdots ||(M_L \oplus K)\big) = H\big(K||(M'_1 \oplus K)|| \cdots ||(M'_L \oplus K)\big)$ and from $(K, M) \neq (K', M')$ we have that $\big(K||(M_1 \oplus K)|| \cdots ||(M_L \oplus K)\big) \neq \big(K||(M'_1 \oplus K)|| \cdots ||(M'_L \oplus K)\big)$. Hence, we have found a collision for H and this contradicts the assumption that H is keyless-CR. Therefore, for the case of the specific dedicated-key hash function $\tilde{H}$ obtained via Randomized Hashing mode, it can be argued that CR implies eTCR.

4 Domain Extension and eTCR Property Preservation

In this section we investigate the eTCR preserving capability of eight domain extension transforms, namely Plain MD [12, 7], Strengthened MD [12, 7], Prefix-free MD [6, 11], Randomized Hashing [10], Shoup [21], Enveloped Shoup [2], XOR Linear Hash (XLH)[4], and Linear Hash (LH) [4] methods.

Assume that we have a compression function $h : \{0,1\}^k \times \{0,1\}^{n+b} \rightarrow \{0,1\}^n$ that can only hash messages of fixed length $(n + b)$ bits. A domain extension transform can use this compression function (as a black-box) to construct a hash function $H : \mathcal{K} \times \mathcal{M} \rightarrow \{0,1\}^n$, where the message space $\mathcal{M}$ can be either $\{0,1\}^*$ or $\{0,1\}^{<2^m}$, for some positive integer m (e.g. $m = 64$). The key space $\mathcal{K}$ is determined by the construction of a domain extender. Clearly $log_2(|\mathcal{K}|) \geq k$, as H involves at least one invocation of h. The difference between $log_2(|\mathcal{K}|)$ (i.e. the key length of H) and k (i.e. the key length of h) is called the 'key expansion' of domain extension transform and is a measure of its efficiency: the less key expansion is, the more efficient the domain extension transform will be.

A domain extension transform comprises of two functions: an injective padding function Pad and an iteration function f_I. First, the padding function $Pad : \mathcal{M} \rightarrow D_I$ is applied to an input message $M \in \mathcal{M}$ to convert it to the padded message $Pad(M)$ in a domain D_I. Then, the iteration function $f_I : \mathcal{K} \times D_I \rightarrow \{0,1\}^n$ uses the compression function h as many times as required, and outputs the final hash value. The full-fledged hash function H is obtained by combining the two functions.

The padding functions used in the eight domain extension transforms that we consider in this paper are defined as follows:

- **Plain:** $pad : \{0,1\}^* \rightarrow \bigcup_{L \geq 1} \{0,1\}^{Lb}$, where $pad(M) = M||10^p$ and p is the minimum number of 0's required to make the length of $pad(M)$ a multiple of block length.
- **Strengthening:** $pad_s : \{0,1\}^{<2^m} \rightarrow \bigcup_{L \geq 1} \{0,1\}^{Lb}$, where $pad_s(M) = M||10^p||\langle |M| \rangle_m$ and p is the minimum number of 0's required to make the length of $pad_s(M)$ a multiple of block length.
- **Prefix-free:** $padPF : \{0,1\}^* \rightarrow \bigcup_{L \geq 1} \{0,1\}^{Lb}$, where $padPF$ transforms the input message space $\{0,1\}^*$ to a prefix-free message space, *i.e.* $padPF(M)$ is not a prefix of $padPF(M')$ for any two *distinct* messages M and M'. An example of a Prefix-free padding function, which we consider in this paper, is as follows. Append 10^p to the message where p is the minimum number of 0's required to make the length of the resulted message a multiple of $b - 1$ bits. Parse the resulted message into blocks of $b - 1$ bits and prepend a '0' to all blocks but the final block where a '1' must be prepended.
- **Strengthened Chain Shift:** $padCS_s : \{0,1\}^{<2^m} \rightarrow \bigcup_{L \geq 1} \{0,1\}^{Lb+b-n}$, where $padCS_s(M) = M||10^r||\langle |M| \rangle_m ||0^p$, and parameters p and r are defined in two ways depending on the block length b. If $b \geq n + m$ then $p = 0$, otherwise $p = b - n$. Then r is the minimum number of 0's required to make the padded message a member of $\{0,1\}^{Lb+b-n}$, for some positive integer L.

The iteration functions for MD, Randomized Hashing, Shoup, Enveloped Shoup, XLH and LH are shown in Fig. 1.

4.1 Merkle-Damgård Does Not Preserve eTCR

MD iteration function as shown in Fig. 1 can be used together with Plain (pad), Strengthening(pad_s), or Prefix-free($padPF$) padding function to construct

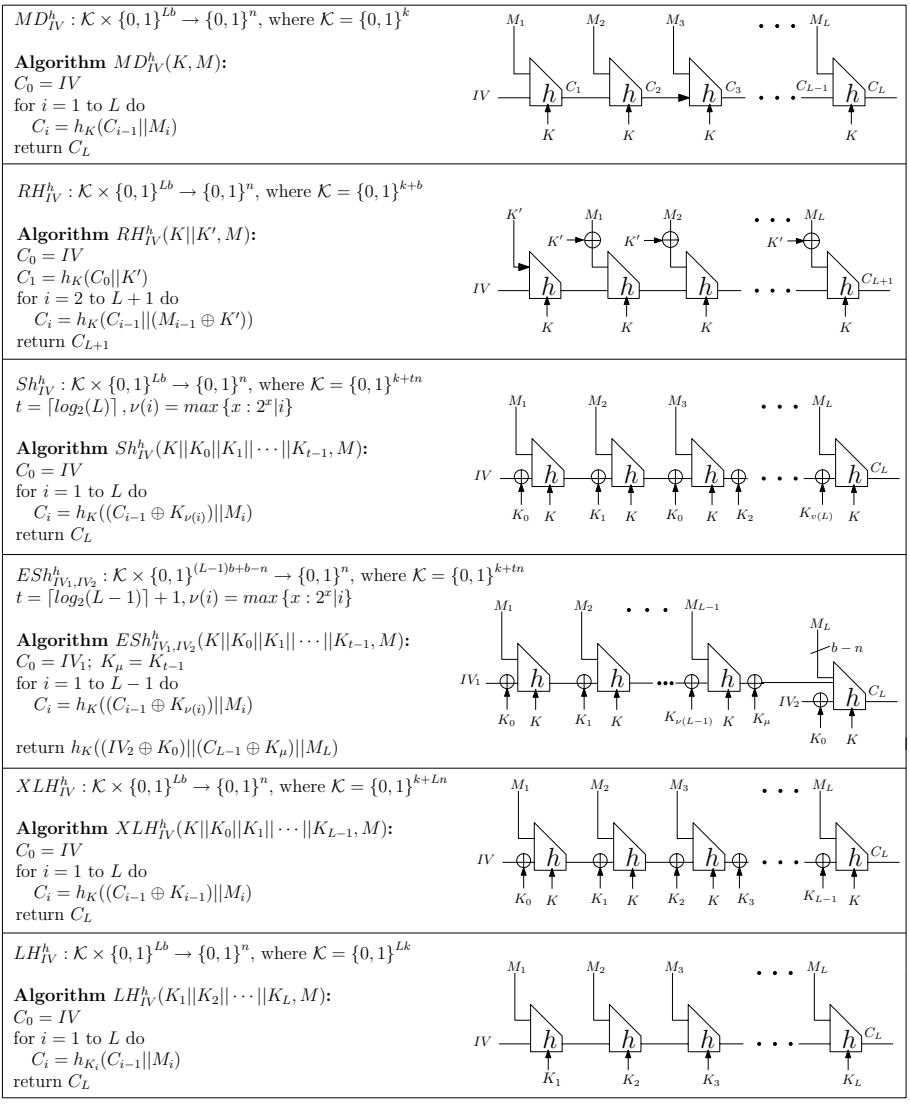

Fig. 1. Iteration functions used in domain extension transforms: Merkle-Damgård (MD), Randomized Hashing (RH), Shoup (Sh), Enveloped Shoup (ESh), XLH and LH. The iteration functions are ordered top-down based on their efficiency in terms of key expansion, MD iteration does not expand the key length of underlying compression function and is the most efficient transform and LH is the least efficient transform.

a domain extension transform, which is called Plain MD, Strengthened MD, or Prefix-free MD, respectively. In this section we show that none of these three domain extension transforms can be used as an eTCR preserving domain extender.

Theorem 1 (Negative Result). *Plain MD, Strengthened MD, and Prefix-free MD do not preserve eTCR.*

Proof. We borrow the construction of the following counterexample from [4] where it was used in the context of TCR property. Assume that there is a dedicated-key compression function $g : \{0,1\}^k \times \{0,1\}^{n+b} \to \{0,1\}^n$ with $b > k$ which is (t, ϵ)-eTCR secure. Set $b = k + b'$ where $b' > 0$ by the assumption that $b > k$. Consider the following dedicated-key compression function $h : \{0,1\}^k \times \{0,1\}^{(n+k)+b'} \to \{0,1\}^{n+k}$:

$$h(K, X||Y||Z) = h_K(X||Y||Z) = \begin{cases} g_K(X||Y||Z)||K & \text{if } K \neq Y \\ 1^{n+k} & \text{if } K = Y \end{cases}$$

where $K \in \{0,1\}^k, X \in \{0,1\}^n, Y \in \{0,1\}^k, Z \in \{0,1\}^{b'}$ ($n + k$ is chaining variable length and b' is block length for h).

To complete the proof, we first show in Lemma 3 that h_K inherits the eTCR property from g_K. Note that this cannot be directly inferred from the proof in [4] that h_K inherits the weaker notion TCR from g_K. Then, we show a simple attack in each case to show that the hash function obtained via either of Plain, Strengthened, or Prefix-free MD transform by extending domain of h_K is completely insecure in eTCR sense.

Lemma 3. *The dedicated-key compression function h is (t', ϵ')-eTCR secure, where $\epsilon' = \epsilon + 2^{-k+1} \approx \epsilon$ and $t' = t - c$, for a small constant c.*

Proof. Let $A = (A_1, A_2)$ be an adversary which wins the eTCR game against h_K with probability ϵ' and using time complexity t'. We construct an adversary $B = (B_1, B_2)$ which uses A as a subroutine and wins eTCR game against g_K with success probability of at least $\epsilon = \epsilon' - 2^{-k+1} (\approx \epsilon'$, for large $k)$ and spending time complexity $t = t' + c$ where small constant c can be determined from the description of algorithm B. Algorithm B is as follows:

Algorithm $B_1()$	**Algorithm** $B_2(K_1, M_1, State)$
$(M_1 = X_1\|\|Y_1\|\|Z_1, State) \xleftarrow{\$} A_1()$;	Parse M_1 as $M_1 = X_1\|\|Y_1\|\|Z_1$
return $(M_1, State)$;	if $\left[K_1 = Y_1 \bigvee K_1 = 1^k\right]$ **return 'Fail'**;
	$(M_2{=}X_2\|\|Y_2\|\|Z_2, K_2) \xleftarrow{\$} A_2(K_1, M_1, State)$;
	return (M_2, K_2);

At the first stage of eTCR attack, B_1 just merely runs A_1 and returns whatever it returns as the first message (*i.e.* $M_1 = X_1||Y_1||Z_1$) and any possible state information to be passed to the second stage algorithm. At the second stage of the attack, let **Bad** be the event that $[K_1 = Y_1 \bigvee K_1 = 1^k]$. If **Bad** happens then algorithm B_2 (and hence B) will fail in eTCR attack; otherwise (*i.e.* if $\overline{\textbf{Bad}}$ happens) we show that B will be successful in eTCR attack against g whenever A succeeds in eTCR attack against h.

Assume that the event $\overline{\textbf{Bad}}$ happens; that is, $[K_1 \neq Y_1 \bigwedge K_1 \neq 1^k]$. We claim that in this case if A succeeds then B also succeeds. Referring to the construction

of (counterexample) compression function h in this lemma, it can be seen that if A succeeds, *i.e.*, whenever $(M_1, K_1) \neq (M_2, K_2) \bigwedge h_{K_1}(M_1) = h_{K_2}(M_2)$, it must be the case that $g_{K_1}(M_1)||K_1 = g_{K_2}(M_2)||K_2$ which implies that $g_{K_1}(M_1) = g_{K_2}(M_2)$ (and also $K_1 = K_2$). That is, (M_1, K_1) and (M_2, K_2) are also valid a colliding pair for the eTCR attack against g. (Remember that $M_1 = X_1||Y_1||Z_1$ and $M_2 = X_2||Y_2||Z_2$.)

Now note that $\Pr[\mathbf{Bad}] \leq \Pr[K_1 = Y_1] + \Pr[K_1 = 1^k] = 2^{-k} + 2^{-k} = 2^{-k+1}$, as K_1 is selected uniformly at random just after the message M_1 is fixed in the eTCR game. Therefore, we have $\epsilon = \Pr[B \text{ succeeds}] = \Pr[A \text{ succeeds} \wedge \overline{\mathbf{Bad}}] \geq \Pr[A \text{ succeeds}] - \Pr[\mathbf{Bad}] \geq \epsilon' - 2^{-k+1}$.

To complete the proof of Theorem 1, we need to show that MD transforms cannot preserve eTCR while extending the domain of this specific compression function h_K. For this part, the same attacks that used in [4, 2] against TCR property also work for our purpose here as clearly breaking TCR implies breaking its strengthened variant eTCR. The eTCR attacks are as follows:

The Case of Plain MD and Strengthened MD:
Let's denote the Plain MD and Strengthened MD domain extension transforms, respectively, by **pMD** and **sMD**. The following adversary $A = (A_1, A_2)$ can break the hash function obtained using either of **pMD** or **sMD** transforms, in the eTCR sense. A_1 outputs $M_1 = 0^{b'}||0^{b'}$ and A_2, on receiving the first key K, outputs a different message as $M_2 = 1^{b'}||0^{b'}$ together with the same key K as the second key. Considering that the initial value $IV = IV_1||IV_2 \in \{0,1\}^{n+k}$ is fixed before adversary starts the attack game and K is chosen at random afterward in the second stage of the game, we have $\Pr[K = IV_2] = 2^{-k}$. If $K \neq IV_2$ which is the case with probability $1 - 2^{-k}$ then adversary becomes successful as we have:

$$\begin{aligned} MD^h_{IV}(K, 0^{b'}||0^{b'}) &= h_K(h_K(IV_1||IV_2||0^{b'})||0^{b'}) = h_K(g_K(IV_1||IV_2||0^{b'})||K||0^{b'}) \\ &= 1^{n+k} \\ MD^h_{IV}(K, 1^{b'}||0^{b'}) &= h_K(h_K(IV_1||IV_2||1^{b'})||0^{b'}) = h_K(g_K(IV_1||IV_2||1^{b'})||K||0^{b'}) \\ &= 1^{n+k} \end{aligned}$$

$$\mathbf{pMD}: \begin{cases} MD^h_{IV}(K, pad(0^{b'}||0^{b'})) = h_K(MD^h_{IV}(K, 0^{b'}||0^{b'})||10^{b'-1}) \\ \qquad\qquad\qquad\qquad = h_K(1^{n+k}||10^{b'-1}) \\ MD^h_{IV}(K, pad(1^{b'}||0^{b'})) = h_K(MD^h_{IV}(K, 1^{b'}||0^{b'})||10^{b'-1}) \\ \qquad\qquad\qquad\qquad = h_K(1^{n+k}||10^{b'-1}) \end{cases}$$

$$\mathbf{sMD}: \begin{cases} MD^h_{IV}(K, pad_s(0^{b'}||0^{b'})) = h_K(MD^h_{IV}(K, 0^{b'}||0^{b'})||10^{b'-m-1}||\langle 2b' \rangle_m) \\ \qquad\qquad\qquad\qquad = h_K(1^{n+k}||10^{b'-m-1}||\langle 2b' \rangle_m) \\ MD^h_{IV}(K, pad_s(1^{b'}||0^{b'})) = h_K(MD^h_{IV}(K, 1^{b'}||0^{b'})||10^{b'-m-1}||\langle 2b' \rangle_m) \\ \qquad\qquad\qquad\qquad = h_K(1^{n+k}||10^{b'-m-1}||\langle 2b' \rangle_m) \end{cases}$$

The Case of Prefix-free MD: Denote Prefix-free MD domain extension transform by **preMD**. The full-fledged hash function $H : \{0,1\}^k \times \mathcal{M} \rightarrow \{0,1\}^{n+k}$ will be defined as $H(K,M) = \mathbf{preMD}^h_{IV}(K,M) = MD^h_{IV}(K, padPF(M))$. Note that we have $\mathcal{M} = \{0,1\}^*$ due to the application of $padPF$ function. The following adversary $A = (A_1, A_2)$ which is used for TCR attack against Prefix-free MD in [2], can also break H in eTCR sense, as clearly any TCR attacker against H is an eTCR attacker as well. Here, we provide the description of the attack for eTCR, for completeness. A_1 outputs $M_1 = 0^{b'-1}||0^{b'-2}$ and A_2 on receiving the first key K outputs a different message as $M_2 = 1^{b'-1}||0^{b'-2}$ together with the same key K as the second key. Considering that the initial value $IV = IV_1||IV_2 \in \{0,1\}^{n+k}$ is fixed before the adversary starts the attack game and K is chosen at random afterward, we have $\Pr[K = IV_2] = 2^{-k}$. If $K \neq IV_2$ which is the case with probability $1 - 2^{-k}$ then the adversary becomes successful as we have:

$$\begin{aligned}
MD^h_{IV}(K, padPF(0^{b'-1}||0^{b'-2})) &= MD^h_{IV}(K, 0^{b'}||10^{b'-2}1) \\
&= h_K(h_K(IV_1||IV_2||0^{b'})||10^{b'-2}1) \\
&= h_K(g_K(IV_1||IV_2||0^{b'})||K||10^{b'-2}1) \\
&= 1^{n+k} \\
MD^h_{IV}(K, padPF(1^{b'-1}||0^{b'-2})) &= MD^h_{IV}(K, 01^{b'-1}||10^{b'-2}1) \\
&= h_K(h_K(IV_1||IV_2||01^{b'-1})||10^{b'-2}1) \\
&= h_K(g_K(IV_1||IV_2||01^{b'-1})||K||10^{b'-2}1) \\
&= 1^{n+k}
\end{aligned}$$

4.2 Randomized Hashing Does Not Preserve eTCR

Our aim in this section is to show that Randomized Hashing (RH) construction, *if considered as a domain extension for a dedicated-key compression function*, does not preserve eTCR property. Note that (this dedicated-key variant of) RH method as shown in Fig. 1 expands the key length of the underlying compression function by only a constant additive factor of b bits, that is $log_2(|\mathcal{K}|) = k + b$. This characteristic, *i.e.* a small and message-length-independent key expansion could have been considered a stunning advantage from efficiency viewpoint, if RH had been able to preserve eTCR. Nevertheless, unfortunately we shall show that randomized hashing does not preserve eTCR.

Following the specification of the original scheme for Randomized Hashing in [10], we assume that the padding function is the strengthening padding pad_s. The full-fledged hash function $H : \{0,1\}^k \times \mathcal{M} \rightarrow \{0,1\}^{n+k}$ will be defined as $H(K||K', M) = RH^h_{IV}(K||K', pad_s(M))$. Note that we have $\mathcal{M} = \{0,1\}^{<2^m}$ due to the application of pad_s function.

Theorem 2 (Negative Result). *The Randomized Hashing transform does not preserve eTCR.*

Proof. We use the same counterexample as used in the proof of Theorem 1 to show that Randomized Hashing transform does not preserve eTCR property.

As we have previously shown in Lemma 3 that the constructed h_K inherits the eTCR property of g_K, it just remains to show that RH^h_{IV} cannot extend the domain of h_K while preserving its eTCR property. Consider an adversary $A = (A_1, A_2)$ that plays the eTCR game against the hash function H, obtained via Randomized Hashing, as follows. A_1 outputs a one-block long target message $M_1 = 0^{b'}$ (note that for the counterexample compression function h_K, b' is the block length and $n + k$ is the chaining variable length). A_2 on getting the first key $K||K'$ for H (in the second stage of eTCR attack), outputs the second message as $M_2 = 1^{b'}$ and puts the second key the same as the first key. As $M_2 \neq M_1$, we just need to show that these two messages collide under the same key, *i.e.* $K||K'$. Considering that the initial value $IV = IV_1||IV_2 \in \{0,1\}^{n+k}$ for RH^h_{IV} is (selected and) fixed before the adversary starts the attack game and $K||K'$ is chosen at random latter in the second stage of the game, we have $\Pr[K = IV_2] = 2^{-k}$. If $K \neq IV_2$ (which is the case with probability $1 - 2^{-k}$) then the adversary $A = (A_1, A_2)$ becomes successful as we have:

$$
\begin{aligned}
&RH^h_{IV}(K||K', pad_s(0^{b'})) = RH^h_{IV}(K||K', 0^{b'}||10^{b'-1-m}\langle b'\rangle_m)\\
&= h_K\Big(h_K\big(h_K(IV_1||IV_2||K')||(K' \oplus 0^{b'})\big)||(K' \oplus 10^{b'-1-m}\langle b'\rangle_m)\Big)\\
&= h_K\Big(h_K\big(g_K(IV_1||IV_2||K')||K||(K' \oplus 0^{b'})\big)||(K' \oplus 10^{b'-1-m}\langle b'\rangle_m)\Big)\\
&= h_K(1^{n+k}||(K' \oplus 10^{b'-1-m}\langle b'\rangle_m))
\end{aligned}
$$

$$
\begin{aligned}
&RH^h_{IV}(K||K', pad_s(1^{b'})) = RH^h_{IV}(K||K', 1^{b'}||10^{b'-1-m}\langle b'\rangle_m)\\
&= h_K\Big(h_K\big(h_K(IV_1||IV_2||K')||(K' \oplus 1^{b'})\big)||(K' \oplus 10^{b'-1-m}\langle b'\rangle_m)\Big)\\
&= h_K\Big(h_K\big(g_K(IV_1||IV_2||K')||K||(K' \oplus 1^{b'})\big)||(K' \oplus 10^{b'-1-m}\langle b'\rangle_m)\Big)\\
&= h_K(1^{n+k}||(K' \oplus 10^{b'-1-m}\langle b'\rangle_m))
\end{aligned}
$$

□

4.3 Shoup, Enveloped Shoup and XLH Do Not Preserve eTCR

In previous subsections, we have shown that neither MD nor RH are eTCR preserving transforms. The next three most efficient candidates from key expansion viewpoint that we consider are Shoup (Sh), Enveloped Shoup (ESh) and XLH transforms.

Theorem 3 (Negative Results). *Sh, ESh, and XLH transforms do not preserve eTCR.*

Proof. The proof is quite simple but the results are stronger than previous counterexample based proofs, as here the negative results hold for any arbitrary compression function (irrespective of how secure the compression function h is), and not only for some specific counterexamples. That is, *these XOR masking based domain extension transforms are structurally insecure in eTCR sense*. Intuitively, the inability if these domain extenders to preserve eTCR is due to the fact that they use XOR operation to add the key to the internal state

(*i.e.* chaining variable), and hence an eTCR adversary will be able to cancel internal differences by taking advantage of its ability to select the value of the second key in the second stage of eTCR attack. (We note that this is also the case for the XTH scheme of [4].)

For the formal proof, we provide the following simple eTCR attack against Shoup construction. The attacks for the cases of ESh and XLH are quite similar and can be found in the full version of this paper in [18].

Consider the hash function obtained via Shoup domain extension transform, *i.e.* pad_s padding function followed by Sh^h_{IV} iteration method. The following simple adversary $A = (A_1, A_2)$ can break it in the eTCR sense. At the first stage of the eTCR attack, A_1 outputs a two-block message $M = M_1||M_2$ as the target message which after applying pad_s will become a three-block message $M_1||M_2||(10^{b-1-m}\langle 2b\rangle_m)$ to be input to the three-round Sh^h_{IV} iteration. In the second stage of eTCR game, A_2, after receiving the first key as $K||K_0||K_1||K_0$ from the challenger, chooses the second two-block message as $M' = M'_1||M_2$ which after padding becomes $M'_1||M_2||(10^{b-1-m}\langle 2b\rangle_m)$. A_2 also puts the second key as $K||K_0||K'_1||K_0$, where the value of K'_1 is computed as $K'_1 = K_1 \oplus h_K\big((IV \oplus K_0)||M_1\big) \oplus h_K\big((IV \oplus K_0)||M'_1\big)$. It is easy to see (referring to Fig. 1) that this value for K' cancel the introduced difference in chaining variable which was created due to the different message blocks M_1 and M'_1. So, $(K||K_0||K_1, M)$ and $(K||K_0||K'_1, M')$ constitute a colliding pair for H in eTCR sense. (Note that the key sequence used for iteration function Sh^h_{IV} is $K||K_0||K_1||K_0$ because padded message $pad_s(M)$ has an extra third block containing the length information.) □

4.4 LH Transform and Its Nested Variant

Up to know we have shown that neither of MD, RH, Sh, or XLH transforms can preserve eTCR property. Henceforth, we have lost all efficient methods from key expansion viewpoint and now we have reached to the same starting-point (and the least efficient) transform for TCR preserving scenario as in [4], *i.e.* the LH method whose key expansion is linear in the message length. We now consider whether at least (but hopefully not the last) this LH transform or its variants can be used for eTCR preserving domain extension or not. Fortunately, we gather a positive answer for this. The proof for this positive result is a straightforward extension of the methodology used in [4] for the case of TCR, but with some necessary adaptations required for considering eTCR attack scenario where adversary has more power in second stage by getting to choose a different key as well as a different message. Firstly, in Theorem 4 we show that if the compression function h is eTCR secure then the hash function LH^h_{IV} will be secure against a restricted class of eTCR adversaries which only find equal-length colliding pairs. Let's denote this equal-length eTCR notion by eTCR*. Secondly, it is shown in Theorem 5 that a *nested variant* of LH can be made eTCR secure, *i.e.* against any arbitrary adversary. The proofs for these two theorems can be found in the full version of this paper in [18].

Assume that the input messages have length a multiple of block length and the maximum length in blocks is some positive integer N, *i.e.* $|M| \leq Nb$ where b is the length of one block in bits. This restriction of message space to a domain with messages of variable but multiple-block length can be easily removed by using any proper injective padding function like plain padding function *pad*. LH^h_{IV} iteration function can be used to define a hash function as $H(K_1||\cdots||K_N, M) \triangleq LH^h_{IV}(K_1||\cdots||K_m, M)$, where m is the length of M in blocks.

Theorem 4 (Positive Result: eTCR*). *Assume that the compression function* $h : \{0,1\}^k \times \{0,1\}^{n+b} \rightarrow \{0,1\}^n$ *is* (t,ϵ)*-eTCR. Then the hash function* $H : \{0,1\}^{Nk} \times \{0,1\}^{\leq Nb} \rightarrow \{0,1\}^n$ *obtained using* LH^h_{IV} *iteration of* h*, will be* (t',ϵ')*-eTCR***, where* $\epsilon' = N\epsilon$*,* $t' = t - \Theta(N)\big(T_h + n + b + k\big)$*, where* T_h *is the time for one computation of the compression function* h*.*

Theorem 5 (From eTCR* to eTCR). *Assume that* $H_1 : \{0,1\}^{k_1} \times \mathcal{M} \rightarrow \{0,1\}^n$ *is* (t_1,ϵ_1)*-eTCR** *hash function and* $h : \{0,1\}^{k_2} \times \{0,1\}^{n+b} \rightarrow \{0,1\}^n$ *is* (t_2,ϵ_2)*-eTCR compression function, where* $b \geq \lceil log_2(|M|)\rceil$*, for any* $M \in \mathcal{M}$*. Then the composition function* $H : \{0,1\}^{k_1+k_2} \times \mathcal{M} \rightarrow \{0,1\}^n$*, defined as* $H(K1||K2, M) = h(K2, H_1(K1, M)||\langle|M|\rangle_b)$*, will be* (t,ϵ)*-eTCR; where* $\epsilon = \epsilon_1 + 2\epsilon_2$*, and* $t = min\{t_1 - k_2, t_2 - k_1 - 2T_{H_1} - 2b\}$*.*

Nested Linear Hash: Let H_1 be the equal-length eTCR hash function obtained via LH transform as stated in Theorem 4. From Theorem 5 we can obtain a variant of LH which is eTCR secure. This variant which we call it Nested LH is obtained by the composition of H_1 with an eTCR compression function h, that is, LH nested by this final application of the compression function in the way stated in Theorem 5 (*i.e.* final block is just $\langle|M|\rangle_b$). Theorem 5 and Theorem 4 show that this Nested LH will be eTCR if the compression function is eTCR. Alternatively, this Nested LH construction can be seen as obtained using a *variant* of strengthening padding followed by LH iteration on the compression function h. This variant of strengthening padding, which can be called full-final-block strengthening, acts as follows. On input a message M, append the message by 10^r to make its length a multiple of block length and then append another full block which only contains the representation of length of M in an exactly b-bit string, i.e. $\langle|M|\rangle_b$.

5 Conclusion

The introduction of the eTCR property by Halevi and Krawczyk [10] has been proven to be very useful to enrich the notions of hash functions, in particular with its application to construct the Randomized Hashing mode which has been announced by NIST as Draft SP 800-106. Nonetheless, the relationships between eTCR with the existing properties of hash functions need to be further studied. In this paper, we showed that there is a *separation* between the new eTCR property with the well-known collision resistance (CR) property, where both properties

are considered for a dedicated-key hash function. Furthermore, when considering the problem of eTCR property preserving domain extension, we found that the only eTCR preserving method is a nested variant of LH which has a drawback of having high key expansion factor. Therefore, it is interesting to design a new eTCR preserving domain extension in the *standard model*, which is *efficient*. We left this as an open problem in this paper.

Acknowledgments. We would like to thank the anonymous reviewers of FSE 2009 for their insightful comments and suggestions.

References

1. Andreeva, E., Neven, G., Preneel, B., Shrimpton, T.: Seven-Property-Preserving Iterated Hashing: ROX. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 130–146. Springer, Heidelberg (2007)
2. Bellare, M., Ristenpart, T.: Hash Functions in the Dedicated-Key Setting: Design Choices and MPP Transforms. In: Arge, L., Cachin, C., Jurdziński, T., Tarlecki, A. (eds.) ICALP 2007. LNCS, vol. 4596, pp. 399–410. Springer, Heidelberg (2007)
3. Bellare, M., Ristenpart, T.: Multi-Property-Preserving Hash Domain Extension and the EMD Transform. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 299–314. Springer, Heidelberg (2006)
4. Bellare, M., Rogaway, P.: Collision-Resistant Hashing: Towards Making UOWHFs Practical. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 470–484. Springer, Heidelberg (1997)
5. Contini, S., Yin, Y.L.: Forgery and Partial Key-Recovery Attacks on HMAC and NMAC using Hash Collisions. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 37–53. Springer, Heidelberg (2006)
6. Coron, J.S., Dodis, Y., Malinaud, C., Puniya, P.: Merkle-Damgård Revisited: How to Construct a Hash Function. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 430–448. Springer, Heidelberg (2005)
7. Damgård, I.: A Design Principle for Hash Functions. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 416–427. Springer, Heidelberg (1990)
8. den Boer, B., Bosselaers, A.: Collisions for the Compressin Function of MD5. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 293–304. Springer, Heidelberg (1993)
9. Dodis, Y., Puniya, P.: Getting the Best Out of Existing Hash Functions; or What if We Are Stuck with SHA? In: Bellovin, S.M., Gennaro, R., Keromytis, A.D., Yung, M. (eds.) ACNS 2008. LNCS, vol. 5037, pp. 156–173. Springer, Heidelberg (2008)
10. Halevi, S., Krawczyk, H.: Strengthening Digital Signatures Via Randomized Hashing. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 41–59. Springer, Heidelberg (2006)
11. Maurer, U.M., Sjödin, J.: Single-Key AIL-MACs from Any FIL-MAC. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, pp. 472–484. Springer, Heidelberg (2005)
12. Merkle, R.C.: One Way Hash Functions and DES. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 428–446. Springer, Heidelberg (1990)
13. Mironov, I.: Hash Functions: From Merkle-Damgård to Shoup. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 166–181. Springer, Heidelberg (2001)

14. Naor, M., Yung, M.: Universal One-Way Hash Functions and Their Cryptographic Applications. In: STOC 1989, pp. 33–43. ACM, New York (1989)
15. National Institute of Standards and Technology. Cryptographic Hash Algorithm Competition, http://csrc.nist.gov/groups/ST/hash/sha-3/index.html
16. National Institute of Standards and Technology. Draft NIST SP 800-106: Randomized Hashing for Digital Signatures (August 2008), http://csrc.nist.gov/publications/PubsDrafts.html#SP-800-106
17. Preneel, B.: The State of Cryptographic Hash Functions. In: Damgård, I.B. (ed.) EEF School 1998. LNCS, vol. 1561, pp. 158–182. Springer, Heidelberg (1998)
18. Reyhanitabar, M.R., Susilo, W., Mu, Y.: Enhanced Target Collision Resistant Hash Functions Revisited. IACR ePrint Archive, Report 2009/051 (2009), http://eprint.iacr.org/2009/051
19. Rogaway, P.: Formalizing Human Ignorance: Collision-Resistant Hashing without the Keys. In: Nguyên, P.Q. (ed.) VIETCRYPT 2006. LNCS, vol. 4341, pp. 211–228. Springer, Heidelberg (2006)
20. Rogaway, P., Shrimpton, T.: Cryptographic Hash-Function Basics: Definitions, Implications, and Separations for Preimage Resistance, Second-Preimage Resistance, and Collision Resistance. In: Roy, B., Meier, W. (eds.) FSE 2004. LNCS, vol. 3017, pp. 371–388. Springer, Heidelberg (2004)
21. Shoup, V.: A Composition Theorem for Universal One-Way Hash Functions. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 445–452. Springer, Heidelberg (2000)
22. Simon, D.R.: Finding Collisions on a One-Way Street: Can Secure Hash Functions be Based on General Assumptions? In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 334–345. Springer, Heidelberg (1998)
23. Wang, X., Lai, X., Feng, D., Chen, H., Yu, X.: Cryptanalysis of the Hash Functions MD4 and RIPEMD. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 1–18. Springer, Heidelberg (2005)
24. Wang, X., Yin, Y.L., Yu, H.: Finding Collisions in the Full SHA-1. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 17–36. Springer, Heidelberg (2005)
25. Wang, X., Yu, H.: How to Break MD5 and Other Hash Functions. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 19–35. Springer, Heidelberg (2005)