An Investigation of Enhanced Target Collision Resistance Property for Hash Functions: Implications, Separations, and Domain Extension ⋆

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. In this paper we investigate the newly emerged security property called the enhanced target collision resistance (eTCR) for hash functions. eTCR property 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 for strengthening digital signatures. 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). Contributions of this paper are twofold. As our first contribution, we compare the new eTCR property with all of the seven security properties for a hash function formalized by Rogaway and Shrimpton in FSE 2004. We provide a full picture of relationships (i.e. implications and separations) between eTCR and each of the seven properties, namely Collision Resistance (CR), three variants of Second Preimage Resistance (Sec, aSec, eSec) and three variants of Preimage Resistance (Pre, aPre, ePre) where all these properties are considered for a dedicated-key hash function. One of the most interesting results is that there is a *separation* between eTCR and CR, that is *in general*, the new eTCR property as required for randomized hashing based signatures cannot be claimed to be weaker (or stronger) than CR property for *an arbitrary dedicated-key hash function*. As our second contribution, we analyze eTCR property preserving capabilities of several hash domain extension transforms, 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. From this analysis it turns out that with the exception of a *nested variant* of LH construction *none* of the remaining investigated iterative schemes are eTCR preserving.

Key words: Hash Functions, Security Property, eTCR, Domain Extension

1 Introduction

Cryptographic hash function are functions that can map variable length strings to fixed length strings while providing some required security properties. They are used in a vast variety of cryptographic applications and are indispensable part of digital signatures and message authentication codes (e.g. HMAC). Originally designed to make digital signatures more efficient, application of hash functions in schemes following hash-and-sign paradigm, like DSA, requires them to provide collision resistance (CR) property. Hash functions are also asked to provide several different security properties depending on the specific security requirements of the higher-level protocols utilizing them. Although CR is one of the most important and well-known security properties for a hash function, they are often asked to provide many other security properties that, depending on the requirements of the higher-level applications, may range from merely being a one-way function (i.e. preimage resistance property) to acting as a truly random function (i.e. a random oracle). Hence, unlike many other cryptographic primitives which are only aimed to fulfill a specific security notion, hash functions as workhorses of cryptography are usually assumed to provide a wide application dependent spectrum of security properties.

Despite existence of numerous works on design of hash functions and on attacking the hash functions, the current literature contains many different informal and formal definitions for security properties of hash functions and several hash constructions that fulfill (heuristically or provably) some security properties

⋆ A preliminary version of this paper appeared in [20].

but not the others. In regard to any newly introduced security property for hash functions, considering two questions is of an essential theoretical interest, namely the formal relationships with previously known properties and the problem of property preserving hash domain extension (or mode of operation).

Working out the relationships between a newly introduced security notion for hash functions and other well-known and in-use security properties like CR is an essential step to clarify the position of the new property among previously known ones. This should be done by showing implications or separations between properties using formal proofs or counterexamples. There are few works in this line of research investigating and clarifying formal relationships between numerous security notions for hash functions, e.g. [15, 26, 23]. The comprehensive work of Rogaway and Shrimpton in [22] provides formal definitions for seven security notions, namely CR (denoted by 'Coll' in [22]), three variants of second preimage resistance (Sec, aSec, eSec) and three variants of preimage resistance (Pre, aPre, ePre) as well as all relationships between these seven basic properties.

The possibility of designing a 'property preserving hash domain extension' is another important issue to be considered in regard to any new security property. The problem is that whether it is possible to construct a full-fledged (i.e. arbitrary-input-length) hash function to achieve the target security property assuming that one has a compression function (i.e. a fixed-input-length hash function) possessing that security property. In the case of CR property, the seminal works of Merkle [14] and Damgård [8] showed 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 [5, 3, 2, 1].

In this paper we consider the above-mentioned two essential questions, namely investigation of formal relationships and property preserving domain extension, in regard to the newly defined security property of enhanced target collision resistance (eTCR). eTCR was put forth by Halevi and Krawczyk in Crypto 2006 [12] in conjunction with the randomized hashing mode that is used to realize such a hash function family for strengthening digital signatures. This is motivated by the fact that CR property, despite being a widely-desirable security property, has been shown to be a very strong and demanding property for hash functions from theoretical viewpoint [25, 5, 19] as well as being a practically endangered property by the recent advances in cryptanalysis of widely-used standard hash functions like MD5 [29] and SHA-1 [28, 9]. In response to these recent cryptanalytic results against most in-use standard hash functions, NIST created a design competition for the next generation hash function standard which will be called SHA-3 [17]. It is hoped that SHA-3 standard will be able to resist against all known attacks, especially the powerful statistical methods like differential cryptanalysis which have been used to attack MD5, SHA-1 and many other hash functions [29, 28, 27].

While looking forward to such a new SHA-3 standard, Halevi and Krawczyk's approach in randomized hashing mode aims at providing a "safety net" by relaxing the current complete reliance on CR property without having to change the internals of an already implemented and in-use hash function like SHA-1. In a nutshell, Randomized Hashing construction (see Fig. 1) which is proposed in [12] and announced by NIST as Draft SP 800-106 [18] 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.)

Although the main motivation for the design of a randomized hashing mode in [12] was to free reliance on strong collision resistance assumption on the underlying hash function (by making off-line attacks ineffective by using anew and random key), in parallel to this aim, a new security property was also introduced and defined for hash functions, namely the 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 [12], 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

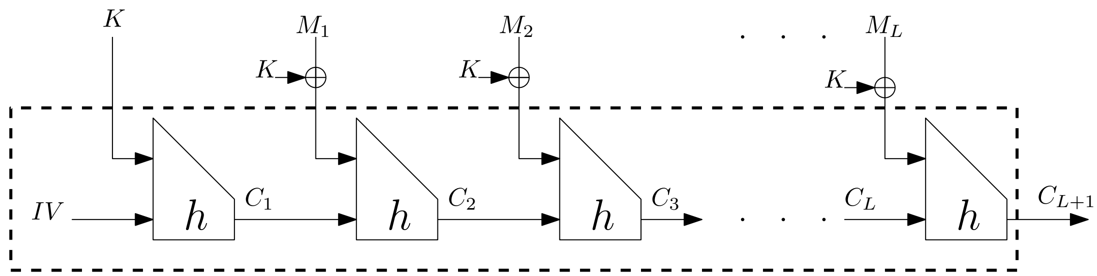

Fig. 1. Randomized Hashing construction

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 [12]. This is an improvement compared to using a TCR (UOWHF) hash function family where one needs to sign $H_K(M)||K$ [5].

Contributions. Our aim in this paper is to investigate the eTCR hashing as a new and interesting notion. As our first contribution, we provide a full picture of relationships between the eTCR property and all of the seven security properties for a hash function formalized by Rogaway and Shrimpton in [22], namely CR, Sec, aSec, eSec, Pre, aPre and ePre notions of security. This is done by working out all implications and separations between eTCR and each of the seven properties by formal proofs or counterexamples. The summary of results is shown in Fig. 1, where in conformance with [22] we have used following conventions to represent relationships: a directed path from a notion x to a notion y shows that x implies y (dashed lines represent "provisional implications" [22] in which the strength of the implications depends on the amount of compression achieved by the hash function) and lack of any path between two notions shows that there is a separation between the two notions.

One of the most interesting results among these relations is the *separation* between eTCR and CR (or 'Coll'), that is *in general*, the new eTCR property cannot be claimed to be weaker (or stronger) than collision resistance property when both notions are considered for *an arbitrary dedicated-key hash function*. At first glance, this may seem to be discouraging for the applications of eTCR hashing, but we emphasize that the separation results actually show the incomparability between two notions. For instance, the separation between eTCR and CR in general does not necessarily imply that for any *specific* construction of a dedicated-key hash function (say the Randomized Hashing construction), achieving the eTCR property will be harder than CR. Although our separation results do not rule out the possibility of designing specific dedicated-key hash functions in which eTCR might be easier to achieve compared to other notions, 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. We investigate eight domain extension transforms for this purpose; namely Plain MD [14, 8], Strengthened MD [14, 8], Prefix-free MD [7, 13], Randomized Hashing [12] (considered in dedicated-key hash setting), Shoup [24], Enveloped Shoup [2], XOR Linear Hash (XLH) [5], and a variant of Linear Hash (LH) [5] 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 [5]) which has the drawback of having linear key expansion factor. From this analysis, design of a new and more *efficient* eTCR preserving domain extender can be considered as an interesting open problem for future research.

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.

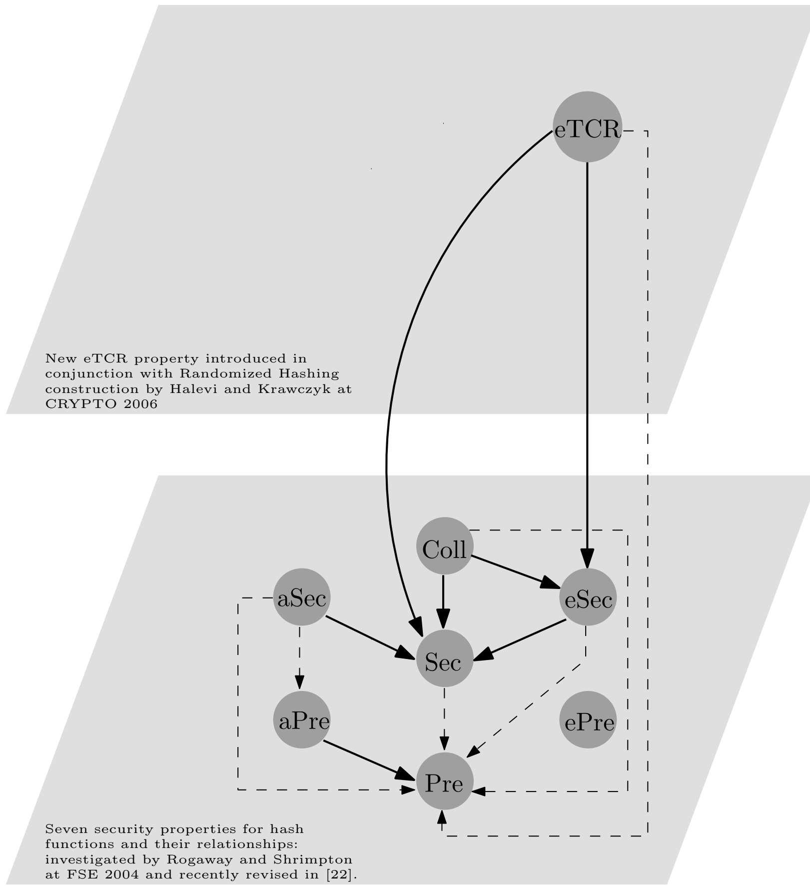

Fig. 2. Relationships between eTCR and the seven security notions for hash functions: a directed path shows an implication (dashed lines represent "provisional implications" [22] in which the strength of the implications depends on the amount of compression achieved by the hash function) and lack of a path shows a separation. Implications and separations between eTCR and each of the seven properties are shown by formal proofs and counterexamples in Sec. 3 of this paper.

Scheme	CR	TCR	eTCR
Plain MD	× [14, 8]	× [5]	<u>×</u>
Strengthened MD	✓[14, 8]	× [5]	<u>×</u>
Prefix-free MD	× [2]	× [2]	<u>×</u>
Randomized Hashing	✓[1]	× [1]	<u>×</u>
Shoup	✓[24]	✓[24]	<u>×</u>
Enveloped Shoup	✓[2]	✓[2]	<u>×</u>
XOR Linear Hash (XLH)	✓[1]	✓[5]	<u>×</u>
Nested Linear Hash (LH)	✓[5]	✓[5]	<u>✓</u>

Table 1. Overview of constructions and the properties they preserve.

2 Preliminaries

2.1 Notations

If A is a randomized algorithm then by $y = A(x_1, \cdots, x_n; R)$ it is meant that y is the output of A on inputs $x_1, \cdots, x_n$ when it is provided with random coins (tape) R. By $y \xleftarrow{\$} A(x_1, \cdots, x_n)$ it is meant that the tape R is chosen at random and y is set to be $y = A(x_1, \cdots, x_n; R)$. 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, and 1^m0^n denote the concatenation of 0^n to 1^m. By (x, y) we mean an injective encoding of two strings x and y, from which one can efficiently recover x and y. 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. Let $val(.)$ be the function that on input a binary string M, considered as an unsigned binary number (with some fixed bit position numbering), returns its decimal value. 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 $M \in \mathcal{M}$. For a formal treatment 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 [22, 12]. 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 [21].

2.3 Definition of Security Notions

In this section, we recall formal definitions of eight security notions for hash functions; namely, the seven notions (Coll, Sec, aSec, eSec, Pre, aPre and ePre) formalized in [22] and the new eTCR notion introduced in [12]. The eSec property in [22] is the same as the well known UOWHF or TCR property [16, 5] and we will use CR to stand for Collision Resistance which is denoted by Coll in [22]. All definitions are for a dedicated-key hash function $H : \mathcal{K} \times \mathcal{M} \rightarrow \{0,1\}^n$, where the key space $\mathcal{K}$ is some nonempty set and the message space $\mathcal{M} \subseteq \{0,1\}^*$ such that $\{0,1\}^m \subseteq \mathcal{M}$ for at least a positive integer m. The advantage measures for an adversary A attacking H are defined in Fig. 3 for the eight security notions.

We say that H is (t, l, ϵ)-xxx, for xxx $\in$ {Coll, Sec, aSec, eSec, Pre, aPre, ePre, eTCR}, if the advantage of any adversary A with time complexity at most t and using messages of length at most l, is less than

$$
\begin{array}{rcl}
\mathrm{Adv}_H^{CR}(A) & = & \Pr\left[K \stackrel{\$}{\leftarrow} \mathcal{K}; (M, M') \stackrel{\$}{\leftarrow} A(K) : \quad M \neq M' \ \wedge \ H_K(M) = H_K(M')\right] \\
\mathrm{Adv}_H^{Sec[m]}(A) & = & \Pr\left[\begin{array}{ll} K \stackrel{\$}{\leftarrow} \mathcal{K}; M \stackrel{\$}{\leftarrow} \{0,1\}^m; & \\ M' \stackrel{\$}{\leftarrow} A(K, M) & : \quad M \neq M' \ \wedge \ H_K(M) = H_K(M') \end{array}\right] \\
\mathrm{Adv}_H^{aSec[m]}(A) & = & \Pr\left[\begin{array}{ll} (K, State) \stackrel{\$}{\leftarrow} A_1(); & \\ M \stackrel{\$}{\leftarrow} \{0,1\}^m; & \\ M' \stackrel{\$}{\leftarrow} A_2(M, State) & : \quad M \neq M' \ \wedge \ H_K(M) = H_K(M') \end{array}\right] \\
\mathrm{Adv}_H^{eSec[m]}(A) & = & \Pr\left[\begin{array}{ll} (M, State) \stackrel{\$}{\leftarrow} A_1(); & \\ K \stackrel{\$}{\leftarrow} \mathcal{K}; & \\ M' \stackrel{\$}{\leftarrow} A_2(K, State) & : \quad M \neq M' \ \wedge \ H_K(M) = H_K(M') \end{array}\right] \\
\mathrm{Adv}_H^{Pre[m]}(A) & = & \Pr\left[\begin{array}{ll} K \stackrel{\$}{\leftarrow} \mathcal{K}; M \stackrel{\$}{\leftarrow} \{0,1\}^m; Y \leftarrow H_K(M); & \\ M' \stackrel{\$}{\leftarrow} A(K, Y) & : \quad H_K(M') = Y \end{array}\right] \\
\mathrm{Adv}_H^{aPre[m]}(A) & = & \Pr\left[\begin{array}{ll} (K, State) \stackrel{\$}{\leftarrow} A_1(); & \\ M \stackrel{\$}{\leftarrow} \{0,1\}^m; Y \leftarrow H_K(M); & \\ M' \stackrel{\$}{\leftarrow} A_2(Y, State) & : \quad H_K(M') = Y \end{array}\right] \\
\mathrm{Adv}_H^{ePre}(A) & = & \Pr\left[(Y, State) \stackrel{\$}{\leftarrow} A_1(); K \stackrel{\$}{\leftarrow} \mathcal{K}; M' \stackrel{\$}{\leftarrow} A_2(K, State) : \quad H_K(M') = Y\right] \\
\mathrm{Adv}_H^{eTCR[m]}(A) & = & \Pr\left[\begin{array}{l} (M, State) \stackrel{\$}{\leftarrow} A_1(); \\ K \stackrel{\$}{\leftarrow} \mathcal{K}; \\ K', M' \stackrel{\$}{\leftarrow} A_2(K, State) : \ (K, M) \neq (K', M') \ \wedge \ H_K(M) = H_{K'}(M') \end{array}\right]
\end{array}
$$

Fig. 3. Definitions of security notions for a hash function family H [22, 12]. In the case of eSec and eTCR notions the parameter m is assumed to be the length of the first (i.e. target) message M output by the adversary in the first stage of attack, i.e. A_1.

ϵ, in attacking H in xxx sense. Note that some of the notions (namely, Sec$[m]$, aSec$[m]$, eSec$[m]$, Pre$[m]$, aPre$[m]$ and eTCR$[m]$) are parameterized by m where $\{0,1\}^m \subseteq \mathcal{M}$. In the case of eSec (i.e. TCR) and eTCR notions the parameter m is implicit in the definitions and assumed to be the length of the first (i.e. target) message M output by the adversary. If H is a compression function (i.e. an FIL hash function), then parameter m and the resource parameter l for the adversary will be the same as the fixed input length of the compression function and hence omitted from the notations. It is shown in [22] that the strength of provisional implications between different notions depends on the relative size of m and the hash size n.

CR for a Keyless Hash Function. Collision resistance as a security property cannot be formally defined for a keyless hash function $H : \mathcal{M} \rightarrow \{0,1\}^n$. *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 [21], 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 [21]. 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 [29] and theoretically endangered assumption for SHA-1 [28].

3 Relationships between eTCR and the Seven Security Notions

In this section we provide a full picture of relationships, i.e. implicatiosns or separations, between eTCR security notion as defined in [12] as a new property for a hash function and all the seven notions of security for hash functions in [22], namely CR, Sec, aSec, eSec (a.k.a. TCR), Pre, aPre, ePre. The relationships between eTCR and three of these notions, namely eSec (TCR), Sec and Pre can be obtained by simply considering the definitions of these notions and using the results of [22] about relationships among eSec, Sec and Pre. Hence we will only briefly review the main ideas regarding these three notions. For the remaining four notions, i.e. CR, aSec, aPre and ePre, we provide a full analysis of the relationships. Briefly saying, the results are as follows:

- There is a separation between eTCR and each one of the CR, aSec, aPre and ePre notions, i.e. eTCR neither implies any of these four notions nor is implied by any of these notions.
- eTCR implies Sec, eSec (TCR) and also (provisionally) Pre, but *none* of the Sec, eSec or Pre notions implies eTCR (i.e. eTCR is a stronger security notion than these three notions).

In the sequel, we first provide our main results which are showing the separations between eTCR and the four notions, namely CR, aSec, aPre, and ePre. Among these results, the separation between eTCR and CR is of more theoretical and practical interest, as CR is one of the most in use and challenged properties for a hash function as we discussed in Sec. 1 of this paper. The following four subsections provide separation results, and then we will proceed to consider the implications.

3.1 eTCR Property *vs.* CR Property

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. In other words, we follow the comparison methodology in the dedicated-key hash function setting as in [22]. The CR property considered in this section should not be mixed with the strong keyless-CR assumption for a keyless hash function.

Theorem 1. *There is a separation between eTCR and CR notions of security.*

The proof for this separation result is obtained by combining the results of Lemma 1 and Lemma 2 as follows.

We firstly want to show that the CR property does not imply the eTCR property (CR $\nRightarrow$ eTCR). 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.*

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

Note that the condition in line (3) of definition of G (implicitly denoted as "*otherwise*") actually can be explicitly shown as: [if $M \neq M^* \bigwedge K \neq K^* \bigwedge H_K(M) \neq M^*_{1\cdots n}$]. It is easily seen that this condition and the other two conditions in line (1) and (2) cover the all possibility for K and M in defining $G_K(M)$.

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 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 is as follows (note that K is selected at random and given to the adversary at the beginning of CR game):

Algorithm $B(K)$
10: **if** $K = K^*$ then **return** "Fail"
20: $(M, M') \xleftarrow{\$} A(K)$;
30: **if** $[M = M^* \bigwedge H_K(M') = M^*_{1\cdots n}]$ **return** (M, M')
40: **if** $[M' = M^* \bigwedge H_K(M) = M^*_{1\cdots n}]$ **return** (M, M')
50: **if** $[M \neq M^* \bigwedge H_K(M) = M^*_{1\cdots n} \bigwedge M' \neq M^* \bigwedge H_K(M') \neq M^*_{1\cdots n}]$ **return** (M^*, M')
60: **if** $[M' \neq M^* \bigwedge H_K(M') = M^*_{1\cdots n} \bigwedge M \neq M^* \bigwedge H_K(M) \neq M^*_{1\cdots n}]$ **return** (M, M^*)
70: **else return** (M, M')

Let **Bad** denote the event that in line 10 of algorithm B we have $K = K^*$. As K^* is a fixed parameter and the key K is selected uniformly at random from key space $\{0,1\}^k$ and given to A, we have $Pr[\mathbf{Bad}] = 2^{-k}$. Let $\overline{\mathbf{Bad}}$ denote the complement event for **Bad**, *i.e.* $K \neq K^*$, so we have $Pr[\overline{\mathbf{Bad}}] = 1 - 2^{-k}$. We claim that unless **Bad** happens (in which case B will fail as specified in line 10 of its pseudocode), B will return a valid collision for H whenever A is successful in returning a valid collision (M, M') for G. To prove this claim first note that if **Bad** does not happen then algorithm B will return a message pair depending on which of the conditions specified in lines 30-70 of its code are satisfied. Referring to the definition of hash function family G, if A returns a valid collision (M, M') under G_K, we can analyze all possible cases that this can happen and show that in each case algorithm B also returns a collision for H_K. Let **(i)-(j) Coll** mean that the colliding messages M and M' output by A for G_K, respectively, satisfy conditions in line (i) and line (j) in definition of the function G. Then we have the following cases (*remember that we assume* $\overline{\mathbf{Bad}}$, *that is* $K \neq K^*$):

1. **(1)-(1) Coll**, **(1)-(3) Coll** and **(3)-(1) Coll** are not possible. A **(1)-(1) Coll** implies that $M = M'$ which is not possible as it is assumed that (M, M') is a valid collision for G_K. Now, note that the condition in line (3) of definition of G (implicitly denoted as "*otherwise*") actually can be explicitly shown as:

[if $M \neq M^* \bigwedge K \neq K^* \bigwedge H_K(M) \neq M^*_{1\cdots n}$]. Hence, the hash value computed on line (3) is always different from $M^*_{1\cdots n}$ and therefore **(1)-(3) Coll** and **(3)-(1) Coll** are impossible.

2. **(1)-(2) Coll**: When adversary A outputs a valid **(1)-(2) Coll** for hash function G (*i.e.* $M' \neq M \ \wedge$ $G_K(M') = G_K(M)$), referring to definition of G and remembering the assumption $\overline{\textbf{Bad}} : K \neq K^*$, it can be seen that $M = M^*$ and $H_K(M') = M^*_{1\cdots n}$ because this is a **(1)-(2) Coll** and from $G_K(M') = G_K(M)$ we have $H_K(M^*) = M^*_{1\cdots n}$. In this case, the adversary B returns (M, M') in line 30 of its code as collision for H_K and wins because $H_K(M) = H_K(M^*) = M^*_{1\cdots n} = H_K(M')$.
3. **(2)-(1) Coll**: The proof for this case is symmetric to the case of **(1)-(2) Coll** and this time adversary B returns (M, M') in line 40 of its code as collision for H_K.
4. **(2)-(3) Coll**: We show that in this case, the adversary B returns (M^*, M') as a collision for H_K in line 50 of its code and wins. It is easy to see as whenever the adversary A outputs a valid **(2)-(3) Coll** for hash function G then (by referring to the definition of G, remembering the assumption $\overline{\textbf{Bad}} : K \neq K^*$ and considering the condition for line (3) of G explicitly,) it can be seen that $M \neq M^*$, $H_K(M) = M^*_{1\cdots n}$, $M' \neq M^*$ and $H_K(M') \neq M^*_{1\cdots n}$. Hence, as (M, M') output by A is a valid collision for G, *i.e.* $G_K(M') = G_K(M)$, we have that $H_K(M') = H_K(M^*)$ and therefor (M^*, M') returned by B in line 50, is a valid collision for H_K.
5. **(3)-(2) Coll**: The proof for this case is symmetric to the case of **(2)-(3) Coll** and this time the adversary B returns (M, M^*) in line 60 of its code as collision for H_K.
6. **(2)-(2) Coll** and **(3)-(3) Coll**: It can be seen that in these two cases the adversary B returns (M, M') as a collision for H_K in line 70 of its code. Referring to the definition of function G, it is seen that whenever the adversary A outputs a valid collision (M, M') for G_K as either a **(2)-(2) Coll** or **(3)-(3) Coll** (that is, $M \neq M' \wedge G_K(M) = G_K(M')$ and both M and M' belong to the same sub-domain of G) then (M, M') will also be a valid collision for H_K. Note that $G_K(M) = G_K(M')$ implies that in **(2)-(2) Coll** case we have $H_K(M) = H_K(M') = H_K(M^*)$ and in **(3)-(3) Coll** case we have $H_K(M) = H_K(M')$.

The above case analysis shows that when **Bad** does not happen (*i.e.* when $K \neq K^*$) then the adversary B will be successful in finding a valid collision for H_K if the adversary A can find a valid collision for G_K. If **Bad** happens then B will fail and return "Fail" in line 10 of its code. Therefore, we have $\epsilon = \Pr[B \text{ succeeds}] = \Pr[A \text{ succeeds} \wedge \overline{\textbf{Bad}}] \geq \Pr[A \text{ succeeds}] - \Pr[\textbf{Bad}] = \epsilon' - 2^{-k}$. □

We now want to demonstrate that the eTCR property does not imply the CR property (eTCR $\nRightarrow$ CR). 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* $H : \{0,1\}^k \times \{0,1\}^m \rightarrow \{0,1\}^n$, *where* $m > k \geq n$, *which is* $(t, \epsilon) - eTCR$. *The dedicated-key hash function* $G : \{0,1\}^k \times \{0,1\}^m \rightarrow \{0,1\}^n$ *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.)*

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

Note that the structural assumption about $H : \{0,1\}^k \times \{0,1\}^m \rightarrow \{0,1\}^n$, namely that we have $m > k \geq n$ is quite reasonable even for practical scenarios. For instance, in Randomized Hashing which should provide a dedicated-key hash function with eTCR property, the key length k is fixed and equal to the block length of the underlying keyless hash function (e.g using SHA-1 we have $k = 512$, $n = 160$) while message length m can be very large (just less than 2^{64}).

Proof. We firstly demonstrate that G as a dedicated-key hash function is not secure in CR sense. This can be 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 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** $B_2(K, M, State)$
10: $(M, State) \stackrel{\$}{\leftarrow} A_1()$;	30: if $\left[M = 1^{m-k}\vert\vert K \bigvee M = 0^{m-k}\vert\vert K\right]$ **return 'Fail'**;
20: **return** $(M, State)$;	40: $(M', K') \stackrel{\$}{\leftarrow} A_2(K, M, State)$;
	50: if $M' = 1^{m-k}\vert\vert K$ then **return** $(0^{m-k}\vert\vert K, K')$;
	60: else **return** (M', K');

As it can be seen from B's description, in the first stage of eTCR attack B_1 just merely runs A_1 and returns whatever it returns as the first message(M) and any possible state information to be passed to the second stage algorithm. In the second stage of the attack, let **Bad** be the event that $[M = 1^{m-k}||K \bigvee M = 0^{m-k}||K]$. It can be observed that 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 H assuming that A is successful in eTCR attack against G.

Note that an adversary A against G is successful in eTCR attack whenever $(M, K) \neq (M', K')$ and $G_K(M) = G_{K'}(M')$. Assuming that the event $\overline{\textbf{Bad}}$ happens; that is, $[M \neq 1^{m-k}||K \bigwedge M \neq 0^{m-k}||K]$ and referring to the description of function G in this lemma, it can be shown that if A succeeds then B also succeeds as follows:

1. **Case 1:** $M' = 1^{m-k}||K$**.** In this case from the success condition for A we have $G_K(M) = G_{K'}(1^{m-k}||K)$ and according to the description of G this is translated to $H_K(M) = H_{K'}(0^{m-k}||K)$. Now it can be shown that B becomes successful, by returning $(0^{m-k}||K, K')$ in (line 50 of its code in) the second stage, as follows. We note that the event $\overline{\textbf{Bad}}$ implies that $M \neq 0^{m-k}||K$ and hence $(M, K) \neq (0^{m-k}||K, K')$. So, the pairs (M, K) and $(0^{m-k}||K, K')$ output by B is a valid colliding pair for H according to winning condition in eTCR game.
2. **Case 2:** $M' \neq 1^{m-k}||K$**.** In this case (which is the complement of Case 1), B succeeds by just returning (M', K') in (line 60 of its code in) the second stage, *i.e.* the same message and key pair as A returns in its second stage. This is easy to verify as in this case from the description of G we have $G_K(M) = H_K(M)$ and $G_{K'}(M') = H_{K'}(M')$, and so B wins against H if A wins against G.

Now note that $\Pr[\textbf{Bad}] = \Pr[M = 1^{m-k}||K] + \Pr[M = 0^{m-k}||K] = 2^{-k} + 2^{-k} = 2^{-k+1}$, as K is selected uniformly at random just after the message M is fixed in the eTCR game. Hence, we have $\epsilon = \Pr[B \text{ succeeds}] = \Pr[A \text{ succeeds} \wedge \overline{\textbf{Bad}}] \geq \Pr[A \text{ succeeds}] - \Pr[\textbf{Bad}] = \epsilon' - 2^{-k+1}$.

□

The Case for Randomized Hashing. Randomized Hashing method as shown in Fig. 1 is a simple method to obtain a dedicated-key hash function $\tilde{H} : \mathcal{K} \times M \to \{0,1\}^n$ 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 $h : \{0,1\}^{n+b} \to \{0,1\}^n$ and using a *fixed* initial chaining value IV. The analysis in [12] 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 [29, 28], 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.

3.2 Relationships among eTCR, aSec and aPre

In this section we compare eTCR property with aSec and aPre properties. In both cases, we show a separation. That is, we show that eTCR neither implies any of these two notions nor is implied by any of these two notions, when considering an arbitrary dedicated-key hash function. The results are stated in the following two theorems.

Theorem 2. *There is a separation between eTCR and aSec notions of security.*

Theorem 3. *There is a separation between eTCR and aPre notions of security.*

The proofs for these two theorems (separation results) are obtained by combining the results of the following three lemmas. In Lemma 3 we show that eTCR does not imply either of aSec or aPre properties. Lemma 4 and 5 are easily deduced from the previously known relations in [22, 12], and respectively show that aSec does not imply eTCR, and aPre does not imply eTCR. Hence, the proof of Theorem 2 (separation between eTCR and aSec) is obtained by combining Lemma 3 (eTCR $\nRightarrow$ aSec) and Lemma 4 (aSec $\nRightarrow$ eTCR), and that of Theorem 3 (separation between eTCR and aPre) is obtained from Lemma 3 (eTCR $\nRightarrow$ aPre) and Lemma 5 (aPre $\nRightarrow$ eTCR).

Lemma 3 (eTCR property does not imply either of aSec or aPre properties). *Assume that there exists a dedicated-key hash function $H : \{0,1\}^k \times \{0,1\}^m \to \{0,1\}^n$ which is $(t, \epsilon) - eTCR$. Select (and fix) an arbitrary key $K^* \in \{0,1\}^k$ and an arbitrary hash value $C^* \in \{0,1\}^n$ (e.g. $K^* = 0^k$ and $C^* = 0^n$). 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 $t' = t - T_H - c$ and $\epsilon' = \epsilon + \sqrt{\epsilon} + 2^{-k+1}$, but it is completely insecure in both aSec and aPre senses. T_H denotes the time for one computation of H and c is a small constant.*

$$G_K(M) = \begin{cases} C^* & \textit{if } K = K^* \quad (1) \\ \\ H_K(M) & \textit{otherwise} \quad (2) \end{cases}$$

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

Proof. Let's first demonstrate that G as a dedicated-key hash function is not secure in either of aSec or aPre senses.

– The case of aSec: Consider the following simple adversary $A = (A_1, A_2)$ playing aSec game against G. In the first stage of aSec attack, A_1 chooses the key as $K = K^*$. In the second stage of the attack, A_2, after receiving the first randomly selected message M (where $M \xleftarrow{\$} \{0,1\}^m$), outputs any different message $M' \neq M$. It can be seen easily that the adversary $A = (A_1, A_2)$ wins the aSec game, as $M' \neq M$ and by the construction of G we have $G_{K^*}(M') = G_{K^*}(M) = C^*$. Therefore, the hash function family G is completely insecure in aSec sense.
– The case of aPre: Consider the following simple attack. In the first stage of aPre attack, A_1 chooses the key as $K = K^*$. In the second stage of the attack, A_2, after receiving the hash value $Y = G_{K^*}(M) = C^*$, outputs any arbitrary message $M' \in \{0,1\}^m$. It can be seen easily that the adversary $A = (A_1, A_2)$ always wins the aPre game as, according to the construction of the hash family G, we have $G_{K^*}(M') = C^*$ for any $M^* \in \{0,1\}^m$. Therefore, the hash function family G is completely insecure in aPre sense.

To complete the proof of the lemma, we need to show that the hash function family G inherits the eTCR property of H. Note that in construction of hash function G from H it is assumed that H is (t, ϵ)-eTCR, i.e. the success probability of any adversary having time complexity t in attacking eTCR property of H is upperbounded by ϵ. Let $A = (A_1, A_2)$ be any adversary that can win eTCR game against G with success probability ϵ' and having time complexity at most t'. We want to show that $\epsilon' \leq \epsilon + \sqrt{\epsilon} + 2^{-k+1}$ as stated in the lemma. Consider the following adversary $B = (B_1, B_2)$ against eTCR property of H which uses A as a subroutine:

Algorithm $B_1()$	Algorithm $B_2(K, M, State)$
10: $(M, State) \xleftarrow{\$} A_1()$;	30: if $[K = K^* \bigvee H_K(M) = C^*]$ **return 'Fail'**;
20: **return** $(M, State)$;	40: $(M', K') \xleftarrow{\$} A_2(K, M, State)$;
	50: **return** (M', K');

As it can be seen from the description of algorithm B, in the first stage of eTCR attack B_1 merely runs A_1 and returns whatever it returns as the first message(M) and any possible state information to be passed to the second stage algorithm. In the second stage of the attack, let **Bad** be the event that $[K = K^* \bigvee H_K(M) = C^*]$. 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 H assuming that A is successful in eTCR attack against G.

Note that adversary A succeeds in eTCR attack against G whenever $(M, K) \neq (M', K')$ and $G_K(M) = G_{K'}(M')$. Assuming that the event $\overline{\textbf{Bad}}$ happens; that is, $[K \neq K^* \bigwedge H_K(M) \neq C^*]$ and referring to the construction of the (counterexample) hash function G, it can be observed that in this case $G_K(M) = G_{K'}(M')$ will imply $H_K(M) = H_{K'}(M')$, that is adversary B also succeeds in eTCR attack against H. Hence we have: $\epsilon \geq \Pr[B \text{ succeeds}] = \Pr[A \text{ succeeds} \wedge \overline{\textbf{Bad}}] \geq \Pr[A \text{ succeeds}] - \Pr[\textbf{Bad}] = \epsilon' - \Pr[\textbf{Bad}]$. Rearranging the terms we have:

$$\epsilon' \leq \epsilon + \Pr[\textbf{Bad}] \tag{1}$$

Now we need to upperbound $\Pr[\textbf{Bad}] = \Pr[K = K^* \bigvee H_K(M) = C^*]$. Using the union bound we have:

$$\Pr[\mathbf{Bad}] \leq \Pr[K = K^*] + \Pr[H_K(M) = C^*] = 2^{-k} + \Pr[H_K(M) = C^*] \tag{2}$$

It remains to upperbound $P = \Pr[H_K(M) = C^*]$. We claim that:

Claim. $P = \Pr[H_K(M) = C^*] \leq 2^{-k} + \sqrt{\epsilon}$.

Before continuing to prove this claim, note that the inequalities (1), (2) and the above claim complete the proof of the lemma, i.e. we get the target upperbound as $\epsilon' \leq \epsilon + \sqrt{\epsilon} + 2^{-k+1}$. Clearly, the time complexity of B (denote by t) is that of A (denote by t') plus the the time for one computation of H and a small constant time c, i.e. $t = t' + T_H + c$.

Now let's proceed to prove the aforementioned *claim.*

Proof (Claim). We use a two step method. The first and main step is to express our problem in a format which can be considered as an special case of Reset Lemma of [4] and then we can apply the probabilistic analysis of the Reset Lemma.
Note that according to the description of adversary $B = (B_1, B_2)$, this probability is taken over the random coins used by algorithm $A = (A_1, A_2)$ and the random selection of the first key K. Referring to the description of $B = (B_1, B_2)$ it can be seen that P equals to the probability that the following experiment returns 1:

Experiment I
$(M, State) \xleftarrow{\$} A()$;
$K \xleftarrow{\$} \{0,1\}^k$
If $H_K(M) = C^*$ then **return 1** else **return 0**;

Let $R \in \{0,1\}^r$ denote the random tape (i.e. coins) used by the (randomized) algorithm $A = (A_1, A_2)$. Let $\text{Verify}(M, K)$ be a predicate which is defined as follows:

$$\text{Verify}(M, K) = \begin{cases} 1 & \text{if } H_K(M) = C^* \\ 0 & \text{otherwise} \end{cases}$$

Now, we can rewrite Experiment I (using our notations in subsection 2.1) as below, where $\varnothing$ means 'no input':

Experiment I
$R \xleftarrow{\$} \{0,1\}^r$; $(M, State) = A(\varnothing; R)$;
$K \xleftarrow{\$} \{0,1\}^k$; $d = \text{Verify}(M, K)$;
Return d

Let Q be the probability that the following (reset) experiment returns 1:

Experiment II (*Reset Experiment*)
$R \xleftarrow{\$} \{0,1\}^r$; $(M, State) = A(\varnothing; R)$;
$K_1 \xleftarrow{\$} \{0,1\}^k$; $d_1 = \text{Verify}(M, K_1)$;
$K_2 \xleftarrow{\$} \{0,1\}^k$; $d_2 = \text{Verify}(M, K_2)$;
If $(d_1 = 1 \bigwedge d_2 = 1 \bigwedge K_1 \neq K_2)$ then **return 1** else **return 0**

Proposition 1. *Probability that Experiment I returns 1 (denoted by P) is upperbounded by the square root of the probability that Experiment B returns 1 plus inverse of the size of key space, i.e.* $P \leq \sqrt{Q} + 2^{-k}$.

The proof of this proposition can be deduced as a special case of that of Reset Lemma in [4]. We provide the proof here for completeness. For any $R \in \{0,1\}^r$, let M_R denote the target message output by A on random tape (coins) specified by R, that is, $(M_R, State_R) = A(\varnothing; R)$. Define two functions $X : \{0,1\}^r \rightarrow [0,1]$ and $Y : \{0,1\}^r \rightarrow [0,1]$ as follows:

$$X(R) \triangleq \Pr[\text{Verify}(M_R, K) = 1] \tag{3}$$

where the probability is taken over random selection of K from the key space $\{0,1\}^k$, and

$$Y(R) \triangleq \Pr[\text{Verify}(M_R, K_1) = 1 \bigwedge \text{Verify}(M_R, K_2) = 1 \bigwedge K_1 \neq K_2] \tag{4}$$

where the probability is taken over random and *independent* selection of K_1 and K_2 from the key space $\{0,1\}^k$. By a simple argument, noting that K_1 and K_2 are chosen independently and using the fact that $\Pr(E \bigwedge \overline{F}) \geq \Pr(E) - \Pr(F)$ for any two events E and F, we have:

$$Y(R) = \Pr[\text{Verify}(M_R, K_1) = 1] \,.\, \Pr[\text{Verify}(M_R, K_2) = 1 \bigwedge K_1 \neq K_2] \geq X(R)[X(R) - 2^{-k}] \tag{5}$$

We can view functions X and Y as random variables over sample space $\{0,1\}^r$ of random tape (coins) used by probabilistic algorithm A. Now, note that the probabilities that Experiment I and Experiment II return 1 are, respectively, the expected values of the random variables X and Y with respect to R, i.e. $P = \mathbf{E}[X]$ and $Q = \mathbf{E}[Y]$. Using the inequality (5) and letting $c = 2^{-k}$ we have:

$$Q = \mathbf{E}[Y] \geq \mathbf{E}[X(X-c)] = \mathbf{E}[X^2] - c\mathbf{E}[X] \geq \mathbf{E}[X]^2 - c\mathbf{E}[X] = P^2 - cP$$

Using the above relation we have:

$$(P - \frac{c}{2})^2 = P^2 - cP + \frac{c^2}{4} \leq Q + \frac{c^2}{4}$$

and using the fact that $\sqrt{a+b} \leq \sqrt{a} + \sqrt{b}$ for $a, b \geq 0$ we have:

$$P - \frac{c}{2} \leq \sqrt{Q} + \frac{c}{2}$$

that is, (remembering $c = 2^{-k}$) we get the final result as $P \leq \sqrt{Q} + 2^{-k}$.

Proposition 2. *Probability that Experiment II returns 1 is upperbounded by the success probability of algorithm A in eTCR attack against H, i.e. we have $Q \leq \epsilon$.*

The proof is obtained by considering the condition that Experiment B returns 1, noting the definition of eTCR attack game in Fig. 3 and definition of predicate Verify(.,.). Note that Experiment II returns 1 if $(\text{Verify}(M, K_1) = 1 \bigwedge \text{Verify}(M, K_2) = 1 \bigwedge K_1 \neq K_2)$, and from the definition of Verify(.,.) this means that $(H(K_1, M) = H(K_2, M) = C^* \bigwedge K_1 \neq K_2)$. Hence whenever Experiment II returns 1 the pair $(K_1, M) \neq (K_2, M)$ and $H(K_1, M) = H(K_2, M)$, i.e. A succeeds in eTCR attack against H.

□

Lemma 4. *aSec property does not imply eTCR property.*

Proof. The proof is easily deduced from the following known relations:

- eTCR implies TCR (eTCR $\Rightarrow$ TCR) [12]. (In other words, if a hash function is not TCR secure then it cannot be eTCR secure either.)
- aSec does not imply TCR (aSec $\nRightarrow$ TCR), i.e. it is possible to construct (counterexample) dedicated-key hash functions which are secure in aSec sense but completely insecure in TCR sense [22].

From the above two facts it is deduced that aSec does not imply eTCR (aSec $\nRightarrow$ eTCR) either. $\square$

Lemma 5. *aPre property does not imply eTCR property.*

Proof. The proof is easily obtained from the following facts:

- eTCR implies TCR (eTCR $\Rightarrow$ TCR) [12].
- aPre does not imply TCR (aPre $\nRightarrow$ TCR) [22].

From the above two facts it is deduced that aPre does not imply eTCR (aPre $\nRightarrow$ eTCR) either. $\square$

3.3 eTCR Property *vs.* ePre Property

In this section we compare eTCR property with ePre property and show that there is a separation between these two notions of security, that is, in general for an arbitrary dedicated-key hash function eTCR property neither implies ePre property nor is implied by ePre property. The result is stated in the following theorem:

Theorem 4. *There is a separation between eTCR and ePre notions of security.*

The proof is obtained by combining Lemma 6 and Lemma 7 as follows.

Lemma 6 (eTCR property does not imply ePre property). *Assume that there exists a dedicated-key hash function $H : \{0,1\}^k \times \{0,1\}^m \rightarrow \{0,1\}^n$, where $m \geq k$, which is $(t,\epsilon) - eTCR$. Select (and fix) an arbitrary hash value $C^* \in \{0,1\}^n$ (e.g. $C^* = 0^n$). The dedicated-key hash function $G : \{0,1\}^k \times \{0,1\}^m \rightarrow \{0,1\}^n$ shown in this lemma is $(t',\epsilon') - eTCR$, where $t' = t - T_H - c$ and $\epsilon' = \epsilon + \sqrt{\epsilon} + 2^{-k+1}$, but it is completely insecure in ePre sense. T_H denotes the time for one computation of H, c is a small constant and $val(.)$ returns the decimal value of its binary string input.*

$$G_K(M) = \begin{cases} C^* & \text{if } val(K) = val(M) \\ \\ H_K(M) & \text{otherwise} \end{cases}$$

Proof. Let's first show that G as a dedicated-key hash function is not secure in ePre sense. Consider the following adversary. In the first stage of ePre attack, A_1 chooses the target hash value as $Y = C^*$. In the second stage of the attack, A_2, after receiving the random key K, outputs a message $M \in \{0,1\}^m$ whose decimal value equals to that of K, i.e. $M = \langle val(K) \rangle_m$. Clearly, the adversary $A = (A_1, A_2)$ always wins the ePre game as, according to the construction of the hash family G, we have $G_K(M) = C^* = Y$ for such a message M and K (where $val(K) = val(M)$). Therefore, the hash function family G is completely insecure in ePre sense. $\square$

To complete the proof of the Lemma 6, we need to show that hash function G indeed is an eTCR secure function as stated in the lemma. The proof for this part is similar to that of Lemma 3 and is briefly provided below for completeness.

Let $A = (A_1, A_2)$ be any adversary that can win eTCR game against G with success probability ϵ' and having time complexity at most t'. We show that $\epsilon' \leq \epsilon + \sqrt{\epsilon} + 2^{-k+1}$ as stated in the lemma. Consider the following adversary $B = (B_1, B_2)$ against eTCR property of H which uses A as its subroutine:

Algorithm $B_1()$	**Algorithm** $B_2(K, M, State)$
$(M, State) \xleftarrow{\$} A_1()$;	if $[val(K) = val(M) \bigvee H_K(M) = C^*]$ **return 'Fail'**;
return $(M, State)$;	$(M', K') \xleftarrow{\$} A_2(K, M, State)$;
	return (M', K');

As it can be seen from the description of algorithm B, *unless* a special event **Bad** happens B runs A and returns whatever it returns. Here, **Bad** is the event that $[val(K) = val(M) \bigvee H_K(M) = C^*]$ in the second stage of eTCR attack (i.e. B_2). If **Bad** happens then algorithm B will fail in its second phase; otherwise (*i.e.* if $\overline{\mathbf{Bad}}$ happens) we show that B will be successful in eTCR attack against H whenever A is successful in eTCR attack against G.

Note that adversary A succeeds in eTCR attack against G whenever $(M, K) \neq (M', K')$ and $G_K(M) = G_{K'}(M')$. Assuming that the event $\overline{\mathbf{Bad}}$ happens; that is, $[val(K) \neq val(M) \bigwedge H_K(M) \neq C^*]$ and referring to the construction of G, it can be seen that in this case $G_K(M) = G_{K'}(M')$ will imply $H_K(M) = H_{K'}(M')$, that is B also succeeds in eTCR attack against H. Hence we have: $\epsilon \geq \Pr[B \text{ succeeds}] = \Pr[A \text{ succeeds} \wedge \overline{\mathbf{Bad}}] \geq \Pr[A \text{ succeeds}] - \Pr[\mathbf{Bad}] = \epsilon' - \Pr[\mathbf{Bad}]$. Rearranging the terms we have:

$$\epsilon' \leq \epsilon + \Pr[\mathbf{Bad}] \tag{6}$$

Now we need to upperbound $\Pr[\mathbf{Bad}] = \Pr[val(K) = val(M) \bigvee H_K(M) = C^*]$. Using the union bound we have:

$$\Pr[\mathbf{Bad}] \leq \Pr[val(K) = val(M)] + \Pr[H_K(M) = C^*] \leq 2^{-k} + \Pr[H_K(M) = C^*] \tag{7}$$

It remains to upperbound $P = \Pr[H_K(M) = C^*]$. We claim that $P = \Pr[H_K(M) = C^*] \leq 2^{-k} + \sqrt{\epsilon}$. The proof for this claim is the same as provided in proof of Lemma 3. Now, from the inequalities (6), (7) and the above claim complete the proof of the lemma, i.e. we get the target upperbound as $\epsilon' \leq \epsilon + \sqrt{\epsilon} + 2^{-k+1}$. Clearly, the time complexity of B (denote by t) is that of A (denote by t') plus the the time for one computation of H and a small constant time c, i.e. $t = t' + T_H + c$.

Lemma 7. *ePre property does not imply eTCR property.*

Proof. The proof is easily obtained from the following facts:

- eTCR implies TCR (eTCR $\Rightarrow$ TCR) [12].
- ePre does not imply TCR (ePre $\not\Rightarrow$ TCR) [22].

From the above two facts it is deduced that ePre does not imply eTCR (ePre $\not\Rightarrow$ eTCR) either. □

3.4 Relationships among eTCR, Sec, eSec, and Pre Properties

The relationships between eTCR and the remaining three notions of security, namely Sec, eSec(TCR) and Pre are straightforwardly extractable from the previously known relations between eTCR and TCR in [12] and between TCR, Sec and Pre in [22].

eTCR Property *vs.* Sec Property

- eTCR implies TCR (eTCR $\Rightarrow$ TCR) [12].
- TCR implies Sec (TCR $\Rightarrow$ Sec) [22].
- Sec does not imply TCR (Sec $\not\Rightarrow$ TCR) [22].

From the above three facts it is deduced that eTCR is strictly stronger security notion than Sec, that is (eTCR $\Rightarrow$ Sec) but (Sec $\not\Rightarrow$ eTCR).

eTCR Property *vs.* eSec (TCR) Property

- eTCR implies TCR (eTCR $\Rightarrow$ TCR) [12].
- Claim: TCR does not imply eTCR (TCR $\nRightarrow$ eTCR).

Hence, eTCR is strictly stronger security notion than TCR (= eSec).

Now we need to prove the claim that TCR $\nRightarrow$ eTCR. This can be shown using the following counterexample borrowed from [22]:

$$H6_K(M) = \begin{cases} 0^n & \text{if } M = 0^m \\ H_K(M) & \text{if } M \neq 0^m \bigwedge H_K(M) \neq 0^n \\ H_K(0^m) & \text{otherwise} \end{cases}$$

It is proved in [22] that $H6$ inherits eSec (TCR) property from H, so it remains to show that $H6$ is insecure in eTCR sense. The following simple adversary $A = (A_1, A_2)$ can break $H6$ in eTCR sense. A_1 outputs $M = 0^m$ as the target message. A_2, on receiving a random key K, simply outputs $M' = M = 0^m$ but with a different second key $K' \neq K$. Note that $(M, K) \neq (M', K')$ and $H6_K(M) = H6_K(M') = 0^n$, that is A always win in eTCR attack against $H6$.
Therefore, assuming that there is a TCR secure hash function H (otherwise the whole discussion become moot), it is possible to construct a hash function $H6$ which is TCR secure but completely insecure in eTCR sense, i.e. TCR $\nRightarrow$ eTCR.

eTCR Property *vs.* Pre Property

- eTCR implies TCR (eTCR $\Rightarrow$ TCR) [12].
- TCR (provisionally) implies Pre (TCR $\Rightarrow$ Pre) [22].
- Pre does not imply eTCR (Pre $\nRightarrow$ TCR) [22].

Hence, eTCR (provisionally) implies Pre but Pre does not imply eTCR. By 'provisional' implication it is meant that the strength of the implication depends on the amount of compression by the hash function. If the hash function is highly compressing, i.e. $m >> n$ then we have a strong implication but when $m \approx n$ then the implication essentially vanishes. For more details we refer to [22].

4 Domain Extension and eTCR Property Preservation

In this section we investigate the eTCR preserving capability of eight domain extension transforms, namely Plain MD [14, 8], Strengthened MD [14, 8], Prefix-free MD [7, 13], Randomized Hashing [12], Shoup [24], Enveloped Shoup [2], XOR Linear Hash (XLH)[5], and Linear Hash (LH) [5] methods.

Assume that we have a compression function $h : \{0,1\}^k \times \{0,1\}^{n+b} \to \{0,1\}^n$ 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 $H : \mathcal{K} \times \mathcal{M} \to \{0,1\}^n$, 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 $Pad : \mathcal{M} \to D_I$ 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 $f_I : \mathcal{K} \times D_I \to \{0,1\}^n$ 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. It is known that the property preserving capability of a domain extension transform depends on both the padding function and iteration function, for example 'Plain MD' (*i.e.*, plain padding and MD iteration) is not CR preserving domain extender, but 'Strengthened MD' (*i.e.*, strengthening padding and MD iteration) does preserve CR [14, 8, 2]. Hence, precisely speaking, we can have several domain extenders using the same iteration function but with different padding function, e.g. Plain MD, Strengthened MD, Prefix-free MD, which are considered as three different domain extenders that have different capabilities from property preserving viewpoint [2].

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. 4.

4.1 Merkle-Damgård Does not Preserve eTCR

MD iteration function as shown in Fig. 4 can be used together with Plain (pad), Strengthening(pad_s), or Prefix-free($padPF$) padding function to construct 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 5 (Negative Result). *Plain MD, Strengthened MD, and Prefix-free MD do not preserve eTCR.*

Proof. We borrow the construction of the following counterexample from [5] where it was used in the context of TCR property. Assume that there is a dedicated-key compression function $g : \{0,1\}^k \times \{0,1\}^{n+b} \rightarrow \{0,1\}^n$ 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 $h : \{0,1\}^k \times \{0,1\}^{(n+k)+b'} \rightarrow \{0,1\}^{n+k}$:

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

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 8 that h_K inherits the eTCR property from g_K. Note that this cannot be directly inferred from the proof in [5] 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.

$MD^h_{IV}: \mathcal{K} \times \{0,1\}^{Lb} \rightarrow \{0,1\}^n$, where $\mathcal{K} = \{0,1\}^k$

Algorithm $MD^h_{IV}(K, M)$**:**
$C_0 = IV$
for $i = 1$ to L do
  $C_i = h_K(C_{i-1}||M_i)$
return C_L

$RH^h_{IV}: \mathcal{K} \times \{0,1\}^{Lb} \rightarrow \{0,1\}^n$, where $\mathcal{K} = \{0,1\}^{k+b}$

Algorithm $RH^h_{IV}(K||K', M)$**:**
$C_0 = IV$
$C_1 = h_K(C_0||K')$
for $i = 2$ to $L+1$ do
  $C_i = h_K(C_{i-1}||(M_{i-1} \oplus K'))$
return C_{L+1}

$Sh^h_{IV}: \mathcal{K} \times \{0,1\}^{Lb} \rightarrow \{0,1\}^n$, where $\mathcal{K} = \{0,1\}^{k+tn}$
$t = \lceil log_2(L) \rceil, \nu(i) = max\{x : 2^x|i\}$

Algorithm $Sh^h_{IV}(K||K_0||K_1||\cdots||K_{t-1}, M)$**:**
$C_0 = IV$
for $i = 1$ to L do
  $C_i = h_K((C_{i-1} \oplus K_{\nu(i)})||M_i)$
return C_L

$ESh^h_{IV_1,IV_2}: \mathcal{K} \times \{0,1\}^{(L-1)b+b-n} \rightarrow \{0,1\}^n$, where $\mathcal{K} = \{0,1\}^{k+tn}$
$t = \lceil log_2(L-1) \rceil + 1, \nu(i) = max\{x : 2^x|i\}$

Algorithm $ESh^h_{IV_1,IV_2}(K||K_0||K_1||\cdots||K_{t-1}, M)$**:**
$C_0 = IV_1;\ K_\mu = K_{t-1}$
for $i = 1$ to $L-1$ do
  $C_i = h_K((C_{i-1} \oplus K_{\nu(i)})||M_i)$

return $h_K((IV_2 \oplus K_0)||(C_{L-1} \oplus K_\mu)||M_L)$

$XLH^h_{IV}: \mathcal{K} \times \{0,1\}^{Lb} \rightarrow \{0,1\}^n$, where $\mathcal{K} = \{0,1\}^{k+Ln}$

Algorithm $XLH^h_{IV}(K||K_0||K_1||\cdots||K_{L-1}, M)$**:**
$C_0 = IV$
for $i = 1$ to L do
  $C_i = h_K((C_{i-1} \oplus K_{i-1})||M_i)$
return C_L

$LH^h_{IV}: \mathcal{K} \times \{0,1\}^{Lb} \rightarrow \{0,1\}^n$, where $\mathcal{K} = \{0,1\}^{Lk}$

Algorithm $LH^h_{IV}(K_1||K_2||\cdots||K_L, M)$**:**
$C_0 = IV$
for $i = 1$ to L do
  $C_i = h_{K_i}(C_{i-1}||M_i)$
return C_L

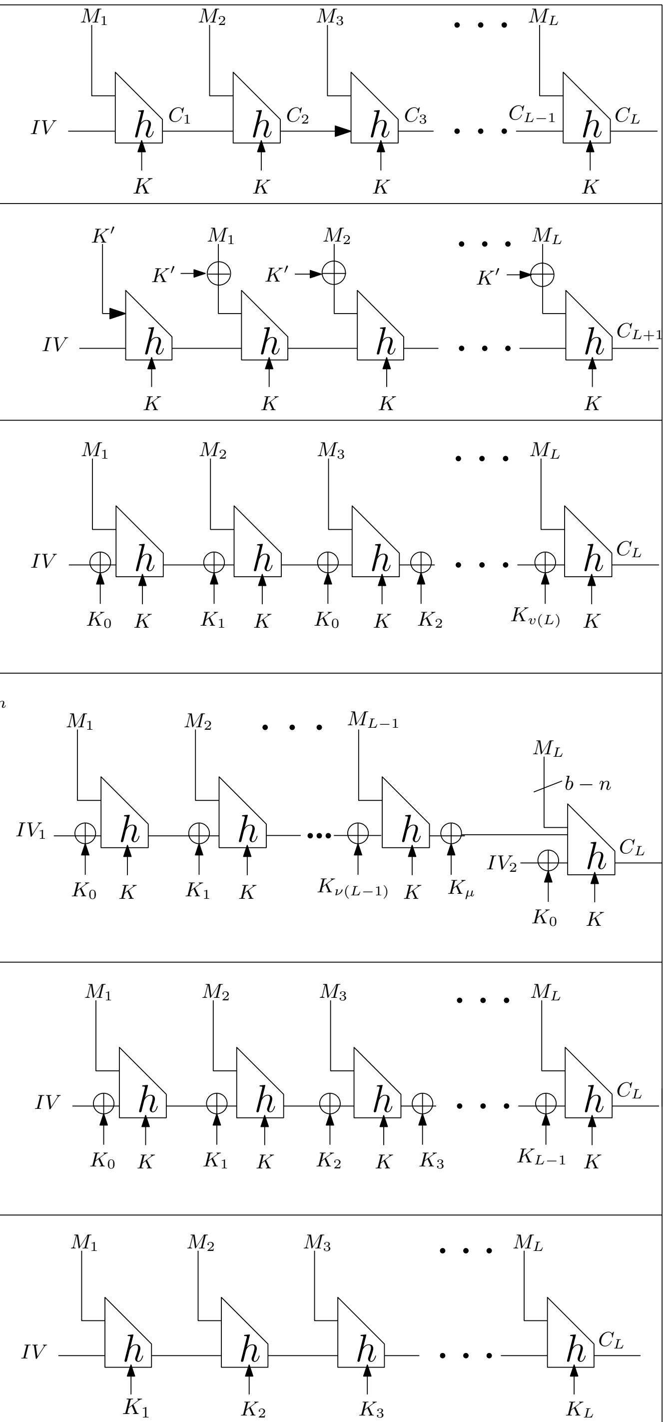

Fig. 4. Iteration functions used in domain extension transforms: Merkle-Damgård (MD), Randomized Hashing (RH), Shoup (Sh), Enveloped Shoup (ESh), XLH and LH. The iteration functions are ordered top-down based on their efficiency in terms of key expansion, MD iteration does not expand the key length of underlying compression function and is the most efficient transform and LH is the least efficient transform.

Lemma 8. *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 $B_1()$
$(M_1 = X_1||Y_1||Z_1, State) \xleftarrow{\$} A_1()$;
return $(M_1, State)$;

Algorithm $B_2(K_1, M_1, State)$
Parse M_1 as $M_1 = X_1||Y_1||Z_1$
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 $\Pr[\textbf{Bad}] \leq \Pr[K_1 = Y_1] + \Pr[K_1 = 1^k] = 2^{-k} + 2^{-k} = 2^{-k+1}$, as K_1 is selected uniformly at random just after the message M_1 is fixed in the eTCR game. Therefore, we have $\epsilon = \Pr[B \text{ succeeds}] = \Pr[A \text{ succeeds} \wedge \overline{\textbf{Bad}}] \geq \Pr[A \text{ succeeds}] - \Pr[\textbf{Bad}] \geq \epsilon' - 2^{-k+1}$.

To complete the proof of Theorem 5, 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 [5, 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 Plain MD and Strengthened MD domain extension transforms applied on the counterexample compression function h and using an initial value IV, respectively, by $\textbf{pMD}^h_{IV}$ and $\textbf{sMD}^h_{IV}$. Note that MD^h_{IV} is used to denote the MD *iteration* function (Fig. 4). Then the full-fledged hash function $H : \{0,1\}^k \times \mathcal{M} \to \{0,1\}^{n+k}$ will be defined as $H(K, M) = \textbf{pMD}^h_{IV}(K, M) = MD^h_{IV}(K, pad(M))$ and $H(K, M) = \textbf{sMD}^h_{IV}(K, M) = MD^h_{IV}(K, pad_s(M))$, for Plain and Strengthened MD case, respectively.

The following adversary $A = (A_1, A_2)$ can break H in eTCR sense for both Plain MD and Strengthened MD cases. 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:

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

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

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

The Case of Prefix-free MD: Denote Prefix-free MD domain extension transform by **preMD**. The full-fledged hash function $H : \{0,1\}^k \times \mathcal{M} \to \{0,1\}^{n+k}$ 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:

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

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

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. 4 expands the key length of the underlying compression function by only a constant additive factor of b bits, that is $log_2(|\mathcal{K}|) = k + b$ which is independent from input message length. That is, after MD transfrom, RH is the most efficient method from key expansion point of view. The latter 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 [12], we assume that the padding function is the strengthening padding pad_s and so we use the same name for domain extension as its iteration function, *i.e.* RH^h_{IV} (Fig. 4). The full-fledged hash function $H : \{0,1\}^k \times \mathcal{M} \to \{0,1\}^{n+k}$ 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 6 (Negative Result). *The Randomized Hashing transform does not preserve eTCR.*

Proof. We need to show as a counterexample, a dedicated-key compression function h which is eTCR but for which the dedicated-key hash function H obtained via Randomized Hashing method is completely insecure in eTCR sense. The same counterexample used in Theorem 5 can also be used to show that Randomized Hashing transform (in dedicated-key hash function setting) does not preserve eTCR property.

As we have previously shown in Lemma 8 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. In Sh and ESh transforms the key expansion depends logarithmically on the input message length. For Sh iteration $log_2(|\mathcal{K}|) = k + \lceil log_2(L)\rceil\, n$ and for ESh iteration $log_2(|\mathcal{K}|) = k + (\lceil log_2(L-1)\rceil + 1)n$, where L is the length of *the padded message* in blocks which is input to the iteration function. (Note that Fig. 4 just shows the iteration function of the domain extensions and padding functions are not shown Fig. 4).

We assume the same padding functions as proposed in the original schemes, that is, for Shoup [24] and XLH [5] the strengthening padding function (pad_s) is used, and for Enveloped Shoup [2] the padding function is the strengthened chain shift padding ($padCS_s$). So, the full-fledged hash function $H : \{0,1\}^k \times \mathcal{M} \to \{0,1\}^{n+k}$, obtained via these three domain extension methods, will be defined accordingly as follows:

$$\begin{aligned}
&\text{Sh: } H(K||K_0||\cdots||K_t, M) = Sh^h_{IV}(K||K_0||\cdots||K_{t-1}, pad_s(M)) && ;\text{ where } t = \lceil log_2(L)\rceil\\
&\text{ESh: } H(K||K_0||\cdots||K_t, M) = ESh^h_{IV}(K||K_0||\cdots||K_{t-1}, padCS_s(M)) && ;\text{ where } t = \lceil log_2(L-1)\rceil + 1\\
&\text{XLH: } H(K||K_0||\cdots||K_t, M) = XLH^h_{IV}(K||K_0||\cdots||K_{L-1}, pad_s(M))
\end{aligned}$$

In the following theorem we show that none of Sh, ESh and XLH transforms can preserve eTCR. That is, we lose the best TCR preserving transform, *i.e.* Sh, as well as the multi-property preserving ESh transform when it comes to eTCR preservation.

Theorem 7 (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. For the formal proof, we provide the following simple attacks.

The Case of Shoup:
The following adversary $A = (A_1, A_2)$ can break the hash function H, obtained via Shoup domain extension transfrom (*i.e.* pad_s padding function followed by Sh^h_{IV} iteration method), 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. 4) 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.)

The Case of Enveloped Shoup:
For the ESh transform the attack strategy is quite similar to Sh case. Adversary $A = (A_1, A_2)$ plays the eTCR game as follows. A_1 outputs two different $(L-1)$-block messages $M = M_1||\cdots||M_{L-1}$ and $M' = M'_1||\cdots||M'_{L-1}$ which after applying $padCS_s$ padding function will become $M_1||\cdots||M_{L-1}||(10^{b-1-m-n}||\,\langle(L-1)b\rangle_m)$ and $M'_1||\cdots||M'_{L-1}||(10^{b-1-m-n}||\,\langle(L-1)b\rangle_m)$, respectively. That is, the inputs to ESh iteration function will have the same last block as $M_L = M'_L = 10^{b-1-m-n}\,\langle|M|\rangle_m$, but their first $(L-1)$ blocks are different (note that in ESh the length of the last block which is used in the final envelop is $b-n$ bits). In the second stage of eTCR attack, A_2, on receiving the first key, puts all blocks of the second key the same as the first given key *except* the last key block K_μ. A_2 simply adjusts the value of this last key block to a new key block $K'_\mu = K_\mu \oplus C_{L-1} \oplus C'_{L-1}$ to cancel the introduced difference in the chaining variables C_{L-1} and C'_{L-1} (related to the computation for M and M', respectively). We stress that this adjustment of the value of K_μ to K'_μ to cancel the difference that appears in final chaining value is possible because "K_μ is only used for the chaining variable fed into the envelope " as stated in [2].

The Case of XLH:
The attack is similar to the case of Shoup. Consider an adversary $A = (A_1, A_2)$ that can break the hash function H, obtained via XLH domain extension transform (*i.e.* pad_s padding function followed by XLH^h_{IV} iteration method), in eTCR sense. 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 the input to the three-round XLH^h_{IV} iteration. In the second stage of eTCR game, A_2, on receiving the first key as

$K||K_0||K_1||K_2$ 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 then puts the second key as $K||K_0||K'_1||K_2$, 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. 4) 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. Hence, $(K||K_0||K_1||K_2, M)$ and $(K||K_0||K'_1||K_2, M')$ constitute a colliding pair for H in eTCR sense. □

Remark. The eTCR adversaries used in the above proofs take advantage of XOR masking based structure of XLH, Sh and ESh transforms to cancel the effect of all accumulated differences in the internal state that may have been introduced by previous different message blocks, by simply adjusting the value of a last free key block. This implies that any class of such XOR masking based transforms that allows this cancellation phenomenon to happen will not be suitable for designing an eTCR preserving domain extender. It can be seen that this is the case for the XTH scheme of [5] as well.

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 for TCR preserving scenario as in [5], where it was shown that the LH method can be used to preserve TCR *only with respect to equal-length-collision finding adversaries* and its *nested variant* can be used to archive TCR for any variable-length-collision finding adversaries. We should mention that it was pointed out in [5] and latter shown by an explicit counterexample in [1] that LH iteration cannot preserve TCR with respect to variable length collisions.

After the previous series of negative results about inability of several efficient transforms to preserve eTCR, we now consider whether at least (but hopefully not the last) this most non-efficient 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 [5] 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 8 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 9 that a nested variant of LH can be made eTCR secure, *i.e.* against any arbitrary adversary.

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 8 (Positive Result). *Assume that the compression function $h : \{0,1\}^k \times \{0,1\}^{n+b} \to \{0,1\}^n$ is (t,ϵ)-eTCR. Then the hash function $H : \{0,1\}^{Nk} \times \{0,1\}^{\leq Nb} \to \{0,1\}^n$ 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.*

Proof. Assume that $A = (A_1, A_2)$ is an adversary which can break LH^h_{IV} in eTCR* sense (*i.e.* equal-length eTCR sense) with success probability ϵ' and using time complexity t'. We construct an adversary B that uses A to break the compression function h in eTCR sense. First we make the observation that if the adversary A is successful in finding two equal-length colliding messages $M = M_1 \cdots M_m$ and $M' = M'_1 \cdots M'_m$ under the keys $K = K_1|| \cdots ||K_m$ and $K' = K'_1|| \cdots K'_m$, then there must be an $i \in \{1, \cdots, m\}$ which the following two conditions hold:

(1): $LH^h_{IV}(K_1 \cdots K_i, M_1 \cdots M_i) = LH^h_{IV}(K'_1 \cdots K'_i, M'_1 \cdots M'_i)$
(2): $LH^h_{IV}(K_1 \cdots K_{i-1}, M_1 \cdots M_{i-1})||M_i \neq LH^h_{IV}(K'_1 \cdots K'_{i-1}, M'_1 \cdots M'_{i-1})||M'_i$ **OR** $K_i \neq K'_i$

This can be seen by noting that $|M| = |M'|$ and tracing back the computation in LH^h_{IV} iteration which may have made the final collision happen, that is $LH^h_{IV}(K_1 \cdots K_m, M_1 \cdots M_m) = LH^h_{IV}(K'_1 \cdots K'_m, M'_1 \cdots M'_m)$ where $(K, M) \neq (K', M')$ by winning condition for eTCR game.

Using the aforementioned observation we can build an adversary $B = (B_1, B_2)$ which can break eTCR property of h as follows:

Algorithm $B_1()$	Algorithm $B_2(Key, X, St)$
$(M, State) \xleftarrow{\$} A_1();\ \ m = \lvert M\rvert_b;$	$(j, M, K_1, \cdots, K_{j-1}, State) \leftarrow St;\ K_j = Key;$
$j \xleftarrow{\$} \{1, \cdots, m\};$	$K_{j+1}, \cdots, K_N \xleftarrow{\$} \{0,1\}^k;$
$K_1, \cdots, K_{j-1} \xleftarrow{\$} \{0,1\}^k;$	$(K', M') \xleftarrow{\$} A_2(K_1, \cdots, K_N, M, State);$
$X = LH^h_{IV}(K_1 \cdots K_{j-1}, M_1 \cdots M_{j-1})\lvert\rvert M_j;$	$X' = LH^h_{IV}(K'_1 \cdots K'_{j-1}, M'_1 \cdots M'_{j-1})\lvert\rvert M'_j;$
$St = (j, M, K_1, \cdots, K_{j-1}, State);$	$Key' = K'_j;$
return $(X, St);$	**return** $(Key', X');$

At the first stage of the eTCR game, B_1 outputs X as the target message together with the state information St to be passed to B_2 in the second stage of eTCR attack game. B_2 gets the first key for the compression function h denoted by Key which is selected uniformly at random by the challenger according to eTCR game. It outputs (Key', X') as the second key and message to finish eTCR game. It can be seen from the description of B that the distribution on key $K = K_1, \cdots, K_N$ given to A_2 is also uniform as expected in eTCR game against LH^h_{IV}. Now note that if A succeeds, there must be at least one index $i \in \{1, \cdots, m\}$ satisfying the two conditions (aforementioned conditions (1) and (2)) and as index j is selected at random by B_1 and independently from K, the probability that i matches to such an index is at least $\frac{1}{n} \geq \frac{1}{N}$. To complete the proof note that in this case, B also succeeds, that is, we have $(Key, X) \neq (Key', X')$ and $h(Key, X) = h(Key', X')$. This is seen from the way that messages X and X' are computed by algorithms B_1 and B_2, noting that $K_j = Key$ and $K'_j = Key'$ and referring to the two aforementioned conditions. Hence, if A succeeds with probability ϵ' then B also succeeds with probability $\epsilon \geq \frac{\epsilon'}{N}$. The time complexity of B (denote by t) is that of A (denote by t') plus the overhead $\Theta(N).\big(T_h + n + b + k\big)$ by the above reduction, where T_h is the time for one computation of the compression function h. □

The following theorem shows that the composition of a variable input length hash function which is secure only in the equal-length eTCR sense with a compression function which is eTCR secure will yield a variable input length hash function that is secure in eTCR sense.

Theorem 9 (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 $t = min\,\{t_1 - k_2, t_2 - k_1 - 2T_{H_1} - 2b\}$.*

Proof. Let $A = (A_1, A_2)$ be a (t, ϵ)-breaking adversary against H, *i.e.* having time complexity t and $\text{Adv}^{eTCR}_H(A) = \epsilon$. The experiment defining the eTCR attack by $A = (A_1, A_2)$ against H is as follows:

$$(M, State) \xleftarrow{\$} A_1(); K1 \xleftarrow{\$} \{0,1\}^{k_1}; K2 \xleftarrow{\$} \{0,1\}^{k_2}; (M', K1'||K2') \xleftarrow{\$} A_2(K1||K2, M, State) \qquad (8)$$

$\text{Adv}^{eTCR}_H(A)$ is defined as the probability that, after running the above experiment in (8), the following success event happens: $H(K1||K2, M) = H(K1'||K2', M') \ \bigwedge\ (K1||K2, M) \neq (K1'||K2', M')$. Let $x = H_1(K1, M)$ and $x' = H_1(K1', M')$. Let E1, E2, E3 be three events as follows:

- E1: A is successful **AND** $|M| = |M'|$ **AND** $x = x'$ **AND** $K2 = K2'$
- E2: A is successful **AND** $|M| = |M'|$ **AND** $x = x'$ **AND** $K2 \neq K2'$
- E3: A is successful **AND** ($|M| \neq |M'|$ **OR** $x \neq x'$)

Clearly E1, E2, and E3 are three disjoint events, and their union is the event that A succeeds in the eTCR attack against H. Let $p_1 = \Pr[\text{E1}]$, $p_2 = \Pr[\text{E2}]$, $p_3 = \Pr[\text{E3}]$, where probabilities are under the experiment defined in equation (8). That is, we have $\text{Adv}_H^{eTCR}(A) = p_1 + p_2 + p_3$. Therefore, we need to bound p_1, p_2, and p_3. To achieve this goal, using A as a subroutine, we show three adversaries $B = (B_1, B_2)$, $C = (C_1, C_2)$, and $D = (D_1, D_2)$: B can break H_1 in equal-length eTCR sense (whenever E1 happens) and has $\text{Adv}_{H_1}^{eTCR^*}(B) = p_1$, C can break h in eTCR sense (whenever E2) happens and has $\text{Adv}_h^{eTCR}(C) = p_2$, and D can break h in eTCR sense (whenever E3 happens) and has $\text{Adv}_h^{eTCR}(D) = p_3$. From our assumption in the statement of the Theorem 9 hat H_1 is (t_1, ϵ_1)-eTCR* and h is (t_2, ϵ_2)-eTCR, it must be the case that $\text{Adv}_h^{eTCR}(B) = p_1 \leq \epsilon_1$, $\text{Adv}_h^{eTCR}(C) = p_2 \leq \epsilon_2$, $\text{Adv}_h^{eTCR}(D) = p_3 \leq \epsilon_2$, and hence, we have $\text{Adv}_H^{eTCR}(A) = p_1 + p_2 + p_3 \leq \epsilon_1 + 2\epsilon_2$ as stated in the Theorem.

Now, we just need to show the algorithms for $B = (B_1, B_2)$, $C = (C_1, C_2)$ and $D = (D_1, D_2)$. The algorithms are as follows:

Algorithm $B_1()$
$(M, State) \xleftarrow{\$} A_1()$
return $(M, State)$

Algorithm $B_2(K1, M, State)$
$K2 \xleftarrow{\$} \{0,1\}^{k_2}$
$(K1'||K2', M') \xleftarrow{\$} A_2(K1||K2, M, State)$
return $(K1', M')$

Algorithm $C_1()$
$(M, State) \xleftarrow{\$} A_1()$
$K1 \xleftarrow{\$} \{0,1\}^{k_1}$
$x = H_1(K1, M)$
$y = x|| \langle |M| \rangle_b$
return $(y, (M, State, K1))$

Algorithm $C_2(K2, y, (M, State, K1))$
$(K1'||K2', M') \xleftarrow{\$} A_2(K1||K2, M, State)$
return $(K2', y)$

Algorithm $D_1()$
$(M, State) \xleftarrow{\$} A_1()$
$K1 \xleftarrow{\$} \{0,1\}^{k_1}$
$x = H_1(K1, M)$
$y = x|| \langle |M| \rangle_b$
return $(y, (M, State, K1))$

Algorithm $D_2(K2, y, (M, State, K1))$
$(K1'||K2', M') \xleftarrow{\$} A_2(K1||K2, M, State)$
$x' = H_1(K1', M')$
$y' = x'|| \langle |M'| \rangle_b$
return $(K2', y')$

The analysis is straightforward. Consider the eTCR attack experiment in Equation (8) and definition of the events E1, E2, E3. We claim that whenever E1 happens, the adversary $B = (B_1, B_2)$ becomes successful in attacking H_1. Note that when E1 happens $|M| = |M'|$ and hence B is an equal length eTCR attacker against H_1. To prove this claim, consider the definition of E1. Note that when A becomes successful in eTCR attack against $H = h \circ H_1$, we have $(K1||K2, M) \neq (K1'||K2', M')$ and $h(K2, H_1(K1, M)|| \langle |M| \rangle_b) = h(K2', H_1(K1', M')|| \langle |M'| \rangle_b)$. By definition of E1 we know that $x = H_1(K1, M) = H_1(K1', M') = x'$ and $K2 = K2'$, so the collision found by A must be an internal collision, *i.e.* a collision for H_1 and so adversary $B = (B_1, B_2)$ which attacks H_1 will be successful. That is, we have $\text{Adv}_{H_1}^{eTCR^*}(B) = \Pr[\text{E1}] = p_1$. The time complexity of B is $t_B = t + k_2$ and this is at most t_1 due to the assumption that H_1 is (t_1, ϵ_1)-eTCR*, that is, $t \leq t_1 - k_2$.

The analysis of success probability for the adversaries C and D which attack the eTCR property of the outer function h in $H = h \circ H_1$ can be provided similarly, just by noting the definitions for E2 and E3 events and the description of these adversaries.

Note that when E2 happens, we have $h(K2, x|| \langle |M| \rangle_b) = h(K2', x|| \langle |M| \rangle_b)$ (because A is successful) and $K2 \neq K2'$, hence adversary C becomes successful in eTCR attack against h as it outputs $y = x|| \langle |M| \rangle_b$ in the first stage and $(K2', y)$ in the second stage. Hence $(K2, y) \neq (K2', y)$ and $h(K2, y) = h(K2', y)$ as required for winning eTCR game against h. Therefore, we have $\mathrm{Adv}_h^{eTCR}(C) = \Pr[\mathrm{E2}] = p_2$. The time complexity of C is $t_C = t + k_1 + T_{H_1} + b$ and this is at most t_2 due to the assumption that h is (t_2, ϵ_2)-eTCR, that is, $t \leq t_2 - k_1 - T_{H_1} - b$.

When E3 happens, we have $h(K2, x|| \langle |M| \rangle_b) = h(K2', x'|| \langle |M'| \rangle_b)$ (because A is successful) and either $|M| \neq |M|'$ or $x \neq x'$. Hence, adversary D becomes successful in eTCR attack against h as it outputs $y = x|| \langle |M| \rangle_b$ in the first stage and $(K2', y' = x'|| \langle |M'| \rangle_b)$ in the second stage. Hence $(K2, y) \neq (K2', y')$ (because $y \neq y'$) and $h(K2, y) = h(K2', y')$ as required for winning eTCR game against h. Therefore, we have $\mathrm{Adv}_h^{eTCR}(D) = \Pr[\mathrm{E3}] = p_3$. Therefore, we have $\mathrm{Adv}_h^{eTCR}(C) = \Pr[\mathrm{E2}] = p_2$. The time complexity of D is $t_D = t + k_1 + 2T_{H_1} + 2b$ and this is at most t_2 due to the assumption that h is (t_2, ϵ_2)-eTCR, that is, $t \leq t_2 - k_1 - 2T_{H_1} - 2b$.

Note that the bound t in the statement of the Theorem, i.e. $t = min\{t_1 - k_2, t_2 - k_1 - 2T_{H_1} - 2b\}$, satisfies all the three bounds for t as required. □

Nested Linear Hash: Let H_1 be the equal-length eTCR hash function obtained via LH transform as stated in Theorem 8. From Theorem 9 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 9 (*i.e.* final block is just $\langle |M| \rangle_b$). Theorem 9 and Theorem 8 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 might 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 invention of the Enhanced Target Collision Resistance (eTCR) property by Halevi and Krawczyk [12] 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 compared the eTCR property with all of the seven security properties for a hash function, formalized by Rogaway and Shrimpton in FSE 2004, and provided a full picture of relationships between eTCR and each of the properties, namely CR, Sec, aSec, eSec, Pre, aPre and ePre, where all these properties are considered formally 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 *standard model*, which is *efficient*. We left this as an open problem in this paper.

References

[1] E. Andreeva, G. Neven, B. Preneel, T. Shrimpton: Seven-Property-Preserving Iterated Hashing: ROX. In: K. Kurosawa (ed.): ASIACRYPT 2007. LNCS, vol. 4833, pp. 130–146. Springer (2007)

[2] M. Bellare, T. Ristenpart: Hash Functions in the Dedicated-Key Setting: Design Choices and MPP Transforms. In L. Arge, C. Cachin, T. Jurdzinski, A. Tarlecki (eds.): ICALP 07. LNCS, vol. 4596, pp. 399–410. Springer (2007)

[3] M. Bellare, T. Ristenpart: Multi-Property-Preserving Hash Domain Extension and the EMD Transform. In X. Lai, K. Chen (eds.): ASIACRYPT 2006. LNCS, vol. 4284, pp. 299–314. Springer (2006)

[4] M. Bellare, A. Palacio: GQ and Schnorr Identification Schemes: Proofs of Security against Impersonation under Active and Concurrent Attacks. In M. Yung (ed.): CRYPTO 2002. LNCS, vol. 2442, pp. 162–177. Springer (2002)

[5] M. Bellare, P. Rogaway: Collision-Resistant Hashing: Towards Making UOWHFs Practical. In B.S. Kaliski Jr. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 470–484. Springer (1997)

[6] S. Contini, Y.L. Yin: Forgery and Partial Key-Recovery Attacks on HMAC and NMAC using Hash Collisions. In: X. Lai, K. Chen (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 37–53. Springer (2006)

[7] J.S. Coron, Y. Dodis, C. Malinaud, P. Puniya: Merkle-Damgård Revisited: How to Construct a Hash Function. In V. Shoup (ed.): CRYPTO 2005. LNCS, vol. 3621, pp. 430–448. Springer (2005)

[8] I. Damgård: A Design Principle for Hash Functions. In G. Brassard (ed.): CRYPTO 1989. LNCS, vol. 435, pp. 416–427. Springer (1990)

[9] C. De Cannière, C. Rechberger: Finding SHA-1 Characteristics: General Results and Applications. In: X. Lai, K. Chen (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 1–20. Springer (2006)

[10] B. den Boer, A. Bosselaers: Collisions for the Compressin Function of MD5. In: T. Helleseth (ed.): EUROCRYPT 1993. LNCS, vol. 765, pp. 293–304. Springer (1993)

[11] Y. Dodis, P. Puniya: Getting the Best Out of Existing Hash Functions; or What if We Are Stuck with SHA? In S.M. Bellovin, R. Gennaro, A.D. Keromytis, M. Yung (eds.): ACNS 2008. LNCS, vol. 5037, pp. 156–173. Springer (2008)

[12] S. Halevi, H. Krawczyk: Strengthening Digital Signatures Via Randomized Hashing. In C. Dwork (ed.): CRYPTO 2006. LNCS, vol. 4117, pp. 41–59. Springer (2006)

[13] U. Maurer, J. Sjödin: Single-Key AIL-MACs from Any FIL-MAC. In L. Caires, G.F. Italiano, L. Monteiro, C. Palamidessi, M. Yung (eds.): ICALP 05. LNCS, vol. 3580, pp. 472–484. Springer (2005)

[14] R.C. Merkle: One Way Hash Functions and DES. In G. Brassard (ed.): CRYPTO 1989. LNCS, vol. 435, pp. 428–446. Springer (1990)

[15] I. Mironov: Hash Functions: From Merkle-Damgård to Shoup. In B. Pfitzmann (ed.): EUROCRYPT 2001. LNCS, vol. 2045, pp. 166–181. Springer (2001)

[16] M. Naor, M. Yung: Universal One-Way Hash Functions and Their Cryptographic Applications. In: STOC 1989, pp. 33–43. ACM (1989)

[17] National Institute of Standards and Technology. Cryptographic Hash Algorithm Competition. http://csrc.nist.gov/groups/ST/hash/sha-3/index.html.

[18] 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.

[19] B. Preneel: The State of Cryptographic Hash Functions. In I. Damgård (ed.): Lectures on Data Security. LNCS, vol. 1561, pp. 158–182. Springer (1999)

[20] M.R. Reyhanitabar, W. Susilo, Y. Mu: Enhanced Target Collision Resistant Hash Functions Revisited. In O. Dunkelman (ed.): FSE 2009. LNCS, vol. 5665, pp. 327–344. Springer (2009)

[21] P. Rogaway: Formalizing Human Ignorance: Collision-Resistant Hashing without the Keys. In P.Q. Nguyen (ed.): VIETCRYPT 2006. LNCS, vol. 4341, pp. 211–228. Springer (2006)

[22] P. Rogaway, T. Shrimpton: Cryptographic Hash-Function Basics: Definitions, Implications, and Separations for Preimage Resistance, Second-Preimage Resistance, and Collision Resistance. Cryptology ePrint Archive: Report 2004/035 (Latest revised version: 9 Aug 2009).

[23] P. Rogaway, T. Shrimpton: Cryptographic Hash-Function Basics: Definitions, Implications, and Separations for Preimage Resistance, Second-Preimage Resistance, and Collision Resistance. In B.K. Roy, W. Meier (eds.): FSE 2004. LNCS, vol. 3017, pp. 371–388. Springer (2004)

[24] V. Shoup: A Composition Theorem for Universal One-Way Hash Functions. In: B. Preneel (ed.): EUROCRYPT 2000. LNCS, vol. 1807, pp. 445–452. Springer (2000)

[25] D.R. Simon: Finding Collisions on a One-Way Street: Can Secure Hash Functions be Based on General Assumptions? In: K. Nyberg (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 334–345. Springer (1998)

[26] D.R. Stinson: Some Observation on the Theory of Cryptographic Hash Functions. Journal of *Design, Codes and Cryptography*, Vol. 38, 259–277, 2006.

[27] X. Wang, X. Lai, D. Feng, H. Chen, X. Yu: Cryptanalysis of the Hash Functions MD4 and RIPEMD. In R. Cramer (ed.): EUROCRYPT 2005. LNCS, vol. 3494, pp. 1–18. Springer (2005)

[28] X. Wang, Y.L. Yin, H. Yu: Finding Collisions in the Full SHA-1. In V. Shoup (ed.): CRYPTO 2005. LNCS, vol. 3621, pp. 17–36. Springer (2005)

[29] X. Wang, H. Yu: How to Break MD5 and Other Hash Functions. In R. Cramer (ed.): EUROCRYPT 2005. LNCS, vol. 3494, pp. 19–35. Springer (2005)