How to Fill Up Merkle-Damgård Hash Functions

Kan Yasuda

NTT Information Sharing Platform Laboratories, NTT Corporation
3-9-11 Midoricho Musashino-shi, Tokyo 180-8585 Japan
yasuda.kan@lab.ntt.co.jp

Abstract. Many of the popular Merkle-Damgård hash functions have turned out to be not collision-resistant (CR). The problem is that we no longer know if these hash functions are even second-preimage-resistant (SPR) or one-way (OW), without the underlying compression functions being CR. We remedy this situation by introducing the "split padding" into a current Merkle-Damgård hash function H. The patched hash function $\bar{H}$ resolves the problem in the following ways: (i) $\bar{H}$ is SPR if the underlying compression function h satisfies an "SPR-like" property, and (ii) $\bar{H}$ is OW if h satisfies an "OW-like" property. The assumptions we make about h are provided with simple definitions and clear relations to other security notions. In particular, they belong to the class whose existence is ensured by that of OW functions, revealing an evident separation from the strong CR requirement. Furthermore, we get the full benefit from the patch at almost no expense: The new scheme requires no change in the internals of a hash function, runs as efficiently as the original, and as usual inherits CR from h. Thus the patch has significant effects on systems and applications whose security relies heavily on the SPR or OW property of Merkle-Damgård hash functions.

Keywords: hash function, Merkle-Damgård, padding, second-preimage resistance, one-wayness.

1 Introduction

Most of the modern cryptographic hash functions follow a design principle called the Merkle-Damgård construction. A main feature of such a hash function is that its collision resistance (CR) is guaranteed by that of its underlying compression function [21,8], yet unfortunately popular hash functions MD5 [28] and SHA-1 [24] are now shown to be *not* CR [37,38], hence losing the CR of their respective compression functions. These attacks have a profound impact on current systems using hash functions, not to mention those applications whose security is entirely based on the CR property of their hash-function components.

The loss of CR also exerts a strong influence on schemes whose security depends on the second-preimage resistance (SPR) or one-wayness (OW) of Merkle-Damgård hash functions. This is due to the fact that the SPR or OW security of such a hash function is hitherto ensured only by its CR (Recall that SPR is immediately implied by CR [23] and that OW is also implied by CR as long as

J. Pieprzyk (Ed.): ASIACRYPT 2008, LNCS 5350, pp. 272–289, 2008.

the hash function is "uniform" [36] or "sufficiently compressing" [31]). Now that the popular hash functions are not CR, we lose our proof-based assurance of the SPR and OW properties for these hash functions. To summarize:

> ***We have no guarantee whatsoever of the SPR or OW property for a Merkle-Damgård hash function without CR by its underlying compression function.***

This is the main problem we explore in the paper. We come up with a solution by first making a slight modification to the design of current hash functions. The change is fully compatible with a standard Merkle-Damgård interface. We then show that the patched hash functions indeed accomplish SPR and OW, assuming weaker-than-CR properties of the underlying compression functions.

Obtaining Upper-Bound Results for SPR and OW. A more direct way to overcome our problem in hand is to analyze exactly what sorts of properties the underlying compression function must possess in order to ensure the SPR and OW security of the Merkle-Damgård construction. [26,10] takes this approach and identifies complexity assumptions about the underlying compression function, which assure the SPR or OW property of the whole hash function. However, they do not consider the non-randomness involved in the padding or length-encoding bits.

Rather, we treat the problem of padding and length-encoding bits in detail.[1] The importance of these bits are already pointed out by [18,4]. We then come up with simple formulations of the complexity assumptions about the compression function, showing that these assumptions are indeed weaker than the CR requirement.

Need for SPR and OW Hash Functions. CR, SPR and OW are the three classical requirements for security of keyless hash functions (*e.g.*, [32,23]). We already know that the notion of CR plays an important role in designing cryptographic schemes. However, there are situations in which CR is not necessarily required but SPR or OW is essential to the security of systems. For example, adversaries might be unable to control input data to the hash function, say by protocol specification or by the fact that inputs are encrypted under a secret key before hashing.

A CR hash function may not be best suited to above scenarios due to its large hash size. Recall that for n-bit security the hash size of a CR hash function needs to be (at least) $2n$ bits. Suppose we want to use an SPR hash function with n-bit security. If the SPR security of hash functions were guaranteed only by their CR, then we would have needed to use a $2n$-bit hash function, whereas we could just use an n-bit hash function if the SPR security is directly guaranteed (not via its CR).

Our Results. We apply a patch to the Merkle-Damgård construction so that the SPR and OW properties are now guaranteed by certain reasonable and simple assumptions about the compression function.

[1] On the other hand, we assume that messages are distributed uniformly at random, which may not hold true in some of the practical applications, as pointed out by [3].

Split Padding. This is the patch. The new scheme works exactly the same as the original hash function except for the very end; the split-padding method alters processing of the last two blocks of a message. Message expansion is minimal, requiring at most one extra block (and such a case is rare). The new scheme is compatible with fixed-IV (initialization vector) usage and Merkle-Damgård strengthening. It can also handle a message in stream by delaying processing and buffering the two most recent blocks of the message.

CR Preservation. There is "nothing to lose" by applying the patch. Namely, we show that the CR preservation property of the original Merkle-Damgård construction is still in action with our new scheme. This motivates us to apply the patch to the systems whose hash functions are still CR (*e.g.*, SHA-256).

SPR Guarantee. This is one of the main features of our new mode of operation. The entire hash function can be proven SPR based on the assumption that the underlying compression function satisfies a property we call "cs-SPR" (chosen-suffix SPR).

OW Guarantee. This is the other main feature of the new construction. We show that the OW security of the entire hash function is ensured by the "cs-OW" (chosen-suffix OW) property of the underlying compression function.

Justification for cs-SPR and cs-OW. We give the rationale behind our choice of these assumptions by demonstrating their relationships with several known versions of SPR and OW properties. In particular, we show that these assumptions are strictly weaker than the strong CR requirement, by proving that they belong to the class whose existence is guaranteed by that of an OW function.

Without Random Oracles. We avoid use of random oracles in our proofs of security. The proofs of the SPR and OW properties are conducted in the standard model, following the concrete-security-reduction methodology. For the proof of CR preservation, we adopt the "human-ignorance" approach developed by [29].

Organization of the Paper. In Sect. 2 we review previous work related to the topic. Section 3 provides necessary definitions for the security of hash functions. In Sect. 4 we give a description of our patch "split padding." The proofs for the CR, SPR and OW properties of our scheme are given in Sect. 5, 6 and 7, respectively. We analyze the assumption cs-SPR in Sect. 8, followed by a similar analysis of cs-OW in Sect. 9. Section 10 presents certain application of our scheme.

2 Related Work

Merkle-Damgård Construction. [2] points out that the SPR or OW assumption about a compression function alone is *not* sufficient for the SPR or OW property of its Merkle-Damgård iteration. [19] observes that the use of a fixed IV precludes the trivial "truncation" or "free-start" SPR attack and that Merkle-Damgård strengthening (*i.e.*, length encoding in the final block) appears to defeat "long-message" SPR attacks. Then later it is shown that birthday-type

SPR attacks are still possible on the Merkle-Damgård construction [9,17], disproving the effectiveness of Merkle-Damgård strengthening against long-message SPR attacks.

Keyed (Randomized) Hash Functions. Hash functions in theory are often in the dedicated-key setting [6]. [31] describes seven security notions in such a setting: Coll, Sec, eSec, aSec, Pre, ePre and aPre. ROX [2] is a powerful mode of operation that preserves all the seven properties of the underlying compression function. Since aSec and aPre correspond with SPR and OW in the keyless setting, ROX can be used with its keys fixed, thereby as a keyless hash function. Unfortunate aspects of ROX are that it requires major modifications to the design of current hash functions and that its security is based on the use of random oracles.

BCM [3] achieves Coll, Sec and Pre properties with its proof of Sec security conducted in the standard model (That of Pre is in the random oracle model). However, aSec property is not achieved, and hence BCM is not suited to our keyless setting (BCM construction yields a keyed, Sec-secure hash function, and the keys cannot be fixed because it does not assure aSec).

There exist a number of domain-extension constructions for eSec(=TCR, UOWHF). Prominent one is the Randomized Hashing [13]. The Randomized Hashing has a close connection with our split padding, *cf.* Sect. 10.

SPR and OW Attacks on Specific Hash Algorithms. MD4 [27] is now shown to be not OW [18]. The attack first finds "pseudo-preimages" for the compression function and then extends them to the entire hash function. [4] studies the SPR and OW properties of the Snefru, being already aware of the importance of padding scheme to these security notions.

Other Weaknesses of Merkle-Damgård Construction. It has been pointed out that there exist a number of properties that the Merkle-Damgård construction does not achieve. These include multi-collisions [15], herding attacks [16], indifferentiability [7] and balance [5]. It is certainly *not* the purpose of our construction to achieve these goals. Rather, we focus on the classical three notions of CR, SPR and OW.

3 Definitions

Notation. Given a finite bit string $x \in \{0,1\}^*$ we define $|x|$ as the bit length of x. The notation $x_1 \| x_2$ represents the concatenation of two strings $x_1, x_2 \in \{0,1\}^*$. We write $\lceil \cdot \rceil$ for the ceiling function. By $x \xleftarrow{\$} X$ we indicate the operation of selecting an element uniformly at random from the set X and assigning its value to variable x. We write 0^n for the bit string $00 \cdots 0$ (n times).

Given a function $f : X \to Y$, we say that a pair $(x, x') \in X \times X$ is *colliding* with respect to f if $f(x) = f(x')$ and $x \neq x'$. We write $x \bowtie_f x'$, or simply $x \bowtie x'$, to indicate the fact that x and x' are colliding. Similarly, given a keyed function $f_k : X \to Y$ we write $(k, x) \bowtie_f (k', x')$ when $f_k(x) = f_{k'}(x')$ and $(k, x) \neq (k', x')$.

Algorithm *MD-strengthening*(y)
101 Set $\eta \leftarrow \lceil (|y|+1)/(m-n) \rceil$
102 Divide $y = y[1] \,\|\, \cdots \,\|\, y[\eta-1] \,\|\, y[\eta]$
so that $\big|y[1]\big| = \cdots = \big|y[\eta-1]\big| = m-n$ and $0 \le \big|y[\eta]\big| \le m-n-1$
103 If $\big|y[\eta]\big| \le m-n-\sigma-1$ then $z \leftarrow y \| 10^{m-n-\sigma-1-|y[\eta]|} \| \langle |y| \rangle$ EndIf
104 If $\big|y[\eta]\big| \ge m-n-\sigma$ then $z \leftarrow y \| 10^{m-n-1-|y[\eta]|} \| 0^{m-n-\sigma} \| \langle |y| \rangle$ EndIf
105 Output z

Algorithm *MD-iteration*(z) // Accepts only z with $|z|$ being a multiple of $m-n$
201 Set $\zeta \leftarrow |z|/(m-n)$
202 Divide $z = z[1] \,\|\, \cdots \,\|\, z[\zeta]$ so that $\big|z[1]\big| = \cdots = \big|z[\zeta]\big| = m-n$
203 Set $v[0] \leftarrow \mathit{IV}$; For $i = 1, \ldots, \zeta$ do $v[i] \leftarrow h\big(z[i] \,\|\, v[i-1]\big)$ EndFor; Output $v[\zeta]$

Fig. 1. Definitions of functions *MD-strengthening* and *MD-iteration*

An adversary A is a probabilistic algorithm that takes inputs. An adversary A may often be a pair of such algorithms, as $A = (A_1, A_2)$. We write $y \leftarrow A(x)$ to mean that adversary A outputs a value upon its input x, the output value being assigned to variable y.

Merkle-Damgård Construction. Throughout the paper we fix a compression function $h : \{0,1\}^m \to \{0,1\}^n$ with $m > n$. We also fix a value $\mathit{IV} \in \{0,1\}^n$. We choose a length-encoding function $\langle \cdot \rangle$, which takes an integer as its input and returns a σ-bit representation of the input value (A typical value of σ is 64). This restricts message lengths to a maximum of $2^\sigma - 1$ bits. Hence, the domain of the hash function should be written as $\{0,1\}^{\le 2^\sigma - 1}$ formally, but for simplicity we write $\{0,1\}^*$ to indicate the message space. Now the Merkle-Damgård hash function $H : \{0,1\}^* \to \{0,1\}^n$ is defined as

$$H(x) \stackrel{\text{def}}{=} \mathit{MD\text{-}iteration}\big(\mathit{MD\text{-}strengthening}(x)\big),$$

where the functions *MD-strengthening* and *MD-iteration* are as described in Fig. 1.[2] We adopt the convention that on empty input (*i.e.*, null string) function *MD-iteration* returns the value *IV*.

Let z and z' be two strings whose lengths are multiples of $m-n$ bits. Divide them into $(m-n)$-bit blocks as $z = z[1] \,\|\, \cdots \,\|\, z[\zeta]$ and $z' = z'[1] \,\|\, \cdots \,\|\, z'[\zeta']$. Suppose $\mathit{MD\text{-}iteration}(z) = \mathit{MD\text{-}iteration}(z')$ and $z \neq z'$. We define

$$\mathit{index}(z, z') \stackrel{\text{def}}{=} \begin{cases} \zeta & \text{if } \zeta \neq \zeta', \\ i & \text{if } \zeta = \zeta', \end{cases}$$

where i is defined to be the largest integer in $\{1, 2, \ldots, \zeta\}$ such that following (i), (ii) and (iii) hold: (i) $\mathit{MD\text{-}iteration}\big(z[1] \,\|\, \cdots \,\|\, z[i]\big) = \mathit{MD\text{-}iteration}$

[2] For a technical reason we deliberately define the iteration as $h\big(z[i] \,\|\, v[i-1]\big)$ at line 203 rather than as $h\big(v[i-1] \,\|\, z[i]\big)$.

Table 1. Complexity assumptions about keyless compression function $h : \{0,1\}^m \to \{0,1\}^n$

	(Alias)	Game
fp-CR	(c-SPR [13])	$\tilde{x} \xleftarrow{\$} \{0,1\}^\mu$, $(a, x') \leftarrow A(\tilde{x})$, $\tilde{x}\|a \stackrel{?}{\bowtie} x'$
cs-SPR		$(a, \mathsf{St}) \leftarrow A_1(\cdot)$, $\tilde{x} \xleftarrow{\$} \{0,1\}^\mu$, $x' \leftarrow A_2(\tilde{x}, \mathsf{St})$, $\tilde{x}\|a \stackrel{?}{\bowtie} x'$
SPR	(weak CR)	$x \xleftarrow{\$} \{0,1\}^m$, $x' \leftarrow A(x)$, $x \stackrel{?}{\bowtie} x'$
cs-OW		$(a, \mathsf{St}) \leftarrow A_1(\cdot)$, $\tilde{x} \xleftarrow{\$} \{0,1\}^\mu$, $v \leftarrow h(\tilde{x}\|a)$, $x' \leftarrow A_2(v, \mathsf{St})$, $v \stackrel{?}{=} h(x')$
ks-OW	(partial OW [23])	
		$a \xleftarrow{\$} \{0,1\}^{m-\mu}$, $\tilde{x} \xleftarrow{\$} \{0,1\}^\mu$, $v \leftarrow h(\tilde{x}\|a)$, $x' \leftarrow A(a, v)$, $v \stackrel{?}{=} h(x')$
OW	(preimage resistance)	$x \xleftarrow{\$} \{0,1\}^m$, $v \leftarrow h(x)$, $x' \leftarrow A(v)$, $v \stackrel{?}{=} h(x')$

$\big(z'[1] \,\|\, \cdots \,\|\, z'[i]\big)$, (ii) $z[j] = z'[j]$ for all $j \geq i+1$, (iii) Either $z[i] \neq z'[i]$ or $\mathsf{MD\text{-}iteration}\big(z[1] \,\|\, \cdots \,\|\, z[i-1]\big) \neq \mathsf{MD\text{-}iteration}\big(z'[1] \,\|\, \cdots \,\|\, z'[i-1]\big)$.

Complexity Assumptions about Keyless Compression Functions. Table 1 lists six notions of security for the compression function $h : \{0,1\}^m \to \{0,1\}^n$. Here we have a fixed security parameter $1 \leq \mu \leq m$. A typical value of μ is $\mu = n/2$ or $\mu = n$. Of the six notions, the two most important ones in the current work are cs-SPR and cs-OW, because these are the assumptions that we make about the underlying compression function h. Others are fp-CR (forced-prefix CR, *cf.* [35]), SPR, ks-OW (known-suffix OW) and OW. These four appear in the list only for the purpose of analyzing the nature of cs-SPR and cs-OW in Sect. 8 and 9, respectively.

The notion of cs-SPR is a variant of SPR where a suffix is chosen by adversaries. Informally, the game of cs-SPR for the compression function $h : \{0,1\}^m \to \{0,1\}^n$ proceeds as follows: First a suffix a of a given length is chosen by an adversary, and then a challenge $\tilde{x}$ of a given length is randomly drawn and shown to the adversary. The goal of the adversary is to find a second preimage $x' \neq \tilde{x}\|a$ such that $h(\tilde{x}\|a) = h(x')$. Here we emphasize that the adversary is required to commit on the value a before observing the challenge $\tilde{x}$.

Security Goals for Keyless Hash Functions. Our SPR and OW goals for a hash function $H : \{0,1\}^* \to \{0,1\}^n$ is formalized in Table 2. Note that an adversary can choose the challenge length $\lambda \geq \mu$ at the beginning of each game. Also note that the adversary's response x' may be of length different from λ.

Security Notions for Keyed Function Family. We utilize four notions, Coll, eColl (enhanced Coll), TCR (target collision resistance) and eTCR (enhanced

Table 2. Security goals for keyless hash function $H : \{0,1\}^* \to \{0,1\}^n$

	Game
SPR	$(\lambda, \mathsf{St}) \leftarrow A_1(\cdot)$, $x \xleftarrow{\$} \{0,1\}^\lambda$, $x' \leftarrow A_2(x, \mathsf{St})$, $x \stackrel{?}{\bowtie} x'$
OW	$(\lambda, \mathsf{St}) \leftarrow A_1(\cdot)$, $x \xleftarrow{\$} \{0,1\}^\lambda$, $v \leftarrow H(x)$, $x' \leftarrow A_2(v, \mathsf{St})$, $v \stackrel{?}{=} H(x')$

Table 3. Security notions for keyed function family $\varphi_k : \{0,1\}^m \to \{0,1\}^n$

(Alias)	Game
Coll	$k \xleftarrow{\$} K,\ (x,x') \leftarrow A(k),\ (k,x) \overset{?}{\bowtie} (k,x')$
eColl	$k \xleftarrow{\$} K,\ (x,k',x') \leftarrow A(k),\ (k,x) \overset{?}{\bowtie} (k',x')$
TCR (eSec, UOWHF)	$(x,\mathsf{St}) \leftarrow A_1(\cdot),\ k \xleftarrow{\$} K,\ x' \leftarrow A_2(k,\mathsf{St}),\ (k,x) \overset{?}{\bowtie} (k,x')$
eTCR	$(x,\mathsf{St}) \leftarrow A_1(\cdot),\ k \xleftarrow{\$} K,\ (k',x') \leftarrow A_2(k,\mathsf{St}),\ (k,x) \overset{?}{\bowtie} (k',x')$

TCR) for analyzing cs-SPR in Sect. 8. See Table 3. The notion of eColl appears to be new.

Advantage Functions and Adversarial Resources. For a CR-like or SPR-like goal, we define the advantage function of an adversary A as $\mathbf{Adv}^{\mathrm{goal}}_f(A) \stackrel{\mathrm{def}}{=} \Pr\big[\overset{?}{\bowtie} \text{ holds}\big]$, where f is the target function (h, H, *etc.*). Similarly, we define $\mathbf{Adv}^{\mathrm{goal}}_f(A) \stackrel{\mathrm{def}}{=} \Pr\big[\overset{?}{=} \text{ holds}\big]$ for an OW-like goal. The probabilities are over all coins defined in game and used by A. We fix a model of computation and measure the time complexity of adversaries. The time complexity of an adversary is the time for execution of its overlying game plus its code size. We let $\mathit{time}(h)$ denote the time complexity necessary for one computation of h. Define $\mathbf{Adv}^{\mathrm{goal}}_f(t,\ell) \stackrel{\mathrm{def}}{=} \max_A \mathbf{Adv}^{\mathrm{goal}}_f(A)$, where max runs over all adversaries A, with its time complexity being at most t, with λ, $|\mathsf{St}|$ and $|x'|$ each being at most ℓ blocks. A "block" is $m-n$ bits. ℓ may be omitted from the notation if irrelevant in the context.

4 How to Insert Split Padding

Our new hash function $\bar{H}$ operates exactly the same as the original hash H except for the last two blocks of messages. More precisely, the new hash function is defined as

$$\bar{H}(x) \stackrel{\mathrm{def}}{=} H\big(\mathit{split\text{-}padding}(x)\big),$$

with a plain Merkle-Damgård hash function $H : \{0,1\}^* \to \{0,1\}^n$. The definition of *split-padding* is given in Fig. 2 along with a pictorial representation in Fig. 3.

The basic idea of the "split padding" method is to make sure that every block input to the compression function h has at least μ bits of a message,[3] rather than being entirely padding bits or length-encoding bits. Indeed, when combined with *split-padding*, the function *MD-strengthening* never invokes line 104 of Fig. 1.

For this mechanism to work, we need to impose a condition $\sigma+1+2\mu \le m-n$ on $h : \{0,1\}^m \to \{0,1\}^n$. As long as this condition is fulfilled, the algorithm *split-padding* is well-defined. Also observe that no message bits are shared across the blocks. We will come back to this issue after proving the following basic result.

Proposition 1. *The function* split-padding *is one-to-one (i.e., injective).*

[3] Of course, here we are assuming that the message is at least μ bits long to begin with.

Algorithm *split-padding*(x)
301 Set $\xi \leftarrow \lceil (|x|+1)/(m-n) \rceil$
302 Divide $x = x[1] \parallel \cdots \parallel x[\xi-1] \parallel x[\xi]$
so that $\big|x[1]\big| = \cdots = \big|x[\xi-1]\big| = m-n$ and $0 \le \big|x[\xi]\big| \le m-n-1$
303 If $\mu \le \big|x[\xi]\big| \le m-n-\sigma-2$ then *pad-plain* EndIf
304 If $\big|x[\xi]\big| \le \mu - 1$ then *pad-with-borrow* EndIf
305 If $\big|x[\xi]\big| \ge m-n-\sigma-1$ then *pad-with-carry* EndIf
306 Output y

310 **Subroutine** *pad-plain*
311 Put $y \leftarrow x \| 0$

320 **Subroutine** *pad-with-borrow*
321 If $\xi \ge 2$ then divide $x[\xi-1] = \tilde{x}[\xi-1] \parallel \mathit{brw}$
so that $|\mathit{brw}| = \mu$ and $\big|\tilde{x}[\xi-1]\big| = m-n-\mu$
322 Set $\tilde{x}[\xi] \leftarrow \mathit{brw} \| x[\xi]$
323 Put $y \leftarrow x[1] \parallel \cdots \parallel x[\xi-2] \parallel \tilde{x}[\xi-1] \parallel 10^{\mu-1} \parallel \tilde{x}[\xi] \parallel 1$ EndIf
324 If $\xi = 1$ then put $y \leftarrow x[1] \parallel 1$ EndIf

330 **Subroutine** *pad-with-carry*
331 Divide $x[\xi] = \tilde{x}[\xi] \parallel \mathit{cry}$ so that $|\mathit{cry}| = \mu$ and $\big|\tilde{x}[\xi]\big| = \big|x[\xi]\big| - \mu$
332 Set $\tilde{x}[\xi+1] \leftarrow \mathit{cry}$
333 Put $y \leftarrow x[1] \parallel \cdots \parallel x[\xi-1] \parallel \tilde{x}[\xi] \parallel 10^{m-n-|\tilde{x}[\xi]|-1} \parallel \tilde{x}[\xi+1] \parallel 1$

Fig. 2. Description of "split padding" algorithm

Proof. Let $x, x' \in \{0,1\}^*$. We want to prove that the equality *split-padding*$(x) =$ *split-padding*(x') implies $x = x'$. So suppose we have x and x' such that the condition *split-padding*$(x) =$ *split-padding*(x') holds. Set $y \leftarrow$ *split-padding*(x) and $y' \leftarrow$ *split-padding*(x'). Divide $y = w\|b$ and $y' = w'\|b'$ so that $|b| = |b'| = 1$. The equality $y = y'$ tells us that $b = b'$ and $w = w'$.

Case A: $b = b' = 0$. In this case we know that both y and y' come from *pad-plain*. Hence, we must have $x = w$ and $x' = w'$, which yields $x = x'$.

Case B: $b = b' = 1$. We observe that in this case both y and y' originate from either *pad-with-borrow* or *pad-with-carry*. We divide the case according to the size $|y| = |y'|$.

Case B1: $|y| = |y'| \le m - n$. Note that *pad-with-carry* always produces more than or equal to two blocks of output, which implies that both y and y' must have been processed through *pad-with-borrow* in this case. Consequently, we get $x = w$, $x' = w'$ and $x = x'$.

Case B2: $|y| = |y'| > m - n$. Put $\eta \leftarrow \lceil (|y|+1)/(m-n) \rceil$. We must have $\eta \ge 2$. Divide $y = y[1] \parallel \cdots \parallel y[\eta-1] \parallel y[\eta]$ and $y' = y'[1] \parallel \cdots \parallel y'[\eta-1] \parallel y'[\eta]$, so that $\big|y[1]\big| = \cdots = \big|y[\eta-1]\big| = \big|y'[1]\big| = \cdots = \big|y'[\eta-1]\big| = m-n$. Recall that *pad-with-borrow* sets the last block to

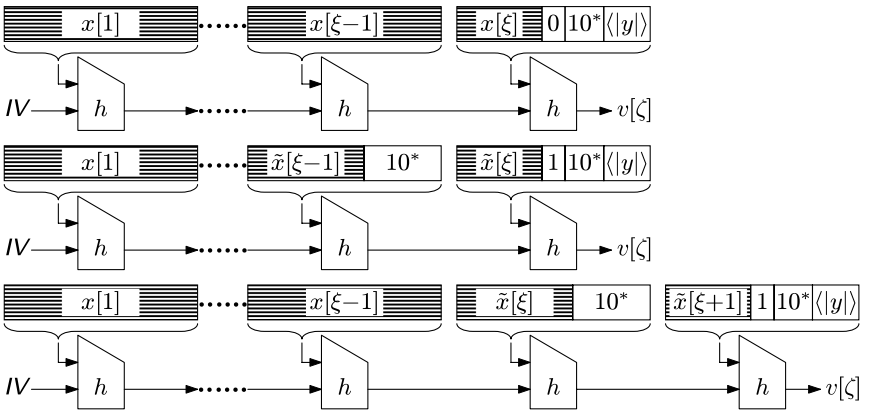

Fig. 3. Split padding: "plain" (top), "borrow" (middle) and "carry" (bottom), combined with *MD-strengthening*. Note that the last $10^* \| \langle |y| \rangle$ comes from *MD-strengthening*, not from *split-padding*. 10^* means $10 \cdots 0$ with an appropriate number of zeros.

a length between $\mu + 1$ and 2μ bits, whereas *pad-with-carry* always sets the last block to a length of $\mu + 1$ bits. Now we further divide the case according to the value $\big|y[\eta]\big|$.

Case B2a: $\big|y[\eta]\big| = \big|y'[\eta]\big| \geq \mu + 2$. This case assures that both y and y' originate from *pad-with-borrow*. It means that we can write $y[\eta - 1] = \tilde{y}[\eta - 1] \,\|\, 10^{\mu-1}$ and $y'[\eta - 1] = \tilde{y}'[\eta - 1] \,\|\, 10^{\mu-1}$. We can also write $y[\eta] = \tilde{y}[\eta] \,\|\, 1$ and $y'[\eta] = \tilde{y}'[\eta] \,\|\, 1$. Therefore, we obtain

$$x = y[1] \,\|\, \cdots \,\|\, y[\eta - 2] \,\|\, \tilde{y}[\eta - 1] \,\|\, \tilde{y}[\eta],$$
$$x' = y'[1] \,\|\, \cdots \,\|\, y'[\eta - 2] \,\|\, \tilde{y}'[\eta - 1] \,\|\, \tilde{y}'[\eta],$$

which immediately implies that $x = x'$.

Case B2b: $\big|y[\eta]\big| = \big|y'[\eta]\big| = \mu + 1$. There are multiple possibilities in this case: y and y' may come from either *pad-with-borrow* or *pad-with-carry*. To identify the case, we further divide $y[\eta - 1] = \tilde{y}[\eta - 1] \| 10^\alpha$ and $y'[\eta - 1] = \tilde{y}'[\eta - 1] \| 10^\alpha$ for some integer α. Observe that *pad-with-borrow* always sets $\alpha = \mu - 1$, whereas *pad-with-carry* sets $\alpha \geq \mu$. Hence, by looking at the value α we see that either (i) both y and y' are from *pad-with-borrow*, or (ii) both y and y' are from *pad-with-carry*. Write $y[\eta] = \tilde{y}[\eta] \,\|\, 1$ and $y'[\eta] = \tilde{y}'[\eta] \,\|\, 1$. We see that

$$x = y[1] \,\|\, \cdots \,\|\, y[\eta - 2] \,\|\, \tilde{y}[\eta - 1] \,\|\, \tilde{y}[\eta],$$
$$x' = y'[1] \,\|\, \cdots \,\|\, y'[\eta - 2] \,\|\, \tilde{y}'[\eta - 1] \,\|\, \tilde{y}'[\eta],$$

which gives us the desired equality $x = x'$.

Thus, we have shown that $\mathit{split\text{-}padding}(x) = \mathit{split\text{-}padding}(x')$ always implies $x = x'$. This proves the injectivity of the function $\mathit{split\text{-}padding}$. □

On the Constraint $\sigma + 1 + 2\mu \leq m - n$. We need to impose this constraint on the underlying compression function h. Thanks to the constraint, every block ends up containing at least μ bits of a message after the *split-padding* procedure.

The constraint is necessary for handling the last block properly. Recall that the subroutine *pad-with-borrow* produces the last block with a length *at most* $\mu + \mu - 1 + 1 = 2\mu$ bits. The subroutine *pad-with-carry* produces the last block with a length always equal to $\mu + 1$ bits. The constraint guarantees that the last block of either type, padded with *MD-strengthening*, fits neatly in a single block. The subroutine *pad-plain* by definition handles only the case when the last block fits in one block.

The constraint also guarantees that the second last block contains at least μ bits of the message. This holds true for *pad-plain*, *pad-with-borrow* and *pad-with-carry*.

The constraint is not problematic as long as we are dealing with MD5 or SHA-1 with $\mu = n/2$ or $n = \mu$. It puts an obstacle in the way of using SHA-256 with $\sigma = 64$ and $\mu = 256$. In such a case we are limited to setting the value of μ only up to $\mu \leq 223$.

5 CR of Merkle-Damgård with Split Padding

We follow the "human-ignorance" approach developed by [29] for formalizing the notion of CR.

Proposition 2. *Let $\bar{H} : \{0,1\}^* \to \{0,1\}^n$ be the Merkle-Damgård hash function with split padding constructed of a compression function $h : \{0,1\}^m \to \{0,1\}^n$. If there exists an explicitly-given adversary that finds a pair of colliding messages for $\bar{H}$ with a probability ε, spending time complexity at most t, each message being at most ℓ blocks, then there exists an explicitly-given adversary that finds a collision for h with a probability ε, spending time complexity at most $t' \approx t + 2\ell \cdot \mathit{time}(h)$.*

Proof. This statement immediately follows from the injectivity of *split-padding* and the well-known CR reduction of the plain Merkle-Damgård iteration [21,8]. □

6 SPR of Merkle-Damgård with Split Padding

In this section we prove that the patched hash function $\bar{H}$ is SPR assuming that the underlying compression function h is cs-SPR. After proving our result, we also discuss the birthday bound implied by the reduction.

Adversary B

410 Run $A = (A_1, A_2)$ and obtain a bit length $(\lambda, \mathsf{St}) \leftarrow A_1(\cdot)$ // $\lambda \geq \mu$
411 Generate a random message $x \xleftarrow{\$} \{0,1\}^\lambda$
412 Put $z \leftarrow \mathsf{MD\text{-}strengthening}\big(\mathsf{split\text{-}padding}(x)\big)$
413 Divide $z = z[1] \,\|\, z[2] \,\|\, \cdots \,\|\, z[\zeta]$ so that $|z[1]| = |z[2]| = \cdots = |z[\zeta]| = m - n$
414 Choose an index $i \xleftarrow{\$} \{1, 2, \ldots, \zeta\}$
415 Compute $v[i-1] \leftarrow \mathsf{MD\text{-}iteration}\big(z[1] \,\|\, z[2] \,\|\, \cdots \,\|\, z[i-1]\big)$
416 Divide $z[i] = \alpha \| \beta$ so that $|\alpha| = \mu$ and $|\beta| = m - n - \mu$
420 Submit $\beta \| v[i-1]$ as a committed suffix and receive a challenge $\tilde{x} \in \{0,1\}^\mu$
421 Put $w \leftarrow (\mathsf{MD\text{-}strengthening} \circ \mathsf{split\text{-}padding})^{-1}\big(z[1] \,\|\, \cdots \,\|\, z[i-1] \,\|\, \tilde{x} \,\|\, \beta \,\|\, z[i+1] \,\|\, \cdots \,\|\, z[\zeta]\big)$
430 Feed w and St to A_2 and obtain a second preimage $x' \leftarrow A_2(w, \mathsf{St})$
431 Put $z' \leftarrow \mathsf{MD\text{-}strengthening}\big(\mathsf{split\text{-}padding}(x')\big)$
432 Divide $z' = z'[1] \,\|\, \cdots \,\|\, z'[\zeta']$ so that $|z'[1]| = \cdots = |z'[\zeta']| = m - n$
433 If $\zeta \neq \zeta'$ then $v'[\zeta' - 1] \leftarrow \mathsf{MD\text{-}iteration}\big(z'[1] \,\|\, z'[2] \,\|\, \cdots \,\|\, z'[\zeta' - 1]\big)$
434 $x^* \leftarrow z'[\zeta'] \,\|\, v'[\zeta' - 1]$ EndIf
435 If $\zeta = \zeta'$ then $v'[i-1] \leftarrow \mathsf{MD\text{-}iteration}\big(z'[1] \,\|\, z'[2] \,\|\, \cdots \,\|\, z'[i-1]\big)$
436 $x^* \leftarrow z'[i] \,\|\, v'[i-1]$ EndIf
440 Output x^*

Fig. 4. Definition of adversary B attacking $h : \{0,1\}^m \to \{0,1\}^n$ in the cs-SPR sense

Theorem 1. *Let $\bar{H} : \{0,1\}^* \to \{0,1\}^n$ be the Merkle-Damgård hash function with split padding constructed of a compression function $h : \{0,1\}^m \to \{0,1\}^n$. Then $\bar{H}$ is SPR if h is cs-SPR. More concretely, we have*

$$\mathbf{Adv}^{\mathrm{spr}}_{\bar{H}}(t, \ell) \leq (\ell + 1) \cdot \mathbf{Adv}^{\text{cs-spr}}_{h}(t'),$$

where $t' \approx t + 2\ell \cdot \mathsf{time}(h)$. Note that the security parameter μ is implicit in the statement.

Proof. Let $A = (A_1, A_2)$ be an adversary attacking the hash function $\bar{H} : \{0,1\}^* \to \{0,1\}^n$ in the SPR sense. Assume that A has time complexity at most t and only handles strings whose lengths are at most ℓ blocks. We shall construct an adversary B that uses A_1 and A_2 as black-boxes and that attacks the underlying compression function h in the cs-SPR sense. The definition of B is given in Fig. 4. The basic idea is that B simulates an SPR game for A with B's challenge embedded into a randomly chosen block. Then B "hopes" that A finds a second preimage colliding at that block.

Let us first check if B simulates an SPR game for A correctly. In order to do this, we only need to verify that the distribution of simulated challenges w at line 421 is uniformly random on the set $\{0,1\}^\lambda$. The only difference between this w and the $x \in \{0,1\}^\lambda$ at line 411 is that the α in the i-th block is replaced with the challenge $\tilde{x}$. Because of the split padding, all the bits of $\alpha \in \{0,1\}^\mu$

come from the random message x. Since $\tilde{x}$ is random and independent from x, we see that w is indeed drawn uniformly at random from the set $\{0,1\}^\lambda$. The point here is that B can choose any block, as every block contains at least μ bits of a message owing to the split padding.

We next evaluate the success probability of B. We see that B succeeds whenever A succeeds, provided that at the same time B correctly guesses the index i (Here we need (i) injectivity of $\mathsf{split\text{-}padding}$, (ii) injectivity of $\mathsf{MD\text{-}strengthening}$, and (iii) the length encoding in $\mathsf{MD\text{-}strengthening}$). The choice of index i does not affect the overall distribution of $w \in \{0,1\}^\lambda$; the distribution is the same as $w \xleftarrow{\$} \{0,1\}^\lambda$, being independently random from i. Moreover, the value i is completely hidden from A. Therefore, the choice of i is independent from the transcript of A producing x'. Putting $u \leftarrow \mathsf{MD\text{-}strengthening}\big(\mathsf{split\text{-}padding}(w)\big)$ we get:

$$\begin{aligned}\mathbf{Adv}_h^{\text{cs-spr}}(B) &\geq \Pr\big[w \bowtie_{\bar{H}} x' \text{ and } i = \mathsf{index}(u, z')\big] \\ &= \Pr\big[w \bowtie x'\big] \times \Pr\big[i = \mathsf{index}(u, z') \;\big|\; w \bowtie x'\big] \\ &= \Pr\big[A \text{ succeeds}\big] \times (1/\zeta) \geq 1/(\ell+1) \cdot \mathbf{Adv}_{\bar{H}}^{\text{spr}}(A).\end{aligned}$$

Lastly, we compute the time complexity of adversary B. It is about equal to the time complexity of A plus *two* executions of $\mathsf{MD\text{-}iteration}$ at lines 415, 433 and 435, each of which costs at most $\ell \cdot \mathsf{time}(h)$. This proves the theorem. □

Remarks on the Birthday Bound. We note that our bound for SPR is of *quadratic* degradation in ℓ (a linear term in the coefficient of the advantage function and another one in time complexity). This means that the security guarantee becomes vacuous when $\ell \approx 2^{n/2}$ (with $\mu = n$; recall that with $t \approx 2^{n/2} \cdot \mathsf{time}(h)$ the advantage increases to about $2^{-n/2}$, *cf.* [23]). In fact, the long-message SPR attacks described in [9,17] are still applicable to the patched construction. It also implies that our reduction essentially gives a tight bound.

This is neither regression to the plain Merkle-Damgård construction nor severe limitation in practice. In the original Merkle-Damgård construction, the SPR of a hash function is assured up to the birthday bound based on the strong CR assumption about the underlying compression function rather than cs-SPR. The birthday bound by CR is at $t \approx 2^{n/2} \cdot \mathsf{time}(h)$ irrespective of the message length ℓ. Thus, our result provides a stronger bound than the one originally assured by CR. Also, in practice a typical value of σ restricts message lengths to less than $2^{n/2}$ blocks, so speaking of the security beyond $\ell \approx 2^{n/2}$ often becomes moot.

Moreover, many of the provably secure SPR (TCR) constructions, including Randomized Hashing [13] and Higher-Order UOWHF [14], are susceptible to the long-message SPR attacks, hence giving security only up to the birthday bound. It is true that there exist some constructions that accomplish SPR beyond the birthday bound, such as Wide-Pipe [20] and ROX [2], but the security of these constructions relies on the use of random oracles.

It seems a non-trivial task for us to construct a mode of operation that achieves the full SPR security without random oracles: The "dithering" and "checksum" require major modifications to the current Merkle-Damgård construction, and

these techniques are shown to be not effective in precluding long-message attacks [1,11].

7 OW of Merkle-Damgård with Split Padding

In this section we prove that the Merkle-Damgård construction combined with split padding is OW provided that the underlying compression function is cs-OW. The result contrasts sharply with the one for SPR of the previous section in that we have a security reduction without the birthday bound.

Theorem 2. *Let $\bar{H} : \{0,1\}^* \to \{0,1\}^n$ be the Merkle-Damgård hash function with split padding constructed of a compression function $h : \{0,1\}^m \to \{0,1\}^n$. Then $\bar{H}$ is OW if h is cs-OW. More concretely, we have*

$$\mathbf{Adv}^{\mathrm{ow}}_{\bar{H}}(t,\ell) \le \mathbf{Adv}^{\text{cs-ow}}_{h}(t'),$$

where $t' \approx t + 2\ell \cdot \mathsf{time}(h)$. Note that the security parameter μ is implicit in the statement.

Proof. Let $A = (A_1, A_2)$ be an adversary trying to invert the hash function $\bar{H} : \{0,1\}^* \to \{0,1\}^n$ in the OW sense. Assume that A has time complexity at most t and only handles strings whose lengths are at most ℓ blocks. We shall construct an adversary B that uses A_1 and A_2 as black-boxes and that tries to invert the underlying compression function h in the cs-OW sense. The definition of B is given in Fig. 5. A significant difference from the SPR case is that B simulates an OW game for A with its challenge embedded always into the last block.

We first check if B simulates an OW game for A correctly. For this, we need to verify that the distribution of the challenge v given at line 520 indeed coincides with the distribution $v \leftarrow \bar{H}(x)$, $x \xleftarrow{\$} \{0,1\}^\lambda$. By definition of cs-OW oracle, the value v at line 520 is computed as $v \leftarrow h(\tilde{x}\|\beta\|v[\zeta-1])$ with $\tilde{x} \xleftarrow{\$} \{0,1\}^\mu$. Now note that all the bits of α at line 515 come from the random message $x \in \{0,1\}^\lambda$ and appear in no other blocks, owing to the split padding. Since the randomness of $\tilde{x}$ is independent from that of x, replacing α with $\tilde{x}$ does not affect the distribution of v. Thus we see that B indeed simulates the correct distribution of v.

It remains to evaluate the success probability of B. We observe that B succeeds whenever A succeeds in inversion, so $\mathbf{Adv}^{\text{cs-ow}}_{h}(B) \ge \mathbf{Adv}^{\mathrm{ow}}_{\bar{H}}(A)$ holds. The time complexity of B is about that of A plus two executions of *MD-iteration* at lines 514 and 533. This proves the theorem. □

Tightness of the Bound. Unlike the case of SPR, this time the degradation is only linear in ℓ (*i.e.*, we do not have the coefficient $\ell+1$ in front of the advantage function).[4] This means that we still have some security left even when $\ell \approx 2^{n/2}$

[4] [26,10] obtains an OW result based on the OW assumption about h and the "output regularity" of h. The result, however, has a coefficient of ℓ in front of the advantage function (associated with the regularity).

Adversary B

510 Run $A = (A_1, A_2)$ and obtain a bit length $(\lambda, \mathsf{St}) \leftarrow A_1(\cdot)$ // $\lambda \geq \mu$
511 Generate a random message $x \xleftarrow{\$} \{0,1\}^\lambda$
512 Put $z \leftarrow \mathsf{MD\text{-}strengthening}\big(\mathsf{split\text{-}padding}(x)\big)$
513 Divide $z = z[1] \,\|\, z[2] \,\|\, \cdots \,\|\, z[\zeta]$ so that $|z[1]| = |z[2]| = \cdots = |z[\zeta]| = m - n$
514 Compute $v[\zeta - 1] \leftarrow \mathsf{MD\text{-}iteration}\big(z[1] \,\|\, z[2] \,\|\, \cdots \,\|\, z[\zeta - 1]\big)$
515 Divide $z[\zeta] = \alpha \| \beta$ so that $|\alpha| = \mu$ and $|\beta| = m - n - \mu$
520 Submit $\beta \| v[\zeta - 1]$ as a committed suffix and receive a challenge $v \in \{0,1\}^n$
530 Feed v to A_2 and obtain a preimage $x' \leftarrow A_2(v, \mathsf{St})$
531 Put $z' \leftarrow \mathsf{MD\text{-}strengthening}\big(\mathsf{split\text{-}padding}(x')\big)$
532 Divide $z' = z'[1] \,\|\, \cdots \,\|\, z'[\zeta']$ so that $|z'[1]| = \cdots = |z'[\zeta']| = m - n$
533 Compute $v'[\zeta' - 1] \leftarrow \mathsf{MD\text{-}iteration}\big(z'[1] \,\|\, z'[2] \,\|\, \cdots \,\|\, z'[\zeta' - 1]\big)$
534 Set $x^* \leftarrow z'[\zeta'] \,\|\, v'[\zeta' - 1]$
540 Output x^*

Fig. 5. Definition of adversary B attacking $h : \{0,1\}^m \to \{0,1\}^n$ in the cs-OW sense

(with $\mu = n$) and that long-message birthday attacks do not apply to the OW case.

Our bound for OW is "essentially" tight, except for the ℓ-degradation in time complexity. To see this, consider an inverter A (in the OW sense) attacking $\bar{H}$, who outputs $\lambda = \mu$ at the beginning of each game and receives a challenge $v \in \{0,1\}^n$. Then the challenge is computed as $v \leftarrow h(\tilde{x} \| a)$, $\tilde{x} \xleftarrow{\$} \{0,1\}^\mu$ with the suffix $a = 010^* \| \langle |\mu| \rangle \| IV$. This is "essentially" a cs-OW game on h, except that the suffix $a \in \{0,1\}^{m-\mu}$ is not completely "chosen" by A but rather "known" to A. We shall discuss more on the gap between cs-OW and ks-OW in Sect. 9.

8 Analysis of cs-SPR

The purpose of this section is to reveal the nature of cs-SPR. It is clear that CR implies cs-SPR[5] but not vice versa, so cs-SPR is a strictly weaker requirement than CR. Here we do want to say more; that is, we claim that cs-SPR is an assumption which is inherently weaker than CR, by showing:

A cs-SPR function exists if an OW function exists.

This is a complexity-theoretic result. It is known that the existence of Coll functions implies that of OW functions [12], but not vice versa—[34] shows that there exists no black-box construction of Coll functions from OW functions. This is a strong evidence of separation between the Coll property and the OW, and we show that cs-SPR belongs to the latter class.

[5] More formally, it should read "fp-CR implies cs-SPR."

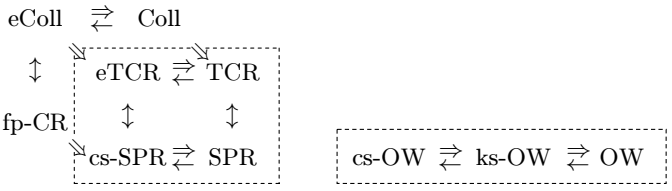

Fig. 6. A $\Rightarrow$ B indicates "A-secure implies B-secure," while A $\rightarrow$ B indicates "A-existence implies B-existence." Dotted boxes indicate black-box separation from the Coll requirement.

Our claim is based on the results [25,30] which prove that the existence of an OW function implies that of a TCR function family.[6] The existence of a TCR function family is equivalent to that of an SPR function [33], so we actually present explicit black-box construction of a cs-SPR function from an SPR function, thereby showing the existence of a cs-SPR function based on that of an OW function. For better understanding of cs-SPR, we also point out a symmetry between cs-SPR and eTCR; roughly speaking, cs-SPR can be regarded as an unkeyed version of eTCR. The diagram on the left in Fig. 6 summarizes the relationships among these various notions.

Proposition 3. *A cs-SPR function exists if an SPR function exists.*

Proof. Let $f : \{0,1\}^{\mu} \rightarrow \{0,1\}^{\nu}$ be an SPR function. Define $g : \{0,1\}^{m} \rightarrow \{0,1\}^{\nu+m-\mu}$ as $g(\tilde{x}\|a) \stackrel{\text{def}}{=} f(\tilde{x})\|a$ for $\tilde{x} \in \{0,1\}^{\mu}$ and $a \in \{0,1\}^{m-\mu}$. Then it can be directly verified that $\mathbf{Adv}_{g}^{\text{cs-spr}}(t) \leq \mathbf{Adv}_{f}^{\text{spr}}(t')$ where $t' \approx t$. □

Proposition 4. *A cs-SPR function exists if and only if an eTCR function family exists.*

Proof. Let $f : \{0,1\}^{m} \rightarrow \{0,1\}^{n}$ be a cs-SPR function with a security parameter $\mu < m$. Define a family of functions $\varphi_k : \{0,1\}^{m-\mu} \rightarrow \{0,1\}^{n}$ with $k \in \{0,1\}^{\mu}$ as $\varphi_k(a) \stackrel{\text{def}}{=} f(k\|a)$. Then it is easy to see that $\mathbf{Adv}_{\varphi}^{\text{etcr}}(t) \leq \mathbf{Adv}_{f}^{\text{cs-spr}}(t')$ where $t' \approx t$. Conversely, let $\varphi_k : \{0,1\}^{m-\mu} \rightarrow \{0,1\}^{n}$ be an eTCR function family with $k \in \{0,1\}^{\mu}$. Define $f : \{0,1\}^{m} \rightarrow \{0,1\}^{n}$ as $f(\tilde{x}\|a) \stackrel{\text{def}}{=} \varphi_{\tilde{x}}(a)$. Then we see that $\mathbf{Adv}_{f}^{\text{cs-spr}}(t) \leq \mathbf{Adv}_{\varphi}^{\text{etcr}}(t')$ where $t' \approx t$. □

The notion of cs-SPR provides a bridge between SPR and CR, depending on the value μ. This can be viewed as an unkeyed version of the continuum developed in [22]. Also, the notion of cs-SPR contrasts sharply with that of fp-CR, as there is a clear distinction between the two: The notion of fp-CR is open to generic birthday attacks, whereas that of cs-SPR is not.

[6] These results are based on polynomially-bounded reductions. [25] proves the existence based on that of an OW *permutation*, while [30] on that of an OW function.

9 Analysis of cs-OW

There are obvious implications: cs-OW-secure ⇒ ks-OW-secure ⇒ OW-secure. Thus our assumption cs-OW is the strongest of these three notions. We show that cs-OW is, however, not "too far" from OW, by proving (see the diagram on the right in Fig. 6):

A cs-OW function exists if an OW function exists.

Unlike the case of cs-SPR, we have a direct black-box construction this time:

Proposition 5. *A cs-OW function exists if an OW function exists.*

Proof. Let $f : \{0,1\}^\mu \to \{0,1\}^\nu$ be an OW function. Define $g : \{0,1\}^m \to \{0,1\}^{\nu+m-\mu}$ as $g(\tilde{x}\|a) \stackrel{\text{def}}{=} f(\tilde{x})\|a$ for $\tilde{x} \in \{0,1\}^\mu$ and $a \in \{0,1\}^{m-\mu}$. Then it can be directly verified that $\mathbf{Adv}_g^{\text{cs-ow}}(t) \leq \mathbf{Adv}_f^{\text{ow}}(t')$ where $t' \approx t$. □

10 Application to Randomized Hashing

In closing the paper we point out that our split padding is compatible with the Randomized Hashing [13]. Recall that the Randomized Hashing first mixes a message with a randomly generated mask and then hashes the data using the Merkle-Damgård construction. A problem arises when line 104 of the algorithm *MD-strengthening* in Fig. 1 is invoked, because it would then result in an insufficient amount of randomness in the last block. The Randomized Hashing suggests using "double padding" for dealing with this problem. Our split padding offers an alternative to this method, assuring at least μ bits of randomness in the very last invocation to the compression function.

References

1. Andreeva, E., Bouillaguet, C., Fouque, P.A., Hoch, J.J., Kelsey, J., Shamir, A., Zimmer, S.: Second preimage attacks on dithered hash functions. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 270–288. Springer, Heidelberg (2008)
2. Andreeva, E., Neven, G., Preneel, B., Shrimpton, T.: Seven-property-preserving iterated hashing: ROX. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 130–146. Springer, Heidelberg (2007)
3. Andreeva, E., Preneel, B.: A three-property-secure hash function. In: Avanzi, R., Keliher, L., Sica, F. (eds.) SAC 2008. Workshop Records, pp. 208–224 (2008)
4. Biham, E.: New techniques for cryptanalysis of hash functions and improved attacks on Snefru. In: Nyberg, K. (ed.) FSE 2008. LNCS, vol. 5086, pp. 444–461. Springer, Heidelberg (2008)
5. Bellare, M., Kohno, T.: Hash function balance and its impact on birthday attacks. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 401–418. Springer, Heidelberg (2004)

6. Bellare, M., Ristenpart, T.: Hash functions in the dedicated-key setting: Design choices and MPP transforms. In: Arge, L., Cachin, C., Jurdziński, T., Tarlecki, A. (eds.) ICALP 2007. LNCS, vol. 4596, pp. 399–410. Springer, Heidelberg (2007)
7. Coron, J.S., Dodis, Y., Malinaud, C., Puniya, P.: Merkle-Damgård revisited: How to construct a hash function. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 430–448. Springer, Heidelberg (2005)
8. Damgård, I.: A design principle for hash functions. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 416–427. Springer, Heidelberg (1990)
9. Dean, R.D.: Formal Aspects of Mobile Code Security. PhD thesis, Princeton University (1999)
10. Dodis, Y., Puniya, P.: Getting the best out of existing hash functions; or what if we are stuck with SHA? In: Bellovin, S.M., Gennaro, R., Keromytis, A.D., Yung, M. (eds.) ACNS 2008. LNCS, vol. 5037, pp. 156–173. Springer, Heidelberg (2008)
11. Gauravaram, P., Kelsey, J.: Linear-XOR and additive checksums don't protect Damgård-Merkle hashes from generic attacks. In: Malkin, T.G. (ed.) CT-RSA 2008. LNCS, vol. 4964, pp. 36–51. Springer, Heidelberg (2008)
12. Goldreich, O.: The Foundations of Cryptography, vol. 2. Cambridge University Press, Cambridge (2004)
13. Halevi, S., Krawczyk, H.: Strengthening digital signatures via randomized hashing. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 41–59. Springer, Heidelberg (2006)
14. Hong, D., Preneel, B., Lee, S.: Higher order universal one-way hash functions. In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329, pp. 201–213. Springer, Heidelberg (2004)
15. Joux, A.: Multicollisions in iterated hash functions. Application to cascaded constructions. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 306–316. Springer, Heidelberg (2004)
16. Kelsey, J., Kohno, T.: Herding hash functions and the Nostradamus attack. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 183–200. Springer, Heidelberg (2006)
17. Kelsey, J., Schneier, B.: Second preimages on n-bit hash functions for much less than 2^n work. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 474–490. Springer, Heidelberg (2005)
18. Leurent, G.: MD4 is not one-way. In: Nyberg, K. (ed.) FSE 2008. LNCS, vol. 5086, pp. 412–428. Springer, Heidelberg (2008)
19. Lai, X., Massey, J.L.: Hash function based on block ciphers. In: Rueppel, R.A. (ed.) EUROCRYPT 1992. LNCS, vol. 658, pp. 55–70. Springer, Heidelberg (1993)
20. Lucks, S.: A failure-friendly design principle for hash functions. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 474–494. Springer, Heidelberg (2005)
21. Merkle, R.C.: One way hash functions and DES. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 428–446. Springer, Heidelberg (1990)
22. Mironov, I.: Hash functions: From Merkle-Damgård to Shoup. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 166–181. Springer, Heidelberg (2001)
23. Menezes, A.J., van Oorschot, P.C., Vanstone, S.A.: Handbook of Applied Cryptography. CRC Press, Boca Raton (1996)
24. NIST: Secure Hash Standard. FIPS 180-1 (1995)
25. Naor, M., Yung, M.: Universal one-way hash functions and their cryptographic applications. In: ACM 21st STOC, pp. 33–43. ACM, New York (1989)
26. Puniya, P.: New Design Criteria for Hash Functions and Block Ciphers. PhD thesis, New York University (2007)

27. Rivest, R.L.: The MD4 message digest algorithm. In: Menezes, A., Vanstone, S.A. (eds.) CRYPTO 1990. LNCS, vol. 537, pp. 303–311. Springer, Heidelberg (1991)
28. Rivest, R.L.: The MD5 Message-Digest Algorithm. RFC 1321. IETF (1992)
29. Rogaway, P.: Formalizing human ignorance: Collision-resistant hashing without the keys. In: Nguyên, P.Q. (ed.) VIETCRYPT 2006. LNCS, vol. 4341, pp. 211–228. Springer, Heidelberg (2006)
30. Rompel, J.: One-way functions are necessary and sufficient for secure signatures. In: ACM 22nd STOC, pp. 387–394. ACM, New York (1990)
31. Rogaway, P., Shrimpton, T.: Cryptographic hash-function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance. In: Roy, B., Meier, W. (eds.) FSE 2004. LNCS, vol. 3017, pp. 371–388. Springer, Heidelberg (2004)
32. Schneier, B.: Applied Cryptography, 2nd edn. John Wiley & Sons, Chichester (1996)
33. Shoup, V.: A composition theorem for universal one-way hash functions. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 445–452. Springer, Heidelberg (2000)
34. Simon, D.R.: Finding collisions on a one-way street: Can secure hash functions be based on general assumptions? In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 334–345. Springer, Heidelberg (1998)
35. Stevens, M., Lenstra, A.K., de Weger, B.: Chosen-prefix collisions for MD5 and colliding X.509 certificates for different identities. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 1–22. Springer, Heidelberg (2007)
36. Stinson, D.R.: Some observations on the theory of cryptographic hash functions. Des. Codes Cryptography 38(2), 259–277 (2006)
37. Wang, X., Yu, H.: How to break MD5 and other hash functions. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 19–35. Springer, Heidelberg (2005)
38. Wang, X., Yin, Y.L., Yu, H.: Finding collisions in the full SHA-1. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 17–36. Springer, Heidelberg (2005)