A preliminary version of this paper appears in the ACM Conference on Computer and Communications Security (CCS), 2017. This is the full version.

Better Than Advertised: Improved Collision-Resistance Guarantees for MD-Based Hash Functions

Mihir Bellare[1] Joseph Jaeger[2] Julia Len[3]

June 2018

Abstract

The MD transform that underlies the MD and SHA families iterates a compression function h to get a hash function H. The question we ask is, what property X of h guarantees collision resistance (CR) of H? The classical answer is that X itself be CR. We show that weaker conditions X, in particular forms of what we call constrained-CR, suffice. This reduces demands on compression functions, to the benefit of security, and also, forensically, explains why collision-finding attacks on compression functions have not, historically, lead to immediate breaks of the corresponding hash functions. We obtain our results via a definitional framework called RS security, and a parameterized treatment of MD, that also serve to unify prior work and variants of the transform.

[1] Department of Computer Science & Engineering, University of California San Diego, 9500 Gilman Drive, La Jolla, California 92093, USA. Email: `mihir@eng.ucsd.edu`. URL: `http://cseweb.ucsd.edu/~mihir/`. Supported in part by NSF grants CNS-1526801, CNS-1717640, ERC Project ERCC FP7/615074 and a gift from Microsoft corporation.

[2] Department of Computer Science & Engineering, University of California San Diego, 9500 Gilman Drive, La Jolla, California 92093, USA. Email: `jsjaeger@eng.ucsd.edu`. URL: `https://cseweb.ucsd.edu/~jsjaeger/`. Supported in part by grants of first author.

[3] Department of Computer Science & Engineering, University of California San Diego, 9500 Gilman Drive, La Jolla, California 92093, USA. Email: `jlen2734@gmail.com`.

Contents

1 Introduction

The so-called MD transform [25, 16] iterates a compression function h to get a hash function H. The question we ask is, what property X of h guarantees collision resistance (CR) of H? The classical answer is that X itself be CR [25, 16]. We show that weaker conditions X, in particular forms of what we call constrained-CR, suffice.

The benefit is that if we ask less of compression functions (as we can now do), they are less likely to disappoint. Put another way, our result lowers the bar for the compression function designer, and raises it for the compression function attacker. It also explains an historical cryptanalytic phenomenon, namely that collision-finding attacks on compression functions [17, 31] have not immediately led to breaks of the corresponding hash functions. (Our explanation is that the attacks on the compression functions did not break constrained collision resistance.) In this (second) light, our work formalizes existing cryptanalytic intuition.

We obtain our results via a broader treatment that also serves to unify prior work and different variants of the transform, and to formalize folklore. It involves (1) a definitional framework called RS security that allows us to formulate both classical and new security goals in a unified way, and (2) a modular treatment of MD that parameterizes it via a splitting function and a compression function.

The MD transform was used in the MD series of hash functions (MD4 [28] and MD5 [27]) and now underlies the most widely used hash functions in practice, namely the SHA series (SHA-1, SHA-256, SHA-512) [26]. An improved understanding of its security, as we provide, is thus of both historical and current interest.

<u>MD FRAMEWORK.</u> We formulate MD in a general, parameterized way, as a transform taking (1) a compression function $\mathsf{h} : \{0,1\}^k \times (\{0,1\}^\mu \times \{0,1\}^\sigma) \to \{0,1\}^\sigma$, (2) a splitting function $\mathsf{Split} : D \to (\{0,1\}^\mu)^*$ and (3) a set $\mathrm{S} \subseteq \{0,1\}^\sigma$ of starting points (also called initial vectors) to return a hash function $\mathsf{H} = \mathbf{MD}[\mathsf{h}, \mathsf{Split}, \mathrm{S}] : D \to \{0,1\}^\sigma$. The compression function takes a key k and an input $x = (m, c)$ consisting of a message block m and chaining variable c, and returns output c' $= \mathsf{h}_k((m, c))$. The domain D is intended to be large, usually the set of all strings of length up to some big maximum length. The key (k, s) for H consists of a random key k for h and a random starting point s from S. The splitting function breaks the input M to $\mathsf{H}_{(k,\mathrm{s})}$ into a sequence $\mathbf{m} = \mathbf{m}[1] \ldots \mathbf{m}[n]$ of μ-bit blocks. To compute $\mathsf{H}_{(k,\mathrm{s})}(M)$, set $\mathbf{c}[1] \leftarrow \mathrm{s}$, iterate the compression function via

For $i = 1, \ldots, n$ do $\mathbf{c}[i+1] \leftarrow \mathsf{h}_k((\mathbf{m}[i], \mathbf{c}[i]))$,

and return $\mathbf{c}[n+1]$ as the value of $\mathsf{H}_{(k,\mathrm{s})}(M)$.

<u>CHARACTERIZING CR PRESERVATION.</u> We start by revisiting the classical question of showing that H is CR assuming h is CR (X=CR). Several works have noted that suffix-freeness of Split is sufficient for this purpose [18, 19, 1, 5]. (Some of these attribute the result to [25, 16], but neither paper appears to actually contain such a claim.) For completeness, we give, in our setting, a formal claim (suffix-freeness of Split plus CR of h implies CR of H, Theorem 5.3) together with the (easy) proof. We then complement this with a novel result: we show that the sufficient condition of suffix-freeness on Split is also *necessary*. We do this by showing that given *any* Split that is not suffix-free, we can construct a compression function h and set S such that (1) h is CR but (2) $\mathsf{H} = \mathbf{MD}[\mathsf{h}, \mathsf{Split}, \mathrm{S}]$ is *not* CR. This fully characterizes MD for the (classical) case where the assumption X made on h is CR.

<u>UNIFYING VARIANTS.</u> Papers, textbooks and standards present variants of the MD transform that differ in details. We can capture them as special cases, corresponding to different choices of splitting

function Split and set S of starting points. Together with our above-mentioned characterization, this unifies prior work.

To elaborate, a basic version of MD, from Merkle [24], MOV [23] and KL [21], corresponds to the splitting function that pads the message M to a multiple of the block length μ and appends a block encoding the length of M. Stinson's [32] version corresponds to the last block encoding the amount of padding rather than the message length. Damgård's version [16] starts each block of the padded message with a 1 bit except the first, which it starts with a 0 bit, and also appends a block encoding the amount of padding. The SHA functions [26] use yet another variant where the (padded) message may spill into the last block so that the latter does not encode just the message length (cf. Fig. 3). In papers and textbooks the starting point s is usually 0^σ [24, 16, 23, 21, 32], but, in the SHA series, s differs across hash functions. (For example, for SHA-256 it is the first 32 bits of the square roots of each of the first 8 primes.) All these MD variants can be captured in our framework as making particular choices of suffix-free splitting functions Split and (singleton) spaces S.

CR is not necessary. We would like to show CR of $\mathsf{H} = \mathbf{MD}[\mathsf{h}, \mathsf{Split}, \mathrm{S}]$ under an assumption X on the compression function h that is weaker than CR. We first ask, is this even possible? Or, is CR of h necessary for CR of H? We show in Section 6 that CR of h is *not* necessary. Given a suffix-free Split, we build a compression function h and set S such that (1) h is *not* CR, yet (2) $\mathsf{H} = \mathbf{MD}[\mathsf{h}, \mathsf{Split}, \mathrm{S}]$ is CR. This opens the door to proving CR of H under relaxed assumptions on h.

RS security. But what would these assumptions be? Towards finding and formulating them, we step back to give a framework to define security goals for h. Security is parameterized by a relation R and a set S. The game gives the adversary $\mathcal{A}$ a random key k for h and a random point s in S. It returns an object denoted *out*, and wins if R, given k, s, out, returns true. Its RS-advantage is the probability that it wins. Classical X=CR is captured by viewing *out* as a pair of strings that R checks are a collision under h_k, with s not being involved, formally $\mathrm{S} = \{\varepsilon\}$. A form of pre-image resistance that we will use is captured by having R check that *out* gives a pre-image of $\mathrm{s} \in \mathrm{S} = \{0,1\}^\sigma$ under h_k. We can also capture constrained forms of collision resistance (ccr), so called because extra requirements are made on the collision, thereby constraining it. In particular, we define $\mathsf{R}_{\mathrm{ccr}}\mathrm{S}$ security. Here winning requires that *out* contains, not only a collision $(m_1, c_1), (m_2, c_2)$ for h_k, but also, for both $j = 1, 2$, if $c_j \neq \mathrm{s}$, a further pre-image of it under h_k. Providing the auxiliary information in addition to a collision makes the adversary's job harder, so X=$\mathsf{R}_{\mathrm{ccr}}\mathrm{S}$ security is a weaker assumption on h than CR. We can define other relaxations of CR as well.

CR from CCR. Theorem 6.4 relaxes the CR assumption, made on the compression function h in Theorem 5.3, to $\mathsf{R}_{\mathrm{ccr}}\mathrm{S}$ security. That is, we show that if Split is suffix free and h is $\mathsf{R}_{\mathrm{ccr}}\mathrm{S}$ secure, then hash function $\mathsf{H} = \mathbf{MD}[\mathsf{h}, \mathsf{Split}, \mathrm{S}]$ is CR secure. The first consequence of this is that the bar is lowered for the compression function designer (their design only needs to provide $\mathsf{R}_{\mathrm{ccr}}\mathrm{S}$ security, which is easier than providing CR) and raised for the cryptanalyst (their attack needs to violate $\mathsf{R}_{\mathrm{ccr}}\mathrm{S}$ security, which is harder than violating CR). We now discuss another consequence, namely to (possibly) better understand some cryptanalytic history.

Already in 1996, Dobbertin had found collisions for the compression function md5 of MD5 [17]. This did not, however, yield collisions on MD5 itself. This, to us, was an indication that MD was "better than advertised:" it was (possibly) able to promote a non-CR compression function to a CR hash function. Our work is an attempt to capture this intuition formally. Now, it is true that in this particular case the hope was not realized, meaning MD5 failed to be CR, as shown by direct attack [35, 34]. What that tells us is that the compression function md5 is even weaker than we thought: not only is it not CR, it is not even $\mathsf{R}_{\mathrm{ccr}}$S. In fact, starting from known MD5 collisions, our reduction will construct collisions, and accompanying auxiliary information, to violate $\mathsf{R}_{\mathrm{ccr}}$S security of md5. The story repeats with SHA-1, where collisions found for the compression function sha-1 [31] did not immediately yield collisions for SHA-1, but the latter have now been found [30]. Again, it means sha-1 is not even $\mathsf{R}_{\mathrm{ccr}}$S. This, in our view, improves our understanding of compression function security.

Speeding up MD. Suppose the message $M = 0^{2\mu-1}$ to be hashed is a bit short of twice the block length μ. A typical suffix-free encoding (for example that of SHA-256) will pad M and append an encoding of the length $|M|$, to result in a 3-block string $\mathbf{m}$, over which the compression function is iterated. The compression function is thus called 3 times. One might hope for better, just 2 calls. More generally, the savings from dropping one compression function call are significant since messages in practice are often short. This leads us to ask why not use a minimal splitting scheme, like just 10^* pad the message, which in our example results in $\mathbf{m} = 0^{2\mu-1}1$ being 2 blocks long, so that MD will use only two calls to the compression function. But this splitting function is not suffix-free, and did we not show that the suffix-freeness assumption on Split is necessary for CR of H? Yes, but that was when the assumption on h is CR. Hence it is of course also true when the assumption is $\mathsf{R}_{\mathrm{ccr}}$S, since that is implied by CR, but in Section 7 we show that mere injectivity of Split (in particular 10^* padding of the message) does guarantee CR of $\mathsf{H} = \mathbf{MD}[\mathsf{h}, \mathsf{Split}, \mathrm{S}]$ under *alternative* assumptions on h, specifically that it is both CCR and fixed-point resistant. The assumption seems quite plausible compared to CR, so the performance gain and simplicity of the splitting could make this version of MD attractive.

We note that the proceedings version of this paper [10] assumes pre-image resistance of the compression function instead of the weaker notion of fixed-point resistance. Our new result thus improves on this version as well as a similar result of Dodis and Puniya [18], which uses full collision resistance instead of CCR.

In Section 8 we then specify relationships between all of the notions we defined in the RS security framework. Furthermore, we extend proof techniques of Dodis and Puniya [18] to show that injectivity of Split can additionally be proven to suffice assuming that H is CCR and satisfies a notion of output uniformity.

Reduction complexity. As indicated above, many prior works have claimed or proved that CR of h implies CR of H, either for particular choices of Split or assuming the latter is suffix free. It is interesting that, with the exception of a work on formal verification [5], not only papers [25, 16, 18, 19, 1, 5], but also textbooks [32, 21, 23], fail to explicitly specify the reduction underlying the proof. This takes attention away from, and makes it difficult to address, the important question of the (computational) complexity (efficiency) of the reduction. Whether in showing CR or CCR of h implies CR of H, we in contrast are interested in the precise complexity of the reduction. We accordingly give explicit, pseudocode reductions. In the main sections, we give the reductions that emanate naturally from the proof. Then, in Section 9, we revisit the question of complexity to give alternative reductions that are more memory-efficient [4].

Discussion and related work. MD-based hash functions are also used for HMAC [8]. If we contemplate changes in splitting functions, we want to ensure HMAC security is preserved. However, current analyses of HMAC security [6, 20] show that suffix-free, and even injective, splitting functions suffice.

Our focus is on MD as a way to achieve collision resistance. Other works have looked at it for other ends. Use of MD with prefix-free (as opposed to suffix-free) encodings has been shown in [9, 7] to preserve PRF security. Its ability to provide indifferentiability from a random oracle is studied in [19]. More broadly, MD is one of many possible domain extension methods, and some works [11, 2] consider methods that preserve multiple properties.

2 Notation and conventions

If $\mathbf{m}$ is a vector then $|\mathbf{m}|$ denotes its length, $\mathbf{m}[i]$ denotes its i-th coordinate and $\mathbf{m}[i..j]$ denotes the vector consisting of coordinates i through j of $\mathbf{m}$. For example if $\mathbf{m} = (010, 11, 10, 111)$ then $|\mathbf{m}| = 4$, $\mathbf{m}[2] = 11$, $\mathbf{m}[2..4] = (11, 10, 111)$. By ε we denote the empty vector, which has length 0. If D is a set, we say that $\mathbf{m}$ is a vector over D if all its components belong to D, and we let D^* denote the set of all finite-length vectors over D. If $\mathbf{m}, \mathbf{y}$ are vectors, their concatenation, denoted $\mathbf{m}\|\mathbf{y}$, is the vector $(\mathbf{m}[1], \ldots, \mathbf{m}[|\mathbf{m}|], \mathbf{y}[1], \ldots, \mathbf{y}[|\mathbf{y}|])$. For example $(01, 11, 1)\|(10, 000) = (01, 11, 1, 10, 000)$.

A string y is identified with a vector over $\{0,1\}$, so that $|y|$ denotes its length, $y[i]$ denotes its i-th bit and $y[i..j]$ denotes bits i through j of y. For example if $y = 0100$ then $|y| = 4$, $y[2] = 1$ and $y[2..4] = 100$. In this case, ε denotes the empty string, $\{0,1\}^*$ is the set of all binary strings, $x\|y$ denotes the concatenation of strings x, y. For example $010\|11 = 01011$. By $\overline{y}$ we denote the bitwise complement of string y. (For example if $y = 010$ then $\overline{y} = 101$.)

By $\mathbb{N} = \{0, 1, 2, ...\}$ we denote the set of all non-negative integers. For $p \in \mathbb{N}$ with $p \geq 2$, we let $\mathbb{Z}_p = \{0, 1, \ldots, p-1\}$ denote the set of integers modulo p. If $x, n \in N$ satisfy $0 \leq x < 2^n$ then $\langle x \rangle_n$ denotes the encoding of x as a binary string of length (exactly) n. For example $\langle 7 \rangle_4 = 0111$.

If X is a finite non-empty set, we let $x \leftarrow\$ X$ denote picking an element of X uniformly at random and assigning it to x. Algorithms may be randomized unless otherwise indicated. Running time and memory usage are worst case. If A is an algorithm, we let $y \leftarrow A(x_1, \ldots; r)$ denote running A with random coins r on inputs $x_1, \ldots$ and assigning the output to y. We let $y \leftarrow\$ A(x_1, \ldots)$ be the result of picking r at random and letting $y \leftarrow A(x_1, \ldots; r)$. We let $[A(x_1, \ldots)]$ denote the set of all possible outputs of A when invoked with inputs $x_1, \ldots$.

We use the code based game playing framework of [12]. (See Fig. 1 for an example.) By Pr[G] we denote the probability of the event that the execution of game G results in the game returning true. We adopt the convention that the running time of an adversary refers to the worst-case execution time of the game with the adversary. We adopt the analogous convention for the memory usage. This means that usually in reductions, adversary time and memory complexity can be roughly maintained.

3 RS Security framework

<u>Function families.</u> A function family $\mathsf{F} : \mathsf{F.Keys} \times \mathsf{F.Inp} \to \mathsf{F.Out}$ is a 2-argument function taking a key fk in the keyspace $\mathsf{F.Keys}$ and an input x in the input space $\mathsf{F.Inp}$ to return an output $\mathsf{F}(fk, x)$ in the output space $\mathsf{F.Out}$. For $fk \in \mathsf{F.Keys}$ we let $\mathsf{F}_{fk} : \mathsf{F.Inp} \to \mathsf{F.Out}$ be defined by $\mathsf{F}_{fk}(x) = \mathsf{F}(fk, x)$ for all $x \in \mathsf{F.Inp}$.

<u>RS security.</u> Our definition of security for a function family F is parameterized by a relation $\mathsf{R} : \{0,1\}^* \times \{0,1\}^* \times \{0,1\}^* \to \{\mathsf{true}, \mathsf{false}\}$ and a set $\mathrm{S} \subseteq \{0,1\}^*$. Different choices of the pair (R, S) allow us to recover classical definitions including collision resistance, and to specify extensions and variants including constrained collision resistance. The formalism considers game $\mathbf{G}^{\mathsf{RS}}_{\mathsf{F}}(\mathcal{A})$ of Fig. 1 associated to R, S, F and adversary $\mathcal{A}$. The latter is given the key fk and a challenge point s drawn randomly from S, and returns some output denoted out. It wins (the game returns true) if

```
Game $\mathbf{G}^{\mathsf{RS}}_{\mathsf{F}}(\mathcal{A})$
$fk \leftarrow\!\!\$\ \mathsf{F.Keys}$ ; $\mathrm{s} \leftarrow\!\!\$\ \mathrm{S}$ ; $out \leftarrow\!\!\$\ \mathcal{A}(fk, \mathrm{s})$
Return $\mathsf{R}(fk, \mathrm{s}, out)$
```

R	out	$\mathsf{R}(fk, \mathrm{s}, out)$ returns true iff	
R_{cr}	(x_1, x_2)	$x_1 \neq x_2$ and $\mathsf{F}_{fk}(x_1) = \mathsf{F}_{fk}(x_2)$	Collision resistance
$\mathsf{R}_{\mathrm{pre}}$	x	$\mathsf{F}_{fk}(x) = \mathrm{s}$	Pre-image resistance
$\mathsf{R}_{\mathrm{ccr}}$	$((m_1, c_1), (m_2, c_2), (a_1, a_2))$	$\mathsf{R}_{\mathrm{cr}}(fk, \varepsilon, (m_1, c_1), (m_2, c_2)) \wedge (c_1 \in \{\mathrm{s}, \mathsf{F}_{fk}(a_1)\}) \wedge (c_2 \in \{\mathrm{s}, \mathsf{F}_{fk}(a_2)\})$	Constrained CR
$\mathsf{R}_{\mathrm{fix}}$	$\mathbf{m}$	$\mathsf{F}_{fk}(\mathbf{m}[\lvert\mathbf{m}\rvert], \ldots (\mathsf{F}_{fk}(\mathbf{m}[2], \mathsf{F}_{fk}(\mathbf{m}[1], \mathrm{s})))) = \mathrm{s}$	Fixed-point resistance

Figure 1: **Top:** Game for defining R-security of function family F. **Bottom:** Some relations we will use. For R_{cr} we have $\mathrm{s} = \varepsilon$. For $\mathsf{R}_{\mathrm{ccr}}$ and $\mathsf{R}_{\mathrm{fix}}$, function family $\mathsf{F} : \mathsf{F.Keys} \times (\mathsf{F.Bl} \times \mathsf{F.Out}) \to \mathsf{F.Out}$ is a compression function.

relation R returns true on inputs fk, s, out. The advantage of $\mathcal{A}$ relative to R, S, also called its RS advantage, is defined as $\mathbf{Adv}^{\mathsf{RS}}_{\mathsf{F}}(\mathcal{A}) = \Pr[\mathbf{G}^{\mathsf{RS}}_{\mathsf{F}}(\mathcal{A})]$, the probability that the game returns true.

COLLISION RESISTANCE. Recall that a *collision* for a function f is a pair of distinct points x_1, x_2 in the domain of f such that $f(x_1) = f(x_2)$. Classical collision resistance of a function family F asks that it be hard for an adversary $\mathcal{A}$, given fk, to find a collision x_1, x_2 for the function F_{fk}. In our framework, this is $\mathsf{R}_{\mathrm{cr}}\mathrm{S}_{\varepsilon}$ security, where $\mathrm{S}_{\varepsilon} = \{\varepsilon\}$ consists of just the empty string and $\mathsf{R}_{\mathrm{cr}}(fk, \mathrm{s}, out)$ parses out as a pair, $(x_0, x_1) \leftarrow out$, and returns true iff $\mathsf{F}(fk, x_1) = \mathsf{F}(fk, x_2)$ and $x_1 \neq x_2$, meaning x_1, x_2 is a collision for F_{fk}. We recover familiar notation for collision resistance by letting game $\mathbf{G}^{\mathrm{cr}}_{\mathsf{F}}(\mathcal{A}) = \mathbf{G}^{\mathsf{R}_{\mathrm{cr}}\mathrm{S}_{\varepsilon}}_{\mathsf{F}}(\mathcal{A})$ and $\mathbf{Adv}^{\mathrm{cr}}_{\mathsf{F}}(\mathcal{A}) = \mathbf{Adv}^{\mathsf{R}_{\mathrm{cr}}\mathrm{S}_{\varepsilon}}_{\mathsf{F}}(\mathcal{A})$.

PRE-IMAGE RESISTANCE. This is a form of one-wayness where the adversary, given fk and challenge s, tries to recover a pre-image of s under F_{fk}. Generalizing [29], our formalization is parameterized by the set S from which s is drawn, and is obtained via our RS framework, as follows. Let $\mathsf{R}_{\mathrm{pre}}(fk, \mathrm{s}, out)$ return true iff $\mathsf{F}_{fk}(out) = \mathrm{s}$, meaning $out \in \mathsf{F.Inp}$ is a pre-image of s under F_{fk}. Then $\mathsf{R}_{\mathrm{pre}}\mathrm{S}$ security captures pre-image resistance for challenges drawn from S. We further discuss this notion, and its relation to other types of pre-image resistance, in Section **??**.

RESTRICTED COLLISION RESISTANCE. Restricted collision resistance makes the adversary's job harder by asking that the collision x_1, x_2 satisfy some additional condition that will be specified by R. We will describe the particular restriction we are interested in later in Section 6.

FIXED-POINT RESISTANCE. Fixed-point resistance is strictly more difficult for an adversary to satisfy than pre-image resistance because it requires that the adversary instead provides a vector of messages such that when iterated upon through the MD transform under F_{fk}, it outputs s. We will describe this in more detail in Section 7.

4 The MD transform

COMPRESSION FUNCTIONS. Let h be a family of functions with domain $\mathsf{h.Inp} = \mathsf{h.Bl} \times \mathsf{h.Out}$, meaning $\mathsf{h} : \mathsf{h.Keys} \times (\mathsf{h.Bl} \times \mathsf{h.Out}) \to \mathsf{h.Out}$. A point in the domain is a pair (m, c) where c, called the chaining variable, is in the range of h, and m, called a message block, is in the space $\mathsf{h.Bl}$ of message blocks. Such an h is called a compression function. For example, the compression function $\mathsf{h} = \mathsf{sha256}$ of $\mathsf{SHA256}$ has $\mathsf{sha256.Bl} = \{0,1\}^{512}$ and $\mathsf{sha256.Out} = \{0,1\}^{256}$. Its key space $\mathsf{sha256.Keys} = \{k\}$ is a singleton, where k consists of 64 32-bit strings which are the first 32 bits of the fractional parts of the cube roots of the first 64 primes [26].

```
$\mathsf{H}_{(k,\mathrm{s})}(M)$
$\mathbf{m} \leftarrow \mathsf{Split}(M)$ ; $c \leftarrow \mathrm{s}$ ; $n \leftarrow |\mathbf{m}|$
For $i = 1, \ldots, n$ do $c \leftarrow \mathsf{h}_k((\mathbf{m}[i], c))$
Return $c$
```

Figure 2: Function family $\mathsf{H} = \mathbf{MD}[\mathsf{h}, \mathsf{Split}, \mathrm{S}]$ obtained by applying the MD transform to compression function h, splitting function Split and space S of initial vectors.

<u>SPLITTING FUNCTIONS.</u> Let $\mathsf{Split} : \mathsf{Split.Inp} \to \mathsf{Split.Bl}^*$ be a function that takes a message $M \in \mathsf{Split.Inp}$ and returns a vector $\mathbf{m} = \mathsf{Split}(M)$ over a set $\mathsf{Split.Bl}$. We require that this function is injective, and there is an inverse $\mathsf{Split}^{-1} : \mathsf{Split.Bl}^* \to \mathsf{Split.Inp} \cup \{\perp\}$ such that $\mathsf{Split}^{-1}(\mathbf{m}) = M$ if $\mathbf{m} = \mathsf{Split}(M)$ and $\perp$ otherwise. We call Split a splitting function. The domain $\mathsf{Split.Inp}$ is expected to be a large set, usually all strings of length up to some very high maximum. In usage, $\mathsf{Split.Bl} = \mathsf{h.Bl}$ will be the set of message blocks for a compression function.

<u>THE MD TRANSFORM.</u> Let $\mathsf{h} : \mathsf{h.Keys} \times (\mathsf{h.Bl} \times \mathsf{h.Out}) \to \mathsf{h.Out}$ be a compression function. Let $\mathsf{Split} : \mathsf{Split.Inp} \to \mathsf{h.Bl}^*$ be a splitting function whose range $\mathsf{Split.Bl}^*$, as the notation indicates, is $\mathsf{h.Bl}^*$. Let $\mathrm{S} \subseteq \mathsf{h.Out}$ be a set of *starting points*, also called *initial vectors*. The MD transform $\mathbf{MD}[\mathsf{h}, \mathsf{Split}, \mathrm{S}]$ associates to them the family of functions H that is defined as follows. Let $\mathsf{H.Inp} = \mathsf{Split.Inp}$ be the set of messages that are possible inputs to the splitting function. Let $\mathsf{H.Out} = \mathsf{h.Out}$. Let $\mathsf{H.Keys} = \mathsf{h.Keys} \times \mathrm{S}$, so that a key for H is a pair (k, s) consisting of a key k for the compression function and a particular starting point (initial vector) $\mathrm{s} \in \mathrm{S}$. Then H is specified in Fig. 2.

<u>SPLITTING IN SHA.</u> Our rendition of MD generalizes prior ones, both from the literature [24, 15, 1] and from standards [26], all of which can be seen as particular choices of Split and S. We illustrate by recovering $\mathsf{SHA256}$ as $\mathbf{MD}[\mathsf{sha256}, \mathsf{SplitSha}_{(\mu,\mathfrak{e})}, \{\mathrm{s}\}]$ for choices of the components that we now specify. The compression function $\mathsf{sha256} : \{k\} \times (\{0,1\}^{512} \times \{0,1\}^{256}) \to \{0,1\}^{256}$ is of course the compression function of $\mathsf{SHA256}$ as per [26], with k the $64 \cdot 32$ bit key discussed above. The starting point s, as specified in [26], is a 256-bit string, viewed as 8 32-bit blocks which are the first 32 bits of the square roots of the first 8 primes. We define $\mathsf{SplitSha}_{(\mu,\mathfrak{e})}$ as the general splitting function for the SHA function families: $\mathsf{SHA1}$, $\mathsf{SHA256}$, $\mathsf{SHA512}$. It is parameterized by μ, the block length, and $\mathfrak{e}$, the length of the encoding of the message length. These values are shown for each SHA function in Fig. 4. Specifically for $\mathsf{SHA256}$, $\mu = 512$ and $\mathfrak{e} = 64$. To define $\mathsf{SplitSha}_{(\mu,\mathfrak{e})}$, first define function $\mathsf{pad}_{(\mu,\mathfrak{e})}$ to take as input an integer L, with $0 \le L < 2^{\mathfrak{e}}$, and return $1 \| 0^{\ell} \| \langle L \rangle_{\mathfrak{e}}$, where $\langle L \rangle_{\mathfrak{e}}$ is an $\mathfrak{e}$-bit encoding of L, and $\ell \ge 0$ is the smallest integer such that $L + \mathfrak{e} + 1 + \ell$ is a multiple of μ. Let $\mathsf{SplitSha.Inp}$ be the set of all strings of length at most $2^{\mathfrak{e}}$, and let $\mathsf{SplitSha.Bl} = \{0,1\}^{\mu}$. The function $\mathsf{SplitSha} : \mathsf{SplitSha.Inp} \to \mathsf{SplitSha.Bl}^*$, on input M, lets $L = |M|$ be the length of M, and lets $X = M \| \mathsf{pad}_{(\mu,\mathfrak{e})}(L)$. Note that the length of string X is a multiple of μ. Let $n \leftarrow |X|/\mu$, and let $\mathbf{m}[i] = X[1 + \mu(i-1)..\mu i]$ be the μ bit-block consisting of bits $1 + \mu(i-1)$ through μi of X, for $1 \le i \le n$. Then $\mathsf{SplitSha}(M)$ returns $\mathbf{m}$, which is a vector over $\{0,1\}^{\mu}$.

5 CR preservation of MD

Here we recall the classical problem of showing collision resistance of the hash function $\mathsf{H} = \mathbf{MD}[\mathsf{h}, \mathsf{Split}, \mathrm{S}]$ assuming *only* collision resistance of the compression function h. As noted in the introduction, several works have noted that suffix-freeness of Split is sufficient for this purpose [18, 19, 1, 5]. For completeness, we will provide a formal claim together with the (easy) proof in our setting.

```
SplitSha_(μ,𝔢)(M)                          pad_(μ,𝔢)(L)
  $L \leftarrow |M|$                         $\ell \leftarrow (\mu - \mathfrak{e} - 1 - L) \mod \mu$
  $X \leftarrow M \| \mathsf{pad}_{(\mu,\mathfrak{e})}(L)$ ; $n \leftarrow |X|/\mu$    Return $1\|0^{\ell}\|\langle L\rangle_{\mathfrak{e}}$
  For $1 \le i \le n$ do
      $\mathbf{m}[i] \leftarrow X[1+\mu(i-1)..\mu i]$
  Return $\mathbf{m}$
```

Figure 3: SplitSha and pad, the splitting function and padding function, respectively, of the SHA function families. They are parameterized by μ, the block length, and $\mathfrak{e}$, the length of the encoding of the message length.

Function	μ	σ	$\mathfrak{e}$
SHA1	512	160	64
SHA256	512	256	64
SHA512	1024	512	128

Figure 4: Choices of parameters across different hash functions.

We then expand on this knowledge to establish a novel result that the property of being suffix-free is *precisely* the property required for this proof; it is a necessary condition in addition to being sufficient. To demonstrate this, we construct, for any splitting function which is not suffix-free, a compression function that is collision resistant in isolation, but for which the result of applying the MD transform is not collision resistant.

SUFFIX-FREENESS. Let $x, y \in D^*$ be vectors over a set D. We say that x is a suffix of y, written $y \sqsupseteq x$, if there exists a vector $z \in D^*$ such that $y = z\|x$. (The notation $y \sqsupseteq x$ is intended to visualize x being the right-hand side of y.) For example, $(10, 11)$ is a suffix of $(00, 11, 10, 11)$, namely $(00, 11, 10, 11) \sqsupseteq (10, 11)$, by letting $z = (00, 11)$. However, $(01, 11)$ is not a suffix of $(00, 11, 10, 11)$, namely $(00, 11, 10, 11) \not\sqsupseteq (01, 11)$. We say that splitting function $\mathsf{Split}\colon \mathsf{Split.Inp} \to \mathsf{Split.Bl}^*$ is *suffix-free* if for any two distinct messages $M_1, M_2 \in \mathsf{Split.Inp}$ we have $\mathsf{Split}(M_1) \not\sqsupseteq \mathsf{Split}(M_2)$, that is, $\mathsf{Split}(M_2)$ is not a suffix of $\mathsf{Split}(M_1)$.

SUFFIX-FREENESS OF SplitSha. We discussed above how SHA256 is underlain by a particular splitting function that we defined and called $\mathsf{SplitSha}_{(\mu,\mathfrak{e})}$. Here we show that this function is suffix-free to provide an example of a suffix-free scheme.

Proposition 5.1 *The function* $\mathsf{SplitSha}_{(\mu,\mathfrak{e})}$ *is suffix-free.*

Proof: (of *Proposition* 5.1) Let $M_1, M_2 \in \mathsf{SplitSha}_{(\mu,\mathfrak{e})}.\mathsf{Inp}$ be distinct. Consider when $|M_1| \neq |M_2|$. Then the last blocks of vectors $\mathsf{SplitSha}_{(\mu,\mathfrak{e})}(M_1)$ and $\mathsf{SplitSha}_{(\mu,\mathfrak{e})}(M_2)$ are, respectively, $\mathsf{pad}_{(\mu,\mathfrak{e})}(|M_1|)$ and $\mathsf{pad}_{(\mu,\mathfrak{e})}(|M_2|)$. But then $\langle |M_1| \rangle_{\mathfrak{e}} \neq \langle |M_2| \rangle_{\mathfrak{e}}$, which implies $\mathsf{pad}_{(\mu,\mathfrak{e})}(|M_1|) \neq \mathsf{pad}_{(\mu,\mathfrak{e})}(|M_2|)$ and so neither vector can be a suffix of the other.

We now consider the case when $|M_1| = |M_2|$. In this case we have that $|\mathsf{SplitSha}_{(\mu,\mathfrak{e})}(M_1)| = |\mathsf{SplitSha}_{(\mu,\mathfrak{e})}(M_2)|$. Then in order for $\mathsf{SplitSha}_{(\mu,\mathfrak{e})}(M_1) \sqsupseteq \mathsf{SplitSha}_{(\mu,\mathfrak{e})}(M_2)$ or the opposite to hold it must be that $\mathsf{SplitSha}_{(\mu,\mathfrak{e})}(M_1) = \mathsf{SplitSha}_{(\mu,\mathfrak{e})}(M_2)$. Notice that $\mathsf{SplitSha}_{(\mu,\mathfrak{e})}(M)$ prepends the message M to its output. Since $M_1 \neq M_2$, we have that $\mathsf{SplitSha}_{(\mu,\mathfrak{e})}(M_1) \neq \mathsf{SplitSha}_{(\mu,\mathfrak{e})}(M_2)$ and so $\mathsf{SplitSha}_{(\mu,\mathfrak{e})}(M_1) \not\sqsupseteq \mathsf{SplitSha}_{(\mu,\mathfrak{e})}(M_2)$ as required. ▌

THE STRUCTURE OF MD COLLISIONS. We now proceed to a simple lemma about the structure of collisions in the MD transform. This will be used for our proof that the MD transform preserves collision resistance when the splitting function is suffix-free. This lemma argues the correctness of an algorithm $B_{\rm cr}$ to formalize the observation that if M_1, M_2 form a collision for the MD transform with a suffix-free splitting function, then by examining the computation of the hash function on these inputs we can easily find a collision for the underlying compression function.

Lemma 5.2 *Let* h *be a compression function, let* Split *be a splitting function with* $\mathsf{Split.Bl} = \mathsf{h.Bl}$ *and let* $\mathrm{S} \subseteq \mathsf{h.Out}$ *be a set of possible starting points. Let* $\mathsf{H} = \mathbf{MD}[\mathsf{h}, \mathsf{Split}, \mathrm{S}]$ *be the hash function associated to these components via the MD transform of Fig. 2. Let* $k \in \mathsf{h.Keys}, \mathrm{s} \in \mathrm{S}$*. Suppose* $M_1, M_2 \in \mathsf{Split.Inp}$ *are a pair of distinct messages satisfying (1)* $\mathsf{Split}(M_1) \not\sqsupseteq \mathsf{Split}(M_2)$ *and* $\mathsf{Split}(M_2) \not\sqsupseteq \mathsf{Split}(M_1)$ *and (2)* M_1, M_2 *are a collision for* $\mathsf{H}_{(k,\mathrm{s})}$*. Then, on inputs* $(k, \mathrm{s}), M_1, M_2$*, algorithm* $B_{\rm cr}$ *of Fig. 5 returns* $(x_1, c_1), (x_2, c_2)$ *that form a collision for* h_k*.*

The algorithm $B_{\rm cr}$ simply creates vectors of the chaining variables and message blocks used when computing $\mathsf{H}_{(k,\mathrm{s})}(M_1)$ and $\mathsf{H}_{(k,\mathrm{s})}(M_2)$. The it iterates backwards until it finds the first place that these differ between the two computations. The following proof shows that these will necessarily be a collision for h_k.

Proof: (of Lemma 5.2) From algorithm $B_{\rm cr}$, let $\mathbf{m}_1 = \mathsf{Split}(M_1)$, $\mathbf{m}_2 = \mathsf{Split}(M_2)$, $n_1 = |\mathbf{m}_1|$, and $n_2 = |\mathbf{m}_2|$. First $B_{\rm cr}$ computes the vectors of chaining variables, $\mathbf{c}_1$ and $\mathbf{c}_2$ as shown in the pseudocode. Assume (without loss of generality) that $n_1 \geq n_2$, i.e. that $|\mathbf{m}_1| \geq |\mathbf{m}_2|$. Since M_1, M_2 are a collision for $\mathsf{H}_{(k,\mathrm{s})}$, we have $\mathsf{H}_{(k,\mathrm{s})}(M_1) = \mathsf{H}_{(k,\mathrm{s})}(M_2)$. Because $\mathbf{m}_1 \not\sqsupseteq \mathbf{m}_2$ and $\mathbf{m}_2 \not\sqsupseteq \mathbf{m}_1$, there must exist $i \in \{0, \ldots, n_2 - 1\}$ such that $(\mathbf{m}_1[n_1 - i], \mathbf{c}_1[n_1 - i]) \neq (\mathbf{m}_2[n_2 - i], \mathbf{c}_2[n_2 - i])$. Let j represent the minimal such value. Then it will hold that $\mathbf{c}_1[n_1 - j + 1] = \mathbf{c}_2[n_2 - j + 1]$. Thus, the pair $(\mathbf{m}_1[n_1 - j], \mathbf{c}_1[n_1 - j])$, $(\mathbf{m}_2[n_2 - j], \mathbf{c}_2[n_2 - j])$ returned by $B_{\rm cr}$ will form a collision for h_k. ▌

SUFFIX-FREENESS PRESERVES CR. Finally, we show a reduction from the collision resistance of the hash function $\mathsf{H} = \mathbf{MD}[\mathsf{h}, \mathsf{Split}, S]$ to the collision resistance of the compression function h when using a suffix-free splitting function Split.

Theorem 5.3 *Let* h *be a compression function, let* Split *be a suffix-free splitting function with* $\mathsf{Split.Bl} = \mathsf{h.Bl}$ *and let* $\mathrm{S} \subseteq \mathsf{h.Out}$ *be a set of possible starting points. Let* $\mathsf{H} = \mathbf{MD}[\mathsf{h}, \mathsf{Split}, \mathrm{S}]$ *be the hash function associated to these components via the MD transform of Fig. 2. Given an adversary* $\mathcal{A}_{\mathsf{H}}$*, we can build* $\mathcal{A}_{\mathsf{h}}$ *(specified in the proof) using* $B_{\rm cr}$ *from Fig. 5. Then*

$$\mathbf{Adv}^{\rm cr}_{\mathsf{H}}(\mathcal{A}_{\mathsf{H}}) \leq \mathbf{Adv}^{\rm cr}_{\mathsf{h}}(\mathcal{A}_{\mathsf{h}}). \tag{1}$$

The time complexity of $\mathcal{A}_{\mathsf{h}}$ *is the sum of the time complexities of* $\mathcal{A}_{\mathsf{H}}$ *and* $B_{\rm cr}$*. The memory complexity of* $\mathcal{A}_{\mathsf{h}}$ *is the maximum of the memory complexity of* $\mathcal{A}_{\mathsf{H}}$ *and the memory complexity of* $B_{\rm cr}$*.*

In Section 9, we revisit this and other reductions to give alternative reductions that are more memory-efficient [4].

Proof: (of *Theorem* 5.3) The adversary $\mathcal{A}_{\mathsf{h}}$ is defined as follows.

```
Adversary $\mathcal{A}_{\mathsf{h}}(k, \varepsilon)$
$\mathrm{s} \leftarrow\$ \mathrm{S}$ ; $(M_1, M_2) \leftarrow \mathcal{A}_{\mathsf{H}}((k, \mathrm{s}), \varepsilon)$
Return $B_{\rm cr}((k, \mathrm{s}), M_1, M_2)$
```

```
Algorithm $B_{\rm cr}((k,\mathrm{s}), M_1, M_2)$
$\mathbf{m}_1 \leftarrow \mathsf{Split}(M_1)$ ; $\mathbf{m}_2 \leftarrow \mathsf{Split}(M_2)$
$n_1 \leftarrow |\mathbf{m}_1|$ ; $n_2 \leftarrow |\mathbf{m}_2|$
$\mathbf{c}_1[1] \leftarrow \mathrm{s}$ ; $\mathbf{c}_2[1] \leftarrow \mathrm{s}$
For $i = 1, \ldots, n_1$ do $\mathbf{c}_1[i+1] \leftarrow \mathsf{h}_k((\mathbf{m}_1[i], \mathbf{c}_1[i]))$
For $i = 1, \ldots, n_2$ do $\mathbf{c}_2[i+1] \leftarrow \mathsf{h}_k((\mathbf{m}_2[i], \mathbf{c}_2[i]))$
$n \leftarrow \min(n_1, n_2)$
For $i = 0, \ldots, n-1$ do
   $(x_1, c_1) \leftarrow (\mathbf{m}_1[n_1 - i], \mathbf{c}_1[n_1 - i])$
   $(x_2, c_2) \leftarrow (\mathbf{m}_2[n_2 - i], \mathbf{c}_2[n_2 - i])$
   If $(x_1, c_1) \neq (x_2, c_2)$ then return $((x_1, c_1), (x_2, c_2))$
Return $\bot$
```

Figure 5: Algorithm $B_{\rm cr}$ for Lemma 5.2 and Theorem 5.3.

It is clear that the time and memory complexity of adversary $\mathcal{A}_{\mathsf{h}}$ are as stated in the theorem. Note the memory is the maximum of $\mathcal{A}_{\mathsf{H}}$ and $B_{\rm cr}$ because $\mathcal{A}_{\mathsf{h}}$ can re-use the memory of $\mathcal{A}_{\mathsf{H}}$ when running $B_{\rm cr}$.

Let $k \in \mathsf{h.Keys}, \mathrm{s} \in \mathrm{S}$ be the values sampled when $\mathcal{A}_{\mathsf{h}}$ is executed and $M_1, M_2 \in \mathsf{Split.Inp}$ be the values returned by $\mathcal{A}_{\mathsf{H}}$. Suppose they form a collision for $\mathsf{H}_{(k,\mathrm{s})}$.

Then we have $\mathsf{Split}(M_1) \not\sqsupseteq \mathsf{Split}(M_2)$ and $\mathsf{Split}(M_2) \not\sqsupseteq \mathsf{Split}(M_1)$ because Split is suffix-free, so they fulfill the conditions of Lemma 5.2 and $B_{\rm cr}$ is guaranteed to return a collision for h_k. As an immediate result, Equation (1) holds, completing the proof. ▌

Necessity. We can now complete the picture for splitting functions when assuming the compression function is collision resistant by showing that the suffix-free restriction is *precisely* the correct restriction on the splitting function. In particular, we will establish that Split being suffix-free is a necessary condition for proving that $\mathbf{MD}[\mathsf{h}, \mathsf{Split}, \mathrm{S}]$ is secure under the assumption that h is collision resistant, in addition to a sufficient one.

Given an arbitrary splitting function Split and a pair of inputs M_1, M_2 such that $\mathsf{Split}(M_1) \sqsupseteq \mathsf{Split}(M_2)$, we construct a compression function h which is collision resistant (from another function which we assume to be collision resistant), but for which the pair M_1, M_2 is a collision for $\mathbf{MD}[\mathsf{h}, \mathsf{Split}, \mathrm{S}]$ (with high probability over the choice of $\mathrm{s} \in \mathrm{S}$ in the case that $\mathrm{S} = \mathsf{h.Out}$).

For simplicity, we will first consider the simpler case when S consists of a single, fixed value s on which our choice of compression function can depend. For this case, we can directly construct the compression function so that when chained with the starting value s on the blocks contained uniquely in $\mathsf{Split}(M_1)$ but not those in $\mathsf{Split}(M_2)$, it "loops" back to s. We then extend this technique to cover the case when S is some larger set from which s is sampled randomly.

It will be convenient to describe our results in terms of the MD transform applied to messages that have already been split into blocks. For any set Bl we let $\mathsf{I} : \mathsf{Bl}^* \to \mathsf{Bl}^*$ be the splitting function which simply outputs its input unchanged. For some compression function h and set S, let $\mathsf{H}^{\mathsf{I}} = \mathbf{MD}[\mathsf{h}, \mathsf{I}, \mathrm{S}]$. We will informally say that h loops on $(\mathrm{s}, \mathbf{u})$ if $\mathsf{H}^{\mathsf{I}}_{(k,\mathrm{s})}(\mathbf{u}) = \mathrm{s}$ for all $k \in \mathsf{h.Keys}$. The following lemma observes that if h loops on $(\mathrm{s}, \mathbf{u})$ and $\mathsf{Split}(M_1) = \mathbf{u} \| \mathsf{Split}(M_2)$, then the pair M_1, M_2 is a collision for $\mathbf{MD}[\mathsf{h}, \mathsf{Split}, \mathrm{S}]$.

Lemma 5.4 *Let* $\mathsf{h} : \mathsf{h.Keys} \times (\mathsf{h.Bl} \times \mathsf{h.Out}) \to \mathsf{h.Out}$ *be a compression function,* $\mathsf{Split} : \mathsf{Split.Inp} \to \mathsf{h.Bl}^*$ *be a splitting function, and* $\mathrm{S} \subseteq \mathsf{h.Out}$ *be a set of starting points. Let* $\mathsf{H} = \mathbf{MD}[\mathsf{h}, \mathsf{Split}, \mathrm{S}]$ *be the hash function associated to these components via the MD transform of Fig. 2. Let* I *be the* *splitting function described above and* $\mathsf{H}^{\mathsf{I}} = \mathbf{MD}[\mathsf{h}, \mathsf{I}, \mathrm{S}]$ *be the corresponding hash function obtained via the MD transform. Suppose* $M_1, M_2 \in \mathsf{Split.Inp}$ *are a pair of distinct messages satisfying* $\mathsf{Split}(M_1) \sqsupseteq \mathsf{Split}(M_2)$*. Let* $\mathbf{u}$ *be the vector for which* $\mathsf{Split}(M_1) = \mathbf{u} \| \mathsf{Split}(M_2)$*. For any choice of* $(k, \mathrm{s}) \in \mathsf{H.Keys}$*, if* $\mathsf{H}^{\mathsf{I}}_{(k,\mathrm{s})}(\mathbf{u}) = \mathrm{s}$ *then* M_1, M_2 *is a collision for* $\mathsf{H}_{(k,\mathrm{s})}$*.*

Proof: (of Lemma 5.4) First note that for any vectors $\mathbf{m}, \mathbf{y} \in \mathsf{BI}^*$, $\mathsf{H}^{\mathsf{I}}_{(k,\mathrm{s})}(\mathbf{m}\|\mathbf{y}) = \mathsf{H}^{\mathsf{I}}_{(k,\mathrm{s}')}(\mathbf{y})$ where $\mathrm{s}' = \mathsf{H}^{\mathsf{I}}_{(k,\mathrm{s})}(\mathbf{m})$. This is a simple observation from the code of the MD transform shown in Fig. 2. The chaining variable $\mathbf{c}[|\mathbf{m}| + 1]$ obtained during the computation of $\mathsf{H}^{\mathsf{I}}_{(k,\mathrm{s})}(\mathbf{m}\|\mathbf{y})$ would be the output of $\mathsf{H}^{\mathsf{I}}_{(k,\mathrm{s}')}(\mathbf{m})$. The rest of the computation then exactly mirrors H^{I} applied to $\mathbf{y}$ with $\mathbf{c}[|\mathbf{m}| + 1]$ serving the role of the starting point.

Using this observation, the proof is straightforward. We can rewrite H on input M_1 as follows

$$\begin{aligned}
\mathsf{H}_{(k,\mathrm{s})}(M_1) &= \mathsf{H}^{\mathsf{I}}_{(k,\mathrm{s})}(\mathbf{u}\|\mathsf{Split}(M_2)) \\
&= \mathsf{H}^{\mathsf{I}}_{(k,\mathrm{s}')}(\mathsf{Split}(M_2)) \\
&= \mathsf{H}_{(k,\mathrm{s}')}(M_2)
\end{aligned}$$

where $\mathrm{s}' = \mathsf{H}^{\mathsf{I}}_{(k,\mathrm{s})}(\mathbf{u})$. From our assumption, this equals s. Thus $\mathsf{H}_{(k,\mathrm{s})}(M_1) = \mathsf{H}_{(k,\mathrm{s})}(M_2)$. ▌

First we will handle the case when the s used for the MD transform is an a priori fixed value.

Proposition 5.5 *Let* Split *be a splitting function and* $M_1, M_2 \in \mathsf{Split.Inp}$ *satisfy* $\mathsf{Split}(M_1) \sqsupseteq \mathsf{Split}(M_2)$*. Let* $\mathbf{u}$ *be the vector for which it holds that* $\mathsf{Split}(M_1) = \mathbf{u}\|\mathsf{Split}(M_2)$*. Let* $b \in \mathbb{N}$ *and* $a = b + |\mathbf{u}|$*. Let* f *be a family of functions with* $\mathsf{f.Inp} = \mathsf{Split.BI} \times \mathbb{Z}_a$ *and* $\mathsf{f.Out} = \mathbb{Z}_b$*. Then we can build a compression function* $\mathsf{g}^{\mathbf{u}}$ *(shown in Fig. 6, with* $\mathsf{g.Inp} = \mathsf{f.Inp}$ *and* $\mathsf{g.Out} = \mathbb{Z}_a$*) such that for all adversaries* $\mathcal{A}$*, we have* $\mathbf{Adv}^{\mathrm{cr}}_{\mathsf{g}^{\mathbf{u}}}(\mathcal{A}) \leq \mathbf{Adv}^{\mathrm{cr}}_{\mathsf{f}}(\mathcal{A})$*. Furthermore, letting* $\mathsf{G} = \mathbf{MD}[\mathsf{g}^{\mathbf{u}}, \mathsf{Split}, \{0\}]$*, we can build an efficient adversary* $\mathcal{B}$ *(shown in Fig. 6) such that*

$$\mathbf{Adv}^{\mathrm{cr}}_{\mathsf{G}}(\mathcal{B}) = 1.$$

The compression function $\mathsf{g}^{\mathbf{u}}$ above is specifically defined in a contrived way so that it loops on $(0, \mathbf{u})$ and thus M_1, M_2 is a collision for the MD transform.

In the above we fixed the starting point s to 0 and the set of chaining variables $\mathsf{g}^{\mathbf{u}}.\mathsf{Out}$ to $\mathbb{Z}_a$. This is without loss of generality because the lemma can easily be extended to any reasonable choice of $\mathsf{g}^{\mathbf{u}}.\mathsf{Out}$ and fixed $\mathrm{s} \in \mathsf{g}^{\mathbf{u}}.\mathsf{Out}$ by using an efficiently computable and invertible mapping $e(\cdot) : \mathsf{g}^{\mathbf{u}}.\mathsf{Out} \to \mathbb{Z}_{|\mathsf{g}^{\mathbf{u}}.\mathsf{Out}|}$ which satisfies $e(\mathrm{s}) = 0$.

Proof: (of Proposition 5.5) We will first show that any collision for $\mathsf{g}^{\mathbf{u}}$ is also a collision for f by proving that if $\mathsf{g}^{\mathbf{u}}_k((m, c)) = \mathsf{g}^{\mathbf{u}}_k((m', c'))$ it either holds that $(m, c) = (m', c')$ or that $\mathsf{f}_k((m, c)) = \mathsf{f}_k((m', c'))$. As such, suppose $\mathsf{g}^{\mathbf{u}}_k((m, c)) = \mathsf{g}^{\mathbf{u}}_k((m', c'))$.

Note that the first return statement of $\mathsf{g}^{\mathbf{u}}$ always outputs a value less than $|\mathbf{u}|$ while the second always outputs a value greater than $|\mathbf{u}|$. We can consider these two cases separately.

Let us first suppose that $\mathsf{g}^{\mathbf{u}}_k((m, c)) < |\mathbf{u}|$. This then means that $c + 1 = c' + 1 \pmod{|\mathbf{u}|}$, so c and c' must be the same (because the condition of the if statement guarantees that both are less than $|\mathbf{u}|$). The if statement inside $\mathsf{g}^{\mathbf{u}}$ must evaluate to true for both pairs, so we have $m = \mathbf{u}[c + 1] = \mathbf{u}[c' + 1] = m'$ and so $(m, c) = (m', c')$.

Now consider the other case, that $\mathsf{g}^{\mathbf{u}}_k((m, c)) \geq |\mathbf{u}|$. Then this must mean that $\mathsf{f}_k((m, c)) + |\mathbf{u}|$ and $\mathsf{f}_k((m', c')) + |\mathbf{u}|$ are the same and so $\mathsf{f}_k((m, c)) = \mathsf{f}_k((m', c'))$.

```
$\mathsf{g}^{\mathbf{u}}_k((m,c))$
If $(c < |\mathbf{u}|)$ and $(m = \mathbf{u}[c+1])$ then
    Return $c + 1 \bmod |\mathbf{u}|$
Return $\mathsf{f}_k((m,c)) + |\mathbf{u}|$
```

```
Adversary $\mathcal{B}(k, \mathrm{s})$
Return $(M_1, M_2)$
```

```
$\mathsf{h}^{\mathbf{u}}_k((m,c))$
$(b_c, q_c, r_c) \leftarrow c$
If $(b_c = 0)$ and $(m = \mathbf{u}[r_c+1])$ then
    Return $(0, q_c, (r_c + 1 \bmod |\mathbf{u}|))$
$(q, r) \leftarrow \mathsf{f}_k((m,c))$
Return $(1, q, r)$
```

Figure 6: **Left:** Compression function used in Proposition 5.5. **Middle:** Adversary used in Proposition 5.5, Theorem 5.6, and Proposition 6.1. **Right:** Compression function used in Theorem 5.6.

Because any collision for $\mathsf{g}^{\mathbf{u}}$ is also a collision for f, for any adversary $\mathcal{A}$ it must hold that $\mathbf{Adv}^{\mathrm{cr}}_{\mathsf{g}^{\mathbf{u}}}(\mathcal{A}) \le \mathbf{Adv}^{\mathrm{cr}}_{\mathsf{f}}(\mathcal{A})$.

To prove that $\mathcal{B}$ has advantage 1, we will make use of Lemma 5.4 by showing that $\mathsf{H}^{\mathsf{l}}_{(k,0)}(\mathbf{u}) = 0$ where H^{l} is defined as in the lemma. Let $\mathbf{c}$ be the vector of values that would be obtained in the computation of $\mathsf{H}^{\mathsf{l}}_{(k,0)}(\mathbf{u})$; that is, let $\mathbf{c}[1] = 0$ and $\mathbf{c}[i+1] = \mathsf{g}^{\mathbf{u}}_k((\mathbf{u}[i], \mathbf{c}[i]))$ for $i = 1, \ldots, |\mathbf{u}|$.

Following the code of $\mathsf{g}^{\mathbf{u}}$ we can then see that its if statement will always evaluate to true in this computation and so $\mathbf{c}[i+1] = \mathbf{c}[i] + 1 \pmod{|\mathbf{u}|}$ holds for all i. Consequently, $\mathbf{c}[i] = i$ for $i = 1, \ldots, |\mathbf{u}|$ and then $\mathbf{c}[|\mathbf{u}|+1] = 0$. The latter is the value returned by H^{l} so $\mathsf{H}^{\mathsf{l}}_{(k,0)}(\mathbf{u}) = 0$ and the pair M_1, M_2 is a collision for H. It follows that the advantage of $\mathcal{B}$ is exactly 1. ▌

The lemma above might seem somewhat contrived, because we allowed our compression function g to depend on the starting point used for the MD transform. This makes it, in some senses, a weak result and one might naturally wonder if this dependency is necessary for the result. It is not. At the cost of some added complexity and lost success probability for $\mathcal{B}$, we can extend this to the case when s is randomly chosen from some set instead of fixed.

Theorem 5.6 *Let $a \in \mathbb{N}$. Let* Split *be a splitting function. Suppose* $\mathsf{Split}(M_1) \sqsupseteq \mathsf{Split}(M_2)$ *and in particular* $\mathsf{Split}(M_1) = \mathbf{u} \| \mathsf{Split}(M_1)$*. Let* f *be a family of functions with* $\mathsf{f.Inp} = \mathsf{Split.Bl} \times (\mathbb{Z}_2 \times \mathbb{Z}_a \times \mathbb{Z}_{|\mathbf{u}|})$ *and* $\mathsf{f.Out} = \mathbb{Z}_a \times \mathbb{Z}_{|\mathbf{u}|}$*. Now let* $\mathsf{h}^{\mathbf{u}} : \mathsf{Split.Bl} \times (\mathbb{Z}_2 \times \mathbb{Z}_a \times \mathbb{Z}_{|\mathbf{u}|}) \to \mathbb{Z}_2 \times \mathbb{Z}_a \times \mathbb{Z}_{|\mathbf{u}|}$ *be the compressions function shown in Fig. 6. For all adversaries $\mathcal{A}$, it holds that* $\mathbf{Adv}^{\mathrm{cr}}_{\mathsf{h}^{\mathbf{u}}}(\mathcal{A}) \le \mathbf{Adv}^{\mathrm{cr}}_{\mathsf{f}}(\mathcal{A})$*. Furthermore, letting* $\mathrm{S} = \mathsf{h}^{\mathbf{u}}.\mathsf{Out}$ *and* $\mathsf{H} = \mathbf{MD}[\mathsf{h}^{\mathbf{u}}, \mathsf{Split}, \mathrm{S}]$*, we can build an efficient adversary $\mathcal{B}$ (shown in Fig. 6) satisfying,*

$$\mathbf{Adv}^{\mathrm{cr}}_{\mathsf{H}}(\mathcal{B}) \ge 1/(2|\mathbf{u}|).$$

The compression function $\mathsf{h}^{\mathbf{u}}$ above is specifically designed so that it loops on $\mathrm{s}, \mathbf{u}$ for any s of the form $(0, q, 0)$, giving the desired collision between M_1 and M_2 with the specified probability over the random choice of s

Again, this theorem can be extended to cover any reasonable choice of $\mathsf{h}^{\mathbf{u}}.\mathsf{Out}$. One can first map $\mathsf{h}^{\mathbf{u}}.\mathsf{Out}$ to $\mathbb{Z}_{|\mathsf{h}^{\mathbf{u}}.\mathsf{Out}|}$ analogously to earlier. Then from $c \in \mathbb{Z}_{|\mathsf{h}^{\mathbf{u}}.\mathsf{Out}|}$ one can obtain the tuple (b_c, q_c, r_c) via $b_c \leftarrow c \bmod 2$, $y \leftarrow \lfloor c/2 \rfloor$, $r_c \leftarrow y \bmod |\mathbf{u}|$, and $q_c \leftarrow \lfloor y/2 \rfloor$. There are technical details to be considered regarding the fact that 2 and $|\mathbf{u}|$ may not be divisors of $|\mathsf{h}^{\mathbf{u}}.\mathsf{Out}|$ and that S may not be "nicely" distributed in $\mathbb{Z}_{|\mathsf{h}^{\mathbf{u}}.\mathsf{Out}|}$, but for any reasonable choice of $\mathsf{h}^{\mathbf{u}}.\mathsf{Out}$ and S this should not be an issue.

For our theorems we have assumed that we were given a pair M_1, M_2 such that $\mathsf{Split}(M_1) \sqsupseteq \mathsf{Split}(M_2)$. It is not difficult to come up with (contrived) splitting functions which are not suffix-free, but for which we believe it is computationally difficult to find such a pair. We chose our formalization that M_1 and M_2 are a priori known because, for specific, prior splitting functions, either they were suffix-free, or it was trivially easy to find M_1 and M_2 violating suffix-freeness. An alternative way to address this would be to make suffix-freeness a computational condition, and then say that, given an adversary returning M_1 and M_2 violating suffix-freeness with high probability, we build our compression function and adversary. (Of course, one might then ask about finding the adversary, analogous to keyless collision resistance, but the philosophical position would at least seem on par with prior ones.)

Proof: (of Theorem 5.6) The basic structure of this proof closely follows that of the proof for Proposition 5.5. Throughout this proof for a string c we will let b_c, q_c, r_c denote the corresponding values used by $\mathsf{h}^{\mathbf{u}}$ on input (m, c) for some c.

To start, we will show that any collision for $\mathsf{h}^{\mathbf{u}}$ is also a collision for f by proving that if $\mathsf{h}^{\mathbf{u}}_k((m,c)) = \mathsf{h}^{\mathbf{u}}_k((m',c'))$, then it either holds that $(m,c) = (m',c')$ or that $\mathsf{f}_k((m,c)) = \mathsf{f}_k((m',c'))$. As such, suppose $\mathsf{h}^{\mathbf{u}}_k((m,c)) = \mathsf{h}^{\mathbf{u}}_k((m',c'))$.

Note that the first return statement of $\mathsf{h}^{\mathbf{u}}$ always outputs a tuple whose first element is 1 while the second always outputs a tuple whose first element is 0. We will consider these two cases separately. Let $y = \mathsf{h}^{\mathbf{u}}_k((m,c))$ and $y' = \mathsf{h}^{\mathbf{u}}_k((m',c'))$.

Let us first suppose that $y[1] = 0$. This means that $q_c = q_{c'}$ and $r_c + 1 = r_{c'} + 1 \pmod{|\mathbf{u}|}$. The latter implies $r_c = r_{c'}$, because both are always less than $|\mathbf{u}|$. The if statement in $\mathsf{h}^{\mathbf{u}}$ must have evaluated to true on both inputs so we have $b_c = 0 = b_{c'}$ and $m = \mathbf{u}[r_c + 1] = \mathbf{u}[r_{c'} + 1] = m'$. Putting this all together, we have shown that $(m,c) = (m',c')$.

Now consider the other case when $y[1] = 1$. Then we have that $\mathsf{f}_k((m,c)) = \mathsf{f}_k((m',c'))$.

Because any collision for $\mathsf{h}^{\mathbf{u}}$ is also a collision for f, for any adversary $\mathcal{A}$ it must hold that $\mathbf{Adv}^{\mathrm{cr}}_{\mathsf{h}}(\mathcal{A}) \leq \mathbf{Adv}^{\mathrm{cr}}_{\mathsf{f}}(\mathcal{A})$.

To prove our statement about the advantage of $\mathcal{B}$ we will make use of Lemma 5.4 andlower bound the probability that $\mathsf{H}^{\mathsf{I}}_{(k,\mathrm{s})}(\mathbf{u}) = \mathrm{s}$ over the random choice of s (where H^{I} is defined as in the lemma).

Suppose s is of the form $(0, q, 0)$ for some $q \in \mathbb{Z}_a$. Let $\mathbf{c}$ be the vector of values that would have been obtained in the computation of $\mathsf{H}^{\mathsf{I}}_{(k,\mathrm{s})}(\mathbf{u})$; that is, let $\mathbf{c}[1] = \mathrm{s}$ and $\mathbf{c}[i+1] = \mathsf{h}^{\mathbf{u}}_k((\mathbf{u}[i], \mathbf{c}[i]))$ for $i = 1, \ldots, |\mathbf{u}|$.

Following the code of $\mathsf{h}^{\mathbf{u}}$ we can see that the if statement will always return true in this computation, and so $\mathbf{c}[i] = (0, q, i)$ for $i = 1, \ldots, |\mathbf{u}|$ and then $\mathbf{c}[|\mathbf{u}| + 1] = (0, q, 0) = \mathrm{s}$. The latter is the value returned by H^{I} so $\mathsf{H}^{\mathsf{I}}_{(k,\mathrm{s})}(\mathbf{u}) = \mathrm{s}$ as desired and the pair M_1, M_2 is a collision for H.

Then the advantage of $\mathcal{B}$ is bounded by the probability that s is of the form $(0, q, 0)$ which is exactly $1/(2|\mathbf{u}|)$. ▮

6 Weakening assumptions on h

In this section we improve on the classic result that the collision resistance of h guarantees that H will be collision resistant. In particular, we will explore the possibility of weakening the assumption made of h and provide a natural, less stringent variation of collision resistance from which we are able to assure the collision resistance of h obtained via the MD transform.

Using a non-CR h. We have shown that the collision resistance of the compression function h implies the collision resistance of the hash function H obtained by the MD transform. However,

```
$\underline{\mathsf{h}^{\mathrm{s}}_k((m,c))}$
If $c[1] = \mathrm{s}[1]$ then
    Return $\mathrm{s}[1] \| \mathsf{h}'_k((m, c[2..|c|]))$
Return $\bar{\mathrm{s}}$
```

```
$\underline{\text{Adversary } \mathcal{A}_{\mathsf{h}'}(k,\varepsilon)}$
$\mathrm{s} \leftarrow\!\!\$\ \mathrm{S}$ ; $(M_1, M_2) \leftarrow \mathcal{A}_{\mathsf{H}}((k,\mathrm{s}),\varepsilon)$
$((m_1,c_1),(m_2,c_2)) \leftarrow B_{\mathrm{cr}}((k,\mathrm{s}), M_1, M_2)$
Return $((m_1, c_1[2..|c_1|]), (m_2, c_2[2..|c_2|]))$
```

Figure 7: **Left:** Compression function $\mathsf{h}^{\mathrm{s}}_k$ for Proposition 6.1, Proposition 6.2, and Proposition 6.3. **Right:** Adversary $\mathcal{A}_{\mathsf{h}'}$ for the proof of Proposition 6.3.

the collision resistance of H may not always rely on h being collision resistant. We will show by construction that H can be collision resistant even when h is not.

Let $b, c \in \mathbb{N}$. Given a compression function $\mathsf{h}' : \mathsf{h}'.\mathsf{Keys} \times (\{0,1\}^b \times \{0,1\}^c) \to \{0,1\}^c$ and some $\mathrm{s} \in \{0,1\}^{c+1}$, we construct the compression function $\mathsf{h}^{\mathrm{s}} : \mathsf{h}'.\mathsf{Keys} \times (\{0,1\}^b \times \{0,1\}^{c+1}) \to \{0,1\}^{c+1}$ shown in Fig. 7. Recall $\bar{\mathrm{s}}$ denotes the bitwise complement of s. Let Split be a suffix-free splitting function with $\mathsf{Split.Bl} = \mathsf{h.Bl}$ and define the set of starting points by $\mathrm{S} = \{\mathrm{s}\}$. Let $\mathsf{H} = \mathbf{MD}[\mathsf{h}^{\mathrm{s}}, \mathsf{Split}, \mathrm{S}]$ be the hash function associated to these components via the MD transform of Fig. 2. We will think of h' as being a good collision resistant compression function. Then we will show that while h^{s} is a poor collision resistant compression function, H nonetheless remains a good collision resistant hash function.

The idea motivating our construction of h^{s} should be clear. Creating a collision for h^{s} is trivial by making the if statement evaluate to false. However, when h^{s} is used inside of the MD transform with s as a starting point, this case will never occur.

Proposition 6.1 *Let $M_1 = (0^b, \bar{\mathrm{s}})$, $M_2 = (1^b, \bar{\mathrm{s}})$, and $\mathcal{B}$ be the adversary shown in Fig. 6. Then* $\mathbf{Adv}^{\mathrm{cr}}_{\mathsf{h}^{\mathrm{s}}}(\mathcal{B}) = 1$.

Put simply, the above tells us that h^{s} is not collision resistant because $\mathcal{B}$ is clearly efficient.

Proof: (of Proposition 6.1) When we compute $\mathsf{h}^{\mathrm{s}}_k(M_1)$, we see that $\bar{\mathrm{s}}[1] \neq \mathrm{s}[1]$, so $\bar{\mathrm{s}}$ is returned. Similarly, $\bar{\mathrm{s}}$ is returned when we compute $\mathsf{h}^{\mathrm{s}}_k(M_2)$. Notice that $M_1 \neq M_2$ yet $\mathsf{h}^{\mathrm{s}}_k(M_1) = \mathsf{h}^{\mathrm{s}}_k(M_2)$. Thus, h^{s} is not collision resistant. ▌

The following proposition is a useful stepping stone for showing that H is collision resistant if h' is.

Proposition 6.2 *Let $k \in \mathsf{h}'.\mathsf{Keys}$. Then for each excution of $\mathsf{h}^{\mathrm{s}}_k$ in the computation of H_k, the output of $\mathsf{h}^{\mathrm{s}}_k$ is never $\bar{\mathrm{s}}$.*

Proof: (of Proposition 6.2) Fix $M \in \mathsf{Split.Inp}$ and let $\mathbf{m}, \mathbf{c}$ be the vectors computed by $\mathsf{H}_k(M)$. Suppose, for a contradiction, that for some i from 1 to $|\mathbf{m}|$, $\mathsf{h}^{\mathrm{s}}_k(\mathbf{m}[i], \mathbf{c}[i]) = \bar{\mathrm{s}}$ and let $d = \bar{\mathrm{s}}[1]$. Note then the first bit of $\mathbf{c}[i]$ must be d because the if statement in $\mathsf{h}^{\mathrm{s}}_k$ must have evaluated to false. Essentially the same reasoning implies the first bit of $\mathbf{c}[i-1]$ is d.

We can continue this argument for each i back to 1. However, this contradicts the fact that $\mathbf{c}[1] = \mathrm{s}$, so $\mathsf{h}^{\mathrm{s}}_k$ never returns $\bar{\mathrm{s}}$. ▌

Proposition 6.3 *Given an adversary $\mathcal{A}_{\mathsf{H}}$, let $\mathcal{A}_{\mathsf{h}'}$ be the adversary of Fig. 7. Then*

$$\mathbf{Adv}^{\mathrm{cr}}_{\mathsf{H}}(\mathcal{A}_{\mathsf{H}}) \leq \mathbf{Adv}^{\mathrm{cr}}_{\mathsf{h}'}(\mathcal{A}_{\mathsf{h}'}). \qquad (2)$$

The time complexity of $\mathcal{A}_{\mathsf{h}}$ is the sum of the time complexities of $\mathcal{A}_{\mathsf{H}}$ and B_{cr}. The memory complexity of $\mathcal{A}_{\mathsf{h}}$ is the maximum of the memory complexity of $\mathcal{A}_{\mathsf{H}}$ and the memory complexity of B_{cr}.

Notice that Equation (2) tells us that if h' is collision resistant, then H is as well. Let $\mathcal{A}_{\mathsf{H}}$ be a practical adversary against H. Then $\mathcal{A}_{\mathsf{h}'}$ is also practical because its efficiency is about that of $\mathcal{A}_{\mathsf{H}}$. This means that if h' is collision resistant, $\mathbf{Adv}^{\mathrm{cr}}_{\mathsf{h}'}(\mathcal{A}_{\mathsf{h}'})$ is low. Equation (2) tells us that $\mathbf{Adv}^{\mathrm{cr}}_{\mathsf{H}}(\mathcal{A}_{\mathsf{H}})$ will be at most $\mathbf{Adv}^{\mathrm{cr}}_{\mathsf{h}'}(\mathcal{A}_{\mathsf{h}'})$, which means H is also collision resistant.

Proof: (of Proposition 6.3) The facts about the time and memory of $\mathcal{A}_{\mathsf{h}'}$ are clear from its pseudocode.

Now we claim that if the message pair M_1, M_2 returned by $\mathcal{A}_{\mathsf{H}}$ is a collision for $\mathsf{H}_{(k,\mathrm{s})}$ then $\mathcal{A}_{\mathsf{h}'}$ will return a collision for h'_k. Adversary $\mathcal{A}_{\mathsf{h}'}$ takes input $k \in \mathsf{h.Keys}$. It then runs $\mathcal{A}_{\mathsf{H}}$ on input ε given key (k, s) to get a pair of messages (M_1, M_2) in $\mathsf{Split.Inp}$. Then it runs B_{cr} to obtain a pair of inputs to h, which we will refer to as (m_1, c_1) and (m_2, c_2). It then returns these (after removing the first bits of c_1 and c_2).

Suppose M_1, M_2 is a collision for $\mathsf{H}_{(k,\mathrm{s})}$. Since Split is suffix-free, $\mathsf{Split}(M_1) \not\sqsupseteq \mathsf{Split}(M_2)$ and $\mathsf{Split}(M_2) \not\sqsupseteq \mathsf{Split}(M_1)$. Then by Lemma 5.2, we know that B_{cr} will have returned a collision for $\mathsf{h}^{\mathrm{s}}_k$.

From Proposition 6.2 we also know that $\mathsf{h}^{\mathrm{s}}_k((m_1, c_1)) \neq \bar{\mathrm{s}}$ and $\mathsf{h}^{\mathrm{s}}_k((m_2, c_2)) \neq \bar{\mathrm{s}}$. Then it must be the case that they cause the if statement in h^{s} to evaluate to true and so $\mathsf{h}'_k((m_1, c_1[2..|c_1|])) = \mathsf{h}'_k((m_2, c_2[2..|c_2|]))$. Furthermore, $(m_1, c_1) \neq (m_2, c_2)$ and $c_1[1] = c_2[1] = \mathrm{s}[1]$, so $(m_1, c_1[2..|c_1|]) \neq (m_2, c_2[2..|c_2|])$ and thus they form a collision for h'_k.

Therefore, adversary $\mathcal{A}_{\mathsf{h}'}$ finds a collision in h'_k whenever $\mathcal{A}_{\mathsf{H}}$ find a collision for $\mathsf{H}_{(k,\mathrm{s})}$. This justifies Equation (2), completing the proof. ▮

<u>DEFINING A NEW CONSTRAINT FOR CR.</u> The previous example established that traditional definitions of collision resistance with the MD transform do not fully capture the security behind the construction. Although the compression function h used to construct the hash function H was not collision resistant, we were still able to prove the collision resistance of H.

An obvious question at this point is whether there is a natural, weaker assumption we could place on h from which we can still prove H is collision resistant. We answer this in the affirmative with a new security definition in the RS security framework. For this, we now define a new relation which is strictly harder for the adversary to satisfy than R_{cr}, making it a weaker assumption on h. Despite this, we can still obtain the result that the MD transform is *fully* collision resistant under the assumption that h fulfills this weaker security assumption for any suffix-free splitting function. We call our new security definition constrained collision resistance, or $\mathsf{R}_{\mathrm{ccr}}$, and provide the pseudocode for the relation below. We previously defined $\mathsf{R}_{\mathrm{ccr}}$ in Fig. 1.

```
Relation $\mathsf{R}_{\mathrm{ccr}}(k, \mathrm{s}, out)$
$(x_1, x_2, a_1, a_2) \leftarrow out$ ; $(m_1, c_1) \leftarrow x_1$ ; $(m_2, c_2) \leftarrow x_2$
$\mathsf{coll} \leftarrow \mathsf{R}_{\mathrm{cr}}(k, \varepsilon, ((m_1, c_1), (m_2, c_2)))$
$\mathsf{valid} \leftarrow ((c_1 \in \{\mathrm{s}, \mathsf{h}_k(a_1)\})$ and $(c_2 \in \{\mathrm{s}, \mathsf{h}_k(a_2)\}))$
Return ($\mathsf{coll}$ and $\mathsf{valid}$)
```

This relation makes the adversary's job harder than for collision resistance by putting further restrictions of the collisions it is allowed to submit. In particular, it requires that for both chaining variables in the collision submitted by the adversary, this chaining variable must be s or the adversary must know a pre-image for it.

Now we proceed to proving that the MD transform gives a collision resistant hash function if the splitting function is suffix-free and the compression function is constrained-collision resistant. This result helps provide some theoretical understanding to the observation that collisions in the compression functions underlying MD-style hash functions tend not to immediately result in the entire hash function being broken.

Theorem 6.4 *Let* h *be a compression function, let* Split *be a suffix-free splitting function with* $\mathsf{Split.Bl} = \mathsf{h.Bl}$ *and let* $\mathrm{S} \subseteq \mathsf{h.Out}$ *be a set of possible starting points. Let* $\mathsf{H} = \mathbf{MD}[\mathsf{h}, \mathsf{Split}, \mathrm{S}]$ *be the hash function associated to these components via the MD transform of Fig. 2. Given an adversary* $\mathcal{A}_{\mathsf{H}}$*, let* $\mathcal{A}_{\mathsf{h}}$ *be the adversary of Fig. 8 using algorithm* B_{ccr}*. Then*

$$\mathbf{Adv}^{\mathrm{cr}}_{\mathsf{H}}(\mathcal{A}_{\mathsf{H}}) \leq \mathbf{Adv}^{\mathsf{R}_{\mathrm{ccr}}\mathrm{S}}_{\mathsf{h}}(\mathcal{A}_{\mathsf{h}}). \tag{3}$$

The time complexity of $\mathcal{A}_{\mathsf{h}}$ *is the sum of the time complexities of* $\mathcal{A}_{\mathsf{H}}$ *and* B_{ccr}*. The memory complexity of* $\mathcal{A}_{\mathsf{h}}$ *is the maximum of the memory complexity of* $\mathcal{A}_{\mathsf{H}}$ *and the memory complexity of* B_{ccr}*.*

The algorithm B_{ccr} mentioned above (and defined in Fig. 8) is an extension of B_{cr} to also return the values a_1, a_2 expected by $\mathsf{R}_{\mathrm{ccr}}$. We discuss it in more detail in the proof.

Equation (1) tells us that if h is constrained-collision resistant, then H is collision resistant. Let $\mathcal{A}_{\mathsf{H}}$ be a practical adversary against H. Then $\mathcal{A}_{\mathsf{h}}$ is also practical because its efficiency is about that of $\mathcal{A}_{\mathsf{H}}$. If h is constrained collision resistant, then $\mathbf{Adv}^{\mathsf{R}_{\mathrm{ccr}}\mathrm{S}}_{\mathsf{h}}(\mathcal{A}_{\mathsf{h}})$ will be low. Equation (3) tells us that $\mathbf{Adv}^{\mathrm{cr}}_{\mathsf{H}}(\mathcal{A}_{\mathsf{H}})$ will be at most $\mathbf{Adv}^{\mathsf{R}_{\mathrm{ccr}}\mathrm{S}}_{\mathsf{h}}(\mathcal{A}_{\mathsf{h}})$, which means H is collision resistant.

Proof: (of Theorem 6.4) The claimed bounds on the complexity of $\mathcal{A}_{\mathsf{h}}$ are clear from its pseudocode.

Adversary $\mathcal{A}_{\mathsf{h}}$ takes as input a random $(k, \mathrm{s}) \in \mathsf{h.Keys} \times \mathrm{S}$. It runs $\mathcal{A}_{\mathsf{H}}$ on input ε and key (k, s) to get a pair of messages (M_1, M_2) in $\mathsf{Split.Inp}$. Note this exactly matches the input distribution $\mathcal{A}_{\mathsf{H}}$ expects to be given. Adversary $\mathcal{A}_{\mathsf{h}}$ can then run the algorithm B_{ccr} shown in Fig. 8 with inputs $((k, \mathrm{s}), M_1, M_2)$ for it to extract a collision and appropriate information about the pre-images of this collision, if required.

Assume that M_1, M_2 is a collision for $\mathsf{H}_{(k,\mathrm{s})}$. Since Split is suffix-free, $\mathsf{Split}(M_1) \not\sqsupseteq \mathsf{Split}(M_2)$ and $\mathsf{Split}(M_2) \not\sqsupseteq \mathsf{Split}(M_1)$.

We may think of B_{ccr} as a similar algorithm to B_{cr}, with the added task of finding pre-images for the chaining variables in its colliding messages. Indeed, B_{ccr} creates the vectors of chaining variables, $\mathbf{c}_1$ and $\mathbf{c}_2$, and searches for a collision in the same way as B_{cr}, returning this message pair at which it found a collision. Thus, Lemma 5.2 guarantees that the pair $(m_1, c_1), (m_2, c_2)$ forms a collision for h_k. We must verify that a_1, a_2 returned by B_{ccr} additionally satisfies $c_1 \in \{\mathrm{s}, \mathsf{h}_k(a_1)\}$ and $c_2 \in \{\mathrm{s}, \mathsf{h}_k(a_2)\}$. Let $\mathbf{m}_1, \mathbf{c}_1, n_1$ and $\mathbf{m}_2, \mathbf{c}_2, n_2$ be the values calculated by B_{ccr} when run by $\mathcal{A}_{\mathsf{h}}$.

First suppose that B_{ccr} halts in the middle of the execution of its for loop. Then it is clear for the manner they were created that a_1 will be a pre-image for c_1 and a_2 will be a pre-image for c_2.

Now suppose that B_{ccr} does not halt until after the for loop is complete. We will separately analyze the case that $n_1 = n_2$ and the case that $n_1 \neq n_2$. In the former case the chaining variables c_1 and c_2 specifying the collision are $c_1[1]$ and $c_2[1]$, respectively. Since these are both equal to s, $\mathcal{A}_{\mathsf{h}}$ does not need to provide a pre-image for them and we are done. In the latter case the above reasoning tells

```
Algorithm $B_{\mathrm{ccr}}((k,\mathrm{s}),M_1,M_2)$
$\mathbf{m}_1 \leftarrow \mathsf{Split}(M_1)$ ; $\mathbf{m}_2 \leftarrow \mathsf{Split}(M_2)$
$n_1 \leftarrow |\mathbf{m}_1|$ ; $n_2 \leftarrow |\mathbf{m}_2|$
$\mathbf{c}_1[1] \leftarrow \mathrm{s}$ ; $\mathbf{c}_2[1] \leftarrow \mathrm{s}$
For $i = 1,\ldots,n_1$ do $\mathbf{c}_1[i+1] \leftarrow \mathsf{h}_k((\mathbf{m}_1[i],\mathbf{c}_1[i]))$
For $i = 1,\ldots,n_2$ do $\mathbf{c}_2[i+1] \leftarrow \mathsf{h}_k((\mathbf{m}_2[i],\mathbf{c}_2[i]))$
$b \leftarrow \mathrm{argmin}_d(n_d)$
For $i = 0,\ldots,n_b-2$ do
    $(m_1,c_1) \leftarrow (\mathbf{m}_1[n_1-i],\mathbf{c}_1[n_1-i])$
    $(m_2,c_2) \leftarrow (\mathbf{m}_2[n_2-i],\mathbf{c}_2[n_2-i])$
    $a_1 \leftarrow (\mathbf{m}_1[n_1-i-1],\mathbf{c}_1[n_1-i-1])$
    $a_2 \leftarrow (\mathbf{m}_2[n_2-i-1],\mathbf{c}_2[n_2-i-1])$
    If $(m_1,c_1) \neq (m_2,c_2)$ then
        Return $((m_1,c_1),(m_2,c_2),a_1,a_2)$
If $n_1 = n_2$ then
    $(m_1,c_1) \leftarrow (\mathbf{m}_1[1],\mathbf{c}_1[1])$ ; $(m_2,c_2) \leftarrow (\mathbf{m}_2[1],\mathbf{c}_2[1])$
    $a_1 \leftarrow 1; a_2 \leftarrow 2$
    Return $((m_1,c_1),(m_2,c_2),a_1,a_2)$
$(m_1,c_1) \leftarrow (\mathbf{m}_1[n_1-n_b+1],\mathbf{c}_1[n_1-n_b+1])$
$(m_2,c_2) \leftarrow (\mathbf{m}_2[n_2-n_b+1],\mathbf{c}_2[n_2-n_b+1])$
$a_{3-b} \leftarrow (\mathbf{m}_{3-b}[n_{3-b}-n_b],\mathbf{c}_{3-b}[n_{3-b}-n_b])$
$a_b \leftarrow a_{3-b}$
Return $((m_1,c_1),(m_2,c_2),a_1,a_2)$
```

```
Adversary $\mathcal{A}_{\mathsf{h}}(k,\mathrm{s})$
$(M_1,M_2) \leftarrow \mathcal{A}_{\mathsf{H}}((k,\mathrm{s}),\varepsilon)$
Return $B_{\mathrm{ccr}}((k,\mathrm{s}),M_1,M_2)$
```

Figure 8: Adversary $\mathcal{A}_{\mathsf{h}}$ and algorithm B_{ccr} used for Theorems 6.4 and 7.2.

us that $c_b = \mathrm{s}$ (because c_b corresponds to the shorter vector) and so a pre-image is not required for it. This will, presumably, not hold for c_{3-b} (which corresponds to the longer vector), so a pre-image is required for it. As with our earlier analysis we can see that a_{3-b} is a pre-image for c_{3-b} under h_k. Then for compactness $\mathcal{A}_{\mathsf{h}}$ arbitrarily returns this pre-image for both messages and we are again done.

Thus, on any input (k,s), adversary $\mathcal{A}_{\mathsf{h}}$ finds a constrained collision in h_k when $\mathcal{A}_{\mathsf{H}}$ finds a collision in $\mathsf{H}_{(k,\mathrm{s})}$. This justifies Equation (3). ▮

A CCR h. With the introduction of $\mathsf{R}_{\mathrm{ccr}}\mathrm{S}$ security, one might ask whether this assumption is necessary for any h to produce an MD transform that is collision resistant. It is, in fact, not necessary, although it is sufficient. Indeed, the compression function $\mathsf{h}_k^{\mathrm{s}}$ given in Fig. 7 is itself not $\mathsf{R}_{\mathrm{ccr}}\mathrm{S}$ secure, yet we have shown that it results in a collision resistant MD transform. We prove this result below. This shows that an assumption on h even weaker than CCR could suffice, and the benefit of our framework is that one could easily specify such an assumption. However, one has to make some value judgment about the tradeoff between the assumptions and the result. In the extreme, the assumption on h could just be that the MD transform on it is $\mathsf{R}_{\mathrm{cr}}\mathrm{S}$ secure, which is not a useful result. The advantage of $\mathsf{R}_{\mathrm{ccr}}\mathrm{S}$ is that it is appropriately balanced: it is meaningfully weaker than $\mathsf{R}_{\mathrm{cr}}\mathrm{S}$, yet the implication that the MD transform is $\mathsf{R}_{\mathrm{cr}}\mathrm{S}$ secure is still non-trivial.

```
Adversary $\mathcal{B}_{\mathsf{h}^{\mathrm{s}}}(k, \mathrm{s})$
Return $(M_1, M_2, a_1, a_2)$
```

```
$\mathsf{h}^{\dagger}_k((m, c))$
If $(m, c) \in \{(0^b, 1\|0^c), (1^b, 1^2\|0^{c-1})\}$ then
    Return $1^{c+1}$
Return $0\|\mathsf{h}'_k((m, c))$
```

```
Adversary $\mathcal{B}_{\mathsf{h}''}(k, \varepsilon)$
$(x_1, x_2, a_1, a_2) \leftarrow \mathcal{A}_{\mathsf{h}^{\dagger}}(k, \varepsilon)$
Return $(x_1, x_2)$
```

Figure 9: Adversary $\mathcal{B}_{\mathsf{h}^{\mathrm{s}}}$ for Proposition 6.5. Compression function $\mathsf{h}^{\dagger}_k$ and adversary $\mathcal{B}_{\mathsf{h}''}$ for Proposition 6.6.

Proposition 6.5 *Let* $M_1 = a_1 = (0^b, \bar{\mathrm{s}})$*,* $M_2 = a_2 = (1^b, \bar{\mathrm{s}})$*,* $\mathrm{S} \subseteq \mathsf{h}^{\mathrm{s}}.\mathsf{Out}$*, and* $\mathcal{B}_{\mathsf{h}^{\mathrm{s}}}$ *be the adversary shown in Fig. 9. Then* $\mathbf{Adv}^{\mathsf{R}_{\mathrm{ccr}}\mathrm{S}}_{\mathsf{h}^{\mathrm{s}}}(\mathcal{B}_{\mathsf{h}^{\mathrm{s}}}) = 1$.

Put simply, the above tells us that h^{s} is not constrained collision resistant because $\mathcal{B}_{\mathsf{h}^{\mathrm{s}}}$ is clearly efficient.

Proof: (of Proposition 6.5) As shown in the proof of Proposition 6.1, M_1 and M_2 form a collision for h^{s}. Notice that for $M_1 = (0^b, \bar{\mathrm{s}})$, a preimage for $\bar{\mathrm{s}}$ is simply M_1 itself. Similarly M_2 is a preimage for $\bar{\mathrm{s}}$. We thus let $a_1 = M_1$ and $a_2 = M_2$. Therefore, h^{s} is not constrained collision resistant. ▌

A natural question that might arise is whether $\mathsf{R}_{\mathrm{ccr}}\mathrm{S}$ security is actually strictly weaker than $\mathsf{R}_{\mathrm{cr}}\mathrm{S}$. With the revelation that $\mathsf{h}^{\mathrm{s}}_k$ in Fig. 7 is not $\mathsf{R}_{\mathrm{cr}}\mathrm{S}$ secure, is there any compression function that is $\mathsf{R}_{\mathrm{ccr}}\mathrm{S}$ secure yet is not $\mathsf{R}_{\mathrm{cr}}\mathrm{S}$ secure? We claim that such a compression function does exist and give an example in Fig. 9.

We again let $b, c \in \mathbb{N}$. Given a good collision resistant function $\mathsf{h}'' : \mathsf{h}''.\mathsf{Keys} \times (\{0,1\}^b \times \{0,1\}^{c+1}) \to \{0,1\}^c$, we construct the compression function $\mathsf{h}^{\dagger} : \mathsf{h}''.\mathsf{Keys} \times (\{0,1\}^b \times \{0,1\}^{c+1}) \to \{0,1\}^{c+1}$ shown in Fig. 9.

It is clear that $\mathsf{h}^{\dagger}$ is not collision resistant, since for the distinct inputs $(0^b, 1\|0^c)$ and $(1^b, 1^2\|0^{c-1})$ it returns 1^{c+1}. Despite this, we now show that $\mathsf{h}^{\dagger}$ is instead constrained collision resistant.

Proposition 6.6 *Let* $\mathrm{S} \subseteq \mathsf{h}^{\dagger}.\mathsf{Out}$*. Given an adversary* $\mathcal{A}_{\mathsf{h}^{\dagger}}$*, let* $\mathcal{B}_{\mathsf{h}''}$ *be the adversary of Fig. 9. Then* $\mathbf{Adv}^{\mathsf{R}_{\mathrm{ccr}}\mathrm{S}}_{\mathsf{h}^{\dagger}}(\mathcal{A}_{\mathsf{h}^{\dagger}}) \leq \mathbf{Adv}^{\mathrm{cr}}_{\mathsf{h}''}(\mathcal{B}_{\mathsf{h}''})$ *and both the time and memory complexity of* $\mathcal{B}_{\mathsf{h}''}$ *are about that of* $\mathcal{A}_{\mathsf{h}^{\dagger}}$*.*

Notice that given bound on the advantage of the adversaries tells us that if h'' is collision resistant, then $\mathsf{h}^{\dagger}$ is constrained collision resistant. Let $\mathcal{A}_{\mathsf{h}^{\dagger}}$ be a practical adversary against $\mathsf{h}^{\dagger}$. Then $\mathcal{B}_{\mathsf{h}''}$ is also practical because its efficiency is about that of $\mathcal{A}_{\mathsf{h}^{\dagger}}$. This means that if h'' is collision resistant, $\mathbf{Adv}^{\mathrm{cr}}_{\mathsf{h}''}(\mathcal{B}_{\mathsf{h}''})$ is low. The bound tells us that $\mathbf{Adv}^{\mathsf{R}_{\mathrm{ccr}}\mathrm{S}}_{\mathsf{h}^{\dagger}}(\mathcal{A}_{\mathsf{h}^{\dagger}})$ will be at most $\mathbf{Adv}^{\mathrm{cr}}_{\mathsf{h}''}(\mathcal{B}_{\mathsf{h}''})$, which means $\mathsf{h}^{\dagger}$ is constrained collision resistant.

Proof: (of Proposition 6.6) The complexity claims are clear from the pseudocode. We claim that if the tuple (x_1, x_2, a_1, a_2) returned by $\mathcal{A}_{\mathsf{h}^{\dagger}}$ is a constrained collision for $\mathsf{h}^{\dagger}$ then $\mathcal{B}_{\mathsf{h}''}$ will return a collision for h''_k. Adversary $\mathcal{B}_{\mathsf{h}''}$ takes input $k \in \mathsf{h}.\mathsf{Keys}$. It then runs $\mathcal{A}_{\mathsf{h}^{\dagger}}$ on input the given key k and ε to get the tuple (x_1, x_2, a_1, a_2). Letting $(x_1, x_2) = (m_1, c_1), (m_2, c_2)$ we have $(m_1, c_1), (m_2, c_2) \in \{0,1\}^b \times \{0,1\}^{c+1}$ and $a_1, a_2 \in \{0,1\}^{c+1}$. Since $\mathcal{A}_{\mathsf{h}^{\dagger}}$ returns a constrained collision, it must be true that $(m_1, c_1) \neq (m_2, c_2)$, $\mathsf{h}^{\dagger}_k((m_1, c_1)) = \mathsf{h}^{\dagger}_k((m_2, c_2))$, $c_1 \in \{\mathrm{s}, \mathsf{h}^{\dagger}_k(a_1)\}$, and $c_2 \in \{\mathrm{s}, \mathsf{h}^{\dagger}_k(a_2)\}$.

Since $\mathsf{h}^{\dagger}_k$ will only output strings of all 1s or strings that start with 0, it can never output $1\|0^c$ or $1^2\|0^{c-1}$. Thus, neither string has a pre-image, so $(m_1, c_1), (m_2, c_2) \notin \{(0^b, 1\|0^c), (1^b, 1^2\|0^{c-1})\}$. On input (m_1, c_1), the if statement in $\mathsf{h}^{\dagger}_k$ will be false, so $\mathsf{h}^{\dagger}_k$ will return $0\|\mathsf{h}''_k((m_1, c_1))$. Similarly, on input (m_2, c_2), $\mathsf{h}^{\dagger}_k$ will return $0\|\mathsf{h}''_k((m_2, c_2))$. Since $\mathsf{h}^{\dagger}_k((m_1, c_1)) = \mathsf{h}^{\dagger}_k((m_2, c_2))$, we can conclude that $\mathsf{h}''_k((m_1, c_1)) = \mathsf{h}''_k((m_2, c_2))$, therefore forming a collision for h''_k. Adversary $\mathcal{B}_{\mathsf{h}''}$ returns this message pair, so it finds a collision in h''_k whenever $\mathcal{A}_{\mathsf{h}^{\dagger}}$ finds a constrained collision for $\mathsf{h}^{\dagger}$. This justifies the stated advantage bound, completing the proof. ▌

7 A minimal transform

Having to use a suffix-free splitting function necessarily adds some computational overhead to the computation of the MD hash function over what would have been necessary if we were able to use a minimal splitting function. For instance, one such splitting function could be padding M with a single one bit and then with as many zeros as necessary to be of a block size. If the message M is particularly short, this padding scheme may only require one invocation of h when a suffix-free padding function would likely have increased the length of M enough to require a second such invocation.

As such, in some use cases it would be beneficial to use such a minimal splitting function in the transform. We saw earlier that we cannot hope to use a splitting function which is not suffix-free assuming only that the underlying compression function is collision resistant, so it may seem that this efficiency gain would be countered by a loss in provable security.

In this section we show this is not the case, establishing that if the compression function is constrained collision resistant and is fixed-point resistant (a notion introduced by [18]), then it suffices for the splitting function to be injective.

It is important to emphasize here that our proof assumes only the constrained collision resistance of the compression function and, thus, this use of MD may similarly enjoy collision resistance even after a collision is found in the underlying compression function. We leave it to others to decide when this gain in efficiency is worth the security tradeoff required in assuming an additional security property of the hash function.

The main result of this section was inspired by an inquiry from Dodis as to how the analogous result of the proceedings version of this paper [10] (which uses pre-image resistance instead of fixed-point resistance) compared to some of his results with Puniya [18] (which uses collision resistance instead of constrained collision resistance). The current result improves on both of these by using both fixed-point resistance and constrained collision-resistance. In Section 8, we show that the assumption of fixed-point resistance can be dropped assuming that the compression function satisfies a notion of uniformity and the set of message blocks is sufficiently large. This, again, is an improvement on distinct results shown in [10] and [18].

FIXED-POINT RESISTANCE. First we will show how fixed-point resistance is captured in the RS framework by relation R_{fix}. Note that our definition generalizes that of Dodis and Puniya slightly by allowing compression functions to be keyed and allowing the set of starting points, S, to be arbitrary instead of fixed as $\mathrm{S} = \mathsf{h.Out}$. We define a fixed-point witness for starting point s under f_k as a vector of messages $\mathbf{m}$ such that, when $\mathbf{m}$ is iterated upon through the MD transform under f_k, it outputs s. In other words, $\mathsf{f}_k(\mathbf{m}[|\mathbf{m}|], \ldots (\mathsf{f}_k(\mathbf{m}[2], \mathsf{f}_k(\mathbf{m}[1], \mathrm{s})))) = \mathrm{s}$.

The relation R_{fix} requires the adversary to provide a fixed-point witness for s under f_k. This is strictly more difficult to satisfy than R_{pre}, for which the adversary is required to return the pre-image of s under f_k. We provide the pseudocode for relation R_{fix} below.

```
Relation $\mathsf{R}_{\text{fix}}(k, \mathrm{s}, out)$
$\mathbf{m} \leftarrow out; \mathbf{c}[1] \leftarrow \mathrm{s}$
For $i = 1, \ldots, |\mathbf{m}|$ do $\mathbf{c}[i+1] \leftarrow \mathsf{f}_k((\mathbf{m}[i], \mathbf{c}[i]))$
Return $(\mathbf{c}[|\mathbf{m}|+1] = \mathrm{s})$
```

Note if $\mathbf{m}$ is a fixed-point witness for s under f_k, then $(\mathbf{m}[|\mathbf{m}|], c)$ is a pre-image for s under f_k where $c = \mathsf{f}_k(\mathbf{m}[|\mathbf{m}|-1], \ldots (\mathsf{f}_k(\mathbf{m}[2], \mathsf{f}_k(\mathbf{m}[1], \mathrm{s}))))$. Consequently, it is strictly harder to find a fixed-point witness for s than a pre-image, making fixed-point resistance a weaker assumption than pre-image resistance.

Security of the minimal transform. We now proceed with the proof that if the compression function is constrained collision resistant and is fixed-point resistant, then it suffices for the splitting function to be injective. It will be convenient to first prove a lemma we will use in our proof. The lemma establishes that any collision in the MD transform must necessarily give either a collision in the underlying compression function or a fixed-point witness of s for that compression function. This builds on Lemma 5.2 to classify MD collisions by additionally considering what happens when the splitting function is not necessarily suffix-free. This is not a computational statement; it is a fact about the structure of collisions for the MD transform.

Lemma 7.1 *Let* h *be a compression function,* Split *be a splitting function with* $\mathsf{Split.Bl} = \mathsf{h.Bl}^*$*, and* $\mathrm{S} \subseteq \mathsf{h.Out}$ *be a set of possible starting points. Let* $\mathsf{H} = \mathbf{MD}[\mathsf{h}, \mathsf{Split}, \mathrm{S}]$*. Let* $k \in \mathsf{h.Keys}$*,* $\mathrm{s} \in \mathrm{S}$ *and suppose* $M_1, M_2 \in \mathsf{Split.Inp}$ *form a collision for* $\mathsf{H}_{(k,\mathrm{s})}$*. Then at least one of the following two conditions holds:*

1. *On inputs* $(k, \mathrm{s}), M_1, M_2$*, the algorithm* B_{ccr} *shown in Fig. 8 returns* $((m_1, c_1), (m_2, c_2), a_1, a_2)$ *such that* $(m_1, c_1), (m_2, c_2)$ *form a collision for* h_k *and both* $(c_1 \in \{\mathrm{s}, \mathsf{h}_k(a_1)\})$ *and* $(c_2 \in \{\mathrm{s}, \mathsf{h}_k\})$ *hold.*

2. *Given* $(k, \mathrm{s}), M_1, M_2$*, algorithm* B_{fix} *of Fig. 10 returns a fixed-point witness for* s *under* h_k*.*

Proof: (of Lemma 7.1) Let $\mathbf{m}_1 = \mathsf{Split}(M_1)$, $\mathbf{m}_2 = \mathsf{Split}(M_2)$, $n_1 = |\mathbf{m}_1|$, and $n_2 = |\mathbf{m}_2|$, as defined in algorithms B_{ccr} and B_{fix}.

As detailed in the proof of Theorem 6.4 and from Lemma 5.2, if $\mathbf{m}_1 \not\sqsupseteq \mathbf{m}_2$ and $\mathbf{m}_2 \not\sqsupseteq \mathbf{m}_1$, then the first condition holds. So suppose without loss of generality that $\mathbf{m}_1 \sqsupseteq \mathbf{m}_2$. Note it then must hold that $n_1 \geq n_2$.

First suppose there exists an $i \in \{0, \ldots, n_2-1\}$ such that $(\mathbf{m}_1[n_1-i], \mathbf{c}_1[n_1-i]) \neq (\mathbf{m}_2[n_2-i], \mathbf{c}_2[n_2-i])$. Let j be the smallest such value. It will then hold that $\mathbf{c}_1[n_1-(j-1)] = \mathbf{c}_2[n_2-(j-1)]$, so $(\mathbf{m}_1[n_1-j], \mathbf{c}_1[n_1-j])$ and $(\mathbf{m}_2[n_2-j], \mathbf{c}_2[n_2-j])$ form a collision for h_k. During the execution of B_{ccr} it will then return this collision when $i = j$ in the for loop (or after the for loop in the case that $j = n_2 - 1$). The same reasoning as in the proof of Theorem 6.4 tells us that the condition $(c_1 \in \{\mathrm{s}, \mathsf{h}_{Hk}(a_1)\})$ and $(c_2 \in \{\mathrm{s}, \mathsf{h}_{Hk}\})$ will hold of the values returned by B_{ccr}.

Now suppose $(\mathbf{m}_1[n_1-i], \mathbf{c}_1[n_1-i]) = (\mathbf{m}_2[n_2-i], \mathbf{c}_2[n_2-i])$ for all $i \in \{0, \ldots, n_2-1\}$. This implies, in particular, that $\mathbf{c}_1[n_1-(n_2-1)] = \mathbf{c}_2[n_2-(n_2-1)] = \mathrm{s}$. Note it must then hold that $n_1 > n_2$ (because otherwise we would have $\mathbf{m}_1 = \mathbf{m}_2$ contradicting the fact that $M_1 \neq M_2$). Hence, $\mathsf{h}_k((\mathbf{m}_1[n_1-n_2], \mathbf{c}_1[n_1-n_2])) = \mathrm{s}$. From this it is immediate that $\mathbf{m}_1[1, .., n]$ is a fixed-point witness for s under h_k, concluding the proof. ▌

Having established the above lemma we can move on to the main result of this section, that Split being merely injective (and not necessarily suffix-free) suffices to prove that the MD transform

Algorithm $B_{\rm fix}((k,{\rm s}),M_1,M_2)$	Adversary $\mathcal{C}_{\mathsf{h}}(k,{\rm s})$
$\mathbf{m}_1 \leftarrow \mathsf{Split}(M_1)$; $n_1 \leftarrow \lvert\mathbf{m}_1\rvert$	$(M_1,M_2) \leftarrow\$\, \mathcal{A}_{\mathsf{H}}(k,{\rm s})$
$\mathbf{m}_2 \leftarrow \mathsf{Split}(M_2)$; $n_2 \leftarrow \lvert\mathbf{m}_2\rvert$	Return $B_{\rm fix}((k,{\rm s}),M_1,M_2)$
$b \leftarrow \operatorname{argmax}_d(n_d)$	
$n \leftarrow n_b - n_{3-b}$	
Return $\mathbf{m}_b[1,..,n]$	

Figure 10: Algorithm $B_{\rm fix}$ and adversary $\mathcal{C}_{\mathsf{h}}$ for Theorem 7.2.

gives collision resistance if h is fixed-point resistant. Mirroring Theorem 6.4, we will in fact show the result assuming only the weaker notion of constrained collision resistance for the compression function h.

Theorem 7.2 *Let* h *be a compression function, let* Split *be an injective splitting function with* $\mathsf{Split.Bl} = \mathsf{h.Bl}^*$ *and let* ${\rm S} \subseteq \mathsf{h.Out}$ *be a set of possible starting points. Let* $\mathsf{H} = \mathbf{MD}[\mathsf{h},\mathsf{Split},{\rm S}]$. *Given an adversary* $\mathcal{A}_{\mathsf{H}}$*, let* $\mathcal{A}_{\mathsf{h}}$ *be the adversary of Fig. 8 and* $\mathcal{C}_{\mathsf{h}}$ *be the adversary of Fig. 10. Then*

$$\mathbf{Adv}^{\rm cr}_{\mathsf{H}}(\mathcal{A}_{\mathsf{H}}) \le \mathbf{Adv}^{\mathsf{R}_{\rm ccr}\mathsf{S}}_{\mathsf{h}}(\mathcal{A}_{\mathsf{h}}) + \mathbf{Adv}^{\mathsf{R}_{\rm fix}\mathsf{S}}_{\mathsf{h}}(\mathcal{C}_{\mathsf{h}}). \qquad (4)$$

The time complexity of $\mathcal{A}_{\mathsf{h}}$ *is about that of* $\mathcal{A}_{\mathsf{H}}$ *plus that of* $B_{\rm ccr}$*. The memory complexity of* $\mathcal{A}_{\mathsf{h}}$ *is the maximum of that of* $\mathcal{A}_{\mathsf{H}}$ *and that of* $B_{\rm ccr}$*. The time complexity of* $\mathcal{C}_{\mathsf{h}}$ *is about that of* $\mathcal{A}_{\mathsf{H}}$ *plus that of* $B_{\rm fix}$*. The memory complexity of* $\mathcal{C}_{\mathsf{h}}$ *is the maximum of that of* $\mathcal{A}_{\mathsf{H}}$ *and that of* $B_{\rm fix}$*.*

Stating that Split is injective is redundant (splitting functions are required to be injective), but we state this explicitly above to emphasize that injectivity is the *only* property we assume of Split. The theorem proceeds fairly easily from Lemma 7.1 because the relevant adversaries simply run $B_{\rm ccr}$ and $B_{\rm fix}$.

Proof: (of Theorem 7.2) Consider the view of adversary $\mathcal{A}_{\mathsf{H}}$ when run by either $\mathcal{A}_{\mathsf{h}}$, $\mathcal{C}_{\mathsf{h}}$, or in $\mathbf{G}^{\rm cr}_{\mathsf{F}}(\mathcal{A}_{\mathsf{H}})$. In each, it consists of a key k and starting point s, both of which were chosen uniformly at random from their respective sets. From Lemma 7.1, we know if $\mathcal{A}_{\mathsf{H}}$ successfully finds a collision in H for $(k,{\rm s})$ it must be the case that one of $\mathcal{A}_{\mathsf{h}}$ or $\mathcal{C}_{\mathsf{h}}$ will be successful in their respective games (because they simply run $B_{\rm ccr}$ and $B_{\rm fix}$, respectively).

Then we have the following inequality, that establishes the result,

$$\begin{aligned}\mathbf{Adv}^{\rm cr}_{\mathsf{H}}(\mathcal{A}_{\mathsf{H}}) &= \Pr[\mathbf{G}^{\rm cr}_{\mathsf{F}}(\mathcal{A}_{\mathsf{H}})]\\ &\le \Pr[\mathbf{G}^{\mathsf{R}_{\rm ccr}\mathsf{S}}_{\mathsf{h}}(\mathcal{A}_{\mathsf{h}})] + \Pr[\mathbf{G}^{\mathsf{R}_{\rm fix}\mathsf{S}}_{\mathsf{h}}(\mathcal{C}_{\mathsf{h}})]\\ &= \mathbf{Adv}^{\mathsf{R}_{\rm ccr}\mathsf{S}}_{\mathsf{h}}(\mathcal{A}_{\mathsf{h}}) + \mathbf{Adv}^{\mathsf{R}_{\rm fix}\mathsf{S}}_{\mathsf{h}}(\mathcal{C}_{\mathsf{h}}).\end{aligned}$$

The claims on the time and memory complexities of the adversaries are clear. ▌

8 Relationships between notions

We showed that the MD transform can be simplified by additionally assuming that the compression function h satisfies a fixed-point resistance property (and is constrained collision resistant). To help understand this result we will establish the relationships between all of the security notions we have considered in the RS security framework for compression functions.

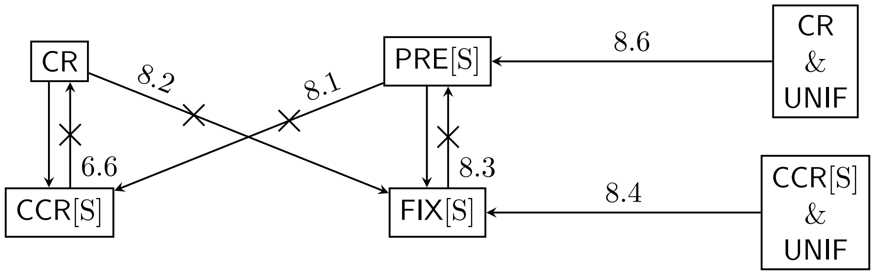

Figure 11: Relationships between security notions in the RS framework. Edges are labeled with theorem/proposition numbers. The implications using UNIF additionally require that the input space of the function be much larger than the output space.

Fig. 11 shows these relations. For a relation $\mathsf{R_a}$ and set S we let A[S] denote $\mathsf{R_a}$S security. For collision resistance we omit the set S because it does not affect the security definition. Unif is a security definition we will introduce momentarily which is not implied by, nor implies any of the other security notions by itself. An arrow from A[S] to B[S] means that for all choices of S any A[S]-secure compression function is also B[S]-secure. A crossed out arrow from A[S] to B[S] means for some choice of S there exists an A[S]-secure compression function which is not B[S]-secure (for Proposition 8.1 this requires the assumption that collision resistant compression functions exist).

As noted previously, the implications CR $\rightarrow$ CCR[S] and Pre[S] $\rightarrow$ Fixed[S] hold trivially from the respective definitions. Proofs for the other relations shown in Fig. 11 are provided below. The relations not shown in the graph between the four security notions are all separations which are implied by the shown relations and transitivity.

NEGATIVE RESULTS. Beyond the trivial implications mentioned above, none of the security notions we consider will imply any other.

Pre-image resistance does not suffice to imply collision resistance or even constrained collision resistance. Consider the hash function g which on any input x returns its key k as output. This is trivially pre-image resistant but not constrained collision resistant. We formalize this in the following proposition.

Proposition 8.1 *Let* g *be the compression function which on any input returns its key,* g.Keys = g.Out*, and* g.BI *is an arbitrary set with at least two distinct elements. Let* S $\subseteq$ Out*. Then for any* $\mathcal{A}$ *it holds that* $\mathbf{Adv}_{\mathsf{g}}^{\mathsf{R}_{\mathrm{pre}}\mathsf{S}}(\mathcal{A}) \leq 1/|\mathsf{Out}|$*. If* m_1 *and* m_2 *are two distinct elements of* g.BI*, then the efficient adversary* $\mathcal{B}$ *defined in the proof satisfies* $\mathbf{Adv}_{\mathsf{g}}^{\mathsf{R}_{\mathrm{ccr}}\mathsf{S}}(\mathcal{B}) = 1$*.*

Proof: (of Proposition 8.1) The first part of this claim holds because the only way that a pre-image of the chosen s $\in$ S will even *exist* is if k happens to equal it. This happens with an exact probability of $1/|\mathsf{Out}|$.

Let $\mathcal{B}$ be the adversary which on input (k, s) simply returns $((m_1, \mathrm{s}), (m_2, \mathrm{s}), (\varepsilon, \varepsilon))$. Then the second part of this claim holds because two distinct elements of g.Inp always form a collision for g_k. ▌

It is also the case that fixed-point resistance is not implied by collision-resistance. The following proposition establishes the result for a particular S = h.Out and can easily be extended to any reasonable choice of a "large" S. Note that an immediate implication of this result is that constrained collision resistance also does not imply fixed-point resistance.

Proposition 8.2 *Let $a \in \mathbb{N}$. Let h' be a family of functions with $\mathsf{h}'.\mathsf{Inp} = \{0,1\} \times \{0,1\}^a$ and $\mathsf{h}'.\mathsf{Out} = \{0,1\}^{a-1}$. Then there exists a compression function h (defined in the proof) with $\mathsf{h}.\mathsf{Inp} = \mathsf{h}'.\mathsf{Inp}$, $\mathsf{h}.\mathsf{Out} = \{0,1\}^a$, and $\mathsf{h}.\mathsf{Keys} = \mathsf{h}'.\mathsf{Keys}$ such that the following holds. For all adversaries $\mathcal{A}$, $\mathbf{Adv}^{\mathrm{cr}}_{\mathsf{h}}(\mathcal{A}) \leq \mathbf{Adv}^{\mathrm{cr}}_{\mathsf{h}'}(\mathcal{A})$. Furthermore, letting $\mathrm{S} = \mathsf{h}.\mathsf{Out}$, we can build an efficient adversary $\mathcal{B}$ (defined in the proof) satisfying, $\mathbf{Adv}^{\mathsf{R}_{\mathrm{fix}}\mathsf{S}}_{\mathsf{h}}(\mathcal{B}) \geq 1/2$.*

Proof: (of Proposition 8.2) Let h and $\mathcal{B}$ be defined as follows.

```
$\mathsf{h}_k((m,c))$
If $((m, c[1]) = (0,0))$ then
    Return $c$
$r \leftarrow \mathsf{h}'_k((m,c))$
Return $1||r$
```

```
Adversary $\mathcal{B}'_{\mathsf{h}}(k, \mathrm{s})$
$\mathbf{m}[1] \leftarrow 0$
Return $\mathbf{m}$
```

To see that the first claim of the above statement holds, note that no collisions in h are possible unless the if statement evaluates to false for both inputs of the collision. In this case, a collision for h is immediately a collision for h'. To see that the second claim holds, observe that for half the choices of $\mathrm{s} \in \mathrm{S}$, specifically those whose first bit are 0, the compression function h_k returns s when given input $(0, \mathrm{s})$. ▌

For our final negative result, the following propositon shows that pre-image resistance is not implied by fixed-point resistance.

Proposition 8.3 *Let $a \in \mathbb{N}$ and $S = \{0,1\}^a$. There exists a compression function $\mathsf{h} : S \times (\{0\} \times S) \to S$ and efficient adversary $\mathcal{A}$ such that $\mathbf{Adv}^{\mathsf{R}_{\mathrm{pre}}\mathsf{S}}_{\mathsf{h}}(\mathcal{A}) \geq 1/2$. Furthermore, for any adversary $\mathcal{B}$ it holds that $\mathbf{Adv}^{\mathsf{R}_{\mathrm{fix}}\mathsf{S}}_{\mathsf{h}}(\mathcal{B}) \leq 2^{-a+1}$.*

Proof: (of Proposition 8.3) Let h and $\mathcal{A}$ be defined as follows.

```
$\mathsf{h}_k((b,c))$
If $c[1] = 1$ then
    Return $\overline{c}$
Return $k$
```

```
Adversary $\mathcal{A}(k, \mathrm{s})$
Return $(0, \overline{s})$
```

The given advantage of $\mathcal{A}$ holds from the fact that if $\mathrm{s}[1] = 0$, then $\mathsf{h}_k((0, \overline{s})) = s$. To bound the advantage of any adversary $\mathcal{B}$, first suppose that $k \notin \{\mathrm{s}, \overline{\mathrm{s}}\}$. If $\mathrm{s}[1] = 1$, then s has no pre-image under h_k and so, therefore, there cannot be a fixed-point witness for s. If $\mathrm{s}[1] = 0$, then the only pre-image for s is $(0, \overline{\mathrm{s}})$. However, notice that $\overline{\mathrm{s}}$ has no pre-image because $\overline{s}[1] = 1$. Thus, in this case it is clear that there also does not exist any fixed-point witness for s. The event that $k \in \{\mathrm{s}, \overline{\mathrm{s}}\}$ happens with probability 2^{-a+1}, so for any adversary $\mathcal{B}$ it holds that $\mathbf{Adv}^{\mathsf{R}_{\mathrm{fix}}\mathsf{S}}_{\mathsf{h}}(\mathcal{B}) \leq 2^{-a+1}$. ▌

<u>Positive results.</u> In Proposition 8.2 we showed that fixed-point resistance is not, in general, implied by collision resistance. This implies that the fixed-point resistance assumed for Theorem 7.2 is necessarily a separate assumption than the assumption of constrained collision resistance. There is, however, an assumption we can make on the structure of the compression function for which fixed-point resistance will not be a separate assumption. In particular, constrained collision resistance will imply fixed-point resistance when the image of a random point is indistinguishable from a random range point. This result extends a result from [18] that fixed-point resistance is implied by

```
Game $\mathbf{G}^{\mathrm{unif}}_{\mathsf{f}}(\mathcal{A})$
$b \leftarrow_{\$} \{0,1\}$ ; $k \leftarrow_{\$} \mathsf{f.Keys}$ ; $x \leftarrow_{\$} \mathsf{f.Inp}$
$\mathrm{s}_0 \leftarrow \mathsf{f}_k(x)$ ; $\mathrm{s}_1 \leftarrow_{\$} \mathsf{f.Out}$
$b' \leftarrow_{\$} \mathcal{A}(k, \mathrm{s}_b)$
Return $(b = b')$
```

```
Adversary $\mathcal{A}_1(k, \mathrm{s})$
$x \leftarrow \mathcal{A}(k, \mathrm{s})$
If $(\mathsf{f}_k(x) = \mathrm{s})$ then
    Return 1
Return 0
```

```
Adversary $\mathcal{A}_2(k, \epsilon)$
$x_1 \leftarrow_{\$} \mathsf{f.Inp}$
$\mathrm{s} \leftarrow \mathsf{f}_k(x_1)$
$x_2 \leftarrow \mathcal{A}(k, \mathrm{s})$
Return $(x_1, x_2)$
```

Figure 12: **Left:** Game defining uniformity of compression function F. **Middle and Right:** adversaries used in proof of Theorem 8.6.

```
Game $\boxed{\mathrm{G}_0}$,$\mathrm{G}_1$
$k \leftarrow_{\$} \mathsf{f.Keys}$
$\mathsf{bad} \leftarrow \mathsf{false}$
$x_1 \leftarrow_{\$} \mathsf{f.Inp}$
$\mathrm{s} \leftarrow \mathsf{f}_k(x_1)$
$\boxed{\mathrm{s} \leftarrow_{\$} \mathsf{f.Out}}$
$x_2 \leftarrow_{\$} \mathcal{A}(k, s)$
If $(x_1 = x_2)$ then
    $\mathsf{bad} \leftarrow \mathsf{true}$
Return $(\mathsf{f}_k(x_2) = s)$
```

```
Game $\mathrm{G}_2$
$k \leftarrow_{\$} \mathsf{f.Keys}$
$\mathsf{bad} \leftarrow \mathsf{false}$
$x_1 \leftarrow_{\$} \mathsf{f.Inp}$
$\mathrm{s} \leftarrow \mathsf{f}_k(x_1)$
$x_2 \leftarrow_{\$} \mathcal{A}(k, s)$
If $(x_1 = x_2)$ then
    $\mathsf{bad} \leftarrow \mathsf{true}$
    Return $\mathsf{false}$
Return $(\mathsf{f}_k(x_2) = s)$
```

Figure 13: Games used in proof of Theorem 8.6. Boxed code is only in game G_0.

collision resistance and the latter property (which they refer to as output regularity) by replacing collision resistance with the weaker notion of constrained collision resistance.

We will define the uniformity of a hash function by the game $\mathbf{G}^{\mathrm{unif}}_{\mathsf{f}}(\mathcal{A})$ shown in Fig. 12. This game measures an adversary's ability to distinguish between the pairs (k, y) and $(k, \mathsf{f}_k(x))$ when y is picked at random from $\mathsf{f.Out}$ and x is picked randomly from $\mathsf{f.Inp}$. The advantage of an adversary is defined by $\mathbf{Adv}^{\mathrm{unif}}_{\mathsf{f}}(\mathcal{A}) = 2\Pr[\mathbf{G}^{\mathrm{unif}}_{\mathsf{f}}(\mathcal{A})] - 1$. It is important to note that this requires the output of f to look uniformly random *even given* k. The most natural way to achieve this property is if for all $k \in \mathsf{f.Keys}$ and $y \in \mathsf{f.Out}$, the set of pre-images of y under f_k has size approximately $\mathsf{f.Inp}/\mathsf{f.Out}$.

The following theorem tells us that if f is sufficiently uniform then constrained collision resistance will imply fixed-point resistance as long as $1/|\mathsf{f.Bl}|$ is small.

Theorem 8.4 *Let* f *be a compression function,* $\mathrm{S} = \mathsf{f.Out}$*, and* $\mathcal{A}$ *be an adversary. Then we can build adversaries* $\mathcal{A}_5$ *and* $\mathcal{A}_6$ *(shown in Fig. 14) such that,*

$$\mathbf{Adv}^{\mathsf{R}_{\mathrm{fix}}\mathsf{S}}_{\mathsf{f}}(\mathcal{A}) \leq 2 \cdot \mathbf{Adv}^{\mathrm{unif}}_{\mathsf{f}}(\mathcal{A}_6) + \mathbf{Adv}^{\mathsf{R}_{\mathrm{ccr}}\mathsf{S}}_{\mathsf{f}}(\mathcal{A}_5) + 1/|\mathsf{f.Bl}|.$$

Both $\mathcal{A}_5$ *and* $\mathcal{A}_6$ *have approximately the same time and memory complexities as* $\mathcal{A}$*.*

To prove the result, we use the uniformity of f to switch to a game in which $\mathcal{A}$ is trying to find a fixed-point witness for $\mathsf{f}_k(x_1)$, where x_1 is chosen at random, instead of for a random s. We then again use the uniformity of f to switch to a game in which $\mathcal{A}$ is trying to find a fixed-point witness now for $\mathsf{f}_k((m, c))$, where m is chosen at random and $c = \mathsf{f}_k(x)$ for a randomly chosen x. Then we consider the standard constrained collision resistance adversary, which chooses $\mathrm{s} = \mathsf{f}_k((m, c))$, where m is chosen at random and $c = \mathsf{f}_k(x)$ for a randomly chosen x. It asks $\mathcal{A}$ to produce a vector that is a fixed-point witness of s, for which it computes the MD transform, after which it can deduce the constrained collision. Analyzing the success of this adversary requires bounding the

```
Adversary $\mathcal{A}_3(k, \mathrm{s})$
$\mathbf{m} \leftarrow_\$ \mathcal{A}(k, s)$
$\mathbf{c}[1] \leftarrow \mathrm{s}$
For $i = 1, \dots, |\mathbf{m}|$ do
    $\mathbf{c}[i+1] \leftarrow \mathsf{f}_k((\mathbf{m}[i], \mathbf{c}[i]))$
If $(\mathbf{c}[|\mathbf{m}|+1] = \mathrm{s})$ then
    Return 1
Return 0
```

```
Adversary $\mathcal{A}_6(k, \mathrm{s})$
$d \leftarrow_\$ \{3, 4\}$
$b \leftarrow_\$ \mathcal{A}_d(k, \mathrm{s})$
Return $b$
```

```
Adversary $\mathcal{A}_4(k, \mathrm{s})$
$m \leftarrow_\$ \mathsf{f.Bl}$
$x_1 \leftarrow (m, s)$
$\mathrm{s} \leftarrow \mathsf{f}_k(x_1)$
$\mathbf{m} \leftarrow_\$ \mathcal{A}(k, s)$
$\mathbf{c}[1] \leftarrow \mathrm{s}$
For $i = 1, \dots, |\mathbf{m}|$ do
    $\mathbf{c}[i+1] \leftarrow \mathsf{f}_k((\mathbf{m}[i], \mathbf{c}[i]))$
If $(x_1 = x_2)$ then
    Return 0
If $(\mathbf{c}[|\mathbf{m}|+1] = \mathrm{s})$ then
    Return 1
Return 0
```

```
Adversary $\mathcal{A}_5(k, \epsilon)$
$x \leftarrow_\$ \mathsf{f.Inp}$
$c \leftarrow \mathsf{f}_k(x)$
$m \leftarrow_\$ \mathsf{f.Bl}$
$x_1 \leftarrow (m, c)$
$\mathrm{s} \leftarrow \mathsf{f}_k(x_1)$
$\mathbf{m} \leftarrow_\$ \mathcal{A}(k, s)$
$\mathbf{c}[1] \leftarrow \mathrm{s}$
For $i = 1, \dots, |\mathbf{m}|$ do
    $\mathbf{c}[i+1] \leftarrow \mathsf{f}_k((\mathbf{m}[i], \mathbf{c}[i]))$
$x_2 \leftarrow (\mathbf{m}[|\mathbf{m}|], \mathbf{c}[|\mathbf{m}|])$
If $(|m| > 1)$ then
    $a_2 \leftarrow (\mathbf{m}[|\mathbf{m}|-1], \mathbf{c}[|\mathbf{m}|-1])$
Else do
    $a_2 \leftarrow x_2$
Return $(x_1, x_2, (x, a_2))$
```

Figure 14: Adversaries used in proof of Theorem 8.4.

probability that the pre-image of s produced in this MD transform computation is not itself (m, c) because $((m, c), (m, c))$ is not a valid collision.

Proof: (of Theorem 8.4) Consider the sequence of games G_0, G_1, G_2, and G_3 shown in Fig. 15. The boxed code is only included in G_0, meaning that s will be chosen uniformly at random. Game G_0 is identical to $\mathbf{G}_{\mathsf{f}}^{\mathsf{R}_{\text{fix}}\mathsf{S}}(\mathcal{A})$ with f and $\mathcal{A}$ hardcoded. In both, $\mathcal{A}$ is given a random key k and string s chosen at random from $\mathsf{f.Out}$. It wins if it correctly returns a fixed-point witness for s under f_k. Thus we have

$$\begin{aligned}\mathbf{Adv}_{\mathsf{f}}^{\mathsf{R}_{\text{fix}}\mathsf{S}}(\mathcal{A}) &= \Pr[\mathbf{G}_{\mathsf{f}}^{\mathsf{R}_{\text{fix}}\mathsf{S}}(\mathcal{A})] \\ &= \Pr[\mathrm{G}_0] \\ &= (\Pr[\mathrm{G}_0] - \Pr[\mathrm{G}_1]) + (\Pr[\mathrm{G}_1] - \Pr[\mathrm{G}_2]) + (\Pr[\mathrm{G}_2] - \Pr[\mathrm{G}_3]) + \Pr[\mathrm{G}_3].\end{aligned}$$

To bound the first difference, note that the first two games differ only in whether s is sampled uniformly at random or as $\mathsf{f}_k(x_1)$ for a randomly chosen x_1. We use a reduction to the uniformity of f defined by adversary $\mathcal{A}_3$ in Fig. 14. Consider the view of $\mathcal{A}$ when run by $\mathcal{A}_3$ (during the execution of $\mathbf{G}_{\mathsf{f}}^{\text{unif}}(\mathcal{A}_3)$). Let b_{unif} denote the bit chosen by the game $\mathbf{G}_{\mathsf{f}}^{\text{unif}}(\mathcal{A}_3)$ and b_3 denote the bit output by $\mathcal{A}_3$. When $b_{\text{unif}} = 1$, the view of $\mathcal{A}$ is k and s chosen uniformly at random. Then $\mathcal{A}_3$ returns $b_3 = 1$ if $\mathcal{A}$ returns a fixed-point for s. Similarly, when $b_{\text{unif}} = 0$, $\mathcal{A}$ is given k and $\mathrm{s} = \mathsf{f}_k(x)$ for a uniformly random x. In this case, $\mathcal{A}_3$ once again returns $b_3 = 1$ if $\mathcal{A}$ returns a fixed-point for s. Thus we have

$$\begin{aligned}\Pr[\mathrm{G}_0] - \Pr[\mathrm{G}_1] &= \Pr[b_3 = 1 | b_{\text{unif}} = 1] - \Pr[b_3 = 1 | b_{\text{unif}} = 0] \\ &= \mathbf{Adv}_{\mathsf{f}}^{\text{unif}}(\mathcal{A}_3).\end{aligned}$$

such that the latter equality holds by a standard conditioning argument.

Now games G_1 and G_2 are identical until bad, so the fundamental lemma of game playing [12] says that $\Pr[\mathrm{G}_1] - \Pr[\mathrm{G}_2] \leq \Pr[\mathrm{G}_1 \text{ sets } \mathsf{bad}]$. In particular, the bad flag is set when $\mathcal{A}$ happens to choose

x_2 exactly as the pre-image x_1 used to define s in G_1. This probability is then

$$\begin{aligned}\Pr[x_1 = x_2] &= \sum_{s^* \in \mathrm{S}} \Pr[x_1 = x_2 \wedge \mathrm{s} = \mathrm{s}^*] \\ &= \sum_{s^* \in \mathrm{S}} \Pr[\mathrm{s} = \mathrm{s}^*] \Pr[x_1 = x_2 | s = s^*] \\ &\leq \sum_{s^* \in \mathrm{S}} (|\mathsf{f}_k^{-1}(\mathrm{s}^*)|/|\mathsf{f.Inp}|)(1/|\mathsf{f}_k^{-1}(\mathrm{s}^*)|) \\ &= \sum_{s^* \in \mathrm{S}} 1/|\mathsf{f.Inp}| = |\mathsf{f.Out}|/|\mathsf{f.Inp}| = 1/|\mathsf{f.Bl}|.\end{aligned}$$

Here probabilities are over the random choice of x_1 and k in G_1 and the coins of $\mathcal{A}$. We use $\mathsf{f}_k^{-1}(\mathrm{s})$ to denote the set of all pre-images of s under f_k. The probability that $\mathcal{A}$ is given a particular s is then exactly $|\mathsf{f}_k^{-1}(\mathrm{s})|/|\mathsf{f.Inp}|$ and the probability that $x_1 = x_2$ given that $\mathcal{A}$ is given s is at most $1/|\mathsf{f}_k^{-1}(\mathrm{s})|$ because the view of $\mathcal{A}$ depends on x_1 only via s. Summing over all possible choices of s and recalling that $\mathsf{f.Inp} = \mathsf{f.Bl} \times \mathsf{f.Out}$ we then have that bad is set with probability at most $1/|\mathsf{f.Bl}|$. This was independent of the key k, so the probability averaged over all choices of k will be bounded by the same probability. Hence, $\Pr[\mathrm{G}_1 \text{ sets } \mathsf{bad}] \leq 1/|\mathsf{f.Bl}|$.

Bounding the difference $\Pr[\mathrm{G}_2] - \Pr[\mathrm{G}_3]$ is similar to that which was shown for the first difference. In both games the first component of x_1 is chosen randomly from $\mathsf{f.Bl}$. The differ in whether the second component is chosen uniformly at random or as $\mathsf{f}_k(x)$ for a randomly chosen x. Consider the adversary $\mathcal{A}_4$ shown in Fig. 14. Using analogous analysis to that used for the bound between games G_0 and G_1 we get $\Pr[\mathrm{G}_2] - \Pr[\mathrm{G}_3] \leq \mathbf{Adv}_{\mathsf{f}}^{\mathrm{unif}}(\mathcal{A}_4)$.

Finally, we argue that the probability $\mathcal{A}$ succeeds in G_3 is at least the probability that adversary $\mathcal{A}_5$ succeeds in $\mathbf{G}_{\mathsf{f}}^{\mathsf{R}_{\mathrm{ccr}}\mathrm{S}}(\mathcal{A}_5)$. The former succeeds if, given $\mathsf{f}_k(x_1)$, it produces a fixed-point witness $\mathbf{m}$ for s under f_k such that when $\mathbf{m}$ is iterated through the MD transform, $(\mathbf{m}[|\mathbf{m}|], \mathbf{c}[|\mathbf{m}|])$ forms a collision with x_1 for f_k. Notice also that this allows $\mathcal{A}_5$ to win the constrained-collision game, since it can return the pre-image for each chaining variable of the messages forming the collision. By definition x is a pre-image for c, the chaining variable of x_1. If $|m| > 1$, then $(\mathbf{m}[|\mathbf{m}|-1], \mathbf{c}[|\mathbf{m}|-1])$ is defined and is a pre-image for $\mathbf{c}[|m|]$ the chaining variable of x_2. If $|m| = 1$, then $\mathbf{c}[|m|] = \mathbf{c}[1] = \mathrm{s}$ so x_2 is itself a pre-image of this whenever it collides with x_1. Hence, $\Pr[\mathrm{G}_3] \leq \mathbf{G}_{\mathsf{f}}^{\mathsf{R}_{\mathrm{ccr}}\mathrm{S}}(\mathcal{A}_5) = \mathbf{Adv}_{\mathsf{f}}^{\mathsf{R}_{\mathrm{ccr}}\mathrm{S}}(\mathcal{A}_5)$.

Combining the given equation gives the following bound.

$$\mathbf{Adv}_{\mathsf{f}}^{\mathsf{R}_{\mathrm{fix}}\mathrm{S}}(\mathcal{A}) \leq \mathbf{Adv}_{\mathsf{f}}^{\mathrm{unif}}(\mathcal{A}_3) + \mathbf{Adv}_{\mathsf{f}}^{\mathrm{unif}}(\mathcal{A}_4) + \mathbf{Adv}_{\mathsf{f}}^{\mathsf{R}_{\mathrm{ccr}}\mathrm{S}}(\mathcal{A}_5) + 1/|\mathsf{f.Bl}|.$$

The final bound is obtained by letting $\mathcal{A}_6$ be the adversary (shown in Fig. 14) which samples $d \leftarrow\$ \{3, 4\}$ and then runs $\mathcal{A}_d$. The stated complexities of the adversaries are apparent from the code of $\mathcal{A}_3$, $\mathcal{A}_4$, and $\mathcal{A}_5$. ▮

As an immediate corollary of Theorem 7.2 and Theorem 8.4 we have the following result that if Split is injective, h is constrained collision resistant and uniform and $1/|\mathsf{h.Bl}|$ is small, then the MD transform achieves full collision resistance.

Corollary 8.5 *Let* h *be a compression function, let* Split *be a splitting function with* $\mathsf{Split.Bl} = \mathsf{h.Bl}^*$ *and let* $\mathrm{S} = \mathsf{h.Out}$*. Let* $\mathsf{H} = \mathbf{MD}[\mathsf{h}, \mathsf{Split}, \mathrm{S}]$*. Given an adversary* $\mathcal{A}_{\mathsf{H}}$*, we can build adversaries* $\mathcal{A}_{\mathrm{ccr}}$ *and* $\mathcal{A}_{\mathrm{unif}}$ *such that*

$$\mathbf{Adv}_{\mathsf{H}}^{\mathrm{cr}}(\mathcal{A}_{\mathsf{H}}) \leq 2 \cdot \mathbf{Adv}_{\mathsf{h}}^{\mathsf{R}_{\mathrm{ccr}}\mathrm{S}}(\mathcal{A}_{\mathrm{ccr}}) + 2 \cdot \mathbf{Adv}_{\mathsf{h}}^{\mathrm{unif}}(\mathcal{A}_{\mathrm{unif}}) + 1/|\mathsf{h.Bl}|. \quad (5)$$

```
Game $\boxed{\mathrm{G}_0}$,$\mathrm{G}_1$
$k \leftarrow_\$ \mathsf{f.Keys}$
$\mathsf{bad} \leftarrow \mathsf{false}$
$x_1 \leftarrow_\$ \mathsf{f.Inp}$
$\mathrm{s} \leftarrow \mathsf{f}_k(x_1)$
$\boxed{\mathrm{s} \leftarrow_\$ \mathsf{f.Out}}$
$\mathbf{m} \leftarrow_\$ \mathcal{A}(k, s)$
$\mathbf{c}[1] \leftarrow \mathrm{s}$
For $i = 1, \ldots, |\mathbf{m}|$ do
   $\mathbf{c}[i+1] \leftarrow \mathsf{f}_k((\mathbf{m}[i], \mathbf{c}[i]))$
$x_2 \leftarrow (\mathbf{m}[|\mathbf{m}|], \mathbf{c}[|\mathbf{m}|])$
If $(x_1 = x_2)$ then
   $\mathsf{bad} \leftarrow \mathsf{true}$
Return $(\mathbf{c}[|\mathbf{m}|+1] = \mathrm{s})$
```

```
Game $\mathrm{G}_2$
$k \leftarrow_\$ \mathsf{f.Keys}$
$\mathsf{bad} \leftarrow \mathsf{false}$
$x_1 \leftarrow_\$ \mathsf{f.Inp}$
$\mathrm{s} \leftarrow \mathsf{f}_k(x_1)$
$\mathbf{m} \leftarrow_\$ \mathcal{A}(k, s)$
$\mathbf{c}[1] \leftarrow \mathrm{s}$
For $i = 1, \ldots, |\mathbf{m}|$ do
   $\mathbf{c}[i+1] \leftarrow \mathsf{f}_k((\mathbf{m}[i], \mathbf{c}[i]))$
$x_2 \leftarrow (\mathbf{m}[|\mathbf{m}|], \mathbf{c}[|\mathbf{m}|])$
If $(x_1 = x_2)$ then
   $\mathsf{bad} \leftarrow \mathsf{true}$
   Return $\mathsf{false}$
Return $(\mathbf{c}[|\mathbf{m}|+1] = \mathrm{s})$
```

```
Game $\mathrm{G}_3$
$k \leftarrow_\$ \mathsf{f.Keys}$
$\mathsf{bad} \leftarrow \mathsf{false}$
$x \leftarrow_\$ \mathsf{f.Inp}$
$c \leftarrow \mathsf{f}_k(x)$
$m \leftarrow_\$ \mathsf{f.Bl}$
$x_1 \leftarrow (m, c)$
$\mathrm{s} \leftarrow \mathsf{f}_k(x_1)$
$\mathbf{m} \leftarrow_\$ \mathcal{A}(k, s)$
$\mathbf{c}[1] \leftarrow \mathrm{s}$
For $i = 1, \ldots, |\mathbf{m}|$ do
   $\mathbf{c}[i+1] \leftarrow \mathsf{f}_k((\mathbf{m}[i], \mathbf{c}[i]))$
$x_2 \leftarrow (\mathbf{m}[|\mathbf{m}|], \mathbf{c}[|\mathbf{m}|])$
If $(x_1 = x_2)$ then
   $\mathsf{bad} \leftarrow \mathsf{true}$
   Return $\mathsf{false}$
Return $(\mathbf{c}[|\mathbf{m}|+1] = \mathrm{s})$
```

Figure 15: Games used in proof of Theorem 8.4. Boxed code is only in game G_0.

The time complexity of $\mathcal{A}_{\mathrm{ccr}}$ is about that of $\mathcal{A}_{\mathsf{H}}$ plus the maximum of B_{ccr} and B_{fix}. The memory complexity of $\mathcal{A}_{\mathrm{ccr}}$ is the maximum of that of $\mathcal{A}_{\mathsf{H}}$, that of B_{ccr}, and that of B_{fix}. The time complexity of $\mathcal{A}_{\mathrm{unif}}$ is about that of $\mathcal{A}_{\mathsf{H}}$ plus that of B_{fix}. The memory complexity of $\mathcal{A}_{\mathrm{unif}}$ is the maximum of that of $\mathcal{A}_{\mathsf{H}}$ and that of B_{fix}.

It is also the case that uniformity and collision resistance together suffice to prove pre-image resistance (assuming $|\mathsf{f.Out}|/|\mathsf{f.Inp}|$ is small). We omit the proof, because it is essentially a simplied version of the proof for Theorem 8.4. The interested reader can find a detailed proof in [10].

Theorem 8.6 *Let f be a family of functions, $\mathrm{S} = \mathsf{f.Out}$, and $\mathcal{A}$ be an adversary. Then we can build adversaries $\mathcal{A}_1$ and $\mathcal{A}_2$ (shown in Fig. 12) such that,*

$$\mathbf{Adv}_{\mathsf{f}}^{\mathsf{R}_{\mathrm{pre}}\mathrm{S}}(\mathcal{A}) \leq \mathbf{Adv}_{\mathsf{f}}^{\mathrm{unif}}(\mathcal{A}_1) + \mathbf{Adv}_{\mathsf{f}}^{\mathrm{cr}}(\mathcal{A}_2) + |\mathsf{f.Out}|/|\mathsf{f.Inp}|.$$

Both $\mathcal{A}_1$ and $\mathcal{A}_2$ have approximately the same time and memory complexities as $\mathcal{A}$.

<u>Prior use of pre-image resistance.</u> The notion of $\mathsf{R}_{\mathrm{pre}}\mathrm{S}$, defined in Fig. 1, with $\mathrm{S} = \mathsf{h.Out}$ has been considered under several different names. In [14], Brown refers to it as both one-wayness and pre-image resistance and uses it as one of several assumptions on a hash function to prove a digital signature scheme is secure. BRS [13] refer to it as inversion resistance and analyze the security of hash functions in an idealized model. In [22], Laccetti and Schmid refer to it as pre-image resistance and analyze the success probability of a brute force attack. In [33], Stinson refers to it as the pre-image problem and bounds the advantage of an adversary in the random oracle model as well as giving reductions between it and some other security notions. Andreeva and Stam [3] refer to it as range-oriented pre-image finding and relate its security to various other notions of pre-image resistance. Finally, DRS [19] refer to it as a variant of pre-image resistance and cite [18] for the definition, though the latter actually considers a weaker notion. The notion of $\mathsf{R}_{\mathrm{pre}}\{z\}$ (where $z \in \{0,1\}^*$ is a fixed string referred to as the zero string) is considered by both [14] and [33], where it is referred to, respectively, as zero-finder-resistance and the zero pre-image problem.

9 Reduction complexity

We will now briefly revisit some of our reductions to provide alternative reductions that are more memory-efficient [4].

Consider, for example, Theorem 5.3. The primary technical component underlying that theorem is the collision-finding algorithm B_{cr} which, given as input a collision for the hash function H, finds a collision for the underlying compression function h. This collision-finding algorithm naturally emanates from various proofs of the MD transform's collision resistance which do not explicitly give an algorithm.[25, 16, 18, 19, 1, 5, 32, 21, 23]. We observe that B_{cr} is a less memory efficient algorithm than the algorithm specified by BBBGKSZ in [5].

Recall the algorithm B_{cr} shown in Fig. 5. It first precomputes the entire vectors $\mathbf{m}_1$, $\mathbf{m}_2$, $\mathbf{c}_1$, and $\mathbf{c}_2$, then processes them *backwards* to find the collision described in the proof of the lemma. Its memory complexity is thus the memory required to store the entire precomputed vectors.

However, this reduction can be done in a more memory efficient manner. We present such an algorithm, called B_{mem}, in Fig. 5. It scans for the collision in the opposite direction, as compared to B_{cr}. By doing so, it avoids the need to precompute the vector $\mathbf{c}$ and instead only computes the individual blocks of $\mathbf{c}$ that it needs in a streaming fashion. Furthermore, for most choices of splitting function considered in practice (e.g. $\mathsf{SplitSha}$), the individual blocks of $\mathsf{Split}(M)$ can be computed independently in a memory-efficient manner which would allow B_{mem} to use only a *constant* amount of memory overhead.

Theorem 9.1 *Let* h *be a compression function, let* Split *be a suffix-free splitting function with* $\mathsf{Split.Bl} = \mathsf{h.Bl}$ *and let* $\mathrm{S} \subseteq \mathsf{h.Out}$ *be a set of possible starting points. Let* $\mathsf{H} = \mathbf{MD}[\mathsf{h}, \mathsf{Split}, \mathrm{S}]$ *be the hash function associated to these components via the MD transform of Fig. 2. Given an adversary* $\mathcal{A}_{\mathsf{H}}$*, let* $\mathcal{A}_{\mathrm{mem}}$ *be the adversary of Fig. 16 using* B_{mem} *from the same figure. Then*

$$\mathbf{Adv}^{\mathrm{cr}}_{\mathsf{H}}(\mathcal{A}_{\mathsf{H}}) \leq \mathbf{Adv}^{\mathrm{cr}}_{\mathsf{h}}(\mathcal{A}_{\mathrm{mem}}). \tag{6}$$

The time complexity of $\mathcal{A}_{\mathrm{mem}}$ *is the sum of the time complexities of* $\mathcal{A}_{\mathsf{H}}$ *and* B_{cr}*. The memory complexity of* $\mathcal{A}_{\mathrm{mem}}$ *is the maximum of the memory complexity of* $\mathcal{A}_{\mathsf{H}}$ *and the memory complexity of* B_{mem}*.*

We make no value statement on how likely the improved memory usage of B_{mem} is to matter in practice. Our point is simply that it *is* more memory efficient and that these differences are hidden by proofs in which the explicit reduction algorithms are not provided.

The proof of the theorem mirrors that of Theorem 5.3, only requiring arguing that B_{mem} also correctly returns a collision whenever $\mathcal{A}_{\mathsf{H}}$ does.

Proof: (of Theorem 9.1) It is clear that the time and memory complexity of adversary $\mathcal{A}_{\mathrm{mem}}$ are as stated in the theorem.

Let $k \in \mathsf{h.Keys}, \mathrm{s} \in \mathrm{S}$ be the values sampled when $\mathcal{A}_{\mathrm{mem}}$ is executed and $M_1, M_2 \in \mathsf{Split.Inp}$ be the values returned by $\mathcal{A}_{\mathsf{H}}$. We have $\mathsf{Split}(M_1) \not\supseteq \mathsf{Split}(M_2)$ and $\mathsf{Split}(M_2) \not\supseteq \mathsf{Split}(M_1)$, so if they form a collision for $\mathsf{H}_{(k,\mathrm{s})}$, then they fulfill the conditions of Lemma 5.2 so B_{cr} would be guaranteed to return a collision for h_k. It is clear from examining the code that if B_{cr} finds a collision on any input, then B_{mem} will find a collision on the same input (though they might output different collisions). As an immediate result Equation (6) holds, completing the proof. ▌

In Fig. 17 we present an analogous memory efficient algorithm B_{mem2} and the corresponding $\mathcal{A}_{\mathrm{mem2}}$ which would obtain the same sorts of memory savings for Theorem 6.4 and Theorem 7.2. For notational convenience, the presented pseudocode of B_{mem2} uses vectors $\mathbf{c}_1$ and $\mathbf{c}_2$, but we note

```
Algorithm $B_{\mathrm{mem}}((k, \mathrm{s}), M_1, M_2)$
$\mathbf{m}_1 \leftarrow \mathsf{Split}(M_1)$ ; $\mathbf{m}_2 \leftarrow \mathsf{Split}(M_2)$
$n_1 \leftarrow |\mathbf{m}_1|$ ; $n_2 \leftarrow |\mathbf{m}_2|$
$n \leftarrow \min(n_1, n_2)$
$c_1 \leftarrow s$; $c_2 \leftarrow s$
If $(n_1 > n_2)$ then
  For $i = 1, \ldots, n_1 - n_2$ do $c_1 \leftarrow \mathsf{h}_k((\mathbf{m}_1[i], c_1))$
If $(n_2 > n_1)$ then
  For $i = 1, \ldots, n_2 - n_1$ do $c_2 \leftarrow \mathsf{h}_k((\mathbf{m}_2[i], c_2))$
For $i = 1, \ldots, n$ do
  $c_1' \leftarrow \mathsf{h}_k((\mathbf{m}_1[n_1 - n + i], c_1))$
  $c_2' \leftarrow \mathsf{h}_k((\mathbf{m}_2[n_2 - n + i], c_2))$
  If $(c_1' = c_2')$ and $(\mathbf{m}_1[n_1 - n + i], c_1) \neq (\mathbf{m}_2[n_2 - n + i], c_2)$ then
    Return $((\mathbf{m}_1[n_1 - n + i], c_1), (\mathbf{m}_2[n_2 - n + i], c_2))$
  $c_1 \leftarrow c_1'$; $c_2 \leftarrow c_2'$
Return $\perp$
```

```
Adversary $\mathcal{A}_{\mathrm{mem}}(k, \varepsilon)$
$\mathrm{s} \leftarrow_{\$} \mathrm{S}$ ; $(M_1, M_2) \leftarrow \mathcal{A}_{\mathsf{H}}((k, \mathrm{s}), \varepsilon)$
Return $B_{\mathrm{mem}}((k, \mathrm{s}), M_1, M_2)$
```

Figure 16: Memory efficient algorithm B_{mem} and adversary $\mathcal{A}_{\mathrm{mem}}$ used for Theorem 9.1.

that the computation can easily be performed using constant memory because only two consecutive chaining variables will need to be stored at a time. Furthermore, we use the convention that out of bounds accesses to an array are accesses to its first element (this simplifies notation for the case that the initialization vector is part of the collision).

We note that ACFK [4] make the claim that collision-resistance is not a memory sensitive problem, saying for example "t-collision-resistance is not memory sensitive for $t = 2$." This seems to imply that there is no reason to worry about memory tightness in our setting because we are doing a reduction to collision resistance. However, this statement is somewhat deceptive. When one unpacks this statement, the actual claim they are making is that the *best known generic attack* does not require much memory. This tells us nothing about whether there may exist better, not yet known, generic attacks or whether there exist better non-generic attacks against specific hash functions for which memory is a dominating factor.

We also observe that, as a community, there is much work to be done in this setting to determine how the memory usage of an adversary "should" be measured to best capture the reality. For example, in their work ACFK observe that many reductions in the random oracle model are highly inefficient in terms of memory complexity. They then show that, in some cases, a PRF can be used to make reductions more tight. However, the value of these points depends heavily on the fact that they adopt a convention of the memory used by the underlying game not counting towards the memory complexity of the adversary. When using our convention that the memory complexity of the adversary includes the memory used by the game in which it is executed (this way of measuring of memory complexity is referred to as **LocalMem** in their work), the value of this observation disappears. In this setting, the straightforward security reductions typically being done in the literature would *already* be memory-tight. By giving our reduction algorithms explicitly, we aim to make it easy for their memory complexity to be analyzed using whichever convention one desires.

```
Algorithm $B_{\mathrm{mem2}}((k,\mathrm{s}),M_1,M_2)$
$\mathbf{m}_1 \leftarrow \mathsf{Split}(M_1)$ ; $\mathbf{m}_2 \leftarrow \mathsf{Split}(M_2)$
$n_1 \leftarrow |\mathbf{m}_1|$ ; $n_2 \leftarrow |\mathbf{m}_2|$
$\mathbf{c}_1[1] \leftarrow \mathrm{s}$ ; $\mathbf{c}_2[1] \leftarrow \mathrm{s}$; $n \leftarrow \min(n_1,n_2)$
If $(n_1 > n_2)$ then
  For $i = 1,\ldots,n_1 - n_2$ do $\mathbf{c}_1[i+1] \leftarrow \mathsf{h}_k((\mathbf{m}_1[i],\mathbf{c}_1[i]))$
If $(n_2 > n_1)$ then
  For $i = 1,\ldots,n_2 - n_1$ do $\mathbf{c}_2[i+1] \leftarrow \mathsf{h}_k((\mathbf{m}_2[i],\mathbf{c}_2[i]))$
For $i = 1,\ldots,n$ do
  $m_1 \leftarrow \mathbf{m}_1[n_1 - n + i]$; $c_1 \leftarrow \mathbf{c}_1[n_1 - n + i]$
  $m_2 \leftarrow \mathbf{m}_2[n_2 - n + i]$; $c_2 \leftarrow \mathbf{c}_2[n_2 - n + i]$
  $c_1' \leftarrow \mathsf{h}_k((m_1,c_1))$
  $c_2' \leftarrow \mathsf{h}_k((m_2,c_2))$
  If $(c_1' = c_2')$ and $(m_1,c_1) \neq (m_2,c_2)$ then
    $a_1 \leftarrow (\mathbf{m}_1[n_1 - n + i - 1], \mathbf{c}_1[n_1 - n + i - 1])$
    $a_2 \leftarrow (\mathbf{m}_2[n_2 - n + i - 1], \mathbf{c}_2[n_2 - n + i - 1])$
    Return $((m_1,c_1),(m_2,c_2),a_1,a_2)$
  $\mathbf{c}_1[n_1 - n + i + 1] \leftarrow c_1'$
  $\mathbf{c}_2[n_2 - n + i + 1] \leftarrow c_2'$
Return $\perp$
```

```
Adversary $\mathcal{A}_{\mathrm{mem2}}(k,\mathrm{s})$
$(M_1,M_2) \leftarrow \mathcal{A}_{\mathsf{H}}((k,\mathrm{s}),\varepsilon)$
Return $B_{\mathrm{mem2}}((k,\mathrm{s}),M_1,M_2)$
```

Figure 17: Memory efficient algorithm and adversary to improve Theorems 6.4 and 7.2.

10 Conclusions

This paper revisited the MD transform to unify prior work and variants, improve security guarantees and formalize folklore results. We introduced the RS security framework for hash functions with which we simultaneously capture several standard notions of security for hash functions and introduce our new notion of constrained collision resistance. Our new security notion allows us to understand ways in which an MD hash function can satisfy collision resistance despite collisions being known for its underlying compression function. In more detail, we have considered a parameterized MD transform that constructs a hash function $\mathsf{H} = \mathbf{MD}[\mathsf{h},\mathsf{Split},\mathrm{S}]$ from a compression function h, splitting function Split, and set S of starting points. We have then comprehensively investigated what assumptions on h and Split guarantee collision resistance (CR) of H. We have shown that MD is better than advertised in the sense that conditions on h weaker than CR, formalized in our RS framework as constrained collision resistance ($\mathsf{R}_{\mathrm{ccr}}\mathrm{S}$), suffice for H to be CR. This strengthens guarantees on hash functions and partially explains why, historically, attacks on compression functions have not immediately translated to attacks on the hash functions. The consequences are the usual benefits of weakening assumptions, namely that weaker compression functions are easier to design, harder to break and more likely to last. Furthermore, we have also shown how to speed up hashing by using very simple Split functions.

References

[1] E. Andreeva, B. Mennink, and B. Preneel. Security reductions of the second round SHA-3 candidates. In M. Burmester, G. Tsudik, S. S. Magliveras, and I. Ilic, editors, *ISC 2010*, volume 6531 of *LNCS*, pages 39–53. Springer, Heidelberg, Oct. 2011. 3, 5, 8, 29

[2] E. Andreeva, G. Neven, B. Preneel, and T. Shrimpton. Seven-property-preserving iterated hashing: ROX. In K. Kurosawa, editor, *ASIACRYPT 2007*, volume 4833 of *LNCS*, pages 130–146. Springer, Heidelberg, Dec. 2007. 6

[3] E. Andreeva and M. Stam. The symbiosis between collision and preimage resistance. In L. Chen, editor, *13th IMA International Conference on Cryptography and Coding*, volume 7089 of *LNCS*, pages 152–171. Springer, Heidelberg, Dec. 2011. 28

[4] B. Auerbach, D. Cash, M. Fersch, and E. Kiltz. Memory-tight reductions. In J. Katz and H. Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 101–132. Springer, Heidelberg, August 20–24 2017. 5, 10, 29, 30

[5] M. Backes, G. Barthe, M. Berg, B. Grégoire, C. Kunz, M. Skoruppa, and S. Z. Béguelin. Verified security of merkle-damgård. In *Computer Security Foundations Symposium (CSF), 2012 IEEE 25th*, pages 354–368. IEEE, 2012. 3, 5, 8, 29

[6] M. Bellare. New proofs for NMAC and HMAC: Security without collision-resistance. In C. Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 602–619. Springer, Heidelberg, Aug. 2006. 5

[7] M. Bellare, D. J. Bernstein, and S. Tessaro. Hash-function based PRFs: AMAC and its multi-user security. In M. Fischlin and J.-S. Coron, editors, *EUROCRYPT 2016, Part I*, volume 9665 of *LNCS*, pages 566–595. Springer, Heidelberg, May 2016. 6

[8] M. Bellare, R. Canetti, and H. Krawczyk. Keying hash functions for message authentication. In N. Koblitz, editor, *CRYPTO'96*, volume 1109 of *LNCS*, pages 1–15. Springer, Heidelberg, Aug. 1996. 5

[9] M. Bellare, R. Canetti, and H. Krawczyk. Pseudorandom functions revisited: The cascade construction and its concrete security. In *37th FOCS*, pages 514–523. IEEE Computer Society Press, Oct. 1996. 6

[10] M. Bellare, J. Jaeger, and J. Len. Better than advertised: Improved collision-resistance guarantees for MD-based hash functions. In B. M. Thuraisingham, D. Evans, T. Malkin, and D. Xu, editors, *ACM CCS 17*, pages 891–906. ACM Press, Oct. / Nov. 2017. 5, 20, 28

[11] M. Bellare and T. Ristenpart. Multi-property-preserving hash domain extension and the EMD transform. In X. Lai and K. Chen, editors, *ASIACRYPT 2006*, volume 4284 of *LNCS*, pages 299–314. Springer, Heidelberg, Dec. 2006. 6

[12] M. Bellare and P. Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In S. Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 409–426. Springer, Heidelberg, May / June 2006. 6, 26

[13] J. Black, P. Rogaway, and T. Shrimpton. Black-box analysis of the block-cipher-based hash-function constructions from PGV. In M. Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 320–335. Springer, Heidelberg, Aug. 2002. 28

[14] D. R. L. Brown. Generic groups, collision resistance, and ECDSA. Contributions to IEEE P1363a, Feb. 2002. Updated version for "The Exact Security of ECDSA." Available from `http://grouper.ieee.org/groups/1363/`. 28

[15] I. Damgård. Collision free hash functions and public key signature schemes. In D. Chaum and W. L. Price, editors, *EUROCRYPT'87*, volume 304 of *LNCS*, pages 203–216. Springer, Heidelberg, Apr. 1988. 8

[16] I. Damgård. A design principle for hash functions. In G. Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 416–427. Springer, Heidelberg, Aug. 1990. 3, 4, 5, 29

[17] H. Dobbertin. Cryptanalysis of md5 compress. 1996. 3, 4

[18] Y. Dodis and 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, and M. Yung, editors, *ACNS 08*, volume 5037 of *LNCS*, pages 156–173. Springer, Heidelberg, June 2008. 3, 5, 8, 20, 24, 28, 29

[19] Y. Dodis, T. Ristenpart, and T. Shrimpton. Salvaging Merkle-Damgård for practical applications. In A. Joux, editor, *EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 371–388. Springer, Heidelberg, Apr. 2009. 3, 5, 6, 8, 28, 29

[20] P. Gaži, K. Pietrzak, and M. Rybár. The exact PRF-security of NMAC and HMAC. In J. A. Garay and R. Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 113–130. Springer, Heidelberg, Aug. 2014. 5

[21] J. Katz and Y. Lindell. *Introduction to modern cryptography*. CRC press, 2014. 4, 5, 29

[22] G. Laccetti and G. Schmid. On a probabilistic approach to the security analysis of cryptographic hash functions. Cryptology ePrint Archive, Report 2004/324, 2004. `http://eprint.iacr.org/2004/324`. 28

[23] A. J. Menezes, P. C. Van Oorschot, and S. A. Vanstone. *Handbook of applied cryptography*. CRC press, 1996. 4, 5, 29

[24] R. C. Merkle. A fast software one-way hash function. *Journal of Cryptology*, 3(1):43–58, 1990. 4, 8

[25] R. C. Merkle. One way hash functions and DES. In G. Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 428–446. Springer, Heidelberg, Aug. 1990. 3, 5, 29

[26] NIST. Fips 180-4, secure hash standard, August 2015. 3, 4, 7, 8

[27] R. Rivest. The md5 message-digest algorithm, 1992. *RFC1321, Internet Engineering Task Force*, 2004. 3

[28] R. L. Rivest. The MD4 message digest algorithm. In A. J. Menezes and S. A. Vanstone, editors, *CRYPTO'90*, volume 537 of *LNCS*, pages 303–311. Springer, Heidelberg, Aug. 1991. 3

[29] P. Rogaway and T. Shrimpton. Cryptographic hash-function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance. In B. K. Roy and W. Meier, editors, *FSE 2004*, volume 3017 of *LNCS*, pages 371–388. Springer, Heidelberg, Feb. 2004. 7

[30] M. Stevens, E. Bursztein, P. Karpman, A. Albertini, and Y. Markov. The first collision for full SHA-1. In J. Katz and H. Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 570–596. Springer, Heidelberg, Aug. 2017. 5

[31] M. Stevens, P. Karpman, and T. Peyrin. Freestart collision for full SHA-1. In M. Fischlin and J.-S. Coron, editors, *EUROCRYPT 2016, Part I*, volume 9665 of *LNCS*, pages 459–483. Springer, Heidelberg, May 2016. 3, 5

[32] D. R. Stinson. *Cryptography: theory and practice*. CRC press, 2005. 4, 5, 29

[33] D. R. Stinson. Some observations on the theory of cryptographic hash functions. *Designs, Codes and Cryptography*, 38(2):259–277, 2006. 28

[34] X. Wang, D. Feng, X. Lai, and H. Yu. Collisions for hash functions MD4, MD5, HAVAL-128 and RIPEMD. Cryptology ePrint Archive, Report 2004/199, 2004. `http://eprint.iacr.org/2004/199`. 5

[35] X. Wang and H. Yu. How to break MD5 and other hash functions. In R. Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 19–35. Springer, Heidelberg, May 2005. 5