Multi-Property Preserving Combiners for Hash Functions

Marc Fischlin and Anja Lehmann

Darmstadt University of Technology, Germany
www.minicrypt.de

Abstract. A robust combiner for hash functions takes two candidate implementations and constructs a hash function which is secure as long as at least one of the candidates is secure. So far, hash function combiners only aim at preserving a single property such as collision-resistance or pseudorandomness. However, when hash functions are used in protocols like TLS they are often required to provide several properties simultaneously. We therefore put forward the notion of multi-property preserving combiners, clarify some aspects on different definitions for such combiners, and propose a construction that provably preserves collision resistance, pseudorandomness, "random-oracle-ness", target collision resistance and message authentication according to our strongest notion.

1 Introduction

Recent attacks on collision-resistant hash functions [17,19,18] have raised the question how to achieve constructions that are more tolerant to cryptanalytic results. One approach has been suggested by Herzberg in [11], where robust combiners have been proposed as a viable strategy for designing less vulnerable hash functions. The classical hash combiner takes two hash functions H_0, H_1 and combines them into a failure-tolerant function by concatenating the outputs of both functions, such that the combiner is collision resistant as long as at least one of the two functions H_0 or H_1 obeys this property.

However, hash functions are currently used for various tasks that require numerous properties beyond collision resistance, e.g., the HMAC construction [2] based on a keyed hash function is used (amongst others) in the IPSec and TLS protocols as a pseudorandom function and as a MAC. In the standardized protocols RSA-OAEP [5] and RSA-PSS [6] even stronger properties are required for the hash functions (cf. [3,4]), prompting Coron et al. [9] to give constructions which propagate the random-oracle property from the compression function to the hash function. A further example for the need of multiple properties is given by Katz and Shin [13], where collision-resistant pseudorandom functions are required in order to protect authenticated group key exchange protocols against insider attacks.[1]

[1] Technically, they require *statistical* collision-resistance for the keys of the pseudorandom function.

Adhering to the usage of hash functions as “swiss army knives” Bellare and Ristenpart [7,8] have shown how to preserve multiple properties in the design of hash functions. In contrast to their approach, which starts with a compression function and aims at constructing a single multi-property preserving (MPP) hash function, a combiner takes two full-grown hash functions and tries to build a hash function which should preserve the properties, even if one of the underlying hash functions is already broken.

The Problem with Multiple Properties. Combiners which preserve a single property such as collision-resistance or pseudorandomness are quite well understood. Multi-property preserving combiners, on the other hand, are not covered by these strategies and require new techniques instead. As an example we discuss this issue for the case of collision-resistance and pseudorandomness.

Recall that the classical combiner for collision-resistance simply concatenates the outputs of both hash functions $\mathsf{Comb}(M) = H_0(M)||H_1(M)$. Obviously, the combiner is collision-resistant as long es either H_0 or H_1 has this property. Yet, it does not guarantee for example pseudorandomness (assuming that the hash functions are keyed) if only one of the underlying hash functions is pseudorandom. An adversary can immediatly distinguish the concatenated output from a truly random value by simply examining the part of the insecure hash function.

An obvious approach to obtain a hash combiner that preserves pseudorandomness is to set $\mathsf{Comb}(M) = H_0(M) \oplus H_1(M)$. However, this combiner is not known to preserve collision-resistance anymore, since a collision for the combiner does not necessarily require collisions on both hash functions. In fact, this combiner also violates the conditions of [1,16] and [10], who have shown that the output of a (black-box) collision-resistant combiner cannot be significantly shorter than the concatenation of the outputs from all employed hash functions. Thus, already the attempt of combining only two properties in a robust manner indicates that finding a multi-property preserving combiner is far from trivial.

Our Construction. In this work we show how to build a combiner that provably preserves multiple properties, where we concentrate on the most common properties as proposed in [8], namely, collision resistance (CR), pseudorandomness (PRF), pseudorandom oracle (PRO), target collision resistance (TCR) and message authentication (MAC).

To explain the underlying idea of our construction it is instructive to recall the bit commitment scheme introduced by Naor [15]. There, the receiver sends a random $3n$-bit string t to the committing party who applies a pseudorandom generator to a random n-bit seed r and returns $G(r) \oplus t$ to commit to 1, or $G(r)$ to commit to 0. Due to the pseudorandomness of the generator's output, the receiver does not learn anything about the committed bit. An ambiguous opening of the commitment by the sender requires to find some $r' \neq r$ such that $G(r) = G(r') \oplus t$. Yet, since there are only 2^{2n} pairs of seeds for the pseudorandom generator but 2^{3n} random strings t, the probability that such a seed pair exists is at most 2^{-n}.

Adopting the approach of Naor we proceed as follows for each hash function H_b. First we hash the large message M with H_b into a short n-bit "seed" x_b. Then we expand this value into a $5n$-bit string (similar to the pseudorandom generator). Next, we xor the result with a subset of n random strings $t_i^b \in \{0,1\}^{5n}$, where the subset is determined by the bits $x_b[i] = 1$. We denote this output by $H_b^{\text{prsrv}}(M)$. Only in the final step we combine the two resulting values for each function H_b into one output $\mathsf{Comb}(M) = H_0^{\text{prsrv}}(M) \oplus H_1^{\text{prsrv}}(M)$ by xor-ing them.

Due to the internal expansion of the short string x_b into five hash values, one can use a similar argumentation as in [15] together with the collision-resistance of one of the hash functions to prove that collision-resistance is preserved. At the same time, pseudorandomness is preserved by the final xor-combination of the results of the two hash functions. Moreover, we also show that this construction propagates several other properties, including PRO, TCR and MAC.

Weak vs. Strong Preservation. We prove our construction to be a *strongly* multi-property preserving combiner for $\{\mathsf{CR}, \mathsf{PRF}, \mathsf{PRO}, \mathsf{TCR}, \mathsf{MAC}\}$. That is, it suffices that each property is provided by at least one hash function, e.g., if H_0 or H_1 has property MAC, then so does the combiner, independently of the other properties. We also introduce further relaxations of MPP, denoted by weakly MPP and mildly MPP. In the weak case the combiner only inherits a set of multiple properties if they are all provided by at least one hash function (i.e., if there is a strong candidate which has all properties at the same time). Mildly MPP combiners are between strongly MPP and weakly MPP combiners, where all properties are granted, but different hash functions may cover different properties.

Our work then adresses several questions related to the different notions of multi-property preservation. Namely, we show that strongly MPP is indeed strictly stronger than mildly MPP which, in turn, implies weakly MPP (but not vice versa). We finally discuss the case of general tree-based combiners for more than two hash functions built out of combiners for two hash functions, as suggested in a more general setting by Harnik et al. [12]. As part of this result we show that such tree-combiners inherit the weakly and strongly MPP property of two-function combiners, whereas mildly MPP two-function combiners suprisingly do not propagate their security to trees.

Organization. We start by defining the three notions of multi-property preserving combiners and giving definitions of the desired properties in Section 2. In Section 3 we give the construction of our MPP combiner and prove that it achieves the strongest MPP notion. A brief discussion about variations of our construction, e.g., to reduce the key size, conclude this section. Section 4 deals again with the different notions of property preservation by showing the correlations between strongly, mildly and weakly MPP combiners. The issue of composing combiners resp. multi-hash combiners is then addressed in Section 5.

2 Preliminaries

2.1 Hash Function Properties

A hash function $\mathcal{H} = (\mathsf{HKGen}, \mathsf{H})$ is a pair of efficient algorithms such that HKGen for input 1^n returns (the description of) a hash function H, and H for input H and $M \in \{0,1\}^*$ deterministically outputs a digest $H(M) \in \{0,1\}^n$. Often, the hash function is also based on a public initial value IV and we therefore occassionally write $H(\mathrm{IV}, M)$ instead of $H(M)$. Similarly, we often identify the hash function with its digest values $H(\cdot)$ if the key generation algorithm is clear from the context.

A hash function may be attributed different properties $\mathsf{P}_1, \mathsf{P}_2, \ldots,$ among which five important ones stand out (cf. [8]):

collision resistance (CR): The hash function is called *collision-resistant* if for any efficient algorithm $\mathcal{A}$ the probability that for $H \leftarrow \mathsf{HKGen}(1^n)$ and $(M, M') \leftarrow \mathcal{A}(H)$ we have $M \neq M'$ but $\mathsf{H}(H, M) = \mathsf{H}(H, M')$, is negligible (as a function of n).

pseudorandomness (PRF): A hash function can be used as a pseudorandom function if the inital value IV is replaced by a randomly chosen key K of the same size (i.e., the key generation algorithm outputs a public part (H, IV) and IV is replaced by a secret key K). Such a keyed hash function $H(K, \cdot)$ is called *pseudorandom* if for any efficient adversary $\mathcal{D}$ the advantage $\mathrm{Prob}\left[\mathcal{D}^{H(K,\cdot)}(H) = 1\right] - \mathrm{Prob}\left[\mathcal{D}^{f}(H) = 1\right]$ is negligible, where the probability in the first case is over $\mathcal{D}$'s coin tosses, the choice of $H \leftarrow \mathsf{HKGen}(1^n)$ and the key K, and in the second case over $\mathcal{D}$'s coin tosses, the choice of $H \leftarrow \mathsf{HKGen}(1^n)$, and the choice of the random function $f : \{0,1\}^* \to \{0,1\}^n$.

pseudorandom oracle (PRO): A hash function H^f based on a random oracle f is called a *pseudorandom oracle* if for any efficient adversary the construction H^f is indifferentiable from a random oracle $\mathcal{F}$, where indifferentiability [14] is a generalization of indistinguishability allowing to consider random oracles that are used as a public component. More formally, a hash function H^f is *indifferentiable* from a random oracle $\mathcal{F}$ if for any efficient adversary $\mathcal{D}$ there exists an efficient algorithm $\mathcal{S}$ such that the advantage $\mathrm{Prob}\left[\mathcal{D}^{H^f, f}(H) = 1\right] - \mathrm{Prob}\left[\mathcal{D}^{\mathcal{F}, \mathcal{S}^{\mathcal{F}}(H)}(H) = 1\right]$ is negligible in n, where the probability in the first case is over $\mathcal{D}$'s coin tosses, $H \leftarrow \mathsf{HKGen}(1^n)$ and the choice of the random function f, and in the second case over the coin tosses of $\mathcal{D}$ and $\mathcal{S}$, and $H \leftarrow \mathsf{HKGen}(1^n)$ and over the choice of $\mathcal{F}$.

target collision-resistance (TCR): Target collision-resistance is a weaker security notion than collision-resistance which obliges the adversary to first commit to a target message M before getting the description $H \leftarrow \mathsf{HKGen}(1^n)$ of the hash function. For the given H the adversary must then find a second message $M' \neq M$ such that $H(M) = H(M')$. More formally, an adversary $\mathcal{A}$ consists of two efficient algorithms $(\mathcal{A}^1, \mathcal{A}^2)$ where $\mathcal{A}^1(1^n)$ first generates the target message M and possibly some additional state information st. Then, a hash function $H \leftarrow \mathsf{HKGen}(1^n)$ is chosen and $\mathcal{A}^2$ has to compute on input (H, M, st) a colliding message $M' \neq M$. A hash function is called *target* *collision-resistant* if or any efficient adversary $\mathcal{A} = (\mathcal{A}^1, \mathcal{A}^2)$ the probability that for $(M, \mathsf{st}) \leftarrow \mathcal{A}^1(1^n)$, $H \leftarrow \mathsf{HKGen}(1^n)$ and $M' \leftarrow \mathcal{A}^2(H, M, \mathsf{st})$ we have $M \neq M'$ but $H(M) = H(M')$, is negligible.

message authentication (MAC): We assume again that the intial value is replaced by a secret random key K. We say that the hash function is a *secure MAC* if for any efficient adversary $\mathcal{A}$ the probability that for $H \leftarrow \mathsf{HKGen}(1^n)$ and random K and $(M, \tau) \leftarrow \mathcal{A}^{H(K,\cdot)}(H)$ we have $\tau = H(K, M)$ and M has never been queried to oracle $H(K, \cdot)$, is negligible.

For a set $\textsc{prop} = \{\mathsf{P}_1, \mathsf{P}_2, \ldots, \mathsf{P}_N\}$ of properties we write $\textsc{prop}(\mathcal{H}) \subseteq \textsc{prop}$ for the properties which hash function $\mathcal{H}$ has.

2.2 Multi-Property Preserving Combiners

A hash function combiner $\mathcal{C} = (\mathsf{CKGen}, \mathsf{Comb})$ for hash functions $\mathcal{H}_0, \mathcal{H}_1$ itself is also a hash function which combines the two functions $\mathcal{H}_0, \mathcal{H}_1$ such that, if at least one of the hash functions obeys property P, then so does the combiner. For multiple properties $\textsc{prop} = \{\mathsf{P}_1, \mathsf{P}_2, \ldots, \mathsf{P}_N\}$ one can either demand that the combiner inherits the properties if one of the candidate hash functions is strong and has all the properties (weakly preserving), or that for each property at least one of the two hash functions has the property (strongly preserving). We also consider a notion in between but somewhat closer to the weak case, called mildly preserving, in which case all properties from $\textsc{prop}$ must hold, albeit different functions may cover different properties (instead of one function as in the case of weakly preserving combiners).[2] More formally,

Definition 1 (Multi-Property Preservation). *For a set* $\textsc{prop} = \{\mathsf{P}_1, \mathsf{P}_2, \ldots, \mathsf{P}_N\}$ *of properties a hash function combiner* $\mathcal{C} = (\mathsf{CKGen}, \mathsf{Comb})$ *for hash functions* $\mathcal{H}_0, \mathcal{H}_1$ *is called* weakly multi-property preserving *(wMPP) for* $\textsc{prop}$ *iff*

$$\textsc{prop} = \textsc{prop}(\mathcal{H}_0) \text{ or } \textsc{prop} = \textsc{prop}(\mathcal{H}_1) \quad \Longrightarrow \quad \textsc{prop} = \textsc{prop}(\mathcal{C}),$$

mildly multi-property preserving *(mMPP) for* $\textsc{prop}$ *iff*

$$\textsc{prop} = \textsc{prop}(\mathcal{H}_0) \cup \textsc{prop}(\mathcal{H}_1) \quad \Longrightarrow \quad \textsc{prop} = \textsc{prop}(\mathcal{C}),$$

and strongly multi-property preserving *(sMPP) for* $\textsc{prop}$ *iff for all* $\mathsf{P}_i \in \textsc{prop}$,

$$\mathsf{P}_i \in \textsc{prop}(\mathcal{H}_0) \cup \textsc{prop}(\mathcal{H}_1) \quad \Longrightarrow \quad \mathsf{P}_i \in \textsc{prop}(\mathcal{C}).$$

We remark that for weak and mild preservation all individual properties $\mathsf{P}_1, \mathsf{P}_2, \ldots, \mathsf{P}_N$ from $\textsc{prop}$ are guaranteed to hold, either by a single function as in weak preservation, or possibly by different functions as in mild preservation. The combiner may therefore depend on some strong property $\mathsf{P}_i \in \textsc{prop}$ which

[2] One may also refine these notions further. We focus on these three "natural" cases.

one of the hash functions has, and which helps to implement some other property P_j in the combined hash function. But then, for a subset $\text{PROP}' \subseteq \text{PROP}$ which, for instance, misses this strong property P_i, the combiner may no longer preserve the properties PROP'. This is in contrast to strongly preserving combiners which support such subsets of properties by definition.

Note that for a singleton $\text{PROP} = \{\mathsf{P}\}$ all notions coincide and we simply say that $\mathcal{C}$ is P-preserving in this case. However, for two or more properties the notions become strictly stronger from weak to mild to strong, as we show in Section 4. Finally, we note that our definition allows the case $\mathcal{H}_0 = \mathcal{H}_1$, which may require some care when designing combiners, especially if the hash functions are based on random oracles (see also the remark after Lemma 3).

3 Constructing Multi-Property Preserving Combiners

In this section we propose our combiner for the properties CR, PRF, PRO, TCR and MAC. We then show it to be strongly multi-property preserving for these properties.

3.1 Our Construction

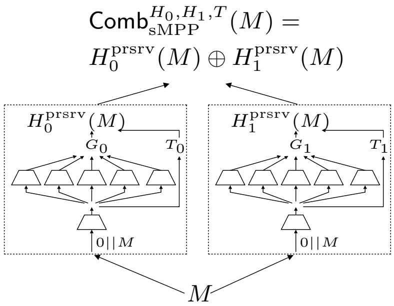

Fig. 1. Combiner $\mathsf{Comb}^{H_0,H_1,T}_{\text{sMPP}}$

Our combiner for functions $\mathcal{H}_0, \mathcal{H}_1$ is a pair of efficient algorithms $\mathcal{C}_{\text{sMPP}} = (\mathsf{CKGen}_{\text{sMPP}}, \mathsf{Comb}_{\text{sMPP}})$. The key generation algorithm $\mathsf{CKGen}_{\text{sMPP}}(1^n)$ generates a triple (H_0, H_1, T) consisting of two hash functions $H_0 \leftarrow \mathsf{HKGen}_0(1^n)$, $H_1 \leftarrow \mathsf{HKGen}_1(1^n)$ and a public string $T = (T_0, T_1)$ where $T_b = (t_1^b, \ldots, t_n^b)$ consists of n uniformly chosen values $t_i^b \in \{0,1\}^{5n}$.

The evaluation algorithm $\mathsf{Comb}^{H_0,H_1,T}_{\text{sMPP}}$ for parameters H_0, H_1, T and message M first computes two hash values $H_b^{\text{prsrv}}(M)$ for $b = 0, 1$, each value based on hash function H_b and string T_b. For this it proceeds in three stages (see Figure 2):

- First hash the large message M into a short string $x_b \in \{0,1\}^n$ via the hash function H_b. For this step we prepend a 0-bit to M in order to make the hash function evaluation here somewhat independent from the subsequent stages.
- Then expand the short string x_b into five hash values $h_i^b = H_b(1||x_b||\langle i \rangle_3)$ for $i = 0, 1, \ldots, 4$, where $\langle i \rangle_3$ denotes the number i represented in binary with 3 bits. Concatenate these strings and denote the resulting $5n$-bit string by $G_b(x_b) = h_0^b||h_1^b||\ldots||h_4^b$.
- Compute $T_b(x_b) = \oplus_{x_b[i]=1} t_i^b$ and add this value to $G_b(x_b)$. Denote the output by $H_b^{\text{prsrv}}(M) = G_b(x_b) \oplus T_b(x_b)$.

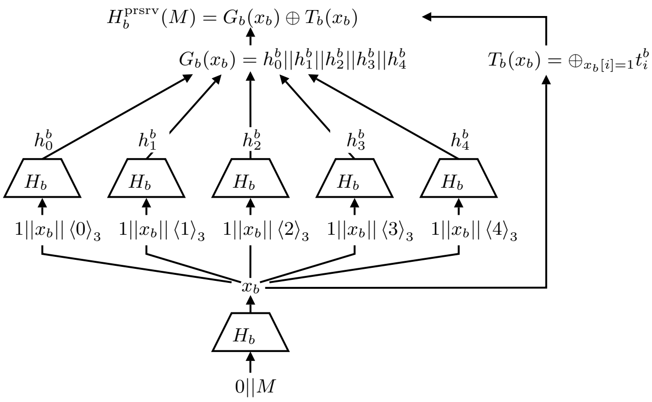

Fig. 2. Construction of H_b^{prsrv} based on hash function H_b

Our combiner now sets $\mathsf{Comb}_{\text{sMPP}}^{H_0,H_1,T}(M) = H_0^{\text{prsrv}}(M) \oplus H_1^{\text{prsrv}}(M)$ as the final output.

3.2 Multi-Property Preservation

We next show that the construction satisfies our strongest notion for combiners:

Theorem 1. *The combiner $\mathcal{C}_{sMPP}$ in Section 3.1 is a strongly multi-property preserving combiner for* PROP $= \{\mathsf{CR}, \mathsf{PRF}, \mathsf{PRO}, \mathsf{TCR}, \mathsf{MAC}\}$.

The theorem is proven in five lemmas, each lemma showing that the combiner preserves one of the properties (as long as at least one hash functions guarantees this property). Since each lemma holds independently of further assumptions, the strong multi-property preservation follows.

Lemma 1. *The combiner $\mathcal{C}_{sMPP}$ is* CR*-preserving.*

Proof. The proof is by contradiction. Assume that an adversary $\mathcal{A}_{\mathsf{Comb}}$ on input H_0, H_1, T, with noticeable probability, outputs $M \neq M'$ with $\mathsf{Comb}_{\text{sMPP}}^{H_0,H_1,T}(M) = \mathsf{Comb}_{\text{sMPP}}^{H_0,H_1,T}(M')$. Then a collision

$$H_0^{\text{prsrv}}(M) \oplus H_1^{\text{prsrv}}(M) = H_0^{\text{prsrv}}(M') \oplus H_1^{\text{prsrv}}(M')$$
$$(G_0(x_0) \oplus T_0(x_0)) \oplus (G_1(x_1) \oplus T_1(x_1)) = (G_0(x_0') \oplus T_0(x_0')) \oplus (G_1(x_1') \oplus T_1(x_1'))$$

implies

$$G_0(x_0) \oplus G_0(x_0') \oplus G_1(x_1) \oplus G_1(x_1') = T_0(x_0) \oplus T_0(x_0') \oplus T_1(x_1) \oplus T_1(x_1'), \quad (1)$$

where x_b denotes the hash value $H_b(0||M)$ of the first hash function evaluation and $G_b(x_b)$ the subsequent computation $h_0^b||h_1^b||h_2^b||h_3^b||h_4^b$ of the the hash values $h_i^b = H_b(1||x_b||\langle i\rangle_3)$ for $i = 0, 1, \ldots, 4$.

The short inputs x_0, x'_0, x_1, x'_1 of n bits only give 2^{4n} possible values on the left side of equation (1), while the probability (over the random choice of the t_i's) that for such a fixed tuple with $x_0 \neq x'_0$ or $x_1 \neq x'_1$ a collision with $T_0(x_0) \oplus T_0(x'_0) \oplus T_1(x_1) \oplus T_1(x'_1)$ occurs, is 2^{-5n}. This follows since for $x_0 \neq x'_0$ or $x_1 \neq x'_1$ at least one of the sums $T_0(x_0) \oplus T_0(x'_0) = T_0(x_0 \oplus x'_0)$ or $T_1(x_1) \oplus T_1(x'_1) = T_1(x_1 \oplus x'_1)$ on the right hand side cannot cancel out. Hence the possibility that there exists some tuple x_0, x'_0, x_1, x'_1 with $x_0 \neq x'_0$ or $x_1 \neq x'_1$ such that equation (1) is statisfied, is at most $2^{4n} \cdot 2^{-5n} = 2^{-n}$ and therefore negligible.

Thus, with overwhelming probability a collision on the combiner only occurs if already the hash values x_b, x'_b at the first stage of the construction collide, i.e., $H_0(0||M) = H_0(0||M')$ and $H_1(0||M) = H_1(0||M')$ for $M \neq M'$. This, however, contradicts the assumption that at least one of the underlying hash functions is collision-resistant. This can be easily formalized through an adversary $\mathcal{A}_b$ for $b \in \{0,1\}$ which, on input $H_b \leftarrow \mathsf{HKGen}_b(1^n)$, samples the other public values $H_{\overline{b}} \leftarrow \mathsf{HKGen}_{\overline{b}}(1^n)$ and T and runs the adversary $\mathcal{A}_{\mathsf{Comb}}$ against the combiner on these data. Whenever $\mathcal{A}_{\mathsf{Comb}}$ outputs (M, M') adversary $\mathcal{A}_b$ returns $(0||M, 0||M')$. By assumption, both adversaries $\mathcal{A}_0, \mathcal{A}_1$ find collisions for H_0 and H_1, respectively, with noticeable probability then. □

Lemma 2. *The combiner $\mathcal{C}_{sMPP}$ is PRF-preserving.*

Proof. The combiner $\mathsf{Comb}_{\text{sMPP}}^{H_0,H_1,T}$ is pseudorandom if the distribution of the combiner's output cannot be distinguished from a truly random function by any polynomial-time adversary. Assume that one of the hash functions H_0 or H_1 is pseudorandom, yet the combiner is not pseudorandom, i.e., there is an adversary $\mathcal{D}_{\mathsf{Comb}}$ that can distinguish the function $\mathsf{Comb}_{\text{sMPP}}^{H_0,H_1,T}(K_0, K_1, \cdot)$ from a random function F with non-negligible probability. We show that this allows to construct a successful distinguisher $\mathcal{D}_b$ for each underlying hash function H_b, which will contradict our initial assumption.

Recall that adversary $\mathcal{D}_{\mathsf{Comb}}$ has oracle access to a function that is either a random function $F : \{0,1\}^* \rightarrow \{0,1\}^{5n}$ or the keyed version of our construction $\mathsf{Comb}_{\text{sMPP}}^{H_0,H_1,T}(K_0, K_1, \cdot)$, where the initial values IV_1, IV_2 in the applications of H_0 and H_1 are replaced by random strings K_0, K_1 of the same size. Then any efficient adversary $\mathcal{D}_{\mathsf{Comb}}$ can be transformed into an adversary $\mathcal{D}_b$ (for some $b \in \{0,1\}$) that distinguishes a random function $f : \{0,1\}^* \rightarrow \{0,1\}^n$ and a keyed hash function $H_b(K_b, \cdot) : \{0,1\}^* \rightarrow \{0,1\}^n$ for a randomly chosen key K_b with the same advantage.

First, the adversary $\mathcal{D}_b$ on input H_b samples $H_{\overline{b}} \leftarrow \mathsf{HKGen}_{\overline{b}}(1^n)$ and a key $K_{\overline{b}}$ and chooses random strings T. It then simulates $\mathcal{D}_{\mathsf{Comb}}$ on input (H_0, H_1, T). For each oracle query M of $\mathcal{D}_{\mathsf{Comb}}$, the adversary $\mathcal{D}_b$ computes a response by simulating the hash construction with the previously chosen key T, the function $H_{\overline{b}}(K_{\overline{b}}, \cdot)$ and its own oracle, i.e., each evaluation of the underlying hash function H_b in the computation of $H_b^{\text{prsrv}}(K_b, M)$ is replaced by the response of $\mathcal{D}_b$'s oracle for the corresponding query. When $\mathcal{D}_{\mathsf{Comb}}$ eventually stops with output bit d algorithm $\mathcal{D}_b$, too, stops and returns d.

For the analysis recall that the underlying oracle of $\mathcal{D}_b$ is either a random function f or the hash function $H_b(K_b, \cdot)$. In the latter case, $\mathcal{D}_b$ perfectly simulates applications of our combiner and therefore generates outputs that are identically distributed to the hash values of the combiner. Hence, the output distribution of $\mathcal{D}_b$ in this case equals the one of $\mathcal{D}_{\mathsf{Comb}}$ with access to $\mathsf{Comb}_{\mathrm{sMPP}}^{H_0,H_1,T}$, i.e.,

$$\mathrm{Prob}\left[\mathcal{D}_b^{H_b(K_b,\cdot)}(H_b) = 1\right] = \mathrm{Prob}\left[\mathcal{D}_{\mathsf{Comb}}^{\mathsf{Comb}_{\mathrm{sMPP}}^{H_0,H_1,T}(K_0,K_1,\cdot)}(H_0,H_1,T) = 1\right].$$

If the oracle of $\mathcal{D}_b$ returns random values using a truly random function f, then the simulated response originates from a structured computation involving f. Yet we claim that the output still looks like a truly random function as long as no collision on the first stage of the construction occurs. With probability at most 2^{-n} any pair of queries $M \neq M'$ of $\mathcal{D}_{\mathsf{Comb}}$ yields a collision under f, i.e., such that $f(0||M) = f(0||M')$ which implies a collision on the final output of the simulation of H_b^{prsrv}. The probability that any collision among $q = q(n) = \mathrm{poly(n)}$ queries of $\mathcal{D}_{\mathsf{Comb}}$ occurs, is therefore at most $\binom{q}{2} \cdot 2^{-n}$. Given that this does not happen, each value $h_i^b = H_b(1||x_b||\langle i\rangle_3)$ for $i = 0, \ldots, 4$ for the second stage is unique and the corresponding images under f are therefore independently and uniformly distributed. Hence $(h_0^b||h_1^b||h_2^b||h_3^b||h_4^b) \oplus T_b(x_b)$ is an independent random string, even when adding the value $H_{\bar{b}}^{\mathrm{prsrv}}(K_{\bar{b}}, M)$. This shows our claim.

Overall, the output distribution of $\mathcal{D}_b^f$ satisfies

$$\begin{aligned}\mathrm{Prob}\left[\mathcal{D}_b^f(H_b) = 1\right] &\leq \mathrm{Prob}\left[\mathcal{D}_b^f(H_b) = 1 \,\middle|\, \text{no Collision}\right] + \mathrm{Prob}\left[\text{Collision}\right]\\ &= \mathrm{Prob}\left[\mathcal{D}_{\mathsf{Comb}}^F(H_0,H_1,T) = 1\right] + \binom{q}{2} \cdot 2^{-n}.\end{aligned}$$

Hence, the probability that $\mathcal{D}_b$ distinguishes H_b from f is

$$\begin{aligned}&\mathrm{Prob}\left[\mathcal{D}_b^{H_b(K_b,\cdot)}(H_b) = 1\right] - \mathrm{Prob}\left[\mathcal{D}_b^f(H_b) = 1\right]\\ &\quad\geq \mathrm{Prob}\left[\mathcal{D}_{\mathsf{Comb}}^{\mathsf{Comb}_{\mathrm{sMPP}}^{H_0,H_1,T}(K_0,K_1,\cdot)}(H_0,H_1,T) = 1\right]\\ &\qquad - \mathrm{Prob}\left[\mathcal{D}_{\mathsf{Comb}}^F(H_0,H_1,T) = 1\right] - \binom{q}{2} \cdot 2^{-n}\end{aligned}$$

and thereby not negligible. This contradicts the assumption that either hash function H_0 or H_1 is a pseudorandom function. □

Lemma 3. *The combiner $\mathcal{C}_{sMPP}$ is PRO-preserving.*

There is a small caveat here. Our definition of combiners allows to use the same hash function $\mathcal{H}_0 = \mathcal{H}_1$, albeit our combiner samples independent instances of the hash functions then. In this sense, it is understood that, if hash function $\mathcal{H}_0$ is given by a random oracle (as required for property PRO), then in case $\mathcal{H}_0 = \mathcal{H}_1$ the other hash function instance uses an independent random oracle.

Proof. We show that $\mathsf{Comb}_{\text{sMPP}}^{H_0,H_1,T}$ is indifferentiable from a random oracle $\mathcal{F}: \{0,1\}^* \to \{0,1\}^{5n}$, when at least one underlying hash function H_0 or H_1 is a random oracle. By symmetry we can assume without loss of generality that $H_0 : \{0,1\}^* \to \{0,1\}^n$ is a random oracle. The (efficient) function H_1 can be arbitrary (but H_1 is sampled independently). It suffices that combiner and the simulator only have black-box access to H_1. The value T, required for the final output of the combiner, is chosen at random and given as input to all participating parties.

The adversary $\mathcal{D}$ has now oracle access either to the combiner $\mathsf{Comb}_{\text{sMPP}}^{H_0,H_1,T}$ and the random oracle H_0 or to $\mathcal{F}$ and a simulator $\mathcal{S}^{\mathcal{F}}$. Our $\mathsf{Comb}_{\text{sMPP}}^{H_0,H_1,T}$ is indifferentiable to $\mathcal{F}$ if there exists a simulator $\mathcal{S}^{\mathcal{F}}$, such that adversary $\mathcal{D}$ cannot have a significant advantage on deciding whether its interacting with $\mathsf{Comb}_{\text{sMPP}}^{H_0,H_1,T}$ and H_0, or with $\mathcal{F}$ and $S^{\mathcal{F}}$. We will use the simulator described below:

Simulator $\mathcal{S}^{\mathcal{F}}_{H_0,H_1,T}(X)$: //use setEntry(), getEntry() to maintain list of queries/answers
on query X check if some entry $Y \leftarrow \text{getEntry}(X)$ already exists
  if $Y = \perp$ //no entry so far
    if $X = 0||M$ for some M
      setEntry$(X) = x_0$ where x_0 is randomly chosen from $\{0,1\}^n$
      get $U \leftarrow \mathcal{F}(M)$ for query M
      get $x_1 \leftarrow H_1(0||M)$ and subsequently $h_i^1 \leftarrow H_1(1||x_1||\langle i\rangle_3)$ for $i = 0,\dots,4$
      calculate $(h_0^0||h_1^0||h_2^0||h_3^0||h_4^0) = U \oplus (h_0^1||h_1^1||h_2^1||h_3^1||h_4^1) \oplus T_1(x_1) \oplus T_0(x_0)$
      save values $h_0^0,\dots,h_4^0$ of potential queries $1||x_0||\langle 0\rangle_3,\dots,1||x_0||\langle 4\rangle_3$:
        setEntry$(1||x_0||\langle i\rangle_3) = h_i^0$ for $i = 0,1,\dots,4$

    if $X \neq 0||M$, choose a random $Y \in \{0,1\}^n$
        and save the value by setEntry$(X) = Y$
output $Y \leftarrow \text{getEntry}(X)$

The simulator's goal is to mimic H_0, i.e., to produce an output that looks consistent to what the distinguisher can obtain from $\mathcal{F}$. To simulate H_0, the simulator $\mathcal{S}$ creates a database, where in addition to the previously processed queries and answers also answers to potential subsequent queries of $\mathcal{D}$ are stored. Those additional entries are generated if $\mathcal{S}$ receives a new query $X = 0||M$, that might be an attempt of $\mathcal{D}$ to simulate the construction of our combiner with the answers of $\mathcal{S}$. In this case, the simulator first chooses a random answer x_0. Then $\mathcal{S}$ invokes the random oracle $\mathcal{F}$ on input M and the black-box function H_1 on input X, where the answer $x_1 \leftarrow H_1(X)$ is used for further queries $1||x_1||\langle i\rangle_3$ to H_1. The responses to those queries correspond to the values $h_0^1,\dots,h_4^1$ at the second stage of the H_1^{prsrv} evaluation. With the help of those values and the output $\mathcal{F}(M)$ of the random oracle, the simulator is able to compute the "missing" answers $h_0^0,\dots,h_4^0$ that it has to return. Each h_i^0 for $i = 0,\dots,4$ is stored for the corresponding query $1||x_0||\langle i\rangle_3$ which $\mathcal{D}$ might submit later. For any new query X that is not of type $0||M$ the simulator responds with a random value from $\{0,1\}^n$ and stores the value.

Except for two events E_1, E_2 (defined below), the simulator will provide outputs that are consistent with $\mathcal{F}$, such that $\mathcal{D}$ cannot distinguish between

$(\mathsf{Comb}_{\text{sMPP}}^{H_0,H_1,T}, H_0)$ and $(\mathcal{F}, S^{\mathcal{F}})$. The first event E_1 is a collision for $\mathcal{S}$ with $\mathcal{S}(0||M) = \mathcal{S}(0||M')$, $M \neq M'$ but $\mathcal{F}(M) \neq \mathcal{F}(M')$, that occurs with probability at most $\binom{q}{2} \cdot 2^{-n}$ where q denotes the number of queries by $\mathcal{D}$.

The second event E_2 occurs if $\mathcal{D}$ makes queries to $\mathcal{S}$ of the form $1||x_0||\langle i \rangle_3$ where x_0 has not been an answer of the simulator before, but on a subsequent query $X = 0||M$ the simulator picks x_0 as its answer. In this case $\mathcal{S}$ has already fixed at least one value h_i^0 for $i = 0, \ldots, 4$ and cannot later define this value after learning $\mathcal{F}(M)$. In particular, $\mathcal{S}$ is then unable to provide a consistent output. But, since $\mathcal{S}$ returns random values from $\{0,1\}^n$ on new queries X, the probability for $\mathcal{S}(X) = x_0$ for any previous query x_0 is at most $q \cdot 2^{-n}$, where q is the maximal number of queries of type $1||x_0||\langle i \rangle_3$ in $\mathcal{D}$'s execution. Overall, event E_2 happens with probability at most $q^2 \cdot 2^{-n}$.

Given that neither event occurs all replies by $\mathcal{S}$ are random (but consistent with the values provided by $\mathcal{F}$). Comparing the two games we note that, for a consistent run, the simulator's random choices and the replies of $\mathcal{F}$ to the simulator's queries implicitly define a random function f, where the only difference to the original construction of H_0^{prsrv} and the "forward" usage of f is that f in the simulation here is defined "backwards" through $\mathcal{F}$. Still, the two experiments look identical from $\mathcal{D}$'s viewpoint.

The advantage of the adversary $\mathcal{D}$ is thus at most the probability that one of the events E_1 or E_2 happens, i.e., $\text{Prob}[E_2 \vee E_2] \leq \left(\binom{q}{2} + q^2\right) \cdot 2^{-n}$. Hence, the probability that $\mathcal{D}$ can distinguish whether it is communicating with $(\mathsf{Comb}_{\text{sMPP}}^{H_0,H_1,T}, H_0)$ or with $(\mathcal{F}, S^{\mathcal{F}})$, is negligible. □

Lemma 4. *The combiner $\mathcal{C}_{sMPP}$ is TCR-preserving.*

The proof that our combiner is target collision-resistant follows the argument for collision-resistance closely and appears in the full version.

Lemma 5. *The combiner $\mathcal{C}_{sMPP}$ is MAC-preserving.*

Proof. Assume towards contradiction that our combiner is *not* a secure MAC. Then there exists an adversary $\mathcal{A}_{\mathsf{Comb}}$ which, after learning several values $\tau_i = \mathsf{Comb}_{\text{sMPP}}^{H_0,H_1,T}(K_0, K_1, M_i)$ for adaptively chosen M_i's, outputs $M \neq M_1, M_2, \ldots, M_q$ and τ such that $\tau = \mathsf{Comb}_{\text{sMPP}}^{H_0,H_1,T}(K_0, K_1, M)$ with noticeable probability.

Given $\mathcal{A}_{\mathsf{Comb}}$ we construct a MAC-adversary $\mathcal{A}_b$ against hash function $\mathcal{H}_b$ for $b \in \{0,1\}$. This adversary $\mathcal{A}_b$ is given H_b as input and oracle access to a function $H_b(K_b, \cdot)$ and uses the attacker $\mathcal{A}_{\mathsf{Comb}}$ in a black-box way to produce a forgery. To this end, $\mathcal{A}_b$ first samples T and $H_{\overline{b}} \leftarrow \mathsf{HKGen}_{\overline{b}}(1^n)$ and $K_{\overline{b}}$ as specified by the combiner, and then invokes $\mathcal{A}_{\mathsf{Comb}}$ for input (H_0, H_1, T). For each query M_i of $\mathcal{A}_{\mathsf{Comb}}$ our adversary computes the combiner's output with the help of its oracle $H_b(K_b, \cdot)$ and knowledge of the other parameters. In particular, for each query adversary $\mathcal{A}_b$ calls its oracle six times about $0||M_i$ and $1||x_{b,i}||\langle 0 \rangle_3, \ldots, 1||x_{b,i}||\langle 4 \rangle_3$.

If, at the end, $\mathcal{A}_{\mathsf{Comb}}$ returns M and τ such that M is not among the previous q queries M_i, then adversary $\mathcal{A}_b$ flips a coin $c \leftarrow \{0,1\}$ and proceeds as follows:

- If $c = 0$ then $\mathcal{A}_b$ chooses an index i at random between 1 and q and looks up the answer $x_{b,i}$ it received in response to its query $0||M_i$. It stops with output $(0||M, x_{b,i})$.
- If $c = 1$ then $\mathcal{A}_b$ queries its oracle about $0||M$ to receive an answer x_b. It then uses its knowledge about the other parameters to compute $H_{\overline{b}}^{\text{prsrv}}(M)$ and calculates $y = \tau \oplus H_{\overline{b}}^{\text{prsrv}}(M) \oplus T_b(x_b)$. It outputs the message $1||x_b||000$ and the first n bits of y and stops.

If $\mathcal{A}_{\mathsf{Comb}}$ fails to output a pair (M, τ) or returns a previously queried message $M = M_i$, then $\mathcal{A}_b$ reports failure and terminates.

For the analysis we consider the two exclusive cases of an successful $\mathcal{A}_{\mathsf{Comb}}$. First, the adversary $\mathcal{A}_{\mathsf{Comb}}$ manages to find a new M and a valid τ such that $H_b(K_b, 0||M)$ collides with some value $H_b(K_b, 0||M_i)$ for some query $0||M_i$. Given this, adversary $\mathcal{A}_b$ outputs $0||M$ and $H_b(K_b, 0||M)$ with probability $\frac{1}{2q}$, namely, if $c = 0$ and the guess for i is correct. But then $0||M$ is distinct from all of $\mathcal{A}_b$'s previous queries (because all $0||M_i$'s are distinct from $0||M$ and all other queries of $\mathcal{A}_b$ are prepended by a 1-bit). Hence, if $\mathcal{A}_{\mathsf{Comb}}$ successfully forges such a MAC with noticeable probability, then so does $\mathcal{A}_b$. Put differently, the probability that $\mathcal{A}_{\mathsf{Comb}}$ succeeds for such cases is negligible by the security of H_b.

The second case occurs if $\mathcal{A}_{\mathsf{Comb}}$ outputs a fresh M and a valid tag τ such that $x_b = H_b(K_b, 0||M)$ is distinct from all values $x_{b,i} = H_b(K_b, 0||M_i)$ for the queries $0||M_i$. In this case, if $c = 1$, adversary $\mathcal{A}_b$ "unmasks" τ to recover $y = H_b(K_b, 1||x_b||\langle 0\rangle_3)||\ldots||H_b(K_b, 1||x_b||\langle 4\rangle_3)$. Note that this requires $\mathcal{A}_b$ to make a further oracle query about value $0||M$. But this value (in addition to all other queries) is different from $1||x_b||000$, and $\mathcal{A}_b$ therefore returns a valid forgery with noticeable probability (if $\mathcal{A}_{\mathsf{Comb}}$ would succeed with noticeable probability for this case).

In summary, it follows that any successful adversary on the combiner MAC immediately yields successful attacks on both hash functions, proving the claim. $\square$

3.3 Variations

In this section we briefly deal with some variations of our previous construction.

Reducing the Key Size. To reduce the key size in our construction we may assume that one of the hash functions is a random oracle and has property PRO, and move from strongly preserving combiners to mildly preserving ones. This also shows that such weaker combiners may come with a gain in efficiency.

If we assume that one hash function behaves like a random function then, instead of picking the t_i's at random and putting them into the key, we define $t_i^b := H_0(\ddagger||b||i) \oplus H_1(\ddagger||b||i)$ for a special symbol $\ddagger$ different from 0 and 1 (e.g., in practice encode 0 and 1 as 00 and 01, respectively, and set $\ddagger = 11$). The prefix $\ddagger$ makes the values independent of the intermediate values in the computation, and the values t_i^b can now be computed "on the fly" instead of storing them in the key.

Given that either hash function has property PRO the values t_i^b are pseudorandom and the proofs in the previous section carry over and we get a *mildly* multi-property preserving combiner for $\textsc{prop} = \{\mathsf{CR}, \mathsf{TCR}, \mathsf{MAC}, \mathsf{PRF}, \mathsf{PRO}\}$. The key size now equals the one for the two underlying hash functions.

Hash Functions with Different Output Sizes. Our construction utilizes the fact that both hash functions have the same output length n. This implies that the concatenation of 5 hash function values $H_b(1||x_b|| \langle i \rangle_3)$ for each function H_b^{prsrv} has the same length.

If we consider two hash functions with distinct output sizes n_0 and n_1, then we need to concatenate $5 \cdot \max\{n_0, n_1\}$ bits of output. For this we simply concatenate enough hash values $H_b(1||x_b|| \langle i \rangle_{\ell_b})$ (with increasing counter values i) for $\ell_b = \lceil \log_2(5 \cdot \max\{n_0, n_1\}/n_b) \rceil$, and truncate longer outputs to $5 \cdot \max\{n_0, n_1\}$ bits. At the same time the t_i^b's are also chosen to be of length $5 \cdot \max\{n_0, n_1\}$. With these modifications all the proofs carry over straightforwardly.

Combining More Hash Functions. To combine $h \geq 3$ hash functions, each with output size $n_0, n_1, \ldots, n_{h-1}$, we set again $n := \max\{n_0, n_1, \ldots, n_{h-1}\}$ and, this time, produce $(2h+1) \cdot n$ output bits for each function H_b^{prsrv}. Accordingly, we let the t_i^b's be of length $(2h+1) \cdot n$. As long as h is polynomial the proofs can be easily transferred to this case.

Alternatively, one can apply our general method to combine three or more hash functions as discussed in Section 5. Yet, this general construction yields a less efficient solution than the tailor-made solution above.

4 Weak vs. Mild vs. Strong Preservation

The first proposition shows that strong preservation implies mild preservation which, in turn, implies weak preservation. The proof is straightforward and given only for sake of completeness:

Proposition 1. *Let* $\textsc{prop}$ *be a set of properties. Then any strongly multi-property preserving combiner for* $\textsc{prop}$ *is also mildly preserving for* $\textsc{prop}$*, and any mildly preserving combiner for* $\textsc{prop}$ *is also weakly preserving for* $\textsc{prop}$*.*

Proof. Assume that the combiner is sMPP for $\textsc{prop}$. Suppose further that $\textsc{prop}(\mathcal{C}) \not\subseteq \textsc{prop}$ such that there is some property $\mathsf{P}_i \in \textsc{prop} - \textsc{prop}(\mathcal{C})$. Then, since the combiner is sMPP, we must also have $\mathsf{P}_i \notin \textsc{prop}(\mathcal{H}_0) \cup \textsc{prop}(\mathcal{H}_1)$, else we derive a contradiction to the strong preservation. We therefore have $\textsc{prop} \not\subseteq \textsc{prop}(\mathcal{H}_0) \cup \textsc{prop}(\mathcal{H}_1)$, implying mild preservation via the contrapositive statement.

Now consider an mMPP combiner and assume $\textsc{prop} = \textsc{prop}(\mathcal{H}_0)$ or $\textsc{prop} = \textsc{prop}(\mathcal{H}_1)$. Then, in particular, $\textsc{prop} = \textsc{prop}(\mathcal{H}_0) \cup \textsc{prop}(\mathcal{H}_1)$ and the mMPP property says that also $\textsc{prop} = \textsc{prop}(\mathcal{C})$. This proves sMPP. $\square$

To separate the notions we consider the collision-resistance property CR and the property NZ (*n*on-*z*ero output) that the hash function should return $0 \cdots 0$ with small probability only. This may be for example required if the hash value should be inverted in a field:

non-zero output (NZ): A hash function $\mathcal{H}$ has property NZ if for any efficient adversary $\mathcal{A}$ the probability that for $H \leftarrow \mathsf{HKGen}(1^n)$ and $M \leftarrow \mathcal{A}(H)$ we have $H(M) = 0 \cdots 0$, is negligible.

Lemma 6. *Let* PROP $= \{\mathsf{CR}, \mathsf{NZ}\}$ *and assume that collision-intractable hash functions exist. Then there is a hash function combiner which is weakly multi-property perserving for* PROP*, but not mildly multi-property preserving for* PROP*.*

Proof. Consider the following combiner (with standard key generation, $(H_0, H_1) \leftarrow \mathsf{CKGen}(1^n)$ for $H_0 \leftarrow \mathsf{HKGen}_0(1^n)$ and $H_1 \leftarrow \mathsf{HKGen}_1(1^n)$):

> The combiner for input M first checks that the length of M is even, and if so, divides $M = L||R$ into halves L and R, and
> - checks that $H_0(L) \neq H_0(R)$ if $L \neq R$, and that $H_0(M) \neq 0 \cdots 0$,
> - verifies that $H_1(L) \neq H_1(R)$ if $L \neq R$, and that $H_1(M) \neq 0 \cdots 0$.
>
> If the length of M is odd or any of the two properties above holds, then the combiner outputs $H_0(M)||H_1(M)$. In any other case, it returns 0^{2n}.

We first show that the combiner is weakly preserving. For this assume that the hash function H_b for $b \in \{0, 1\}$ has both properties. Then the combiner returns the exceptional output 0^{2n} only with negligible probability, namely, if one finds an input with a non-trivial collision under H_b and which also refutes property NZ. In any other case, the combiner's output $H_0(M)||H_1(M)$ inherits the properties CR and NZ from hash function H_b.

Next we show that the combiner is not mMPP. Let H_1' be a collision-resistant hash function with $n - 1$ bits output (and let H_1 include a description of H_1'). Define the following hash functions:

$$H_0(M) = 1^n, \qquad\qquad H_1(M) = \begin{cases} 0^n & \text{if } M = 0^n 1^n \\ 1||H_1'(M) & \text{else} \end{cases}.$$

Clearly, H_0 has property NZ but is not collision-resistant. On the other hand, H_1 obeys CR but not NZ, as $0^n 1^n$ is mapped to zeros. But then we have PROP $= \{\mathsf{CR}, \mathsf{NZ}\} = \text{PROP}(H_0) \cup \text{PROP}(H_1)$ and mild preservation now demands that the combiner, too, has these two properties. Yet, for input $M = 0^n 1^n$ the combiner returns 0^{2n} since the length of M is even, but $L = 0^n$ and $R = 1^n$ collide under H_0, and M is thrown to 0^n under H_1. This means that the combiner does not obey property NZ. □

Lemma 7. *Let* PROP $= \{\mathsf{CR}, \mathsf{NZ}\}$*. Then there exists a hash function combiner which is mildly multi-property perserving for* PROP*, but not strongly multi-property preserving for* PROP*.*

Proof. Consider the following combiner (again with standard key generation):

> The combiner for input M first checks that the length of M is even, and if so, divides $M = L||R$ into halves L and R and then verifies that $H_0(L) \neq H_1(R)$ or $H_1(L) \neq H_1(R)$ or $L = R$. If any of the latter conditions holds, or the length of M is odd, then the combiner outputs $H_0(M)||H_1(M)$. In any other case it returns 0^{2n}.

We first prove that the combiner above is mMPP. Given that $\text{PROP} \subseteq \text{PROP}(H_0) \cup \text{PROP}(H_1)$ at least one of the two hash functions is collision-resistant. Hence, even for $M = L||R$ with even length and $L \neq R$, the hash values only collide with negligible probability. In other words, the combiner outputs $H_0(M)||H_1(M)$ with overwhelming probability, implying that the combiner too has properties CR and NZ.

Now consider the constant hash functions $H_0(M) = H_1(M) = 1^n$ for all M. Clearly, both hash functions obey property $\mathsf{NZ} \in \text{PROP}(H_0) \cup \text{PROP}(H_1)$. Yet, for input 0^n1^n the combiner returns 0^{2n} such that $\mathsf{NZ} \notin \text{PROP}(\mathcal{C})$, implying that the combiner is not strongly preserving. □

The proof indicates how mildly (or weakly) preserving combiners may take advantage of further properties to implement other properties. It remains open if one can find similar separations for the popular properties like CR and PRF, or for CR and PRO.

5 Multiple Hash Functions and Tree-Based Composition of Combiners

So far we have considered combiners for two hash functions. The multi-property preservation definition extends to the case of more hash functions as follows:

Definition 2. *For a set* $\text{PROP} = \{\mathsf{P}_1, \mathsf{P}_2, \ldots, \mathsf{P}_N\}$ *of properties an* m*-function combiner* $\mathcal{C} = (\mathsf{CKGen}, \mathsf{Comb})$ *for hash functions* $\mathcal{H}_0, \mathcal{H}_1, \ldots, \mathcal{H}_{m-1}$ *is called* weakly multi-property preserving *(wMPP) for* PROP *iff*

$$\exists j \in \{0, 1, \ldots, m-1\} \textit{ s.t. } \text{PROP} = \text{PROP}(\mathcal{H}_j) \quad \Longrightarrow \quad \text{PROP} = \text{PROP}(\mathcal{C}),$$

mildly multi-property preserving *(mMPP) for* PROP *iff*

$$\text{PROP} = \bigcup_{j=0}^{m-1} \text{PROP}(\mathcal{H}_j) \quad \Longrightarrow \quad \text{PROP} = \text{PROP}(\mathcal{C}),$$

and strongly multi-property preserving *(sMPP) for* PROP *iff for all* $\mathsf{P}_i \in \text{PROP}$*,*

$$\mathsf{P}_i \in \bigcup_{j=0}^{m-1} \text{PROP}(\mathcal{H}_j) \quad \Longrightarrow \quad \mathsf{P}_i \in \text{PROP}(\mathcal{C}).$$

For the above definitions we still have that sMPP implies mMPP and mMPP implies wMPP. The proof is a straightforward adaption of the case of two hash functions.

Given a combiner for two hash functions one can build a combiner for three or more hash functions by considering the two-function combiner itself as a hash function and applying it recursively. For instance, to combine three hash functions $\mathcal{H}_0, \mathcal{H}_1, \mathcal{H}_2$ one may define the "cascaded" combiner by $\mathcal{C}_2(\mathcal{C}_2(\mathcal{H}_0, \mathcal{H}_1), \mathcal{H}_2)$, where we assume that the output of $\mathcal{C}_2$ allows to be used again as input to the combiner on the next level.

More generally, given m hash functions and a two-function combiner $\mathcal{C}_2$ we define an m-function combiner $\mathcal{C}_{\mathrm{multi}}$ as a binary tree, as suggested for general combiners by [12]. Each leaf is labeled by one of the m hash functions (different leaves may be labeled by the same hash function). Each inner node, including the root, with two descendants labeled by $\mathcal{F}_0$ and $\mathcal{F}_1$, is labeled by $\mathcal{C}_2(\mathcal{F}_0, \mathcal{F}_1)$.

The key generation algorithm for this tree-based combiner now runs the key generation algorithm for the label at each node (each run independent of the others, even if two nodes contain the same label). To evaluate the multi-hash function combiner one inputs M into each leaf and computes the functions outputs recursively up to the root. The output of the root node is then the output of $\mathcal{C}_{\mathrm{multi}}$. We call this a *combiner tree for* $\mathcal{C}_2$ *and* $\mathcal{H}_0, \mathcal{H}_1, \ldots, \mathcal{H}_{m-1}$.

For efficiency reasons we assume that there are at most polynomially many combiner evaluations in a combiner tree. Also, to make the output dependent on all hash functions we assume that each hash function appears in (at least) one of the leaves. If a combiner tree obeys these properties, we call it an *admissible combiner tree for* $\mathcal{C}_2$ *and* $\mathcal{H}_0, \mathcal{H}_1, \ldots, \mathcal{H}_{m-1}$.

We first show that weak MPP and strong MPP preserve their properties for admissible combiner trees:

Proposition 2. *Let $\mathcal{C}_2$ be a weakly (resp. strongly) multi-property preserving two-function combiner for* PROP*. Then any admissible combiner tree for $\mathcal{C}_2$ and functions $\mathcal{H}_0, \mathcal{H}_1, \ldots, \mathcal{H}_{m-1}$ for $m \geq 2$ is also weakly (resp. strongly) multi-property preserving for* PROP*.*

Proof. We give the proof by induction for the depth of the tree. For depth $d = 1$ we have $m = 2$ and $\mathcal{C}_{\mathrm{multi}}(\mathcal{H}_0, \mathcal{H}_1) = \mathcal{C}_2(\mathcal{H}_0, \mathcal{H}_1)$ or $\mathcal{C}_{\mathrm{multi}}(\mathcal{H}_0, \mathcal{H}_1) = \mathcal{C}_2(\mathcal{H}_1, \mathcal{H}_0)$ and the claim follows straightforwardly for both cases.

Now assume $d > 1$ and that combiner Comb_2 is wMPP. Then the root node applies $\mathcal{C}_2$ to two nodes N_0 and N_1, labeled by $\mathcal{F}_0$ and $\mathcal{F}_1$. Note that by the wMPP prerequisite we assume that there exists one hash function $\mathcal{H}_j$ which has all properties in PROP. Since this hash functions appears in at least one of the subtrees under N_0 or N_1, it follows by induction that at least one of the functions $\mathcal{F}_0$ and $\mathcal{F}_1$, too, has properties PROP. But then the combiner application in the root node also inherits these properties from its descendants.

Now consider $d > 1$ and the case of strong MPP. It follows analogoulsy to the previous case that for each property $\mathsf{P}_i \in$ PROP, one of the hash functions in the subtrees rooted at N_0 and N_1 must have property P_i as well. This carries over to the combiners at nodes N_0 or N_1 by induction, and therefore to the root combiner. □

Somewhat surprisingly, mild MPP in general does not propagate security for tree combiners, as we show by a counter-example described in the full version. Note that we still obtain, via the previous proposition, that the mMPP combiner is also wMPP and that the resulting tree combiner is thus also wMPP. Yet, it loses its mMPP property.

Proposition 3. *Let* PROP $= \{\mathsf{CR}, \mathsf{NZ}\}$ *and assume that there are collision-intractable hash functions. Then there exists a two-function weakly multi-property preserving combiner* $\mathcal{C}_2$ *for* PROP*, and an admissible tree combiner for* $\mathcal{C}_2$ *and hash functions* $\mathcal{H}_0, \mathcal{H}_1, \mathcal{H}_2$ *which is not mildly multi-property preserving for* PROP.

Note that the cascading combiner can also be applied to our combiner in Section 3 to compose three or more hash functions (with the adaption for hash functions with different output lengths discussed in Section 3.3). The derived combiner, however, is less efficient than the direct construction sketched there.

Acknowledgments

We thank the anonymous reviewers for valuable comments. Both authors are supported by the Emmy Noether Program Fi 940/2-1 of the German Research Foundation (DFG).

References

1. Dan Boneh and Xavier Boyen. *On the Impossibility of Efficiently Combining Collision Resistant Hash Functions.* Advances in Cryptology — Crypto 2006, Volume 4117 of Lecture Notes in Computer Science, pages 570–583. Springer-Verlag, 2006.
2. Mihir Bellare, Ran Canetti, and Hugo Krawczyk. *Keying hash functions for message authentication.* Advances in Cryptology — Crypto 1996, Volume 96 of Lecture Notes in Computer Science, pages 1–15. Springer-Verlag, 1996.
3. Alexandra Boldyreva and Marc Fischlin. *Analysis of Random Oracle Instantiation Scenarios for OAEP and Other Practical Schemes.* Advances in Cryptology — Crypto 2005, Volume 3621 of Lecture Notes in Computer Science, pages 412–429. Springer-Verlag, 2005.
4. Alexandra Boldyreva and Marc Fischlin. *On the Security of OAEP.* Advances in Cryptology — Asiacrypt 2006, Volume 4284 of Lecture Notes in Computer Science, pages 210–225. Springer-Verlag, 2006.
5. Mihir Bellare and Phillip Rogaway. *Optimal Asymmetric Encryption — How to Encrypt with RSA.* Advances in Cryptology — Eurocrypt'94, Volume 950 of Lecture Notes in Computer Science, pages 92–111. Springer-Verlag, 1995.
6. Mihir Bellare and Phillip Rogaway. *The exact security of digital signatures — How to sign with RSA and Rabin.* Advances in Cryptology — Eurocrypt'96, Volume 1070 of Lecture Notes in Computer Science, pages 399–416. Springer-Verlag, 1996.

7. Mihir Bellare and Thomas Ristenpart. *Multi-Property Preserving Hash Domain Extensions and the EMD Transform.* Advances in Cryptology — Asiacrypt'06, Volume 4284 of Lecture Notes in Computer Science, pages 299–314. Springer-Verlag, 2006.
8. Mihir Bellare and Thomas Ristenpart. *Hash Functions in the Dedicated-Key Setting: Design Choices and MPP Transforms.* International Colloquium on Automata, Languages, and Progamming (ICALP), Volume 4596 of Lecture Notes in Computer Science, pages 399–410. Springer-Verlag, 2007.
9. Jean-Sebastien Coron, Yevgeniy Dodis, Cecile Malinaud, and Prashant Puniya. *Merkle-Damgard revisited: How to construct a hash function.* Advances in Cryptology — Crypto 2005, Volume 3621 of Lecture Notes in Computer Science. Springer-Verlag, 2005.
10. Ran Canetti, Ronald L. Rivest, Madhu Sudan, Luca Trevisan, Salil P. Vadhan, and Hoeteck Wee. *Amplifying Collision Resistance: A Complexity-Theoretic Treatment.* Advances in Cryptology — Crypto 2007, Volume 4622 of Lecture Notes in Computer Science, pages 264–283. Springer-Verlag, 2007.
11. Amir Herzberg. *On Tolerant Cryptographic Constructions.* Topics in Cryptology — Cryptographer's Track, RSA Conference (CT-RSA) 2005, Volume 3376 of Lecture Notes in Computer Science, pages 172–190. Springer-Verlag, 2005.
12. Danny Harnik, Joe Kilian, Moni Naor, Omer Reingold, and Alon Rosen. *On Robust Combiners for Oblivious Transfer and other Primitives.* Advances in Cryptology — Eurocrypt 2005, Volume 3494 of Lecture Notes in Computer Science, pages 96–113. Springer-Verlag, 2005.
13. Jonathan Katz and Ji Sun Shin. *Modeling Insider Attacks on Group Key-Exchange Protocols.* Proceedings of the Annual Conference on Computer and Communications Security (CCS). ACM Press, 2005.
14. Ueli Maurer, Renato Renner, and Clemens Holenstein. *Indifferentiability, Impossibility Results on Reductions, and Applications to the Random Oracle Methodology.* Theory of Cryptography Conference (TCC) 2004, Volume 2951 of Lecture Notes in Computer Science, pages 21–39. Springer-Verlag, 2004.
15. Moni Naor. *Bit Commitment Using Pseudo-Randomness. Journal of Cryptology*, 4(2):151–158, 1991.
16. Krzysztof Pietrzak. *Non-Trivial Black-Box Combiners for Collision-Resistant Hash-Functions don't Exist.* Advances in Cryptology — Eurocrypt 2007, Lecture Notes in Computer Science. Springer-Verlag, 2007.
17. Xiaoyun Wang, Xuejia Lai, Dengguo Feng, Hui Chen, and Xiuyuan Yu. *Cryptanalysis of the Hash Functions MD4 and RIPEMD.* Advances in Cryptology — Eurocrypt 2005, Volume 3494 of Lecture Notes in Computer Science, pages 1–18. Springer-Verlag, 2005.
18. Xiaoyun Wang and Hongbo Yu. *How to break MD5 and other hash functions.* Advances in Cryptology — Eurocrypt 2005, Volume 3494 of Lecture Notes in Computer Science, pages 19–35. Springer-Verlag, 2005.
19. Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu. *Finding collisions in the full SHA-1.* Advances in Cryptology — Crypto 2005, Volume 3621 of Lecture Notes in Computer Science, pages 17–36. Springer-Verlag, 2005.