To Hash or Not to Hash Again? (In)differentiability Results for H^2 and HMAC

Yevgeniy Dodis[1], Thomas Ristenpart[2], John Steinberger[3], and Stefano Tessaro[4]

[1] New York University, dodis@cs.nyu.edu
[2] University of Wisconsin–Madison, rist@cs.wisc.edu
[3] Tsinghua University, jpsteinb@gmail.com
[4] Massachusetts Institute of Technology, tessaro@csail.mit.edu

Abstract. We show that the second iterate $H^2(M) = H(H(M))$ of a random oracle H cannot achieve strong security in the sense of indifferentiability from a random oracle. We do so by proving that indifferentiability for H^2 holds only with poor concrete security by providing a lower bound (via an attack) and a matching upper bound (via a proof requiring new techniques) on the complexity of any successful simulator. We then investigate HMAC when it is used as a general-purpose hash function with arbitrary keys (and not as a MAC or PRF with uniform, secret keys). We uncover that HMAC's handling of keys gives rise to two types of weak key pairs. The first allows trivial attacks against its indifferentiability; the second gives rise to structural issues similar to that which ruled out strong indifferentiability bounds in the case of H^2. However, such weak key pairs do not arise, as far as we know, in any deployed applications of HMAC. For example, using keys of any fixed length shorter than $d - 1$, where d is the block length in bits of the underlying hash function, completely avoids weak key pairs. We therefore conclude with a positive result: a proof that HMAC is indifferentiable from a RO (with standard, good bounds) when applications use keys of a fixed length less than $d - 1$.

Keywords: Indifferentiability, Hash functions, HMAC.

1 Introduction

Cryptographic hash functions such as those in the MD and SHA families are constructed by extending the domain of a fixed-input-length compression function via the Merkle-Damgård (MD) transform. This applies some padding to a message and then iterates the compression function over the resulting string to compute a digest value. Unfortunately, hash functions built this way are vulnerable to extension attacks that abuse the iterative structure underlying MD [22, 34]: given the hash of a message $H(M)$ an attacker can compute $H(M \| X)$ for some arbitrary X, even without knowing M.

In response, suggestions for shoring up the security of MD-based hash functions were made. The simplest is due to Ferguson and Schneier [20], who advocate a hash-of-hash construction: $H^2(M) = H(H(M))$, the second iterate of H. An earlier example is HMAC [5], which similarly applies a hash function H twice, and can be interpreted as giving a hash function with an additional key input. Both constructions enjoy many desirable features: they use H as a black box, do not add large overheads, and appear to prevent the types of extension attacks that plague MD-based hash functions.

Still, the question remains whether they resist other attacks. More generally, we would like that H^2 and HMAC behave like random oracles (ROs). In this paper, we provide the first analysis of these functions as being indifferentiable from ROs in the sense of [13, 29], which (if true) would provably rule out most structure-abusing attacks. Our main results surface a seemingly paradoxical fact, that the hash-of-hash H^2 cannot be indifferentiable from a RO with good bounds, *even* if H is itself modeled as a keyed RO. We then explore the fall out, which also affects HMAC.

Indifferentiability. Coron et al. [13] suggest that hash functions be designed so that they "behave like" a RO. To define this, they use the indifferentiability framework of Maurer et al. [29]. Roughly, this captures that no adversary can distinguish between a pair of oracles consisting of the construction (e.g., H^2) and its underlying ideal primitive (an ideal hash H) and the pair of oracles consisting of a RO and a simulator (which is given access to the RO). A formal definition is given in Section 2. Indifferentiability is an attractive goal because of the MRH composition theorem [29]: if a scheme is secure when using a RO it is also secure when the RO is replaced by a hash construction that is indifferentiable from a RO. The MRH theorem is widely applicable (but not ubiquitously, c.f., [31]), and so showing indifferentiability provides broad security guarantees.

While there exists a large body of work showing various hash constructions to be indifferentiable from a RO (c.f., [1, 7, 11–13, 15, 16, 23]), none have yet analyzed either H^2 or HMAC. Closest is the confusingly named HMAC construction from [13], which hashes a message by computing $H^2(0^d \parallel M)$ where H is MD using a compression function with block size d bits. This is not the same as HMAC proper nor H^2, but seems close enough to both that one would expect that the proofs of security given in [13] apply to all three.

1.1 The Second Iterate Paradox

Towards refuting the above intuition, consider that $H^2(H(M)) = H(H^2(M))$. This implies that an output of the construction $H^2(M)$ can be used as an intermediate value to compute the hash of the message $H(M)$. This property does not exist in typical indifferentiable hash constructions, which purposefully ensure that construction outputs are unlikely to coincide with intermediate values. However, and unlike where extension attacks apply (they, too, take advantage of outputs being intermediate values), there are no obvious ways to distinguish H^2 from a RO.

Our first technical contribution, then, is detailing how this structural property might give rise to vulnerabilities. Consider computing a hash chain of length ℓ using H^2 as the hash function. That is, compute $Y = H^{2\ell}(M)$. Doing so requires 2ℓ H-applications. But the structural property of H^2 identified above means that, given M and Y one can compute $H^{2\ell}(H(M))$ using only one H-application: $H(Y) = H(H^{2\ell}(M)) = H^{2\ell}(H(M))$. Moreover, the values computed along the first hash chain, namely the values $Y_i \leftarrow H^{2i}(M)$ and $Y_i' \leftarrow H^{2i}(H(M))$ for $0 \leq i \leq \ell$ are disjoint with overwhelming probability (when ℓ is not unreasonably large). Note that for chains of RO applications, attempting to cheaply compute such a second chain would not lead to disjoint chains. This demonstrates a way in which a RO and H^2 differ.

We exhibit a cryptographic setting, called *mutual proofs of work*, in which the highlighted structure of H^2 can be exploited. In mutual proofs of work, two parties prove to each other that they have computed some asserted amount of computational effort. This task is inspired by, and similar to, client puzzles [18, 19, 24, 25, 33] and puzzle auctions [35]. We give a protocol for mutual proofs of work whose computational task is computing hash chains. This protocol is secure when using a random oracle, but when using instead H^2 an attacker can cheat by abusing the structural properties discussed above.

Indifferentiability lower bound. The mutual proofs of work example already points to the surprising fact that H^2 does not "behave like" a RO. In fact, it does more, ruling out proofs of indifferentiability for H^2 with good bounds. (The existence of a tight proof of indifferentiability combined with the composition theorem of [29] would imply security for mutual proofs of work, yielding a contradiction.) However, we find that the example does not surface well why simulators must fail, and the subtlety of the issues here prompt further investigation. We therefore provide a direct negative result in the form of an indifferentiability distinguisher. We prove that should the distinguisher make q_1, q_2 queries to its two oracles, then for any simulator the indifferentiability advantage of the distinguisher is lower-bounded by $1 - (q_1 q_2)/q_S - q_S^2/2^n$. (This is slightly simpler than the real bound, see Section 3.2.) What this lower bound states is that the simulator must make very close to $\min\{q_1 q_2, 2^{n/2}\}$ queries to prevent this distinguisher's success. The result extends to structured underlying hash functions H as well, for example should H be MD-based.

To the best of our knowledge, our results are the first to show lower bounds on the number of queries an indifferentiability simulator must use. That a simulator must make a large number of queries hinders the utility of indifferentiability. When one uses the MRH composition theorem, the security of a scheme when using a monolothic RO must hold up to the number of queries the simulator makes. For example, in settings where one uses a hash function needing to be collision-resistant and attempts to conclude security via some (hypothetical) indifferentiability bound, our results indicate that the resulting security bound for the application can be at most $2^{n/4}$ instead of the expected $2^{n/2}$.

Upper bounds for second iterates. We have ruled out good upper bounds on indifferentiability, but the question remains whether weak bounds exist. We provide proofs of indifferentiability for H^2 that hold up to about $2^{n/4}$ distinguisher queries (our lower bounds rule out doing better) when H is a RO. We provide some brief intuition about the proof. Consider an indifferentiability adversary making at most q_1, q_2 queries. The adversarial strategy of import is to compute long chains using the left oracle, and then try to "catch" the simulator in an inconsistency by querying it on a value at the end of the chain and, afterwards, filling in the intermediate values via further left and right queries. But the simulator can avoid being caught if it prepares long chains itself to help it answer queries consistently. Intuitively, as long as the simulator's chains are a bit longer than q_1 hops, then the adversary cannot build a longer chain itself (being restricted to at most q_1 queries) and will never win. The full proofs of these results are quite involved, and so we defer more discussion until the body. We are unaware of any indifferentiability proofs that requires this kind of nuanced strategy by the simulator.

1.2 HMAC with Arbitrary Keys

HMAC was introduced by Bellare, Canetti, and Krawczyk [5] to be used as a pseudorandom function or message authentication code. It uses an underlying hash function H; let H have block size d bits and output length n bits. Computing a hash $\mathrm{HMAC}(K, M)$ works as follows [26]. If $|K| > d$ then redefine $K \leftarrow H(K)$. Let K' be K padded with sufficiently many zeros to get a d bit string. Then $\mathrm{HMAC}(K, M) = H(K' \oplus \mathsf{opad} \parallel H(K' \oplus \mathsf{ipad} \parallel M))$ where opad and ipad are distinct d-bit constants. The original (provable security) analyses of HMAC focus on the setting that the key K is honestly generated and secret [3, 5]. But what has happened is that HMAC's speed, ubiquity, and assumed security properties have lead it to be used in a wide variety of settings.

Of particular relevance are settings in which existing (or potential) proofs of security model HMAC as a keyed RO, a function that maps each key, message pair to an independent and uniform point. There are many examples of such settings. The HKDF scheme builds from HMAC a general-purpose key derivation function [27, 28] that uses as key a public, uniformly chosen salt. When used with a source of sufficiently high entropy, Krawczyk proves security using standard model techniques, but when not proves security assuming HMAC is a keyed RO [28]. PKCS#5 standardizes password-based key derivation functions that use HMAC with key being a (low-entropy) password [30]. Recent work provides the first proofs of security when modeling HMAC as a RO [9]. Ristenpart and Yilek [32], in the context of hedged cryptography [4], use HMAC in a setting whose cryptographic security models allow adversarially specified keys. Again, proofs model HMAC as a keyed RO.

As mentioned previously, we would expect a priori that one can show that HMAC is indifferentiable from a keyed RO even when the attacker can query arbitrary keys. Then one could apply the composition theorem of [29] to derive proofs of security for the settings just discussed.

WEAK KEY PAIRS IN HMAC. We are the first to observe that HMAC has weak key pairs. First, there exist $K \neq K'$ for which $\mathrm{HMAC}(K, M) = \mathrm{HMAC}(K', M)$.

These pairs of keys arise because of HMAC's ambiguous encoding of differing-length keys. Trivial examples of such "colliding" keys include any K, K' for which either $|K| < d$ and $K' = K \parallel 0^s$ (for any $1 \leq s \leq d - |K|$), or $|K| > d$ and $K' = H(K)$. Colliding keys enable an easy attack that distinguishes $\text{HMAC}(\cdot, \cdot)$ from a random function $\mathcal{R}(\cdot, \cdot)$, which also violates the indifferentiability of HMAC. On the other hand, as long as H is collision-resistant, two keys of the same length can never collide. Still, even if we restrict attention to (non-colliding) keys of a fixed length, there still exist weak key pairs, but of a different form that we term ambiguous. An example of an ambiguous key pair is K, K' of length d bits such that $K \oplus \mathsf{ipad} = K' \oplus \mathsf{opad}$. Because the second least significant bit of ipad and opad differ (see Section 4) and assuming $d > n - 2$, ambiguous key pairs of a fixed length k only exist for $k \in \{d - 1, d\}$. The existence of ambiguous key pairs in HMAC leads to negative results like those given for H^2. In particular, we straightforwardly extend the H^2 distinguisher to give one that lower bounds the number of queries any indifferentiability simulator must make for HMAC.

Upper bounds for HMAC. Fortunately, it would seem that weak key pairs do not arise in typical applications. Using HMAC with keys of some fixed bit length smaller than $d - 1$ avoids weak key pairs. This holds for several applications, for example the recommendation with HKDF is to use n-bit uniformly chosen salts as HMAC keys. This motivates finding positive results for HMAC when one avoids the corner cases that allow attackers to exploit weak key pairs.

Indeed, as our main positive result, we prove that, should H be a RO or an MD hash with ideal compression functions, *HMAC is indifferentiable from a keyed RO for all distinguishers that do not query weak key pairs.* Our result holds for the case that the keys queried are of length d or less. This upper bound enjoys the best, birthday-bound level of concrete security possible (up to small constants), and provides the *first positive result about the indifferentiability of the HMAC construction.*

1.3 Discussion

The structural properties within H^2 and HMAC are, in theory, straightforward to avoid. Indeed, as mentioned above, Coron et al. [13] prove indifferentiable from a RO the construction $H^2(0^d \parallel M)$ where H is MD using a compression function with block size d bits and chaining value length $n \leq d$ bits. Analogously, our positive results about HMAC imply as a special case that $\text{HMAC}(K, M)$, for any fixed constant K, is indifferentiable from a RO.

We emphasize that we are unaware of any deployed cryptographic application for which the use of H^2 or HMAC leads to a vulnerability. Still, our results show that future applications should, in particular, be careful when using HMAC with keys which are under partial control of the attacker. More importantly, our results demonstrate the importance of provable security in the design of hash functions (and elsewhere in cryptography), as opposed to the more common "attack-fix" cycle. For example, the hash-of-hash suggestion of Ferguson and Schneier [20] was motivated by preventing the extension attack. Unfortunately, in so doing they accidentally introduced a more subtle (although less dangerous) attack, which was not present on the original design.[5] Indeed, we discovered the subtlety of the problems within H^2 and HMAC, including our explicit attacks, only after attempting to prove indifferentiability of these constructions (with typical, good bounds). In contrast, the existing indifferentiability proofs of (seemingly) small modifications of these hash functions, such as $H^2(0^d \,\|\, M)$ [13], provably rule out these attacks.

1.4 Prior Work

There exists a large body of work showing hash functions are indifferentiable from a RO (c.f., [1, 7, 11–13, 15, 16, 23]), including analyses of variants of H^2 and HMAC. As mentioned, a construction called HMAC was analyzed in [13] but this construction is not HMAC as standardized. Krawczyk [28] suggests that the analysis of $H^2(0 \,\|\, M)$ extends to the case of HMAC, but does not offer proof.[6] HMAC has received much analysis in other contexts. Proofs of its security as a pseudorandom function under reasonable assumptions appear in [3, 5]. These rely on keys being uniform and secret, making the analyses inapplicable for other settings. Analysis of HMAC's security as a randomness extractor appear in [14, 21]. These results provide strong information theoretic guarantees that HMAC can be used as a key derivation function, but only in settings where the source has a relatively large amount of min-entropy. This requirement makes the analyses insufficient to argue security in many settings of practical importance. See [28] for further discussion.

Full version. Due to space constraints, many of our technical results and proofs are deferred to the full version of this paper [17].

2 Preliminaries

Notation and games. We denote the empty string by λ. If $|X| < |Y|$ then $X \oplus Y$ signifies that the X is padded with $|Y| - |X|$ zeros first. For set $\mathcal{X}$ and value x, we write $\mathcal{X} \xleftarrow{\cup} x$ to denote $\mathcal{X} \leftarrow \mathcal{X} \cup \{x\}$. For non-empty sets Keys, Dom, and Rng with $|\mathit{Rng}|$ finite, a random oracle $f\colon\ \mathit{Keys} \times \mathit{Dom} \to \mathit{Rng}$ is a function taken randomly from the space of all possible functions $\mathit{Keys} \times \mathit{Dom} \to \mathit{Rng}$. We will sometimes refer to random oracles as keyed when Keys is non-empty, whereas we omit the first parameter when $\mathit{Keys} = \emptyset$.

We use code-based games [10] to formalize security notions and within our proofs. In the execution of a game G with adversary $\mathcal{A}$, we denote by $G^{\mathcal{A}}$ the event that the game outputs true and by $\mathcal{A}^G \Rightarrow y$ the event that the adversary

[5] We note the prescience of the proposers of H^2, who themselves suggested further analysis was needed [20].

[6] Fortunately, the HKDF application of [28] seems to avoid weak key pairs, and thus our positive results for HMAC appear to validate this claim [28] for this particular application.

outputs y. Fixing some RAM model of computation, our convention is that the running time $\mathsf{Time}(\mathcal{A})$ of an algorithm $\mathcal{A}$ includes its code size. Queries are unit cost, and we will restrict attention to the absolute worst case running time which must hold regardless of queries are answered.

Hash functions. A *hash function* $H[P]\colon\ \mathit{Keys} \times \mathit{Dom} \to \mathit{Rng}$ is is a family of functions from Dom to Rng, indexed by a set Keys, that possibly uses (black-box) access to an underlying primitive P (e.g., a compression function). We call the hash function *keyed* if Keys is non-empty, and key-less otherwise. (In the latter case, we omit the first parameter.) We assume that the number of applications of P in computing $H[P](K, M)$ is the same for all K, M with the same value of $|K| + |M|$. This allows us to define the *cost* of computing a hash function $H[P]$ on a key and message whose combined length is ℓ, denoted $\mathsf{Cost}(H, \ell)$, as the number of calls to P required to compute $H[P](K, M)$ for K, M with $|K| + |M| = \ell$. For a keyed random oracle $\mathcal{R}\colon\ \mathit{Keys} \times \mathit{Dom} \to \mathit{Rng}$, we fix the convention that $\mathsf{Cost}(\mathcal{R}, \ell) = 1$ for any ℓ for which there exists a key $K \in \mathit{Keys}$ and message $M \in \mathit{Dom}$ such that $|K| + |M| = \ell$.

A *compression function* is a hash function for which $\mathit{Dom} = \{0,1\}^n \times \{0,1\}^d$ and $\mathit{Rng} = \{0,1\}^n$ for some numbers $n, d > 0$. Our focus will be on keyless compression functions, meaning those of the form $f\colon\ \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^n$. Our results lift in a straightforward way to the dedicated-key setting [8]. The ℓ-th iterate of $H[P]$ is denoted $H^\ell[P]$, and defined for $\ell > 0$ by $H^\ell[P](X) = H[P](H[P](\cdots H[P](X))\cdots)$ where the number of applications of H is ℓ. We let $H^0[P](X) = X$. We will often write H instead of $H[P]$ when the underlying primitive P is clear or unimportant.

Merkle-Damgård. Let $\mathsf{Pad}\colon\ \{0,1\}^{\leq L} \to (\{0,1\}^n)^+$ be an injective padding function. The one used in many of the hash functions within the SHA family outputs $M \parallel 10^r \parallel \langle |M| \rangle_{64}$ where $\langle |x| \rangle_{64}$ is the encoding of the length of M as a 64-bit string and r is the smallest number making the length a multiple of d. This makes $L = 2^{64} - 1$. The function $\mathrm{MD}[f]\colon\ (\{0,1\}^n)^+ \to \{0,1\}^n$ is defined as

$$\mathrm{MD}[f](M) = f(f(\cdots f(f(IV, M_1), M_2), \cdots), M_k)$$

where $|M| = kd$ and $M_1 \parallel \cdots \parallel M_k$. The function $\mathrm{SMD}[f]\colon\ \{0,1\}^{\leq L} \to \{0,1\}^n$ is defined as $\mathrm{SMD}[f](M) = \mathrm{MD}[f](\mathsf{Pad}(M))$.

Indifferentiability from a RO. Let $\mathcal{R}\colon\ \mathit{Keys} \times \mathit{Dom} \to \mathit{Rng}$ be a random oracle. Consider a hash construction $H[P]\colon\ \mathit{Keys} \times \mathit{Dom} \to \mathit{Rng}$ from an ideal primitive P. Let game $\mathrm{Real}_{H[P]}$ be the game whose main procedure runs an adversary $\mathcal{A}^{\mathsf{Func},\mathsf{Prim}}$ and returns the bit that $\mathcal{A}$ outputs. The procedure Func on input $K \in \mathit{Keys}$ and $M \in \mathit{Dom}$ returns $H[P](K, M)$. The procedure Prim on input X returns $P(X)$. For a simulator $\mathcal{S}$, let game $\mathrm{Ideal}_{\mathcal{R},\mathcal{S}}$ be the game whose main procedure runs an adversary $\mathcal{A}^{\mathsf{Func},\mathsf{Prim}}$ and returns the bit that $\mathcal{A}$ outputs. The procedure Func on input $K \in \mathit{Keys}$ and $M \in \mathit{Dom}$ returns $\mathcal{R}(K, M)$. The procedure Prim on input X returns $\mathcal{S}^{\mathcal{R}}(X)$. The indifferentiability advantage of $\mathcal{D}$ is defined as

$$\mathbf{Adv}^{\mathrm{indiff}}_{H[P],\mathcal{R},\mathcal{S}}(\mathcal{D}) = \Pr\left[\,\mathrm{Real}^{\mathcal{D}}_{H[P]} \Rightarrow y\,\right] - \Pr\left[\,\mathrm{Ideal}^{\mathcal{D}}_{\mathcal{R},\mathcal{S}} \Rightarrow y\,\right]\ .$$

We focus on simulators that must work for any adversary, though our negative results extend as well to the weaker setting in which the simulator can depend on the adversary. The total query cost σ of an adversary $\mathcal{D}$ is the cumulative cost of all its Func queries plus q_2. (This makes σ the total number of P uses in game $\mathrm{Real}_{H[P]}$. In line with our worst-case conventions, this means the same maximums hold in $\mathrm{Ideal}_{\mathcal{R},\mathcal{S}}$ although here it does not translate to P applications.)

We note that when *Keys* is non-empty, indifferentiability here follows [8] and allows the distinguisher to choose keys during an attack. This reflects the desire for a keyed hash function to be indistinguishable from a keyed random oracle for arbitrary uses of the key input.

3 Second Iterates and their Security

Our investigation begins with the second iterate of a hash function, meaning $H^2(M) = H(H(M))$ where $H\colon\ \mathit{Dom} \to \mathit{Rng}$ for sets $\mathit{Dom} \supseteq \mathit{Rng}$. For simplicity, let $\mathit{Rng} = \{0,1\}^n$ and assume that H is itself modeled as a RO. Is H^2 good in the sense of being like a RO? Given that we are modeling H as a RO, we would expect that the answer would be "yes". The truth is more involved. As we'll see in Section 4, similar subtleties exist in the case of the related HMAC construction.

We start with the following observations. When computing $H^2(M)$ for some M, we refer to the value $H(M)$ as an intermediate value. Then, we note that the value $Y = H^2(M)$ is in fact the intermediate value used when computing $H^2(X)$ for $X = H(M)$. Given $Y = H^2(M)$, then, one can compute $H^2(H(M))$ directly by computing $H(Y)$. That the hash value Y is also the intermediate value used in computing the hash of another message is cause for concern: other hash function constructions that are indifferentiable from a RO (c.f., [2, 7, 8, 13, 23]) explicitly attempt to ensure that outputs are *not* intermediate values (with overwhelming probability over the randomness of the underlying idealized primitive). Moreover, prior constructions for which hash values are intermediate values have been shown to *not* be indifferentiable from a RO. For example Merkle-Damgård-based iterative hashes fall to extension attacks [13] for this reason. Unlike with Merkle-Damgård, however, it is not immediately clear how an attacker might abuse the structure of H^2.

We turn our attention to hash chains, where potential issues arise. For a hash function H, we define a hash chain $Y = (Y_0, \ldots, Y_\ell)$ to be a sequence of $\ell + 1$ values where Y_0 is a message and $Y_i = H(Y_{i-1})$ for $1 \le i \le \ell$. Likewise when using H^2 a hash chain $Y = (Y_0, \ldots, Y_\ell)$ is a sequence of $\ell + 1$ values where Y_0 is a message and $Y_i = H^2(Y_{i-1})$ for $1 \le i \le \ell$. We refer to Y_0 as the *start* of the hash chain and Y_ℓ as the *end*. Two chains Y, Y' are *non-overlapping* if no value in one chain occurs in the other, meaning $Y_i \ne Y'_j$ for all $0 \le i \le j \le \ell$.

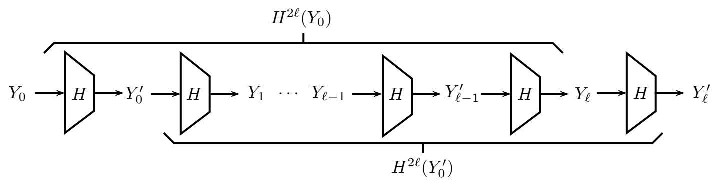


Fig. 1. Diagram of two hash chains $Y = (Y_0, \ldots, Y_\ell)$ and $Y' = (Y_0', \ldots, Y_\ell')$ for hash function H^2.

For any hash function and given the start and end of a hash chain $Y = (Y_0, \ldots, Y_\ell)$, one can readily compute the start and end of a new chain with just two hash calculations. That is, set $Y_0' \leftarrow H(Y_0)$ and $Y_\ell' \leftarrow H(Y_\ell)$. However, the chain $Y' = (Y_0', \ldots, Y_\ell')$ and the chain Y overlap. For good hash functions (i.e., ones that behave like a RO) computing the start and end of a non-overlapping chain given the start and end of a chain Y_0, Y_ℓ requires at least ℓ hash computations (assuming $\ell \ll 2^{n/2}$).

Now consider H^2. Given the start and end of a chain $Y = (Y_0, \ldots, Y_\ell)$, one can readily compute a non-overlapping chain $Y' = (Y_0', \ldots, Y_\ell')$ using just two hash computations instead of the expected 2ℓ computations. Namely, let $Y_0' \leftarrow H(Y_0)$ and $Y_\ell' \leftarrow H(Y_\ell)$. Then these are the start and end of the chain $Y' = (Y_0', \ldots, Y_\ell')$ because

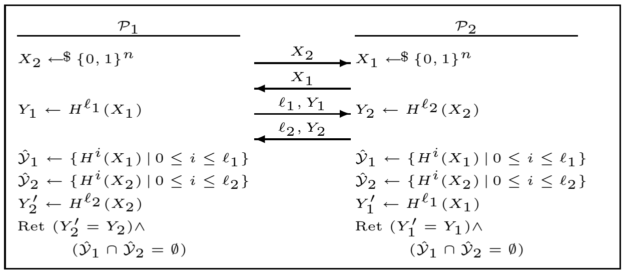
$$H^{2\ell}(Y_0') = H^{2\ell}(H(Y_0)) = H(H^{2\ell}(Y_0))$$

which we call the chain-shift property of H^2. Moreover, assuming H is itself a RO outputing n-bit strings, the two chains Y, Y' do not overlap with probability at least $1 - (2\ell+2)^2/2^n$. Figure 1 provides a pictoral diagram of the two chains Y and Y'.

3.1 A Vulnerable Application: Mutual Proofs of Work

In the last section we saw that the second iterate fails to behave like a RO in the context of hash chains. But the security game detailed in the last section may seem far removed from real protocols. For example, it's not clear where an attacker would be tasked with computing hash chains in a setting where it, too, was given an example hash chain. We suggest that just such a setting could arise in protocols in which parties want to assert to each other, in a verifiable way, that they performed some amount of computation. Such a setting could arise when parties must (provably) compare assertions of computational power, as when using cryptographic puzzles [18, 19, 24, 25, 33, 35]. Or this might work when trying to verifiably calibrate differing computational speeds of the two parties' computers. We refer to this task as a *mutual proof of work*.

MUTUAL PROOFS-OF-WORK. For the sake of brevity, we present an example hash-chain-based protocol and dispense with a more general treatment of mutual proofs of work. Consider the two-party protocol shown in the left diagram


$\mathcal{P}_1$
$\mathcal{P}_2$
$X_2 \twoheadleftarrow\!\$ \; \{0,1\}^n$
X_2
$X_1 \twoheadleftarrow\!\$ \; \{0,1\}^n$
X_1
$Y_1 \leftarrow H^{\ell_1}(X_1)$
ℓ_1, Y_1
$Y_2 \leftarrow H^{\ell_2}(X_2)$
ℓ_2, Y_2
$\hat{\mathcal{Y}}_1 \leftarrow \{H^i(X_1) \mid 0 \le i \le \ell_1\}$
$\hat{\mathcal{Y}}_2 \leftarrow \{H^i(X_2) \mid 0 \le i \le \ell_2\}$
$Y_2' \leftarrow H^{\ell_2}(X_2)$
Ret $(Y_2' = Y_2) \wedge (\hat{\mathcal{Y}}_1 \cap \hat{\mathcal{Y}}_2 = \emptyset)$
$\hat{\mathcal{Y}}_1 \leftarrow \{H^i(X_1) \mid 0 \le i \le \ell_1\}$
$\hat{\mathcal{Y}}_2 \leftarrow \{H^i(X_2) \mid 0 \le i \le \ell_2\}$
$Y_1' \leftarrow H^{\ell_1}(X_1)$
Ret $(Y_1' = Y_1) \wedge (\hat{\mathcal{Y}}_1 \cap \hat{\mathcal{Y}}_2 = \emptyset)$

```
main POW_{H[P],n,ℓ_1}:
$X_2 \twoheadleftarrow\!\$ \; \{0,1\}^n$
$X_1 \twoheadleftarrow\!\$ \; \mathcal{A}^{\mathsf{Prim}}(X_2)$
$Y_1 \leftarrow H^{\ell_1}[P](X_1)$
$(\ell_2, Y_2) \twoheadleftarrow\!\$ \; \mathcal{A}^{\mathsf{Prim}}(\ell_1, Y_1)$
$\hat{\mathcal{Y}}_1 \leftarrow \{H^i[P](X_1) \mid 0 \le i \le \ell_1\}$
$\hat{\mathcal{Y}}_2 \leftarrow \{H^i[P](X_2) \mid 0 \le i \le \ell_2\}$
$Y_2' \leftarrow H^{\ell_2}[P](X_2)$
If $q \ge \ell_2 \cdot \mathsf{Cost}(H,n)$ then
    Ret false
Ret $(Y_2' = Y_2 \wedge \hat{\mathcal{Y}}_1 \cap \hat{\mathcal{Y}}_2 = \emptyset)$

subroutine Prim($u$)
$q \leftarrow q+1$ ; Ret $P(u)$
```

Fig. 2. Example protocol **(left)** and adversarial $\mathcal{P}_2$ security game **(right)** for mutual proofs of work.

of Figure 2. Each party initially chooses a random nonce and sends it to the other. Then, each party computes a hash chain of some length —chosen by the computing party— starting with the nonce chosen by the other party, and sends the chain's output along with the chain's length to the other party. At this point, both parties have given a witness that they performed a certain amount of work. So now, each party checks the other's asserted computation, determining if the received value is the value resulting from chaining together the indicated number of hash applications and checking that the hash chains used by each party are non-overlapping. Note that unlike puzzles, which require fast verification, here the verification step is as costly as puzzle solution.

The goal of the protocol is to ensure that the other party did compute exactly their declared number of iterations. Slight changes to the protocol would lead to easy ways of cheating. For example, if during verification the parties did not check that the chains are non-overlapping, then $\mathcal{P}_2$ can easily cheat by choosing X_1 so that it can reuse a portion of the chain computed by $\mathcal{P}_1$

Security would be achieved should no cheating party succeed at convincing an honest party using less than ℓ_1 (resp. ℓ_2) work to compute Y_1 (resp. Y_2). The game $\mathrm{POW}_{H[P],n,\ell_1}$ formalizes this security goal for a cheating $\mathcal{P}_2$; see the right portion of Figure 2. We let $\mathbf{Adv}^{\mathrm{pow}}_{H[P],n,\ell_1}(\mathcal{A}) = \Pr\left[\,\mathrm{POW}^{\mathcal{A}}_{H[P],n,\ell_1}\,\right]$. Note that the adversary $\mathcal{A}$ only wins should it make $q < \ell_2 \cdot \mathsf{Cost}(H,n)$ queries, where ℓ_2 is the value it declared and $\mathsf{Cost}(H)$ is the cost of computing H. Again we will consider both the hash function $H[P](M) = P(M)$ that just applies a RO P and also $H^2[P](M) = P(P(M))$, the second iterate of a RO. In the former case the can make only $\ell_2 - 1$ queries and in the latter case $2\ell_2 - 1$.

When $H[P](M) = P(M)$, no adversary making $q < \ell_2$ queries to Prim can win the $\mathrm{POW}_{H[P],n,\ell_1}$ game with high advantage. Intuitively, the reason is that, despite being given X_1 and Y_1 where $Y_1 = P^{\ell_1}(X_1)$, a successful attacker must still compute a full ℓ_2-length chain and this requires ℓ_2 calls to P. A more formal treatment appears in the full version.

Attack against any second iterate. Now let us analyze this protocol's security when we use as hash function $H^2[P] = P(P(M))$ for a RO $P\colon\ \mathit{Dom} \to \mathit{Rng}$ with $\mathit{Rng} \subseteq \mathit{Dom}$. We can abuse the chain-shift property of H^2 in order to win the $\mathrm{POW}_{H^2,P,n,\ell_1}$ game for any $n > 0$ and $\ell_1 > 2$. Our adversary $\mathcal{A}$ works as follows. It receives X_2 and then chooses it's nonce as $X_1 \leftarrow \mathsf{Prim}(X_2)$. When it later receives $Y_1 = P^{2\ell_1}(X_1)$, the adversary proceeds by setting $\ell_2 = \ell_1 + 1$ and setting $Y_2 \leftarrow \mathsf{Prim}(Y_1)$. Then by the chain-shift property we have that

$$Y_2 = P(Y_1) = P(P^{2\ell_1}(X_1)) = P(P^{2\ell_1}(P(X_2))) = P^{2\ell_2}(X_2)\ .$$

The two chains will be non-overlapping with high probability (over the coins used by P). Finally, $\mathcal{A}$ makes only 2 queries to Prim, so the requirement that $q < 2\ell_2$ is met whenever $\ell_1 > 1$.

Discussion. As far as we are aware, mutual proofs of work have not before been considered — the concept may indeed be of independent interest. A full treatment is beyond the scope of this work. We also note that, of course, it is easy to modify the protocols using H^2 to be secure. Providing secure constructions was not our goal, rather we wanted to show protocols which are insecure using H^2 but secure when H^2 is replaced by a monolothic RO. This illustrates how, hypothetically, the structure of H^2 could give rise to subtle vulnerabilities in an application.

3.2 Indifferentiability Lower and Upper Bounds

In this section we prove that *any* indifferentiability proof for the double iterate H^2 is subject to inherent quantitative limitations. Recall that indifferentiability asks for a simulator $\mathcal{S}$ such that no adversary can distinguish between the pair of oracles $H^2[P], P$ and $\mathcal{R}, \mathcal{S}$ where P is some underlying ideal primitive and $\mathcal{R}$ is a RO with the same domain and range as H^2. The simulator can make queries to $\mathcal{R}$ to help it in its simulation of P. Concretely, building on the ideas behind the above attacks in the context of hash chains, we show that in order to withstand a differentiating attack with q queries, any simulator for $H^2[P]$, for *any* hash construction $H[P]$ with output length n, must issue *at least* $\Omega(\min\{q^2, 2^{n/2}\})$ queries to the RO $\mathcal{R}$. As we explain below, such a lower bound severely limits the concrete security level which can be inferred by using the composition theorem for indifferentiability, effectively neutralizing the benefits of using indifferentiability in the first place.

The distinguisher. In the following, we let $H = H[P]$ be an *arbitrary* hash function with n-bit outputs relying on a primitive P, such as a fixed input-length random oracle or an ideal cipher. We are therefore addressing an arbitrary second iterate, and not focusing on some particular ideal primitive P (such as a RO as in previous sections) or construction H. Indeed, H could equally well be Merkle-Damgård and P an ideal compression function, or H could be any number of indifferentiable hash constructions using appropriate ideal primitive P.

Recall that Func and Prim are the oracles associated with construction and primitive queries to $H^2 = H^2[P]$ and P, respectively. Let w, ℓ be parameters (for now, think for convenience of $w = \ell$). The attacker $\mathcal{D}_{w,\ell}$ starts by issuing ℓ queries to Func to compute a *chain* of n-bit values $(x_0, x_1, \ldots, x_\ell)$ where $x_i = H^2(x_{i-1})$ and x_0 is a random n-bit string. Then, it also picks a random index $j \in [1 \mathinner{..} w]$, and creates a list of n-bit strings $\mathbf{u}[1], \ldots, \mathbf{u}[w]$ with $\mathbf{u}[j] = x_\ell$, and all remaining $\mathbf{u}[i]$ for $i \neq j$ are chosen uniformly and independently. Then, for all $i \in [1 \mathinner{..} w]$, the distinguisher $\mathcal{D}_{w,\ell}$ proceeds in asking all Prim queries in order to compute $\mathbf{v}[i] = H(\mathbf{u}[i])$. Subsequently, the attacker compute $y_0 = H(x_0)$ via Prim queries, and also computes the chain $(y_0, y_1, \ldots, y_\ell)$ such that $y_i = H^2(y_{i-1})$ by making ℓ Func queries. Finally, it decides to output 1 if and only if $y_\ell = \mathbf{v}[j]$ and x_ℓ as well as $\mathbf{v}[i]$ for $i \neq j$ are not in $\{y_0, y_1, \ldots, y_\ell\}$. The attacker $\mathcal{D}_{w,\ell}$ therefore issues a total of 2ℓ Func queries and $(2w + 1) \cdot \mathsf{Cost}(H, n)$ Prim queries.

In the real-world experiment, the distinguisher $\mathcal{D}_{w,\ell}$ outputs 1 with very high probability, as the condition $y_\ell = \mathbf{v}[j]$ *always* holds by the chain-shifting property of H^2. In fact, the only reason for $\mathcal{D}$ outputting 0 is that one of x_ℓ and $\mathbf{v}[i]$ for $i \neq j$ incidentally happens to be in $\{y_0, y_1, \ldots, y_\ell\}$. The (typically small) probability that this occurs obviously depends on the particular construction $H[P]$ at hand; it is thus convenient to define the shorthand

$$p(H, w, \ell) = \Pr\left[\, \{x_\ell, H(U_1), \ldots, H(U_{w-1})\} \cap \{y_0, y_1, \ldots, y_\ell\} \neq \emptyset \,\right] ,$$

where $x_0, y_0, x_1, \ldots, y_{\ell-1}, x_\ell, y_\ell$ are the intermediate value of a chain of $2\ell + 1$ consecutive evaluations of $H[P]$ starting at a random n-bit string x_0, and $U_1, \ldots, U_{w-1}$ are further independent random n-bit values. In the full version of this paper we prove that for $H[P] = P = \mathcal{R}$ for a random oracle $\mathcal{R} : \{0,1\}^* \to \{0,1\}^n$ we have $p(H, w, \ell) = \Theta((w\ell + \ell^2)/2^n)$. Similar reasoning can be applied to essentially all relevant constructions.

In contrast, in the ideal-world experiment, we expect the simulator to be completely ignorant about the choice of j as long as it does not learn x_0, and in particular it does not know j while answering the Prim queries associated with the evaluations of $H(\mathbf{u}[i])$. Consequently, the condition required for $\mathcal{D}_{w,\ell}$ to output 1 appears to force the simulator, for all $i \in [1 \mathinner{..} w]$, to prepare a distinct chain of ℓ consecutive $\mathcal{R}$ evaluations ending in $\mathbf{v}[i]$, hence requiring $w \cdot \ell$ random oracle queries.

The following theorem quantifies the advantage achieved by the above distinguisher $\mathcal{D}_{w,\ell}$ in differentiating against any simulator for the construction $H[P]$. Its proof is given in the full version.

Theorem 1. [Attack against H^2] *Let $H[P]$ be an arbitrary hash construction with n-bit outputs, calling a primitive P, and let $\mathcal{R} : \{0,1\}^* \to \{0,1\}^n$ be a random oracle. For all integer parameters $w, \ell \geq 1$, there exists an adversary $\mathcal{D}_{w,\ell}$ making 2ℓ Func-queries and $(w+1) \cdot \mathsf{Cost}(H, n)$ Prim-queries such that for all simulators $\mathcal{S}$,*

$$\mathbf{Adv}^{\mathrm{indiff}}_{H^2[P],\mathcal{R},\mathcal{S}}(\mathcal{D}_{w,\ell}) \geq 1 - p(H, w, \ell) - \frac{5\ell^2}{2^{n+1}} - \frac{q_{\mathcal{S}}\ell}{2^n} - \frac{q_{\mathcal{S}}^2}{2^n} - \frac{q_{\mathcal{S}}}{w \cdot \ell} - \frac{1}{w} ,$$

where $q_{\mathcal{S}}$ is the overall number of $\mathcal{R}$ queries by $\mathcal{S}$ when replying to $\mathcal{D}_{w,\ell}$'s Prim queries. ▮

Discussion. We now elaborate on Theorem 1. If we consider the distinguisher $\mathcal{D}_{w,\ell}$ from Theorem 1, we observe that by the advantage lower bound in the theorem statement, if $\ell, w \ll 2^{n/4}$ and consequently $p(H, w, \ell) \approx 0$, the number of queries made by the simulator, denoted $q_{\mathcal{S}} = q_{\mathcal{S}}(2\ell, w+1)$ must satisfy $q_{\mathcal{S}} = \Omega(w \cdot \ell) = \Omega(q_1 \cdot q_2)$ to ensure a sufficiently small indifferentiability advantage. This in particular means that in the case where both q_1 and q_2 are large, the simulator must make a *quadratic* effort to prevent the attacker from distinguishing. Below, in Theorem 2, we show that this simulation effort is essentially optimal.

In many scenarios, this quadratic lower bound happens to be a problem, as we now illustrate. As a concrete example, let $\mathcal{SS} = (\mathsf{key}, \mathsf{sign}, \mathsf{ver})$ be an arbitrary signature scheme signing n bits messages, and let $\widetilde{\mathcal{SS}}[\mathcal{R}] = (\widetilde{\mathsf{key}}^{\mathcal{R}}, \widetilde{\mathsf{sign}}^{\mathcal{R}}, \widetilde{\mathsf{ver}}^{\mathcal{R}})$ for $\mathcal{R} : \{0,1\}^* \to \{0,1\}^n$ be the scheme obtained via the hash-then-sign paradigm such that $\widetilde{\mathsf{sign}}^{\mathcal{R}}(sk, m) = \mathsf{sign}(sk, \mathcal{R}(m))$. It is well known that for an adversary $\mathcal{B}$ making q_{sign} signing and $q_{\mathcal{R}}$ random oracle queries, there exists an adversary $\mathcal{C}$ making q_{sign} signing queries such that

$$\mathbf{Adv}^{\text{uf-cma}}_{\widetilde{\mathcal{SS}}[\mathcal{R}]}(\mathcal{B}^{\mathcal{R}}) \leq \frac{(q_{\mathsf{sign}} + q_{\mathcal{R}})^2}{2^n} + \mathbf{Adv}^{\text{uf-cma}}_{\mathcal{SS}}(\mathcal{C}) \ , \tag{1}$$

where $\mathbf{Adv}^{\text{uf-cma}}_{\widetilde{\mathcal{SS}}[\mathcal{R}]}(\mathcal{B}^{\mathcal{R}})$ and $\mathbf{Adv}^{\text{uf-cma}}_{\mathcal{SS}}(\mathcal{C})$ denote the respective advantages in the standard uf-cma game for security of signature schemes (with and without a random oracle, respectively). This in particular means that $\widetilde{\mathcal{SS}}$ is secure for q_{sign} and $q_{\mathcal{R}}$ as large as $\Theta(2^{n/2})$, provided $\mathcal{SS}$ is secure for q_{sign} signing queries. However, let us now replace $\mathcal{R}$ by $H^2[P]$ for an arbitrary construction $H = H[P]$. Then, for all adversaries $\mathcal{A}$ making q_P queries to P and q_{sign} signing queries, we can combine the concrete version of the MRH composition theorem proven in [31] and (1) to infer that there exists an adversary $\mathcal{C}$ and a distinguisher $\mathcal{D}$ such that

$$\mathbf{Adv}^{\text{uf-cma}}_{\widetilde{\mathcal{SS}}[H^2[P]]}(\mathcal{A}^P) \leq \Theta\left(\frac{(q_{\mathsf{sign}} \cdot q_P)^2}{2^n}\right) + \mathbf{Adv}^{\text{uf-cma}}_{\mathcal{SS}}(\mathcal{C}) + \mathbf{Adv}^{\text{indiff}}_{H^2[P],\mathcal{R},\mathcal{S}}(\mathcal{D}) \ ,$$

where $\mathcal{C}$ makes q_{sign} signing queries . Note that even if the term $\mathbf{Adv}^{\text{indiff}}_{H^2[P],\mathcal{R},\mathcal{S}}(\mathcal{D})$ is really small, this new bound can only ensure security for the resulting signature scheme as long as $q_{\mathsf{sign}} \cdot q_P = \Theta(2^{n/2})$, i.e., if $q_{\mathsf{sign}} = q_P$, we only get security up to $\Theta(2^{n/4})$ queries, a remarkable loss with respect to the security bound in the random oracle model.

We note that of course this does *not* mean that $H^2[P]$ for a concrete H and P is unsuitable for a certain application, such as hash-then-sign. In fact, $H^2[P]$ may well be optimally collision resistant. However, our result shows that a sufficiently strong security level cannot be inferred from *any* indifferentiability statement via the composition theorem, taking us back to a direct ad-hoc analysis and completely loosing the one main advantage of having indifferentiability in the first place.

Upper bound. Our negative results do not rule out positive results completely: there could be indifferentiability upper bounds, though for simulators that make around $\mathcal{O}(q^2)$ queries. Ideally, we would like upper bounds that match closely the lower bounds given in prior sections. We do so for the special case of $H^2[g](M) = g(g(M))$ for $g\colon \{0,1\}^n \to \{0,1\}^n$ being a RO.

Theorem 2. *Let $q_1, q_2 \geq 0$ and $N = 2^n$. Let $g : \{0,1\}^n \to \{0,1\}^n$ and $\mathcal{R} : \{0,1\}^n \to \{0,1\}^n$ be uniform random functions. Then there exists a simulator $\mathcal{S}$ such that*

$$\mathbf{Adv}^{\mathrm{indiff}}_{G[g],\mathcal{R},\mathcal{S}}(\mathcal{D}) \leq \frac{2((4q_1+3)q_2+2q_1)^2}{N} + \frac{2((4q_1+3)q_2+2q_1)(q_1+q_2)}{(N-2q_2-2q_1)}$$

for any adversary $\mathcal{D}$ making at most q_1 queries to its left oracle and at most q_2 queries to its right oracle. Moreover, for each query answer that it computes, $\mathcal{S}$ makes at most $3q_1+1$ queries to RO *and runs in time $\mathcal{O}(q_1)$.* $\square$

The proof of the theorem appears in the full version of the paper. We note that the simulator used must know the maximum number of queries the attacker will make, but does not otherwise depend on the adversary's strategy. The security bound of the theorem is approximately $(q_1q_2)^2/N$, implying that security holds up to $q_1q_2 \approx 2^{n/2}$.

4 HMAC as a General-purpose Keyed Hash Function

HMAC [5] uses a hash function to build a keyed hash function, i.e. one that takes both a key and message as input. Fix some hash function[7] $H\colon \{0,1\}^* \to \{0,1\}^n$. HMAC assumes this function H is built by iterating an underlying compression function with a message block size of $d \geq n$ bits. We define the following functions:

$$F_K(M) = H((\rho(K) \oplus \mathsf{ipad}) \parallel M) \qquad \text{where } \rho(K) = \begin{cases} H(K) & \text{if } |K| > d \\ K & \text{otherwise.} \end{cases}$$
$$G_K(M) = H((\rho(K) \oplus \mathsf{opad}) \parallel M)$$

The two constants used are $\mathsf{ipad} = \mathsf{0x36}^{d/8}$ and $\mathsf{opad} = \mathsf{0x5c}^{d/8}$. These constants are given in hexadecimal, translating to binary gives $\mathsf{0x36} = \mathsf{0011\,0110}_2$ and $\mathsf{0x5c} = \mathsf{0101\,1100}_2$. Recall that we have defined the $\oplus$ operator so that, if $|K| < d$, it first silently pads out the shorter string by sufficiently many zeros before computing the bitwise xor. It will also be convenient to define $\mathsf{xpad} = \mathsf{ipad} \oplus \mathsf{opad}$. The function $\mathrm{HMAC}\colon \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^n$ is defined by

$$\mathrm{HMAC}(K, M) = G_K(F_K(M)) = (G_K \circ F_K)(M)\ .$$

We sometimes write $\mathrm{HMAC}_d[P]$, HMAC_d, or $\mathrm{HMAC}[P]$ instead of HMAC when we want to make the reliance on the block size and/or an underlying ideal primitive explicit.

[7] RFC 2104 defines HMAC over strings of bytes, but we chose to use bits to provide more general positive results — all our negative results lift to a setting in which only byte strings are used. Note also that for simplicity we assumed H with domain $\{0,1\}^*$. In practice hash functions often do have some maximal length (e.g., 2^{64}), and in this case HMAC must be restricted to smaller lengths.

In the following sections, we will therefore analyze the security of HMAC in the sense of being indifferentiable from a keyed RO. As we will see, the story is more involved than one might expect.

4.1 Weak Key Pairs in HMAC

Towards understanding the indifferentiability of HMAC, we start by observing that the way HMAC handles keys gives rise to two worrisome classes of weak key pairs.

Colliding keys. We say that keys $K \neq K'$ *collide* if $\rho(K) \parallel 0^{d-|\rho(K)|} = \rho(K') \parallel 0^{d-|\rho(K')|}$. For any message M and colliding keys K, K' it holds that $\mathrm{HMAC}(K, M) = \mathrm{HMAC}(K', M)$. Colliding keys exist because of HMAC's ambiguous encoding of different-length keys. Examples of colliding keys include any K, K' for which $|K| < d$ and $K' = K \parallel 0^s$ where $1 \leq s \leq d - |K|$. Or any K, K' such that $|K| > d$ and $K' = H(K)$. As long as H is collision-resistant, two keys of the same length can never collide.

Colliding keys enable a simple attack against indifferentiability. Consider $\mathrm{HMAC}[P]$ for any underlying function P. Then let $\mathcal{A}$ pick two keys $K \neq K'$ that collide and an arbitrary message M. It queries its Func oracle on (K, M) and (K', M) to retrieve two values Y, Y'. If $Y = Y'$ then it returns 1 (guessing that it is in game $\mathrm{Real}_{\mathrm{HMAC}[P],\mathcal{R}}$) and returns 0 otherwise (guessing that it is in game $\mathrm{Ideal}_{\mathcal{R},\mathcal{S}}$). The advantage of $\mathcal{A}$ is equal to $1 - 2^n$ regardless of the simulator $\mathcal{S}$, which is never invoked.

Note that this result extends directly to rule out related-key attack security [6] of HMAC as a PRF should a related-key function be available that enables deriving colliding keys.

Ambiguous keys. A pair of keys $K \neq K'$ is *ambiguous* if $\rho(K) \oplus \mathsf{ipad} = \rho(K') \oplus \mathsf{opad}$. For any X, both $F_K(X) = G_{K'}(X)$ and $G_K(X) = F_{K'}(X)$ when K, K' are ambiguous. An example such pair is K, K' of length d bits for which $K \oplus K' = \mathsf{xpad}$.

For any key K, there exists one key K' that is easily computable and for which K, K' are ambiguous: set $K' = \rho(K) \oplus \mathsf{xpad}$. Finding a third key K'' that is also ambiguous with K is intractable should H be collision resistant. The easily-computable K' will not necessarily have the same length as K. In fact, there exist ambiguous key pairs of the same length k only when $k \in \{d-1, d\}$. For a fixed length shorter than $d-1$, no ambiguous key pairs exist due to the fact that the second least significant bit of xpad is 1. For a fixed length longer than d bits, if $n < d - 1$ then no ambiguous key pairs exist and if $n \geq d - 1$ then producing ambiguous key pairs would require finding K, K' such that $H(K) \oplus H(K')$ equals the first n bits of xpad. This is intractable for any reasonable hash function H.

Ambiguous key pairs give rise to a chain-shift like property. Let M be some message and K, K' be an ambiguous key pair. Then, we have that $\rho(K') = \rho(K) \oplus \mathsf{xpad}$ and so $F_K(M) = G_{K'}(M)$. Thus,

$$\mathrm{HMAC}(K', F_K(M)) = G_{K'}(\mathrm{HMAC}(K, M)) \ .$$

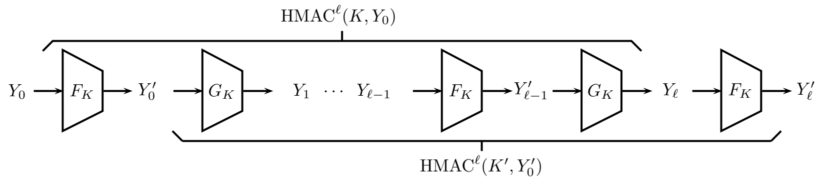


Fig. 3. Diagram of two hash chains $(K, Y) = (Y_0, \ldots, Y_\ell)$ and $(K', Y') = (Y'_0, \ldots, Y'_\ell)$ for HMAC where $\rho(K') = \rho(K) \oplus \mathsf{xpad}$.

As with H^2, this property gives rise to problems in the context of hash chains. A hash chain $Y = (K, Y_0, \ldots, Y_\ell)$ is a key K, a message Y_0, and a sequence of ℓ values $Y_i = H(K, Y_{i-1})$ for $1 \le i \le \ell$. So a keyed hash chain $Y = (K, Y_0, \ldots, Y_\ell)$ for HMAC has $Y_i = \mathrm{HMAC}(K, Y_{i-1})$ for $1 \le i \le \ell$. Given K, Y_0, Y_ℓ for a chain $Y = (K, Y_0, \ldots, Y_\ell)$, it is easy for an adversary to compute the start and end of a new chain $Y' = (K', Y'_0, \ldots, Y'_\ell)$ that does not overlap with Y. See Figure 3. In the full version, we detail how this structure can be abused in the context of an HMAC-based mutual proofs of work protocol. We also give an analogue of Theorem 1, i.e., a lower bound on the indifferentiability of HMAC from a RO when ambiguous key pairs can be queried.

4.2 Indifferentiability of HMAC with Restricted Keys

We have seen that HMAC's construction gives rise to two kinds of weak key pairs that can be abused to show that HMAC is not indifferentiable from a keyed RO (with good bounds). But weak key pairs are serendipitously avoided in most applications. For example, the recommended usage of HKDF [28] specifies keys of a fixed length less than $d-1$. Neither kind of weak key pairs exist within this subset of the key space.

While one can show indifferentiability for a variety of settings in which weak key pairs are avoided, we focus for simplicity on the case mentioned above. That is, we restrict to keys K for which $|K| = k$ and k is a fixed integer different less than $d-1$. The full version provides a more general set of results, covering also, for example, use of HMAC with a fixed key of any length less than or equal to d.

As our first positive result, we have the following theorem, which establishes the security of HMAC when modeling the underlying hash function as a RO.

Theorem 3. *Fix $d, k, n > 0$ with $k < d-1$. Let $P\colon \{0,1\}^* \to \{0,1\}^n$ be a RO, and consider $\mathrm{HMAC}_d[P]$ restricted to k-bit keys. Let $\mathcal{R}\colon \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^n$ be a keyed RO. Then there exists a simulator $\mathcal{S}$ such that for any distinguisher $\mathcal{A}$ whose total query cost is σ it holds that*

$$\mathbf{Adv}^{\mathrm{indiff}}_{\mathrm{HMAC}_d[P],\mathcal{R},\mathcal{S}}(\mathcal{A}) \le \mathcal{O}\left(\frac{\sigma^2}{2^n}\right)$$

$\mathcal{S}$ makes at most q_2 queries and runs in time $\mathcal{O}(q_2 \log q_2)$ where q_2 is the number of Prim queries made by $\mathcal{A}$. $\square$

The use of $\mathcal{O}(\cdot)$ just hides small constants. The proof is given in the full version. Combining Theorem 3 with the indifferentiability composition theorem allows us to conclude security for $\mathrm{HMAC}_d[H]$ for underyling hash function H that is, itself, indifferentiable from a RO. For example, should H be one of the proven-indifferentiable SHA-3 candidates. This does not, however, give us a security guarantee should H not be indifferentiable from a RO, as is the case with MD based hash functions. We therefore also prove, in the full version, the following theorem that establishes indifferentiability of HMAC using an underlying hash function built via the strengthened Merkle-Damgård (SMD) domain extension transform.

Theorem 4. *Fix $d, k, n > 0$ with $k < d - 1$ and $d \geq n$. Let $f\colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^n$ be a RO and consider $\mathrm{HMAC}_d[\mathrm{SMD}[f]]$ restricted to k-bit keys. Let $\mathcal{R}\colon \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^n$ be a keyed RO. Then there exists a simulator $\mathcal{S}$ such that for any distinguisher $\mathcal{A}$ whose total query cost is $\sigma \leq 2^{n-2}$ it holds that*

$$\mathbf{Adv}^{\mathrm{indiff}}_{\mathrm{HMAC}_d[\mathrm{SMD}[f]],\mathcal{R},\mathcal{S}}(\mathcal{A}) \leq \mathcal{O}\left(\frac{\sigma^2}{2^n}\right)$$

$\mathcal{S}$ makes at most q_2 queries and runs in time $\mathcal{O}(q_2 \log q_2)$ where q_2 is the number of Prim queries by $\mathcal{A}$. $\square$

We note that the restriction to $\sigma \leq 2^{n-2}$ in the theorem statement is just a technicality to make the bound simpler and likewise the use of $\mathcal{O}(\cdot)$ in the advantage statement hides just a small constant.

Unlike our positive results about H^2, the bounds provided by Theorems 3 and 4 match, up to small constants, results for other now-standard indifferentiable constructions (c.f., [13]). First, the advantage bounds both hold up to the birthday bound, namely $\sigma \approx 2^{n/2}$. Second, the simulators are efficient and, specifically, make at most one query per invocation. All this enables use of the indifferentiability composition theorem in a way that yields strong, standard concrete security bounds.

Acknowledgments

The authors thank Hugo Krawczyk for providing significant feedback and suggestions, in particular encouraging the authors to include positive results for the indifferentiability of HMAC; Niels Ferguson for in-depth discussions regarding the security of H^2; and the anonymous reviewers for their helpful suggestions. Dodis was supported in part by NSF grants CNS-1065134, CNS-1065288, CNS-1017471, CNS-0831299. Ristenpart was supported in part by NSF grant CNS-1065134. Steinberger is supported by the National Basic Research Program of China Grant 2011CBA00300, 2011CBA00301, the National Natural Science Foundation of China Grant 61033001, 61061130540, 61073174, and by NSF grant 0994380. Tessaro was supported in part by NSF grants CCF-0915675, CCF-1018064, and DARPA contracts FA8750-11-C-0096, FA8750-11-2-0225.

References

1. Elena Andreeva, Bart Mennink, and Bart Preneel. On the indifferentiability of the Grøstl hash function. In Juan A. Garay and Roberto De Prisco, editors, *SCN 10: 7th International Conference on Security in Communication Networks*, volume 6280 of *Lecture Notes in Computer Science*, pages 88–105. Springer, September 2010.
2. Elena Andreeva, Gregory Neven, Bart Preneel, and Thomas Shrimpton. Seven-property-preserving iterated hashing: ROX. In Kaoru Kurosawa, editor, *Advances in Cryptology – ASIACRYPT 2007*, volume 4833 of *Lecture Notes in Computer Science*, pages 130–146. Springer, December 2007.
3. Mihir Bellare. New proofs for NMAC and HMAC: Security without collision-resistance. In Cynthia Dwork, editor, *Advances in Cryptology – CRYPTO 2006*, volume 4117 of *Lecture Notes in Computer Science*, pages 602–619. Springer, August 2006.
4. Mihir Bellare, Zvika Brakerski, Moni Naor, Thomas Ristenpart, Gil Segev, Hovav Shacham, and Scott Yilek. Hedged public-key encryption: How to protect against bad randomness. In Mitsuru Matsui, editor, *Advances in Cryptology – ASIACRYPT 2009*, volume 5912 of *Lecture Notes in Computer Science*, pages 232–249. Springer, December 2009.
5. Mihir Bellare, Ran Canetti, and Hugo Krawczyk. Keying hash functions for message authentication. In Neal Koblitz, editor, *Advances in Cryptology – CRYPTO'96*, volume 1109 of *Lecture Notes in Computer Science*, pages 1–15. Springer, August 1996.
6. Mihir Bellare and Tadayoshi Kohno. A theoretical treatment of related-key attacks: RKA-PRPs, RKA-PRFs, and applications. In Eli Biham, editor, *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 491–506. Springer, May 2003.
7. Mihir Bellare and Thomas Ristenpart. Multi-property-preserving hash domain extension and the EMD transform. In Xuejia Lai and Kefei Chen, editors, *Advances in Cryptology – ASIACRYPT 2006*, volume 4284 of *Lecture Notes in Computer Science*, pages 299–314. Springer, December 2006.
8. Mihir Bellare and Thomas Ristenpart. Hash functions in the dedicated-key setting: Design choices and MPP transforms. In Lars Arge, Christian Cachin, Tomasz Jurdzinski, and Andrzej Tarlecki, editors, *ICALP 2007: 34th International Colloquium on Automata, Languages and Programming*, volume 4596 of *Lecture Notes in Computer Science*, pages 399–410. Springer, July 2007.
9. Mihir Bellare, Thomas Ristenpart, and Stefano Tessaro. Multi-instance security and its application to password-based cryptography. In *Advances in Cryptology – CRYPTO '12*, Lecture Notes in Computer Science. Springer, 2012.
10. Mihir Bellare and Phillip Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In Serge Vaudenay, editor, *Advances in Cryptology – EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 409–426. Springer, May / June 2006.
11. Guido Bertoni, Joan Daemen, Michael Peeters, and Gilles Van Assche. On the indifferentiability of the sponge construction. In Nigel P. Smart, editor, *Advances in Cryptology – EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 181–197. Springer, April 2008.
12. Donghoon Chang and Mridul Nandi. Improved indifferentiability security analysis of chopMD hash function. In Kaisa Nyberg, editor, *Fast Software Encryption –*

FSE 2008, volume 5086 of *Lecture Notes in Computer Science*, pages 429–443. Springer, February 2008.

13. Jean-Sébastien Coron, Yevgeniy Dodis, Cécile Malinaud, and Prashant Puniya. Merkle-Damgård revisited: How to construct a hash function. In Victor Shoup, editor, *Advances in Cryptology – CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 430–448. Springer, August 2005.
14. Yevgeniy Dodis, Rosario Gennaro, Johan Håstad, Hugo Krawczyk, and Tal Rabin. Randomness extraction and key derivation using the CBC, cascade and HMAC modes. In Matthew Franklin, editor, *Advances in Cryptology – CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 494–510. Springer, August 2004.
15. Yevgeniy Dodis, Leonid Reyzin, Ronald L. Rivest, and Emily Shen. Indifferentiability of permutation-based compression functions and tree-based modes of operation, with applications to MD6. In Orr Dunkelman, editor, *Fast Software Encryption – FSE 2009*, volume 5665 of *Lecture Notes in Computer Science*, pages 104–121. Springer, February 2009.
16. Yevgeniy Dodis, Thomas Ristenpart, and Thomas Shrimpton. Salvaging Merkle-Damgård for practical applications. In Antoine Joux, editor, *Advances in Cryptology – EUROCRYPT 2009*, volume 5479 of *Lecture Notes in Computer Science*, pages 371–388. Springer, April 2009.
17. Yevgeniy Dodis, Thomas Ristenpart, John Steinberger, and Stefano Tessaro. To Hash or Not to Hash, Again? On the Indifferentiability of the Second Iterate and HMAC, 2012. Full version of this paper. Available from authors' websites.
18. Cynthia Dwork and Moni Naor. Pricing via processing or combatting junk mail. In Ernest F. Brickell, editor, *Advances in Cryptology – CRYPTO'92*, volume 740 of *Lecture Notes in Computer Science*, pages 139–147. Springer, August 1993.
19. Cynthia Dwork, Moni Naor, and Hoeteck Wee. Pebbling and proofs of work. In Victor Shoup, editor, *Advances in Cryptology – CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 37–54. Springer, August 2005.
20. Niels Ferguson and Bruce Schneier. *Practical cryptography.* Wiley, 2003.
21. Pierre-Alain Fouque, David Pointcheval, and Sébastien Zimmer. HMAC is a randomness extractor and applications to TLS. In Masayuki Abe and Virgil Gligor, editors, *ASIACCS 08: 3rd Conference on Computer and Communications Security*, pages 21–32. ACM Press, March 2008.
22. J. Franks, P. Hallam-Baker, J. Hostetler, P. Leach, A. Luotonen, E. Sink, and L. Stewart. An Extension to HTTP: Digest Access Authentication. RFC 2069 (Proposed Standard), January 1997. Obsoleted by RFC 2617.
23. Shoichi Hirose, Je Hong Park, and Aaram Yun. A simple variant of the Merkle-Damgård scheme with a permutation. In Kaoru Kurosawa, editor, *Advances in Cryptology – ASIACRYPT 2007*, volume 4833 of *Lecture Notes in Computer Science*, pages 113–129. Springer, December 2007.
24. Ari Juels and John G. Brainard. Client puzzles: A cryptographic countermeasure against connection depletion attacks. In *ISOC Network and Distributed System Security Symposium – NDSS'99*. The Internet Society, February 1999.
25. Ghassan Karame and Srdjan Capkun. Low-cost client puzzles based on modular exponentiation. In Dimitris Gritzalis, Bart Preneel, and Marianthi Theoharidou, editors, *ESORICS 2010: 15th European Symposium on Research in Computer Security*, volume 6345 of *Lecture Notes in Computer Science*, pages 679–697. Springer, 2010.
26. H. Krawczyk, M. Bellare, and R. Canetti. HMAC: Keyed-Hashing for Message Authentication. RFC 2104, February 1997.

27. H. Krawczyk and P. Eronen. Hmac-based extract-and-expand key derivation function (hkdf). RFC 5869 (Proposed Standard), January 2010.
28. Hugo Krawczyk. Cryptographic extraction and key derivation: The HKDF scheme. In Tal Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 631–648. Springer, August 2010.
29. Ueli M. Maurer, Renato Renner, and Clemens Holenstein. Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology. In Moni Naor, editor, *TCC 2004: 1st Theory of Cryptography Conference*, volume 2951 of *Lecture Notes in Computer Science*, pages 21–39. Springer, February 2004.
30. PKCS #5: Password-based cryptography standard (rfc 2898). RSA Data Security, Inc., September 2000. Version 2.0.
31. Thomas Ristenpart, Hovav Shacham, and Thomas Shrimpton. Careful with composition: Limitations of the indifferentiability framework. In Kenneth G. Paterson, editor, *Advances in Cryptology – EUROCRYPT 2011*, volume 6632 of *Lecture Notes in Computer Science*, pages 487–506. Springer, May 2011.
32. Thomas Ristenpart and Scott Yilek. When good randomness goes bad: Virtual machine reset vulnerabilities and hedging deployed cryptography. In *Network and Distributed Systems Security – NDSS '10*. ISOC, 2010.
33. Douglas Stebila, Lakshmi Kuppusamy, Jothi Rangasamy, Colin Boyd, and Juan Manuel González Nieto. Stronger difficulty notions for client puzzles and denial-of-service-resistant protocols. In Aggelos Kiayias, editor, *Topics in Cryptology – CT-RSA 2011*, volume 6558 of *Lecture Notes in Computer Science*, pages 284–301. Springer, February 2011.
34. Gene Tsudik. Message authentication with one-way hash functions. In *Proceedings IEEE INFOCOM'92*, volume 3, pages 2055–2059. IEEE, 1992.
35. XiaoFeng Wang and Michael K. Reiter. Defending against denial-of-service attacks with puzzle auction. In *IEEE Symposium on Security and Privacy*, pages 78–92, 2003.