New Proofs for NMAC and HMAC: Security without Collision-Resistance

Mihir Bellare

Dept. of Computer Science & Engineering 0404, University of California San Diego
9500 Gilman Drive, La Jolla, CA 92093-0404, USA.
Email: mihir@cs.ucsd.edu ; URL: http://www-cse.ucsd.edu/users/mihir

Abstract. HMAC was proved in [3] to be a PRF assuming that (1) the underlying compression function is a PRF, and (2) the iterated hash function is weakly collision-resistant. However, recent attacks show that assumption (2) is false for MD5 and SHA-1, removing the proof-based support for HMAC in these cases. This paper proves that HMAC is a PRF under the sole assumption that the compression function is a PRF. This recovers a proof based guarantee since no known attacks compromise the pseudorandomness of the compression function, and it also helps explain the resistance-to-attack that HMAC has shown even when implemented with hash functions whose (weak) collision resistance is compromised. We also show that an even weaker-than-PRF condition on the compression function, namely that it is a privacy-preserving MAC, suffices to establish HMAC is a secure MAC as long as the hash function meets the very weak requirement of being computationally almost universal, where again the value lies in the fact that known attacks do not invalidate the assumptions made.

1 Introduction

HMAC [3] is a popular cryptographic-hash-function-based MAC. The basic construct is actually NMAC, of which HMAC can be viewed as a derivative.

The constructions. Succinctly:

$$\begin{aligned} \mathsf{NMAC}(K_{\rm out}\|K_{\rm in}, M) &= H^*(K_{\rm out}, H^*(K_{\rm in}, M)) \\ \mathsf{HMAC}(K_{\rm out}\|K_{\rm in}, M) &= H(K_{\rm out}\|H(K_{\rm in}\|M)) \ . \end{aligned}$$

Here H is a cryptographic hash function, eg. MD5 [28], SHA-1 [26], or RIPEMD-160 [17]. Let h: $\{0,1\}^c \times \{0,1\}^b \to \{0,1\}^c$ denote the underlying compression function. (Here $b = 512$ while c is 128 or 160.) Let h^* be the iterated compression function which on input $K \in \{0,1\}^c$ and a message $x = x[1]\ldots x[n]$ consisting of b-bit blocks, lets $a[0] = K$ and $a[i] = h(a[i-1], x[i])$ for $i = 1,\ldots,n$, and finally returns $a[n]$. Then $H^*(K, M) = h^*(K, M^*)$ and $H(M) = H^*(\mathrm{IV}, M)$, where M^* denotes M padded appropriately to a length that is a positive multiple of b and IV is a public c-bit initial vector that is fixed as part of the description of H. Both NMAC and HMAC use two keys, which in the case of NMAC are of length c bits each, and in the case of HMAC of length b bits each and derived from a single b-bit key. HMAC is a non-intrusive version of NMAC in the sense that it uses the cryptographic hash function only as a black box, making it easier to implement.

USAGE. HMAC is standardized via an IETF RFC [22], a NIST FIPS [25] and ANSI X9.71 [1], and implemented in SSL, SSH, IPsec and TLS amongst other places. It is often used as a PRF (pseudorandom function [19]) rather than merely as a MAC. In particular this is the case when it is used for key-derivation, as in TLS [16] and IKE (the Internet Key Exchange protocol of IPsec) [20]. HMAC is also used as a PRF in a proposed standard for one-time passwords [24].

WHAT'S KNOWN. The results are for NMAC but can be lifted to HMAC. It is shown in [3] that NMAC is a secure PRF if (1) the underlying compression function h is a secure PRF, and also (2) that the hash function H is *weakly collision resistant* (WCR). The latter, introduced in [3], is a relaxation of collision resistance (CR) that asks that it be computationally infeasible for an adversary, given an oracle for $H^*(K, \cdot)$ under a hidden key K, to find a collision, meaning distinct inputs M_1, M_2 such that $H^*(K, M_1) = H^*(K, M_2)$.

THE PROBLEM. HMAC is usually implemented with MD5 or SHA-1. But, due to recent attacks [32, 31], these functions are *not* WCR. Thus the assumption on which the proof of [3] is based is not true. This does not reflect any actual weaknesses in the NMAC or HMAC constructs, on which no attacks are known. (Being iterated MACs, the generic birthday based forgery attacks of [27] always break NMAC and HMAC in time $2^{c/2}$, but we mean no better-than-birthday attacks are known.) But it means that we have lost the proof-based guarantees. We are interested in recovering them.

LOSS OF WCR. First we pause to expand on the claim above that our main hash functions are not WCR. Although WCR appears to be a weaker requirement than CR due to the hidden key, in fact, for iterated hash functions, it ends up not usually being so. The reason is that collision-finding attacks such as those on MD5 [32] and SHA-1 [31] extend to find collisions in $H^*(\mathrm{IV}, \cdot)$ for an arbitrary but given IV, and, any such attack, via a further extension attack, can be used to compromise WCR, meaning to find a collision in $H^*(K, \cdot)$, given an oracle for this function, even with K hidden. This was pointed out in [3, 21], and, for the curious, we recall the attack in [2].

MAIN RESULT. We show (Theorem 1) that NMAC is a PRF under the *sole* assumption that the underlying compression function h is itself a PRF. In other words, the additional assumption that the hash function is WCR is dropped. (And, in particular, as long as h is a PRF, the conclusion is true even if H is *not* WCR, let alone CR.)

The main advantage of our result is that it is based on an assumption that is not refuted by any known attacks. (There are to date no attacks that compromise the pseudorandomness of the compression functions of MD5 or SHA-1.) Another feature of our result is that it is the first proof for NMAC that is based solely on an assumption about the compression function rather than also assuming something about the entire iterated hash function.

Techniques. We show (Lemma 1) that if a compression function h is a PRF then the iterated compression function h^* is *computationally almost universal* (cAU), a computational relaxation of the standard information-theoretic notion of almost-universality (AU) of [13, 33, 30]. We then conclude with Lemma 2 which says that the composition of a PRF and a cAU function is a PRF. (This can be viewed as a computational relaxation of the Carter-Wegman paradigm [13, 33].)

Related work. If h^* were a PRF, it would imply it is cAU, but h^* is not a PRF due to the extension attack. It is however shown by [4] that if h is a PRF then h^* (which they call the cascade) is a "pf-PRF" (prefix-free PRF), meaning a PRF as long as no query of the adversary is a prefix of another query. It was pointed out to us by Shoup after seeing an early draft of our paper that it is possible to apply this in a black-box way to show that h^* is cAU. However Lemma 1 is a somewhat stronger result and bound whose proof distills and strengthens the ideas of [4] and also involves some new ones. For comparison, we do present the indirect proof in [2].

Dodis, Gennaro, Håstad, Krawczyk and Rabin show [18, Lemma 4] that the cascade over a family of *random* functions is AU as long as the two messages whose collision probability one considers have the same length. (In this model, $h(K,\cdot)$ is a random function for each $K \in \{0,1\}^c$. That is, it is like Shannon's ideal cipher model, except the component maps are functions not permutations.) This does not imply Lemma 1 (showing the cascade h^* is cAU if h is a PRF), because we need to allow the two messages to have different lengths, and also because it is not clear what implication their result has for the case when h is a PRF. (A PRF does *not* permit one to securely instantiate a *family* of random functions.) A second result [18, Lemma 5] in the same paper says that if $h^*(K,M)$ is close to uniformly distributed then so is $h^*(K,M\|X)$. (Here M is chosen from some distribution, K is a random but known key, and X is a fixed block.) This result only assumes h is a PRF, but again we are not able to discern any implications for the problems we consider, because in our case the last block of the input is not fixed, we are interested in the cAU property rather than randomness, and our inputs are not drawn from a distribution.

Another result. The fact that compression functions are underlain by block ciphers, together with the fact that no known attacks compromise the pseudorandomness of the compression functions of MD5, SHA-1, may give us some confidence that it is ok to assume these are PRFs, but it still behooves us to be cautious. What can we prove about NMAC without assuming the compression function is a PRF? We would not expect to be able to prove it is a PRF, but what about just a secure MAC? (Any PRF is a secure MAC [6, 9], so our main result implies NMAC is a secure MAC, but we are interested in seeing whether this can be proved under weaker assumptions.) We show (Theorem 2) that NMAC is a secure MAC if h is a *privacy-preserving MAC* (PP-MAC) [8] and h^* (equivalently,

H^*) is cAU. A PP-MAC (the definition is provided in Section 4) is stronger than a MAC but weaker than a PRF. This result reverts to the paradigm of [3] of making assumptions both about the compression function and its iteration, but the point is that cAU is a very weak assumption compared to WCR and PP-MAC is a weaker assumption than PRF.

From NMAC to HMAC. The formal results (both previous and new) we have discussed so far pertain to NMAC. However, discussions (above and in the literature) tend to identify NMAC and HMAC security-wise. This is explained by an observation of [3] which says that HMAC inherits the security of NMAC as long as the compression function is a PRF when keyed via the data input. (So far when we have talked of it being a PRF, it is keyed via the chaining variable.) In our case this means that HMAC is a PRF if the compression function is a "dual-PRF," meaning a PRF when keyed by either of its two inputs.

However, the analysis above assumes that the two keys $K_{\mathrm{out}}, K_{\mathrm{in}}$ of HMAC are chosen *independently* at random, while in truth they are equal to $K \oplus \mathsf{opad}$ and $K \oplus \mathsf{ipad}$ respectively, where K is a random b-bit key and $\mathsf{opad}, \mathsf{ipad}$ are fixed, distinct constants. We apply the theory of PRFs under related-key attacks [7] to extend the observation of [3] to this single-key version of HMAC, showing it inherits the security of NMAC as long as the data-input-keyed compression function is a PRF under an appropriate (and small) class of related key attacks. Assuming additionally that the compression function is a PRF in the usual sense, we obtain a (in fact, the first) security proof of the single-key version of HMAC. These results are in Section 5.

2 Definitions

Notation. We denote by $s_1 \| s_2$ the concatenation of strings s_1, s_2, and by $|s|$ the length of string s. Let b be a positive integer representing a block length, and let $B = \{0,1\}^b$. Let B^+ denote the set of all strings of length a positive multiple of b bits. Whenever we speak of blocks we mean b-bit ones. If $M \in B^+$ then $\|M\|_b = |M|/b$ is the number of blocks in M, and $M[i]$ denotes its i-th b-bit block, meaning $M = M[1] \ldots M[n]$ where $n = \|M\|_b$. If $M_1, M_2 \in B^+$, then M_1 is a prefix of M_2, written $M_1 \subseteq M_2$, if $M_2 = M_1 \| A$ for some $A \in B^*$. If S is a set then $s \stackrel{\$}{\leftarrow} S$ denotes the operation of selecting s uniformly at random from S. An adversary is a (possibly randomized) algorithm that may have access to one or more oracles. We let

$$A^{\mathcal{O}_1,\cdots}(x_1,\ldots) \Rightarrow 1 \qquad \text{and} \qquad y \stackrel{\$}{\leftarrow} A^{\mathcal{O}_1,\cdots}(x_1,\ldots)$$

denote, respectively, the event that A with the indicated oracles and inputs outputs 1, and the experiment of running A with the indicated oracles and inputs and letting y be the value returned. (This value is a random variable depending on the random choices made by A and its oracles.) Either the oracles or the inputs (or both) may be absent, and often will be.

A family of functions is a two-argument map $f\colon\ \mathit{Keys} \times \mathit{Dom} \to \mathit{Rng}$ whose first argument is regarded as a key. We fix one such family $h\colon\ \{0,1\}^c \times B \to$

$\{0,1\}^c$ to model a compression function that we regard as being keyed via its c-bit chaining variable. Typical values are $b = 512$ and $c = 128$ or 160. The *iteration* of family $h\colon \{0,1\}^c \times B \to \{0,1\}^c$ is the family of functions $h^*\colon \{0,1\}^c \times B^+ \to \{0,1\}^c$ where $h^*(K, M)$ (for $K \in \{0,1\}^c$ and $M \in B^+$) is defined by the following code: $n \leftarrow \|M\|_b$; $a[0] \leftarrow K$; For $i = 1, \ldots, n$ do $a[i] \leftarrow h(a[i-1], M[i])$; Return $a[n]$. This represents the Merkle-Damgård [23, 14] iteration method used in all the popular hash functions but without the "strengthening," meaning that there is no $|M|$-based message padding.

PRFs. A prf-adversary A against a family of functions $f\colon \mathit{Keys} \times \mathit{Dom} \to \mathit{Rng}$ takes as oracle a function $g\colon \mathit{Dom} \to \mathit{Rng}$ and returns a bit. The prf-advantage of A against f is the difference between the probability that it outputs 1 when its oracle is $g = f(K, \cdot)$ for a random key $K \stackrel{\$}{\leftarrow} \mathit{Keys}$, and the probability that it outputs 1 when its oracle g is chosen at random from the set $\mathsf{Maps}(\mathit{Dom}, \mathit{Rng})$ of all functions mapping Dom to Rng, succinctly written as

$$\mathbf{Adv}_f^{\mathrm{prf}}(A) \;=\; \Pr\left[A^{f(K,\cdot)} \Rightarrow 1\right] - \Pr\left[A^{\$} \Rightarrow 1\right] . \qquad (1)$$

In both cases the probability is over the choice of oracle and the coins of A.

cAU and collision-probability. Let $F\colon \{0,1\}^k \times \mathit{Dom} \to \mathit{Rng}$ be a family of functions. cAU is measured by considering an almost-universal (au) adversary A against F. It (takes no inputs and) returns a pair of messages in Dom. Its au-advantage, denoted $\mathbf{Adv}_F^{\mathrm{au}}(A)$, is

$$\Pr\left[F(K, M_1) = F(K, M_2) \;\wedge\; M_1 \neq M_2 \;:\; (M_1, M_2) \stackrel{\$}{\leftarrow} A \;;\; K \stackrel{\$}{\leftarrow} \mathit{Keys}\right] .$$

This represents a very weak form of collision-resistance since A must produce M_1, M_2 without being given any information about K. WCR [3] is a stronger notion because here A gets an oracle for $F(K, \cdot)$ and can query this in its search for M_1, M_2.

For $M_1, M_2 \in \mathit{Dom}$ it is useful to let $\mathsf{Coll}_F(M_1, M_2) = \Pr[F(K, M_1) = F(K, M_2)]$, the probability being over $K \stackrel{\$}{\leftarrow} \{0,1\}^k$.

3 Security of NMAC

Let $h\colon \{0,1\}^c \times \{0,1\}^b \to \{0,1\}^c$ be a family of functions that represents the compression function, here assumed to be a PRF. Let pad denote a padding function such that $s^* = s \| \mathsf{pad}(|s|) \in B^+$ for any string s. (Such padding functions are part of the description of current hash functions. Note the pad depends only on the length of s.) Then the family $\mathsf{NMAC}\colon \{0,1\}^{2c} \times D \to \{0,1\}^c$ is defined by $\mathsf{NMAC}(K_{\mathrm{out}} \| K_{\mathrm{in}}, M) = h(K_{\mathrm{out}}, h^*(K_{\mathrm{in}}, M^*) \| \mathsf{fpad})$ where $\mathsf{fpad} = \mathsf{pad}(c) \in \{0,1\}^{b-c}$ and h^* is the iterated compression function as defined in Section 2. The domain D is the set of all strings up to some maximum length, which is 2^{64} for current hash functions.

It turns out that our security proof for NMAC does not rely on any properties of pad beyond the fact that $M^* = M \| \mathsf{pad}(|M|) \in B^+$. (In particular, the Merkle-Damgård strengthening, namely inclusion of the message length in the padding, that is used in current hash functions and is crucial to collision resistance of the hash function, is not important to the security of NMAC.) Accordingly, we will actually prove the security of a more general construct that we call generalized NMAC. The family GNMAC: $\{0,1\}^{2c} \times B^+ \to \{0,1\}^c$ is defined by $\mathsf{GNMAC}(K_{\rm out} \| K_{\rm in}, M) = h(K_{\rm out}, h^*(K_{\rm in}, M) \| \mathsf{fpad})$ where fpad is any (fixed) $b-c$ bit string. Note the domain is B^+, meaning inputs have to have a length that is a positive multiple of b bits. (But can be of any length.) NMAC is nonetheless a special case of GNMAC via $\mathsf{NMAC}(K_{\rm out} \| K_{\rm in}, M) = \mathsf{GNMAC}(K_{\rm out} \| K_{\rm in}, M^*)$ and thus the security of NMAC is implied by that of GNMAC. (Security as a PRF or a MAC, respectively, for both.)

3.1 The results

Main Lemma. The following says that if h is a PRF then its iteration h^* is cAU.

Lemma 1. *Let $B = \{0,1\}^b$. Let h: $\{0,1\}^c \times B \to \{0,1\}^c$ be a family of functions, and let A^* be an au-adversary against h^*. Assume that the two messages output by A^* are at most $n_1, n_2 \geq 1$ blocks long, respectively. Then there exists a prf-adversary A against h such that*

$$\mathbf{Adv}^{\rm au}_{h^*}(A^*) \;\leq\; (n_1 + n_2 - 1) \cdot \mathbf{Adv}^{\rm prf}_{h}(A) + \frac{1}{2^c} \,. \tag{2}$$

Furthermore, A has time-complexity at most $O((n_1 + n_2)T_h)$, where T_h is the time for one evaluation of h, and makes at most 2 oracle queries. ▮

The proof is in Section 3.3. The quality of the reduction is good because the time-complexity of the constructed adversary A is small and in particular independent of the time-complexity of A^* (the proof shows how this is possible) and furthermore A makes only two oracle queries.

One might ask whether stronger results hold. For example, assuming h is a PRF, (1) Is h^* a PRF? (2) Is h^* WCR? (Either would imply that h^* is cAU.) But the answer is NO to both questions. The function h^* is *never* a PRF due to the extension attack. On the other hand it is easy to give an example of a PRF h such that h^* is not WCR. Also MD5 and SHA-1 are candidate counter-examples, since their compression functions appear to be PRFs but their iterations are not WCR.

Prf(cAU)=Prf lemma. The *composition* of families h: $\{0,1\}^c \times \{0,1\}^b \to \{0,1\}^c$ and F: $\{0,1\}^k \times D \to \{0,1\}^b$ is the family hF: $\{0,1\}^{c+k} \times D \to \{0,1\}^c$ defined by $hF(K_{\rm out} \| K_{\rm in}, M) = h(K_{\rm out}, F(K_{\rm in}, M))$. The following lemma says that if h is a PRF and F is cAU then hF is a PRF.

Lemma 2. *Let $B = \{0,1\}^b$. Let h: $\{0,1\}^c \times B \to \{0,1\}^c$ and F: $\{0,1\}^k \times D \to B$ be families of functions, and let hF: $\{0,1\}^{c+k} \times D \to \{0,1\}^c$ be defined by*

$$hF(K_{\rm out} \| K_{\rm in}, M) = h(K_{\rm out}, F(K_{\rm in}, M))$$

for all $K_{\rm out} \in \{0,1\}^c, K_{\rm in} \in \{0,1\}^k$ and $M \in D$. Let A_{hF} be a prf-adversary against hF that makes at most $q \geq 2$ oracle queries, each of length at most n, *and has time-complexity at most t. Then there exists a prf-adversary A_h against h and an au-adversary A_F against F such that*

$$\mathbf{Adv}^{\mathrm{prf}}_{hF}(A_{hF}) \;\leq\; \mathbf{Adv}^{\mathrm{prf}}_{h}(A_h) + \binom{q}{2} \cdot \mathbf{Adv}^{\mathrm{au}}_{F}(A_F)\,. \tag{3}$$

Furthermore, A_h has time-complexity at most t and makes at most q oracle queries, while A_F has time-complexity $O(T_F(n))$ and the two messages it outputs have length at most n, where $T_F(n)$ is the time to compute F on an n-bit input. ▮

This extends the analogous PRF(AU)=PRF lemma by relaxing the condition on F from AU to cAU. (The PRF(AU)=PRF lemma is alluded to in [4, 11], and variants are in [11, 12].) A simple proof of Lemma 2, using games [10, 29], is in [2].

GNMAC IS A PRF. We now combine the two lemmas above to conclude that if h is a PRF then so is GNMAC.

Theorem 1. *Assume $b \geq c$ and let $B = \{0,1\}^b$. Let h: $\{0,1\}^c \times B \to \{0,1\}^c$ be a family of functions and let $\mathsf{fpad} \in \{0,1\}^{b-c}$ be a fixed padding string. Let GNMAC: $\{0,1\}^{2c} \times B^+ \to \{0,1\}^c$ be defined by*

$$\mathsf{GNMAC}(K_{\mathrm{out}} \| K_{\mathrm{in}}, M) = h(K_{\mathrm{out}}, h^*(K_{\mathrm{in}}, M) \| \mathsf{fpad})$$

for all $K_{\mathrm{out}}, K_{\mathrm{in}} \in \{0,1\}^c$ and $M \in B^+$. Let A_{GNMAC} be a prf-adversary against GNMAC that makes at most q oracle queries, each of at most m blocks, and has time-complexity at most t. Then there exist prf-adversaries A_1, A_2 against h such that

$$\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{GNMAC}}(A_{\mathsf{GNMAC}}) \;\leq\; \mathbf{Adv}^{\mathrm{prf}}_{h}(A_1) + \binom{q}{2}\left[2m \cdot \mathbf{Adv}^{\mathrm{prf}}_{h}(A_2) + \frac{1}{2^c}\right]\,. \tag{4}$$

Furthermore, A_1 has time-complexity at most t and makes at most q oracle queries, while A_2 has time-complexity at most $O(mT_h)$ and makes at most 2 oracle queries, where T_h is the time for one computation of h. ▮

Proof (Theorem 1). Let F: $\{0,1\}^c \times B^+ \to \{0,1\}^b$ be defined by $F(K_{\mathrm{in}}, M) = h^*(K_{\mathrm{in}}, M) \| \mathsf{fpad}$. Then $\mathsf{GNMAC} = hF$. Apply Lemma 2 (with $k = c$, $D = B^+$ and $A_{hF} = A_{\mathsf{GNMAC}}$) to get prf-adversary A_1 and au-adversary A_F with the properties stated in the lemma. Note that $\mathbf{Adv}^{\mathrm{au}}_{F}(A_F) = \mathbf{Adv}^{\mathrm{au}}_{h^*}(A_F)$. (Because a pair of messages is a collision for $h^*(K_{\mathrm{in}}, \cdot) \| \mathsf{fpad}$ iff it is a collision for $h^*(K_{\mathrm{in}}, \cdot)$.) Now apply Lemma 1 to $A^* = A_F$ to get prf-adversary A_2. ▮

3.2 Tightness of bound

The best known attack on GNMAC is the birthday one of [27]. They show that it is possible to break NMAC with about $2^{c/2}/\sqrt{m}$ queries of at most m blocks each. We now want to assess how close to this is the guarantee we can get from Theorem 1 gets. If t is a time-complexity then let $\bar{t} = t/T_h$. Assume that the best attack against h as a PRF is exhaustive key search. (Birthday attacks do not apply since h is not a family of permutations.) This means that $\mathbf{Adv}^{\mathrm{prf}}_{h}(A) \leq \bar{t} \cdot 2^{-c}$ for any prf-adversary A of time complexity t making $q \leq \bar{t}$ queries.

Game $G_1(M_1, M_2, l)$

$m_1 \leftarrow \|M_1\|_b$; $m_2 \leftarrow \|M_2\|_b$
$a[l] \xleftarrow{\$} \{0,1\}^c$
For $i = l+1$ to m_2 do
  $a[i] \leftarrow h(a[i-1], M_2[i])$
If $a[m_1] = a[m_2]$ then return 1
  else return 0

Adversary $A_3^g(M_1, M_2)$

$m_1 \leftarrow \|M_1\|_b$; $m_2 \leftarrow \|M_2\|_b$
$l \xleftarrow{\$} \{1, \ldots, m_1+1\}$
If $l = m_1 + 1$ then return $A_2^g(M_1, M_2)$
Else return $A_1^g(M_1, M_2, l)$

Adversary $A_1^g(M_1, M_2, l)$

$m_1 \leftarrow \|M_1\|_b$; $m_2 \leftarrow \|M_2\|_b$
$a[l] \leftarrow g(M_2[l])$
For $i = l+1$ to m_2 do
  $a[i] \leftarrow h(a[i-1], M_2[i])$
If $a[m_1] = a[m_2]$ then return 1
  else return 0

Adversary $A_2^g(M_1, M_2)$

$m_1 \leftarrow \|M_1\|_b$; $m_2 \leftarrow \|M_2\|_b$
$a[m_1+1] \leftarrow g(M_2[m_1+1])$
For $i = m_1+2$ to m_2 do
  $a[i] \leftarrow h(a[i-1], M_2[i])$
$y \xleftarrow{\$} B \setminus \{M_2[m_1+1]\}$
If $h(a[m_2], y) = g(y)$ then return 1
  else return 0

Fig. 1. Games and adversaries taking input distinct messages M_1, M_2 such that $M_1 \subseteq M_2$. The adversaries take an oracle $g\colon \{0,1\}^b \to \{0,1\}^c$. For Game $G_1(M_1, M_2, l)$, the input l is the range $0 \le l \le \|M_1\|_b$ while for adversary A_1 it is in the range $1 \le l \le \|M_1\|_b$.

Plugging this into (4) and simplifying, the upper bound on the prf-advantage of any adversary against GNMAC who has time-complexity t and makes at most q queries is $O(\bar{t} + m^2q^2T_h) \cdot 2^{-c}$. If we ignore the T_h term, then this hits 1 when $q \approx 2^{c/2}/m$. This means that the bound justifies NMAC up to roughly $2^{c/2}/m$ queries, off from the number in the above-mentioned attack by a factor of $\sqrt{m}$. It is an interesting open problem to improve our analysis and fill the gap.

3.3 Proof of Lemma 1

Proofs of claims below are omitted and may be found in [2].

Some definitions. In this proof it will be convenient to consider prf-adversaries that take inputs. The advantage of A against h on inputs $x_1, \ldots$ is defined as

$$\mathbf{Adv}_h^{\mathrm{prf}}(A(x_1,\ldots)) \;=\; \Pr\left[A^{h(K,\cdot)}(x_1,\ldots) \Rightarrow 1\right] - \Pr\left[A^{\$}(x_1,\ldots) \Rightarrow 1\right],$$

where in the first case $K \xleftarrow{\$} \{0,1\}^c$ and in the second case the notation means that A is given as oracle a map chosen at random from $\mathsf{Maps}(\{0,1\}^b, \{0,1\}^c)$.

Overview. To start with, we ignore A_{hF} and upper bound $\mathsf{Coll}_F(M_1, M_2)$ as some appropriate function of the prf-advantage of a prf-adversary against h that takes M_1, M_2 as input. We consider first the case that $M_1 \subseteq M_2$ (M_1 is a prefix of M_2) and then the case that $M_1 \not\subseteq M_2$, building in each case a different adversary.

THE CASE $M_1 \subseteq M_2$. We begin with some high-level intuition. Suppose $M_1 \subseteq M_2$ with $m_2 = \|M_2\|_b \geq 1 + m_1$, where $m_1 = \|M_1\|_b$. The argument to upper bound $\mathsf{Coll}_{h^*}(M_1, M_2)$ has two parts. First, a hybrid argument is used to show that $a[m_1] = h^*(K, M_1)$ is computationally close to random when K is drawn at random. Next, we imagine a game in which $a[m_1]$ functions as a key to h. Let $a[m_1 + 1] = h(a[m_1], M_2[m_1 + 1])$ and $a[m_2] = h^*(a[m_1 + 1], M_2[m_1 + 2] \dots M_2[m_2])$. Now, if $a[m_2] = a[m_1]$ then we effectively have a way to recover the "key" $a[m_1]$ given $a[m_1+1]$, amounting to a key-recovery attack on $h(a[m_1], \cdot)$ based on one input-output example of this function. But being a PRF, h is also secure against key-recovery.

In the full proof that follows, we use the games and adversaries specified in Figure 1. Adversaries A_1, A_2 represent, respectively, the first and second parts of the argument outlined above, while A_3 integrates the two.

Claim 1. Let $M_1, M_2 \in B^+$ with $M_1 \subseteq M_2$ and $1 + \|M_1\|_b \leq \|M_2\|_b$. Suppose $1 \leq l \leq \|M_1\|_b$. Then

$$\Pr\left[A_1^{\$}(M_1, M_2, l) \Rightarrow 1\right] = \Pr\left[G_1(M_1, M_2, l) \Rightarrow 1\right]$$
$$\Pr\left[A_1^{h(K,\cdot)}(M_1, M_2, l) \Rightarrow 1\right] = \Pr\left[G_1(M_1, M_2, l-1) \Rightarrow 1\right]. \blacksquare$$

Recall the notation means that in the first case A_1 gets as oracle $g \xleftarrow{\$} \mathsf{Maps}(\{0,1\}^b, \{0,1\}^c)$ and in the second case $K \xleftarrow{\$} \{0,1\}^c$.

Claim 2. Let $M_1, M_2 \in B^+$ with $M_1 \subseteq M_2$ and $1 + \|M_1\|_b \leq \|M_2\|_b$. Then

$$\Pr\left[A_2^{\$}(M_1, M_2) \Rightarrow 1\right] = 2^{-c}$$
$$\Pr\left[A_2^{h(K,\cdot)}(M_1, M_2) \Rightarrow 1\right] \geq \Pr\left[G_1(M_1, M_2, m_1) \Rightarrow 1\right]. \blacksquare$$

Claim 3. Let $M_1, M_2 \in B^+$ with $M_1 \subseteq M_2$ and $1 + \|M_1\|_b \leq \|M_2\|_b$. Let $m_1 = \|M_1\|_b$. Then

$$\mathbf{Adv}_h^{\mathrm{prf}}(A_3(M_1, M_2)) \geq \frac{1}{m_1 + 1}\left(\mathsf{Coll}_{h^*}(M_1, M_2) - 2^{-c}\right). \blacksquare$$

THE CASE $M_1 \not\subseteq M_2$. For $M_1, M_2 \in B^+$ with $\|M_1\|_b \leq \|M_2\|_b$ and $M_1 \not\subseteq M_2$, we let $\mathrm{LCP}(M_1, M_2)$ denote the length of the longest common blockwise prefix of M_1, M_2, meaning the largest integer p such that $M_1[1] \dots M_1[p] = M_2[1] \dots M_2[p]$ but $M_1[p+1] \neq M_2[p+1]$. We consider the games and adversaries of Figure 2.

Claim 4. Let $M_1, M_2 \in B^+$ with $M_1 \not\subseteq M_2$, and $\|M_1\|_b \leq \|M_2\|_b$. Suppose $1 \leq l \leq \|M_1\|_b + \|M_2\|_b - \mathrm{LCP}(M_1, M_2) - 1$. Then

$$\Pr\left[A_4^{\$}(M_1, M_2, l) \Rightarrow 1\right] = \Pr\left[G_2(M_1, M_2, l) \Rightarrow 1\right]$$
$$\Pr\left[A_4^{h(K,\cdot)}(M_1, M_2, l) \Rightarrow 1\right] = \Pr\left[G_2(M_1, M_2, l-1) \Rightarrow 1\right]. \blacksquare$$

Game $G_2(M_1, M_2, l)$ // $0 \le l \le \|M_1\|_b + \|M_2\|_b - \mathrm{LCP}(M_1, M_2) - 1$

200 $m_1 \leftarrow \|M_1\|_b$; $m_2 \leftarrow \|M_2\|_b$; $p \leftarrow \mathrm{LCP}(M_1, M_2)$

210 If $0 \le l \le m_1$ then

220 $\quad a_1[l] \xleftarrow{\$} \{0,1\}^c$; For $i = l+1$ to m_1 do $a_1[i] \leftarrow h(a_1[i-1], M_1[i])$

230 If $m_1 + 1 \le l \le m_1 + m_2 - p$ then $a_1[m_1] \xleftarrow{\$} \{0,1\}^c$

240 If $0 \le l \le p$ then $n \leftarrow l$; $a_2[n] \leftarrow a_1[n]$

250 If $p + 1 \le l \le m_1$ then $n \leftarrow p + 1$; $a_2[n] \xleftarrow{\$} \{0,1\}^c$

260 If $m_1 + 1 \le l \le m_1 + m_2 - p$ then $n \leftarrow l - m_1 + p + 1$; $a_2[n] \xleftarrow{\$} \{0,1\}^c$

270 For $i = n+1$ to m_2 do $a_2[i] \leftarrow h(a_2[i-1], M_2[i])$

280 If $a_1[m_1] = a_2[m_2]$ then return 1 else return 0

Adversary $A_4^g(M_1, M_2, l)$ // $1 \le l \le \|M_1\|_b + \|M_2\|_b - \mathrm{LCP}(M_1, M_2) - 1$

a00 $m_1 \leftarrow \|M_1\|_b$; $m_2 \leftarrow \|M_2\|_b$; $p \leftarrow \mathrm{LCP}(M_1, M_2)$

a10 If $1 \le l \le m_1$ then

a20 $\quad a_1[l] \leftarrow g(M_1[l])$; For $i = l+1$ to m_1 do $a_1[i] \leftarrow h(a_1[i-1], M_1[i])$

a30 If $m_1 + 1 \le l \le m_1 + m_2 - p$ then $a_1[m_1] \xleftarrow{\$} \{0,1\}^c$

a40 If $1 \le l \le p$ then $n \leftarrow l$; $a_2[n] \leftarrow a_1[n]$

a50 If $l = p + 1$ then $n \leftarrow p + 1$; $a_2[n] \leftarrow g(M_2[n])$

a51 If $p + 2 \le l \le m_1$ then $n \leftarrow p + 1$; $a_2[n] \xleftarrow{\$} \{0,1\}^c$

a60 If $m_1 + 1 \le l \le m_1 + m_2 - p$ then $n \leftarrow l - m_1 + p + 1$; $a_2[n] \leftarrow g(M_2[n])$

a70 For $i = n+1$ to m_2 do $a_2[i] \leftarrow h(a_2[i-1], M_2[i])$

a80 If $a_1[m_1] = a_2[m_2]$ then return 1 else return 0

Fig. 2. Games and adversaries taking input distinct messages $M_1, M_2 \in B^+$ such that $M_1 \not\subseteq M_2$ and $\|M_1\|_b \le \|M_2\|_b$. For Game $G_2(M_1, M_2, l)$, the input l is the range $0 \le l \le \|M_1\|_b + \|M_2\|_b - \mathrm{LCP}(M_1, M_2) - 1$ while for adversary A_4 it is in the range $1 \le l \le \|M_1\|_b + \|M_2\|_b - \mathrm{LCP}(M_1, M_2) - 1$.

We now define prf adversary $A_5^g(M_1, M_2)$ against h as follows. It picks $l \xleftarrow{\$} \{1, \ldots, \|M_1\|_b + \|M_2\|_b - \mathrm{LCP}(M_1, M_2) - 1\}$ and returns $A_4^g(M_1, M_2, l)$.

Claim 5. Let $M_1, M_2 \in B^+$ with $M_1 \not\subseteq M_2$ and $\|M_1\|_b \le \|M_2\|_b$. Let $m = \|M_1\|_b + \|M_2\|_b - \mathrm{LCP}(M_1, M_2) - 1$. Then

$$\mathbf{Adv}_h^{\mathrm{prf}}(A_5) \ge \frac{1}{m} \cdot \left(\mathsf{Coll}_{h^*}(M_1, M_2) - 2^{-c}\right) \ . \quad \blacksquare$$

Putting it together. The final step to construct the prf-adversary A against h, claimed in the lemma, is to appropriately combine A_3, A_5. We assume wlog that the two messages M_1, M_2 output by A^* are always distinct, in B^+, and satisfy $\|M_1\|_b \le \|M_2\|_b$. We first define

Adversary $A_6^g(M_1, M_2)$	**Adversary** A_7^g
If $M_1 \subseteq M_2$ then return $A_3^g(M_1, M_2)$	$(M_1, M_2) \xleftarrow{\$} A^*$
Else return $A_5^g(M_1, M_2)$	Return $A_6^g(M_1, M_2)$

Claim 6. $\mathbf{Adv}^{\mathrm{au}}_{h^*}(A^*) \;\le\; 2^{-c} + (n_1 + n_2 - 1) \cdot \mathbf{Adv}^{\mathrm{prf}}_{h}(A_7)$ ∎

Prf-Adversary A_7 achieves the prf-advantage we seek, but has time-complexity that of A^* since it runs the latter. We now use a standard "coin-fixing" argument to reduce this time-complexity. Note that

$$\mathbf{Adv}^{\mathrm{prf}}_{h}(A_7) \;=\; \mathbf{E}_{M_1,M_2}\left[\mathbf{Adv}^{\mathrm{prf}}_{h}(A_6(M_1,M_2))\right]$$

where the expectation is over $(M_1, M_2) \xleftarrow{\$} A^*$. Thus there must exist distinct $M_1, M_2 \in B^+$ ($\|M_1\|_b \le \|M_2\|_b$) such that $\mathbf{Adv}^{\mathrm{prf}}_{h}(A_6(M_1,M_2)) \ge \mathbf{Adv}^{\mathrm{prf}}_{h}(A_7)$. Let A be the prf-adversary that has M_1, M_2 hardwired as part of its code and, given oracle g, runs $A_6^g(M_1, M_2)$. Since the latter has time complexity $O(mT_h)$, the proof of Lemma 1 is complete.

4 MAC-security of NMAC under weaker assumptions

Since any PRF is a secure MAC [6, 9], Theorem 1 implies that NMAC is a secure MAC if the compression function is a PRF. Here we show that NMAC is a secure MAC under a weaker-than-PRF assumption on the compression function —namely that it is a privacy-preserving MAC— coupled with the assumption that the hash function is cAU. This is of interest given the still numerous usages of HMAC as a MAC (rather than as a PRF). This result can be viewed as attempting to formalize the intuition given in [3, Remark 4.4].

MAC forgery. Recall that the mac-advantage of mac-adversary A against a family of functions $f\colon$ *Keys* $\times$ *Dom* $\to$ *Rng* is

$$\mathbf{Adv}^{\mathrm{mac}}_{f}(A) \;=\; \Pr\left[A^{f(K,\cdot),\mathrm{VF}_f(K,\cdot,\cdot)} \text{ forges} \;:\; K \xleftarrow{\$} \mathit{Keys}\right] .$$

The verification oracle $\mathrm{VF}_f(K, \cdot, \cdot)$ associated to f takes input M, T, returning 1 if $f(K, M) = T$ and 0 otherwise. Queries to the first oracle are called mac queries, and ones to the second are called verification queries. A is said to forge if it makes a verification query M, T the response to which is 1 but M was not previously a mac query. Note we allow multiple verification queries [9].

Privacy-preserving MACs. The privacy notion for MACs that we use adapts the notion of left-or-right indistinguishability of encryption [5] to functions that are deterministic, and was first introduced by [8] who called it indistinguishability under distinct chosen-plaintext attack. An oracle query of an ind-adversary A against family $f\colon \{0,1\}^k \times \{0,1\}^l \to \{0,1\}^L$ is a pair of l-bit strings. The reply is provided by one or the other of the following games:

Game Left	**Game** Right
$K \xleftarrow{\$} \{0,1\}^c$	$K \xleftarrow{\$} \{0,1\}^c$
On query (x_0, x_1):	**On query** (x_0, x_1):
Reply $f(K, x_0)$	Reply $f(K, x_1)$

Each game has an initialization step in which it picks a key; it then uses this key in computing replies to all the queries made by A. The ind-advantage of A is

$$\mathbf{Adv}_f^{\mathrm{ind}}(A) \;=\; \Pr\left[\,A^{\mathrm{Right}} \Rightarrow 1\,\right] - \Pr\left[\,A^{\mathrm{Left}} \Rightarrow 1\,\right] .$$

However, unlike for encryption, the oracles here are deterministic. So A can easily win (meaning, obtain a high advantage), for example by making a pair of queries of the form $(x,z),(y,z)$, where x,y,z are distinct, and then returning 1 iff the replies returned are the same. (We expect that $h(K,x) \neq h(K,y)$ with high probability over K for functions h of interest, for example compression functions.) We fix this by simply outlawing such behavior. To be precise, let us say that A is *legitimate* if for any sequence $(x_0^1,x_1^1),\ldots,(x_0^q,x_1^q)$ of oracle queries that it makes, $x_0^1,\ldots,x_0^q$ are all distinct l-bit strings, and also $x_1^1,\ldots,x_1^q$ are all distinct l-bit strings. (As a test, notice that the adversary who queried $(x,z),(y,z)$ was not legitimate.) It is to be understood henceforth that an ind-adversary means a legitimate one. When we say that f is privacy-preserving, we mean that the ind-advantage of any (legitimate) practical ind-adversary is low.

Privacy-preservation is not, by itself, a demanding property. For example, it is achieved by a constant family such as the one defined by $f(K,x) = 0^L$ for all K,x. We will however want the property for families that are also secure MACs.

PP-MAC < PRF. We claim that a privacy-preserving MAC (PP-MAC) is strictly weaker than a PRF, in the sense that any PRF is (a secure MAC [6, 9] and) privacy-preserving, but not vice-versa. This means that when (below) we assume that a compression function h is a PP-MAC, we are indeed assuming less of it than that it is a PRF. Let us now provide some details about the claims made above. First, the following is the formal statement corresponding to the claim that any PRF is privacy-preserving:

Proposition 1. *Let $f\colon \{0,1\}^k \times \{0,1\}^l \to \{0,1\}^L$ be a family of functions, and A_{ind} an ind-adversary against it that makes at most q oracle queries and has time-complexity at most t. Then there is a prf-adversary A_{prf} against f such that $\mathbf{Adv}_f^{\mathrm{ind}}(A_{\mathrm{ind}}) \;\leq\; 2 \cdot \mathbf{Adv}_f^{\mathrm{prf}}(A_{\mathrm{prf}})$. Furthermore, A_{prf} makes at most q oracle queries and has time-complexity at most t.* ▮

The proof is a simple exercise and is omitted. Next we explain why a PP-MAC need not be a PRF. The reason (or one reason) is that if the output of a family of functions has some structure, for example always ending in a 0 bit, it would disqualify the family as a PRF but need not preclude its being a PP-MAC. To make this more precise, let $f\colon \{0,1\}^k \times \{0,1\}^l \to \{0,1\}^L$ be a PP-MAC. Define $g\colon \{0,1\}^k \times \{0,1\}^l \to \{0,1\}^{L+1}$ by $g(K,x) = f(K,x)\|0$ for all $K \in \{0,1\}^k$ and $x \in \{0,1\}^l$. Then g is also a PP-MAC, but is clearly not a PRF.

RESULTS. The following implies that if h is a PP-MAC and F is cAU then their composition hF is a secure MAC.

Lemma 3. *Let $B = \{0,1\}^b$. Let $h\colon \{0,1\}^c \times B \to \{0,1\}^c$ and $F\colon \{0,1\}^k \times D \to B$ be families of functions, and let $hF\colon \{0,1\}^{c+k} \times D \to \{0,1\}^c$ be defined by*

$$hF(K_{\mathrm{out}}\|K_{\mathrm{in}}, M) = h(K_{\mathrm{out}}, F(K_{\mathrm{in}}, M))$$

for all $K_{\text{out}} \in \{0,1\}^c, K_{\text{in}} \in \{0,1\}^k$ and $M \in D$. Let A_{hF} be a mac-adversary against hF that makes at most q_{mac} mac queries and at most q_{vf} verification queries, with the messages in each of these queries being of length at most n. Suppose A_{hF} has time-complexity at most t. Let $q = q_{\text{mac}} + q_{\text{vf}}$ and assume $2 \le q < 2^b$. Then there exists a mac-adversary A_1 against h, an ind-adversary A_2 against h, and an au-adversary A_F against F such that

$$\mathbf{Adv}^{\text{mac}}_{hF}(A_{hF}) \le \mathbf{Adv}^{\text{mac}}_{h}(A_1) + \mathbf{Adv}^{\text{ind}}_{h}(A_2) + \binom{q}{2} \cdot \mathbf{Adv}^{\text{au}}_{F}(A_F) . \quad (5)$$

Furthermore, A_1 makes at most q_{mac} mac queries and at most q_{vf} verification queries and has time-complexity at most t; A_2 makes at most q oracle queries and has time-complexity at most t; and A_F outputs messages of length at most n, makes 2 oracle queries, and has time-complexity $O(T_F(n))$, where $T_F(n)$ is the time to compute F on an n-bit input. ∎

The proof is in [2]. As a corollary we have the following, which says that if h is a PP-MAC and h^* is cAU then GNMAC is a secure MAC.

Theorem 2. *Assume $b \ge c$ and let $B = \{0,1\}^b$. Let h: $\{0,1\}^c \times B \to \{0,1\}^c$ be a family of functions and let $\mathsf{fpad} \in \{0,1\}^{b-c}$ be a fixed padding string. Let* GNMAC: $\{0,1\}^{2c} \times B^+ \to \{0,1\}^c$ *be defined by*

$$\mathsf{GNMAC}(K_{\text{out}} \| K_{\text{in}}, M) = h(K_{\text{out}}, h^*(K_{\text{in}}, M) \| \mathsf{fpad})$$

for all $K_{\text{out}}, K_{\text{in}} \in \{0,1\}^c$ and $M \in B^+$. Let A_{GNMAC} be a mac-adversary against GNMAC *that makes at most q_{mac} mac queries and at most q_{vf} verification queries, with the messages in each of these queries being of at most m blocks. Suppose A_{GNMAC} has time-complexity at most t. Let $q = q_{\text{mac}} + q_{\text{vf}}$ and assume $2 \le q < 2^b$. Then there exists a mac-adversary A_1 against h, an ind-adversary A_2 against h, and an au-adversary A^* against h^* such that*

$$\mathbf{Adv}^{\text{mac}}_{\mathsf{GNMAC}}(A_{\mathsf{GNMAC}}) \le \mathbf{Adv}^{\text{mac}}_{h}(A_1) + \mathbf{Adv}^{\text{ind}}_{h}(A_2) + \binom{q}{2} \cdot \mathbf{Adv}^{\text{au}}_{h^*}(A^*) . \quad (6)$$

Furthermore, A_1 makes at most q_{mac} mac queries and at most q_{vf} verification queries and has time-complexity at most t; A_2 makes at most q oracle queries and has time-complexity at most t; and A^* outputs messages of at most m blocks, makes 2 oracle queries, and has time-complexity $O(mT_h)$, where T_h is the time for one computation of h. ∎

We remark that Lemma 3 can be extended to show that hF is not only a MAC but itself privacy-preserving. (This assumes h is privacy-preserving and F is cAU. We do not prove this here.) This implies that GNMAC is privacy-preserving as long as h is privacy-preserving and h^* is cAU. This is potentially useful because it may be possible to show that a PP-MAC is sufficient to ensure security in some applications where HMAC is currently assumed to be a PRF.

5 Security of HMAC

In this section we show how our security results about NMAC lift to corresponding ones about HMAC. We begin by recalling the observation of [3] as to how this works for HMAC with two independent keys, and then discuss how to extend this to the single-keyed version of HMAC.

THE CONSTRUCTS. Let h: $\{0,1\}^c \times \{0,1\}^b \to \{0,1\}^c$ as usual denote the compression function. Let pad be the padding function as described in Section 3, so that $s^* = s \| \mathsf{pad}(|s|) \in B^+$ for any string s. Recall that the cryptographic hash function H associated to h is defined by $H(M) = h^*(\mathrm{IV}, M^*)$, where IV is a c-bit initial vector that is fixed as part of the description of H and M is a string of any length up to some maximum length that is related to pad. (This maximum length is 2^{64} for current hash functions.) Then $\mathsf{HMAC}(K_{\mathrm{out}} \| K_{\mathrm{in}}, M) = H(K_{\mathrm{out}} \| H(K_{\mathrm{in}} \| M))$, where $K_{\mathrm{out}}, K_{\mathrm{in}} \in \{0,1\}^b$. If we write this out in terms of h^* alone we get

$$\mathsf{HMAC}(K_{\mathrm{out}} \| K_{\mathrm{in}}, M) \;=\; h^*(\mathrm{IV},\; K_{\mathrm{out}} \| h^*(\mathrm{IV}, K_{\mathrm{in}} \| M \| \mathsf{pad}(b+|M|)) \| \mathsf{pad}(b+c)\;)\;.$$

As with NMAC, the details of the padding conventions are not important to the security of HMAC as a PRF, and we will consider the more general construct GHMAC: $\{0,1\}^{2b} \times B^+ \to \{0,1\}^c$ defined by

$$\mathsf{GHMAC}(K_{\mathrm{out}} \| K_{\mathrm{in}}, M) = h^*(\mathrm{IV},\; K_{\mathrm{out}} \;\|\; h^*(\mathrm{IV}, K_{\mathrm{in}} \| M) \;\|\; \mathsf{fpad}\;) \qquad (7)$$

for all $K_{\mathrm{out}}, K_{\mathrm{in}} \in \{0,1\}^b$ and all $M \in B^+$. Here $\mathrm{IV} \in \{0,1\}^c$ and $\mathsf{fpad} \in \{0,1\}^{b-c}$ are fixed strings. HMAC is a special case, via $\mathsf{HMAC}(K_{\mathrm{out}} \| K_{\mathrm{in}}, M) = \mathsf{GHMAC}(M \| \mathsf{pad}(b+|M|))$ with $\mathsf{fpad} = \mathsf{pad}(b+c)$, and thus security properties of GHMAC (as a PRF or MAC) are inherited by HMAC, allowing us to focus on the former.

THE DUAL FAMILY. To state the results, it is useful to define $\overline{h}$: $\{0,1\}^b \times \{0,1\}^c \to \{0,1\}^c$, the *dual* of family h, by $\overline{h}(x,y) = h(y,x)$. The assumption that h is a PRF when keyed by its data input is formally captured by the assumption that $\overline{h}$ is a PRF.

SECURITY OF GHMAC. Let $K'_{\mathrm{out}} = h(\mathrm{IV}, K_{\mathrm{out}})$ and $K'_{\mathrm{in}} = h(\mathrm{IV}, K_{\mathrm{in}})$. The observation of [3] is that

$$\begin{aligned}\mathsf{GHMAC}(K_{\mathrm{out}} \| K_{\mathrm{in}}, M) &= h(K'_{\mathrm{out}}, h^*(K'_{\mathrm{in}}, M) \| \mathsf{fpad}) \\ &= \mathsf{GNMAC}(K'_{\mathrm{out}} \| K'_{\mathrm{in}}, M)\;. \qquad (8)\end{aligned}$$

This effectively reduces the security of GHMAC to GNMAC. Namely, if $\overline{h}$ is a PRF and $K_{\mathrm{out}}, K_{\mathrm{in}}$ are chosen at random, then $K'_{\mathrm{out}}, K'_{\mathrm{in}}$ will be computationally close to random. Now (8) implies that if GNMAC is a PRF then so is GHMAC. The formal statement follows.

Lemma 4. *Assume $b \ge c$ and let $B = \{0,1\}^b$. Let h: $\{0,1\}^c \times B \to \{0,1\}^c$ be a family of functions. Let* $\mathsf{fpad} \in \{0,1\}^{b-c}$ *be a fixed padding string and* $\mathrm{IV} \in \{0,1\}^c$ *a fixed initial vector. Let* GHMAC: $\{0,1\}^{2b} \times B^+ \to \{0,1\}^c$ *be defined by (7) above. Let A be a prf-adversary against* GHMAC *that has time-complexity at most t. Then there exists a prf-adversary $A_{\overline{h}}$ against $\overline{h}$ such that*

$$\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{GHMAC}}(A) \;\le\; 2 \cdot \mathbf{Adv}^{\mathrm{prf}}_{\overline{h}}(A_{\overline{h}}) + \mathbf{Adv}^{\mathrm{prf}}_{\mathsf{GNMAC}}(A)\;.$$

Furthermore, $A_{\overline{h}}$ makes only 1 *oracle query, this being* IV*, and has time-complexity at most t.* ∎

The proof is simple and can be found in [2]. Combining this with Theorem 1 yields the result that GHMAC is a PRF assuming $h, \overline{h}$ are both PRFs. Note that the PRF assumption on $\overline{h}$ is mild because $A_{\overline{h}}$ makes only one oracle query.

SINGLE-KEYED HMAC. $\mathsf{HMAC}, \mathsf{GHMAC}$ as described and analyzed above above use two keys that are assumed to be chosen independently at random. However, HMAC is in fact usually implemented with these keys derived from a single b-bit key. Here we provide the first security proofs for the actually-implemented single-key version of HMAC.

Specifically, let $\mathsf{opad}, \mathsf{ipad} \in \{0,1\}^b$ be distinct, fixed and known constants. (Their particular values can be found in [3] and are not important here.) Then the single-key version of HMAC is defined by

$$\mathsf{HMAC\text{-}1}(K, M) \;=\; \mathsf{HMAC}(K \oplus \mathsf{opad} \| K \oplus \mathsf{ipad}, M)\ .$$

As before, we look at this as a special case of a more general construct, namely $\mathsf{GHMAC\text{-}1}\colon\ \{0,1\}^b \times B^+ \to \{0,1\}^c$, defined by

$$\mathsf{GHMAC\text{-}1}(K, M) = \mathsf{GHMAC}(K \oplus \mathsf{opad} \| K \oplus \mathsf{ipad}, M) \tag{9}$$

for all $K \in \{0,1\}^b$ and all $M \in B^+$. We now focus on $\mathsf{GHMAC\text{-}1}$. We will show that $\mathsf{GHMAC\text{-}1}$ inherits the security of GNMAC as long as $\overline{h}$ is a PRF against an appropriate class of related key attacks. In such an attack, the adversary can obtain input-output examples of h under keys related to the target key. Let us recall the formal definitions following [7].

A related-key attack on a family of functions $\overline{h}\colon\ \{0,1\}^b \times \{0,1\}^c \to \{0,1\}^c$ is parameterized by a set $\Phi \subseteq \mathsf{Maps}(\{0,1\}^b, \{0,1\}^b)$ of *key-derivation* functions. We define the function $\mathrm{RK}\colon\ \Phi \times \{0,1\}^b \to \{0,1\}^b$ by $\mathrm{RK}(\phi, K) = \phi(K)$ for all $\phi \in \Phi$ and $K \in \{0,1\}^b$. A rka-adversary $A_{\overline{h}}$ may make an oracle query of the form ϕ, x where $\phi \in \Phi$ and $x \in \{0,1\}^c$. Its rka-advantage is defined by

$$\mathbf{Adv}^{\mathrm{rka}}_{\overline{h},\Phi}(A_{\overline{h}}) \;=\; \Pr\left[A_{\overline{h}}^{\overline{h}(\mathrm{RK}(\cdot,K),\cdot)} \Rightarrow 1\right] - \Pr\left[A_{\overline{h}}^{G(\mathrm{RK}(\cdot,K),\cdot)} \Rightarrow 1\right]\ .$$

In the first case, K is chosen at random from $\{0,1\}^b$ and the reply to query ϕ, x of $A_{\overline{h}}$ is $\overline{h}(\phi(K), x)$. In the second case, $G \xleftarrow{\$} \mathsf{Maps}(\{0,1\}^b \times \{0,1\}^c, \{0,1\}^c)$ and $K \xleftarrow{\$} \{0,1\}^b$, and the reply to query ϕ, x of $A_{\overline{h}}$ is $G(\phi(K), x)$. For any string $s \in \{0,1\}^b$ let $\Delta_s\colon\ \{0,1\}^b \to \{0,1\}^b$ be defined by $\Delta_s(K) = K \oplus s$.

Lemma 5. *Assume $b \geq c$ and let $B = \{0,1\}^b$. Let $h\colon\ \{0,1\}^c \times B \to \{0,1\}^c$ be a family of functions. Let $\mathsf{fpad} \in \{0,1\}^{b-c}$ be a fixed padding string, $\mathrm{IV} \in \{0,1\}^c$ a fixed initial vector, and $\mathsf{opad}, \mathsf{ipad} \in \{0,1\}^b$ fixed, distinct strings. Let $\mathsf{GHMAC\text{-}1}\colon\ \{0,1\}^b \times B^+ \to \{0,1\}^c$ be defined by (9) above. Let $\Phi = \{\Delta_{\mathsf{opad}}, \Delta_{\mathsf{ipad}}\}$. Let A be a prf-adversary against $\mathsf{GHMAC\text{-}1}$ that has time-complexity at most t. Then there exists a rka-adversary $A_{\overline{h}}$ against $\overline{h}$ such that*

$$\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{GHMAC\text{-}1}}(A) \;\leq\; \mathbf{Adv}^{\mathrm{rka}}_{\overline{h},\Phi}(A_{\overline{h}}) + \mathbf{Adv}^{\mathrm{prf}}_{\mathsf{GNMAC}}(A)\ .$$

Furthermore, $A_{\overline{h}}$ makes 2 oracle queries, these being $\Delta_{\mathsf{opad}}, \mathrm{IV}$ and $\Delta_{\mathsf{ipad}}, \mathrm{IV}$, and has time-complexity at most t. ∎

The proof is simple and can be found in [2]. Combining this with Theorem 1 yields the result that $\mathsf{GHMAC\text{-}1}$ is a PRF assuming h is a PRF and $\overline{h}$ is a PRF under Φ-restricted related-key attacks, where Φ is as in Lemma 5. We remark that Φ is a small set of simple functions, which is important because it is shown in [7] that if Φ is too rich then no family can be a PRF under Φ-restricted related-key attacks. Furthermore, the assumption on $\overline{h}$ is rendered milder by the fact that $A_{\overline{h}}$ makes only two oracle queries, in both of which the message is the same, namely is the initial vector.

Lifting the results of Section 4. The procedure above to lift the NMAC results of Section 3 to HMAC applies also to lift the results of Section 4 to HMAC. Specifically, if h is a PP-MAC, h^* is AU and $\overline{h}$ is a PRF then GHMAC is a (privacy-preserving) MAC. Also if h is a PP-MAC, h^* is AU and $\overline{h}$ is a PRF under Φ-restricted related-key attacks, with Φ as in Lemma 5, then $\mathsf{GHMAC\text{-}1}$ is a (privacy-preserving) MAC. Note that the assumption on $\overline{h}$ continues to be that it is a PRF or PRF against Φ-restricted related-key attacks. (Namely, this has not been reduced to its being a PP-MAC.) This assumption is however mild in this context since (as indicated by Lemmas 5 and 4) it need only hold with respect to adversaries that make very few queries and these of a very specific type.

Remarks. Let h: $\{0,1\}^{128} \times \{0,1\}^{512} \rightarrow \{0,1\}^{128}$ denote the compression function of MD5 [28]. An attack by den Boer and Bosselaers [15] finds values x_0, x_1, K such that $h(x_0, K) = h(x_1, K)$ but $x_0 \neq x_1$. In a personal communication, Rijmen has said that it seems possible to extend this to an attack that finds such x_0, x_1 even when K is unknown. If so, this might translate into the following attack showing h is not a PRF when keyed by its data input. (That is, $\overline{h}$ is not a PRF.) Given an oracle g: $\{0,1\}^{128} \rightarrow \{0,1\}^{128}$, the attacker would find x_0, x_1 and obtain $y_0 = g(x_0)$ and $y_1 = g(x_1)$ from the oracle. It would return 1 if $y_0 = y_1$ and 0 otherwise. How does this impact the above, where we are assuming $\overline{h}$ is a PRF? Interestingly, the actual PRF assumptions we need on $\overline{h}$ are so weak that even such an attack does not break them. In Lemma 4, we need $\overline{h}$ to be a PRF only against adversaries that make just one oracle query. (Because $A_{\overline{h}}$ makes only one query.) But the attack above makes two queries. On the other hand, in Lemma 5, we need $\overline{h}$ to be a related-key PRF only against adversaries that make two related-key queries in both of which the 128-bit message for $\overline{h}$ is the same, this value being the initial vector used by the hash function. Furthermore, the related key functions must be $\Delta_{\mathsf{opad}}, \Delta_{\mathsf{ipad}}$. The above-mentioned attack, however, uses two different messages x_0, x_1 and calls the oracle under the original key rather than the related keys. In summary, the attack does not violate the assumptions made in either of the lemmas.

Acknowledgments

Thanks to Ran Canetti, Hugo Krawczyk, Mridul Nandi, Vincent Rijmen, Phillip Rogaway, Victor Shoup, Paul Van Oorschot and the Crypto 2006 PC for comments and references. Author supported in part by NSF grants ANR-0129617, CCR-0208842 and CNS-0524765 and by an IBM Faculty Partnership Development Award.

References

1. American National Standards Institution. ANSI X9.71, Keyed hash message authentication code, 2000.
2. M. Bellare. New Proofs for NMAC and HMAC: Security without Collision-Resistance. Full version of this paper. *Cryptology ePrint Archive*: Report 2006/043, 2006.
3. M. Bellare, R. Canetti and H. Krawczyk. Keying hash functions for message authentication. *Advances in Cryptology – CRYPTO '96*, Lecture Notes in Computer Science Vol. 1109, N. Koblitz ed., Springer-Verlag, 1996.
4. M. Bellare, R. Canetti and H. Krawczyk. Pseudorandom functions revisited: The cascade construction and its concrete security. `http://www-cse.ucsd.edu/users/mihir`. (Preliminary version in *Proceedings of the 37th Symposium on Foundations of Computer Science*, IEEE, 1996.)
5. M. Bellare, A. Desai, E. Jokipii and P. Rogaway. A concrete security treatment of symmetric encryption. *Proceedings of the 38th Symposium on Foundations of Computer Science*, IEEE, 1997.
6. M. Bellare, J. Kilian and P. Rogaway. The security of the cipher block chaining message authentication code. *Journal of Computer and System Sciences*, Vol. 61, No. 3, Dec 2000, pp. 362–399.
7. M. Bellare and T. Kohno. A theoretical treatment of related-key attacks: RKA-PRPs, RKA-PRFs, and applications. *Advances in Cryptology – EUROCRYPT '03*, Lecture Notes in Computer Science Vol. 2656, E. Biham ed., Springer-Verlag, 2003.
8. M. Bellare, C. Namprempre and T. Kohno. Authenticated Encryption in SSH: Provably Fixing the SSH Binary Packet Protocol. *ACM Transactions on Information and System Security (TISSEC)*, Vol. 7, Iss. 2, May 2004, pp. 206–241.
9. M. Bellare, O. Goldreich and A. Mityagin. The power of verification queries in message authentication and authenticated encryption. *Cryptology ePrint Archive*: Report 2004/309, 2004.
10. M. Bellare and P. Rogaway. The game-playing technique and its application to triple encryption. *Cryptology ePrint Archive*: Report 2004/331, 2004.
11. J. Black, S. Halevi, H. Krawczyk, T. Krovetz and P. Rogaway. UMAC: Fast and Secure Message Authentication. *Advances in Cryptology – CRYPTO '99*, Lecture Notes in Computer Science Vol. 1666, M. Wiener ed., Springer-Verlag, 1999.
12. J. Black and P. Rogaway. CBC MACs for arbitrary-length messages: The three-key constructions. *Advances in Cryptology – CRYPTO '00*, Lecture Notes in Computer Science Vol. 1880, M. Bellare ed., Springer-Verlag, 2000.
13. L. Carter and M. Wegman. Universal classes of hash functions. *Journal of Computer and System Sciences*, Vol. 18, No. 2, 1979, pp. 143–154.
14. I. Damgård. A design principle for hash functions. *Advances in Cryptology – CRYPTO '89*, Lecture Notes in Computer Science Vol. 435, G. Brassard ed., Springer-Verlag, 1989.
15. B. den Boer and A. Bosselaers. Collisions for the compression function of MD5. *Advances in Cryptology – EUROCRYPT '93*, Lecture Notes in Computer Science Vol. 765, T. Helleseth ed., Springer-Verlag, 1993.
16. T. Dierks and C. Allen. The TLS protocol. Internet RFC 2246, 1999.

17. H. Dobbertin, A. Bosselaers and B. Preneel. RIPEMD-160: A strengthened version of RIPEMD. *Fast Software Encryption '96*, Lecture Notes in Computer Science Vol. 1039, D. Gollmann ed., Springer-Verlag, 1996.
18. Y. Dodis, R. Gennaro, J. Håstad, H. Krawczyk, and T. Rabin. Randomness extraction and key derivation using the CBC, Cascade, and HMAC modes. *Advances in Cryptology – CRYPTO '04*, Lecture Notes in Computer Science Vol. 3152, M. Franklin ed., Springer-Verlag, 2004.
19. O. Goldreich, S. Goldwasser and S. Micali. How to construct random functions. *Journal of the ACM*, Vol. 33, No. 4, 1986, pp. 210–217.
20. D. Harkins and D. Carrel. The Internet Key Exchange (IKE). Internet RFC 2409, 1998.
21. S. Hirose. A note on the strength of weak collision resistance. *IEICE Transactions on Fundamentals*, Vol. E87-A, No. 5, May 2004, pp. 1092–1097.
22. H. Krawczyk, M. Bellare and R. Canetti. HMAC: Keyed-hashing for message authentication. Internet RFC 2104, 1997.
23. R. Merkle. One-way hash functions and DES. *Advances in Cryptology – CRYPTO '89*, Lecture Notes in Computer Science Vol. 435, G. Brassard ed., Springer-Verlag, 1989.
24. D. M'Raihi, M. Bellare, F. Hoornaert, D. Naccache, O. Ranen. HOTP: An HMAC-based one time password algorithm. Internet RFC 4226, December 2005.
25. National Institute of Standards and Technology. The keyed-hash message authentication code (HMAC). FIPS PUB 198, March 2002.
26. National Institute of Standards and Technology. Secure hash standard. FIPS PUB 180-2, August 2000.
27. B. Preneel and P. van Oorschot. On the security of iterated message authentication codes. *IEEE Transactions on Information Theory*, Vol. 45, No. 1, January 1999, pp. 188–199. (Preliminary version, entitled "MD-x MAC and building fast MACs from hash functions," in CRYPTO 95.)
28. R. Rivest. The MD5 message-digest algorithm. Internet RFC 1321, April 1992.
29. V. Shoup. Sequences of games: A tool for taming complexity in security proofs. *Cryptology ePrint Archive*: Report 2004/332, 2004.
30. D. Stinson. Universal hashing and authentication codes. *Designs, Codes and Cryptography*, Vol. 4, 1994, 369–380.
31. X. Wang, Y. L. Yin and H. Yu. Finding collisions in the full SHA-1. *Advances in Cryptology – CRYPTO '05*, Lecture Notes in Computer Science Vol. 3621 , V. Shoup ed., Springer-Verlag, 2005.
32. X. Wang and H. Yu. How to break MD5 and other hash functions. *Advances in Cryptology – EUROCRYPT '05*, Lecture Notes in Computer Science Vol. 3494, R. Cramer ed., Springer-Verlag, 2005.
33. M. Wegman and L. Carter. New hash functions and their use in authentication and set equality. *Journal of Computer and System Sciences*, Vol. 22, No. 3, 1981, pp. 265–279.