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

Mihir Bellare*

November 1, 2005

Abstract

HMAC [2] is a cryptographic-hash-function-based MAC that is widely used both as a MAC and as a PRF. To prove it secure as a PRF, previous work [2] assumed both that the compression function was a PRF and that the hash function was weakly collision resistant. In this paper we show that the second assumption is unnecessary, proving the construct is a PRF (and hence a MAC) under the sole assumption that the compression function is a PRF. This helps explain the strength that HMAC has shown even when implemented with hash functions like MD5 and SHA-1 whose weak collision resistance is (fully in the first case and partially in the second) compromised by known attacks. (Known attacks do not compromise the compression functions as PRFs.) 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 MAC as long as the hash function meets the very weak requirement of being computationally almost universal. (Again, known attacks do not invalidate these assumptions.)

Keywords: Message authentication, hash functions, Pseudorandom Functions, Carter-Wegman.

*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`. Supported in part by NSF grants ANR-0129617 and CCR-0208842, and by an IBM Faculty Partnership Development Award.

Contents

1 Introduction

HMAC, designed by Bellare, Canetti and Krawczyk [2], is a popular cryptographic hash function based MAC. It 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.

This paper provides new security proofs for the construct that support its use even in the face of the collision-finding attacks we are now seeing emerge on the underlying hash functions. Furthermore, one of these proofs is the first to be based *only* on an assumption about the underlying compression function. Let us begin by recalling the construction.

The constructions, high level. The basic construct is actually NMAC, of which HMAC can be viewed as a derivative. Succinctly:

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

Here H is a cryptographic hash function (eg. MD5 [30], SHA-1 [27], RIPEMD-160 [16]) while H^*, which we call the extended hash function, is the hash function with the initial vector made explicit as an (additional) first input. Both functions 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, where b is the underlying block length and c the length of the chaining variable. (Typically $b = 512$ while c is 128 or 160.) NMAC keys the extended hash function via the initial vector, while HMAC is non-intrusive, applying only the cryptographic hash function.

The constructions, details. Let h: $\{0,1\}^c \times \{0,1\}^b \to \{0,1\}^c$ denote the underlying compression function, and h^* the iterated compression function which on input $K \in \{0,1\}^c$ and a message $x = x[1]\ldots x[n]$ that is viewed as a sequence 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 $a[n]$. (Here $|x|$ must be a positive multiple of b bits.) Then $H^*(K, M) = h^*(K, M^*)$ where M^* denotes M padded appropriately to reach a length that is a positive multiple of b. (For current hash functions, the padding encodes $|M|$, the so-called Merkle-Damgård strengthening.) Finally, $H(M) = H^*(\text{IV}, M)$ where IV is a (public) c-bit initial vector that is fixed as part of the description of H. We note that the outer application of H^* in NMAC in fact consists of just a single application of h, so that

$$\mathsf{NMAC}(K_{\text{out}}\|K_{\text{in}}, M) = h(K_{\text{out}}, H^*(K_{\text{in}}, M)\|\mathsf{fpad}) \tag{1}$$

where fpad is the fixed $b - c$ bit string that would result from padding a c-bit string. This last formula enables a better understanding of the analyses.

HMAC as a PRF. HMAC is often used as a PRF (pseudorandom function [18]) rather than merely as a MAC. In particular this is the case when it is used for key-derivation, as in TLS [15], IKE (the Internet Key Exchange protocol of IPSEC) [20] and NIST's key establishment draft standard [26]. HMAC is also used as a PRF in a proposed standard for one-time passwords [24]. Thus from the perspective of current practice, we feel it is important that the security property of NMAC and HMAC that one establish is that they are PRFs rather than just MACs. (Since any PRF is a MAC [5, 7], their security as MACs would be immediate. Note it is common to establish the security of a MAC by first establishing it is a PRF. This is done for example for various CBC-MACs [5, 28, 11, 8] and PMAC [12].)

Known results. The main result of [2] is about NMAC. They then discuss how this lifts to HMAC. We will follow this approach too, and postpone the discussion of HMAC. NMAC is proven in [2] to be a secure MAC assuming that the underlying compression function h is a secure MAC and also that the hash function H is *weakly collision resistant*. The latter is a relaxation of collision resistance 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 $H^*(K, M_1) = H^*(K, M_2)$. This results extends in a natural way to show that NMAC is a PRF (rather than merely a MAC) if the compression function is a PRF and the hash function continues to be weakly collision resistant.

Our first result on NMAC. We show (cf. Theorem 3.3) 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 weakly collision-resistant is dropped. (And, in particular, as long as h is a PRF, the conclusion is true even if H is *not* weakly collision resistant.) We now discuss some features and implications of our result.

Mirroring the MD paradigm. The appeal of the Merkle-Damgård [23, 14] iteration paradigm (used in all current cryptographic hash functions) is that the collision-resistance of the hash function is proved based *only* on the collision-resistance of the compression function. A designer needs thus to focus only on the building block. Our result mirrors this for NMAC as a PRF, proving it has the property in question (being a PRF) based *only* on the assumption that the compression function also has the *same* property. Thus, once again, design can focus only on the building block.

The loss of weak collision resistance. To understand what follows we first need to explain something about weak collision resistance. Namely, although it appears to be a weaker requirement than collision resistance 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 (eg. those on MD5 [34] and SHA-1 [33]) typically 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 weak collision resistance, meaning to find a collision in $H^*(K,\cdot)$, given an oracle for this function, even with K hidden. (This extension attack was pointed out already in [2], and, for the curious, we recall it in Appendix A.)

Security without (weak) collision-resistance. Most implementations of HMAC use MD5 or SHA-1. Yet these are hash functions whose collision-resistance is (fully in the first case and partially in the second) compromised [34, 33]. Remarkably, these attacks do not compromise HMAC-MD5 or HMAC-SHA-1. (Being iterated MACs, the generic birthday based forgery attacks of [29] always break NMAC and HMAC in time $2^{c/2}$, but no better-than-birthday attacks are known for HMAC-MD5 or HMAC-SHA-1.) This strength (which is an important part of the appeal of HMAC) is not an accident. (It was a design goal.) However, the proofs of [2] do not explain this strength, because, as explained above, the known attacks compromise the weak collision resistance of the hash functions to the same extent that they compromise collision resistance, and thus the assumptions made for the security proof of [2] either do not hold (for MD5) or may not hold (for SHA-1). Put another way, the constructs have proved to be stronger than their proofs. (This has been well-known for some time.)

However, there are to date no attacks that compromise the pseudorandomness of the compression functions of MD5 or SHA-1. (That is, the assumption that the compression function is a PRF is still valid in these cases.) So our security proof applies, and perhaps helps explain and understand why these constructs "work" in the absence of collision-resistance.

A closer look. We establish our first result about NMAC (that it is a PRF if the compression function is one) via two lemmas. The first (Lemma 3.1) says that if a compression function h is a PRF then the iterated compression function h^* (and hence also the extended hash function H^*) is *computationally almost universal* (cAU). This means that it is computationally infeasible for an adversary to find a pair of distinct messages M_1, M_2 such that $\mathsf{Coll}_{h^*}(M_1, M_2)$, defined as the probability that $h^*(K, M_1) = h^*(K, M_2)$ when key K is chosen at random, is not small. This is a relaxation of the standard notion of almost universality (AU) [13, 35, 32] which would ask that $\mathsf{Coll}_{h^*}(M_1, M_2)$ be small for *every* distinct M_1, M_2. AU (and cAU) are very weak forms of collision resistance compared even to weak collision resistance. The second lemma (Lemma 3.2) says that the composition of a PRF and a cAU function is a PRF. (We call this the PRF(cAU) = PRF lemma. It relaxes the analogous PRF(AU) = PRF lemma [3, 10, 11].) Putting the two lemmas together yields our result. We note that our reduction is tight up to a dependence on the message lengths, meaning justifies NMAC up to the birthday attack of [29]. (That is, up to $2^{c/2}$ queries as long as the messages are short.)

In the past, AU families have been the basis of fast MACs [21, 10], while the PRF(AU) = PRF lemma has been used to simplify [11] and tighten [8] the proof that the encrypted CBC MAC is a PRF, and also applied to other MACs in the CBC family [11]. In all these cases, however, AU has been viewed as a combinatorial property and established unconditionally. Unusually, in our case, AU is viewed as a computational property and is established under an assumption.

A (self-contained) proof of Lemma 3.1 (that h a PRF implies h^* is cAU) is in Section 3.3 . We also indicate in Section 3.4 a different (not self-contained) proof that h a PRF implies h^* is cAU. This exploits the result of [3] that says that if h is a PRF then h^* (which they call the cascade) is a "pf-PRF." (A prefix-free PRF, meaning a PRF as long as no query of the adversary is a prefix of another query. Note that if h^* were a PRF, it would imply it is cAU in a very direct way, but h^* is not a PRF due to the extension attack. The trick for the reduction-to-pf-PRF proof is to reduce bounding $\mathsf{Coll}_{h^*}(M_1, M_2)$, for M_1 a prefix of M_2, to bounding $\mathsf{Coll}_{h^*}(M_1', M_2')$ for messages M_1', M_2' related to M_1, M_2 but neither a prefix of the other.) We have preferred our direct proof because it achieves a (slightly) better reduction factor than the reduction-from-pf-PRF proof, and also because our direct proof is simpler than the one of [3]. We thank Victor Shoup for reminding is of the above-mentioned trick (leading to the reduction-from-pf-PRF proof) after seeing an early draft of this paper that contained only a direct proof.

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 [2] 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 (cf. Section 4).

However, the analysis above assumes that the two keys $K_{\text{out}}, K_{\text{in}}$ 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 [6] to extend the observation of [2] 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 (cf. Section 4). 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.

THE PRF ASSUMPTION ON h. Is it safe to assume a compression function is a PRF? Obviously, we don't know for sure. We do note that there is a block cipher underlying a compression function, and it is standard to assume a block cipher is a PRP or PRF. What is perhaps more relevant is that practice, by using HMAC as a PRF, is implicitly already assuming the compression function is a PRF. (Given the outer application of h in (1), it is hard to imagine NMAC being a PRF without h being one.) What we are saying is that this seemingly necessary condition is also sufficient. It is hard to ask more of a proof.

However, one good question is whether the assumption on h could be relaxed if we wanted to establish the security of NMAC and HMAC merely as MACs. Next, we offer a result in this vein.

Our second result on NMAC. We show (cf. Theorem 5.3) that NMAC is a secure MAC if h^* (equivalently, H^*) is cAU and h is a *privacy-preserving MAC*. (A privacy-preserving MAC, defined in Section 5, is stronger than a MAC but weaker than a PRF.) Underlying this is an extension of the Prf(cAU)=Prf lemma that we call the PP-MAC(cAU)=MAC lemma (cf. Lemma 5.2) which says that the composition of a privacy-preserving MAC and a cAU function is a secure MAC. We note that here we assume (rather than prove) the cAU of h^*, but assume less of h than in our first result. The value of this result (as compared to the weak collision resistance based proof of [2]) is again that, even when the underlying hash function is MD5 or SHA-1, there are no known attacks that invalidate the assumptions made. We remark that, under the same assumptions, we actually establish something stronger, namely that NMAC is itself privacy-preserving. This is of interest because there may be applications where HMAC is currently assumed to be a PRF but in fact privacy-preservation is enough. Again, these results about NMAC lift to HMAC as discussed above.

Further related work. Dodis, Gennaro, Håstad, Krawczyk and Rabin [17] consider the cascade construction defined over a family of random functions. (This means that $h(K,\cdot)$ is a random function for each $K \in \{0,1\}^c$. This is like Shannon's ideal cipher model, except the component maps are functions not permutations.) Here they show [17, Lemma 4] that the cascade is AU as long as the two messages whose collision probability one considers have the same length. This does not appear to imply Lemma 3.1 (showing the cascade h^* is cAU if h is a PRF), for two reasons. First, we need to allow the two messages to have different lengths. Second, it is not clear to us 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 [17, Lemma 5] in their 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 (c)AU property rather than randomness, and our inputs are not drawn from a distribution.

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

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

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: $Keys \times Dom \to Rng$ whose first argument is regarded as a key. We fix one such family h: $\{0,1\}^c \times B \to \{0,1\}^c$ to model a compression function that we regard as being keyed via its chaining variable. Here b is the block length and c is the length of the chaining variable. Typical values are $b = 512$ and $c = 128$ or 160. The *iteration* of family h: $\{0,1\}^c \times B \to \{0,1\}^c$ is the family of functions h^*: $\{0,1\}^c \times B^+ \to \{0,1\}^c$ defined as follows:

Function $h^*(K, M)$ // $K \in \{0,1\}^c, M \in B^+$
$n \leftarrow \|M\|_b$; $a[0] \leftarrow K$
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: $Keys \times Dom \to Rng$ takes as oracle a function g: $Dom \to Rng$ 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 $K \xleftarrow{\$} Keys$, and the probability that it outputs 1 when its oracle g is chosen at random from the set $\mathsf{Maps}(Dom, Rng)$ of all functions mapping Dom to Rng, succinctly written as

$$\mathbf{Adv}_f^{\mathrm{prf}}(A) \;=\; \Pr\left[A^{f(K,\cdot)} \Rightarrow 1\right] - \Pr\left[A^{\$} \Rightarrow 1\right] . \tag{2}$$

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

Computational almost-universality. Let F: $\{0,1\}^k \times Dom \to Rng$ be a family of functions. For $M_1, M_2 \in Dom$ let $\mathsf{Coll}_F(M_1, M_2) = \Pr[F(K, M_1) = F(K, M_2)]$, the probability being over $K \xleftarrow{\$} \{0,1\}^k$. Recall that F is ϵ-almost-universal if $\mathsf{Coll}_F(M_1, M_2) \le \epsilon$ for any distinct $M_1, M_2 \in Dom$ [32, 13, 35]. We introduce a computational relaxation of this information-theoretic notion. An almost-universal (au) adversary A against F (takes no inputs and) returns a pair of messages in Dom. Its au-advantage is

$$\mathbf{Adv}_F^{\mathrm{au}}(A) \;=\; \Pr\left[F(K, M_1) = F(K, M_2) \;\wedge\; M_1 \ne M_2 \;:\; (M_1, M_2) \xleftarrow{\$} A\,;\; K \xleftarrow{\$} Keys\right] .$$

Security and resources. As usual with the concrete security approach [5], there is no formal notion of security of a primitive (eg. a PRF) but informally, when we talk of, say, f being a PRF, we mean that the prf-advantage of any (no input) prf-adversary of "practical" resources is "low." Similarly H is computationally au (cAU) if the au-advantage of any adversary of "practical" resources is "low". Formal results state the concrete security of reductions, bounding the advantage and resources of one adversary as a function of those of others.

The following conventions will be adopted in measuring resource usage. The time-complexity of an adversary is defined as the total execution time of an overlying experiment (meaning, includes not only the running time of the adversary but the time to compute replies to oracle queries and the time to perform any initializations or to test whether the adversary was successful) plus the size of the code of the adversary, in some fixed model of computation. (In cases like PRFs, where equation (2) defining the advantage involves two experiments, we consider the maximum of the two execution times, with the convention that the picking of a random function is done by building an on-line table while responding to oracle queries. More details on these conventions will appear when they are used.) When we say that a resource measure (such as the time-complexity, number of oracle queries, or their lengths) is at most a certain value, we mean this holds for all coin tosses of the adversary and regardless of how its oracle queries are answered. With regard to time-complexity we will permit ourselves the use of big-oh notation, even though there are no asymptotics here, with the intent that it hides some constant depending only on the model of computation.

MACs. Any PRF is a MAC. (This was established for the basic notion of MAC security in [5], but holds even for the most stringent notions and with tight security reductions [7].) Accordingly, in the main part of this paper, which establishes security of NMAC and HMAC as PRFs, we will not explicitly discuss their security as MACs. We will provide MAC-related definitions in Section 5.1, where we consider the security of NMAC and HMAC as MACs under weaker-than-PRF assumptions on the compression function.

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 $\mathsf{NMAC}(K_{\rm out}\|K_{\rm in}, M) = h(K_{\rm out}, h^*(K_{\rm in}, M^*)\|\mathsf{fpad})$ 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 $\mathsf{GNMAC}\colon \{0,1\}^{2c} \times B^+ \to \{0,1\}^c$ 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

h^* is cAU. Towards the proof that GNMAC is a PRF, the first step is the following lemma, which says that if h is a PRF then its iteration h^* is computationally almost-universal.

Lemma 3.1 *Let $B = \{0,1\}^b$. Let $h\colon \{0,1\}^c \times B \to \{0,1\}^c$ 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 $n_1, n_2 \ge 1$ blocks long, respectively. Then there exists a prf-adversary A against h such that*

$$\mathbf{Adv}^{\rm au}_{h^*}(A^*) \;\le\; (n_1+n_2)\cdot \mathbf{Adv}^{\rm prf}_h(A) + \frac{1}{2^c}\,. \tag{3}$$

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. We remark that 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.) Furthermore A makes only two oracle queries. With such limited resources, there is little one would expect such an adversary can do to break a well-designed h as a PRF.

One might ask whether something stronger is true, namely that h^* is itself a PRF assuming h is one. (This would imply it is cAU.) The answer is no: h^* is *not* a PRF, due to the extension attack. It is however shown in [3] to be a pf-PRF (namely a PRF if no query of the adversary is a prefix of another) assuming h is a PRF.

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

Lemma 3.2 *Let $B = \{0,1\}^b$. Let $h\colon \{0,1\}^c \times B \rightarrow \{0,1\}^c$ and $F\colon \{0,1\}^k \times D \rightarrow B$ be families of functions, and let $hF\colon \{0,1\}^{c+k} \times D \rightarrow \{0,1\}^c$ be defined by*

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

for all $K_{\mathrm{out}} \in \{0,1\}^c, K_{\mathrm{in}} \in \{0,1\}^k$ 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

$$\mathbf{Adv}^{\mathrm{prf}}_{hF}(A_{hF}) \;\leq\; \mathbf{Adv}^{\mathrm{prf}}_{h}(A_h) + \binom{q}{2} \cdot \mathbf{Adv}^{\mathrm{au}}_{F}(A_F)\,. \tag{4}$$

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 referred to in [3, 10], and variants are in [10, 11]. A simple proof of Lemma 3.2, using games [9, 31], is in Section 3.5.

At first glance the reduction of Lemma 3.2 may look loose due to the $\binom{q}{2}$ factor. (And in some settings this is indeed the case and has lead to the use of alternative constructs [10].) However in our case this factor turns out only to reflect the existing birthday attack on GNMAC [29] and thus does not reflect a loose reduction. Note that the time-complexity of A_F is small and in particular independent of the time-complexity of A_{hF}.

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

Theorem 3.3 *Assume $b \geq c$ and let $B = \{0,1\}^b$. Let $h\colon \{0,1\}^c \times B \rightarrow \{0,1\}^c$ be a family of functions and let $\mathsf{fpad} \in \{0,1\}^{b-c}$ be a fixed padding string. Let $\mathsf{GNMAC}\colon \{0,1\}^{2c} \times B^+ \rightarrow \{0,1\}^c$ 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

$$\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{GNMAC}}(A_{\mathsf{GNMAC}}) \;\leq\; \mathbf{Adv}^{\mathrm{prf}}_{h}(A_1) + \binom{q}{2}\left[2m \cdot \mathbf{Adv}^{\mathrm{prf}}_{h}(A_2) + \frac{1}{2^c}\right]. \tag{5}$$

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 of Theorem 3.3: Define $F\colon \{0,1\}^c \times B^+ \rightarrow \{0,1\}^b$ by $F(K_{\mathrm{in}}, M) = h^*(K_{\mathrm{in}}, M) \| \mathsf{fpad}$. Then $\mathsf{GNMAC} = hF$. Apply Lemma 3.2 (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 3.1 to $A^* = A_F$ to get prf-adversary A_2. ▮

3.2 Remarks

Theorem 3.3 delivers, first and foremost, an important qualitative guarantee about the security of GNMAC. For practical purposes, however, one should also look at the quantitative implications.

Quantitative implications. If t is a time-complexity then let $\bar{t} = t/T_h$. The best known attacks on GNMAC are either the birthday one of [29] or exhaustive key search. This implies that with q queries and time-complexity t, one can achieve a prf-advantage of about $\alpha(t,q) = (\bar{t} + q^2)2^{-c}$ against GNMAC. This hits 1 when $\bar{t} = q \approx 2^{c/2}$, leading to an estimate of about $2^{c/2}$ queries (and time) to break NMAC.

Theorem 3.3, on the other hand, provides an upper bound on the prf-advantage of *any* adversary of time-complexity t who makes q queries. How close is this to $\alpha(t,q)$? To answer this question, 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. Plugging this into (5) 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 $\beta(t,q,m) = O(\bar{t} + m^2q^2T_F) \cdot 2^{-c}$. Roughly, this matches $\alpha(t,q)$ up to factors that depend on the number m of blocks in the message. In particular, if we ignore the m^2 and T_F terms, then this hits 1 when $\bar{t} = q \approx 2^{c/2}$. This means that the bound justifies NMAC up to roughly $2^{c/2}$ queries in the case that the queried messages are short. (We note that the messages in the attack of [29] are short.)

We do not expect to be able to entirely remove the dependence on m from the upper bound. (Rather, we imagine it possible that the attacks could be extended so that the lower bound grows as some function of m.) But we imagine that it might be possible to reduce it from the current quadratic to a smaller function of m, perhaps via techniques from [17, 8].

Uniformity versus non-uniformity. We draw attention to an aspect of our reductions that may not be apparent in the concrete security framework we are using, but becomes so when we lift the results to an asymptotic setting. In the latter setting, all the function families we are considering become collections of such families, indexed by a security parameter, and to say that a collection of families is a PRF means that the prf-advantage (now a function of the security parameter) of any polynomial-time adversary is negligible. Then, the PRF assumption on h that we need for our results is with respect to non-uniform rather than uniform adversaries. (A uniform adversary is a polynomial-time Turing Machine, while a non-uniform one is a family of polynomial-sized circuits.) We could, if we wished, obtain the results also with the uniform version of the assumption, but at the cost of increasing the resource usage of some of the constructed adversaries. Specifically, the time-complexity of A in Lemma 3.1 would be that of A^* rather than independent of it as now; that of A_F in Lemma 3.2 would be that of A_{hF}; and that of A_2 in Theorem 3.3 would be that of A_{GNMAC}. This would affect the quantitative implications. (It would justify NMAC up to only about $2^{c/3}$ queries rather than $2^{c/2}$.)

We do not consider the use of a non-uniform (as opposed to uniform) PRF assumption to be an issue, because such assumptions are common and standard and also the distinction is moot in the concrete security setting. However, neither do we want to "push this issue under the rug," hence the remark above.

Game $G_1(M_1, M_2, l)$ // $0 \le l \le \|M_1\|_b$
$m_1 \leftarrow \|M_1\|_b$; $m_2 \leftarrow \|M_2\|_b$
$a[l] \xleftarrow{\$} \{0,1\}^c$
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 $A_1^g(M_1, M_2, l)$ // $1 \le l \le \|M_1\|_b$
	$m_1 \leftarrow \|M_1\|_b$; $m_2 \leftarrow \|M_2\|_b$
	$a[l] \leftarrow g(M_2[l])$
	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

Figure 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: $\{0,1\}^b \to \{0,1\}^c$.

AU RATHER THAN cAU. The use of cAU as opposed to AU is more one of convenience than necessity. In fact one can strengthen Lemma 3.1 to show that h^* is AU rather than merely cAU. More precisely, for any integer m let $\epsilon(m)$ be the maximum, over all n_1, n_2 such that $n_1 + n_2 \le m$, and over all prf-adversaries A that have time complexity $O(mT_h)$ and make at most 2 oracle queries, of $2m \cdot \mathbf{Adv}_h^{\mathrm{prf}}(A) + 2^{-c}$. Let $B^{\le m}$ denote the set of all non-empty strings of at most m blocks, and let h^m: $\{0,1\}^c \times B^{\le m} \to \{0,1\}^c$ be the restriction of h^* to inputs in $B^{\le m}$. Then we claim that h^m is $\epsilon(m)$-AU. To see this, for any distinct $M_1, M_2 \in B^{\le m}$ let $A^*_{M_1,M_2}$ be the au-adversary that outputs M_1, M_2 and halts. Let A_{M_1,M_2} be the corresponding prf-adversary given by Lemma 3.1. The lemma tells us that

$$\mathsf{Coll}_{h^m}(M_1, M_2) \;=\; \mathbf{Adv}_{h^m}^{\mathrm{au}}(A^*_{M_1,M_2}) \;\le\; 2m \cdot \mathbf{Adv}_h^{\mathrm{prf}}(A_{M_1,M_2}) + 2^{-c} \;,$$

and our claim follows because this is true for all M_1, M_2. Note that this version of the lemma continues to assume h is a PRF. (This assumption is now implicit in the value of $\epsilon(m)$.)

3.3 Proof of Lemma 3.1

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 $a[m_2] = h^*(a[m_1+1], M_2[m_1+2] \ldots M_2[m_2])$. 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 3.4 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

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

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 $K \xleftarrow{\$} \{0,1\}^c$.

Proof of Claim 3.4: $A_1^g(M_1, M_2, l)$ sets $a[l] = g(M_2[l])$. If g is chosen at random then this is equivalent to the $a[l] \xleftarrow{\$} \{0,1\}^c$ assignment in $G_1(M_1, M_2, l)$. On the other hand if $g = h(K, \cdot)$ for a random K, then K plays the role of $a[l-1]$ in $G_1(M_1, M_2, l-1)$. $\blacksquare$

Claim 3.5 Let $M_1, M_2 \in B^+$ with $M_1 \subseteq M_2$ and $1 + \|M_1\|_b \leq \|M_2\|_b$. Then

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

Proof of Claim 3.5: Suppose g is chosen at random. Since $y \neq M_2[m_1+1]$, the quantity $g(y)$ is not defined until the last line of the code of A_2, at which point $h(a[m_2], y)$ is fixed, and thus the probability that the two are equal is 2^{-c} due to the randomness of $g(y)$. Now suppose $g = h(K, \cdot)$ for a random K. Think of K as playing the role of $a[m_1]$ in $G_1(M_1, M_2, m_1)$. Then $a[m_2] = K$ in $A_2^g(M_1, M_2)$ exactly when $a[m_1] = a[m_2]$ in $G_1(M_1, M_2, m_1)$, meaning exactly when the latter game returns 1. But if $a[m_2] = K$ then certainly $h(a[m_2], y) = h(K, y)$, and the latter is $g(y)$, so $A_1^g(M_1, M_2)$ will return 1. (However, it could be that $h(a[m_2], y) = h(K, y)$ even if $a[m_2] \neq K$, which is why we have an inequality rather than an equality in the claim.) $\blacksquare$

Claim 3.6 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

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

Proof of Claim 3.6: From the description of A_3, whether $g = \$$ or $g = h(K, \cdot)$,

$$\Pr\left[A_3^g(M_1, M_2) \Rightarrow 1 \right] = \frac{1}{m_1+1} \left(\Pr\left[A_2^g(M_1, M_2) \Rightarrow 1 \right] + \sum_{l=1}^{m_1} \Pr\left[A_1^g(M_1, M_2, l) \Rightarrow 1 \right] \right) .$$

Now Claims 3.5 and 3.4 imply that $\Pr\left[\, A_3^{h(K,\cdot)}(M_1,M_2) \Rightarrow 1 \,\right]$ is

$$\begin{aligned} &\geq \frac{1}{m_1+1}\left(\Pr\left[\, G_1(M_1,M_2,m_1) \Rightarrow 1 \,\right] + \sum_{l=1}^{m_1}\Pr\left[\, G_1(M_1,M_2,l-1) \Rightarrow 1 \,\right]\right) \\ &= \frac{1}{m_1+1}\cdot\sum_{l=0}^{m_1}\Pr\left[\, G_1(M_1,M_2,l) \Rightarrow 1 \,\right] . \end{aligned} \tag{6}$$

On the other hand, Claims 3.5 and 3.4 also imply that $\Pr\left[\, A_3^{\$}(M_1,M_2) \Rightarrow 1 \,\right]$ is

$$= \frac{1}{m_1+1}\left(2^{-c} + \sum_{l=1}^{m_1}\Pr\left[\, G_1(M_1,M_2,l) \Rightarrow 1 \,\right]\right) . \tag{7}$$

Subtracting (7) from (6) and exploiting the cancellation of terms, we get

$$\mathbf{Adv}_h^{\mathrm{prf}}(A_3(M_1,M_2)) \;\geq\; \frac{1}{m_1+1}\left(\Pr\left[\, G_1(M_1,M_2,0) \Rightarrow 1 \,\right] - 2^{-c}\right) .$$

Now examination of Game $G_1(M_1,M_2,0)$ shows that that in this game, $a[m_1] = h^*(a[0],M_1)$, $a[m_2] = h^*(a[0],M_2)$, and $a[0]$ is selected at random. Since the game returns 1 iff $a[m_1] = a[m_2]$, the probability that it returns 1 is exactly $\mathsf{Coll}_{h^*}(M_1,M_2)$. ▮

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 $M_1[1]\ldots M_1[p] = M_2[1]\ldots M_2[p]$ but $M_1[p+1] \neq M_2[p+1]$. We consider the games and adversaries of Figure 2.

Claim 3.7 Let $M_1, M_2 \in B^+$ with $M_1 \not\subseteq M_2$, and $\|M_1\|_b \leq \|M_2\|_b$. Suppose $1 \leq l \leq \|M_1\|_b + \|M_2\|_b - \mathrm{LCP}(M_1,M_2)$. Then

$$\begin{aligned} \Pr\left[\, A_4^{\$}(M_1,M_2,l) \Rightarrow 1 \,\right] &= \Pr\left[\, G_2(M_1,M_2,l) \Rightarrow 1 \,\right] \\ \Pr\left[\, A_4^{h(K,\cdot)}(M_1,M_2,l) \Rightarrow 1 \,\right] &= \Pr\left[\, G_2(M_1,M_2,l-1) \Rightarrow 1 \,\right] . \; ▮ \end{aligned}$$

Proof of Claim 3.7: The first equality, namely the one where $g \xleftarrow{\$} \mathsf{Maps}(\{0,1\}^b, \{0,1\}^c)$, is quite easy to see.Let us compare the code of $G_2(M_1,M_2,l)$ and $A_4^g(M_1,M_2,l)$ and look at places where they differ. The first such place is line a20, but since g is random, this is equivalent to line 220. Then we get to line a50. This is equivalent to what line 250 does for $l = p+1$ because $g(M_2[n])$ is distributed uniformly and independently of anything else. (This is true because g is random and we know that $M_2[p+1] \neq M_1[p+1]$ because $p = \mathrm{LCP}(M_1,M_2)$ and M_1 is not a prefix of M_2.) Next, line a60 is equivalent to line 260 because g is random and, in this case, has not previously been invoked.

The second equality, namely the one where $g \xleftarrow{\$} \mathsf{Maps}(\{0,1\}^b, \{0,1\}^c)$, needs more work. We will justify it by considering the games of Figure 3. We claim that

$$\begin{aligned} \Pr\left[\, G_2(M_1,M_2,l-1) \Rightarrow 1 \,\right] &= \Pr\left[\, G_3(M_1,M_2,l) \Rightarrow 1 \,\right] && (8) \\ &= \Pr\left[\, G_4(M_1,M_2,l) \Rightarrow 1 \,\right] && (9) \\ &= \Pr\left[\, G_5(M_1,M_2,l) \Rightarrow 1 \,\right] && (10) \\ &= \Pr\left[\, A_4^{h(K,\cdot)}(M_1,M_2,l) \Rightarrow 1 \,\right] && (11) \end{aligned}$$

Game $G_2(M_1, M_2, l)$ // $0 \le l \le \|M_1\|_b + \|M_2\|_b - \mathrm{LCP}(M_1, M_2)$
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 $A_4^g(M_1, M_2, l)$ // $1 \le l \le \|M_1\|_b + \|M_2\|_b - \mathrm{LCP}(M_1, M_2)$
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

Figure 2: Games and adversaries taking input distinct messages $M_1, M_2 \in B^+$ such that $M_1 \not\subseteq M_2$ and $\|M_1\|_b \le \|M_2\|_b$.

We justify the above relations in turn by arguing that the games are equivalent.

Game $G_3(M_1, M_2, l)$ is just a copy of $G_2(M_1, M_2, l-1)$, so (8) is clear.

Now we argue that $G_3(M_1, M_2, l)$ is equivalent to $G_4(M_1, M_2, l)$. The changes to 310, 330 resulting in 410, 430 have simply moved the case $l = m_1 + 1$, covered by 310, into 430. This is right because when $l = m_1 + 1$, line 320 results in $a_1[m_1]$ being chosen at random. Line 440 has been formed by pulling one iteration of 370 into 340 for the case $1 \le l \le p$, while the case $l = p + 1$ of 340 is covered similarly by 441. Line 450 is clearly equivalent to 350. Lines 460, 461 have been formed by pulling one iteration of 370 into 360. Line 370 results in the same value of $a_2[m_2]$ as 470 because, although we have in some cases incremented n by one in $G_4(M_1, M_2, l)$ as compared to its value in $G_3(M_1, M_2, l)$, we have in these cases pulled the first iteration of the loop of 370 into some previous statement. This justifies (9).

Next we argue that $G_4(M_1, M_2, l)$ is equivalent to $G_5(M_1, M_2, l)$. Line 520 is equivalent to 420 because K was chosen at random at 501. To obtain line 540 from 440, we first use the fact that $a_1[l-1] = K$ from 520. Now since $1 \le l = n \le p < m_1$ we have $M_2[n] = M_1[n]$, and thus from 440 we have $h(a_2[n-1], M_2[n]) = h(a_1[n-1], M_1[n])$. But this equals $a_1[n]$ due to 520. LIne 541 follows from 441 because $a_1[p] = K$ due to 520. (Here $l = p + 1$.) Line 550 mimics 450 but excludes the case $l = m_1 + 1$ of the latter. This case is now covered by 560, 561. (In the case $l = m_1 + 1$, these lines set $n = p + 2$ and $a_2[p + 1] = K$. The latter is equivalent to the random choice of $a_2[p + 1]$

Game $G_3(M_1, M_2, l)$ // $1 \leq l \leq \|M_1\|_b + \|M_2\|_b - \mathrm{LCP}(M_1, M_2)$

300 $m_1 \leftarrow \|M_1\|_b$; $m_2 \leftarrow \|M_2\|_b$; $p \leftarrow \mathrm{LCP}(M_1, M_2)$
310 If $0 \leq l-1 \leq m_1$ then
320 $\quad a_1[l-1] \stackrel{\$}{\leftarrow} \{0,1\}^c$; For $i = l$ to m_1 do $a_1[i] \leftarrow h(a_1[i-1], M_1[i])$
330 If $m_1 + 1 \leq l-1 \leq m_1 + m_2 - p$ then $a_1[m_1] \stackrel{\$}{\leftarrow} \{0,1\}^c$
340 If $0 \leq l-1 \leq p$ then $n \leftarrow l-1$; $a_2[n] \leftarrow a_1[n]$
350 If $p+1 \leq l-1 \leq m_1$ then $n \leftarrow p+1$; $a_2[n] \stackrel{\$}{\leftarrow} \{0,1\}^c$
360 If $m_1 + 1 \leq l-1 \leq m_1 + m_2 - p$ then $n \leftarrow l - m_1 + p$; $a_2[n] \stackrel{\$}{\leftarrow} \{0,1\}^c$
370 For $i = n+1$ to m_2 do $a_2[i] \leftarrow h(a_2[i-1], M_2[i])$
380 If $a_1[m_1] = a_2[m_2]$ then return 1 else return 0

Game $G_4(M_1, M_2, l)$ // $1 \leq l \leq \|M_1\|_b + \|M_2\|_b - \mathrm{LCP}(M_1, M_2)$

400 $m_1 \leftarrow \|M_1\|_b$; $m_2 \leftarrow \|M_2\|_b$; $p \leftarrow \mathrm{LCP}(M_1, M_2)$
410 If $1 \leq l \leq m_1$ then
420 $\quad a_1[l-1] \stackrel{\$}{\leftarrow} \{0,1\}^c$; For $i = l$ to m_1 do $a_1[i] \leftarrow h(a_1[i-1], M_1[i])$
430 If $m_1 + 1 \leq l \leq m_1 + m_2 - p$ then $a_1[m_1] \stackrel{\$}{\leftarrow} \{0,1\}^c$
440 If $1 \leq l \leq p$ then $a_2[l-1] \leftarrow a_1[l-1]$; $n \leftarrow l$; $a_2[n] \leftarrow h(a_2[n-1], M_2[n])$
441 If $l = p+1$ then $a_2[p] \leftarrow a_1[p]$; $n \leftarrow p+1$; $a_2[n] \leftarrow h(a_2[n-1], M_2[n])$
450 If $p+2 \leq l \leq m_1 + 1$ then $n \leftarrow p+1$; $a_2[n] \stackrel{\$}{\leftarrow} \{0,1\}^c$
460 If $m_1 + 2 \leq l \leq m_1 + m_2 - p$ then
461 $\quad n \leftarrow l - m_1 + p + 1$; $a_2[n-1] \stackrel{\$}{\leftarrow} \{0,1\}^c$; $a_2[n] \leftarrow h(a_2[n-1], M_2[n])$
470 For $i = n+1$ to m_2 do $a_2[i] \leftarrow h(a_2[i-1], M_2[i])$
480 If $a_1[m_1] = a_2[m_2]$ then return 1 else return 0

Game $G_5(M_1, M_2, l)$ // $1 \leq l \leq \|M_1\|_b + \|M_2\|_b - \mathrm{LCP}(M_1, M_2)$

500 $m_1 \leftarrow \|M_1\|_b$; $m_2 \leftarrow \|M_2\|_b$; $p \leftarrow \mathrm{LCP}(M_1, M_2)$
501 $K \stackrel{\$}{\leftarrow} \{0,1\}^c$
510 If $1 \leq l \leq m_1$ then
520 $\quad a_1[l-1] \leftarrow K$; For $i = l$ to m_1 do $a_1[i] \leftarrow h(a_1[i-1], M_1[i])$
530 If $m_1 + 1 \leq l \leq m_1 + m_2 - p$ then $a_1[m_1] \stackrel{\$}{\leftarrow} \{0,1\}^c$
540 If $1 \leq l \leq p$ then $n \leftarrow l$; $a_2[n] \leftarrow a_1[n]$
541 If $l = p+1$ then $n \leftarrow p+1$; $a_2[n] \leftarrow h(K, M_2[n])$
550 If $p+2 \leq l \leq m_1$ then $n \leftarrow p+1$; $a_2[n] \stackrel{\$}{\leftarrow} \{0,1\}^c$
560 If $m_1 + 1 \leq l \leq m_1 + m_2 - p$ then
561 $\quad n \leftarrow l - m_1 + p + 1$; $a_2[n-1] \leftarrow K$; $a_2[n] \leftarrow h(a_2[n-1], M_2[n])$
570 For $i = n+1$ to m_2 do $a_2[i] \leftarrow h(a_2[i-1], M_2[i])$
580 If $a_1[m_1] = a_2[m_2]$ then return 1 else return 0

Figure 3: Games equivalent to $G_2(M_1, M_2, l-1)$, where $M_1, M_2 \in B^+$ are such that $M_1 \not\subseteq M_2$ and $\|M_1\|_b \leq \|M_2\|_b$.

in 450 because if $l = m_1 + 1$ then K has not been used before in $G_5(M_1, M_2, l)$. Additionally, one iteration of 470 is pulled into 561, with n incremented accordingly.) On the other hand 560, 561 mimic 460, 461 in the case $m_1 + 2 \leq l \leq m_1 + m_2 - p$ because in this case K has not been used before in $G_5(M_1, M_2, l)$ and thus $a_2[n-1] \leftarrow K$ is equivalent to choosing it at random as in 461. Finally, 570 mimics 470. This justifies (10).

Finally, (11) is easy to see: just identify the key K in the $h(K,\cdot)$ oracle given to A_4 with the one chosen at 501. ▌

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

Claim 3.8 Let $M_1, M_2 \in B^+$ with $M_1 \not\subseteq M_2$ and $\|M_1\|_b \le \|M_2\|_b$. Let $m = \|M_1\|_b + \|M_2\|_b - \mathrm{LCP}(M_1, M_2)$. Then

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

Proof of Claim 3.8: By Claim 3.7,

$$\begin{aligned}
\mathbf{Adv}_h^{\mathrm{prf}}(A_5) &= \frac{1}{m} \cdot \sum_{l=1}^{m} \Pr\left[A_4^{h(K,\cdot)}(M_1, M_2, l) \Rightarrow 1 \right] - \frac{1}{m} \cdot \sum_{l=1}^{m} \Pr\left[A_4^{\$}(M_1, M_2, l) \Rightarrow 1 \right] \\
&= \frac{1}{m} \cdot \sum_{l=1}^{m} \Pr\left[G_2(M_1, M_2, l-1) \Rightarrow 1 \right] - \frac{1}{m} \cdot \sum_{l=1}^{m} \Pr\left[G_2(M_1, M_2, l) \Rightarrow 1 \right] \\
&= \frac{1}{m} \cdot \left(\Pr\left[G_2(M_1, M_2, 0) \Rightarrow 1 \right] - \Pr\left[G_2(M_1, M_2, m) \Rightarrow 1 \right]\right) \ .
\end{aligned}$$

Let $m_1 = \|M_1\|_b$ and $m_2 = \|M_2\|_b$. Examination of Game $G_2(M_1, M_2, 0)$ shows that that, in this game, $a_1[m_1] = h^*(a_1[0], M_1)$, $a_2[m_2] = h^*(a_2[0], M_2)$, and $a_1[0] = a_2[0]$ is selected at random. Since the game returns 1 iff $a_1[m_1] = a_2[m_2]$, the probability that it returns 1 is exactly $\mathsf{Coll}_{h^*}(M_1, M_2)$. On the other hand, since $m_2 \ge m_1 \ge p+1$, the values $a_1[m_1]$ and $a_2[m_2]$ in $G_2(M_1, M_2, m)$ end up being chosen independently at random, and so the probability that they are equal, which is the probability this game returns 1, is 2^{-c}. ▌

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 $A_3^g(M_1, M_2)$	$(M_1, M_2) \stackrel{\$}{\leftarrow} A^*$
Else return $A_5^g(M_1, M_2)$	Return $A_6^g(M_1, M_2)$

Claim 3.9 $\mathbf{Adv}_{h^*}^{\mathrm{au}}(A^*) \le 2^{-c} + (n_1 + n_2) \cdot \mathbf{Adv}_h^{\mathrm{prf}}(A_7)$ ▌

Proof of Claim 3.9: We note that

$$\mathbf{Adv}_{h^*}^{\mathrm{au}}(A^*) = \sum_{M_1 \subseteq M_2} \mathsf{Coll}_{h^*}(M_1, M_2) \cdot \Pr\left[M_1, M_2 \right] + \sum_{M_1 \not\subseteq M_2} \mathsf{Coll}_{h^*}(M_1, M_2) \cdot \Pr\left[M_1, M_2 \right]$$

where $\Pr[M_1, M_2]$ denotes the probability that A^* outputs (M_1, M_2). Now use Claims 3.6 and 3.8, and also the assumptions $\|M_1\|_b \le n_1$, $\|M_2\|_b \le n_2$ and $n_2 \ge 1$ from the lemma statement, to get

$$\begin{aligned}
\mathbf{Adv}_{h^*}^{\mathrm{au}}(A^*) &\le \sum_{M_1 \subseteq M_2} \left[(n_1 + n_2) \cdot \mathbf{Adv}_h^{\mathrm{prf}}(A_3(M_1, M_2)) + 2^{-c}\right] \cdot \Pr\left[M_1, M_2 \right] \\
&\quad + \sum_{M_1 \not\subseteq M_2} \left[(n_1 + n_2) \cdot \mathbf{Adv}_h^{\mathrm{prf}}(A_5(M_1, M_2)) + 2^{-c}\right] \cdot \Pr\left[M_1, M_2 \right] \\
&= 2^{-c} + (n_1 + n_2) \cdot \mathbf{Adv}_h^{\mathrm{prf}}(A_7) \ ,
\end{aligned}$$

where in the last line we used the definition of prf-adversary 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

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

where the expectation is over $(M_1, M_2) \stackrel{\$}{\leftarrow} A^*$. Thus there must exist distinct $M_1, M_2 \in B^+$ ($\|M_1\|_b \leq \|M_2\|_b$) such that $\mathbf{Adv}_h^{\mathrm{prf}}(A_6(M_1,M_2)) \geq \mathbf{Adv}_h^{\mathrm{prf}}(A_7)$. 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 3.1 is complete.

3.4 The reduction-from-pf-PRF proof

We sketch how one can obtain the result that h a PRF implies h^* is cAU by using the result of [3] that says that h a PRF implies h^* is a pf-PRF (a PRF as long as no query of the adversary is a prefix of another query). We then compare this with the direct proof given in Section 3.3 .

The result of [3]. A prf-adversary is said to be prefix-free if no query it makes is a prefix of another. The result of [3] is that if D is a prefix-free prf-adversary against h^* that makes at most q queries, each of at most m blocks, then there is a prf-adversary A against h such that

$$\mathbf{Adv}_{h^*}^{\mathrm{prf}}(D) \;\leq\; qm \cdot \mathbf{Adv}_h^{\mathrm{prf}}(A) \tag{12}$$

and A has about the same time-complexity as D. (We remark that there is a typo in the statement of Theorem 3.1 of the proceedings version of [3] in this regard: the factor q is missing from the bound. This is however corrected in the on-line version of the paper.)

The reduction-from-pf-PRF. The result obtained via the reduction-from-pf-PRF proof will be slightly worse than the one of Lemma 3.1. Namely we claim that, under the same conditions as in that lemma, (3) is replaced by

$$\mathbf{Adv}_{h^*}^{\mathrm{au}}(A^*) \;\leq\; 2 \cdot [\max(n_1, n_2) + 1] \cdot \mathbf{Adv}_h^{\mathrm{prf}}(A) + \frac{1}{2^c}\,, \tag{13}$$

and the time-complexity of A increases from $(n_1 + n_2)$ computations of h to $2\max(n_1, n_2)$ computations of h. For some intuition about the proof, imagine h^* were a PRF. (It is not.) Then $\mathsf{Coll}_{h^*}(M_1, M_2)$ would be about the same as the probability that $f(M_1) = f(M_2)$ for a random function f, because otherwise the prf-adversary who queried its oracle with M_1, M_2 and accepted iff the replies were the same would be successful. With h^* in fact only a pf-PRF, the difficulty is the case that $M_1 \subseteq M_2$, which renders the adversary just described not prefix-free. There is a simple (and well-known) observation —we will call it the extension trick— to get around this. Namely, assuming wlog $M_1 \neq M_2$ and $\|M_1\|_b \leq \|M_2\|_b$, let $x \in B$ be a block different from $M_2[\|M_1\|_b + 1]$, and let $M_1' = M_1\|x$ and $M_2' = M_2\|x$. Then $\mathsf{Coll}_{h^*}(M_1, M_2) \leq \mathsf{Coll}_{h^*}(M_1', M_2')$ but M_1 is not a prefix of M_2. This leads to the prefix-free prf-adversary against h^* below:

Adversary D^f

$(M_1, M_2) \stackrel{\$}{\leftarrow} A^*$
If $M_1 \subseteq M_2$ then $x \stackrel{\$}{\leftarrow} B \setminus \{M_2[\|M_1\|_b + 1]\}$; $M_1' \leftarrow M_1\|x$; $M_2' \leftarrow M_2\|x$
Else $M_1' \leftarrow M_1$; $M_2' \leftarrow M_2$
If $f(M_1') = f(M_2')$ then return 1 else return 0

Game $G0$	Game $G1$	Game $G2$
$K_{\mathrm{in}} \xleftarrow{\$} \{0,1\}^k$	$K_{\mathrm{in}} \xleftarrow{\$} \{0,1\}^k$	$g \xleftarrow{\$} \mathsf{Maps}(D, \{0,1\}^c)$
$K_{\mathrm{out}} \xleftarrow{\$} \{0,1\}^c$	$f \xleftarrow{\$} \mathsf{Maps}(\{0,1\}^b, \{0,1\}^c)$	**On query** M:
On query M:	**On query** M:	Reply $g(M)$
Reply $h(K_{\mathrm{out}}, F(K_{\mathrm{in}}, M))$	Reply $f(F(K_{\mathrm{in}}, M))$	

Figure 4: Games $G0, G1, G2$ for the proof of Lemma 3.2.

Here $f\colon B^+ \to \{0,1\}^c$ and we assume wlog that $M_1, M_2 \in B^+$ are distinct messages with $\|M_1\|_b \leq \|M_2\|_b$. Now the result of [3] gives us a prf-adversary A against h such that (12) holds. Thus:

$$\begin{aligned} \mathbf{Adv}^{\mathrm{au}}_{h^*}(A^*) - 2^{-c} &\leq \mathbf{Adv}^{\mathrm{prf}}_{h^*}(D) \\ &\leq 2 \cdot [\max(n_1, n_2) + 1] \cdot \mathbf{Adv}^{\mathrm{prf}}_{h}(A) + 2^{-c}\ . \end{aligned}$$

Re-arranging terms yields (13).

The time-complexity of A as per [3] is (essentially) the time-complexity t of A^*, rather than being a small quantity independent of t as in Lemma 3.1. It is not clear whether or not the coin-fixing argument of our proof of Section 3.3 can be applied to A to reduce this time-complexity. (One would have to enter into the details of the proof of [3] to check.) However, instead, we can first modify A^* to an adversary that has embedded in its code a pair $M_1, M_2 \in B^+$ of distinct messages that maximize $\mathsf{Coll}_{h^*}(M_1, M_2)$. It just outputs these messages and halts. We then apply the argument above to this modified A^*, and now the A we will have time-complexity that of $2\max(n_1, n_2)$ computations of h plus minor overhead.

Comparisons. For the case that $M_1 \subseteq M_2$, our direct proof (meaning the one of Section 3.3) uses a different (and novel) idea as opposed to the extension trick, which leads to a factor of only $n_1 + 1$ in the bound (Claim 3.6) in this case, as opposed to the $2[\max(n_1, n_2) + 1]$ factor obtained via the reduction-from-pf-PRF proof. The difference can be significant in the case that M_2 is long and M_1 is short. In the case $M_1 \not\subseteq M_2$ our direct proof relies heavily on ideas of [3], but avoids the intermediate reduction to the multi-oracle model they use and exploits the non-adaptive nature of the setting to improve the factor in the bound from $2\max(n_1, n_2)$ to $n_1 + n_2$. We clarify that in this case we do not think the improvement is significant in practice, but we think it is aesthetically and theoretically nice to prove the "right" bounds, meaning those that are intuitively what should be there.

The reduction-from-pf-PRF proof is certainly simpler than our direct proof if one is willing to take the result of [3] as given. However, especially for a construct that is as widely standardized as HMAC, we think it is useful to have from-scratch proofs that are as easily verifiable as possible. If the measure of complexity is that of a from-scratch (i.e. self-contained) proof, we contend that our direct one (although not trivial) is simpler than that of [3]. (In particular because we do not use the multi-oracle model.) We remark that if a reader's primary interest is the simplest possible self-contained proof regardless of the quality of the bound, the way to get it is to use our direct proof for the case $M_1 \not\subseteq M_2$ and then the extension trick (as opposed to our direct proof) for the case $M_1 \subseteq M_2$.

3.5 Proof of Lemma 3.2

Game $G0$ of Figure 4 implements an oracle for $hF(K_{\mathrm{out}}\|K_{\mathrm{in}},\cdot)$ with the keys chosen at random, while Game $G2$ implements an oracle for a random function. So

$$
\begin{aligned}
&\mathbf{Adv}_{hF}^{\mathrm{prf}}(A_{hF}) \\
&= \Pr\left[A_{hF}^{G0} \Rightarrow 1\right] - \Pr\left[A_{hF}^{G2} \Rightarrow 1\right] \\
&= \left(\Pr\left[A_{hF}^{G0} \Rightarrow 1\right] - \Pr\left[A_{hF}^{G1} \Rightarrow 1\right]\right) + \left(\Pr\left[A_{hF}^{G1} \Rightarrow 1\right] - \Pr\left[A_{hF}^{G2} \Rightarrow 1\right]\right) \qquad (14)
\end{aligned}
$$

where in the last step we simply subtracted and then added back in the value $\Pr\left[A_{hF}^{G1} \Rightarrow 1\right]$.

It is easy to construct a prf-adversary A_h against h such that

$$\mathbf{Adv}_{h}^{\mathrm{prf}}(A_h) \;=\; \Pr\left[A_{hF}^{G0} \Rightarrow 1\right] - \Pr\left[A_{hF}^{G1} \Rightarrow 1\right]. \qquad (15)$$

(Namely, A_h, given an oracle for a function $f\colon B \to \{0,1\}^c$, picks $K_{\mathrm{in}} \xleftarrow{\$} \{0,1\}^k$. It then runs A_{hF}, replying to oracle query M by $f(F(K_{\mathrm{in}}, M))$, and returns whatever output A_{hF} returns. We omit the simple analysis that establishes (15).) The main part of the proof is to construct au-adversary A'_F against F such that

$$\Pr\left[A_{hF}^{G1} \Rightarrow 1\right] - \Pr\left[A_{hF}^{G2} \Rightarrow 1\right] \;\le\; \binom{q}{2} \cdot \mathbf{Adv}_{F}^{\mathrm{au}}(A'_F). \qquad (16)$$

This adversary, however, will have time-complexity t (and output messages of at most n bits). A standard "coin-fixing" argument will then be used to derive from A'_F an au-adversary A_F that has time-complexity $O(T_F(n))$ (and also outputs messages of n bits) such that

$$\mathbf{Adv}_{F}^{\mathrm{au}}(A'_F) \;\le\; \mathbf{Adv}_{F}^{\mathrm{au}}(A_F). \qquad (17)$$

Combining (14), (15), (16) and (17) we get (4), completing the proof of the lemma.

Towards constructing A'_F, consider Games $G3, G4, G5$ of Figure 5. (Game $G3$ is defined by the code on the left of the Figure. Game $G4$ is the same except that the boxed code-statement is omitted.) We will assume now that a prf-adversary never repeats an oracle query. This is wlog, and is used below without explicit mention.

Claim 3.10 Game $G4$ is equivalent to Game $G2$ while Game $G3$ is equivalent to Game $G1$. ∎

Proof of Claim 3.10: In Game $G4$, the "If" statement does nothing beyond setting the bad flag, and the reply to query M_s is always the random value Z_s. Thus, Game $G4$ implements a random function just like Game $G2$. Game $G3$ returns random values except that it also ensures that if $F(K_{\mathrm{in}}, M_i) = F(K_{\mathrm{in}}, M_j)$ then the answers to queries M_i, M_j are the same. Thus, it is equivalent to Game $G1$. ∎

Now we have:

$$
\begin{aligned}
\Pr\left[A_{hF}^{G1} \Rightarrow 1\right] - \Pr\left[A_{hF}^{G2} \Rightarrow 1\right] &= \Pr\left[A_{hF}^{G3} \Rightarrow 1\right] - \Pr\left[A_{hF}^{G4} \Rightarrow 1\right] && (18)\\
&\le \Pr\left[A_{hF}^{G4} \text{ sets } \mathsf{bad}\right]. && (19)
\end{aligned}
$$

Above, (18) is by Claim 3.10. Since $G3, G4$ differ only in statements that follow the setting of bad, (19) follows from the Fundamental Lemma of Game Playing [9].

$$
\begin{array}{|l|l|}
\hline
\textbf{Games } G3, G4 & \textbf{Game } G5 \\
\quad K_{\text{in}} \xleftarrow{\$} \{0,1\}^k \,;\, s \leftarrow 0 & \quad s \leftarrow 0 \\
\quad Z_1, \ldots, Z_q \xleftarrow{\$} \{0,1\}^c & \quad Z_1, \ldots, Z_q \xleftarrow{\$} \{0,1\}^c \\
\textbf{On query } M\text{:} & \textbf{On query } M\text{:} \\
\quad s \leftarrow s+1 \,;\, M_s \leftarrow M & \quad s \leftarrow s+1 \,;\, M_s \leftarrow M \\
\quad Y_s \leftarrow F(K_{\text{in}}, M_s) & \quad \text{Reply } Z_s \\
\quad \text{If } (\exists\, r < s : Y_r = Y_s) \text{ then} & \\
\quad\quad \mathsf{bad} \leftarrow \mathsf{true} \,;\, \boxed{Z_s \leftarrow Z_r} & \\
\quad \text{Reply } Z_s & \\
\hline
\end{array}
$$

Figure 5: Games $G3, G4$ are defined by the code on the left, where Game $G3$ includes the boxed statement while Game $G4$ does not.

We define au-adversary A'_F against F, as follows: It runs A_{hF}^{G5}, then picks at random i, j subject to $1 \le i < j \le q$, and finally outputs the messages M_i, M_j. In other words, it runs A_{hF}, replying to the oracle queries of the latter with random values, and then outputs a random pair of messages that A_{hF} queries to its oracle. (In order for M_i, M_j to always be defined, we assume A_{hF} always makes *exactly* q oracle queries rather than at most q where by "always" we mean no matter how its oracle queries are replied to. This is wlog.) We claim that

$$
\Pr\left[\, A_{hF}^{G4} \text{ sets } \mathsf{bad} \,\right] \;\le\; \binom{q}{2} \cdot \mathbf{Adv}_F^{\text{au}}(A'_F)\ . \tag{20}
$$

Combining (19) and (20) yields (16). We now justify (20). Intuitively, it is true because i, j are chosen at random after the execution of A_{hF} is complete, so A_{hF} has no information about them. A rigorous proof however needs a bit more work. Consider the experiment defining the au-advantage of A'_F. (Namely, we run A_{hF}^{G5}, pick i, j at random subject to $1 \le i < j \le q$, and then pick $K_{\text{in}} \xleftarrow{\$} \{0,1\}^k$.) In this experiment, consider the following events defined for $1 \le \alpha < \beta \le q$:

$$
\begin{aligned}
C_{\alpha,\beta} \quad &: \quad F(K_{\text{in}}, M_\alpha) = F(K_{\text{in}}, M_\beta) \\
C \quad &: \quad \textstyle\bigvee_{1 \le \alpha < \beta \le q} C_{\alpha,\beta}\ .
\end{aligned}
$$

Notice that the events "$C_{\alpha,\beta} \wedge (i,j) = (\alpha,\beta)$" ($1 \le \alpha < \beta \le q$) are disjoint. (Even though the events $C_{\alpha,\beta}$ for $1 \le \alpha < \beta \le q$ are not.) Thus:

$$
\begin{aligned}
\mathbf{Adv}_F^{\text{au}}(A'_F) \;&=\; \Pr\left[\textstyle\bigvee_{1 \le \alpha < \beta \le q} \left(C_{\alpha,\beta} \wedge (i,j) = (\alpha,\beta)\right)\right] \\
&=\; \sum_{1 \le \alpha < \beta \le q} \Pr\left[\, C_{\alpha,\beta} \wedge (i,j) = (\alpha,\beta) \,\right]\ .
\end{aligned}
$$

Since i, j are chosen at random after the execution of A_{hF}^{G5} is complete, the events "$(i,j) = (\alpha,\beta)$" and $C_{\alpha,\beta}$ are independent. Thus the above equals

$$
\begin{aligned}
\sum_{1 \le \alpha < \beta \le q} \Pr\left[\, C_{\alpha,\beta} \,\right] \cdot \Pr\left[\, (i,j) = (\alpha,\beta) \,\right] \;&=\; \sum_{1 \le \alpha < \beta \le q} \Pr\left[\, C_{\alpha,\beta} \,\right] \cdot \frac{1}{\binom{q}{2}} \\
&=\; \frac{1}{\binom{q}{2}} \cdot \sum_{1 \le \alpha < \beta \le q} \Pr\left[\, C_{\alpha,\beta} \,\right] \\
&\ge\; \frac{1}{\binom{q}{2}} \cdot \Pr\left[\, C \,\right]\ .
\end{aligned}
$$

The proof of (20) is concluded by noting that $\Pr[C]$ equals $\Pr[G4 \text{ sets } \mathsf{bad}]$.

Finally, note that

$$\mathbf{Adv}^{\mathrm{au}}_F(A'_F) \;=\; \mathbf{E}_{M_1,M_2}\left[\mathsf{Coll}_F(M_1,M_2)\right]$$

where the expectation is over $(M_1,M_2) \xleftarrow{\$} A'_F$. Thus there must exist $M_1, M_2 \in B^+$ such that $\mathsf{Coll}_F(M_1,M_2) \geq \mathbf{Adv}^{\mathrm{au}}_F(A'_F)$. (And these messages are distinct because A_{hF} never repeats an oracle query.) Let A_F be the au-adversary that has M_1, M_2 hardwired as part of its code and, when run, simply outputs these messages and halts. Then (17) follows. Furthermore the time complexity of A_F is $O(mT_h)$. (Remember that by our convention the time-complexity is that of the overlying experiment, so includes the time to compute T_F on the messages that A_F outputs.)

4 Security of HMAC

In this section we discuss how security results about NMAC lift to corresponding ones about HMAC. We begin by recalling the observation of [2] 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

$$\mathsf{GHMAC}(K_{\mathrm{out}}\|K_{\mathrm{in}}, M) = h^*(\mathrm{IV},\; K_{\mathrm{out}} \,\|\, h^*(\mathrm{IV}, K_{\mathrm{in}}\|M) \,\|\, \mathsf{fpad}\;) \tag{21}$$

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.

4.1 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 [2] is that

$$\begin{aligned}\mathsf{GHMAC}(K_{\mathrm{out}}\|K_{\mathrm{in}}, M) &= h(K'_{\mathrm{out}}, h^*(K'_{\mathrm{in}}, M)\|\mathsf{fpad}) \\ &= \mathsf{GNMAC}(K'_{\mathrm{out}}\|K'_{\mathrm{in}}, M)\;.\end{aligned} \tag{22}$$

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 (22) implies that if GNMAC is a PRF then so is GHMAC. The formal statement follows.

Lemma 4.1 *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. 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 (21) 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*

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

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

The proof is simple and is omitted. Combining this with Theorem 3.3 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.

4.2 Single-keyed HMAC

HMAC, 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 [2] 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 GHMAC-1: $\{0,1\}^b \times B^+ \to \{0,1\}^c$, defined by

$$\mathsf{GHMAC\text{-}1}(K, M) = \mathsf{GHMAC}(K \oplus \mathsf{opad} \| K \oplus \mathsf{ipad}, M) \qquad (23)$$

for all $K \in \{0,1\}^b$ and all $M \in B^+$. We now focus on GHMAC-1. We will show that GHMAC-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 [6].

A related-key attack on a family of functions $\overline{h}$: $\{0,1\}^b \times \{0,1\}^c \to \{0,1\}^c$ is parameterized by a set $\Phi \subseteq \mathsf{Maps}(\{0,1\}^b, \{0,1\}^b)$ of *key-derivation* functions. We define the function RK: $\Phi \times \{0,1\}^b \to \{0,1\}^b$ by $\mathrm{RK}(\phi, K) = \phi(K)$ for all $\phi \in \Phi$ and $K \in \{0,1\}^b$. A rka-adversary A 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

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

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

Lemma 4.2 *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. 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* GHMAC-1: $\{0,1\}^b \times B^+ \to \{0,1\}^c$ *be defined by (23) above. Let $\Phi = \{\Delta_{\mathsf{opad}}, \Delta_{\mathsf{ipad}}\}$. Let A be a prf-adversary against* GHMAC-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 and has time-complexity at most t.* ▮

The proof is simple and is omitted. Combining this with Theorem 3.3 yields the result that GHMAC-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 4.2. We remark that Φ is a small set of simple functions, which is important because it is shown in [6] 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.

5 Weakening the PRF assumption on h

Having established that NMAC and HMAC are PRFs, we have automatically established that they are secure MACs [5, 7]. However, it is possible that one could establish the security of these constructs merely as MACs (and not as PRFs) under an assumption on h that is weaker than the PRF one we have used so far. This is of interest given the still numerous usages of HMAC as a MAC (rather than as a PRF).

We now present a result that does this. It is in the style of [2] in that it makes not only an assumption about h but also one about its iteration h^* (our previous result made an assumption only about h) and can be viewed as attempting to formalize the intuition given in [2, Remark 4.4]. The assumption on h is that it is a privacy-preserving MAC, which, as we will show, is stronger than a MAC but weaker than a PRF. The assumption on h^* is that it is cAU, which is considerably weaker than weak collision resistance. In particular, the benefit of this result is that it does apply even in the case of hash functions like MD5 and SHA-1 whose weak collision resistance has been compromised or is in jeopardy. Let us begin by recalling the definition of forgery for MACs and then providing the (new) definition of privacy-preservation for a MAC.

5.1 Privacy preserving MACs

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

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

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 the adversary has no output, and we allow multiple verification queries as opposed to a single one as in [5]. (Important reasons to do so are pointed out in [7].)

PRIVACY-PRESERVING MACS. We define privacy for MACs by adapting the notion of left-or-right indistinguishability of encryption [4] to functions that are deterministic. 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
$K \stackrel{\$}{\leftarrow} \{0,1\}^c$	$K \stackrel{\$}{\leftarrow} \{0,1\}^c$
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, in which case it becomes non-trivial.

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 [5, 7] 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 5.1 *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.

5.2 Results

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

Lemma 5.2 *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 $K_{\mathrm{out}} \in \{0,1\}^c, K_{\mathrm{in}} \in \{0,1\}^k$ 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 $q = q_{\mathrm{mac}} + q_{\mathrm{vf}}$ and assume $2 \leq q < 2^b$. 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

$$\mathbf{Adv}_{hF}^{\mathrm{mac}}(A_{hF}) \leq \mathbf{Adv}_h^{\mathrm{mac}}(A_1) + \mathbf{Adv}_h^{\mathrm{ind}}(A_2) + \binom{q}{2} \cdot \mathbf{Adv}_F^{\mathrm{au}}(A_F) . \tag{24}$$

Furthermore, A_1 makes at most $q_{\rm mac}$ mac queries and at most $q_{\rm vf}$ 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 Section 5.3. 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 5.3 *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 and let $\mathsf{fpad} \in \{0,1\}^{b-c}$ be a fixed padding string. Let $\mathsf{GNMAC}\colon \{0,1\}^{2c} \times B^+ \to \{0,1\}^c$ be defined by*

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

for all $K_{\rm out}, K_{\rm in} \in \{0,1\}^c$ and $M \in B^+$. Let A_{GNMAC} be a mac-adversary against GNMAC that makes at most $q_{\rm mac}$ mac queries and at most $q_{\rm vf}$ verification queries, with the messages in each of these queries being of at most m blocks. Suppose A_{hF} has time-complexity at most t. Let $q = q_{\rm mac} + q_{\rm vf}$ and assume $2 \leq q < 2^b$. 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

$$\mathbf{Adv}^{\rm mac}_{\mathsf{GNMAC}}(A_{\mathsf{GNMAC}}) \;\leq\; \mathbf{Adv}^{\rm mac}_{h}(A_1) + \mathbf{Adv}^{\rm ind}_{h}(A_2) + \binom{q}{2} \cdot \mathbf{Adv}^{\rm au}_{h^*}(A^*)\ . \tag{25}$$

Furthermore, A_1 makes at most $q_{\rm mac}$ mac queries and at most $q_{\rm vf}$ 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 5.2 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. (But we do not know of such an application at this time.)

The procedure used in Section 4 to lift the NMAC results of Section 3 to HMAC applies also to lift the results of the current section to HMAC. Specifically, if h is a PP-MAC, h^* is cAU and $\overline{h}$ is a PRF then GHMAC is a (privacy-preserving) MAC. Also if h is a PP-MAC, h^* is cAU and $\overline{h}$ is a PRF under Φ-restricted related-key attacks, with Φ as in Lemma 4.2, then GHMAC-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 4.2 and 4.1) it need only hold with respect to adversaries that make very few queries.

5.3 Proof of Lemma 5.2

A mac-adversary against h gets a mac oracle $h(K_{\rm out}, \cdot)$ and corresponding verification oracle $\mathrm{VF}_h(K_{\rm out}, \cdot, \cdot)$. By itself picking key $K_{\rm in}$ and invoking these oracles, it can easily simulate the mac oracle $h(K_{\rm out}, F(K_{\rm in}, \cdot))$ and verification oracle $\mathrm{VF}_h(K_{\rm out}, F(K_{\rm in}, \cdot), \cdot)$ required by a mac-adversary against hF. This leads to the following natural construction of A_1:

Adversary $A_1^{h(K_{\rm out},\cdot),\mathrm{VF}_h(K_{\rm out},\cdot,\cdot)}$

Game $G1$

$K_{\text{out}} \xleftarrow{\$} \{0,1\}^c$; $K_{\text{in}} \xleftarrow{\$} \{0,1\}^k$; $i \leftarrow 0$

On mac query M or **verification query** M, T:

$i \leftarrow i+1$; $M_i \leftarrow M$; $y_i \leftarrow F(K_{\text{in}}, M)$
If $\exists\, j < i \,:\, y_i = y_j$ and $M_i \neq M_j$ then $\mathsf{bad} \leftarrow \mathsf{true}$
If $\exists\, j < i \,:\, M_i = M_j$ then $T_i \leftarrow T_j$
Else $T_i \leftarrow h(K_{\text{out}}, y_i)$
If mac query M then reply T_i
If verification query M, T then
If $T_i = T$ then reply 1 else reply 0

Game $G2$

$K_{\text{out}} \xleftarrow{\$} \{0,1\}^c$; $K_{\text{in}} \xleftarrow{\$} \{0,1\}^k$; $i \leftarrow 0$

On mac query M or **verification query** M, T:

$i \leftarrow i+1$; $M_i \leftarrow M$; $y_i \leftarrow F(K_{\text{in}}, M)$
If $\exists\, j < i \,:\, y_i = y_j$ and $M_i \neq M_j$ then $\mathsf{bad} \leftarrow \mathsf{true}$
If $\exists\, j < i \,:\, M_i = M_j$ then $T_i \leftarrow T_j$
Else $T_i \leftarrow h(K_{\text{out}}, \langle i \rangle_b)$
If mac query M then reply T_i
If verification query M, T then
If $T_i = T$ then reply 1 else reply 0

Figure 6: Games for the proof of Lemma 5.2.

$K_{\text{in}} \xleftarrow{\$} \{0,1\}^k$; $i \leftarrow 0$
Run A_{hF}, replying to its oracle queries as follows:
On mac query M **or verification query** M, T:
$i \leftarrow i+1$; $M_i \leftarrow M$; $y_i \leftarrow F(K_{\text{in}}, M)$
If mac query M then reply $h(K_{\text{out}}, y_i)$ to A_{hF}
If verification query M, T then reply $\mathrm{VF}_h(K_{\text{out}}, y_i, T)$ to A_{hF}

Consider the experiment defining the mac-advantage of A_1. Namely, choose $K_{\text{out}} \xleftarrow{\$} \{0,1\}^c$ and run A_1 with oracles $h(K_{\text{out}}, \cdot)$ and $\mathrm{VF}_h(K_{\text{out}}, \cdot, \cdot)$. Let Coll (for "collision") be the event that there exist j, l such that $y_j = y_l$ but $M_j \neq M_l$. Then

$$\begin{aligned} \mathbf{Adv}_h^{\mathrm{mac}}(A_1) &= \Pr\left[\, A_1 \text{ forges} \,\right] \\ &\geq \Pr\left[\, A_{hF} \text{ forges} \wedge \overline{\mathrm{Coll}} \,\right] \\ &\geq \Pr\left[\, A_{hF} \text{ forges} \,\right] - \Pr\left[\, \mathrm{Coll} \,\right] \\ &= \mathbf{Adv}_{hF}^{\mathrm{mac}}(A_{hF}) - \Pr\left[\, \mathrm{Coll} \,\right] . \end{aligned} \tag{26}$$

The rest of the proof is devoted to upper bounding $\Pr[\mathrm{Coll}]$. Consider the games $G1, G2$ of Figure 6, where we denote by $\langle i \rangle_b$ the representation of integer i as a b-bit string. (The assumption $q < 2^b$ made in the lemma statement means that we can always represent i this way in $G2$.) These games differ only in the manner in which tag T_i is computed. In $G1$, it is equal to $hF(K_{\text{out}} \| K_{\text{in}}, M_i)$.

Adversary $A_2^{g(\cdot,\cdot)}$

$K_{\mathrm{in}} \xleftarrow{\$} \{0,1\}^k$; $i \leftarrow 0$

Run A_{hF}, replying to its oracle queries as follows:

On mac query M or **verification query** M, T:

$i \leftarrow i+1$; $M_i \leftarrow M$; $y_i \leftarrow F(K_{\mathrm{in}}, M)$

If $\exists\, j < i \,:\, y_i = y_j$ and $M_i \neq M_j$ then return 1

If $\exists\, j < i \,:\, M_i = M_j$ then $T_i \leftarrow T_j$

Else $T_i \leftarrow g(\langle i \rangle_b, y_i)$

If mac query M then reply T_i to A_{hF}

If verification query M, T then

If $T_i = T$ then reply 1 to A_{hF} else reply 0 to A_{hF}

Return 0

Adversary A'_F

$K_{\mathrm{out}} \xleftarrow{\$} \{0,1\}^c$; $i \leftarrow 0$

Run A_{hF}, replying to its oracle queries as follows:

On mac query M or **verification query** M, T:

$i \leftarrow i+1$; $M_i \leftarrow M$

If $\exists\, j < i \,:\, M_i = M_j$ then $T_i \leftarrow T_j$

Else $T_i \leftarrow h(K_{\mathrm{out}}, \langle i \rangle_b)$

If mac query M then reply T_i to A_{hF}

If verification query M, T then

If $T_i = T$ then reply 1 to A_{hF} else reply 0 to A_{hF}

Pick i, j at random subject to $1 \leq i < j \leq q$

Return M_i, M_j

Figure 7: Ind-adversary A_2 against h, taking an oracle g that on input a pair of b-bit strings returns a c-bit string, and au-adversary A'_F against F, that outputs a pair of strings in D.

In $G2$, however, it is the result of applying $h(K_{\mathrm{out}}, \cdot)$ to the current value of the counter i, and as such does not depend on K_{in}. Now note that

$$\begin{aligned} \Pr[\,\mathrm{Coll}\,] &= \Pr\left[A_{hF}^{G1} \text{ sets } \mathsf{bad}\right] \\ &= \left(\Pr\left[A_{hF}^{G1} \text{ sets } \mathsf{bad}\right] - \Pr\left[A_{hF}^{G2} \text{ sets } \mathsf{bad}\right]\right) + \Pr\left[A_{hF}^{G2} \text{ sets } \mathsf{bad}\right]. \end{aligned} \tag{27}$$

Now consider the adversaries A_2, A'_F described in Figure 7. We claim that

$$\Pr\left[A_{hF}^{G1} \text{ sets } \mathsf{bad}\right] - \Pr\left[A_{hF}^{G2} \text{ sets } \mathsf{bad}\right] \leq \mathbf{Adv}_h^{\mathrm{ind}}(A_2) \tag{28}$$

$$\Pr\left[A_{hF}^{G2} \text{ sets } \mathsf{bad}\right] \leq \binom{q}{2} \cdot \mathbf{Adv}_F^{\mathrm{au}}(A'_F)\,. \tag{29}$$

Adversary A'_F, however, has time-complexity t (and outputs messages of at most n bits). A standard "coin-fixing" argument will be used to derive from A'_F an au-adversary A_F that has time-complexity $O(T_F(n))$ (and also outputs messages of n bits) such that

$$\mathbf{Adv}_F^{\mathrm{au}}(A'_F) \leq \mathbf{Adv}_F^{\mathrm{au}}(A_F)\,. \tag{30}$$

Combining (30), (29), (28), (27) and (26) yields (24) and completes the proof of Lemma 5.2. It remains to prove (28), (29) and (30). We begin with the first of these.

Recall that an ind-adversary against h is given an oracle that takes as input a pair of b-bit strings x_0, x_1. We are denoting this oracle by g. Now it is easy to see that

$$\begin{aligned} \Pr\left[A_2^{\mathrm{Right}} \Rightarrow 1 \right] &= \Pr\left[A_{hF}^{G1} \text{ sets } \mathsf{bad} \right] \\ \Pr\left[A_2^{\mathrm{Left}} \Rightarrow 1 \right] &= \Pr\left[A_{hF}^{G2} \text{ sets } \mathsf{bad} \right] , \end{aligned}$$

which implies (28). However, there is one important thing we still need to verify, namely that A_2 is legitimate. So consider the sequence $(x_1, y_1), (x_2, y_2), \ldots$ of oracle queries it makes. The left halves $x_1, x_2, \ldots$ are values of the counter i in different loop iterations and are thus strictly increasing (although not necessarily successive) and in particular different. On the other hand the right half values $y_1, y_2, \ldots$ are distinct because as soon as $y_i = y_j$ for some $j < i$, adversary A_2 halts (and returns 1), never making an oracle query whose right half is y_i.

Next we turn to A'_F. In order for the messages M_i, M_j it returns to always be defined, we assume wlog that A_{hF} always makes exactly, rather than at most, q_{mac} mac queries and exactly, rather than at most, q_{vf} verification queries. Intuitively, (29) is true because i, j are chosen at random after the execution of A_{hF} is complete, so A_{hF} has no information about them. This can be made rigorous just as in the proof of Lemma 3.2, and the details follow. Consider the experiment defining the au-advantage of A'_F. In this experiment, consider the following events defined for $1 \le \alpha < \beta \le q$:

$$\begin{aligned} C_{\alpha,\beta} &: F(K_{\mathrm{in}}, M_\alpha) = F(K_{\mathrm{in}}, M_\beta) \ \wedge \ M_\alpha \neq M_\beta \\ C &: \textstyle\bigvee_{1 \le \alpha < \beta \le q} C_{\alpha,\beta} \, . \end{aligned}$$

Notice that the events "$C_{\alpha,\beta} \wedge (i,j) = (\alpha, \beta)$" ($1 \le \alpha < \beta \le q$) are disjoint. (Even though the events $C_{\alpha,\beta}$ for $1 \le \alpha < \beta \le q$ are not.) Thus:

$$\begin{aligned} \mathbf{Adv}_F^{\mathrm{au}}(A'_F) &= \Pr\left[\textstyle\bigvee_{1 \le \alpha < \beta \le q} (C_{\alpha,\beta} \wedge (i,j) = (\alpha,\beta)) \right] \\ &= \sum_{1 \le \alpha < \beta \le q} \Pr\left[C_{\alpha,\beta} \wedge (i,j) = (\alpha,\beta) \right] . \end{aligned}$$

Since i, j are chosen at random after the execution of A_{hF} is complete, the events "$(i,j) = (\alpha, \beta)$" and $C_{\alpha,\beta}$ are independent. Thus the above equals

$$\begin{aligned} \sum_{1 \le \alpha < \beta \le q} \Pr\left[C_{\alpha,\beta} \right] \cdot \Pr\left[(i,j) = (\alpha,\beta) \right] &= \sum_{1 \le \alpha < \beta \le q} \Pr\left[C_{\alpha,\beta} \right] \cdot \frac{1}{\binom{q}{2}} \\ &= \frac{1}{\binom{q}{2}} \cdot \sum_{1 \le \alpha < \beta \le q} \Pr\left[C_{\alpha,\beta} \right] \\ &\ge \frac{1}{\binom{q}{2}} \cdot \Pr\left[C \right] . \end{aligned}$$

The proof of (29) is concluded by noting that $\Pr[C]$ equals $\Pr[G2 \text{ sets } \mathsf{bad}]$.

Finally, au-adversary A_F of time-complexity $O(T_F(n))$ satisfying (30) can be obtained from A_F just as in the proof of Lemma 3.2.

Acknowledgments

Thanks to Ran Canetti, Hugo Krawczyk, Phillip Rogaway, Victor Shoup and Paul Van Oorschot for comments and references.

References

[1] American National Standards Institution. ANSI X9.71, Keyed hash message authentication code, 2000.

[2] 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.

[3] 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.)

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] M. Bellare, K. Pietrzak and P. Rogaway. Improved security analyses for CBC MACs. *Advances in Cryptology – CRYPTO '05*, Lecture Notes in Computer Science Vol. 3621 , V. Shoup ed., Springer-Verlag, 2005.

[9] M. Bellare and P. Rogaway. The game-playing technique. *Cryptology ePrint Archive*: Report 2004/331, 2004.

[10] 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.

[11] 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.

[12] J. Black and P. Rogaway. A block-cipher mode of operation for parallelizable message authentication. *Advances in Cryptology – EUROCRYPT '02*, Lecture Notes in Computer Science Vol. 2332, L. Knudsen ed., Springer-Verlag, 2002.

[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] T. Dierks and C. Allen. The TLS protocol. Internet RFC 2246, 1999.

[16] 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.

[17] 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.

[18] O. Goldreich, S. Goldwasser and S. Micali. How to construct random functions. *Journal of the ACM*, Vol. 33, No. 4, 1986, pp. 210–217.

[19] S. Goldwasser, S. Micali and R. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal of Computing*, Vol. 17, No. 2, 1988, pp. 281–308.

[20] D. Harkins and D. Carrel. The Internet Key Exchange (IKE). Internet RFC 2409, 1998.

[21] S. Halevi and H. Krawczyk. MMH: Message authentication in software in the Gbit/sec rate. *Fast Software Encryption '97*, Lecture Notes in Computer Science Vol. 1267, E. Biham ed., Springer-Verlag, 1997.

[22] H. Krawczyk, M. Bellare and R. Canetti. HMAC: Keyed-hashing for message authentication. 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. (Based on an unpublished paper from 1979 and the author's Ph. D thesis, Stanford, 1979).

[24] D. M'Raihi, M. Bellare, F. Hoornaert, D. Naccache, O. Ranen. HOTP: An HMAC-based one time password algorithm. Internet Draft, 2005. `http://www.ietf.org/internet-drafts/draft-mraihi-oath-hmac-otp-04.txt`.

[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. Recommendation on key establishment schemes. DRAFT SP 800-56, July 2005.

[27] National Institute of Standards and Technology. Secure hash standard. FIPS PUB 180-2, August 2000.

[28] E. Petrank and C. Rackoff. CBC MAC for real-time data sources. *Journal of Cryptology*, Vol. 13, No. 3, 2000, pp. 315–338.

[29] 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.)

[30] R. Rivest. The MD5 message-digest algorithm. IETF RFC 1321, April 1992.

[31] V. Shoup. Sequences of games: A tool for taming complexity in security proofs. *Cryptology ePrint Archive*: Report 2004/332, 2004.

[32] D. Stinson. Universal hashing and authentication codes. *Designs, Codes and Cryptography*, Vol. 4, 1994, 369–380.

[33] 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.

[34] 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.

[35] 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.

A Attacking weak collision-resistance

Recall that H represents the cryptographic hash function (eg. MD5, SHA-1) while H^* is the extended hash function, which is the hash function with the initial vector made explicit as an (additional) first input. Let us use the term *general collision-finding attack* to refer to an attack that finds collisions in $H^*(\mathrm{IV}, \cdot)$ for an arbitrary but given IV. As we discussed in Section 1, it was noted in [2] that any general collision-finding attack can be used to compromise the weak collision resistance of H. (And since the known collision-finding attacks on MD5 and SHA-1 [34, 33] do extend to general ones, the weak collision resistance of these functions is no more than their collision-resistance.) Here we recall the argument that shows this. It is a simple extension attack, and works as follows.

To compromise weak collision resistance of H, an attacker given an oracle for $H^*(K, \cdot)$ under a hidden key K must output distinct M_1, M_2 such that $H^*(K, M_1) = H^*(K, M_2)$. Our attacker picks some string x and calls its oracle to obtain $\mathrm{IV} = H^*(K, x)$. Then it runs the given general collision-finding attack on input IV to obtain a collision X_1, X_2 for $H^*(\mathrm{IV}, \cdot)$. (That is, X_1, X_2 are distinct strings such that such that $H^*(\mathrm{IV}, X_1) = H^*(\mathrm{IV}, X_2)$.) Now let $M_1 = x \| \mathsf{pad}(|x|) \| X_1 \| \mathsf{pad}(|X_1|)$ and $M_2 = x \| \mathsf{pad}(|x|) \| X_2 \| \mathsf{pad}(|X_2|)$. (Here $\mathsf{pad}(n)$ is a padding string that when appended to a string of length n results in a string whose length is a positive multiple of b bits where b is the block-length of the underlying compression function. The function pad is part of the description of the cryptographic hash function.) Then it follows that $H^*(K, M_1) = H^*(K, M_2)$.