Another Look at Security Theorems for 1-Key Nested MACs

Neal Koblitz and Alfred Menezes

Abstract We prove a security theorem without collision resistance for a class of 1-key hash function-based MAC schemes that includes HMAC and Envelope MAC. The proof has some advantages over earlier proofs: it is in the uniform model, it uses a weaker related-key assumption, and it covers a broad class of MACs in a single theorem. However, we also explain why our theorem is of doubtful value in assessing the real-world security of these MAC schemes. In addition, we prove a theorem assuming collision resistance. From these two theorems, we conclude that from a provable security standpoint, there is little reason to prefer HMAC to Envelope MAC or similar schemes.

1 Introduction

The purpose of our "Another Look" series of papers [14] is to examine the way the "provable security" paradigm is used in the cryptographic literature. In particular, we hope to foster a less credulous attitude toward some of the claims that are frequently made about "provable" security.

Starting in the early days of "practice-oriented provable security"—a term coined by Bellare and Rogaway [1, 4]—there has been an unfortunate tendency to exaggerate both the security guarantees that are proved and the efficiency advantages of the provably secure protocols. For example, in [5] the authors used the word "optimal" to advertise the OAEP version of RSA encryption (OAEP = "optimal asymmetric encryption padding"). Shortly after Victor Shoup [27] discovered that the security proof in [5] was fallacious, the claim of optimal efficiency was also reexamined. It now seems that Boneh–Rabin encryption [9] comes closer than OAEP to being both provably secure (in a limited sense) and optimally efficient; see Sect. 4 of [15].

N. Koblitz
Department of Mathematics, University of Washington, Box 354350, Seattle, WA 98195, USA
e-mail: koblitz@uw.edu

A. Menezes (✉)
Department of Combinatorics & Optimization, University of Waterloo, Waterloo, ON, Canada N2L 3G1
e-mail: ajmeneze@uwaterloo.ca

Ç.K. Koç (ed.), *Open Problems in Mathematics and Computational Science*,
DOI 10.1007/978-3-319-10683-0_4

Excessive enthusiasm in marketing protocol designs can also be seen in certain statements about hash-based key agreement and message authentication. According to a letter to the *AMS Notices* from Hugo Krawczyk [20], the designer of the HMQV key agreement protocol, "the HMQV work represents a prime example of the success of theoretical cryptography, not only in laying rigorous mathematical foundations for cryptography at large, but also in its ability to guide us in the design of truly practical solutions to real-world problems." Similarly, speaking of his hash-based message authentication code HMAC in an invited talk at Asiacrypt 2010 on "Cryptography: from theory to practice," Krawczyk proclaimed that with HMAC "balance [between engineering and theory was] regained and the rest is history."

One of the conclusions of the present chapter is that Krawczyk's claim of a unique benefit provided by HMAC cannot be justified by provable security considerations. Rather, very similar security results can be proved for a broad class of message authentication codes, including some (such as "Envelope MAC") that arguably are a little more efficient than HMAC.

Another theme that recurs in several of our papers, including the present one, is that the security definitions that are at the heart of any "proof" of security are often open to debate and are far from definitive (see [16] for more discussion of this). In [18] we found that even such a fundamental concept of computer science as the distinction between a uniform and nonuniform algorithm is frequently dealt with in a confusing and inconsistent manner in the cryptographic literature. In the present chapter, we argue that in the MAC setting, two of the basic definitions used by earlier authors—that of a pseudorandom function and that of security against related-key attacks—need to be replaced by more suitable versions.

As we have written on many occasions, starting with [15], we have no objection to formal arguments in cryptography provided that their significance is properly interpreted and they are not misnamed "proofs of security." Indeed, reductionist security arguments for hash functions, message authentication codes, and other symmetric and asymmetric cryptographic protocols can provide a type of baseline guarantee of a certain limited security feature. We show that a broad class of 1-key nested MACs have such a property. But the choice of which MAC in the class one wants to use cannot be made using reductionist security arguments but rather should be based on an ad hoc analysis of efficiency and security in the settings in which it will be deployed.

* * *

A common method of constructing a message authentication code (MAC) is the "nested" construction (NMAC). One first applies a keyed iterated hash function $h(K_1, M)$ (constructed from a compression function f) to the message M, and then one puts this hash value into a second keyed function $\tilde{f}(K_2, h(K_1, M))$ (where $\tilde{f}$ is also a compression function). For efficiency and ease of implementation, one usually wants the MAC to depend on a single key K, and so one sets $K_1 = K_2 = K$ or, more generally, $K_1 = g_1(K)$, $K_2 = g_2(K)$ for some functions g_1, g_2. Our main purpose is to prove a new security theorem without collision resistance that covers arbitrary constructions of this type. The theorem says, roughly speaking, that the MAC is a pseudorandom function (prf) provided that both $\tilde{f}$ and f

are pseudorandom functions and the functions $f, \tilde{f}, g_1, g_2$ satisfy a certain rather weak related-key assumption. This theorem is a generalized 1-key version of our Theorem 10.1 in [17].

The two most important examples of this type of MAC are the "hash-based message authentication code" (HMAC) [6] (standardized in [7, 21]) and Envelope MAC (also called "Sandwich MAC"; see [29] for a recent version). In these cases there are earlier security proofs without collision resistance in [2, 29], but unfortunately those proofs are not valid in the uniform model of complexity.[1] In other words, they use unconstructible adversaries and so have to assume that the cryptographic primitives withstand attack even by unconstructible adversaries. For this reason, as we explained in [17] (see also [8]), they do not give useful concrete bounds on the resources a prf-adversary would need in order to defeat the MAC. In contrast, our theorem is proved in the uniform model; this means that it needs much milder assumptions.

One of the five finalists in the NIST SHA-3 competition used Envelope MAC. The designers of the "Grøstl" construction wrote (Sect. 6.1 of [11]):

> We propose this envelope construction as a dedicated MAC mode using Grøstl. This construction has been proved to be a secure MAC under similar assumptions as HMAC.

Here the designers were referring to the proof in [29], but they were apparently unaware that Yasuda's proof is not valid in the uniform model and for that reason gives much weaker guarantees than one would expect. As we commented in [18], one of the drawbacks of results obtained in the nonuniform model is the possibility that they will be used by other authors who are unaware of the extremely limited nature of such results from a practice-oriented standpoint. In any case, in the present chapter we remove this gap in the security argument in [11] by supplying a uniform proof.

There is a second respect in which our theorem makes a milder assumption than earlier theorems of this type: our related-key assumption is weaker than the one defined in [2, 3]. This not only gives us a stronger theorem but also enables us to unify HMAC and Envelope MAC in a single theory.

Despite these advantages over earlier security theorems, the sad fact is that our main theorem by itself provides very little assurance about the real-world security of these MAC schemes. In Sect. 4 we recall some of the reasons for this.

In Sect. 5 we prove a second theorem, this time assuming collision resistance, that carries over the main result of [6] to 1-key nested MACs. Our two theorems together show that from the standpoint of security reductions, there is little difference between HMAC, Envelope MAC, and other similar constructions. We conclude that security theorems are not of much use in deciding which of the competing schemes—HMAC, Envelope MAC, or some other variant—has better security in practice.

[1]Fischlin [10] has a uniform proof of a security theorem for HMAC without collision resistance, but its usefulness is questionable because of the extremely large tightness gap in his result.

2 Statement of the Main Theorem

Let $f : \{0,1\}^c \times \{0,1\}^b \longrightarrow \{0,1\}^c$ and $\tilde{f} : \{0,1\}^c \times \{0,1\}^c \longrightarrow \{0,1\}^c$ be two compression functions. Here $b \geq 2c$ (typically $b = 512$ and $c = 128$ or 160), so that f compresses by a factor of at least 3, whereas $\tilde{f}$ compresses by a factor of 2. Let $g_i : \{0,1\}^c \longrightarrow \{0,1\}^c$, $i = 1, 2$. We suppose that all of these functions are publicly and efficiently computable.

By a (t, q)-adversary, we mean an adversary that makes $\leq q$ queries and has running time $\leq t$. Recall that f is said to be an (ϵ, t, q)-secure pseudorandom function (prf) if no (t, q)-adversary can distinguish between f with a hidden key and a random function with advantage $\geq \epsilon$. We say that f is strongly (ϵ, t, q) secure (see [17]) if such an adversary before any query is permitted to "reset" the oracle, by which we mean that in response to the adversary's request the oracle chooses either a new random key (if it is $f(K, .)$) or a new random function (if it is a random function $r'(.)$).

We now define the "related-key assumption" that we shall use in our main theorem.

Definition 1 In the above setting, we say that $(f, \tilde{f})$ is (ϵ, t, q)-secure against (g_1, g_2)-related-key attacks if no (t, q)-adversary has an advantage greater than or equal to ϵ in the following interaction with the oracle O_{rka}. First, the oracle chooses a random bit; if it is 0, the oracle chooses two random keys $K_1, K_2 \in \{0,1\}^c$; if it is 1, the oracle chooses one random key $K \in \{0,1\}^c$ and sets $K_1 = g_1(K)$, $K_2 = g_2(K)$. Each query of the adversary is a message M in either $\{0,1\}^b$ or $\{0,1\}^c$, to which the oracle responds with either $f(K_1, M)$ or $\tilde{f}(K_2, M)$, respectively. At the end, the adversary guesses the oracle's random bit.

We recall that in this situation the advantage of the adversary is defined as

$$\text{Prob}(\text{adversary guesses } 1 \mid \text{oracle chose } 1) - \text{Prob}(\text{adversary guesses } 1 \mid \text{oracle chose } 0),$$

where Prob(A|B) denotes the conditional probability of event A given event B.

This setting is general enough to include two of the best-known MAC constructions (see Fig. 1):

1. For HMAC, let IV be a fixed (and publicly known) initialization vector, and let ipad and opad be two fixed elements of $\{0,1\}^b$ (also publicly known). We let a superscript 0 on a bitstring in $\{0,1\}^c$ indicate that we are appending $b - c$ zero bits to it. We set $\tilde{f}(K, M) = f(K, M^0)$, $g_1(K) = f(\text{IV}, K^0 \oplus \text{ipad})$, $g_2(K) = f(\text{IV}, K^0 \oplus \text{opad})$.
2. For Envelope MAC, let IV be a fixed (and publicly known) initialization vector; let $\tilde{f}(K, M) = f(M, K^0)$, $g_1(K) = f(\text{IV}, K^0)$, and $g_2(K) = K$.

Remark 1 The above related-key assumption is weaker than the related-key assumption in [2, 3]. In that assumption, the oracle is required to simply give the adversary the two keys: K_1, K_2. In that case the adversary can of course compute

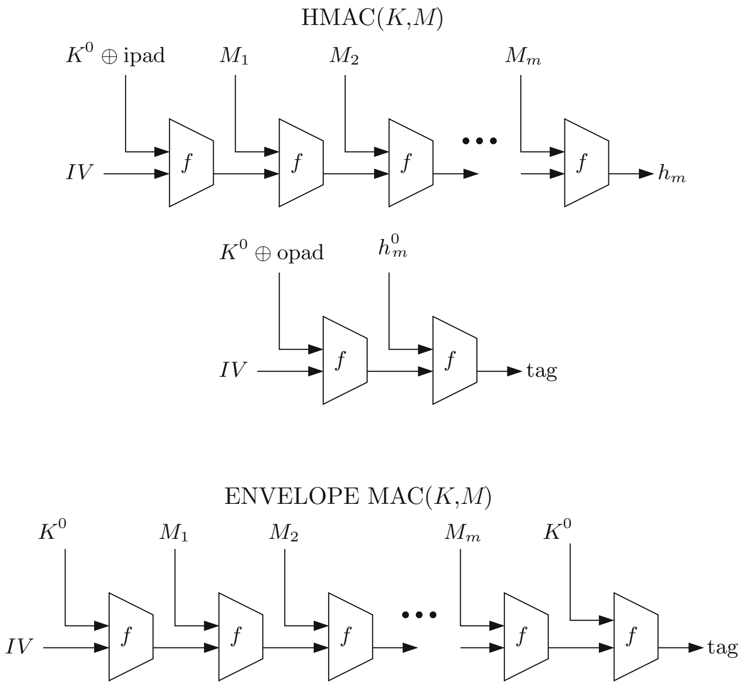

Fig. 1 HMAC and Envelope MAC

any number of desired values $f(K_1, M)$ or $\tilde{f}(K_2, M)$, limited only by the running time bound; in other words, the rka-adversary in our assumption is less powerful (because it has less information) than the rka-adversary in [2, 3]. Moreover, with the rka-assumption in [2, 3], we wouldn't have been able to include Envelope MAC in our theorem, because when $g_2(K) = K$, the adversary, if given K_1 and K_2, can trivially determine whether or not $K_1 = g_1(K_2)$.

In the above setting let $h : \{0,1\}^c \times (\{0,1\}^b)^* \longrightarrow \{0,1\}^c$ denote the iterated hash function that, given a key $K \in \{0,1\}^c$ and a message $M = (M_1, M_2, \ldots, M_m)$, $M_i \in \{0,1\}^b$, successively computes $h_1 = f(K, M_1)$, $h_{i+1} = f(h_i, M_{i+1})$, $i = 1, 2, \ldots, m-1$ and sets $h(K, M) = h_m$. We define the message authentication code $\mathrm{MAC}_{f,\tilde{f},g_1,g_2}$ as follows:

$$\mathrm{MAC}_{f,\tilde{f},g_1,g_2}(K, M) = \tilde{f}(g_2(K), h(g_1(K), M)).$$

Notice that when $g_1(K) = f(\mathrm{IV}, K^0 \oplus \mathrm{ipad})$, and $g_2(K) = f(\mathrm{IV}, K^0 \oplus \mathrm{opad})$ this definition agrees with that of HMAC, and when $g_1(K) = f(\mathrm{IV}, K^0)$ and $g_2(K) = K$, it agrees with that of Envelope MAC (see Fig. 1).

By a (t, q, n)-adversary, we mean an adversary that makes $\leq q$ queries of block length $\leq n$ and has running time $\leq t$. We say that $\mathrm{MAC}_{f,\tilde{f},g_1,g_2}$ is an

(ϵ, t, q, n)-secure pseudorandom function if no (t, q, n)-adversary can distinguish between $\mathrm{MAC}_{f,\tilde{f},g_1,g_2}$ with hidden key and a random function with advantage $\geq \epsilon$.

Theorem 1 *Suppose that f is a strongly (ϵ_1, t, q)-secure pseudorandom function, $\tilde{f}$ is an (ϵ_2, t, q)-secure pseudorandom function, and $(f, \tilde{f})$ is $(\epsilon_3, t, 2q)$-secure against (g_1, g_2)-related-key attacks. Then $MAC_{f,\tilde{f},g_1,g_2}$ is a $(2n(\epsilon_1 + \binom{q}{2}2^{-c}) + \epsilon_2 + 2\epsilon_3, t - (qnT + Cq\log q), q, n)$-secure pseudorandom function. Here C is an absolute constant, and T denotes the time for one evaluation of f or $\tilde{f}$.*

Remark 2 In the statement of the theorem, the expression $2n(\epsilon_1 + \binom{q}{2}2^{-c}) + \epsilon_2 + 2\epsilon_3$ can be replaced by $2n\epsilon_1 + \epsilon_2 + 2\epsilon_3$. The reason is that, as explained in Remark 10.2 of [17], the generic key-guessing attack on the strong pseudorandomness property has advantage roughly $(qt/T)2^{-c}$; since we need $t > qnT$ for the theorem to have content, it follows that $\epsilon_1 \gg \binom{q}{2}2^{-c}$.

Before proving Theorem 1, we give an informal summary of the argument. The first step is to show that a prf-adversary A_{MAC} of $\mathrm{MAC}_{f,\tilde{f},g_1,g_2}$ is also a prf-adversary—with almost the same advantage—of the MAC obtained by replacing the (g_1, g_2)-related keys by independent random keys. Here "almost" means that we can construct a related-key attack A_{rka} on $(f, \tilde{f})$ whose advantage is equal to half the difference between the advantage of A_{MAC} when the keys are $(g_1(K), g_2(K))$ and its advantage when the keys are independent. This step reduces the problem to the case when there are two independent keys, at which point we can essentially follow the proof for NMAC in [17]. Namely, we show that a prf-adversary for the MAC succeeds only when either the prf property of the outer shell $\tilde{f}(K_2, .)$ is attacked (we call its adversary $A_{\tilde{f}}$) or else a collision is produced in the iterated hash function that's inside this shell. In the latter case we use the collision to construct a prf-adversary of f. Since there are two possible types of collisions that can occur and up to n iterations of the hash function, this leads to roughly $2n$ f adversaries. This intuitively explains why $2n\epsilon_1 + \epsilon_2 + 2\epsilon_3$ appears in the conclusion of the theorem. The term $2n\binom{q}{2}2^{-c}$ arises because of the possibility of random collisions between c-bit strings.

We shall give the actual proof in the next section. The above plausibility argument shows that the basic ideas in the proof are simple. However, the organization is a little intricate because of the need to proceed carefully with the reduction using all of the adversaries. We see no way to come up with a more concise self-contained proof, and we apologize to the reader for that.

3 Proof of the Main Theorem

Proof We will prove the following equivalent statement: if f is a strongly $((\frac{\epsilon-\epsilon_2-2\epsilon_3}{2n} - \binom{q}{2}2^{-c}), t + (qnT + Cq\log q), q)$-secure pseudorandom function, $\tilde{f}$ is an $(\epsilon_2, t + (qnT + Cq\log q), q)$-secure pseudorandom function, and $(f, \tilde{f})$ is $(\epsilon_3, t + (qnT + Cq\log q), 2q)$-secure against (g_1, g_2)-related-key attacks, then

$\mathrm{MAC}_{f,\tilde{f},g_1,g_2}$ is an (ϵ, t, q, n)-secure pseudorandom function. The proof starts by supposing that we have a (t, q, n)-adversary A_{MAC} that has advantage $\geq \epsilon$ in the pseudorandomness test for $\mathrm{MAC}_{f,\tilde{f},g_1,g_2}$, and then it proceeds to construct a $(t + (qnT + Cq \log q), 2q)$-adversary A_{rka} of the related-key property, a $(t + (qnT + Cq \log q), q)$-adversary $A_{\tilde{f}}$ of the pseudorandom property of $\tilde{f}$, and a $(t + (qnT + Cq \log q), q)$-adversary A_f of the pseudorandom property of f such that at least one of the following holds:

(i) A_{rka} has advantage $\geq \epsilon_3$ against the (g_1, g_2)-related key property of $(f, \tilde{f})$.
(ii) $A_{\tilde{f}}$ has advantage $\geq \epsilon_2$ in the pseudorandomness test for $\tilde{f}$.
(iii) A_f has advantage $\geq (\epsilon - \epsilon_2 - 2\epsilon_3)/(2n) - \binom{q}{2}2^{-c}$ in the strong pseudorandomness test for f.

Note that if any of these three conditions holds, we have a contradiction that proves the theorem.

For the ith message query M^i, we use the notation M^i_ℓ to denote its ℓth block, we let $M^{i,[m]} = (M^i_1, \ldots, M^i_m)$ be the truncation after the mth block, and we set $M^{i,(m)} = (M^i_m, M^i_{m+1}, \ldots)$, that is, $M^{i,(m)}$ is the message with the first $m-1$ blocks deleted. We say that a message is "non empty" if its block-length is at least 1.

Let h be the corresponding iterated function, and let $\tilde{f}h$ be the MAC that for a key $(K_1, K_2) \in \{0,1\}^c \times \{0,1\}^c$ is defined as follows: $\tilde{f}h(K_1, K_2, M) = \tilde{f}(K_2, h(K_1, M))$, where $M = (M_1, \ldots, M_m)$ is an m-block message, $m \leq n$. Note that $\mathrm{MAC}_{f,\tilde{f},g_1,g_2}(K, M) = \tilde{f}h(g_1(K), g_2(K), M)$. Let $r(M)$ denote a random function of messages, and let $r'(M_1)$ denote a random function of 1-block messages. In response to an input of suitable length, r' or r outputs a random c-bit string, subject only to the condition that if the same input is given a second time (in the same run of the algorithm), the output will be the same. In the test for pseudorandomness, the oracle is either a random function or the function being tested, as determined by a random bit (coin toss).

Now suppose that we have a (t, q, n)-adversary A_{MAC} that, interacting with its oracle O_{MAC}, has advantage $\geq \epsilon$ against the prf test for $\mathrm{MAC}_{f,\tilde{f},g_1,g_2}$. We use A_{MAC} to construct four adversaries $A_{\tilde{f}h}$, A_{rka}, $A_{\tilde{f}}$, and A_f. The last three are the adversaries in the above conditions (i)–(iii); the (t, q, n)-adversary $A_{\tilde{f}h}$ attacks the pseudorandomness property of $\tilde{f}h$. Each adversary makes at most the same number of queries as A_{MAC} (except that the related-key adversary can make up to $2q$ queries) and has a comparable running time. More precisely, the bound $t + (qnT + Cq \log q)$ on the running time of the adversaries A_{rka}, $A_{\tilde{f}}$, and A_f comes from the time required to run A_{MAC}, makes at most q computations of h values, and stores at most q values (coming from oracle responses and h computations) in lexicographical order and sorts them looking for collisions. (An adversary does not in all cases perform all of these steps; rather, this is an upper bound.)

The related-key adversary A_{rka} runs A_{MAC} and interacts with the related-key oracle O_{rka}, which chooses a random bit u. Recall that A_{rka}, after querying the oracle O_{rka} with at most $2q$ b-bit or c-bit messages, must guess whether the keys K_1 and

K_2 that O_{rka} is using are independent (i.e., $u = 0$) or are related by $K_i = g_i(K)$, $i = 1, 2$, for some K (i.e., $u = 1$).

The adversary A_{rka} randomly chooses a bit ℓ, and as A_{MAC} runs, A_{rka} responds to each query M^i as follows. If $\ell = 0$, its response to each query is a random c-bit string (except in the case when a query is repeated, in which case the response is also repeated). If $\ell = 1$, then it first queries O_{rka} with M_1^i and computes $H = h(O_{\text{rka}}(M_1^i), M^{i,(2)})$, where $O_{\text{rka}}(M_1^i)$ denotes the oracle's response. (If M^i is just a 1-block message, then H is set equal to $O_{\text{rka}}(M_1^i)$.) Now A_{rka} makes a second query to O_{rka}—this time the c-bit query H—and responds to A_{MAC}'s query with $O_{\text{rka}}(H)$.[2] At the end A_{rka} guesses that the random bit u chosen by O_{rka} is 1 if A_{MAC} guesses that the random bit ℓ chosen by A_{rka} (which is simulating an oracle) is 1; otherwise, it guesses that $u = 0$. (Note that A_{rka} guesses 0 if A_{MAC} stops or reaches time t without producing a guess; this could happen if A_{rka} is not properly simulating O_{MAC}, which would imply that $u = 0$.) Let δ denote the advantage of A_{rka}.

We also construct an adversary $A_{\tilde{f}h}$ that interacts with its oracle $O_{\tilde{f}h}$ and runs A_{MAC}. When A_{MAC} makes a query M^i, the adversary $A_{\tilde{f}h}$ queries $O_{\tilde{f}h}$ and sends A_{MAC} the response $O_{\tilde{f}h}(M^i)$. If A_{MAC} guesses that the oracle simulated by $A_{\tilde{f}h}$ is a random function, then $A_{\tilde{f}h}$ guesses that its oracle $O_{\tilde{f}h}$ is a random function; otherwise, $A_{\tilde{f}h}$ guesses that its oracle is $\tilde{f}h$ with hidden keys. In particular, note that if A_{MAC} stops or fails to produce a guess in time t—as may happen when $A_{\tilde{f}h}$ is not property simulating O_{MAC}—then $A_{\tilde{f}h}$ guesses that its oracle is $\tilde{f}h$ with hidden keys. (This makes sense, since if $O_{\tilde{f}h}$ were a random function, then the simulation of O_{MAC} would be correct.) Let γ denote the advantage of $A_{\tilde{f}h}$.

Returning to the description of A_{rka}, we see that there are two cases, depending on whether the random bit u of the oracle O_{rka} was (a) 1 (i.e., its keys are related) or (b) 0 (i.e., its keys are independent). In case (a) the interaction of A_{rka} with A_{MAC} precisely simulates O_{MAC}, and in case (b) it precisely simulates $O_{\tilde{f}h}$. (As we noted, in case (b) our original adversary A_{MAC} may stop or run for time t without producing a guess; in this case A_{rka} makes the guess 0.) Let

$$p_1 = \text{Prob}(A_{\text{MAC}} \text{ guesses } 1 \mid \ell = 1 \text{ and } u = 1).$$

$$p_2 = \text{Prob}(A_{\text{MAC}} \text{ guesses } 1 \mid \ell = 1 \text{ and } u = 0).$$

$$p_3 = \text{Prob}(A_{\text{MAC}} \text{ guesses } 1 \mid \ell = 0).$$

Note that when $\ell = 0$, there is no interaction with O_{rka}, and so the guess that A_{MAC} makes is independent of whether $u = 0$ or $u = 1$.

[2] The theorem's query bound for the related-key property is $2q$ because A_{rka} makes two queries for each query of A_{MAC}.

By assumption, A_{MAC} has advantage $\geq \epsilon$ in a prf test for $\text{MAC}_{f,\tilde{f},g_1,g_2}$; in other words, $p_1 - p_3 \geq \epsilon$. We also have $p_2 - p_3 = \gamma$. Subtracting gives $p_1 - p_2 \geq \epsilon - \gamma$. Next, the advantage δ of the related-key adversary A_{rka} is given by

$$\begin{aligned}
&\text{Prob}(A_{\text{rka}} \text{ guesses } 1 \mid u = 1) - \text{Prob}(A_{\text{rka}} \text{ guesses } 1 \mid u = 0)\\
&\quad = \text{Prob}(A_{\text{MAC}} \text{ guesses } 1 \mid u = 1) - \text{Prob}(A_{\text{MAC}} \text{ guesses } 1 \mid u = 0)\\
&\quad = \text{Prob}(A_{\text{MAC}} \text{ guesses } 1 \mid u = 1 \text{ and } \ell = 0) \cdot \text{Prob}(\ell = 0)\\
&\qquad + \text{Prob}(A_{\text{MAC}} \text{ guesses } 1 \mid u = 1 \text{ and } \ell = 1) \cdot \text{Prob}(\ell = 1)\\
&\qquad - \text{Prob}(A_{\text{MAC}} \text{ guesses } 1 \mid u = 0 \text{ and } \ell = 0) \cdot \text{Prob}(\ell = 0)\\
&\qquad - \text{Prob}(A_{\text{MAC}} \text{ guesses } 1 \mid u = 0 \text{ and } \ell = 1) \cdot \text{Prob}(\ell = 1)\\
&\quad = \frac{1}{2}(p_3 + p_1 - p_3 - p_2) = \frac{1}{2}(p_1 - p_2) \geq (\epsilon - \gamma)/2.
\end{aligned}$$

If condition (i) in the first paragraph of the proof does not hold, then $\delta < \epsilon_3$, in which case $\gamma > \epsilon - 2\epsilon_3$. For the remainder of the proof, we assume that the advantage of $A_{\tilde{f}h}$ satisfies this inequality, since otherwise (i) holds and we're done.

The rest of the proof closely follows the proof of Theorem 10.1 of [17]. We shall give the details rather than simply citing [17] because the present setting is slightly more general (with two pseudorandom compression functions f and $\tilde{f}$ rather than just one) and because there is some benefit in having a self-contained proof in one place.

We now construct an $\tilde{f}$-adversary $A_{\tilde{f}}$ and consider its advantage. As before, for any oracle O, we let $O(M)$ denote the response of O to the query M. The adversary $A_{\tilde{f}}$ is given an oracle $O_{\tilde{f}}$ and, using $A_{\tilde{f}h}$ as a subroutine, has to decide whether $O_{\tilde{f}}$ is $\tilde{f}(K_2, .)$ or a random function $r'(.)$ of 1-block messages. She chooses a random K_1 and presents the adversary $A_{\tilde{f}h}$ with an oracle that is either $\tilde{f}(K_2, h(K_1, .))$ or else a random function $r(.)$; that is, she simulates $O_{\tilde{f}h}$ (see below). In time $\leq t$ with $\leq q$ queries $A_{\tilde{f}h}$ is able with advantage $\gamma > \epsilon - 2\epsilon_3$ to guess whether $O_{\tilde{f}h}$ is $\tilde{f}h$ with hidden keys or a random function r. Here is how $A_{\tilde{f}}$ simulates $O_{\tilde{f}h}$: in response to a query M^i from $A_{\tilde{f}h}$, she computes $h(K_1, M^i)$, which she queries to $O_{\tilde{f}}$, and then gives $A_{\tilde{f}h}$ the value $O_{\tilde{f}}(h(K_1, M^i))$. Eventually (unless the simulation is imperfect, see below) $A_{\tilde{f}h}$ states whether it believes that its oracle $O_{\tilde{f}h}$ is $\tilde{f}h$ or r, at which point $A_{\tilde{f}}$ states the same thing for the oracle $O_{\tilde{f}}$—that is, if $A_{\tilde{f}h}$ said $\tilde{f}h$, then she says that $O_{\tilde{f}}$ must have been $\tilde{f}$, whereas if $A_{\tilde{f}h}$ said that $O_{\tilde{f}h}$ is r, then she says that $O_{\tilde{f}}$ is r'. Notice that if the oracle $O_{\tilde{f}}$ is $\tilde{f}(K_2, .)$, then the oracle $O_{\tilde{f}h}$ that $A_{\tilde{f}}$ simulates for $A_{\tilde{f}h}$ is $\tilde{f}h$ (with random key $K = (K_1, K_2)$); if the oracle $O_{\tilde{f}}$ is $r'(.)$, then the oracle that $A_{\tilde{f}}$ simulates for $A_{\tilde{f}h}$ acts as r with the important difference that if $h(K_1, M^i)$ coincides with an earlier $h(K_1, M^j)$ the oracle outputs the same value (even though $M^i \neq M^j$) rather than a second random value.[3] If the latter happens with negligible probability, then this algorithm $A_{\tilde{f}}$ is as successful in distinguishing $\tilde{f}$ from a random function as $A_{\tilde{f}h}$ is in distinguishing $\tilde{f}h$ from a random function. Otherwise, two sequences of f-adversaries $A_f^{(m)}$ and $B_f^{(m)}$ come into the picture, as described below.

The general idea of these adversaries is that they each use the oracle O_f in the pseudorandomness test for f to look for collisions between h values of two different messages M^i, M^j queried by $A_{\tilde{f}h}$. More precisely, the mth adversary in a sequence works not with all of a queried message but rather with the message with its first $m-1$ blocks deleted. If a collision is produced, then with a certain probability, O_f must be $f(K_2, .)$; however, one must also account for the possibility that O_f is $r'(.)$, and in the case of $A_f^{(m)}$, this brings in the next adversary in the sequence $A_f^{(m+1)}$.

First we make a remark about probabilities, which are taken over all possible coin tosses of the adversary, all possible keys, the oracle's "choice bit" (which determines whether it is the function being tested or a random function), and the coin tosses of the oracle in the case when it outputs a random function.[4] If the adversary's oracle is f or $\tilde{f}h$ with hidden keys, then the adversary's queries in general depend on the keys (upon which the oracle's responses depend) as well as the adversary's coin tosses. However, if the adversary's oracle is a random function—which is the situation when $A_{\tilde{f}}$ fails and the sequences of adversaries $A_f^{(m)}$ and $B_f^{(m)}$ are needed—then the oracle responses can be regarded simply as additional coin tosses, and the adversary's queries then depend only on the coin tosses and are independent of the keys. This is an important observation for understanding the success probabilities of the adversaries.

We define α_0 to be the probability, taken over all coin tosses of $A_{\tilde{f}h}$ (including those coming from random oracle responses) and all keys K_1, that the sequence of $A_{\tilde{f}h}$ queries M^i satisfies the following property:

There exist i and j, $j < i$, such that $h(K_1, M^i) = h(K_1, M^j)$.

For $m \geq 1$, we define α_m to be the probability, taken over all coin tosses of $A_{\tilde{f}h}$ and all q-tuples of keys $(K_1, K_2, \ldots, K_q)$, that the sequence of $A_{\tilde{f}h}$ queries M^i satisfies the following property:

(1_m) there exist i and j, $j < i$, such that $M^{i,(m+1)} \neq \emptyset$, $M^{j,(m+1)} \neq \emptyset$,

$$h(K_{\ell_i}, M^{i,(m+1)}) = h(K_{\ell_j}, M^{j,(m+1)}),$$

[3]If $A_{\tilde{f}h}$ fails to produce a guess about the oracle $O_{\tilde{f}h}$ in time t, as can happen if the simulation is imperfect, then $A_{\tilde{f}}$ guesses that $O_{\tilde{f}}$ is a random function. Note that the simulation is perfect if $O_{\tilde{f}}$ is $\tilde{f}$ with hidden key.

[4]The term "over all possible coin tosses" means over all possible runs of the algorithm with each weighted by 2^{-s}, where s is the number of random bits in a given run.

where for any index i for which $M^{i,(m+1)} \neq \emptyset$, we let ℓ_i denote the smallest index for which $M^{\ell_i,(m+1)} \neq \emptyset$ and $M^{i,[m]} = M^{\ell_i,[m]}$.

Finally, for $m \geq 1$, we define β_m to be the probability, taken over all coin tosses of $A_{\tilde{f}h}$ and all q-tuples of keys $(K_1, K_2, \ldots, K_q)$, that the sequence of $A_{\tilde{f}h}$ queries M^i satisfies the following property:

(2_m) there exist i and j such that $M^{i,(m+1)} = \emptyset$, $M^{j,(m+1)} \neq \emptyset$,

$$M^{i,[m]} = M^{j,[m]}, \qquad \text{and} \qquad h(K_i, M^{j,(m+1)}) = K_i.$$

We now return to the situation where with non-negligible probability α_0 the queries made by $A_{\tilde{f}h}$ lead to at least one collision $h(K_1, M^i) = h(K_1, M^j)$. Note that the advantage of the adversary $A_{\tilde{f}}$ is at least $\epsilon - 2\epsilon_3 - \alpha_0$. If condition (ii) fails, i.e., if this advantage is $< \epsilon_2$, it follows that $\alpha_0 > \epsilon - \epsilon_2 - 2\epsilon_3$. In the remainder of the proof, we shall assume that this is the case, since otherwise (ii) holds and we're done.

The first adversary in the sequence $A_f^{(m)}$ is A'_f, which is given the oracle O_f that is either $f(K_1, .)$ with a hidden random key K_1 or else $r'(.)$. As A'_f runs $A_{\tilde{f}h}$, giving random responses to its queries, she queries O_f with the first block M_1^i of each $A_{\tilde{f}h}$ query M^i. If $M^{i,(2)}$ is nonempty, she then computes $y_i = h(O_f(M_1^i), M^{i,(2)})$; if $M^{i,(2)}$ is empty, she just takes $y_i = O_f(M_1^i)$. If O_f is $f(K_1, .)$, then y_i will be $h(K_1, M^i)$, whereas if O_f is $r'(.)$, then y_i will be $h(L_i, M^{i,(2)})$ for a random key $L_i = O_f(M_1^i)$ if $M^{i,(2)}$ is nonempty and will be a random value L_i if $M^{i,(2)}$ is empty. As the adversary A'_f gets these values, she looks for a collision with the y_j values obtained from earlier queries M^j. If a collision occurs, she guesses that O_f is f with hidden key; if not, she guesses that O_f is $r'(.)$.

It is, of course, conceivable that even when O_f is $r'(.)$, there is a collision $h(L_i, M^{i,(2)}) = h(L_j, M^{j,(2)})$ with $M^{i,(2)}$ and $M^{j,(2)}$ nonempty. Note that $L_i = L_j$ if $M_1^i = M_1^j$, but L_i and L_j are independent random values if $M_1^i \neq M_1^j$. In other words, we have (1_1). Recall that the probability that this occurs is α_1.

It is also possible that even when O_f is $r'(.)$ there is a collision involving one or both of the random values L_i or L_j that is produced when $M^{i,(2)}$ or $M^{j,(2)}$ is empty. If both are empty, then the probability that $L_i = L_j$ is 2^{-c}. If, say, $M^{j,(2)}$ is nonempty, then in the case $M_1^i \neq M_1^j$, we again have probability 2^{-c} that $L_i = h(L_j, M^{j,(2)})$, whereas in the case $M_1^i = M_1^j$, we have (2_1) with $K_i = L_i$.

Bringing these considerations together, we see that the advantage of A'_f is $\geq \alpha_0 - \alpha_1 - \beta_1 - \binom{q}{2}2^{-c}$.

We next describe the sequence of adversaries $A_f^{(m)}$, $m \geq 2$. Let O_f again denote the prf-test oracle for f that $A_f^{(m)}$ can query. Like A'_f, he runs $A_{\tilde{f}h}$ once and gives random responses to its queries. As $A_{\tilde{f}h}$ makes queries, he sorts their prefixes (where we are using the word "prefix" to denote the first $m - 1$ blocks of a query that has block length at least m). If the ith query has block length at least m and if its prefix coincides with that of an earlier query, he records the index ℓ_i of the first query that has the same prefix; if it has a different prefix from earlier queries, he sets $\ell_i = i$. After running $A_{\tilde{f}h}$, he goes back to the first query M^{j_1} that has block-length at least m, and for all i for which $\ell_i = j_1$ (i.e., for all queries that have the same prefix as M^{j_1}), he queries M^i_m to O_f and computes $y_i = h(O_f(M^i_m), M^{i,(m+1)})$ if $M^{i,(m+1)}$ is nonempty and otherwise takes $y_i = O_f(M^i_m)$. Then he resets O_f and goes to the first j_2 such that M^{j_2} has block length at least m and a different prefix from M^{j_1}. For all i for which $\ell_i = j_2$, he queries M^i_m to O_f and computes $y_i = h(O_f(M^i_m), M^{i,(m+1)})$ if $M^{i,(m+1)}$ is nonempty and otherwise takes $y_i = O_f(M^i_m)$. He continues in this way until he's gone through all the queries. He then looks for two indices $j < i$ such that $y_j = y_i$. If he finds a collision, he guesses that O_f is f with hidden key; otherwise, he guesses that it is a random function.

The adversary $A^{(m)}_f$ takes advantage of the α_{m-1} probability of a collision of the form (1_{m-1}), and if such a collision occurs, he guesses that O_f is f with hidden key. The possibility that O_f is really $r'(.)$ is due to two conceivable circumstances—a collision of the form (1_m) or a collision among random values (either a collision between two random values L_i and L_j or between L_i and $h(L_j, M^{j,(m+1)})$ or else a collision of the form (2_m) with $K_i = L_i$—here the probability of such a collision is bounded by $\binom{q}{2}2^{-c}$ and by β_m, respectively).

Finally, the sequence of adversaries $B^{(m)}_f$, $m \geq 1$, is defined as follows. As usual, O_f denotes the prf-test oracle for f that $B^{(m)}_f$ can query. She runs $A_{\tilde{f}h}$ once and gives random responses to its queries. As $A_{\tilde{f}h}$ makes queries, she sorts their prefixes (where this time, we are using the word "prefix" to denote the first m blocks of a query that has block length at least m). She makes up a list of pairs $(i, S(i))$, where the ith query has block length exactly m and coincides with the prefix of at least one other query; in that case $S(i)$ denotes the set of indices $j \neq i$ such that $M^{j,[m]} = M^i$. After running $A_{\tilde{f}h}$, she chooses a message block Y that is different from all the blocks M^j_{m+1} of all queries M^j. She goes through all indices i with nonempty $S(i)$. For each such i, she queries Y to O_f, and for each $j \in S(i)$, she also queries M^j_{m+1} to O_f and computes $y_j = h(O_f(M^j_{m+1}), M^{j,(m+2)}, Y)$. She looks for a collision between $O_f(Y)$ and y_j for $j \in S(i)$. Before going to the next i, she resets O_f. If she finds a collision for any of the i, she guesses that O_f is f with hidden key; otherwise, she guesses that it is a random function. The advantage of this adversary is at least $\beta_m - q2^{-c}$, because if O_f is $f(K_i, .)$ and $h(K_i, M^{j,(m+1)}) = K_i$, then $h(O_f(M^j_{m+1}), M^{j,(m+2)}, Y) = f(K_i, Y) = O_f(Y)$, whereas if O_f is a random function, then $O_f(Y)$ has probability only 2^{-c} of coinciding with this h-value.

We thus have the following lower bounds for the advantages of the adversaries:

A'_f: $\alpha_0 - \alpha_1 - \beta_1 - \binom{q}{2}2^{-c}$.
$A^{(m)}_f$, $m \geq 2$: $\alpha_{m-1} - \alpha_m - \beta_m - \binom{q}{2}2^{-c}$.
$B^{(m)}_f$, $m \geq 1$: $\beta_m - q2^{-c}$.

Trivially we have $\alpha_n = \beta_n = 0$, and so the adversaries go no farther than $A_f^{(n)}$ and $B_f^{(n-1)}$. The sum of all the advantages of the $2n-1$ adversaries telescopes and is at least $\alpha_0 - (2n-1)\binom{q}{2}2^{-c}$.

Since we have no way of knowing which of these adversaries has the greatest advantage, we make a random selection. That is, the adversary A_f we use to attack the pseudorandomness of f consists of randomly choosing one of the $2n-1$ adversaries A'_f, $A_f^{(m)}$ $(2 \le m \le n)$, $B_f^{(m)}$ $(1 \le m \le n-1)$ and running it. The advantage of the adversary A_f is the expectation obtained by summing the advantages of the $2n-1$ adversaries with each one weighted by the probability $1/(2n-1)$ that we choose the corresponding adversary. This advantage is at least $\frac{1}{2n-1}(\alpha_0 - (2n-1)\binom{q}{2}2^{-c}) > (\frac{\epsilon-\epsilon_2-2\epsilon_3}{2n} - \binom{q}{2}2^{-c})$. Thus, returning to the first paragraph of the proof, we have shown that if conditions (i) and (ii) do not hold, then condition (iii) holds. □

4 Interpretation

How useful is our theorem as a guarantee of real-world security? As in the case of Theorem 10.1 of [17], there are several reasons for skepticism concerning the practical assurance provided by Theorem 1:

1. In order to conclude that our MAC is an (ϵ, t, q, n)-secure pseudorandom function, we need the inner compression function f to be a strongly $(\epsilon/(2n), t, q)$-secure pseudorandom function. In other words, we have a tightness gap of about $2n$, which is large if, for example, we allow a block-length bound of 2^{20} or 2^{30}.[5]
2. Theorem 1 is in the single-user setting, and its security assurances could fail in the more realistic multiuser setting.
3. The three hypotheses in Theorem 1—pseudorandomness of the outer compression function, strong pseudorandomness of the inner compression function,[6] and the related-key property—are in general extremely difficult to evaluate. When the assumptions in a theorem cannot be evaluated in any convincing manner, we should not be surprised if practitioners view the theorem as having little value.

[5]The tightness gap in our theorem, bad as it is, is not nearly as extreme as the one in Fischlin's theorem [10], which establishes the secure-MAC property for NMAC and HMAC based on assumptions that are slightly weaker than the prf property. The gap in success probabilities in that theorem is roughly qn^2. In [10] this gap is compared to the q^2n gap in Bellare's Theorem 3.3 in [2]. However, any comparison based solely on success probabilities is misleading, since the factor q^2n in Bellare's theorem is multiplied by the advantage of a very low-resource adversary A_2 with running time $\le nT$, much less than that of Fischlin's adversary. One must always include running time comparisons when evaluating tightness gaps, and this is not done in [10].

[6]Note that the inner compression function needs to be strongly (ϵ_1, t, q)-secure for a quite small value of ϵ_1, since the theorem loses content if $\epsilon_1 > 1/(2n)$.

5 Security Theorem with Collision Resistance

There are two types of security theorems that have been proved about nested MACs. Starting with Bellare's paper [2] (see also [10, 17, 24]), some authors have proved theorems without assuming collision resistance of the iterated hash function. The idea is that confidence in security of a MAC scheme should not depend upon the rather strong assumption that an adversary cannot find hash collisions. Our Theorem 1 continues this line of work.

On the other hand, if one is using a hash function that one strongly believes to be collision resistant and one wants to know that an adversary cannot forge message tags, then one can go back to the much earlier and more easily proved security theorem in [6]. The purpose of this section is to carry over the main result of [6] to our class of 1-key nested MACs.

An iterated hash function h is said to be (ϵ, t, q, n) weakly collision resistant if no (t, q, n)-adversary that queries $h(K, .)$ has success probability $\geq \epsilon$ of producing a collision $h(K, M') = h(K, M)$, where M and M' are distinct messages of block-length $\leq n$. (Here $h(K, .)$ is regarded as an oracle, i.e., a black box, and K is a hidden key.) A MAC is said to be (ϵ, t, q, n)-secure against forgery if no (t, q, n)-adversary has success probability $\geq \epsilon$ of producing a tag for an unqueried message of block length $\leq n$.

Theorem 2 *Suppose that the iterated hash function h coming from the compression function f is (ϵ_1, t, q, n) weakly collision resistant, the compression function $\tilde{f}$ is an $(\epsilon_2, t, q, 1)$-secure MAC, and $(f, \tilde{f})$ is $(\epsilon_3, t, 2q + 2)$-secure against (g_1, g_2)-related-key attacks. Then MAC$_{\tilde{f},f,g_1,g_2}$ is an $(\epsilon_1 + \epsilon_2 + \epsilon_3, t - (q + 1)nT, q, n)$-secure MAC, where T is the time required for one evaluation of f or $\tilde{f}$.*

Proof The proof is quite simple. Suppose that we are given a $(t - (q + 1)nT, q, n)$-adversary A_{MAC} that has probability $\geq \epsilon_1 + \epsilon_2 + \epsilon_3$ of forging a MAC$_{\tilde{f},f,g_1,g_2}$-tag. Then we construct three adversaries—a (t, q)-adversary $A_{\tilde{f}}$, a (t, q, n)-adversary A_{wcr}, and a $(t, 2q + 2)$-adversary A_{rka}—such that at least one of the following is true:

(i) $A_{\tilde{f}}$ has probability $\geq \epsilon_2$ of forging a $\tilde{f}$-tag.
(ii) A_{wcr} has probability $\geq \epsilon_1$ of producing an h collision.
(iii) A_{rka} has advantage $\geq \epsilon_3$ against the (g_1, g_2)-related-key property of $(f, \tilde{f})$.

(It does not matter which of (i)–(iii) is true, since any one of the three would contradict the assumptions of the theorem.)

We first use A_{MAC} to construct both an adversary A_{rka} of the related-key property and an adversary $A_{\tilde{f}h}$ that can forge an $\tilde{f}h$ tag. (Recall that $\tilde{f}h$ denotes the MAC $\tilde{f}(K_2, h(K_1, M))$ with independent keys.) The adversary A_{rka} runs A_{MAC} and interacts with the oracle O_{rka}, which chooses a random bit u. After querying the oracle O_{rka} with at most $2q + 2$ b-bit or c-bit messages, A_{rka} must guess whether O_{rka} is using random keys (i.e., $u = 0$) or related keys (i.e., $u = 1$). As A_{MAC} runs, for each of its queries M^i the adversary A_{rka} queries the first block M_1^i to O_{rka}, then computes $H = h(O_{\rm rka}(M_1^i), M^{i,(2)})$, and finally queries H to $O_{\rm rka}$; it gives the value $O_{\rm rka}(H)$ to $A_{\rm MAC}$ as the tag of the queried message. If in time $\le t-(q+1)nT$ the adversary $A_{\rm MAC}$ forges a tag of an unqueried message,[7] then $A_{\rm rka}$ guesses that $u=1$; otherwise, it guesses that $u=0$. Let α denote the advantage of $A_{\rm rka}$, where, by definition

$$\alpha = \text{Prob}\big(A_{\rm rka} \text{ guesses } 1 \bigm| u=1\big) - \text{Prob}\big(A_{\rm rka} \text{ guesses } 1 \bigm| u=0\big). \qquad (1)$$

The time $A_{\rm rka}$ needs to perform these steps—that is, computing the values H, waiting for $A_{\rm MAC}$, and verifying the forgery produced by $A_{\rm MAC}$—that are bounded by $qnT + (t-(q+1)nT) + nT = t$.

We construct $A_{\tilde{f}h}$ as follows. It has an $\tilde{f}h$-oracle $O_{\tilde{f}h}$ that has hidden keys K_1, K_2. The adversary $A_{\tilde{f}h}$ runs $A_{\rm MAC}$, responding to each of its queries M^i by querying $O_{\tilde{f}h}$ and giving $A_{\rm MAC}$ the response $O_{\tilde{f}h}(M^i)$. If $A_{\rm MAC}$ forges the $\tilde{f}h$ tag of an unqueried message $\tilde{M}$ (which $A_{\tilde{f}h}$ can verify with one further query to its oracle), then $A_{\tilde{f}h}$ has succeeded in forging the $\tilde{f}h$ tag of $\tilde{M}$. Let β denote the success probability of this adversary $A_{\tilde{f}h}$.

Note that the interaction of $A_{\tilde{f}h}$ with $A_{\rm MAC}$ is exactly the same as that of $A_{\rm rka}$ with $A_{\rm MAC}$ in the case $u=0$. Thus, the second term on the right in the expression (1) for α is equal to β, whereas the first term is the success probability of $A_{\rm MAC}$, which by assumption is at least $\epsilon_1+\epsilon_2+\epsilon_3$. We hence have $\alpha+\beta \ge \epsilon_1+\epsilon_2+\epsilon_3$, and this means that either $\alpha \ge \epsilon_3$ (which is the alternative (iii) above) or else $\beta \ge \epsilon_1+\epsilon_2$.

We now use $A_{\tilde{f}h}$ to construct an $\tilde{f}$ tag-forging adversary $A_{\tilde{f}}$ and an h collision-finding adversary $A_{\rm wcr}$. The former is constructed as follows. After choosing a random key K_1, $A_{\tilde{f}}$ runs $A_{\tilde{f}h}$. In response to each query, M^i from $A_{\tilde{f}h}$, $A_{\tilde{f}}$ computes $H = h(K_1, M^i)$, queries this H value to its oracle $O_{\tilde{f}} = \tilde{f}(K_2, .)$, and gives the value $\tilde{f}(K_2, H)$ to $A_{\tilde{f}h}$. With probability β in time $\le t-(q+1)nT$, the adversary $A_{\tilde{f}h}$ finds a tag $\tilde{T} = \tilde{f}h(\tilde{M})$, where $\tilde{M}$ is different from all of the queried messages. The bound on the time $A_{\tilde{f}}$ needs to perform these steps—that is, computing the values $h(K_1, M^i)$ and waiting for $A_{\tilde{f}h}$—is $qnT + (t-(q+1)nT) = t-nT$. Then $A_{\tilde{f}}$ takes time $\le nT$ to compute $\tilde{H} = h(K_1, \tilde{M})$, hoping that it is different from all H values that were queried to $O_{\tilde{f}}$, in which case it has succeeded in forging an $\tilde{f}$ tag. Meanwhile, the adversary $A_{\rm wcr}$, which has an oracle $O_{\rm wcr}$ that responds to queries with $h(K_1, .)$ where K_1 is a hidden key, is constructed as follows. It chooses a random key K_2 and runs $A_{\tilde{f}h}$, responding to its queries M^i with $\tilde{f}(K_2, O_{\rm wcr}(M^i))$. $A_{\rm wcr}$ looks for a collision between some

[7]Note that $A_{\rm rka}$ can verify that $A_{\rm MAC}$ has a valid forgery using the same procedure that was used to respond to its queries. This means that $A_{\rm rka}$ needs to be allowed two more queries of $O_{\rm rka}$, and for this reason, the query bound for $A_{\rm rka}$ is $2q+2$ rather than $2q$, and the time bounds have the term $(q+1)nT$ rather than qnT.

$O_{\text{wcr}}(M^i) = h(K_1, M^i)$ and $O_{\text{wcr}}(\tilde{M}) = h(K_1, \tilde{M})$, where $(\tilde{M}, \tilde{T})$ is the forgery produced by $A_{\tilde{f}h}$ in the event that the latter adversary succeeds in its task. Note that the success probability β of $A_{\tilde{f}h}$ is the sum of the success probability of $A_{\tilde{f}}$ and that of A_{wcr}. Since $\beta \geq \epsilon_1 + \epsilon_2$ if alternative (iii) does not hold, it follows that at least one of the above alternatives (i)–(iii) must hold.

This concludes the proof. □

This theorem, like Theorem 1, provides only a very limited type of security assurance. Two of the three "reasons for skepticism" listed in Sect. 4 also apply to Theorem 2: it assumes the (unrealistic) single-user setting, and one of its hypotheses—the secure-MAC property for the compression function—is very difficult to evaluate in practice. On the positive side, at least Theorem 2 is tight, unlike Theorem 1. It's reasonable to regard Theorem 2 as providing a type of assurance that the "domain extender" feature of the MAC scheme is not likely to be a source of security breaches, provided that h is weakly collision resistant.

The proof of Theorem 2 is short, straightforward, and in some sense tautological. Some people would even say that it is "trivial," although we would prefer not to use such a pejorative word in connection with the proof of Theorem 2. But in any case, it seems to us that proofs of this sort merely confirm what is more or less intuitively obvious from the beginning. Such proofs cannot serve as a meaningful source of confidence in a protocol, and they certainly cannot be a substitute for extensive testing and concrete cryptanalysis.

6 HMAC vs. Envelope MAC Comparison

As discussed in [19], it often happens that a type of cryptography enjoys nearly universal acceptance more for reasons of historical happenstance than because of its intrinsic advantages over the alternatives. At present HMAC is widely deployed, whereas Envelope MAC languishes in relative obscurity. But the reasons for this seem to lie in the peculiarities of the history of Envelope MAC, and one can argue that, despite this history, it deserves serious consideration as a secure and practical MAC scheme.

An Envelope MAC scheme was first presented by Tsudik in [28]. His scheme used two independent c-bit keys, and he argued informally that this would give it $2c$ bits of security. However, Preneel and van Oorschot [25] showed that the keys can be recovered in 2^{c+1} steps if one knows approximately $2^{c/2}$ message-tag pairs. That is, Tsudik was wrong, and two independent keys do not give significantly more security than one key.

Soon after, Kaliski and Robshaw [12] and Piermont and Simpson [23] presented a 1-key variant of Envelope MAC, but it had a flaw. To explain this, for concreteness, we'll use MD5 as the underlying hash function with $c = 128$, $b = 512$. Let p denote a 384-bit padding, used to extend a key to fill a 512-bit block. Given a 128-bit key K and a message M of arbitrary length, the tag is $h(K\|p\|M\|K^0)$ (where the 0 superscript indicates that 0s are appended to fill out the last message block). Note that the second K may spill over into two blocks since the bitlength of M is not required to be a multiple of b. Preneel and van Oorschot [26] exploited this overlap to design a key-recovery attack that needs approximately 2^{64} message-tag pairs and has running time 2^{64}. Because a MAC based on MD5 would be expected to require exhaustive search—that is, roughly 2^{128} steps—for key recovery, this attack exposed a serious defect in the variant of Envelope MAC in [12,23]. The Preneel-van Oorschot attack gave Envelope MAC a bad name. However, the attack was possible only because of poor formatting of the second key block.

This flaw can be removed simply by ensuring that each key lies in its own block. This was done by Yasuda [29]. Yasuda also gave a security proof, along the lines of Bellare's HMAC security proof in [2]; in fact, he made crucial use of Bellare's Lemma 3.1. Like Bellare's proof, Yasuda's security theorem requires unconstructible adversaries and so is not valid in the uniform model of complexity. Our Theorem 1 gives a uniform proof for the version of 1-key Envelope MAC described by Yasuda.

As pointed out in [29], Envelope MAC has a minor efficiency advantage over HMAC because the iterated hash function needs to be applied just once. Generally, the accepted procedure when applying an iterated hash function is to append a block at the end of the message that gives the block length of the message. (With this modification, one can give a simple proof that collision resistance of f implies collision resistance of h.) In Envelope MAC, this is done just once, whereas in HMAC it needs to be done twice. Envelope MAC also has the advantage of simplicity—no need for ipad and opad.

In order to argue that HMAC is preferable, one thus has to make a persuasive case that it has better security. Our two theorems give the same security results in both cases, and the same building block (a compression function f) can be used in both. From this standpoint the only grounds for preferring HMAC would be if one of the following holds:

1. The prf assumption on $\tilde{f}$ in Theorem 1 is more credible for HMAC than for Envelope MAC. In the former case, the assumption is weaker than the prf assumption on f and in fact follows from it. In the latter case the assumption also seems to be weaker than the prf assumption on f in practice, but not in the formal sense; in Envelope MAC the prf assumption on $\tilde{f}$ is an additional condition that is not a consequence of the prf assumption on f.[8] One could claim that the need for a separate $\tilde{f}$ condition in Envelope MAC means that it is less secure than HMAC.

[8]In the prf test for f, the adversary gets the values $f(K, M)$ (with K a c-bit hidden key and M a b-bit queried message); in the test for $\tilde{f}$ in HMAC, he gets the values $f(K, M\|p)$ (with M a c-bit message and p a fixed $(b-c)$-bit padding); and in the test for $\tilde{f}$ in Envelope MAC, he gets the values $f(M, K\|p)$. Thus, the only difference is whether the key occurs in the first c bits or in the next c bits.

2. The secure-MAC assumption on $\tilde{f}$ in Theorem 2 is more credible for HMAC than for Envelope MAC. One would be claiming that it's harder to forge a tag if the key occurs in the first c bits than if it occurs in the next c bits.
3. The different choices of the pair of functions g_1, g_2 lead to a real difference in strength of the related-key assumptions. There would be a strong reason to prefer HMAC if one could argue that the choice $g_2(K) = K$ in Envelope MAC makes the related-key assumption less plausible.

However, we know of no evidence for any of the above three claims. To the best of our knowledge, no provable security theorem justifies preferring one of these two MACs over the other. Nor does any such theorem preclude the possibility that one would want to choose some other MAC with entirely different functions $\tilde{f}$, g_1, g_2.

Remark 3 Both HMAC and Envelope MAC offer the convenience of using only an off-the-shelf hash function with built-in IV. However, one can argue that for the outer compression function—which maps only from $\{0,1\}^c \times \{0,1\}^c$ rather than from the much larger set $\{0,1\}^c \times \{0,1\}^b$—it might be more efficient to use a specially chosen $\tilde{f}$. One can even argue that the inner compression function f needs to have better security than $\tilde{f}$ because it is iterated. That is why Theorem 1 has a $2n$ tightness gap with respect to the advantage bound ϵ_1 of an f adversary, but not with respect to the advantage bound ϵ_2 of an $\tilde{f}$ adversary. If one believes that security proofs should guide protocol design and that efficiency should not be sacrificed for security unless a provable security theorem gives grounds for doing so (in [13] Katz and Lindell argue forcefully for this viewpoint), then it is natural to conclude that $\tilde{f}$ *should* be less secure than f. Elsewhere (see [16]) we have raised doubts about this way of thinking, so our personal preference would be *not* to use a weaker $\tilde{f}$. But to someone who needs only short-term security, this might be an acceptable risk in order to gain a slight edge in efficiency.

Conclusion

The Importance of the "Right" Definitions

In their highly regarded textbook [13] on the foundations of cryptography, Katz and Lindell write that the "formulation of exact definitions" is Principle 1 in their list of "basic principles of modern cryptography" and that getting the definitions right is the "essential prerequisite for the...study of any cryptographic primitive or protocol." We agree with this statement, and in [16] we analyzed some of the difficulties and challenges that researchers in both symmetric and asymmetric cryptography have faced in trying to search for a consensus on what the "right" definitions are.

(continued)

In our study of 1-key nested MACs, we have encountered two instances where the standard accepted definitions are not, in our opinion, the natural and useful ones. First, as we explained in Sect. 9 of [17], in the context of iterated hash functions the usual definition of pseudorandomness needs to be replaced by a stronger definition in which the adversary is given the power to "reset" the oracle. In the second place, when analyzing the step from NMAC to 1-key versions such as HMAC and Envelope MAC, we believe that our definition of resistance to related-key attack is preferable to the one used by earlier authors.

We have given arguments justifying our use of these new definitions. Nevertheless, it would be arrogant in the extreme for us to claim that we have resolved the question or that our viewpoint is definitive. Cryptography is as much an art as a science, and to some extent decisions about which are the "right" definitions are matters of personal taste.

The Role of Mathematical Proofs

The NIST documents concerning the SHA-3 competition illustrate the limited role that provable security plays in evaluating real-world cryptosystems. The report [22] that explains the rationale for the selection of the winner devotes about 5 % of the section on security to security proofs. The report highlights the role of proofs in showing a hash function's "security against generic attacks—attacks that only exploit the domain extender and not the internals of the underlying building blocks" (p. 11). It notes that all five finalists have this sort of security proof. In other words, the security proofs are useful as a minimal type of assurance that basically says that concrete cryptanalysis should focus on the underlying building blocks rather than on the extension procedure. But the final decision about what to use must be based on extensive testing and concrete cryptanalysis. NIST makes it clear that provable security, although a necessary component in the security analysis, played no part in ranking the five finalists.[9]

In choosing a MAC scheme, provable security (which, as we argued in [15], is a misnomer) should play no greater role than it did in choosing SHA-3. All methods of the form in Theorem 1 for constructing MACs from compression functions are good domain extenders if they satisfy the

(continued)

[9]How can something be a necessary component, but play no role in the selection? By analogy, when one looks for an apartment, a functioning toilet is a requirement; however, one doesn't normally choose which apartment to rent based on which has the nicest toilet.

hypothesis of that theorem—of course, "good" only in the limited sense guaranteed by the conclusion of the theorem. As in the case of the SHA-3 competition, the final choice has to be made through ad hoc testing rather than mathematical theorems. In particular, the relative merits of HMAC and Envelope MAC cannot be determined from provable security considerations. The choice between the two (or a decision to go with a totally different $\tilde{f}, g_1, g_2$) is a judgment call.

References

1. M. Bellare, Practice-oriented provable-security, in *Proceedings of First International Workshop on Information Security (ISW '97)*. Lecture Notes in Computer Science, vol. 1396 (Springer, Berlin, 1998), pp. 221–231
2. M. Bellare, New proofs for NMAC and HMAC: security without collision resistance, in *Advances in Cryptology—Crypto 2006*. Lecture Notes in Computer Science, vol. 4117 (Springer, Heidelberg, 2006), pp. 602–619. Extended version available at http://cseweb.ucsd.edu/mihir/papers/hmac-new.pdf
3. M. Bellare, T. Kohno, A theoretical treatment of related-key attacks: RKA-PRPs, RKA-PRFs, and applications, in *Advances in Cryptology—Eurocrypt 2003*. Lecture Notes in Computer Science, vol. 2656 (Springer, Heidelberg, 2003), pp. 491–506
4. M. Bellare, P. Rogaway, Random oracles are practical: a paradigm for designing efficient protocols, in *Proceedings of First Annual Conference on Computer and Communications Security* (ACM, New York, 1993), pp. 62–73
5. M. Bellare, R. Rogaway, Optimal asymmetric encryption—how to encrypt with RSA, in *Advances in Cryptology—Eurocrypt '94*. Lecture Notes in Computer Science, vol. 950 (Springer, Heidelberg, 1994), pp. 92–111
6. M. Bellare, R. Canetti, H. Krawczyk, Keying hash functions for message authentication, in *Advances in Cryptology—Crypto '96*. Lecture Notes in Computer Science, vol. 1109 (Springer, Heidelberg, 1996), pp. 1–15. Extended version available at http://cseweb.ucsd.edu/mihir/papers/kmd5.pdf
7. M. Bellare, R. Canetti, H. Krawczyk, HMAC: Keyed-hashing for message authentication, Internet RFC 2104 (1997)
8. D. Bernstein, T. Lange, Non-uniform cracks in the concrete: the power of free precomputation, in *Advances in Cryptology—Asiacrypt 2013*. Lecture Notes in Computer Science, vol. 8270 (Springer, Heidelberg, 2013), pp. 321–340
9. D. Boneh, Simplified OAEP for the RSA and Rabin functions, in *Advances in Cryptology—Crypto 2001*. Lecture Notes in Computer Science, vol. 2139 (Springer, Heidelberg, 2001), pp. 275–291
10. M. Fischlin, Security of NMAC and HMAC based on non-malleability, in *Topics in Cryptology—CT-RSA 2008*. Lecture Notes in Computer Science, vol. 4064 (Springer, Heidelberg, 2008), pp. 138–154
11. P. Gauravaram, L. Knudsen, K. Matusiewicz, F. Mendel, C. Rechberger, M. Schläffer, S. Thomsen, Grøstl—a SHA-3 candidate (2011). Available at http://www.groestl.info/Groestl.pdf
12. B. Kaliski, M. Robshaw, Message authentication with MD5. CryptoBytes **1**(1), 5–8 (1995)
13. J. Katz, Y. Lindell, *Introduction to Modern Cryptography* (Chapman and Hall/CRC, Boca Raton, 2007)

14. N. Koblitz, A. Menezes. http://anotherlook.ca
15. N. Koblitz, A. Menezes, Another look at "provable security." J. Cryptol. **20**, 3–37 (2007)
16. N. Koblitz,A. Menezes, Another look at security definitions. Adv. Math. Commun. **7**, 1–38 (2013)
17. N. Koblitz, A. Menezes, Another look at HMAC. J. Math. Cryptol. **7**, 225–251 (2013)
18. N. Koblitz, A. Menezes, Another look at non-uniformity. Groups Complex. Cryptol. **5**, 117–139 (2013)
19. A.H. Koblitz, N. Koblitz, A. Menezes, Elliptic curve cryptography: the serpentine course of a paradigm shift. J. Number Theory **131**, 781–814 (2011)
20. H. Krawczyk, Koblitz's arguments disingenuous. Not. Am. Math. Soc. **54**(11), 1455 (2007)
21. National Institute of Standards and Technology, The keyed-hash message authentication code (HMAC). FIPS Publication 198 (2002)
22. National Institute of Standards and Technology, Third-round report of the SHA-3 cryptographic hash algorithm competition. Interagency Report 7896 (2012)
23. P. Piermont, W. Simpson, IP authentication using keyed MD5, IETF RFC 1828 (1995)
24. K. Pietrzak, A closer look at HMAC. Available at http://eprint.iacr.org/2013/212
25. B. Preneel, P. van Oorschot, MDx-MAC and building fast MACs from hash functions, in *Advances in Cryptology—Crypto '95*. Lecture Notes in Computer Science, vol. 963 (Springer, Heidelberg, 1995), pp. 1–14
26. B. Preneel, P. van Oorschot, On the security of iterated message authentication codes. IEEE Trans. Inf. Theory **45**, 188–199 (1999)
27. V. Shoup, OAEP reconsidered, in *Advances in Cryptology—Crypto 2001*. Lecture Notes in Computer Science, vol. 2139 (Springer, Heidelberg, 2001), pp. 239–259
28. G. Tsudik, Message authentication with one-way hash functions. ACM SIGCOMM Comput. Commun. Rev. **22**(5), 29–38 (1992)
29. K. Yasuda, "Sandwich" is indeed secure: how to authenticate a message with just one hashing, in *Information Security and Privacy—ACISP 2007*. Lecture Notes in Computer Science, vol. 4586 (Springer, Heidelberg, 2007), pp. 355–369