This paper appears in CT-RSA 2008, Lecture Notes in Computer Science, Springer-Verlag, 2008.

Security of NMAC and HMAC based on Non-Malleability

Marc Fischlin*

Darmstadt University of Technology, Germany
marc.fischlin@gmail.com www.fischlin.de

Abstract. We give an alternative security proof for NMAC and HMAC when deployed as a message authentication code, supplementing the previous result by Bellare (Crypto 2006). We show that (black-box) non-malleability and unpredictability of the compression function suffice in this case, yielding security under different assumptions. This also suggests that some sort of non-malleability is a desirable design goal for hash functions.

1 Introduction

HMAC is one of the most widely deployed cryptographic algorithms today. Proposed by Bellare et al. [BCK96a] it is nowadays standardized in several places like ANSI X9.71 and incorporated into SSL, SSH and IPSec. It is used as a universal tool to derive keys, to provide a pseudorandom function or simply to authenticate messages. Roughly, for keys $k_{\text{in}}, k_{\text{out}}$ algorithm HMAC, and its generalized version NMAC, are defined as

$$\mathsf{HMAC}_{(k_{\text{in}},k_{\text{out}})}(M) := H(\text{IV}, k_{\text{out}}||H(\text{IV}, k_{\text{in}}||M))$$
$$\mathsf{NMAC}_{(k_{\text{in}},k_{\text{out}})}(M) := H(k_{\text{out}}, H(k_{\text{in}}, M))$$

where H is an iterated hash function like MD5 or SHA1, based on some compression function h.

In the original paper of Bellare et al. [BCK96a] algorithms HMAC and NMAC have been shown to be a pseudorandom function assuming that the compression function h is pseudorandom *and* collision-resistant. With the emerging attacks on the collision-resistance on popular hash functions like MD5 and SHA1 [WY05, WYY05] the trustworthiness of HMAC and NMAC was slightly tarnished, but subsequently Bellare [Bel06] proved both algorithms to be pseudorandom under the sole assumption that the compression function is pseudorandom. This result is complemented by several works [CY06, KBPH06, RR07] showing that weaknesses in collision-resistance can be actually exploited to successfully attack HMAC and NMAC.

*This work was supported by the Emmy Noether Program Fi 940/2-1 of the German Research Foundation (DFG).

Our results. Here, we present alternative assumptions about the compression function to yield a security proof for HMAC and NMAC *when used as a message authentication code* (MAC), instead of being deployed as a pseudorandom function. We require two orthogonal properties of the compression function which are both implied simultaneously if, for instance, the compression function h is pseudorandom:

- non-malleability: learning images of the iterated compression function (with an unknown key involved) does not lend any additional power to create another hash value under this key.[1]
- unpredictability: it is infeasible to predict the output of the iterated compression function (with an unknown key involved).

Intuitively, unpredictability says that it is impossibile to guess images from scratch (i.e., with no other images available). This is a very weak form of a MAC but does not guarantee the common notion of security under adaptive chosen-message attacks. Adding non-malleability then provides this stronger security notion, as seeing other images does not facilitate the task.

We also show that, if there are pseudorandom functions at all, then there are compression functions which obey these two properties but are not pseudorandom. Hence, our result shows security under weaker prerequisites on the compression function, strengthening the confidence in the security of NMAC and HMAC when deployed as a MAC. Moreover, the result here indicates that non-malleability (or at least some relaxation thereof) is an eligible property for hash functions and their designs.

Related results. We stress that Bellare [Bel06], although using a stronger assumption, also derives a stronger statement, namely, that the pseudorandomness of the compression function carries over. Since HMAC is used for distinct purposes such as key derivation or as a pseudorandom function, this security claim is required for such cases. Our result merely supplements Bellare's more general result and shows that security of MACs is somewhat easier to achieve than pseudorandomness (cf. [AB99]).

In [Bel06] Bellare also considers weaker requirements for HMAC and NMAC used as a MAC. He introduces the notion of privacy-preserving MACs and shows that this condition suffices to guarantee security of the MAC, together with the fact that the compression function is computationally almost universal which, in turn, follows from the pseudorandomness of the compression function. Although resembling each other, privacy-preserving MACs and our notion of non-malleability are in general incomparable (as we discuss in Section 5.2). Our result can therefore be seen as an alternative security claim based on different assumptions. At the same time, for some specific cases, our notion of non-malleability implies privacy-preservation and thus also helps to characterize this property.

2 Preliminaries

We start by recalling HMAC and NMAC and then define our two properties, non-malleability and unpredictability, before formalizing security of message authentication codes.

[1] Here, depending on the padding of the hash function, we may require a special form of "black-box" non-malleability, implemented via so-called simulatable images.

2.1 HMAC and NMAC

Algorithms HMAC and NMAC are built from iterated hash functions based on a compression function h, mapping $\{0,1\}^n \times \{0,1\}^b$ to $\{0,1\}^n$. For such a compression function let the iteration of the compression function $h^*(k, M)$ for input $k \in \{0,1\}^n$ and $M = M[1] \dots M[n]$, consisting of b-bit blocks $M[i]$, be given by the value z_n, where $z_0 = k$ and $z_{i+1} = h(z_i, M[i+1])$ for $i = 0, 1, \dots, n-1$. For notational convenience we write $B = \{0,1\}^b$ and $B^{\leq N} = \cup_{i \leq N} B^i$ and $B^+ = \cup_{i \in \mathbb{N}} B^i$ such that $h^* : \{0,1\}^n \times B^+ \to \{0,1\}^n$.

Given the iterated compression function we can define the hash function H as follows. Let $H(k, M) = h^*(k, \mathsf{pad}(M))$ where $\mathsf{pad}(M)$ stands for the message padded to a multiple of b bits. Here $\mathsf{pad}(\cdot)$ is an arbitrary one-to-one function, and we write $\mathrm{Time}_L(\mathsf{pad})$ for the time to compute the padding function for any input of length at most L, and $\mathrm{Extend}_L(\mathsf{pad})$ for the maximal number of bits the padding function adds to each string of at most L bits. We furthermore assume that the padding length only depends on the input length. For example, the standard padding appends a 1-bit to M and then adds the smallest number of 0-bits to obtain a multiple of b bits, such that $\mathrm{Time}_L(\mathsf{pad}) = O(L + b)$ and $\mathrm{Extend}_L(\mathsf{pad}) = b + 1$.

We presume that the hash function's description also contains a fixed, public value IV and we set $H(M) = H(\mathrm{IV}, M)$. Define algorithms HMAC and NMAC now as:

$$\mathsf{HMAC}_{(k_{\mathrm{in}}, k_{\mathrm{out}})}(M) := H(k_{\mathrm{out}} || H(k_{\mathrm{in}} || M))$$
$$\mathsf{NMAC}_{(k_{\mathrm{in}}, k_{\mathrm{out}})}(M) := H(k_{\mathrm{out}}, H(k_{\mathrm{in}}, M))$$

where keys $k_{\mathrm{in}}, k_{\mathrm{out}}$ for NMAC consist of n-bits each and are used instead of IV, and keys $k_{\mathrm{in}}, k_{\mathrm{out}} \in \{0,1\}^b$ for HMAC are prepended to the strings.

In practice, HMAC is typically used with dependent keys $k_{\mathrm{in}} = k \oplus \mathrm{ipad}$ and $k_{\mathrm{out}} = k \oplus \mathrm{opad}$ for fixed constants $\mathrm{ipad} = \texttt{0x3636...36}$ and $\mathrm{opad} = \texttt{0x5c5c...5c}$ and a key k of at most b bits. In either case, one can view HMAC as a special case of NMAC with $k_{\mathrm{in}}^{\mathsf{NMAC}} = h(\mathrm{IV}, k_{\mathrm{in}}^{\mathsf{HMAC}})$ and $k_{\mathrm{out}}^{\mathsf{NMAC}} = h(\mathrm{IV}, k_{\mathrm{out}}^{\mathsf{HMAC}})$.

2.2 Non-Malleability and Simulatability

Non-malleability of a cryptographic function refers to the (in)ability to construct an image which is related to previously seen images. This is formalized by considering an experiment in which an adversary $\mathcal{A}$ can first ask to see hash values $y_i = H(x_i)$ of pre-images x_i distributed according to some distribution $\mathcal{X}$. Then $\mathcal{A}$ tries to find a hash value y^* of a related pre-image x^*, where related pre-images are specified through a relation R. The success probability of $\mathcal{A}$ should not be significantly larger than in an experiment of a simulator $\mathcal{S}$ which does not get to learn the images y_i but should still be able to find a related value y^*.

Non-malleability for hash functions has been defined in [BCFW07] and we follow their approach but state the property in terms of concrete security. The authors of [BCFW07] point out that the most general notion for hash functions and arbirtrary distributions and relations is not achievable. Fortunately, here we deal with the easier problem of considering only very special distributions and specific relations.

Below we formalize non-malleability for the *compression function* instead of the hash function to make a claim about the security propagation. In the adversary's experiment we let $\mathcal{A}$ "bias" the distribution of the pre-images x_i via a parameter $p_i \in B^+$ passed to the (stateful) distribution $\mathcal{X}(k, \cdot)$, using a random seed k. Formally, oracle GenSample takes the parameter p_i as input, computes $x_i \leftarrow \mathcal{X}(k, p_i)$ and the image $y_i = h(x_i)$. After having seen some sample images adversary $\mathcal{A}$ outputs an image y^* and a transformation T which maps

$x_1, x_2, \ldots$ to x^*. That is, the adversary does not need to know the pre-image x^* when creating y^*, but must only commit to the transformation which determines x^* once $x_1, x_2, \ldots$ become known (in fact, T produces x^* from $k, p_1, p_2, \ldots$ from which $x_1, x_2, \ldots$ can be derived). The simulator $\mathcal{S}$ only gets a restricted oracle $\mathsf{GenSample}_0$ which samples the x_i's but does not return the image y_i to $\mathcal{S}$. Still, the simulator should create a valid pair (T, y^*).

Definition 2.1 (Non-Malleable Compression Function) *A compression function $h : \{0,1\}^n \times \{0,1\}^b \to \{0,1\}^n$ is called $(t_{\mathcal{A}}, t_{\mathcal{S}}, Q, N, \mu)$-non-malleable with respect to distribution $\mathcal{X}$ and relation R if for any algorithm $\mathcal{A}$ with running time $t_{\mathcal{A}}$ there exists a simulator $\mathcal{S}$ with running time $t_{\mathcal{S}}$, such that*

$$\mathrm{Prob}\Big[\, \boldsymbol{Exp}^{nm\text{-}adv}_{h,\mathcal{A}} = 1 \Big] \leq \mathrm{Prob}\big[\, \boldsymbol{Exp}^{nm\text{-}sim}_{h,\mathcal{S}} = 1 \big] + \mu$$

where

Experiment $\boldsymbol{Exp}^{nm\text{-}adv}_{h,\mathcal{A}}$	***Experiment*** $\boldsymbol{Exp}^{nm\text{-}sim}_{h,\mathcal{S}}$
$k \leftarrow \{0,1\}^n$	$k \leftarrow \{0,1\}^n$
$(T, y^*) \leftarrow \mathcal{A}^{\mathsf{GenSample}(k,\cdot)}$	$(T, y^*) \leftarrow \mathcal{S}^{\mathsf{GenSample}_0(k,\cdot)}()$
where $\mathsf{GenSample}(k, p_i)$ *computes*	*where* $\mathsf{GenSample}_0(k, p_i)$ *computes*
$x_i \leftarrow \mathcal{X}(k, p_i)$	$x_i \leftarrow \mathcal{X}(k, p_i)$
$y_i = h(x_i)$	
and returns y_i	
$x^* \leftarrow T(k, p_1, p_2, \ldots)$	$x^* \leftarrow T(k, p_1, p_2, \ldots)$
Return 1 iff	*Return 1 iff*
$R(T, k, p_1, p_2, \ldots, x^*)$	$R(T, k, p_1, p_2, \ldots, x^*)$
$\wedge\, (x^*, y^*) \notin \{(x_1, y_1), (x_2, y_2), \ldots\}$	
$\wedge\, h(x^*) = y^*$	$\wedge\, h(x^*) = y^*$

Here $\mathcal{A}$ and $\mathcal{S}$ each make at most Q queries to their oracle, each query having at most N blocks.

We consider here a special distribution $\mathcal{X}_{\mathsf{NMAC}}(k, \cdot)$ which, on input $p_i = M_i[1] \ldots M_i[j+1]$ first computes the j-th iteration of the compression function $z_i = h^*(k, M_i[1] \ldots M_i[j])$ for random key k and then returns the pre-image $x_i = (z_i, M_i[j+1])$ (from which $y_i = h(x_i)$ is then derived). The relation R_{NMAC} for input $(T, k, p_1, p_2, \ldots, x^*)$ merely checks that the transformation T computes the output $(h^*(k, M[1] \ldots M[j]), M[j+1])$ for some constants $M[1], \ldots, M[j+1] \in B$ hardwired into T, and such that $M[1] \ldots M[j+1]$ has at most N blocks, and no p_i is a prefix of $M[1] \ldots M[j+1]$. The prefix-check is necessary to prevent standard extension attacks.

In a refinement of the non-malleability notion we consider π-*simulatable* compression functions which allow to simulate images (given only a fraction $\pi(p_i)$ of the parameter) and which can potentially be used to construct a *black-box* non-malleability simulator. Simulatability essentially says that images can be created without the (complete) pre-image, and thus immediately suggests a strategy for constructing the non-malleability simulator. Below we formalize the notion of simulatability by demanding that no efficient distinguisher can tell apart whether it is communicating with the $\mathsf{GenSample}$ oracle or the oracle $\mathsf{SimAnswer}$ simulating images:

Definition 2.2 (π-Simulatability) *A compression function $h : \{0,1\}^n \times \{0,1\}^b \to \{0,1\}^n$ is called $(t_{\mathcal{D}}, t_{\mathsf{Sim}}, Q, N, \sigma)$-$\pi$-simulatable for distribution $\mathcal{X}$ if there is an algorithm* $\mathsf{SimAnswer}$

running in time t_{Sim} such that for any algorithm $\mathcal{D}$ running in time $t_{\mathcal{D}}$ and making at most Q queries, each of at most N blocks,

$$\text{Prob}\left[\mathcal{D}^{\mathsf{GenSample}(k,\cdot)} = 1\right] \leq \text{Prob}\left[\mathcal{D}^{\mathsf{SimAnswer}(\pi(\cdot))} = 1\right] + \sigma$$

where oracle GenSample *is defined as in Definition 2.1, and where we assume that $\mathcal{D}$ never queries its oracles about the same value twice. The probabilites are taken over $\mathcal{D}$'s random choices, and $k \leftarrow \{0,1\}^n$ in the first case and the randomness of* SimAnswer *in the second case.*

Given algorithm SimAnswer one can construct a black-box non-malleability simulator as follows. Consider another algorithm Interface which basically provides the interface between the simulator's oracle $\mathsf{GenSample}_0$ and the queries made by the simulated adversary $\mathcal{A}$. Then the non-malleability simulator is of the form $\mathcal{S} = \mathcal{A}^{\mathsf{Interface}(\cdot)}$, where Interface on input p_i forwards this value to oracle $\mathsf{GenSample}_0$ of $\mathcal{S}$ and then computes $y_i \leftarrow \mathsf{SimAnswer}(\pi(p_i))$ and returns it to $\mathcal{A}$. This simulator basically inherits the properties of SimAnswer, namely, runs in time $t_{\mathcal{S}} = t_{\mathcal{A}} + Q \cdot (\text{Time}(\mathsf{SimAnswer}) + \text{Time}(\pi))$, makes at most Q queries and is σ-close. This is under one condition: it must be possible to map the difference in the output behavior (T, y^*) of $\mathcal{A}$ when communicating with oracle GenSample or with oracle SimAnswer to a distinguisher with binary output. This will indeed be the case for our application.

2.3 Unpredictability

A trivial example of a (π-simulatable) non-malleable compression function is a constant function: any information available through images is known beforehand and therefore redundant. Of course, such examples do not yield a good MAC, and in order to avoid such contrived cases we introduce the mild assumption of unpredictability. Basically, a compression function is unpredictable if one cannot determine a parameter $p \in B^*$ (specifying a distribution as in the non-malleability definition), a message block $m \in B$ and its image $z = h(h^*(k,p),m)$, *before* the key k is chosen:

Definition 2.3 (Unpredictablility) *A compression function $h : \{0,1\}^n \times \{0,1\}^b \to \{0,1\}^n$ is (t, N, ρ)-unpredictable if for any algorithm P running in time t, the probability that for $(p, m, z) \leftarrow P()$ and $k \leftarrow \{0,1\}^n$ we have $h(h^*(k,p),m) = z$ and $p||m \in B^{\leq N}$, is at most ρ.*

In the sequel we often view p and m as one message $M = p||m$ such that the definition says one cannot predict $h^*(k,M) = h(h^*(k,p),m)$. A trivial example of a $(t, N, 2^{-n})$-unpredictable compression function is the identity function (on the key part), $h(k,m) = k$. Another example are pseudorandom compression functions, as we prove formally in Section 5.

The examples of a constant compression function $h(k,m) = 0^n$ and (a modification of) the "identity-on-key-part" function $h(k,m) = k$ also separate the notions of non-malleability and unpredictability. The former function is clearly π-simulatable (for any π) and non-malleable —the simulator can easily simulate the oracle's answers— but not unpredictable. In contrast, a slight modification of the latter function, namely $h(k,m) = k \oplus \text{lsb}_n(m)$ for the n least significant bits $\text{lsb}_n(m)$ of $m \in \{0,1\}^b$, is unpredictable, yet malleable for $\mathcal{X}_{\mathsf{NMAC}}$ and R_{NMAC}. Malleability follows as an adversary is able to recover the key k from a single query about message $m = 0^b$ and can then output the image $k \oplus 1^n$ for message $m^* = 1^b$ (and the corresponding transformation). A simulator, on the other hand, needs to output the image $k \oplus m$ for constant m in clear without having any information about the key k.

We claim that $h(k,m) = k \oplus \mathrm{lsb}_n(m)$ is not π-simulatable for any π with output length strictly less than n. A distinguisher $\mathcal{D}$ merely forwards distinct random values $m_0, m_1 \in \{0,1\}^b$ and checks that the replies y_0, y_1 satisfy $y_0 \oplus y_1 \oplus m_0 \oplus m_1 = 0^n$. For the function h this will be the case with probability 1. For $\mathsf{SimAnswer}$ this probability cannot be more than $\frac{1}{2}$, since $\mathsf{SimAnswer}$ lacks at least one bit of information about $\mathrm{lsb}_n(m_0 \oplus m_1)$ from $\pi(m_0), \pi(m_1)$.

2.4 Message Authentication Codes

Our goal is to show that non-malleability of the compression function (plus unpredictability) gives a secure message authentication code. Formally, a message authentication code $\mathcal{M} = (\mathsf{KeyGen}, \mathsf{MAC}, \mathsf{Vf})$ consists of three (probabilistic) algorithms, the key generation algorithm returning a key $K \leftarrow \mathsf{KeyGen}()$, the MAC algorithm computing a message authentication code $\tau \leftarrow \mathsf{MAC}(K, M)$ for a message M, and a verification algorithm deciding upon acceptance $a \leftarrow \mathsf{Vf}(K, M, \tau)$ for a message M and a putative MAC τ. MACs generated by the key-holder should always be accepted, i.e., for any $K \leftarrow \mathsf{KeyGen}()$, any message M and any $\tau \leftarrow \mathsf{MAC}(K, M)$ we always have $\mathsf{Vf}(K, M, \tau) = 1$.

Definition 2.4 *A MAC $\mathcal{M} = (\mathsf{KeyGen}, \mathsf{MAC}, \mathsf{Vf})$ is called (t, Q, L, ϵ)-unforgeable if for any algorithm $\mathcal{B}$ running in time t, the probability that for $K \leftarrow \mathsf{KeyGen}()$, $(M^*, \tau^*) \leftarrow \mathcal{B}^{\mathsf{MAC}(K,\cdot)}()$ making at most Q queries $M_1, M_2, \ldots$ to oracle $\mathsf{MAC}(K, \cdot)$, each query and M^* of at most L bits, we have $\mathsf{Vf}(K, M^*, \tau^*) = 1$ and $M^* \notin \{M_1, M_2, \ldots\}$, is at most ϵ.*

In the definition above we let the adversary make only a single verification query. Bellare et al. [BGM04] have shown that for HMAC and NMAC this implies security against adversaries which make v arbitrarily interleaved verification queries for fresh messages not queried previously. This comes with a loss of at most a factor v in security (and some minor change in the running time parameter t).

3 Security of NMAC

We first show that NMAC is a secure MAC, given that the compression function is non-malleable, simulatable and unpredictable. For simplicity we refer to NMAC as both the MAC algorithm and the scheme (with straightforward key generation and verification algorithm).

We simply state the theorem and present the proof for π-simulatable, where π is a constant function 0; afterwards we discuss that the theorem holds for more general functions. We also remark that we can give a proof based on non-malleability and unpredictability only (i.e., without simulatability) if the padding function is prefix-free. See Appendix A.

Theorem 3.1 *Let the compression function $h : \{0,1\}^n \times \{0,1\}^b \to \{0,1\}^n$ be $(t_{\mathcal{A}}, t_{\mathcal{S}}, Q, N, \mu)$-non-malleable with respect to distribution $\mathcal{X}_{\mathsf{NMAC}}$ and relation R_{NMAC}, and $(t_{\mathcal{D}}, t_{\mathsf{Sim}}, Q, \sigma)$-0-simulatable for $\mathcal{X}_{\mathsf{NMAC}}$. Assume further that h is $(t_{\mathcal{S}}, N, \rho)$-unpredictable. Then NMAC is a (t', Q', L', ϵ')-unforgeable MAC where*

$$t' = \min\{t_{\mathcal{A}}, t_{\mathcal{S}}, t_{\mathsf{Sim}}\} - (Q+1) \cdot \mathit{Time}_{bN}(\mathsf{pad}) - \Theta(NQ \log Q \cdot (\mathit{Time}(h) + n + b))$$
$$Q' = Q, \qquad L' = bN - \mathit{Extend}_{bN}(\mathsf{pad}), \qquad \epsilon' \leq 4QN^2 \cdot (\rho + \mu) + \sigma.$$

Proof. We first make two simplifying assumptions about the MAC adversary $\mathcal{B}$. First we assume that $\mathcal{B}$ always pads any message before outputting it or submitting it to the MAC

oracle. Since the padding function is one-to-one, the padded forgery attempt is still distinct from all (padded) submissions. Furthermore, the adversary's running time only increases by $(Q+1)\cdot \mathrm{Time}_L(\mathsf{pad})$ and the length by at most $\mathrm{Extend}_L(\mathsf{pad})$ bits for each message. Secondly, we presume that $\mathcal{B}$ never submits the same message twice. Since the MAC computation is deterministic such queries can be easily answered by keeping track of previous queries and these checks only add the running time $O(bNQ \log Q)$.

For a successful attacker $\mathcal{B}$ on the MAC we distinguish between two cases:

Case NoColl: In the final output the MAC adversary $\mathcal{B}$ returns a (now padded) message M^* such that $h^*(k_{\mathrm{in}}, M^*) \neq h^*(k_{\mathrm{in}}, M_i)$ for all previous queries M_i, $i = 1, 2, \ldots, Q$.

Case Coll: The MAC adverary $\mathcal{B}$ returns a forgery attempt M^* such that $h^*(k_{\mathrm{in}}, M^*) = h^*(k_{\mathrm{in}}, M_i)$ for some i.

Adding (the bounds for) the two probabilities then gives an upper bound on $\mathcal{B}$'s success probability.

Case NoColl. Assume $\mathcal{B}$ succeeds and that the first case happens. Consider the following adversary $\mathcal{A}_{\mathsf{out}}$ against the non-malleability for distribution $\mathcal{X}_{\mathsf{NMAC}}$ and relation R_{NMAC}. Adversary $\mathcal{A}_{\mathsf{out}}$ initially chooses a key $k_{\mathrm{in}} \leftarrow \{0,1\}^n$ at random. It next runs a simulation of $\mathcal{B}$ and answers each query M_i to the MAC oracle by computing locally $z_i = h^*(k_{\mathrm{in}}, M_i)$ and submitting $p_i = \mathsf{pad}(z_i)$ to its oracle $\mathsf{GenSample}$. Adversary $\mathcal{A}_{\mathsf{out}}$ sets $\tau_i = y_i$ for the oracle's answer and returns τ_i to $\mathcal{B}$. When $\mathcal{B}$ eventually produces its output (M^*, τ^*) attacker $\mathcal{A}_{\mathsf{out}}$ computes $z^* = h^*(k_{\mathrm{in}}, M^*)$ and prepares the transformation $T(k, p_1, p_2, \ldots)$ which computes $h^*(k, \mathsf{pad}(z^*))$ for fixed value $\mathsf{pad}(z^*)$. $\mathcal{A}_{\mathsf{out}}$ finally outputs (T, y^*) for $y^* = \tau^*$.

It is easy to see that $\mathcal{A}_{\mathsf{out}}$'s success probability in the non-malleability experiment equals the success probability of $\mathcal{B}$ attacking the MAC scheme, given that $h^*(k_{\mathrm{in}}, M^*) \neq h^*(k_{\mathrm{in}}, M_i)$ for all i. The latter implies that $z^* \neq z_1, z_2, \ldots$ and therefore $(x^*, y^*) \neq (x_i, y_i)$ for all i in the non-malleability experiment. Furthermore, the padding function is one-to-one and appends only the same amount of bits for each $z^*, z_1, z_2, \ldots$ such that no p_i is a prefix of $\mathsf{pad}(z^*)$. Hence, by assumption, there exists a simulator $\mathcal{S}$ making $t_{\mathcal{S}}$ steps and which is μ-close to $\mathcal{A}_{\mathsf{out}}$'s probability, but which does not get to see any image under $h^*(k_{\mathrm{out}}, \cdot)$. In particular, without any knowledge about k_{out}, the simulator outputs an image y^* and a transformation T involving a constant $c \in B^{\leq N}$ such that $h^*(k_{\mathrm{out}}, c) = y^*$. But since the compression function is $(t_{\mathcal{S}}, N, \rho)$-unpredictable, the claim for this case follows (in particular, $\epsilon' \leq \mu + \rho$).

Case Coll. Now consider the second case, that $\mathcal{B}$ succeeds and finds a collision in the inner function. In fact, we only need that $\mathcal{B}$ is able to generate such a collision $M^* \neq M_1, M_2, \ldots$, possibly not even succeeding in forging a MAC. We then construct an attacker $\mathcal{A}_{\mathsf{in}}$ on the non-malleability of the compression function, again with respect to distribution $\mathcal{X}_{\mathsf{NMAC}}$ and relation R_{NMAC}. The idea is that such a collision can be guessed in advance and can then be used to predict the image for the second value.

But first we assume that, instead of communicating with oracle $\mathsf{NMAC}_{(k_{\mathrm{in}}, k_{\mathrm{out}})}$, adversary $\mathcal{B}$ instead receives the answers from oracle $\mathsf{SimAnswer}(\pi(h^*(k_{\mathrm{in}}, \cdot))) = \mathsf{SimAnswer}(0)$ guaranteed by the π-simulatability (where π is constantly 0). We claim that the probability of $\mathcal{B}$ producing a collision for the inner function cannot drop by more than σ when communicating with $\mathsf{SimAnswer}$. Else, one can easily devise a distinguisher $\mathcal{D}$ separating $\mathsf{GenSample}$ and $\mathsf{SimAnswer}$.

More formally, distinguisher $\mathcal{D}$ picks k_{in} itself and is given access to an oracle either implementing $h^*(k_{\text{out}}, \cdot)$ or SimAnswer and runs a simulation of $\mathcal{B}$. Each query M_i of $\mathcal{B}$ is answered by first computing $z_i = \mathsf{pad}(h^*(k_{\text{in}}, M_i))$. Then $\mathcal{D}$ checks if this value has appeared before, in which case $\mathcal{D}$ fetches the previous answer and handing it back to $\mathcal{B}$. Else, $\mathcal{D}$ forwards z_i to its oracle and returns the answer to $\mathcal{B}$. When $\mathcal{B}$ eventually outputs a forgery attempt (M^*, τ^*) the distinguisher verifies that $M^* \neq M_1, M_2, \ldots$ (if not, it outputs 0), computes $z^* = \mathsf{pad}(h^*(k_{\text{in}}, M^*))$ and checks that there is a collision between z^* and some z_i. If so, then $\mathcal{D}$ outputs 1, in any other case it returns 0.

The probability of returning 1 when given access to oracle GenSample is identical to the probability that $\mathcal{B}$ generates a collision on the inner function between M^* and some (distinct) M_i when attacking NMAC. On the other hand, if $\mathcal{D}$ is given access to SimAnswer then the probability for returning 1 corresponds exactly to the probability that $\mathcal{B}$ produces such a collision when given access to SimAnswer instead. By assumption this difference cannot be more than σ, taking into account that $\mathcal{D}$ essentially runs in the same time as $\mathcal{B}$ but needs to compute the inner function and check for collisions.

Given $\mathcal{B}$ with access to SimAnswer we now build adversary $\mathcal{A}_{\mathsf{in}}$ playing against the non-malleability of the inner function. Adversary $\mathcal{A}_{\mathsf{in}}$ is granted access to oracle GenSample. It first picks random indices i_0 between 1 and Q as well as j_0, ℓ_0 between 0 and $N-1$. It also flips a coin $c \leftarrow \{0,1\}$. Then it runs a black-box simulation of $\mathcal{B}$ for oracle SimAnswer (which works independent of the actual content of the queries of $\mathcal{B}$). Only for the i_0-th query $M_{i_0} = M_{i_0}[1] \ldots M_{i_0}[n_{i_0}]$, if $c = 0$, adversary $\mathcal{A}_{\mathsf{in}}$ also forwards $p_{i_0} = M_{i_0}[1] \ldots M_{i_0}[j_0 + 1]$ to its oracle GenSample to receive a value $z'_{i_0} = h^*(k_{\text{in}}, M_{i_0}[1] \ldots M_{i_0}[j_0 + 1])$.[2] Note that $\mathcal{A}_{\mathsf{in}}$ does not forward this value to $\mathcal{B}$ but rather uses the value generated by SimAnswer. If $c = 1$ then $\mathcal{A}$ does not call its oracle at this point.

When $\mathcal{B}$ finally outputs (M^*, τ^*) and we have $c = 1$ then $\mathcal{A}_{\mathsf{in}}$ has not queried oracle GenSample so far, and now submits $p^* = M^*[1] \ldots M^*[\ell_0 + 1]$ to receive the value $z' = h^*(k_{\text{in}}, M^*[1] \ldots M^*[\ell_0 + 1])$.[3] Adversary $\mathcal{A}_{\mathsf{in}}$ then returns $y^* = z'$ and the transformation $T(k_{\text{in}}, , p_1, p_2, \ldots)$ which for fixed $j_0, M_{i_0}[1], \ldots, M_{i_0}[j_0 + 1]$ computes the value $x^* = (h^*(k_{\text{in}}, M_{i_0}[1] \ldots M_{i_0}[j_0]), M_{i_0}[j_0 + 1])$. Else, if $c = 0$, then $\mathcal{A}_{\mathsf{in}}$ returns $y^* = z'_{i_0}$ and the transformation $T(k_{\text{in}}, , p_1, p_2, \ldots)$ which for fixed $\ell_0, M^*[1], \ldots, M^*[\ell_0 + 1]$ computes $x^* = (h^*(k_{\text{in}}, M^*[1] \ldots M^*[\ell_0]), M^*[\ell_0 + 1])$.

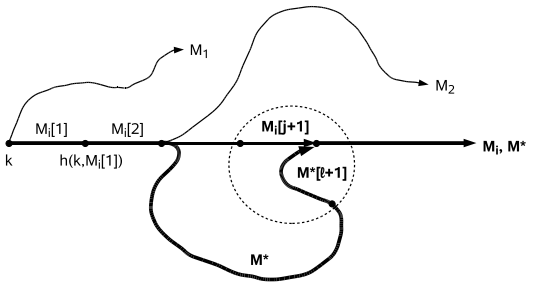

Figure 1: Proof idea to Theorem 3.1: collisions among values

For the success probability note that, given that $\mathcal{B}$ creates a collision between some message M_i and M^*, adversary $\mathcal{A}_{\mathsf{in}}$ predicts $i_0 = i$ and the right block numbers j_0, ℓ_0

[2] We assume that $j_0 < n_{i_0}$; else stop immediately with no output.

[3] Again, if ℓ_0 exceeds the number of blocks of M^* we stop with no output.

such that $M_{i_0}[1]\dots M_{i_0}[j_0+1] \neq M^*[1]\dots M^*[\ell_0+1]$ collide under $h^*(k_{\text{in}},\cdot)$ but $x_i = h^*(k_{\text{in}}, M_i[1]\dots M_i[j_0]) \neq x^* = h^*(k_{\text{in}}, M^*[1]\dots M^*[\ell_0])$, with probability at least $1/QN^2$. See Figure 1. Furthermore, either M_{i_0} is a prefix of M^* or vice versa (or neither one is a prefix of the other one), and we make the "right" choice c with probability at least $1/2$ to submit the message which is not a prefix to oracle $\mathsf{GenSample}$. Hence, $\mathcal{A}_{\mathsf{in}}$'s success probability is only a factor $1/2QN^2$ smaller than the one of $\mathcal{B}$ in this case. The overall running time of $\mathcal{A}_{\mathsf{in}}$ is essentially equal to the one of $\mathcal{B}$, plus some time to prepare the submissions and the final output.

By the non-malleability there must exist a simulator with μ-close success rate to $\mathcal{A}_{\mathsf{in}}$. But this simulator never gets to see any information about k_{in} and must first commit to y^* and $M^* = M^*[1]\dots M^*[n^*]$ for $n^* \leq N$ (via T), allowing only a success probability ρ by the unpredictability of the compression function. Hence, the probability of $\mathcal{B}$ succeeding for a collision on the inner function is at most $2QN^2(\mu+\rho)+\sigma$. □

Compared to the proof in [Bel06], showing that NMAC is pseuodorandom given that h is pseudorandom, we obtain a different loss factor in the success probability, switching the roles of the number of queries and the length of messages ($\Theta(QN^2)$ vs. $\Theta(Q^2N)$ as in [Bel06]).

In the proof we have assumed that $\pi(\cdot)$ is the constant 0-function. We note that the theorem still holds for more general cases, as long as there is a function Π such that $\pi(h^*(k,M)) = \Pi(M)$ for all k, M. Recall the idea from the second part of the proof that we can simulate images for the outer function without knowing $h^*(k_{\text{in}}, M_i)$. Yet, in order to execute $\mathsf{SimAnswer}(\pi(h^*(k_{\text{in}},\cdot)))$ we need the information $\pi(h^*(k_{\text{in}}, M_i))$. This information is easy to derive for constant π, but also if we can deduce from M_i (known to us) via Π. Hence, the proof goes through and we obtain:

Corollary 3.2 *Let the compression function $h : \{0,1\}^n \times \{0,1\}^b \to \{0,1\}^n$ be $(t_{\mathcal{A}}, t_{\mathcal{S}}, Q, N, \mu)$-non-malleable with respect to distribution $\mathcal{X}_{\mathsf{NMAC}}$ and relation R_{NMAC}, and $(t_{\mathcal{D}}, t_{\mathsf{Sim}}, Q, \sigma)$-$\pi$-simulatable for $\mathcal{X}_{\mathsf{NMAC}}$, such that there exists a function Π with $\pi(h^*(k,M)) = \Pi(M)$ for all $k \in \{0,1\}^n$ and all $M \in B^+$. Assume further that h is $(t_{\mathcal{S}}, N, \rho)$-unpredictable. Then NMAC is a (t', Q', L', ϵ')-unforgeable MAC where*

$$t' = \min\{t_{\mathcal{A}}, t_{\mathcal{S}}, t_{\mathsf{Sim}}\} - \Theta(NQ\log Q \cdot (\mathit{Time}(h) + n + b + \mathit{Time}_{bN}(\mathsf{pad}) + \mathit{Time}_{bN}(\Pi)))$$
$$Q' = Q, \qquad L' = bN - \mathit{Extend}_{bN}(\mathsf{pad}), \qquad \epsilon' \leq 4QN^2 \cdot (\rho+\mu) + \sigma.$$

As an example for such a function π consider a 0-simulatable compression function which appends some (fixed) subset of its input M to the output, and let π denote the projection onto the corresponding suffix. Then this information is clearly computable from $\Pi(M)$. Somewhat interestingly, the simulatability requirement depends to the padding function of the hash function. Namely, in Appendix A we show that for prefix-free paddings the simulatability requirement vanishes completely.

4 Security of HMAC

As explained in [BCK96a], one can map HMAC to NMAC by considering $k_{\text{in}}^{\mathsf{NMAC}} = h(\text{IV}, k_{\text{in}}^{\mathsf{HMAC}})$ and $k_{\text{out}}^{\mathsf{NMAC}} = h(\text{IV}, k_{\text{out}}^{\mathsf{HMAC}})$ and

$$\mathsf{HMAC}(k_{\text{in}}^{\mathsf{HMAC}} || k_{\text{out}}^{\mathsf{HMAC}}, M) = \mathsf{NMAC}(k_{\text{in}}^{\mathsf{NMAC}} || k_{\text{out}}^{\mathsf{NMAC}}, M).$$

Put differently, HMAC includes a first step in which the NMAC-keys are computed via $h(\text{IV},\cdot)$ and the HMAC-keys are used as input block.

To transfer the security claims from NMAC, Bellare [Bel06] defines the "dual" compression function $\overline{h} : B \times \{0,1\}^n \to \{0,1\}^n$ with $\overline{h}(b,a) = h(a,b)$. Then, if h and $\overline{h}$ are pseudorandom, the security of NMAC carries over to HMAC. For the precise claim and a discussion about the validity see [Bel06, Sections 5.1 and 5.4].

We can extend the notions of non-malleability, π-simulatability and unpredictability to these special cases. For this consider distribution $\mathcal{X}_{\mathsf{HMAC}}$ which, for input (k,p) computes $k' = h(\mathrm{IV}, k)$ and samples $x \leftarrow \mathcal{X}_{\mathsf{NMAC}}(k', p)$. Analogously, relation R_{HMAC} merely checks that the transformation T in the first step computes $k' = h(\mathrm{IV}, k)$ and proceeds as R_{NMAC}. Then we demand that h is non-malleable with respect to $\mathcal{X}_{\mathsf{HMAC}}$ and R_{HMAC} and π-simulatable for $\mathcal{X}_{\mathsf{HMAC}}$.

Similarly, we say that h is HMAC-unpredictable if it is infeasible to predict $h(h(\mathrm{IV}, k), M)$; a more formal characeterization is easy to deduce. All assumptions are implied if h and $\overline{h}$ are pseudorandom (where we only need that $\overline{h}$ is pseudorandom with respect to distinguishers that make only a single oracle call). Under the assumptions about non-malleability, π-simulatability for suitable π and unpredictability we conclude:

Theorem 4.1 *Let the compression function $h : \{0,1\}^n \times \{0,1\}^b \to \{0,1\}^n$ be $(t_{\mathcal{A}}, t_{\mathcal{S}}, Q, N, \mu)$-non-malleable with respect to distribution $\mathcal{X}_{\mathsf{HMAC}}$ and relation R_{HMAC}, and $(t_{\mathcal{D}}, t_{\mathsf{Sim}}, Q, \sigma)$-$\pi$-simulatable for $\mathcal{X}_{\mathsf{HMAC}}$, such that there exists a function Π with $\pi(h^*(k,M)) = \Pi(M)$ for all $k \in \{0,1\}^n$ and all $M \in B^+$. Assume further that h is $(t_{\mathcal{S}}, N, \rho)$-HMAC-unpredictable. Then HMAC is a (t', Q', L', ϵ')-unforgeable MAC where*

$$t' = \min\{t_{\mathcal{A}}, t_{\mathcal{S}}, t_{\mathsf{Sim}}\} - \Theta(NQ \log Q \cdot (\mathit{Time}(h) + n + b + \mathit{Time}_{bN}(\mathsf{pad}) + \mathit{Time}_{bN}(\Pi)))$$
$$Q' = Q, \qquad L' = bN - \mathit{Extend}_{bN}(\mathsf{pad}), \qquad \epsilon' \leq 4QN^2 \cdot (\rho + \mu) + \sigma.$$

For the single-keyed HMAC variant with $k_{\mathrm{in}} = k \oplus \mathrm{ipad}$ and $k_{\mathrm{out}} = k \oplus \mathrm{opad}$ one can in principle adapt the notions of non-malleability, simulatability and unpredictability for this case, too. Both properties are then implied if $\overline{h}$ is pseudorandom under related-key attacks [BK03] (see also [Bel06, Section 5.3]), and single-keyed HMAC is a secure MAC under these versions of non-malleability, simulatability and unpredictability.

5 Relations among Security Notions

In this section we show that pseudorandom compression function have our two properties but are stronger than both properties together. We also discuss that there are (simulatable) non-malleable compression functions which are not privacy-preserving according to the notion in [Bel06]. Below we simply write $\$_{N,n}$ for a randomly chosen function from all mappings with domain $B^{\leq N}$ and range $\{0,1\}^n$.

Definition 5.1 *A function $f : \{0,1\}^n \times B^+ \to \{0,1\}^n$ is called (t, Q, N, δ)-pseudorandom if, for any algorithm $\mathcal{D}$ running in time t, we have*

$$\mathrm{Prob}\left[\mathcal{D}^{f(k,\cdot)} = 1\right] - \mathrm{Prob}\left[\mathcal{D}^{\$_{N,n}(\cdot)} = 1\right] \leq \delta$$

where the probability in the first case is over $\mathcal{D}$'s coin tosses and $k \leftarrow \{0,1\}^n$, and in the second case over $\mathcal{D}$'s coin tosses and the choice of $\$_{N,n}$. In both cases $\mathcal{D}$ makes at most Q queries to its function oracle, each query of at most N blocks.

Below we sometimes make use of the following fact of Bellare et al. [BCK96b] about cascaded iteration of pseudorandom functions. The statement basically says that the cascaded evaluation is also pseudorandom with respect to *prefix-free distinguishers*, i.e., distinguishers such that no oracle query is a prefix of another one:

Lemma 5.2 ([BCK96b]) *If the compression function $h : \{0,1\}^n \times \{0,1\}^b \to \{0,1\}^n$ is $(t, Q, 1, \delta)$-pseudorandom, then $h^* : \{0,1\}^n \times B^+ \to \{0,1\}^n$ is also (t', Q, N, δ')-pseudorandom with respect to prefix-free distinguishers, where $t' = t - \Theta(QNb + N\,Time(h))$ and $\delta' = N\delta$.*

5.1 Pseudorandom ⇒ Non-Malleable ∧ Unpredictable ∧ Simulatable

Recall from the security of NMAC that we defined $\mathcal{X}_{\mathsf{NMAC}}$ to be the distribution which, for random $k \in \{0,1\}^n$ and parameter $p = M[1] \ldots M[j+1] \in B^+$ outputs the preimage $x = (h^*(k, M[1] \ldots M[j]), M[j+1])$. Also, R_{NMAC} is the relation which, on input $T, k, p_1, p_2, \ldots, x^*$ checks that T computes $x^* = (h^*(k, M^*[1] \ldots M^*[j]), M^*[j+1])$ for some fixed $M^*[1] \ldots M^*[j+1]$ of at most N blocks (and checks that no p_i is a prefix of $M^*[1] \ldots M^*[j+1]$).

Proposition 5.3 *Let h be $(t, Q, 1, \delta)$-pseudorandom. Then it is also $(t_{\mathcal{A}}, t_{\mathcal{S}}, Q, N, \mu)$-non-malleable with respect to distribution $\mathcal{X}_{NMAC}$ and relation R_{NMAC}, where*

$$t_{\mathcal{A}} = t - \Theta(QNb \log Q \cdot (Time(h) + n)), \quad t_{\mathcal{S}} = t + O(bNQ \log Q + n), \quad \mu = 2N\delta.$$

Proof. Consider any adversary $\mathcal{A}$ against non-malleability, running in time $t_{\mathcal{A}}$ and making at most Q queries. We may assume that $\mathcal{A}$ never queries its oracle GenSample about a parameter whose prefix has been submitted before; such values could be computed by $\mathcal{A}$ itself easily and skipping these oracle queries can only increase $\mathcal{A}$'s success probability. This increases the adversary's running time by at most $O(bNQ \log Q \cdot (\mathrm{Time}(h) + n))$.

Construct the non-malleability simulator $\mathcal{S}$ running a black-box simulation of $\mathcal{A}$ as follows. Each time the adversary submits a parameter $p_i = M_i[1] \ldots M_i[j+1]$ to its oracle, $\mathcal{S}$ emulates the oracle perfectly except that, instead of using the iterated compression function, $\mathcal{S}$ uses lazy sampling to simulate a truly random function.[4] When the adversary $\mathcal{A}$ eventually outputs (T, y^*) the simulator, too, stops with this output.

For the analysis we first consider $\mathcal{A}$'s behavior if, instead of giving it access to $h^*(k, \cdot)$ we use a truly random function $\$_{N,n}$, also to check that $y^* = \$_{N,n}(M^*)$ for the final output (instead of verifying $y^* = h^*(k, M^*)$). By the pseudorandomness of h (and therefore of h^*) we get that the probability of $\mathcal{A}$ winning in this new experiment is at least $N\delta$-close to the original success probability. This can be easily turned formally into a (prefix-free[5]) distinguisher, simulating $\mathcal{A}$ in a black-box way and calling its function oracle for each message and the final check that T is of the right form and that y^* is an image for M^*.

Next, consider the slightly changed experiment in which we give $\mathcal{A}$ random answers for its message queries as before, but then evaluate correctly $h^*(k, M^*)$ to compare it to y^*. It is easy to see that the success probability of $\mathcal{A}$ in this experiment cannot grow more than $N\delta$, again by the pseudorandomness. That is, it is once more straightforward to construct a prefix-free distinguisher turning this into a formal statement.

[4] Meaning that $\mathcal{S}$ picks an independent random string when supposed to evaluate the function on a new value, or repeats a previously given answer for previously evaluated values.

[5] Here we use the fact that no p_i is a prefix of the message encoded in T.

But the final experiment is identical to the success probability of the simulator, showing the claim. □

The proposition above shows that the same remains true for π-simulatable compression functions (for arbitrary π) if we let $\mathsf{SimAnswer}$ simply return random strings.

Corollary 5.4 *Let h be $(t, Q, 1, \delta)$-pseudorandom. Then it is $(t_{\mathcal{A}}, t_{\mathsf{Sim}}, Q, N, \mu)$-$\pi$-simulatable with respect to distribution $\mathcal{X}_{\mathsf{NMAC}}$, where π is arbitrary and*

$$t_{\mathcal{A}} = t - \Theta(QNb \log Q \cdot (\mathit{Time}(h) + n)), \quad t_{\mathsf{Sim}} = t + O(bNQ \log Q + n), \quad \mu = N\delta.$$

Finally, we show that pseudrandomness implies unpredictability:

Proposition 5.5 *Let h be $(t, 1, 1, \delta)$-pseudorandom. Then it is (t', N, ρ)-unpredictable, where $t' = t - \Theta(QNb + N \cdot \mathit{Time}(h))$ and $\rho = N(\delta + 2^{-n})$.*

Proof. Since h is pseudorandom it remains pseudorandom if we make a single query of (at most) N blocks. Only the running time drops slightly and the distinguishing advantage increases to $N\delta$. But this implies that any algorithm P trying to predict a function value cannot be better than for a truly random function —which can be predicted with probability at most $N \cdot 2^{-n}$— plus the distinguishing advantage of $N\delta$. □

5.2 Non-Malleable ∧ Unpredictable ∧ Simulatable ⇏ Pseudorandom

We next prove that non-malleability, simulatability and unpredictability together do not imply pseudorandomness. Since this result only serves as a separation we drop the viewpoint of concrete security and adopt the usual "asymptotic" notion (with regard to parameter n). For example, being pseudorandom means to be $(\mathrm{poly}(n), \mathrm{poly}(n), \mathrm{poly}(n), \delta(n))$-pseudorandom for any polynomial $\mathrm{poly}(n)$ and some negligible function $\delta(n)$ (which may depend on the polynomial). The other security notions can be adopted analogously.

Below we show that pseudorandomness is stronger than non-malleability, unpredictability and π-simulatability for $\pi = 0$. Afterwards we discuss that the claim also holds for a non-trivial functions π like the rightmost-bit function rmb.

Proposition 5.6 *Assume that there exists a pseudorandom compression function $h : \{0,1\}^n \times \{0,1\}^b \to \{0,1\}^n$ and a pseudorandom generator $G : \{0,1\}^{\lfloor n/2 \rfloor} \to \{0,1\}^n$. Then there exists a compression function $h_{\mathsf{sep}} : \{0,1\}^n \times \{0,1\}^b \to \{0,1\}^n$ which is (a) non-malleable with respect to distribution $\mathcal{X}_{\mathsf{NMAC}}$ and relation R_{NMAC}, (b) unpredictable, (c) 0-simulatable for $\mathcal{X}_{\mathsf{NMAC}}$, but (c) not pseudorandom.*

Proof. First consider the truncated compression function $h_{\mathsf{trunc}}(k, x)$ which splits the key k into $\lfloor n/2 \rfloor$ bits k_L and the remaining $n - \lfloor n/2 \rfloor$ bits k_R, then computes $h(G(k_L), x)$ and outputs only the first $\lfloor n/2 \rfloor$ bits of the result. It follows easily from the pseudorandomness of G and h that this compression function is pseudorandom, too, and therefore non-malleable, unpredictable and 0-simulatable.

Now define $h_{\mathsf{sep}}(k, x) = h_{\mathsf{trunc}}(k, x) || 0^{n - \lfloor n/2 \rfloor}$. With this definition one can compute the iterated output $h^*_{\mathsf{sep}}(k, M)$ for any $M \in B^+$ by evaluating h_{trunc} and appending $n - \lfloor n/2 \rfloor$ bits $\pi(x)$ to the output in each stage (including the final evaluation).

It is clear that h_{sep} is not pseudorandom, because every function evaluation yields only zeros in the right half of output. It is, however, still unpredictable. If it was not unpredictable

this would easily contradict unpredictability of h_{trunc} by cutting off the $n - \lfloor n/2 \rfloor$ rightmost bits from the output. Moreover, h_{sep} inherits non-malleability for $\mathcal{X}_{\mathsf{NMAC}}$ and R_{NMAC} from h_{trunc} because one can easily transform attackers and simulators for the two cases (by appending or cutting off zeros in the output). Simulatability can be shown easily, too, as one can output random strings followed by a sequence of zeros. □

Again, the claim remains true for π-simulatability for some non-trivial functions π. For instance, if $\pi(x)$ returns the rightmost bit rmb(x) of x and we use the compression function h_{sep} in the proof and replace the padding with zeros by $(\mathrm{rmb}(x))^{n-\lfloor n/2 \rfloor}$ —call this function h_{rmb}— then the proposition still holds. In particular, the rmb-simulatability (together with non-malleability and unpredictability) would suffice to prove NMAC to be secure.

We also remark that the example h_{rmb} shows that there are π-simulatable (and non-malleable and unpredictable) compression functions which are *not* a privacy-preserving MAC according to [Bel06]. Such a MAC has the additional property that one cannot tell if either of two messages has been MACed. However, for inputs with distinct rightmost bit this is easy for h_{rmb}, of course. Still, 0-simulatable compression functions are privacy-preserving, giving an alternative characterization of this notion.

Acknowledgments

We thank the anonymous reviewers for valuable comments.

References

[AB99] Jee Hea An and Mihir Bellare. *Constructing VIL-MACs from FIL-MACs: Message Authentication under Weakened Assumptions*. Advances in Cryptology — Crypto'99, Volume 1666 of Lecture Notes in Computer Science, pages 252–269. Springer-Verlag, 1999.

[BCFW07] Alexandra Boldyreva, David Cash, Marc Fischlin, and Bogdan Warinschi. *Non-Malleable Hash Functions*. manuscript, 2007.

[BCK96a] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. *Keying Hash Functions for Message Authentication*. Advances in Cryptology — Crypto'96, Volume 1109 of Lecture Notes in Computer Science, pages 1–15. Springer-Verlag, 1996.

[BCK96b] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. *Pseudorandom Functions Revisited: The Cascade Construction and Its Concrete Security*. Proceedings of the Annual Symposium on Foundations of Computer Science (FOCS)'96, pages 514–523. IEEE Computer Society Press, 1996.

[Bel06] Mihir Bellare. *New Proofs for NMAC and HMAC: Security Without Collision-Resistance*. Advances in Cryptology — Crypto 2006, Volume 4117 of Lecture Notes in Computer Science, pages 602–619. Springer-Verlag, 2006.

[BGM04] Mihir Bellare, Oded Goldreich, and Anton Mityagin. *The Power of Verification Queries in Message Authentication and Authenticated Encryption*. Number 2004/309 in Cryptology eprint archive. `eprint.iacr.org`, 2004.

[BK03] Mihir Bellare and Tadayoshi Kohno. *A Theoretical Treatment of Related-Key Attacks: RKA-PRPs, RKA-PRFs, and Applications.* Advances in Cryptology — Eurocrypt 2003, Volume 2656 of Lecture Notes in Computer Science, pages 491–506. Springer-Verlag, 2003.

[CY06] Scott Contini and Yiqun Lisa Yin. *Forgery and Partial Key-Recovery Attacks on HMAC and NMAC Using Hash Collisions.* Advances in Cryptology — Asiacrypt 2006, Volume 4284 of Lecture Notes in Computer Science, pages 37–53. Springer-Verlag, 2006.

[KBPH06] Jongsung Kim, Alex Biryukov, Bart Preneel, and Seokhie Hong. *On the Security of HMAC and NMAC Based on HAVAL, MD4, MD5, SHA-0 and SHA-1.* Security in Communication Networks (SCN), Volume 4116 of Lecture Notes in Computer Science, pages 242–256. Springer-Verlag, 2006.

[RR07] Christian Rechberger and Vincent Rijmen. *On Authentication With HMAC and Non-Rondom Properties.* Financial Cryptography (FC) 2007, Lecture Notes in Computer Science. Springer-Verlag, 2007.

[WY05] Xiaoyun Wang and Hongbo Yu. *How to Break MD5 and Other Hash Functions.* Advances in Cryptology — Eurocrypt 2005, Volume 3494 of Lecture Notes in Computer Science, pages 19–35. Springer-Verlag, 2005.

[WYY05] Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu. *Finding Collisions in the Full SHA-1.* Advances in Cryptology — Crypto 2005, Volume 3621 of Lecture Notes in Computer Science, pages 17–36. Springer-Verlag, 2005.

A Security of NMAC for Prefix-Free Paddings

Here we show that regular (i.e., not necessarily black-box) non-malleability suffices to show security of NMAC and HMAC, given that the padding function pad() is prefix-free. Formally, the function pad() is called *prefix-free* if for any distinct $M \neq M'$ the value $\mathsf{pad}(M)$ is not equal to any block-wise prefix of $\mathsf{pad}(M')$. An example of a prefix-free padding is the standard padding with the exception that we prepend a block containg the bit size of the original input. Then, any messages with distinct length are clearly not a prefix of each other, and for equal length messages $\mathsf{pad}(M)$ is either longer than any block-wise prefix of $\mathsf{pad}(M')$ or of equal length, in which case they must be distinct since $M \neq M'$.

Theorem A.1 *Let the compression function $h : \{0,1\}^n \times \{0,1\}^b \to \{0,1\}^n$ be $(t_{\mathcal{A}}, t_{\mathcal{S}}, Q, N, \mu)$-non-malleable with respect to distribution $\mathcal{X}_{\mathsf{NMAC}}$ and relation R_{NMAC}. Assume further that h is $(t_{\mathcal{S}}, N, \rho)$-unpredictable and that pad is prefix-free. Then NMAC is a (t', Q', L', ϵ')-unforgeable MAC where*

$$t' = \min\{t_{\mathcal{A}}, t_{\mathcal{S}}\} - (Q+1) \cdot \mathit{Time}_{bN}(\mathsf{pad}) - \Theta(NQ \log Q \cdot (\mathit{Time}(h) + n + b))$$
$$Q' = Q, \qquad L' = bN - \mathit{Extend}_{bN}(\mathsf{pad}), \qquad \epsilon' \leq 2QN^2 \cdot (\rho + \mu).$$

Proof. We remark that we again presume that $\mathcal{B}$ pads each message first and that it never queries the MAC oracle about the same message twice. The first part of the proof (the case that there is no collision between the inner value for M^* and any M_i) is identical to the "non-prefix-free" case and is therefore omitted. We next consider the second case that $\mathcal{B}$

causes $h^*(k_{\text{in}}, M_i) = h^*(k_{\text{in}}, M_i)$ for some i. We again construct a non-malleability attacker $\mathcal{A}_{\mathsf{in}}$ against the inner function.

Initially, adversary $\mathcal{A}_{\mathsf{in}}$ picks a random index i_0 between 1 and Q and indices j_0, ℓ_0 between 0 and $N-1$. It next invokes the MAC adversary $\mathcal{B}$ and simulates each answer for query M_i as follows.

- For $i \neq i_0$ adversary $\mathcal{A}_{\mathsf{in}}$ first submits $p_i = M_i$ to its oracle $\mathsf{GenSample}$ to get an answer $z_i = y_i$. Then it computes $\tau_i = h^*(k_{\text{out}}, \mathsf{pad}(z_i))$ and returns this value to $\mathcal{B}$.
- For $i = i_0$ and message $M_{i_0} = M_{i_0}[1] \dots M_{i_0}[n_{i_0}]$ it sends $p_{i_0} = M_{i_0}[1] \dots M_{i_0}[j_0+1]$ to the oracle[6] to receive a value $z'_{i_0} = h^*(k_{\text{in}}, M_{i_0}[1] \dots M_{i_0}[j_0+1])$. Compute $z_{i_0} = h(z'_{i_0}, M_{i_0}[j_0+2] \dots M_{i_0}[n_{i_0}])$ and $\tau_{i_0} = h^*(k_{\text{out}}, \mathsf{pad}(z_{i_0}))$ and return the latter value to $\mathcal{B}$.

When $\mathcal{B}$ stops with output (M^*, τ^*) (for the padded message M^*) adversary $\mathcal{A}_{\mathsf{in}}$ sets its output to $y^* = z'_{i_0}$ for the answer of the previously guessed index i_0. The adversary furthermore defines $T(k_{\text{in}}, , p_1, p_2, \dots)$ for fixed $\ell_0, M^*[1], \dots, M^*[\ell_0+1]$ to compute $x^* = (h^*(k_{\text{in}}, M^*[1] \dots M^*[\ell_0]), M^*[\ell_0+1])$.[7]

To analyze $\mathcal{A}_{\mathsf{in}}$'s success probability let ℓ denote some index such that, when computing $h^*(k_{\text{in}}, M^*[1] \dots M^*[\ell+1])$ iteratively, this value matches any of the intermediate values $h(k_{\text{in}}, M_i[1] \dots M_i[j+1])$ for some i, j but such that $M^*[1] \dots M^*[\ell+1] \neq M_i[1] \dots M_i[j+1]$. See again Figure 1 on Page 8. According to the assumption a collision between the value for M^* and an image of some query occurs and such indices must exist.

Furthermore, by the prefix-free padding function pad it follows from $M^* \neq M_1, M_2, \dots$ that no M_i is equal to a block-wise prefix of M^*. Hence, there must exists indices ℓ_0, i_0, j_0 such that we obtain a collision on the inner function and no M_i for $i \neq i_0$ and neither $M_{i_0}[1] \dots M_{i_0}[j_0+1]$ is a prefix of $M^*[1] \dots M^*[\ell_0+1]$. But then $\mathcal{A}_{\mathsf{in}}$ picks the right indices i_0, j_0, ℓ_0 with probability at least $1/QN^2$. Under this condition, $(x^*, y^*) \neq (x_i, y_i)$ for all i and $\mathcal{A}_{\mathsf{in}}$ wins if $\mathcal{B}$ generates a collision for the inner function.

Overall, $\mathcal{A}_{\mathsf{in}}$ has a success probability which is smaller by a factor $1/QN^2$, and there must exist a simulator with μ-close success rate. But this simulator never gets to see any information about k_{in} and must first commit to y^* and $M^* = M^*[1] \dots M^*[n^*]$ for $n^* \leq N$ (via T), allowing only a success probability ρ by the unpredictability of the compression function. The overall running time of $\mathcal{A}_{\mathsf{in}}$ is essentially equal to the one of $\mathcal{B}$, plus the time to pad each message, to check for double queries and to evaluate the compression function for each oracle call. □

[6] We assume that $j_0 < n_{i_0}$; else stop immediately with no output.

[7] We assume that $\ell_0 + 1$ is at most the number of blocks in M^*; else stop immediately with no output.