Boosting Merkle-Damgård Hashing for Message Authentication

Kan Yasuda

NTT Information Sharing Platform Laboratories, NTT Corporation
3-9-11 Midoricho Musashino-shi, Tokyo 180-8585 Japan
yasuda.kan@lab.ntt.co.jp

Abstract. This paper presents a novel mode of operation of compression functions, intended for dedicated use as a message authentication code (MAC.) The new approach is faster than the well-known Merkle-Damgård iteration; more precisely, it is $(1 + c/b)$-times as fast as the classical Merkle-Damgård hashing when applied to a compression function $h : \{0,1\}^{c+b} \rightarrow \{0,1\}^c$. Our construction provides a single-key MAC with provable security; we show that the proposed scheme yields a PRF(pseudo-random function)-based MAC on the assumption that the underlying compression function h satisfies certain PRF properties. Thus our method offers a way to process data more efficiently than the conventional HMAC without losing formal proofs of security. Our design also takes into account usage with prospective compression functions; that is, those compression functions h with relatively weighty load and relatively large c (i.e., "wide-pipe") greatly benefit from the improved performance by our mode of operation.

Key words: Merkle-Damgård, pseudo-random function, related-key attack, message authentication code, hash function, compression function, mode of operation, NMAC, HMAC.

1 Introduction

The Merkle-Damgård iteration [16, 10] is a popular and classical mode of operation for cryptographic hash functions. It is widely used not only for keyless hash functions but also for randomized hash functions, message authentication codes (MACs) and pseudo-random functions (PRFs.) It is popular, widespread and successful in some respects, but nowadays some problems are becoming more and more evident, which initiates investigation into better modes of operation [14, 9].

Inspired by this trend, in this paper we free ourselves from the traditional Merkle-Damgård iteration and devise a novel mode of operation that can be used exclusively as a secure, single-keyed MAC. Our method is the first of its kind that can process a message more efficiently than the conventional Merkle-Damgård iteration and that can be provided with formal proofs of security. More precisely, the proposed scheme is $(1 + c/b)$-times faster than the conservative

Merkle-Damgård hashing (and hence HMAC [2]), when applied to a compression function $h : \{0,1\}^{c+b} \to \{0,1\}^c$. For example, with the compression function $\mathsf{sha256} : \{0,1\}^{256+512} \to \{0,1\}^{256}$ the new method yields a 50% increase in performance as compared to HMAC. As to the security of our new scheme, we obtain results that are similar to the recent ones of NMAC and HMAC [2]; namely, we prove that the proposed mode of operation results in a PRF-based MAC whose security relies on the pseudo-randomness properties of the underlying compression function.

Brief Outline of Our Construction and Its Security. Our construction can be regarded as a derivative of NMAC. Recall that NMAC is based on a nested structure consisting of an inner part of hashing and an outer part of encryption. In our construction we boost up the performance of the inner hashing by introducing a novel method of iteration, where each invocation to the underlying compression function $h : \{0,1\}^{c+b} \to \{0,1\}^c$ processes more input bits. It takes $c + b$ bits of a message, rather than just b bits as in the conventional Merkle-Damgård iteration.

The inner hashing should satisfy a certain form of collision resistance, in order for the nested MAC to be secure. NMAC fulfills this requirement by assuming that the underlying compression function is a PRF [3, 2]. On the other hand, in our construction it turns out that we need to impose an extra condition on the underlying compression function in order to ensure the desired property of the inner hashing. The additional condition is a type of pseudo-randomness in a mild form of related-key setting; in fact, our proofs of security can be viewed as a related-key version of those in [3].

Backgrounds. A motive for this work originates from the recent degradation of existing hash functions such as MD5 and SHA-1. These algorithms are first shown to be vulnerable to collision attacks as keyless hash functions, but the techniques are then extended to forgery and key-recovery attacks against NMAC/HMAC constructed of these hash functions [8, 13]. These attacks tell us that it is high time to move toward new compression functions. In fact, NIST announces ending its support for SHA-1 and recommends migrating to SHA-2 family by the year 2010 [17, 18]. Since SHA-2 family are slower than SHA-1, the replacement would result in lowering performance and losing an advantage of hash-based MACs (as compared to MACs of other types, say block-cipher-based or universal-hash-based ones.) One way to overcome this problem is to use a more efficient mode of operation, absorbing the decrease in performance caused by the new compression function.

Another reason to propose the new mode comes from a security principle of iterated functions that the size c of a chaining variable be relatively large. This requirement is particularly evident for MACs, due to the birthday attack [20] showing that half the size of c of the chaining variable corresponds to a security parameter. Having a large size c of a chaining variable is a good design principle also in the context of keyless hash functions, as illustrated by the "wide-pipe" argument [14]. Such design with large c, unfortunately, results in a performance disadvantage of the traditional Merkle-Damgård iteration. On the other hand, in our approach the size c is irrelevant in terms of efficiency, and indeed large c is welcomed; such large c increases relative performance of our scheme as compared to the conventional Merkle-Damgård iteration.

Organization of This Paper. In the following section we review some of the previous results concerning modes of operation of compression functions and identify the position of this work among them. Section 3 introduces design principles of our approach and a two-key prototype of our MAC construction. In Sect. 4 and 5 we define security notions utilized in this paper and discuss some aspects of them. Section 6 is devoted to security proofs of the two-key construction. In Sect. 7 and 8 we show techniques of constructing a single-key version and those of using a shorter key, respectively. Section 9 summarizes this paper.

2 Related Work

Merkle-Damgård. The Merkle-Damgård iteration gives a way to extend the domain of a compression function, having an attractive property that collision resistance of the compression function extends to the entire hash function (in either a keyless or keyed context) [16, 10]. Owing to standardization and lack of regulation on export control, hash functions such as MD5 and SHA-1 are widely available in software libraries today. The widespread use of these keyless hash functions implemented with the Merkle-Damgård iteration also influences design principles for randomized hash functions and hash-based MACs/PRFs.

Randomized Hash Functions. The question of domain extension of target-collision-resistant (TCR) functions is intensively studied [7, 21], where several modes of operation are suggested, which extend a TCR compression function to TCR hash functions. The common problem of these schemes is that the key size grows as a message length does. This obstacle is resolved in [12], where proposed is a mode of operation that runs as efficiently as the Merkle-Damgård iteration and that requires only a constant-size key. The trick is that its security is based on the assumption that the compression function satisfies new (but reasonable) properties, which are different from the notion of TCR.

MACs and PRFs. The NI and CS constructs [1, 15] provide domain extension of MACs. The problem is that these modes are slower than the Merkle-Damgård iteration. This drawback is absent from HMAC, which achieves the same efficiency as the Merkle-Damgård iteration. This is a natural outcome since HMAC gives domain extension of PRFs, not MACs.[1]

In this paper we push ahead with this idea in order to obtain a PRF via a mode of operation that is even more efficient than HMAC. The trick is that our security result is based on the assumption that the underlying compression

[1] Recall that a PRF is a secure MAC. There is another construct based on a PRF, called XOR-MAC [4]. XOR-MAC is capable of parallel processing, yet without it XOR-MAC is in general slower than the Merkle-Damgård iteration.

function satisfies, in addition to the usual PRF, a new (but reasonable) PRF property (which we call Δ-2PRF.)

Our construction is dedicated to MAC/PRF use. In return, our approach accomplishes higher performance than the Merkle-Damgård iteration, which seems to be hard to realize in the context of keyless or randomized hash functions — we may consider the circumstances as evidence that our mode of operation fully takes advantage of the presence of a "secret" key in the MAC/PRF situation. See Table 1 for comparison of these MAC/PRF modes.

Table 1. Comparison of modes of operation for MAC/PRF

	Performance	Goal	Assumptions[2]	Reference
NI / CS	$<$ Merkle-Damgård	MAC	MAC	[1, 15]
NMAC / HMAC	$=$ Merkle-Damgård	MAC	pp-MAC, 2PRF	[3, 2]
		PRF	PRF	
Proposed construction	$>$ Merkle-Damgård	MAC	pp-MAC, Δ-2PRF	—
		PRF	PRF, Δ-2PRF	

Multi-Property Preservation. EMD [5] and ESh [6] are modes of operation that preserve multiple properties (e.g., collision resistance, pseudo-randomness, etc..) These are integrative approaches, taking the converse point of view to the problem of domain extension; our goal is to construct a mode of operation that is specific to MAC/PRF property. While EMD or ESh offers a single program that can be used for multiple purposes (and hence a small source code, less confusion and a safety net), it may not perform the best with respect to a specific property (e.g., pseudo-randomness.) It should be noted that the code size of our mode of operation is much smaller than that of the compression function: The description of our construction requires only a loop, an XOR and a concatenation.

ENMAC. ENMAC [19] is an improvement over NMAC/HMAC, which is efficient particularly with short messages. This technique is also orthogonal to our approach, but it is so in a compatible way. That is, both ENMAC and our MAC in principle conform to the nested construction of NMAC (Recall that NMAC consists of outer encryption and inner hashing.) While ENMAC is an improvement on the outer function of NMAC, our construction is an improvement on the inner function. Hence ENMAC and our approach can coexist, but throughout the paper we base our construction upon the conventional NMAC for the sake of simplicity.[3]

[2] "pp-MAC" stands for privacy-preserving MAC, and "2PRF" for PRF against just two oracle queries.

[3] Intuitively, ENMAC improves performance mainly for short messages while our construction does so mainly for long messages. To a greater or lesser degree, each scheme alone improves performance essentially for all messages.

3 Design Principles

Merkle-Damgård. Figure 1 depicts the traditional Merkle-Damgård iteration using a compression function $h : \{0,1\}^{c+b} \to \{0,1\}^c$. In this classical hashing, a message M is divided into b-bit blocks as $M = m_1\|m_2\|\cdots$, and it is processed via the iteration $x_i \stackrel{\text{def}}{=} h(x_{i-1}\|m_i)$. Note that each invocation to h processes b-bits of M in this conservative mode of operation.

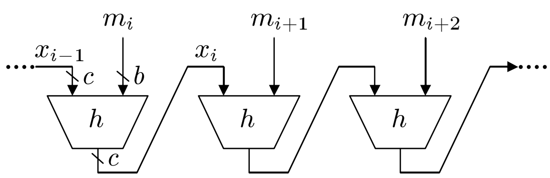

Fig. 1. Usual Merkle-Damgård iteration

Boosting. We start by trying to "maximize" the efficiency of each invocation to the compression function h. Note that h has $(c + b)$-bit input; we devise a mode of iteration we call "hyper-Merkle-Damgård," in which each invocation to h disposes of $c + b$ bits of a message M. We do this by XOR-ing the chaining variable x_i and the next c bits of M on each input. This is illustrated in Fig. 2. In the hyper-Merkle-Damgård iteration, a message M is divided as $M = m_1\|m_2\|\cdots$ so that $|m_1| = |m_3| = \cdots = c$ and $|m_2| = |m_4| = \cdots = b$. We refer to the $(c+b)$-bit segment $m_{2i-1}\|m_{2i}$ as a "chunk." The iteration works as $x_i \stackrel{\text{def}}{=} h\big((x_{i-1}\oplus m_{2i-1})\|m_{2i}\big)$. Thus, the hyper-Merkle-Damgård iteration is c/b as fast again as the usual Merkle-Damgård.

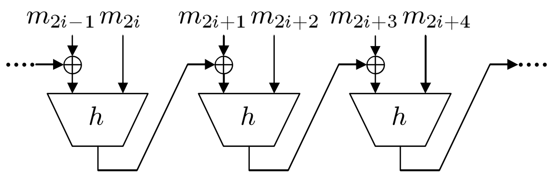

Fig. 2. Hyper-Merkle-Damgård iteration

Keying. We adopt the popular approach of keying a compression function h via its chaining variable. Namely, we obtain $h_K : \{0,1\}^b \to \{0,1\}^c$ by defining $h_K(\cdot) \stackrel{\text{def}}{=} h(K\|\cdot)$ where $K \stackrel{\$}{\leftarrow} \{0,1\}^c$. Also, let $\{0,1\}^{(c+b)*}$ denote the set of bit strings whose lengths are multiples of $c + b$ bits and define $H_K : \{0,1\}^{(c+b)*} \to \{0,1\}^c$ as $x_1 \leftarrow h_{K\oplus m_1}(m_2)$, $x_i \leftarrow h_{x_{i-1}\oplus m_{2i-1}}(m_{2i})$, $H_K(M) \stackrel{\text{def}}{=} x_n$, for an n-chunk message $M = m_1\|\cdots\|m_{2n}$.

Nesting. The keyed function H_K constructed above as it is cannot be used as a secure MAC/PRF. In order to turn it into secure construction, we employ the "nested approach" of the NI and NMAC construction. Namely, define $\mathsf{BNMAC}_{K,K'} : \{0,1\}^{(c+b)*} \to \{0,1\}^c$ via $\mathsf{BNMAC}_{K,K'}(\cdot) \stackrel{\text{def}}{=} h_{K'}\big(H_K(\cdot)\|1^{b-c}\big)$.[4] See Fig. 3 for a pictorial definition of our BNMAC construction. As already pointed out in [2], the conventional NMAC construction can be viewed as a computational version of the Carter-Wegman paradigm. Similarly, our result can be viewed as a related-key version of the result for the conventional NMAC. Since our assumptions concerning the function h include a related-key, non-standard one, we try to base the assumption upon as weak a condition as possible. We successfully do this; the condition only allows an adversary to make just two (related-key) oracle queries in a non-adaptive way.

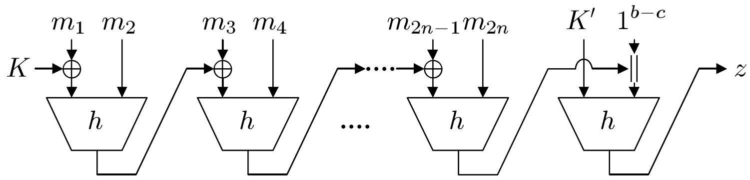

Fig. 3. Proposed MAC construction, double-key version (BNMAC)

Padding. The above $\mathsf{BNMAC}_{K,K'}(\cdot)$ accepts only messages whose lengths are multiples of $c+b$ bits. In order for the scheme to process a message of arbitrary length, the message M needs to be somehow padded. It turns out that any (one-to-one) padding $\{0,1\}^* \to \{0,1\}^{(c+b)*}$ works with our BNMAC construction, so hereafter we assume that a message always has a length multiple of $c+b$ bits (As an example of padding, just append $10\cdots0$.)

4 Definitions

Notation. The concatenation $x\|y$ of strings x and y is sometimes written simply xy. We say that a string x is a prefix of another string y and write $x \subset y$ if there exists a string e such that $xe = y$. We write $x \stackrel{\$}{\leftarrow} X$ to denote the operation of choosing an element uniformly at random from a set X and assigning its value to a variable x. An adversary A is a probabilistic machine that may have access to an oracle $\mathcal{O}$. The notation $A^{\mathcal{O}} \Rightarrow x$ indicates the event that, when run with the oracle $\mathcal{O}$, the adversary A outputs x. An oracle $\mathcal{O}$ is often defined by a game $\mathcal{G}$. In such a case we write $A^{\mathcal{G}}$ in place of $A^{\mathcal{O}}$. We also write $A \Leftarrow x$ to denote the operation of inputting the value x into A.

[4] Here we assume that $b \geq c$. Although we could get around this requirement by extending the outer function via Merkle-Damgård iteration [22], yet for simplicity we assume this condition throughout the paper.

Notion of PRF. Let $\{f_K : \mathcal{M} \to X\}$ be a family of functions with $K \in \{0,1\}^k$. A prf-adversary A tries to distinguish between two oracles, the "real" oracle being $f_K(\cdot)$, $K \xleftarrow{\$} \{0,1\}^k$ and the "random" oracle being $f(\cdot)$, $f \xleftarrow{\$} \{f : \mathcal{M} \to X\}$ (Fixing K fixes the real oracle, and fixing f fixes the random oracle.) Succinctly, define the advantage function of A as

$$\mathrm{Adv}_f^{\mathrm{prf}}(A) \stackrel{\mathrm{def}}{=} \Pr\big[A^f \Rightarrow 1\big] - \Pr\big[A^\$ \Rightarrow 1\big],$$

where by f we denote the real oracle and by \$ the random oracle. The first probability is defined over the coins of A and $K \xleftarrow{\$} \{0,1\}^k$, and the second probability over the coins of A and $f \xleftarrow{\$} \{f : \mathcal{M} \to X\}$.

New Notion of Δ-2PRF. Let $\{f_K : \mathcal{M} \to X\}$ be a family of functions with $K \in \{0,1\}^k$. A Δ-2prf adversary A tries to distinguish between two games, as defined in Table 2. Namely, at the beginning of each game the adversary A queries once (m, Δ, m') with $m, m' \in \mathcal{M}$ and $\Delta \in \{0,1\}^k$. Then the oracle answers (x, x') to the adversary A, whose values are determined differently in each game as described. Finally A outputs 1 or 0. Succinctly define

$$\mathrm{Adv}_f^{\Delta\text{-2prf}}(A) \stackrel{\mathrm{def}}{=} \Pr\big[A^f \Rightarrow 1\big] - \Pr\big[A^\$ \Rightarrow 1\big],$$

where again by f we denote the real oracle and by \$ the random oracle.

Table 2. Real and random games for Δ-2PRF

Real	Random
$A \Rightarrow (m, \Delta, m')$	$A \Rightarrow (m, \Delta, m')$
$K \xleftarrow{\$} \{0,1\}^k$	$x, x' \xleftarrow{\$} X$
$x \leftarrow h_K(m)$; $x' \leftarrow h_{K\oplus\Delta}(m')$	If $\Delta = 0$ and $m = m'$ then $x' \leftarrow x$ EndIf
$A \Leftarrow (x, x')$	$A \Leftarrow (x, x')$

Resource Parameters. An adversary A's resources are quantified with respect to its time complexity t, the number q of oracle queries and the length ℓ (in chunks, if applicable) of each query. We adopt the convention that the time complexity t includes the total execution time of an overlying game (the maximum of each game) plus the code size of A. Define

$$\mathrm{Adv}_f^{\mathrm{goal}}(t, q, \ell) \stackrel{\mathrm{def}}{=} \max_A \mathrm{Adv}_f^{\mathrm{goal}}(A),$$

where max is taken over adversaries A, each having time complexity at most t and making at most q oracle queries, each query being at most ℓ chunks. Often one or two of t, q, ℓ are inappropriate to be quantified, in which case they are omitted from the notation. Here, "goal" indicates the property in question, e.g., "prf." We write $T_f(\ell)$ to denote the time complexity that takes to compute a function f on a input whose length is ℓ chunks (and again, ℓ may be omitted.)

5 Discussion on Δ-2PRF Property

Since we introduce the new notion Δ-2PRF on which our proofs of security are based, in this section we take a closer look at this requirement on the underlying compression function h. Intuitively, we can view the Δ-2PRF condition as a form of pseudo-randomness under a related-key attack. Yet, it is so in one of the weakest forms possible; namely, in Δ-2PRF, an adversary is limited to ask only two queries, and these queries must be performed non-adaptively. In other words, he must submit his entire queries (two messages m, m' and a relation Δ) together at the beginning of the game.

So the notion of Δ-2PRF itself is not a demanding requirement, though it cannot be deduced from the standard PRF (against q queries) assumption. We remark that the condition that h be a Δ-2PRF and the condition that h be a PRF (against q queries) are independent; neither one implies the other.

To get the feel of handling the notion of Δ-2PRF, we give an example of MD5. Let $\mathsf{md5} : \{0,1\}^{128+512} \to \{0,1\}^{128}$ be the compression function of MD5. It is known [11] that md5 is vulnerable to so called a "pseudo-collision" attack. That is, for $\Delta \stackrel{\text{def}}{=}$ 8000 0000 8000 0000 8000 0000 8000 0000 the condition $\mathsf{md5}_K(m) = \mathsf{md5}_{K \oplus \Delta}(m)$ ($K \stackrel{\$}{\leftarrow} \{0,1\}^{128}$, $m \stackrel{\$}{\leftarrow} \{0,1\}^{512}$) holds with a probability of about $1/2^{46} \gg 1/2^{128}$. Using this technique, an adversary A can attack md5 in the Δ-2PRF sense: A queries (m, Δ, m) ($m \stackrel{\$}{\leftarrow} \{0,1\}^{512}$) and receives (x, x'); if $x = x'$, then A outputs 1; otherwise, A outputs 0. Such an A has advantage $\mathrm{Adv}^{\Delta\text{-2prf}}_{\mathsf{md5}}(A) \approx 1/2^{46} - 1/2^{128}$. Thus, md5 does not satisfy the Δ-2PRF property.

This characteristic of md5 is rather critical in its architecture. We expect that this sort of attack be precluded by structural designs of forthcoming compression functions, and certainly we would hope for designs without such a flaw in more "matured" compression functions such as sha256.

At the end of this discussion, we emphasize the point that breaking Δ-2PRF is easier than finding pseudo-collisions. Our proofs of security require that h be a Δ-2PRF, and h just being resistant to pseudo-collisions would not suffice for our purpose according to the current reduction.

6 Security Proofs (Double-Key Version)

This section proves the following:

Theorem 1. *Let* BNMAC *be the two-key construction as defined in Sec. 3. If the underlying compression function h is a PRF and a Δ-2PRF, then* BNMAC *is a PRF. More concretely, we have*

$$\mathrm{Adv}^{\mathrm{prf}}_{\mathsf{BNMAC}}(t,q,\ell) \le \mathrm{Adv}^{\mathrm{prf}}_{h}(t,q) + \binom{q}{2} \cdot \left(2(\ell+1) \cdot \mathrm{Adv}^{\Delta\text{-2prf}}_{h}(t') + \frac{1}{2^c} \right),$$

where $t' = (4\ell+1) \cdot T_h$.

The reduction in the above theorem is essentially tight, due to the birthday attack [20]. For more discussion on the gap from the exactly tight bound, see [2].

In order to prove this theorem, we need the following five lemmas. The five lemmas sequentially reduce the PRF condition on the BNMAC scheme to the PRF and Δ-2PRF conditions on the underlying compression function. Along the proofs, we need several intermediate security notions, which are defined when they first appear. See Fig. 4 for a guide map.

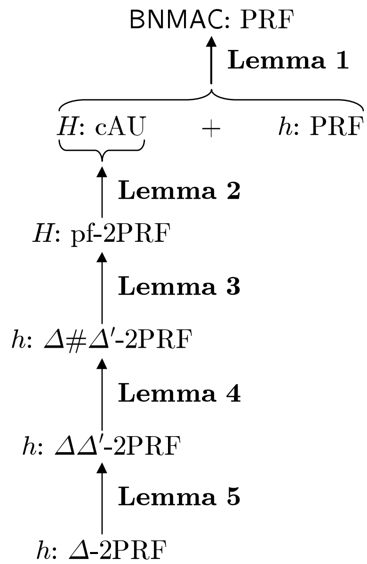

Fig. 4. A proof map

For stating the first lemma, we need to define the notion of cAU (computational almost-universality.) An au-adversary A against a keyed function $H_K : \{0,1\}^{(c+b)*} \to \{0,1\}^c$ (with $K \in \{0,1\}^c$) simply outputs a pair of messages (M, M') with $M, M' \in \{0,1\}^{(c+b)*}$; define

$$\mathrm{Adv}^{\mathrm{au}}_{H}(A) \stackrel{\mathrm{def}}{=} \Pr\big[H_K(M) = H_K(M') \wedge M \neq M' \bigm| A \Rightarrow (M, M'), K \stackrel{\$}{\leftarrow} \{0,1\}^c\big].$$

Here note that such an adversary is non-adaptive. It also means that we can disregard the time complexity of au-adversaries (and often it is set to $2 \cdot T_H(\ell)$.)

Lemma 1. *Let* $H_K : \{0,1\}^{(c+b)*} \to \{0,1\}^c$ *and* $h_{K'} : \{0,1\}^b \to \{0,1\}^c$ *be keyed functions with* $K, K' \in \{0,1\}^c$. *If* H_K *is cAU and* $h_{K'}$ *a PRF, then the composition* $h \circ H_{(K',K)}$ *defined by* $h_{K'}\big(H_K(M)\|1^{b-c}\big)$ *is a PRF. More concretely written, the following holds:*

$$\mathrm{Adv}^{\mathrm{prf}}_{h\circ H}(t,q,\ell) \le \mathrm{Adv}^{\mathrm{prf}}_{h}(t,q) + \binom{q}{2} \cdot \mathrm{Adv}^{\mathrm{au}}_{H}(t',\ell),$$

where $t' = 2 \cdot T_H(\ell)$.

Proof. This lemma (along with its pp-MAC version) is proved in [2]. □

The next lemma relates cAU to pseudo-randomness property, utilizing the iterative structure of the hyper-Merkle-Damgård. See Table 3 for the notion of 2PRF. We say that a 2prf-adversary A is "prefix-free" (pf-2prf) if $M \not\subset M'$ and $M \not\supset M'$, where (M, M') is the query output by A. Note that in particular, prefix-freeness implies $M, M' \neq \varepsilon$ (null) and $M \neq M'$.

Table 3. Real and random games for 2PRF

Real	Random
$A \Rightarrow (M, M')$	$A \Rightarrow (M, M')$
$K \xleftarrow{\$} \{0,1\}^c$	$x, x' \xleftarrow{\$} \{0,1\}^c$
$x \leftarrow H_K(M)$; $x' \leftarrow H_K(M')$	If $M = M'$ then $x' \leftarrow x$ EndIf
$A \Leftarrow (x, x')$	$A \Leftarrow (x, x')$

Lemma 2. *Let* $h : \{0,1\}^{c+b} \to \{0,1\}^c$ *be a compression function and* $H_K : \{0,1\}^{(c+b)*} \to \{0,1\}^c$ *the hyper-Merkle-Damgård iteration constructed of* h, *keyed via its initial chaining variable. If* H_K *is prefix-free 2PRF, then it is cAU. More concretely,*

$$\mathrm{Adv}^{\mathrm{au}}_{H}(t,\ell) \le \mathrm{Adv}^{\text{pf-2prf}}_{H}(t,\ell+1) + \frac{1}{2^c}.$$

Proof. This can be easily proven by using the well-known "extension trick" [2]. □

Now we reduce the condition that H be a prefix-free 2PRF to the condition that h be a $\Delta\#\Delta'$-2PRF, whose definition can be found in Table 4.

Lemma 3. *If* h *is a* $\Delta\#\Delta'$*-2PRF, then its hyper-Merkle-Damgård iteration* H *is a prefix-free PRF. More concretely, we have*

$$\mathrm{Adv}^{\text{pf-2prf}}_{H}(t,\ell) \le \ell \cdot \mathrm{Adv}^{\Delta\#\Delta'\text{-2prf}}_{h}(t'),$$

where $t' = t + 2 \cdot T_H(\ell)$.

Table 4. Real and random games for $\Delta\#\Delta'$-2PRF

Real	Random
$A \Rightarrow (\Delta, m, \#, \Delta', m')$	$A \Rightarrow (\Delta, m, \#, \Delta', m')$
$K, K' \xleftarrow{\$} \{0,1\}^c$	$x, x' \xleftarrow{\$} \{0,1\}^c$
If $\# = 1$ then	If $\# = 1$ and
$x \leftarrow h_{K\oplus\Delta}(m)$; $x' \leftarrow h_{K\oplus\Delta'}(m')$	$(\Delta, m) = (\Delta', m')$ then
Else (i.e., $\# = 2$)	$x' \leftarrow x$
$x \leftarrow h_{K\oplus\Delta}(m)$; $x' \leftarrow h_{K'\oplus\Delta'}(m')$	EndIf
EndIf; $A \Leftarrow (x, x')$	$A \Leftarrow (x, x')$

Proof. Let A be a pf-2prf adversary attacking H, having time complexity at most t and querying messages each of at most ℓ chunks. We would like to bound the advantage $\mathrm{Adv}_H^{\text{pf-2prf}}(A)$. Let (M, M') denote the pair of messages that A outputs, and write $M = m_1 \cdots m_{2n}$ (n chunks) and $M' = m'_1 \cdots m'_{2n'}$ (n' chunks). Note that $n, n' \leq \ell$. Consider the intermediate games $\mathcal{G}_i$ defined in Fig. 5 for $i = 0, \cdots, \ell$. Note that running $A^{\mathcal{G}_0}$ can be identified with running A^H, treating the condition $m_1 \cdots m_{2i} = m'_1 \cdots m'_{2i}$ to be true when $i = 0$. Also, running $A^{\mathcal{G}_\ell}$ coincides with the random game for A. Hence

$$\begin{aligned}\mathrm{Adv}_H^{\text{pf-2prf}}(A) &= \Pr\big[A^H \Rightarrow 1\big] - \Pr\big[A^\$ \Rightarrow 1\big] \\ &= P_0 - P_\ell \\ &= \sum_{i=0}^{\ell-1}(P_i - P_{i+1}),\end{aligned}$$

where $P_i \overset{\text{def}}{=} \Pr\big[A^{\mathcal{G}_i} \Rightarrow 1\big]$ for $i \in \{0, \ldots, \ell\}$.

Game $\mathcal{G}_i$	**Adversary** B_i
$A \Rightarrow (M, M')$	$A \Rightarrow (M, M')$
	If $m_1 \cdots m_{2i} = m'_1 \cdots m'_{2i}$ then
	$(x, x') \leftarrow \mathcal{O}(m_{2i+1}, m_{2i+2}, 1, m'_{2i+1}, m'_{2i+2})$
$x, x' \xleftarrow{\$} \{0,1\}^c$	Else (i.e., $m_1 \cdots m_{2i} \neq m'_1 \cdots m'_{2i}$)
	$(x, x') \leftarrow \mathcal{O}(m_{2i+1}, m_{2i+2}, 2, m'_{2i+1}, m'_{2i+2})$
	EndIf
Define (y, y') as in Table 5	Define (y, y') as in Table 6
$A \Leftarrow (y, y')$	$A \Leftarrow (y, y')$
	Output whatever A outputs

Fig. 5. Intermediate games $\mathcal{G}_i$ and adversaries B_i

Table 5. Definition of (y, y') in game $\mathcal{G}_i$

	$n' \leq i$	$n' \geq i+1$
$n \leq i$	$y \leftarrow x$ $y' \leftarrow x'$	$y \leftarrow x$ $y' \leftarrow H_{x'}(m'_{2i+1} \cdots m'_{2n'})$
$n \geq i+1$	$y \leftarrow H_x(m_{2i+1} \cdots m_{2n})$ $y' \leftarrow x'$	If $m_1 \cdots m_{2i} = m'_1 \cdots m'_{2i}$ then $y \leftarrow H_x(m_{2i+1} \cdots m_{2n})$ $y' \leftarrow H_x(m'_{2i+1} \cdots m'_{2n'})$ Else (i.e., $m_1 \cdots m_{2i} \neq m'_1 \cdots m'_{2i}$) $y \leftarrow H_x(m_{2i+1} \cdots m_{2n})$ $y' \leftarrow H_{x'}(m'_{2i+1} \cdots m'_{2n'})$

Table 6. Definition of (y, y') in adversary B_i

	$n' \leq i$	$n' \geq i+1$
$n \leq i$	$y \leftarrow x$ $y' \leftarrow x'$	$y \leftarrow x$ $y' \leftarrow H_{x'}(m'_{2i+3} \cdots m'_{2n'})$
$n \geq i+1$	$y \leftarrow H_x(m_{2i+3} \cdots m_{2n})$ $y' \leftarrow x'$	$y \leftarrow H_x(m_{2i+3} \cdots m_{2n})$ $y' \leftarrow H_{x'}(m'_{2i+3} \cdots m'_{2n'})$

Now for each $i = 0, \ldots, \ell - 1$ we define an adversary B_i that uses A as a subroutine and attacks h in the $\Delta\#\Delta'$-2PRF sense, as described in Fig. 5. It can be directly verified that $\Pr\big[B_i^h \Rightarrow 1\big] = \Pr\big[A^{\mathcal{G}_i} \Rightarrow 1\big] = P_i$ and $\Pr\big[B_i^\$ \Rightarrow 1\big] = \Pr\big[A^{\mathcal{G}_{i+1}} \Rightarrow 1\big] = P_{i+1}$. Hence

$$
\begin{aligned}
\mathrm{Adv}_H^{\text{pf-2prf}}(A) &= \sum_{i=0}^{\ell-1}(P_i - P_{i+1}) \\
&= \sum_{i=0}^{\ell-1}\Big(\Pr\big[B_i^h \Rightarrow 1\big] - \Pr\big[B_i^\$ \Rightarrow 1\big]\Big) \\
&= \sum_{i=0}^{\ell-1}\mathrm{Adv}_h^{\Delta\#\Delta'\text{-2prf}}(B_i) \\
&\leq \sum_{i=0}^{\ell-1}\mathrm{Adv}_h^{\Delta\#\Delta'\text{-2prf}}(t') \\
&= \ell \cdot \mathrm{Adv}_h^{\Delta\#\Delta'\text{-2prf}}(t').
\end{aligned}
$$

□

Next we reduce the condition that h be a $\Delta\#\Delta'$-2PRF to the condition that h be a $\Delta\Delta'$-2PRF, whose definition can be found in Table 7. The notion of

$\Delta\Delta'$-2PRF is simpler than that of $\Delta\#\Delta'$-2PRF, and it is also closer to that of Δ-2PRF.

Table 7. Real and random games for $\Delta\Delta'$-2PRF

Real	Random
$A \Rightarrow (\Delta, m, \Delta', m')$	$A \Rightarrow (\Delta, m, \Delta', m')$
$K \xleftarrow{\$} \{0,1\}^c$	$x, x' \xleftarrow{\$} \{0,1\}^c$
$x \leftarrow h_{K\oplus\Delta}(m)$	If $(\Delta, m) = (\Delta', m')$ then
$x' \leftarrow h_{K\oplus\Delta'}(m')$	$x' \leftarrow x$ EndIf
$A \Leftarrow (x, x')$	$A \Leftarrow (x, x')$

Lemma 4. *If a compression function h is $\Delta\Delta'$-2PRF, then it is $\Delta\#\Delta'$-2PRF. More concretely, we have*

$$\mathrm{Adv}_h^{\Delta\#\Delta'\text{-2prf}}(t) \le 2 \cdot \mathrm{Adv}_h^{\Delta\Delta'\text{-2prf}}(t'),$$

where $t' = t + T_h$.

Proof. Let B be a $\Delta\#\Delta'$-2prf adversary against h having time complexity at most t. We create $\Delta\Delta'$-2prf adversaries C and C', each using B as its subroutine, as described in Fig. 6. It can be directly verified that $\Pr\big[C^h \Rightarrow 1\big] = \Pr\big[C'^{\$} \Rightarrow 1\big]$, $\Pr\big[C^{\$} \Rightarrow 1\big] = \Pr\big[B^{\$} \Rightarrow 1\big]$ and $\Pr\big[C'^h \Rightarrow 1\big] = \Pr\big[B^h \Rightarrow 1\big]$. Therefore

$$\begin{aligned}
\mathrm{Adv}_h^{\Delta\#\Delta'\text{-2prf}}(B) &= \Pr\big[B^h \Rightarrow 1\big] - \Pr\big[B^{\$} \Rightarrow 1\big] \\
&= \Pr\big[C'^h \Rightarrow 1\big] - \Pr\big[C'^{\$} \Rightarrow 1\big] + \Pr\big[C^h \Rightarrow 1\big] - \Pr\big[C^{\$} \Rightarrow 1\big] \\
&= \mathrm{Adv}_h^{\Delta\Delta'\text{-2prf}}(C') + \mathrm{Adv}_h^{\Delta\Delta'\text{-2prf}}(C) \\
&\le 2 \cdot \mathrm{Adv}_h^{\Delta\Delta'\text{-2prf}}(t').
\end{aligned}$$

□

Finally, we are ready to reach the condition of Δ-2PRF. The last lemma gives us straight-forward reduction of $\Delta\Delta'$-2PRF to Δ-2PRF:

Lemma 5. *If a compression function h is Δ-2PRF, then it is $\Delta\Delta'$-2PRF. More concretely, we have*

$$\mathrm{Adv}_h^{\Delta\Delta'\text{-2prf}}(t) \le \mathrm{Adv}_h^{\Delta\text{-2prf}}(t).$$

Proof. Let C be a $\Delta\Delta'$-2prf adversary against h having time complexity at most t. We construct a Δ-2prf adversary D against h that uses C as its subroutine, as follows.

Adversary C	**Adversary** C'
$B \Rightarrow (\Delta, m, \#, \Delta', m')$	$B \Rightarrow (\Delta, m, \#, \Delta', m')$
If $\# = 1$ then	$K \xleftarrow{\$} \{0,1\}^c$; $x \leftarrow h_{K \oplus \Delta}(m)$
$\quad (x, x') \leftarrow \mathcal{O}(\Delta, m, \Delta', m')$	If $\# = 1$ then
Else (i.e., $\# = 2$)	$\quad x' \leftarrow h_{K \oplus \Delta'}(m')$
$\quad (x, x) \leftarrow \mathcal{O}(\Delta, m, \Delta, m)$	Else (i.e., $\# = 2$)
$\quad x' \xleftarrow{\$} \{0,1\}^c$ EndIf	$\quad (x', x') \leftarrow \mathcal{O}(\Delta', m', \Delta', m')$ EndIf
$B \Leftarrow (x, x')$	$B \Leftarrow (x, x')$
Output whatever B outputs	Output whatever B outputs

Fig. 6. Adversaries C, C'

D runs C and obtains the query (Δ, m, Δ', m'). Then D asks its oracle a query and receives $(x, x') \leftarrow \mathcal{O}(m, \Delta \oplus \Delta', m')$. D forwards (x, x') to C and outputs whatever C outputs.

Here observe that $\Pr[D^h \Rightarrow 1] = \Pr[C^h \Rightarrow 1]$ and that $\Pr[D^\$ \Rightarrow 1] = \Pr[C^\$ \Rightarrow 1]$. Hence we have

$$\begin{aligned} \mathrm{Adv}_h^{\Delta\Delta'\text{-2prf}}(C) &= \mathrm{Adv}_h^{\Delta\text{-2prf}}(D) \\ &\le \mathrm{Adv}_h^{\Delta\text{-2prf}}(t), \end{aligned}$$

neglecting the increase in D's time complexity. $\square$

The above five lemmas prove Theorem 1. Recall that, without loss of generality we can estimate the time complexity of a cAU adversary to be $2 \cdot T_H(\ell)$. So for the time complexity t' in Theorem 1 we get $t' = 2 \cdot T_H(\ell) + 2 \cdot T_H(\ell) + T_h = (4\ell + 1) \cdot T_h$.

7 Single-Key Versions

Our BNMAC construction so far requires two independent keys $K, K' \in \{0,1\}^c$, which may be an undesirable feature in some cases in practice. However, this problem is easily resolved through the pseudo-randomness of h. We show two solutions.

The first method is a trivial way of deriving two keys. Let $K^* \in \{0,1\}^c$ be a master key. From K^* derive two keys as $K \leftarrow h_{K^*}(0^b)$ and $K' \leftarrow h_{K^*}(1^b)$. We then use these two keys in place of $K, K' \in \{0,1\}^c$ in the BNMAC construction. See Fig. 7 for a pictorial description. The only difference between the original double-key version and this single-key variant lies in the way how the two keys K and K' are produced (in the former $K, K' \xleftarrow{\$} \{0,1\}^c$, whereas in the latter these keys are derived via h from $K^* \xleftarrow{\$} \{0,1\}^c$.) Hence distinguishing between the two versions amounts to breaking the pseudo-randomness of h (with two constant queries 0^b and 1^b to the oracle.) It should be noted that if we replace the PRF assumption with that of pp-MAC in Lemma 1, then the pp-MAC version of Theorem 1 still holds for this single-key variant. This is because the 2PRF requirement on h for key derivation is absorbed into Δ-2PRF of h, not PRF (against q queries.)

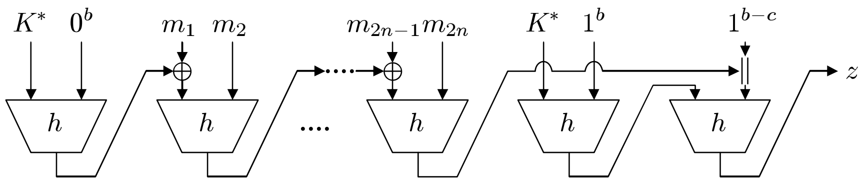

Fig. 7. Single-key version 1

The second method takes the idea from [22]. See Fig. 8 for the description of this variant. While this version saves one extra block of invocation to the compression function, there are two points to be attended to. One is that now we need the condition $b \geq c + 1$ (rather than $b \geq c$.) The other is that the pp-MAC version appears to be infeasible in this case.

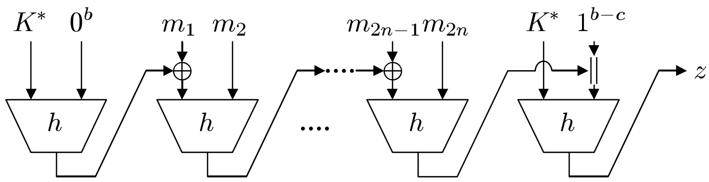

Fig. 8. Single-key version 2

8 Using a Shorter Key

Recall that the size c of a chaining variable may be larger than one's desired security parameter, depending on a choice of compression functions. This means that in practice the desired size k of the master key K^* may be smaller than c, disabling the above single-key construction.

This difficulty can be settled in several ways. One is to use only the first k bits of c in the above single-key variant (and the remaining $c - k$ bits may be padded with zeros.) Another is to fill out the c bits by multiple copies of a k-bit key, as $K^*\|K^*\|\cdots$. In either example, note that we still do not lose our formal proofs of security with the first version of the single-key construction, assuming additionally that the newly keyed function is a 2PRF against corresponding two oracle queries.

9 Summary

This paper proposes a novel mode of operation of compression functions, called hyper-Merkle-Damgård, which can process a message faster than the conventional Merkle-Damgård iteration and can be used exclusively as a MAC/PRF. The proofs of security are based on the assumption that the underlying compression function satisfies some PRF properties. These PRF properties include a new notion which we call Δ-2PRF. We carefully take a look at this property and identify it as not a demanding condition. We first give proofs of security of a double-key version, called BNMAC, and then show that single-key versions can be easily derived, along with flexibility of the key size.

Acknowledgments. The author would like to thank ASIACRYPT 2007 anonymous reviewers for their valuable comments, insightful questions and useful suggestions. The feedback helps the author improve the quality of the paper in its various aspects.

References

1. An, J.H., Bellare, M.: Constructing VIL-MACs from FIL-MACs: Message authentication under weakened assumptions. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 252–269. Springer, Heidelberg (1999)
2. Bellare, M.: New proofs for NMAC and HMAC: Security without collision-resistance. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 602–619. Springer, Heidelberg (2006)
3. Bellare, M., Canetti, R., Krawczyk, H.: Pseudorandom functions revisited: The cascade construction and its concrete security. IEEE Symposium on Foundations of Computer Science, 514–523 (1996)
4. Bellare, M., Guérin, R., Rogaway, P.: XOR MACs: New methods for message authentication using finite pseudorandom functions. In: Coppersmith, D. (ed.) CRYPTO 1995. LNCS, vol. 963, pp. 15–28. Springer, Heidelberg (1995)
5. Bellare, M., Ristenpart, T.: Multi-property-preserving hash domain extension and the EMD transform. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 299–314. Springer, Heidelberg (2006)
6. Bellare, M., Ristenpart, T.: Hash functions in the dedicated-key setting: Design choices and MPP transforms. In: Arge, L., Cachin, C., Jurdzinski, T., Tarlecki, A. (eds.) ICALP 2007. LNCS, vol. 4596, pp. 399–410. Springer, Heidelberg (2007)
7. Bellare, M., Rogaway, P.: Collision-resistant hashing: Towards making UOWHFs practical. In: Kaliski, B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 470–484. Springer, Heidelberg (1997)
8. Contini, S., Yin, Y.L.: Forgery and partial key-recovery attacks on HMAC and NMAC using hash collisions. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 37–53. Springer, Heidelberg (2006)
9. Coron, J.S., Dodis, Y., Malinaud, C., Puniya, P.: Merkle-Damgård revisited: How to construct a hash function. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 430–448. Springer, Heidelberg (2005)
10. Damgård, I.: A design principle for hash functions. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 416–427. Springer, Heidelberg (1989)

11. den Boer, B., Bosselaers, A.: Collisions for the compressin function of MD5. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 293–304. Springer, Heidelberg (1993)
12. Halevi, S., Krawcyzk, H.: Strengthening digital signatures via randomized hashing. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 41–59. Springer, Heidelberg (2006)
13. Kim, J., Biryukov, A., Preneel, B., Hong, S.: On the security of HMAC and NMAC based on HAVAL, MD4, MD5, SHA-0 and SHA-1. In: Prisco, R.D., Yung, M. (eds.) SCN 2006. LNCS, vol. 4116, pp. 242–256. Springer, Heidelberg (2006)
14. Lucks, S.: A failure-friendly design principle for hash functions. In: Roy, B.K. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 474–494. Springer, Heidelberg (2005)
15. Maurer, U.M., Sjödin, J.: Single-key AIL-MACs from any FIL-MAC. In Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, pp. 472–484. Springer, Heidelberg (2005)
16. Merkle, R.C.: One way hash functions and DES. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 428–446. Springer, Heidelberg (1989)
17. NIST: NIST brief comments on recent cryptanalytic attacks on secure hashing functions and the continued security provided by SHA-1 (2004)
18. NIST: NIST brief comments on recent cryptanalytic attacks on SHA-1 (2004)
19. Patel, S.: An efficient MAC for short messages. In: Nyberg, K., Heys, H.M. (eds.) SAC 2002. LNCS, vol. 2595, pp. 353–368. Springer, Heidelberg (2003)
20. Preneel, B., van Oorschot, P.C.: On the security of iterated message authentication codes. IEEE Transactions on Information Theory 45(1), 188–199 (1999)
21. Shoup, V.: A composition theorem for universal one-way hash functions. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 445–452. Springer, Heidelberg (2000)
22. Yasuda, K.: "Sandwich" is indeed secure: How to authenticate a message with just one hashing. In: Pieprzyk, J., Ghodosi, H., Dawson, E. (eds.) ACISP 2007. LNCS, vol. 4586, pp. 355–369. Springer, Heidelberg (2007)