"Sandwich" Is Indeed Secure: How to Authenticate a Message with Just One Hashing

Kan Yasuda

NTT Information Sharing Platform Laboratories, NTT Corporation
1-1 Hikarinooka Yokosuka-shi, Kanagawa-ken 239-0847 Japan
yasuda.kan@lab.ntt.co.jp

Abstract. This paper shows that the classical "Sandwich" method, which prepends and appends a key to a message and then hashes the data using Merkle-Damgård iteration, does indeed provide a secure Message Authentication Code (MAC). The Sandwich construction offers a single-key MAC which can use the existing Merkle-Damgård implementation of hash functions as is, without direct access to the compression function. Hence the Sandwich approach gives us an alternative for HMAC particularly in a situation where message size is small and high performance is required, because the Sandwich scheme is more efficient than HMAC: it consumes only two blocks of "waste" rather than three as in HMAC, and it calls the hash function only once, whereas HMAC requires two invocations of hash function. The security result of the Sandwich method is similar to that of HMAC; namely, we prove that the Sandwich construction yields a PRF(Pseudo-Random Functions)-based MAC, provided that the underlying compression function satisfies PRF properties. In theory, the security reduction of the Sandwich scheme is roughly equivalent to that of HMAC, but in practice the requirements on the underlying compression function look quite different. Also, the security of the Sandwich construction heavily relies on the filling and padding methods to the data, and we show several ways of optimizing them without losing a formal proof of security.

Keywords: Message Authentication Code, MAC, Hash Function, Compression Function, Merkle-Damgård, Envelope MAC, RFC1828, HMAC.

1 Introduction

A Message Authentication Code (MAC) is a symmetric-key cryptographic primitive that is widely used for ensuring authenticity and data integrity. It is an algorithm, usually deterministic, that takes as its input a message M (which may not be encrypted), processes it with a secret key K and then produces a fixed-length output τ called "tag". A secure MAC protects tags from being forged.

A MAC is commonly realized via a cryptographic hash function, like SHA-1 or SHA-256 [1], for its performance and availability in software libraries. A hash

J. Pieprzyk, H. Ghodosi, and E. Dawson (Eds.): ACISP 2007, LNCS 4586, pp. 355–369, 2007.

function is usually constructed by a smaller primitive called "compression function." A compression function only processes messages of a fixed length. In order to create a hash function that accepts messages of variable lengths, the messages are padded and the compression function is iterated via a mode of operation. The most widespread mode is so called Merkle-Damgård strengthening (padding) and iteration. We are interested in hash functions that are implemented in this way.

The hash function, however, is keyless. In order to use it as a MAC, we must somehow make the hash function keyed. We briefly review four types of keying the hash function h, along the course of [2].

"Prefix" Method. This faulty way prepends a key K to a message M and then lets the hash value $\tau = h(K\|M)$ be the tag. It is well known that this method is vulnerable against so called the "extension attack." Namely, an adversary asks its oracle the tag $\tau = h(K\|M)$ for a message M, computes a value $\tau' = h(\tau, M')$ where τ is used as the initial vector for h and M' an arbitrary message, and then succeeds in submitting the pair $(M\|M', \tau')$ as a forgery.

"Suffix" Method. An obvious way to avoid the extension attack is to append, rather than prepend, the key K to the message M and then obtain the tag $\tau = h(M\|K)$. This gets around the extension attack but suffers from the collision attack. Namely, let M, M' be two messages that produce a collision of the keyless hash function h, so that $h(M) = h(M')$. Then an adversary queries its oracle the tag $\tau = h(M\|K)$ and then submits a pair (M', τ) as a forgery. For more discussions on the notion of collision resistance for keyless hash functions, see [3].

"Sandwich" Method. The combination of the above two approaches originates from the "hybrid method" in [4], where the tag τ is computed as $\tau = h(K\|p\|M\|K')$ with two independent keys K, K' and key filling (padding) p. A proof of security of the hybrid method is essentially given in [5]. The single-key version, in which the tag τ is computed as $\tau = h(K\|p\|M\|K)$, appears in the standardization of IPSec version 1 [2,6,7] and is known as the "envelope MAC." The envelope MAC, however, is shown to be vulnerable against key recovery attack [8] (which is more threatening than forgery attack.) We note that it is the lack of appropriate filling between the message M and the last key K, rather than the usage of a single key, that contributes to this key recovery attack.

Nowadays these hybrid/envelope techniques seem to attract little interest, mainly due to the above key recovery attack and the affirmative adoption of HMAC (described below) in IPSec version 2 ([7] is "obsoleted" by [9] which is also now "historic." [6] is still present only for the purpose of backward compatibility.) This paper calls attention back to this classical method. It is the contribution of this paper to show that the "Sandwich" scheme, which basically works as $\tau = h(K\|p\|M\|p'\|K)$, indeed yields a secure, single-key MAC, as long as the underlying compression function satisfies Pseudo-Random-Function (PRF) properties and appropriate fillings p, p' and padding methods are combined with. It should be remarked that as a byproduct the Sandwich scheme precludes the key recovery attack.

HMAC. HMAC is introduced in [10] with a (rather rough but formal) proof of security. Its new proof with complete reduction to the compression function is given in [11]. HMAC works as follows. It first computes the intermediate value $v = h\big((K \oplus IPAD)\|IPAD'\|M\big)$ and then computes the tag $\tau = h\big((K \oplus OPAD)\|OPAD'\|v\big)$, where $\oplus$ denotes bitwise exclusive-OR and $IPAD$, $IPAD'$, $OPAD$, $OPAD'$ are pre-defined constants. Note that unlike the other methods mentioned so far, HMAC requires two invocations of hash function h.

Organization of This Paper. In Sect. 2 we identify the improvements in performance of the Sandwich scheme as compared to that of HMAC. In Sect. 3 we review some preliminaries of hash function, which are necessary in Sect. 4 to define the basic construction of the Sandwich scheme and to state its main security result. In Sect. 5 we compare this result to that of HMAC [11].

Section 7 is devoted for the security proof of the basic Sandwich construction, preceded by necessary definitions in Sect. 6. As to the reduction techniques in Sect. 7, we follow the line of [11], rather than that of [5] which essentially contains a proof of security for two-key hybrid method. Our approach enables us to prove the security of the single-key Sandwich scheme and also to compare our result directly with that of HMAC in [11].

It is also the contribution of this paper to introduce several variants of the basic Sandwich construction. Sections 8, 9 and 10 discuss these derivatives and show ways to modify the filling and padding methods in the basic construction, with improved efficiency and without loss of formal proofs of security. We emphasize the fact that although these improvements seem only subtle and minor, they become valuable in a situation with severe resource requirements and/or with short messages. The security results in Sect. 8 and 10 make use of the multi-oracle families introduced in [5].

2 Performance Comparison to HMAC

In Table 1 we summarize the performance comparison between the Sandwich method and HMAC. The Sandwich method consumes (at most) two blocks of "waste," corresponding to the very first and last blocks for the key. HMAC, on the other hand, consumes one more block for processing the intermediate value v, totaling three blocks (The "waste" is defined to be the number of invocations of compression function in the scheme minus that in the usual Merkle-Damgård.)

Also, the Sandwich method calls a hash function only once, as in $h(K\|M\|K)$, whereas HMAC requires two invocations of a hash function, one for producing the intermediate hash value $v = h\big((K \oplus IPAD)\|IPAD'\|M\big)$, and then another for processing the hash value v with the key K as in $h = \big((K \oplus OPAD)\|OPAD'\|v\big)$.

These problems of HMAC are discussed and improved in [12] (which appears in the standardization of CDMA2000 [13], where these drawbacks are critical.) Yet, the improved algorithm [12] still requires two invocations for long messages. The Sandwich scheme affords a way to authenticate any message with just one invocation.

Table 1. Numbers of waste blocks and hash function calls

	Waste blocks	Hash function calls
Sandwich	1-2	1
HMAC	3	2

3 Hash Function Basics

Compression Function. A compression function f is a keyless function $f : \{0,1\}^{n+d} \to \{0,1\}^n$. The first n bits of the input to f are referred to as a "chaining variable," where as the last d bits of the input are referred to as a "data input" or "message block." Typical values of n and d are $(n,d) = (160, 512)$ for SHA-1 and $(n,d) = (256, 512)$ for SHA-256. Hereafter in this section we fix our choice of compression function f.

Merkle-Damgård Iteration. The Merkle-Damgård iteration allows us to extend the domain of f from $\{0,1\}^{n+d}$ to $\{0,1\}^{d*}$, the set of bit strings whose lengths are multiples of d bits. Namely, the function $F_{IV} : \{0,1\}^{d*} \to \{0,1\}^n$ is constructed as follows: Let $M \in \{0,1\}^{d*}$ and divide M into message blocks as $M = m_1 \| \cdots \| m_\ell$, $m_i \in \{0,1\}^d$. Then the hash value $F_{IV}(M)$ is defined by:

$$v_1 \leftarrow f(IV \| m_1),\ v_i \leftarrow f(v_{i-1} \| m_i) \text{ for } i = 2, \ldots, \ell,\ F_{IV}(M) \stackrel{\text{def}}{=} v_\ell,$$

where the initial vector $IV \in \{0,1\}^n$ is a pre-defined constant.

Padding. The current implementation of hash function is equipped with a padding so called the Merkle-Damgård strengthening. It is a padding method that takes the form of $M \| \pi(|M|) \in \{0,1\}^{d*}$ for messages $M \in \{0,1\}^{\leq \tilde{N}}$ whose lengths are at most $\tilde{N}$ bits (Note that the function π takes as its input the length $|M|$ in bits of the message M.) A typical value of $\tilde{N}$ is 2^{64}. The Merkle-Damgård iteration and strengthening are combined to yield the hash function $h : \{0,1\}^{\leq \tilde{N}} \to \{0,1\}^n$ by $h(M) \stackrel{\text{def}}{=} F_{IV}\big(M \| \pi(|M|)\big)$.

Dual Families. There are two ways of keying the compression function f. One is to key it via the first k bits of data input, yielding $f^{\triangledown}_K : \{0,1\}^{n+p} \to \{0,1\}^n$ with $K \in \{0,1\}^k$ and $p \stackrel{\text{def}}{=} d - k > 0$, precisely defined by $f^{\triangledown}_K(v \| z) \stackrel{\text{def}}{=} f(v \| K \| z)$ for $v \in \{0,1\}^n$, $z \in \{0,1\}^p$. The other way keys the chaining variable, yielding $f^{\triangleright}_{\check{K}} : \{0,1\}^d \to \{0,1\}^n$ with $\check{K} \in \{0,1\}^n$ defined by $f^{\triangleright}_{\check{K}}(m) \stackrel{\text{def}}{=} f(\check{K} \| m)$ for $m \in \{0,1\}^d$. If we are allowed to call only h (and not f), then we do not have direct access to the chaining variable. Hence $f^{\triangledown}$ appears explicitly whenever we try to key, whereas $f^{\triangleright}$ appears only implicitly for the purpose of security analysis.

Other Keyed Families. The Merkle-Damgård iteration F_{IV} can be also (implicitly) keyed, by replacing the initial vector IV with a key $\check{K} \in \{0,1\}^n$. This gives us a function family $\big\{F_{\check{K}} : \{0,1\}^{d*} \to \{0,1\}^n\big\}$. We then extend the domain

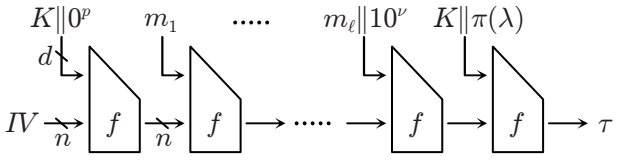

Fig. 1. Sandwich scheme, basic version

$\{0,1\}^{d*}$ to $\{0,1\}^*$ via the trivial padding $M\|10^\nu$, where $\nu \stackrel{\text{def}}{=} d-(|M| \bmod d)-1$ (We view ν as a function of M and often write $\nu(M)$ to denote this quantity.) This defines another function family $\left\{\bar{F}_{\check{K}} : \{0,1\}^* \to \{0,1\}^n\right\}$ via $\bar{F}_{\check{K}}(M) \stackrel{\text{def}}{=} F_{\check{K}}(M\|10^\nu)$.

4 Our Contribution

Figure 1 depicts the basic construction of the Sandwich scheme (We call it "basic," because later in Sect. 8, 9 and 10 we introduce several derivatives with optimized filling or padding.) The basic Sandwich method S takes as its input a message M and a key $K \in \{0,1\}^k$ and lets the hash value $\tau = h(K\|0^p\|M\|10^\nu\|K)$ be the tag, with $\nu = \nu(M)$. The message M is divided into d-bit blocks as $M = m_1\| \ldots \|m_\ell$, where $\ell = \lceil(|M|+1)/d\rceil$ and m_ℓ is a bit string whose length varies from 0 (the null string) to $d-1$ bits. Note that the length of the data $K\|0^p\|M\|10^\nu\|K$ is $\lambda \stackrel{\text{def}}{=} d(\ell+1)+k$ bits, which is input to the padding function π, and we are assuming $|\pi(\lambda)| = p$. We view λ as a function of M and often write $\lambda(M)$ to denote this quantity. Now we have the basic Sandwich scheme $S_K : \{0,1\}^N \to \{0,1\}^n$, where $N = \tilde{N} - d - 1 - k$.

The main contribution of this paper is to show that the basic Sandwich approach S gives a secure, single-key MAC. More precisely, we prove that it yields a PRF-based MAC, under the conditions that $\pi(\lambda) \neq 0^p$ for any λ and that both $f^\triangledown$ and $f^\triangleright$ are PRFs.

5 Security Comparison to HMAC

The security of HMAC also relies on the pseudorandomness of $f^\triangledown$ and $f^\triangleright$ [11]. In order for these functions to be PRFs, they must resist adversary's queries to its oracles. In Table 2 we compare these numbers.

It should be noted that the two "2"s in Table 2 come from very different nature. The "2" in the Sandwich scheme has roots in collision resistance, whereas the "2" in HMAC originates from a key derivation. $f^\triangleright$ in the Sandwich method must resist two oracle queries m, m' of adversary's choice, while $f^\triangledown$ in HMAC only needs to resist constant queries $IV\|IPAD'$ and $IV\|OPAD'$. In this regard, HMAC is based on a weaker assumption.

Theoretically, there is no difference between the requirement that $f^\triangledown$ is a PRF and one that $f^\triangleright$ is a PRF, as long as $k = n$ (The difference is just which

Table 2. Numbers of oracle queries that compression function must resist

	$f^{\triangledown}$ (Keyed via message block)	$f^{\triangleright}$ (Keyed via chaining variable)
Sandwich	$q+1$	2
HMAC	2	q

bits of input are keyed). In practice, however, the nature of data input and that of chaining variable are quite dissimilar, for an adversary can directly access the former but not the latter. In fact, existing compression functions like SHA-1 and SHA-256 are designed so that data input and chaining variable are processed in completely separate procedures. It seems that we have to wait for further research [14,15] on existing compression functions to identify this difference in them.

Also, the coefficients in the security reduction of the Sandwich scheme are fundamentally the same as those in that of HMAC. The result given in Sect. 7 is essentially tight, due to the general "birthday attack" [16]. For more discussions on the exact tightness of this type of reduction, see [11].

6 Security Definitions

The notation $x \xleftarrow{\$} X$ denotes the operation of selecting an element x uniformly at random from a set X. An adversary is an algorithm A, possibly probabilistic, that may have access to a oracle. The notation $A^{\mathcal{O}} \Rightarrow x$ denotes the event that A with the indicated oracle outputs x. Oracles are often defined in a "game" style. We then write $A^{\mathcal{G}} \Rightarrow x$ to denote the event that A outputs x in the experiment of running A as specified in game $\mathcal{G}$.

PRFs. Any PRF is a secure MAC [17]. All the MACs that appear in this paper are PRF-based. Consider a function family $\{f_K : X \to Y\}_{K \in KEY}$. A prf-adversary A tries to distinguish between two oracles, one being $f_K(\cdot), K \xleftarrow{\$} KEY$ and the other being $f(\cdot), f \xleftarrow{\$} \{f : X \to Y\}$. Succinctly, define

$$\mathrm{Adv}_f^{\mathrm{prf}}(A) \stackrel{\mathrm{def}}{=} \Pr\left[A^{f_K, K \xleftarrow{\$}} \Rightarrow 1\right] - \Pr\left[A^{f \xleftarrow{\$}} \Rightarrow 1\right]$$

to be the prf-advantage of A against f.

cAU. The notion of "computationally Almost Universal (cAU)" measures a sort of collision resistance. An au-adversary A, given access to no oracle, just outputs a pair of messages $(M, M') \in X \times X$. Then define

$$\mathrm{Adv}_f^{\mathrm{au}}(A) \stackrel{\mathrm{def}}{=} \Pr\left[f_K(M) = f_K(M') \wedge M \neq M' \;\middle|\; A \Rightarrow (M, M'), K \xleftarrow{\$} KEY\right]$$

to be the au-advantage of A against f.

Resources. An adversary A's resources are measured in terms of the time complexity t, the number q of oracle queries and the length μ in bits of each

query. The time complexity t includes the total execution time of an overlying experiment (the maximum if more than one experiments are involved) plus the size of the code of A, in some fixed model of computation. We write $T_f(\mu)$ to denote the time needed for one computation of f on a input whose length is μ bits. For $* \in \{\mathrm{prf}, \mathrm{au}, \ldots\}$ we write

$$\mathrm{Adv}^*_f(t, q, \mu) \stackrel{\mathrm{def}}{=} \max \mathrm{Adv}^*_f(A),$$

where max is run over adversaries, each having time complexity at most t and making at most q oracle queries, each query of at most μ bits. One or more of the resources are often omitted from the notation if irrelevant in the context. In particular, we often omit the time complexity of an au-adversary A, due to the following lemma.

Lemma 1. *For any time complexity t, we have*

$$\mathrm{Adv}^{\mathrm{au}}_f(t, \mu) \leq \mathrm{Adv}^{\mathrm{au}}_f(2 \cdot T_f(\mu), \mu).$$

Proof. Let A be an au-adversary against f that has time complexity at most t and outputs messages of at most μ bits each. By definition we have

$$\mathrm{Adv}^{\mathrm{au}}_f(A) = \sum_{M, M'} \Big(\Pr\Big[\{M, M'\} = \{\bar{M}, \bar{M}'\} \,\Big|\, A \Rightarrow (\bar{M}, \bar{M}') \Big] \times \Big[f_K(M) = f_K(M') \,\Big|\, K \stackrel{\$}{\leftarrow} KEY \Big] \Big),$$

where the summation is over all pairs $\{M, M'\}$ of two distinct messages whose lengths are at most μ bits each. Hence, there exists a pair (M, M') of distinct messages such that $\mathrm{Adv}^{\mathrm{au}}_f(A) \leq \Pr[f_K(M) = f_K(M')]$. Then we can create a new adversary B that has M, M' hardwired as a part of its code and simply outputs these messages. □

7 Security Proof of the Basic Construction

The following theorem states the security result of the basic Sandwich scheme.

Theorem 1. *Let $f : \{0,1\}^{n+p} \to \{0,1\}^n$ be a compression function and $S_K : \{0,1\}^{\leq N} \to \{0,1\}^n$ the basic Sandwich scheme constructed from f, as described in Sect. 4. Then the basic Sandwich scheme S is a PRF, provided that both $f^{\triangledown}$ and $f^{\triangleright}$ are PRFs. More formally, we have*

$$\mathrm{Adv}^{\mathrm{prf}}_S(t, q, \mu) \leq \mathrm{Adv}^{\mathrm{prf}}_{f^{\triangledown}}(t, q+1) + \binom{q}{2} \cdot \left(\left(2 \cdot \left\lceil \frac{\mu}{d} \right\rceil + 1 \right) \cdot \mathrm{Adv}^{\mathrm{prf}}_{f^{\triangleright}}(t', 2) + \frac{1}{2^n} \right),$$

where $t' = 4 \cdot \lceil (\mu/d) + 1 \rceil \cdot T_f$.

The following three lemmas prove the above theorem.

Adversary B	**Adversary** C
Query $IV\|0^p$ to oracle $f^{\triangledown}$ and obtain $\check{K} = f^{\triangledown}(IV\|0^p)$ Run A; On A's query M do: Compute $v \leftarrow \bar{F}_{\check{K}}(M)$ Query $v\|\pi(\lambda(M))$ to oracle $f^{\triangledown}$ and obtain $\tau = f^{\triangledown}\big(v\|\pi(\lambda)\big)$ Reply τ to A Output whatever A outputs	$s \leftarrow 0$; $\tau_1, \ldots, \tau_q \xleftarrow{\$} \{0,1\}^n$ $i \xleftarrow{\$} \{1, \ldots, q-1\}$; $j \xleftarrow{\$} \{i+1, \ldots, q\}$ Run A; On A's query M do: $s \leftarrow s + 1$ $M_s \leftarrow M$ Reply τ_s to A Output (M_i, M_j)

Fig. 2. Description of adversaries B and C

Game $\mathcal{G}$	**Game** $\mathcal{G}'$
$f^{\triangledown} \xleftarrow{\$} \big\{f : \{0,1\}^{n+p} \to \{0,1\}^n\big\}$ $\check{K} \leftarrow f^{\triangledown}(IV\|0^p)$ On query M reply $f^{\triangledown}\big(\bar{F}_{\check{K}}(M)\|\pi(\lambda(M))\big)$	$f^{\triangledown} \xleftarrow{\$} \big\{f : \{0,1\}^{n+p} \to \{0,1\}^n\big\}$ $\check{K} \xleftarrow{\$} \{0,1\}^n$ On query M reply $f^{\triangledown}\big(\bar{F}_{\check{K}}(M)\|\pi(\lambda(M))\big)$

Fig. 3. Intermediate games $\mathcal{G}$ and $\mathcal{G}'$

Lemma 2. *If $f^{\triangledown}$ is a PRF and $\bar{F}$ (constructed from f as in Sect. 3) is cAU, then the basic Sandwich scheme S is a PRF. More formally, we have*

$$\mathrm{Adv}^{\mathrm{prf}}_{S}(t, q, \mu) \leq \mathrm{Adv}^{\mathrm{prf}}_{f^{\triangledown}}(t, q+1) + \binom{q}{2} \cdot \mathrm{Adv}^{\mathrm{au}}_{\bar{F}}(\mu).$$

Proof. Let A be a prf-adversary against S that has time complexity at most t and makes at most $q \geq 2$ oracle queries, each of at most μ bits. We shall construct a prf-adversary B against $f^{\triangledown}$ and an au-adversary C against $\bar{F}$, each using A as a subroutine, as described in Fig. 2. Note that B has time complexity at most t and makes at most $q+1$ oracle queries, and C outputs two messages, each of at most μ bits. We show that

$$\mathrm{Adv}^{\mathrm{prf}}_{S}(A) \leq \mathrm{Adv}^{\mathrm{prf}}_{f^{\triangledown}}(B) + \binom{q}{2} \cdot \mathrm{Adv}^{\mathrm{au}}_{\bar{F}}(C).$$

Let $\mathcal{G}$, $\mathcal{G}'$ be two games defined in Fig. 3. These games define oracles for the adversary A.

Claim. We have

$$\begin{aligned}\mathrm{Adv}^{\mathrm{prf}}_{f^{\triangledown}}(B) &\stackrel{\mathrm{def}}{=} \Pr\Big[B^{f^{\triangledown}_K, K \xleftarrow{\$}} \Rightarrow 1\Big] - \Pr\Big[B^{f^{\triangledown} \xleftarrow{\$}} \Rightarrow 1\Big] \\ &= \Pr\Big[A^{S_K, K \xleftarrow{\$}} \Rightarrow 1\Big] - \Pr\Big[A^{\mathcal{G}} \Rightarrow 1\Big].\end{aligned}$$

Proof. If oracle $f^{\triangledown}$ to B is given by $f_K^{\triangledown}, K \xleftarrow{\$} \{0,1\}^k$, then observe that B correctly simulates the oracle $S_K, K \xleftarrow{\$} \{0,1\}^k$ for A. Hence $\Pr\left[B^{f_K^{\triangledown},K\xleftarrow{\$}} \Rightarrow 1\right] = \Pr\left[A^{S_K,K\xleftarrow{\$}} \Rightarrow 1\right]$. On the other hand, if oracle $f^{\triangledown}$ to B is given by $f^{\triangledown} \xleftarrow{\$} \left\{f : \{0,1\}^{n+p} \to \{0,1\}^n\right\}$, then running B^A exactly corresponds to running $A^{\mathcal{G}}$. Thus $\Pr\left[B^{f^{\triangledown}\xleftarrow{\$}} \Rightarrow 1\right] = \Pr\left[A^{\mathcal{G}} \Rightarrow 1\right]$.

Claim. Game $\mathcal{G}$ is equivalent to Game $\mathcal{G}'$.

Proof. Recall that we assume the condition $\pi(\lambda(M)) \neq 0^p$ for every M. Hence in Game $\mathcal{G}$ we have $\bar{F}_{\check{K}}(M)\|\pi(\lambda(M)) \neq IV\|0^p$ for every query M, and while replying to A's queries the random function $f^{\triangledown}$ is never invoked on the input value $IV\|0^p$. This means that in Game $\mathcal{G}$ the key $\check{K} = f^{\triangledown}(IV\|0^p)$ is a random value independent from A's queries, and the equivalence to Game $\mathcal{G}'$ follows.

Now we assume, without loss of generality, that the adversary A never repeats a query and that the total number of A's queries is always exactly q rather than at most q, no matter how replies to A's queries are made. Let $M_1, \ldots, M_q$ represent A's queries in order.

Let E be the event that $\bar{F}_{\check{K}}(M_i)\|\pi(\lambda(M_i)) = \bar{F}_{\check{K}}(M_j)\|\pi(\lambda(M_j))$ occurs for some $1 \leq i < j \leq q$. Observe that as long as E does not occur, Game $\mathcal{G}'$ for A and running A with the oracle $S \xleftarrow{\$} \left\{S : \{0,1\}^{\leq N} \to \{0,1\}^n\right\}$ proceed exactly the same. Therefore, by the Fundamental Lemma of Game Playing [18], we obtain

$$\Pr\left[A^{\mathcal{G}'} \Rightarrow 1\right] - \Pr\left[A^{S\xleftarrow{\$}} \Rightarrow 1\right] \leq \Pr[E].$$

Claim. We have

$$\Pr[E] \leq \binom{q}{2} \cdot \mathrm{Adv}^{\mathrm{au}}_{\bar{F}}(C).$$

Proof. Let E' denote the event that $\bar{F}_{\check{K}}(M_i) = \bar{F}_{\check{K}}(M_j)$ for some $1 \leq i < j \leq q$, so that $\Pr[E] \leq \Pr[E']$. For $1 \leq \alpha < \beta \leq q$ let $E'_{\alpha,\beta}$ denote the event that $\bar{F}_{\check{K}}(M_\alpha) = \bar{F}_{\check{K}}(M_\beta)$ occurs while $\bar{F}_{\check{K}}(M_{\bar{\alpha}}) \neq \bar{F}_{\check{K}}(M_{\bar{\beta}})$ for all $1 \leq \bar{\alpha} < \bar{\beta} < \beta$. Notice that the events $E'_{\alpha,\beta}$ for $1 \leq \alpha < \beta \leq q$ are disjoint and $E' = \bigvee_{1\leq\alpha<\beta\leq q} E'_{\alpha,\beta}$. Then

$$\begin{aligned}
\mathrm{Adv}^{\mathrm{au}}_{\bar{F}}(C) &\geq \Pr\left[\bigvee_{1\leq\alpha<\beta\leq q} E'_{\alpha,\beta} \wedge (i,j) = (\alpha,\beta)\right] \\
&= \sum_{1\leq\alpha<\beta\leq q} \Pr\left[E'_{\alpha,\beta} \wedge (i,j) = (\alpha,\beta)\right] \\
&= \sum_{1\leq\alpha<\beta\leq q} \Pr\left[E'_{\alpha,\beta}\right] \cdot \Pr\left[(i,j) = (\alpha,\beta)\right] \\
&= \frac{1}{\binom{q}{2}} \sum_{1\leq\alpha<\beta\leq q} \Pr\left[E'_{\alpha,\beta}\right] = \frac{1}{\binom{q}{2}} \Pr[E'] \geq \frac{1}{\binom{q}{2}} \Pr[E].
\end{aligned}$$

Now we see that

$$\begin{aligned}\mathrm{Adv}^{\mathrm{prf}}_{S}(A) &\stackrel{\mathrm{def}}{=} \Pr\left[A^{S_K, K\xleftarrow{\$}} \Rightarrow 1\right] - \Pr\left[A^{S\xleftarrow{\$}} \Rightarrow 1\right] \\ &= \Pr\left[A^{S_K, K\xleftarrow{\$}} \Rightarrow 1\right] - \Pr\left[A^{\mathcal{G}} \Rightarrow 1\right] + \Pr\left[A^{\mathcal{G}'} \Rightarrow 1\right] - \Pr\left[A^{S\xleftarrow{\$}} \Rightarrow 1\right] \\ &\leq \mathrm{Adv}^{\mathrm{prf}}_{f^{\triangledown}}(B) + \binom{q}{2} \cdot \mathrm{Adv}^{\mathrm{au}}_{\bar{F}}(C).\end{aligned}$$

□

Lemma 3. *Let $f : \{0,1\}^{n+d} \to \{0,1\}^n$ be a compression function. If F (constructed from f as in Sect. 3) is cAU, then so is $\bar{F}$. More formally, we have*

$$\mathrm{Adv}^{\mathrm{au}}_{\bar{F}}(t, \mu) \leq \mathrm{Adv}^{\mathrm{au}}_{F}(t, \mu + d).$$

Proof. Let A be an au-adversary against $\bar{F}$ that has time complexity at most t and outputs messages (M, M') of at most μ bits each. Then we can easily construct an au-adversary B against F, by letting B simply output the pair $(M\|10^{\nu(M)}, M'\|10^{\nu(M')})$. Note that $M \neq M'$ implies $M\|10^{\nu(M)} \neq M'\|10^{\nu(M')}$. □

Lemma 4. *Let $f : \{0,1\}^{n+d} \to \{0,1\}^n$ be a compression function. If $f^{\triangleright}$ is a PRF, then F (constructed from f as in Sect. 3) is cAU. More formally, we have*

$$\mathrm{Adv}^{\mathrm{au}}_{F}(t, \mu) \leq \left(2 \cdot \left\lceil \frac{\mu}{d} \right\rceil - 1\right) \cdot \mathrm{Adv}^{\mathrm{prf}}_{f^{\triangleright}}(t', 2) + \frac{1}{2^n},$$

where $t' = t + 2 \cdot \lceil \mu/d \rceil \cdot T_f$, T_f being the time for one evaluation of f.

Proof. This result is obtained in [11]. □

Now from the above lemmas we have

$$\begin{aligned}\mathrm{Adv}^{\mathrm{prf}}_{S}(t, q, \mu) &\leq \mathrm{Adv}^{\mathrm{prf}}_{f^{\triangledown}}(t, q+1) + \binom{q}{2} \cdot \mathrm{Adv}^{\mathrm{au}}_{\bar{F}}(\mu) \\ &\leq \mathrm{Adv}^{\mathrm{prf}}_{f^{\triangledown}}(t, q+1) + \binom{q}{2} \cdot \mathrm{Adv}^{\mathrm{au}}_{F}(\mu + d) \\ &\leq \mathrm{Adv}^{\mathrm{prf}}_{f^{\triangledown}}(t, q+1) + \binom{q}{2} \cdot \left(\left(2 \cdot \left\lceil \frac{\mu}{d} \right\rceil + 1\right) \cdot \mathrm{Adv}^{\mathrm{prf}}_{f^{\triangleright}}(t', 2) + \frac{1}{2^n}\right),\end{aligned}$$

where $t' = 2 \cdot T_{\bar{F}}(\mu) + 2 \cdot \lceil (\mu/d) + 1 \rceil \cdot T_f \leq 4 \cdot \lceil (\mu/d) + 1 \rceil \cdot T_f$. This proves Theorem 1.

8 Variant A: Reducing the First Filling 0^p

The filling 0^p after the first key K may be considered as consuming, particularly if p is large. Figure 4 describes a variant of the basic Sandwich scheme, which

uses a one-bit filling 0 rather than 0^p. Note that in this variant a message M is now divided into blocks as $M = m_1 \| m_2 \| \dots \| m_\ell$ with $|m_1| = p-1$, $|m_2| = \dots = |m_{\ell-1}| = d$ and $0 \le |m_\ell| \le d-1$. In case $|M| \le p-2$ the entire message $M = m_1$ is processed by the very first block (In this variant the condition $\pi(\mu) \neq 0^{p-1}$ is not required. Also, the functions ν and λ and the number N are re-defined accordingly.)

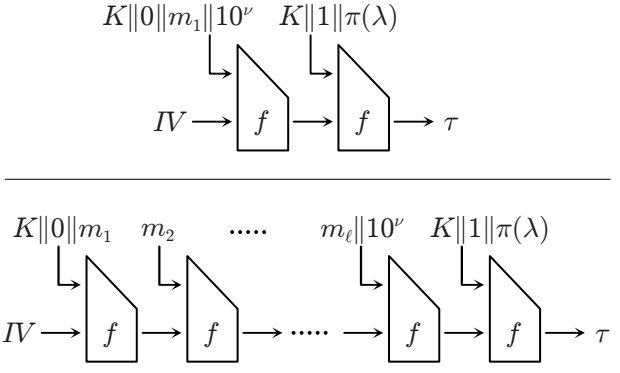

Fig. 4. Variant A: Reducing the filling 0^p to 0

This variant is secure. However, its analysis is more complex than that of the basic construction. The proof of security requires novel techniques that are not included in that of the basic version. Intuitively, this is because now an adversary can change the value of $\check{K}$ via querying different m_1. Owing to the pseudorandomness of $f^{\triangledown}$ (and appropriate fillings 0 and 1 after the key K), for different $m_1, m_1', m_1'', \dots$ the adversary "sees" independently random keys $\check{K}, \check{K}', \check{K}'', \dots$ (This, however, demands that $f^{\triangledown}$ be resistant against $2q$ oracle queries rather than $q+1$.) Now the difficulty lies in the treatment of the event that a "collision" is detected. Observe that there can be two different cases for a collision. One is with the same key as in $\bar{F}_{\check{K}}(M) = \bar{F}_{\check{K}}(M')$ with $M \neq M'$, and the other with different keys as in $\bar{F}_{\check{K}}(M) = \bar{F}_{\check{K}'}(M')$ (and not necessarily $M \neq M'$.) The first case can be handled in the same way as in the basic version. The problem is that we also have to bound the latter probability by the pseudorandomness of $f^{\triangleright}$.

We deal with this problem along the course of prefix-free PRFs and multi-oracle families [5]. Recall that F is a prefix-free PRF if $f^{\triangleright}$ is a PRF. Next we extend the result of multi-oracle families in [5] from PRFs to prefix-free PRFs. We can then bound the collision probability by the multi-oracle family of F. This does not affect the query number for $f^{\triangleright}$ (It still remains to be 2) but worsens the coefficient roughly by a factor of 2. We state the result concretely in the following theorem.

Theorem 2. *Let $f : \{0,1\}^{n+d} \to \{0,1\}^n$ be a compression function and $S_K : \{0,1\}^{\leq N} \to \{0,1\}^n$ the Variant A constructed from f. Then we have*

$$\mathrm{Adv}^{\mathrm{prf}}_{S}(t,q,\mu) \leq \mathrm{Adv}^{\mathrm{prf}}_{f^{\triangledown}}(t,2q) + \binom{q}{2} \cdot \left(4 \cdot \left(\left\lceil \frac{\mu}{d} \right\rceil + 2\right) \cdot \mathrm{Adv}^{\mathrm{prf}}_{f^{\triangleright}}(t',2) + \frac{1}{2^n}\right),$$

where $t' = t + 2q \cdot \lceil \mu/d \rceil \cdot T_f$.

9 Variant B: Improving the Second Filling 10^ν

We go back to the basic Sandwich construction and discuss how to avoid the waste that occurs when the message size $|M|$ happens to be exactly equal to a multiple of d bits. Note that in such a case, the filling bits $1\|0^{d-1}$ is appended after the message M, producing an extra one block of compression function. We show a technique to get rid of this increase.

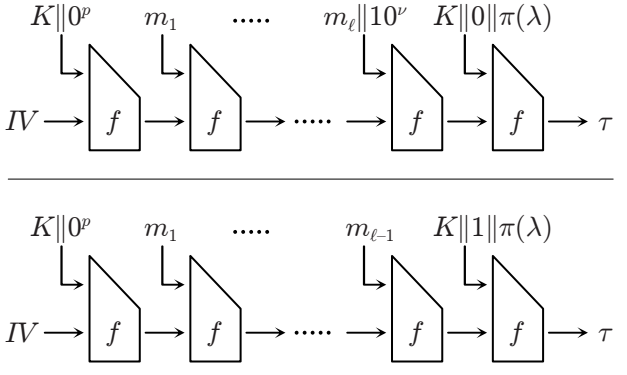

Fig. 5. Variant B: Case m_ℓ is not null (upper) and m_ℓ null (lower)

The technique works as follows: If the message size $|M|$ is not equal to a multiple of d, then the usual filling $1\|0^\nu$ is appended after the message M, and then the key K is appended, followed by $0\|\pi(\lambda)$. On the other hand, if the message size $|M|$ happens to be exactly a multiple of d, then *no* filling is appended after the message M; instead, we directly append the key K after the message M and then append the padding $1\|\pi(\lambda)$ (Again, the function λ and the number N are re-defined accordingly, and we assume $\pi(\lambda) \neq 0^{p-1}$ for all λ in this variant.)

This variant is also secure, and the proof of security does not require much modification to that of the basic version. So let us review the reduction proofs and see this new scheme actually preserves the security. First, the construction of adversary B naturally transforms into the new setting. The equivalence between Game $\mathcal{G}$ and Game $\mathcal{G}'$ still holds, for we assume $\pi(\lambda) \neq 0^{p-1}$, and hence $\check{K}$ is a random value independent from A's queries. A collision on the input value

for $f^{\triangleright}$ can be divided into two cases in accordance with the padding $K\|0$ and $K\|1$, but both cases are bounded by $\mathrm{Adv}^{\mathrm{au}}_{\bar{F}}$. So there is no degradation in the reduction:

Theorem 3. *Let $f : \{0,1\}^{n+d} \to \{0,1\}^n$ be a compression function and $S_K : \{0,1\}^{\leq N} \to \{0,1\}^n$ the Variant B constructed from f. Then*

$$\mathrm{Adv}^{\mathrm{prf}}_{S}(t,q,\mu) \leq \mathrm{Adv}^{\mathrm{prf}}_{f^{\triangledown}}(t,q+1) + \binom{q}{2} \cdot \left(\left(2 \cdot \left\lceil \frac{\mu}{d} \right\rceil + 1 \right) \cdot \mathrm{Adv}^{\mathrm{prf}}_{f^{\triangleright}}(t',2) + \frac{1}{2^n} \right),$$

where $t' = 4 \cdot \lceil (\mu/d) + 1 \rceil \cdot T_f$.

We can extend this idea to gain further improvement, if there is enough "room" in the last block. Namely, let σ be the maximum number such that $K\|1\|m_\ell\|\pi(\lambda)$ fits in the last block with $m_\ell \in \{0,1\}^\sigma$ (Again, the number λ is re-defined accordingly.) For a message M with $|m_\ell| > \sigma$, we use the first case $m_\ell\|10^\nu\|K\|0\|\pi(\lambda)$ as is in the last two blocks. On the other hand, if $|m_\ell| \leq \sigma$ (including the case m_ℓ null), then we process the data $K\|1\|m_\ell\|\pi(\lambda)$ with only one computation in the very last block (and in the latter case note that for $m_\ell \neq m'_\ell$ we require $m_\ell\|\pi(\lambda) \neq m'_\ell\|\pi(\lambda')$.)

10 Variant C: Handling the Last Padding $\pi(\lambda)$

In this section we study the case where the block size d is too small to accommodate both the key K and padding $\pi(\lambda)$ in one block. The purpose of introducing this variant is twofold. One is to show the general applicability of the Sandwich approach with a low-ratio compression function. The other is to point out the powerfulness of the multi-oracle family techniques that we also used in Sect. 8.

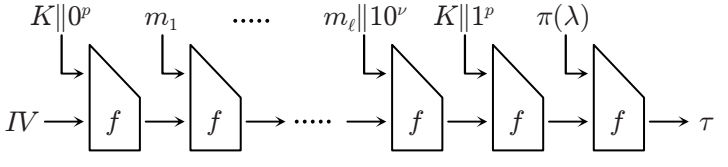

Fig. 6. Variant C: Padding with low-ratio compression function

The difference between this variant and the basic version is in the last padding. In this variant we use two blocks in order to process the second key K and the padding $\pi(\lambda)$ via $K\|1^p\|\pi(\lambda)$ (This, of course, does not provide any improvement in efficiency, and again, the function λ and the number N are re-defined.)

This variant is also secure, but the difficulty in analysis lies in the very last block. That is, we can "extract" a prf-adversary B against $f^{\triangledown}$ and an au-adversary C against $\bar{F}$ as in Sect. 7, but there still remains a "gap" (The gap arises from the very last block.) We have to fill in this gap somehow by the

pseudorandomness of $f^{\triangleright}$. We do this via the multi-oracle family of $f^{\triangleright}$. This does not increase the query number "2," and the degradation in the reduction is only minor:

Theorem 4. *Let $f : \{0,1\}^{n+d} \to \{0,1\}^n$ be a compression function and $S_K : \{0,1\}^{\leq N} \to \{0,1\}^n$ the Variant C constructed from f. Then*

$$\mathrm{Adv}^{\mathrm{prf}}_{S}(t,q,\mu) \leq \mathrm{Adv}^{\mathrm{prf}}_{f^{\triangledown}}(t,q+1) + \binom{q}{2} \cdot \left(\left(2 \cdot \left\lceil \frac{\mu}{d} \right\rceil + 1 \right) \cdot \mathrm{Adv}^{\mathrm{prf}}_{f^{\triangleright}}(t',2) + \frac{1}{2^n} \right) + q \cdot \mathrm{Adv}^{\mathrm{prf}}_{f^{\triangleright}}(t'',1),$$

where $t' = 4 \cdot \lceil (\mu/d) + 1 \rceil \cdot T_f$ and $t'' = t + 2q \cdot T_f$.

11 Concluding Remarks

The Sandwich approach offers a secure, single-key MAC which is more efficient than HMAC. The improvement in performance becomes beneficial especially for situations with severe resource requirements and/or with short messages. For short messages, the optimization techniques in variants A and B are quite effective. Any combination of the three variations A, B and C would work, provided that appropriate filling and padding methods are devised and used with.

The security reduction of the Sandwich scheme, in theory, is roughly equivalent to that of HMAC. They both rely on the pseudorandomness of $f^{\triangledown}$ and $f^{\triangleright}$. The difference between the requirement of $f^{\triangledown}$ being a PRF and that of $f^{\triangleright}$ a PRF would result in a difference between the security of the Sandwich scheme and that of HMAC. Thus in reality we have to wait for further research on existing hash functions like SHA-1 and SHA-256 in order to analyze how they satisfy the two requirements and to identify the differences.

Lastly, we remark that the key recovery attack known for previous hybrid and envelope MACs no longer applies to the Sandwich scheme presented here. A straight-forward observation tells us that a key recovery against the Sandwich scheme essentially amounts to the key recovery against $f^{\triangledown}$.

Acknowledgments

The author would like to thank ACISP2007 anonymous referees for their helpful comments, references to [4,8] and suggestions to improve notation and terminology. The final revision work of this paper has benefited greatly from advisory comments made by Kazumaro Aoki, including the reference to [2].

References

1. NIST: Secure hash standard, FIPS PUB 180-2 (2002)
2. Kaliski, B., Robshaw, M.: Message authentication with MD5. CryptoBytes (The Technical Newsletter of RSA Laboratories) 1(1), 5–8 (1995)

3. Rogaway, P.: Formalizing human ignorance: Collision-resistant hashing without the keys. In: Nguyen, P.Q. (ed.) VIETCRYPT 2006. LNCS, vol. 4341, pp. 211–228. Springer, Heidelberg (2006)
4. Tsudik, G.: Message authentication with one-way hash functions. ACM Computer Communication Review 22(5), 29–38 (1992)
5. 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)
6. Metzger, P., Simpson, W.A.: IP authentication using keyed MD5. IETF, RFC 1828 (1995)
7. Metzger, P., Simpson, W.A.: IP authentication using keyed SHA. IETF, RFC 1852 (1995)
8. Preneel, B., van Oorschot, P.C.: On the security of two MAC algorithms. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 19–32. Springer, Heidelberg (1996)
9. Metzger, P., Simpson, W.A.: IP authentication using keyed SHA1 with interleaved padding (IP-MAC). IETF, RFC 2841 (2000)
10. Bellare, M., Canetti, R., Krawczyk, H.: Keying hash functions for message authentication. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 1–15. Springer, Heidelberg (1996)
11. 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)
12. 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)
13. TR45.AHAG: Enhanced cryptographic algorithms, revision B. TIA (2002)
14. 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: De Prisco, R., Yung, M. (eds.) SCN 2006. LNCS, vol. 4116, pp. 242–256. Springer, Heidelberg (2006)
15. 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)
16. Preneel, B., van Oorschot, P.C.: On the security of iterated message authentication codes. IEEE Transactions on Information Theory 45(1), 188–199 (1999)
17. Bellare, M., Goldreich, O., Mityagin, A.: The power of verification queries in message authentication and authenticated encryption. Cryptology ePrint Archive: Report 2004/304 (2004)
18. Bellare, M., Rogaway, P.: The security of triple encryption and a framework for code-based game-playing proofs. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 409–426. Springer, Heidelberg (2006)