Provably Secure MACs From Differentially-uniform Permutations and AES-based Implementations

Kazuhiko Minematsu[1] and Yukiyasu Tsunoo[1]

NEC Corporation, 1753 Shimonumabe, Nakahara-Ku, Kawasaki 211-8666, Japan
k-minematsu@ah.jp.nec.com

Abstract. We propose message authentication codes (MACs) that combine a block cipher and an additional (keyed or unkeyed) permutation. Our MACs are provably secure if the block cipher is pseudorandom and the additional permutation has a small differential probability. We also demonstrate that our MACs are easily implemented with AES and its 4-round version to obtain MACs that are provably secure and 1.4 to 2.5 times faster than the previous MAC modes of AES such as the CBC-MAC-AES.
Keywords: MAC, Block cipher, AES, Differentially-uniform permutation.

1 Introduction

Message Authentication Codes (MACs) are symmetric cryptographic functions that ensure the authenticities of messages. The CBC-MAC and its variants (such as EMAC [4], XCBC [8], and OMAC [16]) are well-known modes for block ciphers to provide MACs. They are provably secure and efficient; they operate at almost the same throughput as that of the underlying block cipher. However, what can we do if we want a MAC that is faster than these MAC modes to keep the additional implementation as small as possible? In other words, can we have a block cipher based MAC faster than the CBC-MAC?

In this paper, we give a solution to this problem. We propose MACs that combine a block cipher and its component, typically a reduced-round version of the block cipher. These kinds of MACs can easily be implemented on any platform where the block cipher has already been implemented. The additional program size would be quite small. A similar approach called ALRED [11] was recently proposed by Daemen and Rijmen. It was interesting because of its efficiency (in terms of authentication tag generation and preprocessing). It was also shown that ALRED was secure against some attacks. Compared to ALRED, our schemes are secure in a stronger security model: if one can distinguish our MACs from uniform random function, then the underlying block cipher can be distinguished from uniform random permutation. A MAC with this property is called a provably secure MAC.

Formally, our MACs combine an n-bit block pseudorandom function (PRF) and an n-bit *auxiliary permutation* (AXP), which is an unkeyed or keyed permutation. AXPs are naturally expected to be faster than the block cipher, since they do not need to be cryptographically strong: they are only required to be ϵ-*differentially-uniform*, i.e., their maximum differential probability (MDP) or maximum expected differential probability (MEDP) is at most ϵ. Since we have assumed that the AXP is derived from the block cipher we intend to use, not all block ciphers can be used for our MACs. However, a keyed permutation with small MEDP can be obtained as a reduced-round of well-designed block ciphers, since such permutations are essential components of the block ciphers that are secure against differential cryptanalysis. For example, the MEDP of the 4-round AES with independent round keys is very small [26, 18] and thus our proposals can be securely implemented using AES and 4-round AES.

We propose two approaches. They have different characteristics regarding the amount of preprocessing, memory consumption, and the speed for long and short messages. The first approach is based on the modified tree hash (MTH), which was proposed by Boesgaard et al. [9]. It was an improvement on the well-known tree hash [10, 27]. Although they used the Length Annotation (LA) [19] to handle variable message lengths, we demonstrate that this is redundant. Removing LA from MTH improves efficiency, particularly for short messages.

The second uses chaining of the block cipher and the AXP. This is similar to the CBC-MAC, but the block cipher is only called for every d message blocks, where d is a parameter that determines the amount of preprocessing and MAC speed. This scheme is provably secure if the AXP is ϵ-differentially-uniform (for small ϵ) and satisfies an additional weak condition.

If our MACs are built using AES and its 4-round, we have MACs 1.4 to 2.5 times faster than the CBC-MAC-AES, depending on the scheme we use. The key length is short (one block cipher key, K, or K and an additional one-block key), and only one block cipher keyscheduling is needed. Their preprocessing times are moderate (log-order of the message length for the first approach, and constant for the second). We also show a software implementation of our MACs and comparisons between other MACs.

2 Preliminaries

Notations. $\{0, 1\}$ is denoted by Σ and n-bit space is denoted by Σ^n. $(\Sigma^n)^{\leq m}$ denotes the set of binary sequences whose lengths are multiples of n and at most nm. $(\Sigma^n)^+$ is the set of all binary sequences whose lengths are multiples of n, and Σ^* is the set of all finite-length binary sequences. If X is distributed independently and uniformly over set $\mathcal{X}$, we write $X \in_{\mathrm{U}} \mathcal{X}$. If F is a keyed function with domain $\mathcal{X}$, and range $\mathcal{Y}$, and key $K \in_{\mathrm{U}} \mathcal{K}$, then we write $F : \mathcal{X} \to \mathcal{Y}$ and there is function $f : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ such that $\Pr[F(x) = y] = |\{k \in \mathcal{K} : f(k, x) = y\}|/|\mathcal{K}|$. If we want to emphasize that F's key is K, F_K is written and if K is fixed to k, then F_k denotes function $f(k, *)$. Keyed and fixed functions are written by upper and lower case letters, respectively.

Definition 1. *Keyed function $F \in_{\mathrm{U}} \{f : \Sigma^n \to \Sigma^m\}$ is called an n-bit to m-bit uniformly random function (URF) and denoted by $R_{n,m}$. If F is uniform over all*

n-bit permutations, it is called an n-bit uniformly random permutation (URP) and denoted by P_n. Specifically, $R_{*,n}$ denotes the Variable-Input-Length (VIL)-URF such that $R_{*,n} \in_{\mathrm{U}} \{f : \Sigma^* \to \Sigma^n\}$. Here, VIL means that it accepts inputs of all lengths.

We express the elements of field $\mathrm{GF}(2^n)$ by the n-bit coefficient vectors of the polynomials in the field. We alternatively represent n-bit coefficient vectors by integers $0, 1, \ldots, 2^n - 1$, e.g., 2 corresponds to the coefficient vector $(00\ldots010)$ and 1 denotes $(00\ldots01)$, i.e., the identity element.

Definition 2. *Let f be a permutation on group $\mathcal{X}$ and F_K be a keyed permutation on $\mathcal{X}$ with key $K \in_{\mathrm{U}} \mathcal{K}$. The maximum differential probability (MDP) for f, denoted by $\mathrm{MDP}(f)$, is $\max_{a\neq 0,b} \Pr(f(X) - f(X + a) = b)$, where $X \in_{\mathrm{U}} \mathcal{X}$. Similarly, the maximum expected differential probability (MEDP) of F_K is defined as $\mathrm{MEDP}(F_K) \stackrel{\mathrm{def}}{=} \max_{a\neq 0,b} \Pr(F_K(X) - F_K(X + a) = b)$, which can also be written as $\max_{a\neq 0,b} \sum_{k\in\mathcal{K}} \Pr(F_k(X) - F_k(X + a) = b)/|\mathcal{K}|$.*

If $\mathcal{X}$ is a field with characteristic 2 (say, $\mathrm{GF}(2^n)$), then the addition and subtraction in Def. 2 correspond to the bitwise XOR operation, i.e., $\oplus$. In this case, MDP is always no less than $2/2^n$. However, this does not hold true for MEDP.

Definition 3. *Let H be a keyed function: $(\Sigma^n)^{\leq l} \to \Sigma^n$. The maximum collision probability of H for a pair of m-block and m'-block input is defined as*

$$\mathrm{Coll}_H(m, m') \stackrel{\mathrm{def}}{=} \max_{x\in(\Sigma^n)^m, x'\in(\Sigma^n)^{m'}, x\neq x'} \Pr(H(x) = H(x')), \text{ where } m, m' \leq l.$$

If the collision probability is no more than ϵ for all possible inputs, it is called an ϵ-almost universal (ϵ-AU) hash function.

Security notions. We used a standard security notion for symmetric cryptography [5, 6, 13].

Definition 4. *Let F and G be two keyed functions. Let us assume that the oracle has implemented H, which is identical to one of F or G. An adversary, $\mathcal{A}$, guesses if H is F or G using Chosen-plaintext attack (CPA). The maximum CPA-advantage in distinguishing F from G is defined as*

$$\mathtt{Adv}^{\mathtt{cpa}}_{F,G}(q, t, \sigma) \stackrel{\mathrm{def}}{=} \max_{\mathcal{A}:(q,t,\sigma)\text{-CPA}} \left|\Pr(\mathcal{A}^F = 1) - \Pr(\mathcal{A}^G = 1)\right|, \tag{1}$$

where $\mathcal{A}^F = 1$ denotes that $\mathcal{A}$'s guess is 1, which indicates one of F or G, and (q, t, σ)-CPA denotes a CPA that uses q queries with time complexity t, and the total length of q queries is at most σn bits. Instead of σ, we can use ρ, which denotes the maximum length (in n-bit block) of each query, to limit the adversary's resources. We omit σ and ρ if F and G have fixed input length, since they have been determined from q in this case. Also, we omit t if we consider attacks without computational restrictions. Especially, if $F : \Sigma^m \to \Sigma^n$ we have $\mathtt{Adv}^{\mathtt{prf}}_F(q, t) \stackrel{\mathrm{def}}{=} \mathtt{Adv}^{\mathtt{cpa}}_{F,R_{m,n}}(q, t)$. Similarly, if F is an n-bit keyed permutation, then $\mathtt{Adv}^{\mathtt{prp}}_F(q, t) \stackrel{\mathrm{def}}{=} \mathtt{Adv}^{\mathtt{cpa}}_{F,P_n}(q, t)$. Finally, if F is a VIL keyed function:$\Sigma^* \to \Sigma^n$, then $\mathtt{Adv}^{\mathtt{vilprf}}_F(q, t, \kappa) \stackrel{\mathrm{def}}{=} \mathtt{Adv}^{\mathtt{cpa}}_{F,R_{*,n}}(q, t, \kappa)$, where κ is σ or ρ.

If $\mathtt{Adv}^{\mathtt{cpa}}_{F,R_{m,n}}(q,t)$ is small for any sufficiently large q and t, F is called the pseudorandom function (PRF) [13]. The pseudorandom permutation (PRP) and VIL-PRF are defined similarly. As a VIL-PRF is also a secure VIL-MAC (e.g., see Proposition 2.7 of [5]), we focus on building VIL-PRFs.

3 Basic Idea

Let g be a (possibly keyed) function with n-bit domain and X be an n-bit random variable. Then, $g^{\oplus X}$ denotes a function such that $g^{\oplus X}(a) = g(a \oplus X)$. All our MACs are based on the following function.

Lemma 1. *We define the Add-Permute-Add (APA) function*: $(\Sigma^n)^{\le 2} \to \Sigma^n$ *as follows.*

$$\mathrm{APA}_{K,F}(x) = \begin{cases} x & \text{if } x \in \Sigma^n, \\ F^{\oplus K}(x_1) \oplus x_2 & \text{if } x = (x_1, x_2) \in (\Sigma^n)^2, \end{cases} \tag{2}$$

where $K \in_{\mathrm{U}} \Sigma^n$ *and* F *is an* n*-bit (keyed or fixed) permutation. Then,* $\mathrm{APA}_{K,F}$ *is* ϵ*-AU if* F*'s MEDP (for the case of keyed permutation) or MDP (for the case of fixed permutation) is at most* ϵ*.*

Proof. Let $x = (x_1, x_2)$ and $x' = (x'_1, x'_2)$ be two different inputs to $\mathrm{APA}_{K,F}$. If $x_1 \ne x'_1$, then the output collision means $F^{\oplus K}(x_1) \oplus F^{\oplus K}(x'_1) = x_2 \oplus x'_2$, which cannot occur with a probability larger than ϵ. If $x_1 = x'_1$, which implies $x_2 \ne x'_2$, then clearly the collision probability is zero. Moreover, the probability of $x_1 = F^{\oplus K}(x'_1) \oplus x'_2$ is $1/2^n$, since F is invertible. Thus, the maximum collision probability is at most ϵ.

An example of a permutation with small MDP is the following.

Example 1. Let inv be an n-bit permutation such that $\mathrm{inv}(x) = x^{-1}$ for $x \ne 0$ and $\mathrm{inv}(0) = 0$, where x^{-1} satisfies $x \cdot x^{-1} = 1$. Then, $\mathrm{MDP}(\mathrm{inv}) = 4/2^n$ [25].

As this example shows, a fixed permutation with small MDP does exist. However, this cannot be efficiently implemented if input is large (say, more than 32-bit). In contrast, a keyed permutation with small MEDP is more practical. For example, the 4-round AES with independent round keys has an MEDP of less than 2^{-113} [18]. In later sections, this fact enables us to implement our MACs using AES.

As stated in the Introduction, all our MACs combine an n-bit block cipher, E_K, and an n-bit additional keyed or fixed permutation, which is called the auxiliary permutation (AXP). If the AXP has a key, we denote it by G_U, where key $U \in_{\mathrm{U}} \mathcal{U}$. The sequence of m AXPs are denoted by $\mathbf{G} = (G_{U_1}, \ldots, G_{U_m})$. We will abbreviate G_{U_i} to G_i unless it is confusing. Hereafter, we will usually assume that the AXP is a keyed permutation. Since a fixed permutation can be seen as a keyed permutation with a single-point key space, this provides general descriptions of our schemes.

4 Building Variable Input Length Universal Hash

4.1 Modified Tree Hash

As Lemma 1 shows, a double input length AU hash function can be built using one invocation of a differentially-uniform permutation and an n-bit random key. The simplest way to expand the input length of this AU hash is using the well-known tree hash. The original tree hash proposed by Wegman and Carter [27] required some redundant calls of AU hash when the length of an input was not $2^l n$ for some positive integer l. An improvement to remove these redundant calls was proposed by Boesgaard et al., which is as follows.

Definition 5. *(Modified Tree Hash (MTH) [9], the binary case)*
Let $\mathbf{H} = (H_1, H_2, \dots,)$ *be an infinite sequence of keyed functions:* $(\Sigma^n)^2 \to \Sigma^n$. *Let* $x = (x_1, \dots, x_m) \in (\Sigma^n)^m$. *For all* $i \geq 1$, *let* L_{H_i} *be a function defined as:*

$$L_{H_i}(x) = \begin{cases} H_i(x_1, x_2) \| H_i(x_3, x_4) \| \dots \| H_i(x_{m-1}, x_m) & \text{if } m \bmod 2 = 0, \\ H_i(x_1, x_2) \| H_i(x_3, x_4) \| \dots \| H_i(x_{m-2}, x_{m-1}) \| x_m & \text{if } m \bmod 2 = 1. \end{cases}$$

The output of the modified tree hash using $\mathbf{H}$ *for input* x *is*

$$\mathrm{MTH}_{\mathbf{H}}(x) = L_{H_b} \circ L_{H_{b-1}} \circ \dots \circ L_{H_1}(x), \textit{ where } b = \lceil \log_2 m \rceil.$$

Here, $\circ$ denotes the serial composition (i.e., $F_2 \circ F_1(x) = F_2(F_1(x))$).

Collision Probability of MTH. The collision probability of MTH for equal length inputs was proved [9]. To handle inputs with unequal lengths, Boesgaard et al. suggested using a technique called the Length Annotation (LA), i.e., appending the length information of x to $\mathrm{MTH}_{\mathbf{H}}(x)$. However, we here prove that LA is not needed, if some additional conditions are satisfied.

Lemma 2. *In Def. 5, if each* H_i *is independent* ϵ*-AU and satisfies* $\Pr(H_i(x) = y) = 1/2^n$ *for any* $x \in (\Sigma^n)^2$ *and* $y \in \Sigma^n$, *then*

$$\mathrm{Coll}_{\mathrm{MTH}_{\mathbf{H}}}(m, m') \leq \max\{\lceil \log_2 m \rceil, \lceil \log_2 m' \rceil\} \cdot \epsilon, \;\; \textit{for any } (m, m'). \qquad (3)$$

Moreover, if $H_i = \mathrm{APA}_{K_i, G_{U_i}}$ *and* K_i *and* U_i *are independent and random, then Eq.* (3) *holds, where* ϵ *is the MEDP of* G_{U_i} *and* $1/2^n \leq \epsilon$.

Proof. Let us prove the first claim. We start with the case for inputs with equal lengths. Let us abbreviate $\max_{2^{c-1} < i \leq 2^c} \mathrm{Coll}_{\mathrm{MTH}_{\mathbf{H}}}(i, i)$ to $p_c^=$. Clearly $p_0^= = 0$ and $p_1^= \leq \epsilon$ hold. Assume the claim holds for $c = i - 1$ for some $i \geq 1$. Let $x = (x_1, \dots, x_m)$ and $x' = (x'_1, \dots, x'_m)$ be two m-block inputs where $2^{i-1} < m \leq 2^i$. Let S, T, and V denote $\mathrm{MTH}_{\mathbf{H}}(x_1, \dots, x_{2^{i-1}})$, $\mathrm{MTH}_{\mathbf{H}}(x_{2^{i-1}+1}, \dots, x_m)$, and $\mathrm{MTH}_{\mathbf{H}}(x)$, respectively. For x', S', T', and V' are similarly defined. If the first 2^{i-1}-block prefixes of x and x' are identical, then $P(S = S') = 1$ and we have

$$P(V = V') \leq P(T = T', S = S') + P(V = V' | T \neq T', S = S') \leq (i-1)\epsilon + \epsilon = i\epsilon,$$

where the last inequality follows from the assumption and the fact that each H_i is independent. If the first 2^{i-1}-block prefixes are different, we have

$$P(V = V') \leq P(S = S') + P(V = V'|S \neq S') \leq (i-1)\epsilon + \epsilon = i\epsilon.$$

Thus, we have $p^{=}_i \leq i\epsilon$. Let $p^{\neq}_c$ be the maximum collision probability for two inputs that have unequal lengths and their lengths are at most 2^c blocks. Then, $p^{\neq}_1 \leq \epsilon$ follows from the condition of H_i (note that $1/2^n \leq \epsilon$). Let us assume $p^{\neq}_{i-1} \leq (i-1)\epsilon$ holds. Let $x = (x_1, \ldots, x_m)$ and $x' = (x'_1, \ldots, x'_{m'})$ where $m < m'$ and $2^{i-1} < m' \leq 2^i$. If $m < 2^{i-1}$, the computation of V from (S', T') involves the key (for some H_i), $\tilde{K}$, that never appears in the computation of V. If we fix keys other than $\tilde{K}$, the collision of $\mathrm{MTH}_{\mathbf{H}}(x)$ and $\mathrm{MTH}_{\mathbf{H}}(x')$ is equivalent to the event that $H_i(s, t) = v$ for some (s, t, v, i), thus occuring with probability $1/2^n$. If $m > 2^{i-1}$, then we prove the collision probability is at most $i\epsilon$ in a similar way to the case of equal length. Therefore we have $p^{\neq}_i \leq i\epsilon$ and the first claim is proved. The second claim follows from the first claim and Lemma 1.

If LA is used, then more AU hash function calls are needed to obtain an n-bit hash value from $(\mathrm{length}(x)||\mathrm{MTH}_{\mathbf{H}}(x))$. Therefore, removing LA contributes to faster speed (particularly for short messages) and shorter key length.

4.2 Periodic CBC Hash

The MTH is ideally fast, as its theoretical throughput is almost the same as that of the AXP. However, the amounts of preprocessing and working memories required are proportional to b, where 2^b is the maximum message block length. This implies that MTH is not well suited to constrained (e.g., low-power and/or memory) environments. This problem is common to all tree-based MAC functions. In this section, we focus on building AU hash functions that accept any long block inputs with a small constant amount of preprocessing and memory. Interestingly, our proposal is an iterative procedure similar to the CBC-MAC. Since this is iterative, only a small constant working memory is needed for any input in $(\Sigma^n)^+$, as in the CBC-MAC.

For $i = 1, 2, \ldots, m$, let F_i be an n-bit block keyed function and Z be an n-bit random variable. Let $x = (x_1, \ldots, x_{m+1})$. We define two keyed functions: $(\Sigma^n)^+ \to \Sigma^n$ such that

$$\mathrm{Ch}[F_1, \ldots, F_m](x) \stackrel{\text{def}}{=} x_{m+1} \oplus F_m(x_m \oplus F_{m-1}(\ldots F_2(x_2 \oplus F_1(x_1)) \ldots)), \text{ and}$$
$$\mathrm{Ch}[F_1, \ldots, F_m|Z](x) \stackrel{\text{def}}{=} \mathrm{Ch}[F_1, \ldots, F_m](x'), \text{ where } x' = (x_1 \oplus Z, x_2, \ldots, x_{m+1}),$$

i.e., CBC-MAC-like chaining. If the input is longer than $(m + 1)$ blocks, the chaining is iterated using $(F_1, \ldots, F_m)$, and they terminate as soon as the last input block is XORed. Here, the CBC-MAC$[F]$ corresponds to $F \circ \mathrm{Ch}[F]$. For one block input $x = x_1$, the output is itself, i.e., x_1.

Definition 6. *Let E_K be an n-bit block cipher. For $d \geq 0$, let $\mathbf{G} = (G_1, \ldots, G_d)$ be the sequence of d AXPs $\mathbf{G} = (G_1, \ldots, G_d)$ (recall that G_{U_i} has been abbreviated*

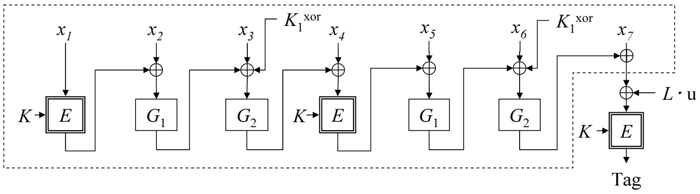

Fig. 1. PC-MAC with $d = 2$. System surrounded by dotted lines denotes $\mathrm{PCH}_2[E_K, \mathbf{G}]$.

to G_i). We call d the interval. We assume that $(d-1)$ n-bit keys, denoted by $K_1^{\mathrm{xor}}, \ldots, K_{d-1}^{\mathrm{xor}}$, are available. The Periodic CBC Hash (PCH) with interval d is a keyed function: $(\Sigma^n)^+ \to \Sigma^n$ defined as

$$\mathrm{PCH}_d[E_K, \mathbf{G}] \stackrel{\mathrm{def}}{=} \mathrm{Ch}[E_K, G_1, G_2^{\oplus K_1^{\mathrm{xor}}}, \ldots, G_d^{\oplus K_{d-1}^{\mathrm{xor}}}].$$

Here, $\mathrm{PCH}_d[E_K, \mathbf{G}]$ terminates as soon as the last input block is XORed.

See Fig. 1 for an example of PCH. If $d = 1$, then no K_i^{xor} is used and E_K and G_1 are called alternately.

Collision Probability of PCH. For any inputs in $(\Sigma^n)^+$, the collision probability of the PCH is small if the AXP has a small differential probability and a small *self-differential* probability, which is defined as follows.

Definition 7. *The maximum self-differential probability (MSDP) of a permutation on group $\mathcal{X}$, f, is defined as $\mathrm{MSDP}(f) \stackrel{\mathrm{def}}{=} \max_{a \in \mathcal{X}} \Pr[X - f(X) = a]$, where $X \in_{\mathrm{U}} \mathcal{X}$. For a keyed permutation, the maximum expected SDP (MESDP) is similarly defined.*

Let $K_{\mathrm{aux}} = (K_1^{\mathrm{xor}}, \ldots, K_{d-1}^{\mathrm{xor}}, U_1, \ldots, U_d)$, where $U_i \in \mathcal{U}$ is the key for G_i. K_{aux} is the key of $\mathbf{G}^{\oplus} \stackrel{\mathrm{def}}{=} (G_1, G_2^{\oplus K_1^{\mathrm{xor}}}, \ldots, G_d^{\oplus K_{d-1}^{\mathrm{xor}}})$ and distributed over $\mathcal{K}_{\mathrm{aux}} \stackrel{\mathrm{def}}{=} (\Sigma^n)^{d-1} \times \mathcal{U}^d$. This determines the operation between two consecutive block cipher calls in PCH. The collision probability of PCH is proved as follows.

Lemma 3. *If $K_{\mathrm{aux}} \in_{\mathrm{U}} \mathcal{K}_{\mathrm{aux}}$, and $\mathrm{MEDP}(G_i) \le \epsilon_{\mathrm{dp}}$ and $\mathrm{MESDP}(G_i) \le \epsilon_{\mathrm{sdp}}$ for $i = 1, \ldots, d$, then,*

$$\mathrm{Coll}_{\mathrm{PCH}_d[R,\mathbf{G}]}(m, m') \le d\epsilon_{\mathrm{dp}} + \epsilon_{\mathrm{sdp}} + \frac{(l + l')^2 + 2}{2^{n+1}}, \tag{4}$$

where $\mathbf{G} = (G_1, \ldots, G_d)$, and R is the n-bit URF, and $l = \lceil \frac{m}{d+1} \rceil$ and $l' = \lceil \frac{m'}{d+1} \rceil$.

Proof. Let $x = (x_1, \ldots, x_m)$ and $x' = (x'_1, \ldots, x'_{m'})$ be two distinct inputs for $\mathrm{PCH}_d[R, \mathbf{G}]$ with $m \le m'$ and let $V = \mathrm{PCH}_d[R, \mathbf{G}](x)$, $V' = \mathrm{PCH}_d[R, \mathbf{G}](x')$. Let Y_i and Z_i (Y'_i and Z'_i) be the i-th input and output of R for x (for x'). For example, when $d = 1$, then $Y_1 = x_1$, $Z_1 = R(Y_1)$, and $Y_2 = x_3 \oplus G_1(x_2 \oplus Z_1)$.

Here, if $m-1=c(d+1)$ for some positive integer c, then Y_{c+1} corresponds to $\mathrm{PCH}_d[R,\mathbf{G}](x)$ and Z_{c+1} does not exist. We also assume that the block length of the longest common prefix (LCP) between x and x' is m_{lcp}. That is, $x_i=x'_i$ for $i=1,\ldots,m_{\mathrm{lcp}}<m$ and $x_{m_{\mathrm{lcp}}+1}\neq x'_{m_{\mathrm{lcp}}+1}$ or $m_{\mathrm{lcp}}=m$ (if $m<m'$). If $x_1\neq x'_1$, the LCP is empty and has a length of 0. Let $l_{\mathrm{lcp}}=\lceil\frac{m_{\mathrm{lcp}}}{d+1}\rceil$ (this means that $Y_i=Y'_i$ for $i=1,\ldots,l_{\mathrm{lcp}}$ with probability 1).

Let D be an event where $Y_\alpha, Y'_\beta, 1\le\alpha\le l, 1\le\beta\le l'$ are distinct, except for the trivial collisions $Y_\gamma=Y'_\gamma$ for $\gamma=1,\ldots,l_{\mathrm{lcp}}$. In addition, let D_{lcp} denote an event where $Y_i\neq Y_j$ (and $Y'_i\neq Y'_j$) for $1\le i<j\le l_{\mathrm{lcp}}$. If the LCP is empty, we define $\Pr(D_{\mathrm{lcp}})=1$. Clearly D_{lcp} is a subevent of D. For any $1\le i\le j\le d+1$, we have

$$\mathrm{Coll}_{\mathrm{Ch}[\mathbf{G}^{\oplus}|\mathrm{Rnd}]}(i,j)\le\begin{cases}(i-1)\epsilon_{\mathrm{dp}}\le d\epsilon_{\mathrm{dp}} & \text{if } i=j,\\ \epsilon_{\mathrm{sdp}} & \text{if } (i,j)=(1,2),\\ \frac{1}{2^n} & \text{otherwise,}\end{cases}\tag{5}$$

where $\mathrm{Rnd}\in_{\mathrm{U}}\Sigma^n$. In Eq. (5), the first case follows from simple inductive analysis. For the second, note that the collision means $\mathrm{Rnd}\oplus x_1=x'_2\oplus G_1(\mathrm{Rnd}\oplus x'_1)$, which occurs with probability (at most) ϵ_{sdp}. The third follows from the fact that the output for the longer input always includes K_j^{xor}, which does not appear in the other output, and that all AXPs are invertible. Here, we show that the probabilities of $\overline{D}$ and $\overline{D_{\mathrm{lcp}}}$ are negligible, where $\overline{D}$ is the negation of D.

Lemma 4. *For any $k_{\mathrm{aux}}\in\mathcal{K}_{\mathrm{aux}}$, we have*

$$\Pr(\overline{D_{\mathrm{lcp}}})=\Pr(\overline{D_{\mathrm{lcp}}}|K_{\mathrm{aux}}=k_{\mathrm{aux}})\le\sum_{i=1}^{l_{\mathrm{lcp}}-1}\frac{i}{2^n}\le\frac{{l_{\mathrm{lcp}}}^2}{2^{n+1}},\ \textit{and}\tag{6}$$

$$\Pr(\overline{D})\le d\epsilon_{\mathrm{dp}}+\frac{(l+l')^2}{2^{n+1}}.\tag{7}$$

The proof of Lemma 4 is in Appendix A. Next, let us analyze the collision probability of $\mathrm{PCH}_d[R,\mathbf{G}]$. Let x_{last} be the last $m-1 \bmod (d+1)$ blocks of x. Then, $V=\mathrm{Ch}[\mathbf{G}^{\oplus}|Z_l](x_{\mathrm{last}})$ if x_{last} is not empty (i.e., $m-1 \bmod (d+1)\ge 1$), and $V=Y_l$ otherwise. x'_{last} is similarly defined for x'. First, we assume that $l=l'=l_{\mathrm{lcp}}$ does not hold true. Then, the occurrence of D means that Z_l and $Z_{l'}$ are independent and uniformly random even if K_{aux} is fixed. Thus, we have

$$\begin{aligned}\Pr(V=V')&\le\sum_{k_{\mathrm{aux}}\in\mathcal{K}_{\mathrm{aux}}}\Pr(V=V'|K_{\mathrm{aux}}=k_{\mathrm{aux}},D)\cdot\Pr(K_{\mathrm{aux}}=k_{\mathrm{aux}}|D)+\Pr(\overline{D}),\\ &\le\frac{1}{2^n}\sum_{k_{\mathrm{aux}}\in\mathcal{K}_{\mathrm{aux}}}\Pr(K_{\mathrm{aux}}=k_{\mathrm{aux}}|D)+d\epsilon_{\mathrm{dp}}+\frac{(l+l')^2}{2^{n+1}}\le d\epsilon_{\mathrm{dp}}+\frac{(l+l')^2+2}{2^{n+1}},\end{aligned}\tag{8}$$

unless both x_{last} and x'_{last} are empty. If both are empty, then $\Pr(V=V')\le P(\overline{D})$ holds. Next, let us assume $l=l'=l_{\mathrm{lcp}}$ holds true. In this case, D is equivalent to D_{lcp} and at least one of x_{last} or x'_{last} is not empty (otherwise we have $x=$

x'), and Z_l $(= Z'_{l'})$ is independent and random if $D_{\rm lcp}$ is given. If both $x_{\rm last}$ and $x'_{\rm last}$ are not empty, then $\Pr(V = V'|D_{\rm lcp})$ equals $\Pr(\mathrm{Ch}[\mathbf{G}^{\oplus}|Z_l](x_{\rm last}) = \mathrm{Ch}[\mathbf{G}^{\oplus}|Z_l](x'_{\rm last})|D_{\rm lcp})$. Here, note that $D_{\rm lcp}$ (or $\overline{D_{\rm lcp}}$) gives no information on $K_{\rm aux}$, since Eq. (6) implies $P(K_{\rm aux} = k_{\rm aux}|D_{\rm lcp}) = P(K_{\rm aux} = k_{\rm aux})$ for all $k_{\rm aux}$. From these observations and Eq. (5), and Lemma 4, we have

$$\Pr(V = V') \leq \Pr(V = V'|D_{\rm lcp}) + \Pr(\overline{D_{\rm lcp}}) \leq \max\{d\epsilon_{\rm dp}, \epsilon_{\rm sdp}, 1/2^n\} + \frac{{l_{\rm lcp}}^2}{2^{n+1}}. \quad (9)$$

It is easy to see that Eq. (9) also holds even if one of $x_{\rm last}$ or $x'_{\rm last}$ is empty. Thus, Eq. (9) holds when $l = l' = l_{\rm lcp}$. We conclude the proof by combining Eqs. (8) and (9).

Relation Between MDP (MEDP) and MSDP (MESDP). It seems that every permutation with a small MDP has a small MSDP, though we have not formally proved this for now. For instance, the inv permutation in Ex. 1 has MSDP $3/2^n$. A more useful fact is that any n-bit keyed permutation that contains an independent key-addition layer has MESDP $1/2^n$, as the output is completely random and independent of the input.

5 Complete Description of Our MACs and Their Securities

The following lemma, proved by Black and Rogaway [8], demonstrates that a VIL-PRF: $\Sigma^* \to \Sigma^n$ can be built with an AU hash: $(\Sigma^n)^+ \to \Sigma^n$ and n-bit PRFs.

Lemma 5. *(Lemma 2 of [8]) Let $H : (\Sigma^n)^+ \to \Sigma^n$ and R, R' be two independent n-bit URFs. We define $\mathrm{CW3}[H, R, R'](x) = R(H(x))$ if the length of x, $|x|$, is a multiple of n, and $R'(H(x \parallel 10^l))$ otherwise, where $|x| \bmod n = n - l - 1$ and 10^i denotes an i-bit sequence $(100\ldots0)$. Then,*

$$\mathtt{Adv}^{\mathtt{vilprf}}_{\mathrm{CW3}[H,R,R']}(q, \sigma) \leq \max_{q, m_1, \ldots, m_q, \sum_{s=1}^{q} m_s = \sigma} \left\{ \sum_{1 \leq i < j \leq q} \mathrm{Coll}_H(m_i, m_j) \right\} \quad (10)$$

holds. In Eq. (10), if σ is substituted with ρ, then the maximum is taken for all $(q, m_1, \ldots, m_q)$ such that $m_s \leq \rho$ for all $s = 1, \ldots, q$.

The Hash-to-MAC (actually Hash-to-PRF) conversion described in Lemma 5 requires two additional n-bit PRFs and thus requires two additional block cipher keyschedulings in practice. However, these keyschedulings can be removed using the idea of tweakable block ciphers [21]. This technique was used to propose the XCBC [8], TMAC [20], and OMAC [16]. In converting our hashing schemes into MACs, we also employed the tweaking technique. Here, we present complete descriptions of our MACs. The first is based on the MTH and called the MT-MAC. It uses a block cipher E_K and an AXP, G_U, and the maximum message length is $n2^b$ bits. See Fig. 2 for the details of MT-MAC. In Fig. 2, $i + a$

Preprocessing	Let L be $E_K(0)$. Let $\mathbf{U} = (U_1, \ldots, U_b)$ be the first $b\|\mathcal{U}\|$ bits of $E_K(1) \cdots E_K(a)$. Let $\mathbf{H} = (H_1, \ldots, H_b)$, where $H_i = \mathrm{APA}_{E_K(i+a), G_{U_i}}$.
Tag Computation	Input message $x \in \Sigma^*$. Let Tag $= \mathrm{CW3}[\mathrm{MTH}_{\mathbf{H}}, E_K^{\oplus L\cdot \mathtt{u}}, E_K^{\oplus L\cdot \mathtt{u}^2}](x)$. Output (x, Tag).

Fig. 2. MT-MAC$_b[E_K|G_U]$. Key of MAC is K, AXP is G_U, and $a = \lceil b|\mathcal{U}|/n \rceil$.

Preprocessing	Let $\mathbf{U} = (U_1, \ldots, U_d)$ be the first $d\|\mathcal{U}\|$ bits of $E_K^{\oplus L}(0) \cdots E_K^{\oplus L}(\hat{a}-1)$, and let $\mathbf{G} = (G_1, \ldots, G_d)$. Let $K^{\mathrm{xor}}_{j-\hat{a}+1}$ be $E_K^{\oplus L}(j)$ for $j = \hat{a}, \ldots, \hat{a}+d-2$.
Tag Computation	Input message $x \in \Sigma^*$. Let Tag $= \mathrm{CW3}[\mathrm{PCH}_d[E_K, \mathbf{G}], E_K^{\oplus L\cdot \mathtt{u}}, E_K^{\oplus L\cdot \mathtt{u}^2}](x)$. Output (x, Tag).

Fig. 3. PC-MAC$_d[E_K, L|G_U]$. Key of MAC is (K, L), AXP is G_U, and $\hat{a} = \lceil d|\mathcal{U}|/n \rceil$.

indicates usual integer addition, and $\mathtt{u}$ is an element of $\mathrm{GF}(2^n)$ that is not 0 or 1 and $L \cdot \mathtt{u}$ denotes the multiplication on $\mathrm{GF}(2^n)$. It can be implemented with shift and conditional XOR (e.g., see [16]). The second is based on PCH and called the PC-MAC. The PC-MAC is shown in Fig. 3.

The security of MT-MAC is proved as follows. The proof is in Appendix B.

Theorem 1. *Let $c = \lceil b|\mathcal{U}|/n \rceil + b + 1$. Then,*

$$\mathtt{Adv}^{\mathtt{vilprf}}_{\text{MT-MAC}_b[E_K|G_U]}(q, t, \sigma) \le \mathtt{Adv}^{\mathtt{prp}}_{E_K}(\sigma + c, t') + \frac{(\sigma + c)^2}{2^n} + \epsilon_{\mathrm{dp}} \sigma^2,$$

where $t' = t + O(\sigma)$ and $\epsilon_{\mathrm{dp}} = \mathrm{MEDP}(G_U)$.

The security proof of PC-MAC can be similarly obtained, which is as follows.

Theorem 2. *Let $c = \lceil d|\mathcal{U}|/n \rceil + d$, where d is the interval parameter. Then,*

$$\mathtt{Adv}^{\mathtt{vilprf}}_{\text{PC-MAC}_d[E_K, L|G_U]}(q, t, \rho) \le \mathtt{Adv}^{\mathtt{prp}}_{E_K}(\rho q + c, t') + \frac{2.5(\rho q + c)^2}{2^n} + (d\epsilon_{\mathrm{dp}} + \epsilon_{\mathrm{sdp}}) \frac{q^2}{2},$$

where $t' = t + O(\rho q)$, and $\epsilon_{\mathrm{dp}} = \mathrm{MEDP}(G_U)$, and $\epsilon_{\mathrm{sdp}} = \mathrm{MESDP}(G_U)$.

The proof of Theorem 2 is in Appendix C.

Security Parameter. In Theorem 2, we used ρ instead of σ, although using σ is generally more preferable than using ρ (see discussion in [15]). If we are forced to use σ, the bound of a PC-MAC's CPA-advantage would be $O(\sigma^2 q^2/2^n)$, which seems a bit too loose. It would be nice if we could obtain a smaller bound for the collision probability to obtain a tight security analysis using σ.

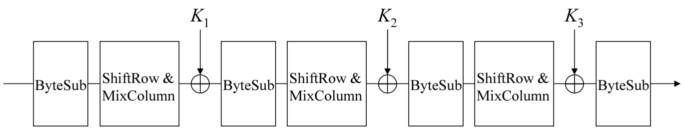

Fig. 4. The simplified 4-round AES. Each K_i is independent and random.

Key Length. The PC-MAC uses two keys, the first for the block cipher and the second to make the block cipher tweakable. It is natural to ask whether this can be reduced to one block cipher key without introducing another block cipher keyscheduling. For example, is it secure to let $L = E_K(0)$, just like in the OMAC or MT-MAC? Unfortunately, we do not have a clear answer for now, but at least we found a counterexample if some generalization was applied to the PC-MAC[1]. Nevertheless, we think that a small change can provide a one-key version of the PC-MAC. This is still a problem that needs to be solved.

6 AES-based Implementation

In this section, we consider the implementation of our MACs using AES. As mentioned earlier, the 4-round AES with independent round keys has MEDP 2^{-113} [18]. Also, MESDP is exactly 2^{-128}, since the 4-round AES contains an independent key-addition layer (see end of Sect. 4.2). Here, the addition of the first round key and the last diffusion layer can be omitted, since they do not affect the differential and self-differential probabilities. Let us denote the simplified 4-round AES in Fig. 4 by 4rAES. If our MACs are implemented with AES and 4rAES, then the securities of the resulting MT-MAC and PC-MAC can be proved by Theorems 1 and 2 with $n = 128$, $|\mathcal{U}| = 384$, $\epsilon_{\text{dp}} = 2^{-113}$, and $\epsilon_{\text{sdp}} = 2^{-128}$.

Some Comparisons. Compared with the previous MAC modes of AES, for example OMAC-AES, our AES-based MACs are faster (MT-MAC is about 2.5 times faster, and PC-MAC is about 1.4 to 2.5 times faster, depending on the interval). Both use the AES encryption, and do not require the AES decryption. Their program sizes are almost the same. Both provide stateless (i.e., no counter or nonce is used) provably secure VIL-MACs. The computational assumptions we need are the same (i.e., the pseudorandomness of the AES). Drawbacks of our MACs are the amount of preprocessing and slightly-degraded security: many CBC-MAC variants have 64-bit security, i.e., they are secure if q (or σ, ρ) $\ll 2^{64}$,

[1] For example, even if $L \cdot \mathtt{u}$ is substituted with $\mathrm{inv}(L \oplus \mathtt{u})$ (note that inv is defined in Ex. 1), OMAC will still be secure, as it satisfies the condition for the "OMAC-family" (see [16] for details). However, if $L = E_K(0)$, and the AXP is the inv permutation, and the above substitution is applied to the PC-MAC with interval 1, then the tag for the 3-block input $(0, \mathtt{u}, x_2)$ is $E_K(x_2)$ for any x_2, i.e., direct access to E_K is possible. This means the complete break of the MAC.

Table 1. Summary of AES-based MACs. "Rounds" denotes average AES rounds to process one message block, and "Preproc." denotes AES encryption blocks needed in preprocessing.

MAC	Max.Message Length	Rounds	Preproc.	Key size	Type
MT-MAC$_b$[AES|4rAES]	$n2^b$	4	$4b+1$	128	Tree
PC-MAC$_d$[AES, L|4rAES]	Infinite	$4+\frac{6}{d+1}$	$4d-1$	256	Iterative
OMAC-AES	Infinite	10	1	128	Iterative

while ours have about 56-bit security. We have summarized the properties of our AES-based MACs below. For comparison, the OMAC-AES is also shown. Table 1 shows only average speed estimates for long messages. However, our MACs are at least as fast as OMAC-AES for any short messages, since the AES rounds needed by our MACs are no more than $10 \cdot m$, when the input is m-block and this holds for all $m \geq 1$.

It may be rather difficult to perform a rigorous comparison between our MACs and the state-of-art CW-MACs, such as UMAC [19], (the MAC part of) GCM [24], and Poly1305 [7], as they use customized functions that can not be derived from AES. For example, CW-MACs are roughly 3 to 5 (or more) times faster than the MAC modes using the optimized AES on software (e.g., see [3]). Therefore our MACs may not be as fast as them on software. Also, some CW-MACs have much shorter keyscheduling time than ours. However, ours can be easily implemented for any platform where an implementation of AES is available. There are many studies on efficient AES implementations for various software and hardware (e.g., see [1]), and we can directly benefit from them. For other comparison items, both provide provably secure VIL-MACs (some CW-MACs are stateful) based on the pseudorandomness of the AES.

The Pelican MAC [12] is an instantiation of the ALRED using AES and its 4-round with all zero round keys. It is similar (but not identical) to the PC-MAC. This is not surprising because the ALRED and the PC-MAC share the same motivation. The Pelican MAC is about 2.5 times faster than the CBC-MAC, thus almost the same speed as that of the MT-MAC or PC-MAC with a long interval. Compared to our MACs, the Pelican MAC's preprocessing time is very short (only one block encryption). From the preliminary analysis of the ALRED construction [11], the Pelican MAC's security was proved against attacks that did not invoke internal collisions. In addition, no attack better than the brute force search has not been found for the moment. However, it is still unclear whether the Pelican MAC is a provably secure (i.e., secure against all attacks) VIL-MAC based on the pseudorandomness of the AES.

These comparisons are rough and might be insufficient. As a future work item, we want to do a more comprehensive and quantitative comparison to clarify the effectiveness of our approach.

Implementations. We also implemented our AES-based MACs on software. We used the public-domain C code written by Rijmen et al.[2]. Our implementation

Table 2. Comparison of AES-based MAC speed on software.

MAC	Tag computation (cycle/byte)	Preprocessing (cycles)
MT-MAC ($b = 32$)	12.5	53777 (estimate)
PC-MAC ($d = 1$)	18.5	1651
PC-MAC ($d = 5$)	14.4	8311
PC-MAC ($d = 17$)	13.1	28444
OMAC	25.1	821

was naive and almost no optimization was performed. We did a speed comparison on a Pentium III 1 Ghz, where raw AES encryption ran at about 25 cycle/byte. We can see from Table 2 that our MACs did not achieve the theoretical limit (i.e., 2.5 times faster than OMAC-AES). This is because some overhead was involved in both AES and 4rAES, such as byte/word conversion. The effect overhead has may change according to the platform and AES implementation.

Acknowledgments

We would like to thank Etsuko Tsujihara for implementing MACs and anonymous reviewers for useful and detailed comments.

References

1. `http://www.iaik.tu-graz.ac.at/research/krypto/AES/index.php`.
2. `http://homes.esat.kuleuven.be/~rijmen/rijndael/rijndael-fst-3.0.zip`.
3. `http://cr.yp.to/streamciphers.html`.
4. B. den Boer, J.P. Boly, A. Bosselaers, J. Brandt, D. Chaum, I. Damgård, M. Dichtl, W. Fumy, M. van der Ham, C.J.A. Jansen, P. Landrock, B. Preneel, G. Roelofsen, P. de Rooij, and J. Vandewalle, *RIPE Integrity Primitives*, final report of RACE Integrity Primitives Evaluation. 1995.
5. M. Bellare, J. Kilian, and P. Rogaway. "The Security of the Cipher Block Chaining Message Authentication Code." *Journal of Computer and System Science*, Vol. 61, No. 3, 2000.
6. M. Bellare, A. Desai, E. Jokipii, and P. Rogaway. " A Concrete Security Treatment of Symmetric Encryption." *Proceedings of the 38th Annual Symposium on Foundations of Computer Science*, FOCS '97, pp. 394-403, 1997.
7. D. J. Bernstein. "The Poly1305-AES Message-Authentication Code." *Fast Software Encryption*, FSE'05, LNCS 3557, pp. 32-49, 2005.
8. J. Black and P. Rogaway. "CBC MACs for Arbitrary-Length Messages: The Three-Key Constructions." *Advances in Cryptology*- CRYPTO '00, LNCS 1880, pp. 197-215, 2000.
9. M. Boesgaard, T. Christensen, and E. Zenner. "Badger - A Fast and Provably Secure MAC." *Applied Cryptography and Network Security*- ACNS'05, LNCS 3531, pp. 176-191, 2005.

10. L. Carter and M. Wegman. "Universal Classes of Hash Functions." *Journal of Computer and System Science*, Vol. 18, pp. 143-154, 1979.
11. J Daemen and V. Rijmen. "A New MAC Construction ALRED and a Specific Instance ALPHA-MAC." *Fast Software Encryption*, FSE'05, LNCS 3557, pp. 1-17, 2005.
12. J Daemen and V. Rijmen. "The Pelican MAC Function." *IACR ePrint Archive*, 2005/088.
13. O. Goldreich. "Modern Cryptography, Probabilistic Proofs and Pseudorandomness." Springer-Verlag, Algorithms and Combinatorics, Vol. 17, 1998.
14. S. Halevi and H. Krawczyk. "MMH:Software Message Authentication in the Gbit/second rates." *Fast Software Encryption*, FSE'97, LNCS 1267, pp. 172-189, 1997.
15. T. Iwata and K. Kurosawa. "Stronger Security Bounds for OMAC, TMAC, and XCBC." *Progress in Cryptology-* INDOCRYPT'03, LNCS 2904, pp. 402-415, 2003.
16. T. Iwata and K. Kurosawa. "OMAC: One-Key CBC MAC." *Fast Software Encryption-* FSE'03, LNCS 2887, pp. 129-153, 2003.
17. T. Iwata and K. Kurosawa. "On the Universal Hash Functions in Luby-Rackoff Cipher." *IEICE Transactions*, Volume 87–A, pp. 60-66, 2004.
18. L. Keliher and J. Sui. "Exact Maximum Expected Differential and Linear Probability for 2-Round Advanced Encryption Standard (AES)." *IACR ePrint Archive*, 2005/321.
19. T. Krovetz. "Software-Optimized Universal Hashing and Message Authentication". *PhD dissertation*, available from `http://www.cs.ucdavis.edu/~rogaway/umac`.
20. K. Kurosawa and T. Iwata. "TMAC: Two-Key CBC MAC." *Topics in Cryptology-* CT-RSA 2003, LNCS 2612, pp. 33-49, 2003.
21. M. Liskov, R. L. Rivest, and D. Wagner. "Tweakable Block Ciphers." *Advances in Cryptology-* CRYPTO'02, LNSC 2442, pp. 31-46, 2002.
22. M. Luby and C. Rackoff. "How to Construct Pseudo-random Permutations from Pseudo-random functions." *SIAM J. Computing*, Vol. 17, No. 2, pp. 373-386, 1988.
23. U. Maurer. "Indistinguishability of Random Systems." *Advances in Cryptology-* EUROCRYPT'02, LNCS 2332, pp. 110-132, 2002.
24. D. McGrew and J. Viega. "The Galois/Counter Mode of Operation (GCM)." *Submission to NIST Modes of Operation Process*, 2004.
25. K. Nyberg. "Differentially Uniform Mappings for Cryptography." *Advances in Cryptology-* EUROCRYPT'93, LNCS 765, pp. 55-64, 1994.
26. S. Park, S. H. Sung, S. Lee, and J. Lim. "Improving the Upper Bound on the Maximum Differential and the Maximum Linear Hull Probability for SPN Structures and AES." *Fast Software Encryption*, FSE'03, LNCS 2887, pp. 247-260, 2003.
27. M. Wegman and L. Carter. "New Hash Functions and Their Use in Authentication and Set Equality." *Journal of Computer and System Sciences*, Vol. 22, pp. 265-279, 1981.

A Proof of Lemma 4

We assumed that LCP was not empty and $l_{\mathrm{lcp}} < l$ (i.e., $Y_{l_{\mathrm{lcp}}+1}$ and $Y'_{l_{\mathrm{lcp}}+1}$ exist). If we fix K_{aux} to k_{aux}, then the operation that accepts Z_i and outputs Y_{i+1} (i.e., $\mathrm{Ch}[\mathbf{G}^{\oplus}]$) is a deterministic n-bit permutation defined by k_{aux} and some $d+1$ input blocks. It is not hard to see that $\Pr(\overline{D_{\mathrm{lcp}}}|K_{\mathrm{aux}} = k_{\mathrm{aux}})$ is exactly the same as the output collision probability of the OFB mode of R, for all $k_{\mathrm{aux}} \in \mathcal{K}_{\mathrm{aux}}$

and inputs. Then, Eq. (6) follows from this observation and a simple collision analysis. Next, let us prove Eq. (7). Consider the following collision events. (I): $Y_{l_{\rm lcp}+1} = Y'_{l_{\rm lcp}+1}$ and (II):$Y_{l_{\rm lcp}+1} = Y_i, Y'_{l_{\rm lcp}+1} = Y_i$ for $i = 1, \ldots, l_{\rm lcp}$. Here, $Y_{l_{\rm lcp}+1}$ and $Y'_{l_{\rm lcp}+1}$ are two outputs of $\mathrm{Ch}[\mathbf{G}^{\oplus}|Z_{l_{\rm lcp}}]$ with different inputs. Since $Z_{l_{\rm lcp}}$ is independent and uniformly random if $D_{\rm lcp}$ is given, we can use Eq. (5) and obtain $\Pr(Y_{l_{\rm lcp}+1} = Y'_{l_{\rm lcp}+1}|D_{\rm lcp}) \le d\epsilon_{\rm dp}$. Moreover, $\Pr(Y_{l_{\rm lcp}+1} = Y_i|D_{\rm lcp})$ (or $\Pr(Y'_{l_{\rm lcp}+1} = Y_i|D_{\rm lcp})$) is $1/2^n$ for $i = 1, \ldots, l_{\rm lcp}$, since $Y_{l_{\rm lcp}+1}$ and $Y'_{l_{\rm lcp}+1}$ are permutations of $Z_{l_{\rm lcp}}$. If no collision events of types (I) and (II) have occurred, $Z_{l_{\rm lcp}+1}$ and $Z'_{l_{\rm lcp}+1}$ are independent and completely random, no matter what $K_{\rm aux}$ is. This implies that other collision events consisting of $\overline{D}|D_{\rm lcp}$ occur with probability $1/2^n$. Consequently, all collision events consisting of $\overline{D}|D_{\rm lcp}$ occur with probability $1/2^n$ except for the event $Y_{l_{\rm lcp}+1} = Y'_{l_{\rm lcp}+1}$. By counting these events and using Eq. (6), we have

$$\begin{aligned}\Pr(\overline{D}) &\le \Pr(\overline{D_{\rm lcp}}) + \Pr(\overline{D}|D_{\rm lcp}),\\ &\le \frac{{l_{\rm lcp}}^2}{2^{n+1}} + d\epsilon_{\rm dp} + \frac{1}{2^n}\left(\binom{l+l'-l_{\rm lcp}}{2} - \binom{l_{\rm lcp}}{2} - 1\right) \le d\epsilon_{\rm dp} + \frac{(l+l')^2}{2^{n+1}}.\end{aligned}$$

For other cases (e.g., the LCP is empty or $l_{\rm lcp} = l$), the above bound also holds true. This proves Eq. (7).

B Proof of Theorem 1

Let Q be $\mathrm{CW3}[\mathrm{MTH}_{\widetilde{\mathbf{H}}}, R, R']$, where $\widetilde{\mathbf{H}} = (\widetilde{H}_1, \widetilde{H}_2, \ldots, \widetilde{H}_b)$ consists of $\widetilde{H}_i = \mathrm{APA}_{\widetilde{K}_i, G_{\widetilde{U}_i}}$ and $\{\widetilde{U}_i, \widetilde{K}_i\}_{i=1,\ldots,b}$ are independent of each other, and R, R' are independent n-bit URFs. From Lemmas 2 and 5, we have

$$\mathtt{Adv}^{\mathtt{vilprf}}_{Q}(q, \sigma) \le \epsilon_{\rm dp} \cdot q\sigma \le \epsilon_{\rm dp}\sigma^2. \tag{11}$$

Then, we use the following lemma.

Lemma 6. *Let R be an n-bit URF. Let $\mathrm{TE}_1 : \Sigma^n \times \Sigma \to \Sigma^n$, where $\mathrm{TE}_1(x, 0) = R^{\oplus L\cdot \mathtt{u}}(x)$ and $\mathrm{TE}_1(x, 1) = R^{\oplus L\cdot \mathtt{u}^2}(x)$ for $L = R(0)$. Consider the following two games, Gm1 and Gm2. In Gm1, one can access TE_1 and $R(c_1), R(c_2), \ldots, R(c_a)$ where $c_1, c_2, \ldots, c_a$ are distinct and fixed constants and $c_i \ne 0$ for all i. In Gm2, one can access the URF: $\Sigma^n \times \Sigma \to \Sigma^n$ and a (an)-bit independent and random sequence. Then, $\mathtt{Adv}^{\mathtt{cpa}}_{\mathrm{Gm1},\mathrm{Gm2}}(q)$[2] is at most $\frac{q^2}{2^{n+1}} + \frac{(a+1)q}{2^n}$.*

Proof. (Sketch) Let S_i be the i-th input to R in Gm1, i.e., S_i equals $x_i \oplus L\cdot\mathtt{u}$ if the i-th query is $(x_i, 0)$ and $x_i \oplus L \cdot \mathtt{u}^2$ if the i-th query is $(x_i, 1)$. Let a_i be the event that $S_1, S_2, \ldots, S_i$ are distinct and $S_j \notin \{0, c_1, \ldots, c_a\}$ for $j = 1, \ldots, i$. Then, using the methodology of Maurer [23], $\mathtt{Adv}^{\mathtt{cpa}}_{\mathrm{Gm1},\mathrm{Gm2}}(q)$ is at most the probability

[2] This should be interpreted as the maximum CPA-advantage in distinguishing two games using q queries with no computational restriction, where a query is in $\Sigma^n \times \Sigma$.

of $\overline{a_q}$ for all (both adaptive and non-adaptive) adversaries using q queries when Gm1 is considered. All collision events consisting of $\overline{a_q}$ have probability $1/2^n$ or 0. By counting the number of collision events and using the union bound, we conclude the proof.

From Lemma 6, we have $\mathtt{Adv}^{\mathtt{cpa}}_{Q,\text{MT-MAC}_b[R|G_U]}(q,\sigma) \leq \frac{\sigma^2}{2^{n+1}} + \frac{(c+1)\sigma}{2^n}$. From this observation and Eq. (11), we have

$$\mathtt{Adv}^{\mathtt{vilprf}}_{\text{MT-MAC}_b[R|G_U]}(q,\sigma) \leq \mathtt{Adv}^{\mathtt{cpa}}_{Q,\text{MT-MAC}_b[R|G_U]}(q,\sigma) + \epsilon_{\text{dp}}\sigma^2 + \frac{\sigma^2}{2^{n+1}} + \frac{(c+1)\sigma}{2^n}. \tag{12}$$

Distinguishing $\text{MT-MAC}_b[R|G_U]$ from $\text{MT-MAC}_b[E_K|G_U]$ with (q,t,σ) implies distinguishing R from E_K with $\sigma+c$ queries and $t' = t + O(\sigma)$ time. Combining this observation and Eq. (12) and the standard PRF/PRP switching lemma (e.g., see Lemma 1 of [8]) proves the theorem.

C Proof of Theorem 2

Let Q be $\text{CW3}[\text{PCH}_d[R,\mathbf{G}],R',R'']$ where three n-bit URFs R, R', and R'' are independent and the auxiliary key K_{aux} is generated by the counter mode of another URF, R''', i.e., $K_{\text{aux}} \in_{\text{U}} \mathcal{K}_{\text{aux}}$. Combining Lemmas 5 and 3, we have

$$\begin{aligned}\mathtt{Adv}^{\mathtt{vilprf}}_{Q}(q,\rho) &\leq \max_{q,m_1,\ldots,m_q,m_s\leq\rho} \sum_{1\leq i<j\leq q} d\epsilon_{\text{dp}} + \epsilon_{\text{sdp}} + \frac{(\lceil\frac{m_i}{d+1}\rceil + \lceil\frac{m_j}{d+1}\rceil)^2+2}{2^{n+1}}\\ &\leq \left(d\epsilon_{\text{dp}} + \epsilon_{\text{sdp}} + \frac{2\rho^2+1}{2^n}\right)\frac{q^2}{2}. \end{aligned}\tag{13}$$

Then, the following lemma is used. It is similar to Lemma 4.1 of [15].

Lemma 7. *Let R be the n-bit URF. Consider the following two games,* Gm1 *and* Gm2*. In* Gm1*, one can access $\text{TE}_2 : \Sigma^n \times \{0,1,2,3\} \to \Sigma^n$ where $\text{TE}_2(x,0) = R(x)$, and $\text{TE}_2(x,i) = R^{\oplus L\cdot \mathtt{u}^{(i-1)}}(x)$ for $i = 1,2,3$, and $L \in_{\text{U}} \Sigma^n$. In* Gm2*, one can access the URF compatible with TE_2. Then, $\mathtt{Adv}^{\mathtt{cpa}}_{\text{Gm1},\text{Gm2}}(q)$ is at most $\frac{q^2}{2^{n+1}}$.*

As the proof of Lemma 7 is a simple extension of the proof of Lemma 6, we have omitted it here. Note that $\text{PC-MAC}_d[R,L|G_U]$ invokes R at most $\rho q + c$ times. From this observation and Lemma 7, it is clear that $\mathtt{Adv}^{\mathtt{cpa}}_{\text{PC-MAC}_d[R,L|G_U],Q}(q,\rho)$ is at most $\frac{(c+\rho q)^2}{2^{n+1}}$. From this and Eq. (13), we have

$$\begin{aligned}&\mathtt{Adv}^{\mathtt{vilprf}}_{\text{PC-MAC}_d[R,L|G_U]}(q,\rho) \leq \frac{(c+\rho q)^2}{2^{n+1}} + \left(d\epsilon_{\text{dp}} + \epsilon_{\text{sdp}} + \frac{2\rho^2+1}{2^n}\right)\frac{q^2}{2}, \quad \text{and}\\ &\mathtt{Adv}^{\mathtt{vilprf}}_{\text{PC-MAC}_d[E_K,L|G_U]}(q,t,\rho) \leq \mathtt{Adv}^{\mathtt{prf}}_{E_K}(\rho q + c, t') + \mathtt{Adv}^{\mathtt{vilprf}}_{\text{PC-MAC}_d[R,L|G_U]}(q,\rho)\end{aligned}\tag{14}$$

where $t' = t + O(\rho q)$. Combining Eq. (14) with the PRF/PRP switching lemma concludes the proof.