Correction to "On the Security of Two New OMAC Variants"

Tetsu Iwata and Kaoru Kurosawa

Department of Computer and Information Sciences,
Ibaraki University
4–12–1 Nakanarusawa, Hitachi, Ibaraki 316-8511, Japan
{iwata, kurosawa}@cis.ibaraki.ac.jp

September 19, 2003

Correction

In "On The Security of Two New OMAC Variants" of September 1, 2003, we wrote that Mitchell proposed two new variants of OMAC, OMAC1$'$ and OMAC1$''$. In this paper, we correct this sentence as follows.

Mitchell proposed a new variant of OMAC (not two variants). [a]

[a] Mitchell recently informed us that he proposes only OMAC1$''$ in his paper. We now agree this after careful reading of his paper. Earlier, we thought that the first paragraph of [12, Sect. 5.4] is OMAC1$'$ and the second paragraph is OMAC1$''$.

Abstract. OMAC is a provably secure MAC scheme which NIST currently intends to specify as the modes recommendation. In August 2003, Mitchell proposed a variant of OMAC. We call it OMAC1$''$. In this paper, we prove that OMAC1$''$ is *less secure* than original OMAC1. We show a security gap between them. As a result, we obtain a negative answer to Mitchell's open question — OMAC1$''$ is *not* provably secure even if the underlying block cipher is a PRP.

Keywords: Message authentication code, OMAC, provable security, pseudorandom permutation.

1 Introduction

1.1 Background

CBC MAC [6, 7] is a well-known and widely used message authentication code (MAC) based on a block cipher E. We denote the CBC MAC value of a message M by $\mathrm{CBC}_K(M)$, where K is the key of E. While Bellare, Kilian, and Rogaway proved that the CBC MAC is secure for fixed length messages [2], it is *not* secure for variable length messages.

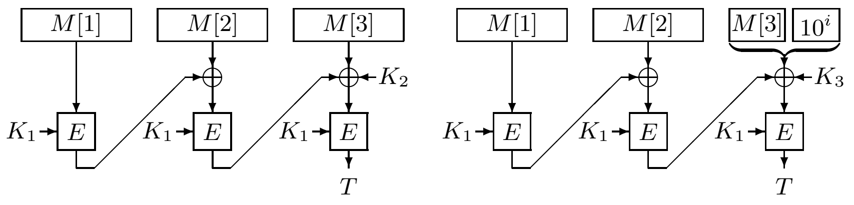

Fig. 1. Illustration of XCBC.

Therefore, several variants of CBC MAC have been proposed which are provably secure for variable length messages: we have EMAC, XCBC, TMAC and OMAC.

EMAC (Encrypted MAC) is obtained by encrypting $\mathrm{CBC}_{K_1}(M)$ by E again with a new key K_2 [3]. That is,

$$\mathrm{EMAC}_{K_1,K_2}(M) = E_{K_2}(\mathrm{CBC}_{K_1}(M)) \ .$$

Petrank and Rackoff proved that EMAC is secure if the message length is a multiple of n, where n is the block length of E [13].

For arbitrary length messages, we can simply append the minimal 10^i to a message M so that the length is a multiple of n. In this method, however, we must append an entire extra block 10^{n-1} if the size of the message is already a multiple of n. This is a "wasting" of one block cipher invocation.

Black and Rogaway next proposed XCBC to solve the above problem [4]. XCBC takes *three* keys: one k-bit key K_1 for E, two n-bit keys K_2 and K_3 (k denotes the key length of E). In XCBC, we do not append 10^{n-1} if the size of the message is already a multiple of n. Only if this is not the case, we append the minimal 10^i. In order to distinguish them, K_2 or K_3 is XORed before encrypting the last block. XCBC is now described as follows (see Fig. 1).

– If $|M| = mn$ for some $m > 0$, then XCBC computes exactly the same as the CBC MAC, except for XORing an n-bit key K_2 before encrypting the last block.
– Otherwise, 10^i padding ($i = n-|M|-1 \bmod n$) is appended to M and XCBC computes exactly the same as the CBC MAC for the padded message, except for XORing another n-bit key K_3 before encrypting the last block.

Kurosawa and Iwata then proposed TMAC which requires *two* keys, one k-bit key K_1 and one n-bit key K_2 [10]. TMAC is obtained from XCBC by replacing (K_2, K_3) with $(K_2 \cdot \mathtt{u}, K_2)$, where $\mathtt{u}$ is some non-zero constant and "$\cdot$" denotes multiplication in $\mathrm{GF}(2^n)$. Sung, Hong, and Lee showed a key recovery attack against TMAC [15].

Finally, Iwata and Kurosawa proposed OMAC which requires only *one* block cipher key K [8]. OMAC is a generic name for OMAC1 and OMAC2. Let $L = E_K(0^n)$. Then OMAC1 is obtained from XCBC by replacing (K_1, K_2, K_3) with $(K, L \cdot \mathtt{u}, L \cdot \mathtt{u}^2)$. Similarly, OMAC2 is obtained from XCBC by replacing (K_1, K_2, K_3) with $(K, L \cdot \mathtt{u}, L \cdot \mathtt{u}^{-1})$.

1.2 A New Variant of OMAC1: OMAC1$''$ [12]

EMAC, XCBC, TMAC and OMAC are all provably secure against chosen message attack if the underlying block cipher is a PseudoRandom Permutation (PRP). Indeed, for all of the above MACs, it has been shown that the forging probability is upper bounded by the birthday bound term plus insecurity function of the underlying block cipher as a PRP, which is a standard and acceptable security bound. In fact, many block cipher modes of operations have this security bound. For example we have CTR mode [1] and CBC mode [1] for symmetric encryption, and PMAC [5] for message authentication. Nevertheless, Mitchell proposed a new variant of OMAC1 to *improve* the security of original OMAC1. We call it OMAC1$''$. OMAC1$''$ is obtained from XCBC by replacing (K_1, K_2, K_3) with $(K \oplus S_1, E_K(S_2), E_K(S_3))$, where S_1 is some fixed k-bit constant, S_2 and S_3 are some distinct n-bit constants.

It was claimed that OMAC1$''$ is *more secure* than OMAC1 [12]. Mitchell also posed an open question of whether OMAC1$''$ is provably secure [12].

1.3 Our Contribution

In this paper, however, we show that the security is not improved. We prove that OMAC1$''$ is *less secure* than original OMAC1. We show a security gap between them.

To derive this result we first consider another variant of OMAC1, called OMAC1$'$. We then show that OMAC1$'$ is completely insecure. There are forgery attacks by using only one oracle query.

We next construct a PRP G with the following property: For any $K \in \{0,1\}^k$,

$$G_K(\cdot) = G_{K \oplus S_1}(\cdot) \ .$$

(A similar PRP is used in [14, 9].) We then show that OMAC1$''$ is completely insecure if G is used as the underlying block cipher. In particular, we show that forgery attacks against OMAC1$'$ can also be applied to OMAC1$''$ if G is used as the underlying block cipher. This implies underlying block cipher being a PRP is *not enough* for proving the security of OMAC1$''$. Equivalently, it is *impossible* for OMAC1$''$ to prove its security under the assumption of the underlying block cipher being a PRP. That is,

- OMAC1 is a secure MAC if the underlying block cipher is a PRP [8], while
- it is *impossible* for OMAC1$''$ to achieve this security notion.

Therefore, there is a security gap between OMAC1 and OMAC1$''$, and OMAC1$''$ is *less secure* than OMAC1. This gives a negative answer to Mitchell's open question — OMAC1$''$ is not provably secure even if the underlying block cipher is a PRP.

2 Preliminaries

2.1 Block Ciphers and MACs

Block cipher, E. A block cipher E is a function $E : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$, where, for each $K \in \{0,1\}^k$, $E(K,\cdot)$ is a permutation over $\{0,1\}^n$. We write $E_K(\cdot)$ for $E(K,\cdot)$. k is called the key length and n is called the block length. For TripleDES, $k = 112, 168$ and $n = 64$, and for the AES, $k = 128, 192, 256$ and $n = 128$.

MAC. A MAC is a function $\mathrm{MAC} : \{0,1\}^k \times \{0,1\}^* \to \{0,1\}^n$. It takes a key $K \in \{0,1\}^k$ and a message $M \in \{0,1\}^*$ to return an n-bit tag $T \in \{0,1\}^n$. We write $\mathrm{MAC}_K(\cdot)$ for $\mathrm{MAC}(K,\cdot)$. In this paper, we only consider deterministic MACs.

2.2 Security Definitions

Our definitions follow from those given in [11] for PRP, and [2] for the security of MACs.

Security of block ciphers (PRP) [11]. Let $\mathrm{Perm}(n)$ denote the set of all permutations on $\{0,1\}^n$. We say that P is a random permutation if P is randomly chosen from $\mathrm{Perm}(n)$.

The security of a block cipher $E : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ as a pseudorandom permutation (PRP) is quantified as $\mathtt{Adv}^{\mathsf{prp}}_E(\mathcal{A})$, the advantage of an adversary $\mathcal{A}$ that tries to distinguish $E_K(\cdot)$ (with a randomly chosen key K) from a random permutation $P(\cdot)$. Let $\mathcal{A}^{E_K(\cdot)}$ denote $\mathcal{A}$ with an oracle which, in response to a query X, returns $E_K(X)$, and let $\mathcal{A}^{P(\cdot)}$ denote $\mathcal{A}$ with an oracle which, in response to a query X, returns $P(X)$. After making queries, $\mathcal{A}$ outputs a bit. Then the advantage is defined as

$$\mathtt{Adv}^{\mathsf{prp}}_E(\mathcal{A}) \stackrel{\mathrm{def}}{=} \left|\Pr(K \stackrel{R}{\leftarrow} \{0,1\}^k : \mathcal{A}^{E_K(\cdot)} = 1) - \Pr(P \stackrel{R}{\leftarrow} \mathrm{Perm}(n) : \mathcal{A}^{P(\cdot)} = 1)\right| .$$

We say that E is a PRP if $\mathtt{Adv}^{\mathsf{prp}}_E(\mathcal{A})$ is sufficiently small for any $\mathcal{A}$.

Security of MACs [2]. Let $\mathrm{MAC} : \{0,1\}^k \times \{0,1\}^* \to \{0,1\}^n$ be a MAC algorithm. Let $\mathcal{A}^{\mathrm{MAC}_K(\cdot)}$ denote $\mathcal{A}$ with an oracle which, in response to a query $M \in \{0,1\}^*$, returns $\mathrm{MAC}_K(M) \in \{0,1\}^n$. We say that an adversary $\mathcal{A}^{\mathrm{MAC}_K(\cdot)}$ *forges* if $\mathcal{A}$ outputs (M,T), where $T = \mathrm{MAC}_K(M)$ and $\mathcal{A}$ never queried M to its oracle $\mathrm{MAC}_K(\cdot)$. We call (M,T) a forgery attempt. Then we define the advantage as

$$\mathtt{Adv}^{\mathsf{mac}}_{\mathrm{MAC}}(\mathcal{A}) \stackrel{\mathrm{def}}{=} \Pr(K \stackrel{R}{\leftarrow} \{0,1\}^k : \mathcal{A}^{\mathrm{MAC}_K(\cdot)} \text{ forges}) .$$

We say that a MAC algorithm is secure if $\mathtt{Adv}^{\mathsf{mac}}_{\mathrm{MAC}}(\mathcal{A})$ is sufficiently small for any $\mathcal{A}$.

```
Algorithm OMAC1_K(M)
$L \leftarrow E_K(0^n)$
$Y[0] \leftarrow 0^n$
Let $M = M[1] \cdots M[m]$, where $|M[i]| = n$ for $i = 1, \ldots, m-1$
for $i \leftarrow 1$ to $m-1$ do
        $X[i] \leftarrow M[i] \oplus Y[i-1]$
        $Y[i] \leftarrow E_K(X[i])$
if $|M[m]| = n$ then $X[m] \leftarrow M[m] \oplus L \cdot \mathtt{u}$
               else $X[m] \leftarrow (M[m]10^{n-1-|M[m]|}) \oplus L \cdot \mathtt{u}^2$
$T \leftarrow E_K(X[m])$
return $T$
```

Fig. 2. Definition of OMAC1.

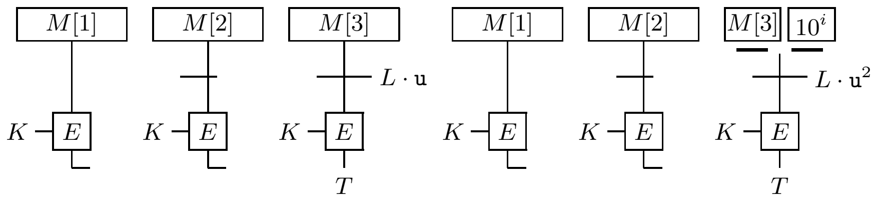

Fig. 3. Illustration of OMAC1.

2.3 OMAC1 [8]

OMAC1 takes just one k-bit key $K \in \{0,1\}^k$. It takes an arbitrary length message $M \in \{0,1\}^*$ to return an n-bit tag $T \in \{0,1\}^n$.

The algorithm of OMAC1 is described in Fig. 2 and illustrated in Fig. 3.

In Fig. 2 and Fig. 3,

$$L \cdot \mathtt{u} = \begin{cases} L \ll 1 & \text{if } \mathtt{msb}(L) = 0, \\ (L \ll 1) \oplus \mathtt{Cst}_n & \text{otherwise,} \end{cases}$$

where: (1) $\mathtt{msb}(L)$ denotes the most significant bit of L (meaning the left most bit), (2) $L \ll 1$ denotes the left shift of L by one bit (the most significant bit disappears and a zero comes into the least significant bit), and (3) $\mathtt{Cst}_n$ is an n-bit constant. For example, $\mathtt{Cst}_{64} = 0^{59}11011$ and $\mathtt{Cst}_{128} = 0^{120}10000111$.

$L \cdot \mathtt{u}^2$ is simply $(L \cdot \mathtt{u}) \cdot \mathtt{u}$. That is,

$$L \cdot \mathtt{u}^2 = \begin{cases} (L \cdot \mathtt{u}) \ll 1 & \text{if } \mathtt{msb}(L \cdot \mathtt{u}) = 0, \\ ((L \cdot \mathtt{u}) \ll 1) \oplus \mathtt{Cst}_n & \text{otherwise.} \end{cases}$$

2.4 A New Variant of OMAC1: OMAC1″ [12]

Mitchell proposed OMAC1″ [12]. Similarly to OMAC1, OMAC1″ takes just one k-bit key $K \in \{0,1\}^k$. It takes an arbitrary length message $M \in \{0,1\}^*$ to return an n-bit tag $T \in \{0,1\}^n$.

```
Algorithm $\mathrm{OMAC1}''_K(M)$
$L_1 \leftarrow K \oplus S_1$
$L_2 \leftarrow E_K(S_2)$
$L_3 \leftarrow E_K(S_3)$
$Y[0] \leftarrow 0^n$
Let $M = M[1] \cdots M[m]$, where $|M[i]| = n$ for $i = 1, \ldots, m-1$
for $i \leftarrow 1$ to $m-1$ do
        $X[i] \leftarrow M[i] \oplus Y[i-1]$
        $Y[i] \leftarrow E_{L_1}(X[i])$
if $|M[m]| = n$ then $X[m] \leftarrow M[m] \oplus L_2$
                else $X[m] \leftarrow (M[m]10^{n-1-|M[m]|}) \oplus L_3$
$T \leftarrow E_{L_1}(X[m])$
return $T$
```

Fig. 4. Definition of OMAC1$''$.

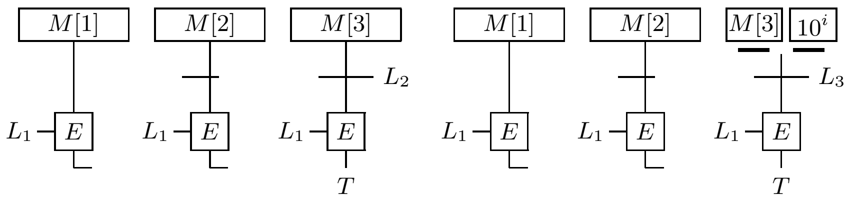

Fig. 5. Illustration of OMAC1$''$. Note that $L_1 = K \oplus S_1$, $L_2 = E_K(S_2)$ and $L_3 = E_K(S_3)$.

The algorithm of OMAC1$''$ is described in Fig. 4 and illustrated in Fig. 5.

In Fig. 4 and Fig. 5, S_1 is some fixed k-bit constant, S_2 and S_3 are some distinct n-bit constants.

2.5 Another Variant of OMAC1: OMAC1$'$

We now consider another variant of OMAC1, called OMAC1$'$. OMAC1$'$ takes just one k-bit key $K \in \{0,1\}^k$. It takes an arbitrary length message $M \in \{0,1\}^*$ to return an n-bit tag $T \in \{0,1\}^n$.

The algorithm of OMAC1$'$ is described in Fig. 6 and illustrated in Fig. 7.

In Fig. 6 and Fig. 7, S_2 and S_3 are some distinct n-bit constants.

3 OMAC1$'$ Is Completely Insecure

We show two equally efficient attacks against OMAC1$'$.

3.1 Attack 1

The adversary first obtains a tag $T \in \{0,1\}^n$ for a two block message $M' = (S_2, S_2) \in \{0,1\}^{2n}$. Then it outputs (M, T), where $M = S_2 \oplus T$, as a forgery attempt.

```
**Algorithm** $\mathrm{OMAC1'}_K(M)$
$L_2 \leftarrow E_K(S_2)$
$L_3 \leftarrow E_K(S_3)$
$Y[0] \leftarrow 0^n$
Let $M = M[1] \cdots M[m]$, where $|M[i]| = n$ for $i = 1, \ldots, m-1$
**for** $i \leftarrow 1$ **to** $m-1$ **do**
        $X[i] \leftarrow M[i] \oplus Y[i-1]$
        $Y[i] \leftarrow E_K(X[i])$
**if** $|M[m]| = n$ **then** $X[m] \leftarrow M[m] \oplus L_2$
                     **else** $X[m] \leftarrow (M[m]10^{n-1-|M[m]|}) \oplus L_3$
$T \leftarrow E_K(X[m])$
**return** $T$
```

Fig. 6. Definition of OMAC1$'$.

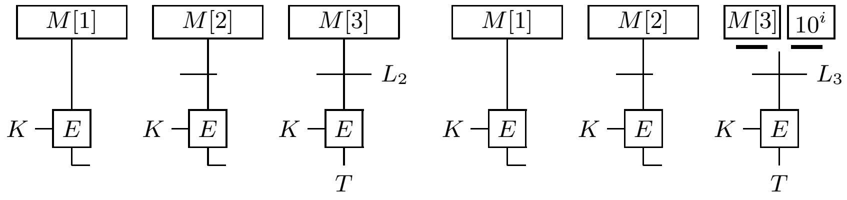

Fig. 7. Illustration of OMAC1$'$. Note that $L_2 = E_K(S_2)$ and $L_3 = E_K(S_3)$.

3.2 Analysis of Attack 1

For a message $M' = (S_2, S_2) \in \{0,1\}^{2n}$, we have

$$T = \mathrm{OMAC1}'_K(M') = E_K(E_K(S_2) \oplus S_2 \oplus L_2) \ .$$

Since $L_2 = E_K(S_2)$, we have

$$E_K(E_K(S_2) \oplus S_2 \oplus L_2) = E_K(L_2 \oplus S_2 \oplus L_2) = E_K(S_2) = L_2 \ .$$

Therefore, $T = L_2$. See Fig. 8.

Now for a message $M = S_2 \oplus T$ in forgery attempt, we have

$$\mathrm{OMAC1}'_K(M) = \mathrm{OMAC1}'_K(S_2 \oplus T) = E_K(S_2 \oplus T \oplus L_2) \ .$$

Since $T = L_2$, we have

$$E_K(S_2 \oplus T \oplus L_2) = E_K(S_2 \oplus L_2 \oplus L_2) = E_K(S_2) = T \ .$$

Therefore, our adversary in Sect. 3.1 forges with probability 1. See Fig. 9.

3.3 Attack 2

The adversary first fix some $M' \in \{0,1\}^*$ such that $1 \le |M'| < n$, and then obtains a tag $T \in \{0,1\}^n$ for a two block message $M'' = (S_3, M')$. Then it outputs (M, T), where $M = (M'10^{n-1-|M'|}, S_3 \oplus T, M')$, as a forgery attempt.

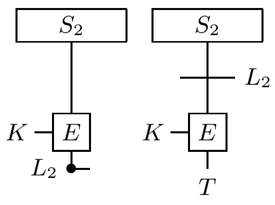

Fig. 8. Illustration of adversary's query. Note that $T = L_2$.

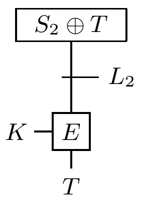

Fig. 9. Illustration of adversary's forgery attempt. We see that $T = \mathrm{OMAC1}'_K(S_2 \oplus T)$.

3.4 Analysis of Attack 2

For a message $M'' = (S_3, M')$, we have

$$T = \mathrm{OMAC1}'_K(M'') = E_K(E_K(S_3) \oplus (M'10^{n-1-|M'|}) \oplus L_3) \enspace .$$

Since $L_3 = E_K(S_3)$, we have

$$\begin{aligned} E_K(E_K(S_3) \oplus (M'10^{n-1-|M'|}) \oplus L_3) &= E_K(L_3 \oplus (M'10^{n-1-|M'|}) \oplus L_3) \\ &= E_K(M'10^{n-1-|M'|}) \enspace . \end{aligned}$$

Therefore, $T = E_K(M'10^{n-1-|M'|})$. See Fig. 10.

Now for a message $M = (M'10^{n-1-|M'|}, S_3 \oplus T, M')$ in forgery attempt, we have

$$\mathrm{OMAC1}'_K(M) = E_K(E_K(E_K(M'10^{n-1-|M'|}) \oplus S_3 \oplus T) \oplus (M'10^{n-1-|M'|}) \oplus L_3) \enspace .$$

Since $T = E_K(M'10^{n-1-|M'|})$, we have

$$\begin{aligned} \mathrm{OMAC1}'_K(M) &= E_K(E_K(T \oplus S_3 \oplus T) \oplus (M'10^{n-1-|M'|}) \oplus L_3) \\ &= E_K(E_K(S_3) \oplus (M'10^{n-1-|M'|}) \oplus L_3) \enspace . \end{aligned}$$

Since $L_3 = E_K(S_3)$, we have

$$\begin{aligned} \mathrm{OMAC1}'_K(M) &= E_K(L_3 \oplus (M'10^{n-1-|M'|}) \oplus L_3) \\ &= E_K(M'10^{n-1-|M'|}) \\ &= T \enspace . \end{aligned}$$

Therefore, our adversary in Sect. 3.3 forges with probability 1. See Fig. 11.

3.5 Theorem

We have the following theorem.

Theorem 3.1. OMAC1′ *is not a secure MAC. There exists an adversary* $\mathcal{A}$ *that makes 1 query and* $\mathtt{Adv}^{\mathtt{mac}}_{\mathrm{OMAC1}'}(\mathcal{A}) = 1$.

Proof. From Sect. 3.1 and 3.3. □

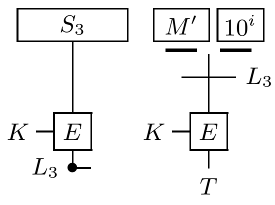

Fig. 10. Illustration of adversary's query. We have $T = E_K(M'10^{n-1-|M'|})$.

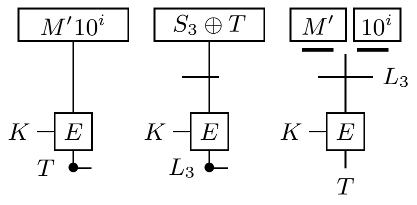

Fig. 11. Illustration of adversary's forgery attempt. We see that $T = \mathrm{OMAC1}'_K(M)$.

4 OMAC1″ Is *Less Secure* Than OMAC1

In this section, we first construct a PRP G with the following property: For any $K \in \{0,1\}^k$,

$$G_K(\cdot) = G_{K \oplus S_1}(\cdot) \ ,$$

where S_1 is a non-zero k-bit constant. We then show that OMAC1″ is completely insecure if G is used as the underlying block cipher. This implies underlying block cipher being a PRP is *not enough* for proving the security of OMAC1″. Equivalently, it is *impossible* for OMAC1″ to prove its security under the assumption of the underlying block cipher being a PRP. That is,

– OMAC1 is a secure MAC if the underlying block cipher is a PRP [8], while
– it is *impossible* for OMAC1″ to achieve this security notion.

Therefore, there is a security gap between OMAC1 and OMAC1″, and OMAC1″ is *less secure* than OMAC1.

4.1 Construction of a PRP, *G*

Let $E : \{0,1\}^{k-1} \times \{0,1\}^n \to \{0,1\}^n$ be a block cipher. It uses a $(k-1)$-bit key K' to encrypt an n-bit plaintext X into an n-bit ciphertext $Y = E_{K'}(X)$, where $E_{K'}(X) \overset{\mathrm{def}}{=} E(K', X)$. For each $K' \in \{0,1\}^{k-1}$, $E_{K'}(\cdot)$ is a permutation over $\{0,1\}^n$.

Now we construct a new block cipher $G : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ from E as in Fig. 12. The inputs to the algorithm are a block cipher E and some non-zero k-bit constant S_1. The output is a new block cipher G.

– For a k-bit string $S_1 = (s_0, s_1, \ldots, s_{k-1})$, $\mathtt{nzi}(S_1)$ denotes the smallest index of non-zero element. That is, $\mathtt{nzi}(S_1) = j$ such that $s_0 = \cdots = s_{j-1} = 0$ and $s_j = 1$. For example, if $k = 4$ and $S_1 = \mathtt{0xA} = \mathtt{1010}$, then $\mathtt{nzi}(S_1) = 0$, and if $S_1 = \mathtt{0x5} = \mathtt{0101}$, then $\mathtt{nzi}(S_1) = 1$.
– $\mathtt{num2str}_{k-1}(i)$ is a $(k-1)$-bit binary representation of i. For example, if $k = 4$ then $\mathtt{num2str}_{k-1}(0) = (0,0,0)$ and $\mathtt{num2str}_{k-1}(6) = (1,1,0)$.

```
Construction of $G$ from $E$ and $S_1$
$j \leftarrow \mathtt{nzi}(S_1)$;
for $i = 0$ to $2^{k-1} - 1$ do {
        $K' \leftarrow \mathtt{num2str}_{k-1}(i)$;
        $K_1 \leftarrow \mathtt{first}_{0..j-1}(K') \| 0 \| \mathtt{last}_{j..k-2}(K')$;
        $K_2 \leftarrow K_1 \oplus S_1$;
        $G_{K_1} \leftarrow E_{K'}$;
        $G_{K_2} \leftarrow E_{K'}$; }
```

Fig. 12. Construction of G from E and S_1.

- For a $(k-1)$-bit string $K' = (K'_0, \ldots, K'_{k-2})$ and an integer $0 \le j \le k-1$, $\mathtt{first}_{0..j-1}(K')$ denotes the first j bits of K'. That is, $\mathtt{first}_{0..j-1}(K') = (K'_0, \ldots, K'_{j-1})$. For example, if $j = 2$ and $K' = (1,1,0)$ then we have $\mathtt{first}_{0..j-1}(K') = (1,1)$, and if $j = 1$ and $K' = (0,1,0)$ then we have $\mathtt{first}_{0..j-1}(K') = (0)$. If $j = 0$, then $\mathtt{first}_{0..j-1}(K')$ is an empty string.
- Similarly, for a $(k-1)$-bit string $K' = (K'_0, \ldots, K'_{k-2})$ and an integer $0 \le j \le k-1$, $\mathtt{last}_{j..k-2}(K')$ denotes the last $(k-1) - j$ bits of K'. That is, $\mathtt{last}_{j..k-2}(K') = (K'_j, \ldots, K'_{k-2})$. For example, if $j = 2$ and $K' = (1,1,0)$ then $\mathtt{last}_{j..k-2}(K') = (0)$, and if $j = 1$ and $K' = (0,1,0)$ then $\mathtt{last}_{j..k-2}(K') = (1,0)$. If $j = k-1$, then $\mathtt{last}_{j..k-2}(K')$ is an empty string.
- $a\|b$ denotes the concatenation of a and b. For example, if $a = 1$ and $b = (1,0,1)$ then $a\|b = (1,1,0,1)$.

Observe that G_K is well defined for all $K \in \{0,1\}^k$. Indeed, "for loop" in the third line contains 2^{k-1} iterations, and for each loop, two Gs are assigned. Let $K'^{(i)}$, $K_1^{(i)}$ and $K_2^{(i)}$ denote K', K_1 and K_2 in the i-th iteration. Then we see that for any distinct i and i',

- $K_1^{(i)} \neq K_1^{(i')}$ and $K_2^{(i)} \neq K_2^{(i')}$ (since $K'^{(i)} \neq K'^{(i')}$), and
- $K_1^{(i)} \neq K_2^{(i')}$ and $K_2^{(i)} \neq K_1^{(i')}$ (since they differ in the j-th bit).

That is, $K_1^{(i)}$ and $K_2^{(i)}$ in the i-th iteration will not be assigned in the i'-th iteration.

Also observe that we have, for any $K \in \{0,1\}^k$, $G_K(\cdot) = G_{K \oplus S_1}(\cdot)$.

We show two small examples. First, let $k = 4$, $S_1 = \mathtt{0xA} = \mathtt{1010}$ and

$$E = \{E_{000}, E_{001}, E_{010}, E_{011}, E_{100}, E_{101}, E_{110}, E_{111}\},$$

where each $E_{K'}$ is a permutation over $\{0,1\}^n$. In this case, $j = 0$, and for $K' = (K'_0, K'_1, K'_2)$, $K_1 = (0, K'_0, K'_1, K'_2)$, and $K_2 = (1, K'_0, K'_1 \oplus 1, K'_2)$. Then we obtain

$$\begin{aligned} G = \{&G_{0000}, G_{0001}, G_{0010}, G_{0011}, G_{0100}, G_{0101}, G_{0110}, G_{0111},\\ &G_{1000}, G_{1001}, G_{1010}, G_{1011}, G_{1100}, G_{1101}, G_{1110}, G_{1111}\} \end{aligned}$$

```
Algorithm $\mathcal{A}^{\mathcal{O}}$
when $\mathcal{B}$ asks its $r$-th query $X_r$:
        return $\mathcal{O}(X_r)$;
when $\mathcal{B}$ halts and output $b$:
        output $b$;
```

Fig. 13. Construction of $\mathcal{A}$.

where

$$\begin{cases} G_{0000} = E_{000}, G_{0001} = E_{001}, G_{0010} = E_{010}, G_{0011} = E_{011}, \\ G_{0100} = E_{100}, G_{0101} = E_{101}, G_{0110} = E_{110}, G_{0111} = E_{111}, \\ G_{1000} = E_{010}, G_{1001} = E_{011}, G_{1010} = E_{000}, G_{1011} = E_{001}, \\ G_{1100} = E_{110}, G_{1101} = E_{111}, G_{1110} = E_{100}, G_{1111} = E_{101}. \end{cases}$$

Next, let $k = 4$, and $S_1 = \texttt{0x5} = \texttt{0101}$. In this case, $j = 1$, and for $K' = (K'_0, K'_1, K'_2)$, $K_1 = (K'_0, 0, K'_1, K'_2)$, and $K_2 = (K'_0, 1, K'_1, K'_2 \oplus 1)$. Then we obtain

$$\begin{cases} G_{0000} = E_{000}, G_{0001} = E_{001}, G_{0010} = E_{010}, G_{0011} = E_{011}, \\ G_{0100} = E_{001}, G_{0101} = E_{000}, G_{0110} = E_{011}, G_{0111} = E_{010}, \\ G_{1000} = E_{100}, G_{1001} = E_{101}, G_{1010} = E_{110}, G_{1011} = E_{111}, \\ G_{1100} = E_{101}, G_{1101} = E_{100}, G_{1110} = E_{111}, G_{1111} = E_{110}. \end{cases}$$

We note that G can be computed efficiently if E can be computed efficiently. Suppose that we are given a k-bit key K and a plaintext X, and we want to compute $G_K(X)$. Now, let $j \leftarrow \texttt{nzi}(S_1)$, and check if the j-th bit of K is zero. If it is, let $K' \leftarrow \texttt{first}_{0..j-1}(K) \| \texttt{last}_{j+1..k-1}(K)$ and return $E_{K'}(X)$. Otherwise let $K' \leftarrow \texttt{first}_{0..j-1}(K \oplus S_1) \| \texttt{last}_{j+1..k-1}(K \oplus S_1)$ and return $E_{K'}(X)$.

We now show that if E is a PRP, then G is a PRP. More precisely, we have the following theorem.

Theorem 4.1. *If $\texttt{Adv}_E^{\textsf{prp}}(\mathcal{A}) \le \epsilon$ for any adversary $\mathcal{A}$ that makes at most q queries, then $\texttt{Adv}_G^{\textsf{prp}}(\mathcal{B}) \le \epsilon$ for any adversary $\mathcal{B}$ that makes at most q queries.*

Proof. We prove through a contradiction argument. Suppose that there exists an adversary $\mathcal{B}$ such that $\texttt{Adv}_G^{\textsf{prp}}(\mathcal{B}) > \epsilon$ where $\mathcal{B}$ asks at most q queries. By using $\mathcal{B}$, we construct an adversary $\mathcal{A}$ such that $\texttt{Adv}_E^{\textsf{prp}}(\mathcal{A}) > \epsilon$ where $\mathcal{A}$ asks at most q queries.

The construction is given in Fig. 13. $\mathcal{A}$ has an oracle $\mathcal{O}$ (either P or $E_{K'}$), and $\mathcal{A}$ simply uses $\mathcal{O}$ to answer $\mathcal{B}$'s queries. Finally $\mathcal{A}$ outputs b which is the output of $\mathcal{B}$.

First, suppose that $\mathcal{O} = P$. Then $\mathcal{A}$ gives $\mathcal{B}$ a perfect simulation of a random permutation. Therefore, we have

$$\Pr(P \overset{R}{\leftarrow} \mathrm{Perm}(n) : \mathcal{B}^{P(\cdot)} = 1) = \Pr(P \overset{R}{\leftarrow} \mathrm{Perm}(n) : \mathcal{A}^{P(\cdot)} = 1) \ .$$

Next, suppose that $\mathcal{O} = E_{K'}$. Then $\mathcal{A}$ gives $\mathcal{B}$ a perfect simulation of G, since from the $\mathcal{B}$'s point of view, each

$$E_{0,\ldots,0}, \ldots, E_{1,\ldots,1}$$

is chosen with probability $1/2^{k-1} = 2/2^k$, which is a precise simulation of G. Note that G is

$$E_{0,\ldots,0}, E_{0,\ldots,0}, \ldots, E_{1,\ldots,1}, E_{1,\ldots,1}$$

and each $E_{K'}$ is chosen with probability $2/2^k$. Therefore, we have

$$\Pr(K \overset{R}{\leftarrow} \{0,1\}^k : \mathcal{B}^{G_K(\cdot)} = 1) = \Pr(K' \overset{R}{\leftarrow} \{0,1\}^{k-1} : \mathcal{A}^{E_{K'}(\cdot)} = 1) \ .$$

⊓

4.2 OMAC1″[G] Is Completely Insecure

Let OMAC1″[G] denote OMAC1″, where G is used as the underlying block cipher.

We have the following theorem.

Theorem 4.2. OMAC1″[G] *is not a secure MAC. There exists an adversary* $\mathcal{A}$ *that makes 1 query and* $\mathtt{Adv}^{\mathsf{mac}}_{\mathrm{OMAC1''}[G]}(\mathcal{A}) = 1$.

Proof. Since we have

$$G_K(\cdot) = G_{K \oplus S_1}(\cdot)$$

for any k-bit key $K \in \{0,1\}^k$, attacks in Sect. 3.1 and 3.3 can be applied to OMAC1″[G]. ⊓

5 Conclusion

In this paper, we showed that OMAC1″ proposed in [12] are *less secure* than OMAC1. More precisely, we showd that it is *impossible* for OMAC1″ to prove its security under the assumption of the underlying block cipher being a PRP.

References

1. 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. 394–405, IEEE, 1997.
2. M. Bellare, J. Kilian, and P. Rogaway. The security of the cipher block chaining message authentication code. *JCSS,* vol. 61, no. 3, 2000. Earlier version in *Advances in Cryptology — CRYPTO '94, LNCS 839,* pp. 341–358, Springer-Verlag, 1994.
3. A. Berendschot, 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. Final Report of RACE Integrity Primitives. *LNCS 1007,* Springer-Verlag, 1995.

4. J. Black and P. Rogaway. CBC MACs for arbitrary-length messages: The three key constructions. *Advances in Cryptology — CRYPTO 2000, LNCS 1880,* pp. 197–215, Springer-Verlag, 2000.
5. J. Black and P. Rogaway. A block-cipher mode of operation for parallelizable message authentication. *Advances in Cryptology — EUROCRYPT 2002, LNCS 2332,* pp. 384–397, Springer-Verlag, 2002.
6. FIPS 113. Computer data authentication. Federal Information Processing Standards Publication 113, U. S. Department of Commerce / National Bureau of Standards, National Technical Information Service, Springfield, Virginia, 1994.
7. ISO/IEC 9797-1. Information technology — security techniques — data integrity mechanism using a cryptographic check function employing a block cipher algorithm. International Organization for Standards, Geneva, Switzerland, 1999. Second edition.
8. T. Iwata and K. Kurosawa. OMAC: One-Key CBC MAC. Pre-proceedings of *Fast Software Encryption, FSE 2003,* pp. 137–162, 2003. To appear in *LNCS,* Springer-Verlag. See `http://crypt.cis.ibaraki.ac.jp/`.
9. T. Iwata and K. Kurosawa. On the correctness of security proofs for the 3GPP confidentiality and integrity algorithms. To appear in *Ninth IMA International Conference on Cryptography and Coding, LNCS,* Springer-Verlag.
10. K. Kurosawa and T. Iwata. TMAC: Two-Key CBC MAC. *Topics in Cryptology — CT-RSA 2003, The Cryptographers' Track at RSA Conference 2003, LNCS 2612,* pp. 33–49, Springer-Verlag, 2003.
11. M. Luby and C. Rackoff. How to construct pseudorandom permutations from pseudorandom functions. *SIAM J. Comput.,* vol. 17, no. 2, pp. 373–386, April 1988.
12. C.J. Mitchell. On the security of XCBC, TMAC and OMAC. Technical Report RHUL-MA-2003-4, 19 August, 2003. Available at `http://www.rhul.ac.uk/mathematics/techreports`. Also available from NIST's web page at `http://csrc.nist.gov/CryptoToolkit/modes/comments/`.
13. E. Petrank and C. Rackoff. CBC MAC for real-time data sources. *J.Cryptology,* vol. 13, no. 3, pp. 315–338, Springer-Verlag, 2000.
14. P. Rogaway. Comments on NIST's RMAC proposal. Comments to NIST. Available at `http://www.cs.ucdavis.edu/~rogaway/xcbc/index.html`. Also available at `http://csrc.nist.gov/CryptoToolkit/modes/comments/`.
15. J. Sung, D. Hong, and S. Lee. Key recovery attacks on the RMAC, TMAC, and IACBC. *The Eighth Australasian Conference on Information Security and Privacy, ACISP 2003, LNCS 2727,* pp. 265–273, Springer-Verlag, 2003.