# OMAC: One-Key CBC MAC — Addendum

Tetsu Iwata 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

March 10, 2003

## 1 Introduction

In [2], we showed OMAC-family and suggested to use OMAC as a concrete choice of the parameters, where each member of OMAC-family is a provably secure CBC-type MAC scheme for any message length which uses only *one* key.

In this note, we propose OMAC1, a new choice of the parameters of OMAC-family (see [4] for the details). Test vectors are also presented.

Accordingly, we rename the previous OMAC as OMAC2. (That is to say, test vectors for OMAC2 were already shown in [3].) We use OMAC as a generic name for OMAC1 and OMAC2.

## 2 Specification of OMAC1

Each member of OMAC-family is obtained by specifying

- a block cipher $E : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$,
- an $n$-bit constant $\mathtt{Cst}$,
- a universal hash function $H : \{0,1\}^n \times X \to \{0,1\}^n$ and two distinct constants $\mathtt{Cst}_1, \mathtt{Cst}_2 \in X$ which satisfy some conditions,

where $k$ is a key length and $n$ is a block length. It takes a block cipher key $K \in \{0,1\}^k$ and a message $M \in \{0,1\}^*$, and returns a tag $T \in \{0,1\}^n$.

Now OMAC1, a new choice of the parameters, is specified by

$$\mathtt{Cst} = 0^n, H_L(x) = L \cdot x, \mathtt{Cst}_1 = \mathtt{u}, \mathtt{Cst}_2 = \mathtt{u}^2,$$

where "$\cdot$" denotes multiplication over GF($2^n$). Equivalently,

$$L = E_K(0^n), H_L(\mathtt{Cst}_1) = L \cdot \mathtt{u}, H_L(\mathtt{Cst}_2) = L \cdot \mathtt{u}^2.$$

OMAC1 is the same as OMAC2 (which is the previous OMAC) except for that $\mathtt{Cst}_2 = \mathtt{u}^2$ instead of $\mathtt{Cst}_2 = \mathtt{u}^{-1}$. For comparison, OMAC2 was specified by

$$L = E_K(0^n), H_L(\mathtt{Cst}_1) = L \cdot \mathtt{u}, H_L(\mathtt{Cst}_2) = L \cdot \mathtt{u}^{-1}.$$

Note that

```
Algorithm $\mathrm{OMAC1}_K(M)$
$L \leftarrow E_K(0^n)$
$Y[0] \leftarrow 0^n$
Partition $M$ into $M[1] \cdots M[m]$
for $i \leftarrow 1$ to $m-1$ do
        $X[i] \leftarrow M[i] \oplus Y[i-1]$
        $Y[i] \leftarrow E_K(X[i])$
$X[m] \leftarrow \mathtt{pad}_n(M[m]) \oplus Y[m-1]$
if $|M[m]| = n$
        then $X[m] \leftarrow X[m] \oplus L \cdot \mathtt{u}$
        else $X[m] \leftarrow X[m] \oplus L \cdot \mathtt{u}^2$
$T \leftarrow E_K(X[m])$
return $T$
```

```
Algorithm $\mathrm{OMAC2}_K(M)$
$L \leftarrow E_K(0^n)$
$Y[0] \leftarrow 0^n$
Partition $M$ into $M[1] \cdots M[m]$
for $i \leftarrow 1$ to $m-1$ do
        $X[i] \leftarrow M[i] \oplus Y[i-1]$
        $Y[i] \leftarrow E_K(X[i])$
$X[m] \leftarrow \mathtt{pad}_n(M[m]) \oplus Y[m-1]$
if $|M[m]| = n$
        then $X[m] \leftarrow X[m] \oplus L \cdot \mathtt{u}$
        else $X[m] \leftarrow X[m] \oplus L \cdot \mathtt{u}^{-1}$
$T \leftarrow E_K(X[m])$
return $T$
```

**Fig. 1.** Description of OMAC1 and OMAC2.

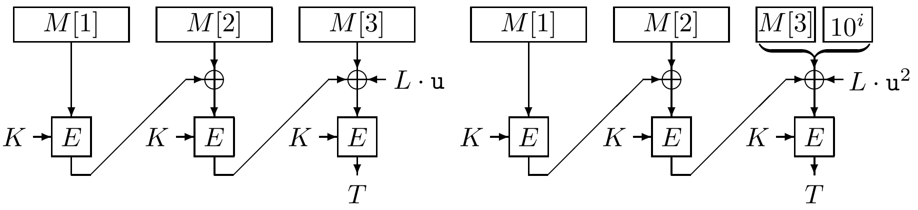

**Fig. 2.** Illustration of OMAC1. Note that $L = E_K(0^n)$.

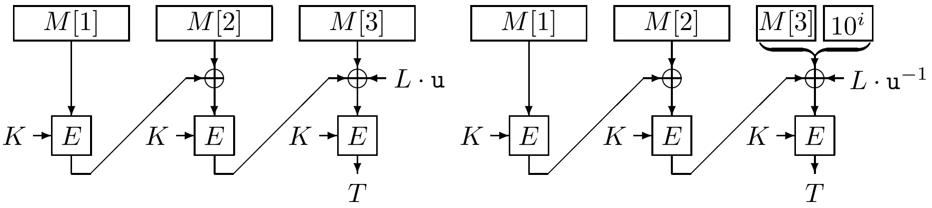

**Fig. 3.** Illustration of OMAC2.

1. $L \cdot \mathtt{u}$ and $L \cdot \mathtt{u}^2 = (L \cdot \mathtt{u}) \cdot \mathtt{u}$ can be computed efficiently from $L$ and $L \cdot \mathtt{u}$ only by one shift and one conditional XOR, respectively.
2. In both OMAC1 and OMAC2, $\epsilon_1 = \cdots = \epsilon_6 = 2^{-n}$ in where $\epsilon_1, \ldots, \epsilon_6$ are defined in [2, Section 3]. This implies that there is no security difference between OMAC1 and OMAC2 (see [4]).

OMAC1 and OMAC2 are described in Fig. 1 and illustrated in Fig. 2 and Fig. 3.

## 3 Discussions

- For OMAC1, we adopted $\mathtt{u}$ and $\mathtt{u}^2$ as $\mathtt{Cst}_1$ and $\mathtt{Cst}_2$, since $L \cdot \mathtt{u}$ and $L \cdot \mathtt{u}^2 = (L \cdot \mathtt{u}) \cdot \mathtt{u}$ can be computed efficiently by one left shift and one conditional XOR from $L$ and $L \cdot \mathtt{u}$, respectively. Note that this choice requires only a left shift. This would ease the implementation of OMAC1, especially in hardware.

- For OMAC2, we adopted $\mathtt{u}^{-1}$ instead of $\mathtt{u}^2$ as $\mathtt{Cst}_2$. It requires one right shift to compute $L \cdot \mathtt{u}^{-1}$ instead of one left shift to compute $(L \cdot \mathtt{u}) \cdot \mathtt{u}$. This would allow to compute both $L \cdot \mathtt{u}$ and $L \cdot \mathtt{u}^{-1}$ from $L$ simultaneously if both left shift and right shift are available (for example, the underlying block cipher uses both shifts).

## 4 OMAC1 Test Vectors

In this section, we consider OMAC1 such that the AES [1] is used as the underlying block cipher. Hence the tag length is $n = 128$ bits and the key consists of only the key of the AES.

For each of $k = 128$, 192, and 256 bits (the allowed key sizes of the AES), we present 4 test vectors. Therefore 12 test vectors are given in total.

In what follows, the AES key is denoted by "$K$," the message is denoted by "Msg," and the output of OMAC1 is denoted by "Tag." All strings are expressed in hexadecimal notation. (We use the same $K$ and Msg as in [3].)

### 4.1 AES-128

**Test Vector for the Empty String**

| | |
|---|---|
| $K$ | `2b7e151628aed2a6abf7158809cf4f3c` |
| Msg | ⟨empty string⟩ |
| Tag | `bb1d6929e95937287fa37d129b756746` |

**Test Vector for 16-Byte Message**

| | |
|---|---|
| $K$ | `2b7e151628aed2a6abf7158809cf4f3c` |
| Msg | `6bc1bee22e409f96e93d7e117393172a` |
| Tag | `070a16b46b4d4144f79bdd9dd04a287c` |

**Test Vector for 40-Byte Message**

| | |
|---|---|
| $K$ | `2b7e151628aed2a6abf7158809cf4f3c` |
| Msg | `6bc1bee22e409f96e93d7e117393172a`<br>`ae2d8a571e03ac9c9eb76fac45af8e51`<br>`30c81c46a35ce411` |
| Tag | `dfa66747de9ae63030ca32611497c827` |

**Test Vector for 64-Byte Message**

| | |
|---|---|
| $K$ | `2b7e151628aed2a6abf7158809cf4f3c` |
| Msg | `6bc1bee22e409f96e93d7e117393172a`<br>`ae2d8a571e03ac9c9eb76fac45af8e51`<br>`30c81c46a35ce411e5fbc1191a0a52ef`<br>`f69f2445df4f9b17ad2b417be66c3710` |
| Tag | `51f0bebf7e3b9d92fc49741779363cfe` |

### 4.2 AES-192

**Test Vector for the Empty String**

| | |
|---|---|
| $K$ | `8e73b0f7da0e6452c810f32b809079e5`<br>`62f8ead2522c6b7b` |
| Msg | ⟨empty string⟩ |
| Tag | `d17ddf46adaacde531cac483de7a9367` |

### Test Vector for 16-Byte Message

| | |
|---|---|
| $K$ | `8e73b0f7da0e6452c810f32b809079e5`<br>`62f8ead2522c6b7b` |
| Msg | `6bc1bee22e409f96e93d7e117393172a` |
| Tag | `9e99a7bf31e710900662f65e617c5184` |

### Test Vector for 40-Byte Message

| | |
|---|---|
| $K$ | `8e73b0f7da0e6452c810f32b809079e5`<br>`62f8ead2522c6b7b` |
| Msg | `6bc1bee22e409f96e93d7e117393172a`<br>`ae2d8a571e03ac9c9eb76fac45af8e51`<br>`30c81c46a35ce411` |
| Tag | `8a1de5be2eb31aad089a82e6ee908b0e` |

### Test Vector for 64-Byte Message

| | |
|---|---|
| $K$ | `8e73b0f7da0e6452c810f32b809079e5`<br>`62f8ead2522c6b7b` |
| Msg | `6bc1bee22e409f96e93d7e117393172a`<br>`ae2d8a571e03ac9c9eb76fac45af8e51`<br>`30c81c46a35ce411e5fbc1191a0a52ef`<br>`f69f2445df4f9b17ad2b417be66c3710` |
| Tag | `a1d5df0eed790f794d77589659f39a11` |

## 4.3 AES-256

### Test Vector for the Empty String

| | |
|---|---|
| $K$ | `603deb1015ca71be2b73aef0857d7781`<br>`1f352c073b6108d72d9810a30914dff4` |
| Msg | ⟨empty string⟩ |
| Tag | `028962f61b7bf89efc6b551f4667d983` |

### Test Vector for 16-Byte Message

| | |
|---|---|
| $K$ | `603deb1015ca71be2b73aef0857d7781`<br>`1f352c073b6108d72d9810a30914dff4` |
| Msg | `6bc1bee22e409f96e93d7e117393172a` |
| Tag | `28a7023f452e8f82bd4bf28d8c37c35c` |

### Test Vector for 40-Byte Message

| | |
|---|---|
| $K$ | `603deb1015ca71be2b73aef0857d7781`<br>`1f352c073b6108d72d9810a30914dff4` |
| Msg | `6bc1bee22e409f96e93d7e117393172a`<br>`ae2d8a571e03ac9c9eb76fac45af8e51`<br>`30c81c46a35ce411` |
| Tag | `aaf3d8f1de5640c232f5b169b9c911e6` |

### Test Vector for 64-Byte Message

| | |
|---|---|
| $K$ | `603deb1015ca71be2b73aef0857d7781`<br>`1f352c073b6108d72d9810a30914dff4` |
| Msg | `6bc1bee22e409f96e93d7e117393172a`<br>`ae2d8a571e03ac9c9eb76fac45af8e51`<br>`30c81c46a35ce411e5fbc1191a0a52ef`<br>`f69f2445df4f9b17ad2b417be66c3710` |
| Tag | `e1992190549f6ed5696a2c056c315410` |

## Acknowledgement

The authors would like to thank Phillip Rogaway of UC Davis, who suggested to use $\mathtt{Cst}_2 = \mathtt{u}^2$. We also thank Eisuke Kuroda and Yuki Ohira of Ibaraki University for implementing OMAC and checking the test vectors.

## References

[1] FIPS Publication 197. Advanced Encryption Standard (AES). Available at `http://csrc.nist.gov/encryption/aes/`.

[2] T. Iwata and K. Kurosawa. OMAC: One-Key CBC MAC. NIST submission, December 20, 2002, Available at `http://csrc.nist.gov/CryptoToolkit/modes/proposedmodes/`.

[3] T. Iwata and K. Kurosawa. OMAC Test Vectors. NIST submission, December 20, 2002, Available at `http://csrc.nist.gov/CryptoToolkit/modes/proposedmodes/`.

[4] T. Iwata and K. Kurosawa. OMAC: One-Key CBC MAC. Proceedings version of [2]. Pre-proceedings of *Fast Software Encryption, FSE 2003,* pp. 137–161, February 2003. See Cryptology ePrint Archive, Report 2002/180 at `http://eprint.iacr.org/2002/180/`.