TMAC: Two-Key CBC MAC

Kaoru Kurosawa and Tetsu Iwata

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

Abstract. In this paper, we propose TMAC, Two-Key CBC Message Authentication Code. TMAC is a refinement of XCBC (which is a variant of CBC MAC) shown by Black and Rogaway. We use only $(k+n)$-bit key for TMAC while XCBC uses $(k+2n)$-bit key, where k is the key length of the underlying block cipher and n is its block length. The cost for reducing the size of secret keys is almost negligible; only one shift and one conditional XOR. Similarly to XCBC, our algorithm correctly and efficiently handles messages of arbitrary bit length.

1 Introduction

Let $E : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ be a block cipher: it uses a k-bit key $K \in \{0,1\}^k$ to encrypt an n-bit block $X \in \{0,1\}^n$ into an n-bit ciphertext $Y = E_K(X)$.

1.1 CBC MAC

The CBC MAC [8, 10] is the simplest and most well-known algorithm to make a MAC from a block cipher. Let $M = M_1||M_2||\cdots||M_m$ be a message string such that $|M_1| = |M_2| = \cdots = |M_m| = n$. Then $\mathrm{CBC}_{E_K}(M)$, the CBC MAC of M under key K, is defined as C_m, where

$$C_i = E_K(M_i \oplus C_{i-1})$$

for $i = 1, \ldots, m$ and $C_0 = 0^n$.

Bellare, Kilian, and Rogaway proved the security of the CBC MAC for fixed message length mn [3]. It is well known, however, that the CBC MAC is *not* secure if the message length varies.

1.2 EMAC

To deal with variable message length in blocks m, Encrypted MAC (EMAC) was developed. EMAC encrypts $\mathrm{CBC}_{E_{K_1}}(M)$ using a new block cipher key K_2. That is,

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

Submitted to NIST on June 21, 2002

EMAC was developed for the RACE project [4]. Petrank and Rackoff then proved the security [12].

A problem is that the message length is limited to a positive multiple of n, that is, the domain is limited to $(\{0,1\}^n)^+$. The simplest approach to deal with messages whose lengths are not a multiple of n is to append the minimal 10^i to M as a padding so that the length is a multiple of n. Note that the padding is appended even if the size of the message is already a multiple of n.

In this way, EMAC can deal with completely variable message length. In other words, the domain is $\{0,1\}^*$. We call this EMAC*.

1.3 RMAC

Jaulmes, Joux, and Valette proposed RMAC [11] which is an extension of EMAC. RMAC encrypts $\mathrm{CBC}_{E_{K_1}}(M)$ with $K_2 \oplus R$, where R is an n-bit random string and it is a part of the tag. That is,

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

They showed that the security of RMAC is beyond the birthday paradox limit. However, the tag length is n bits longer than the other CBC MAC variants.

1.4 XCBC

EMAC* and RMAC require $1 + \lceil (|M|+1)/n \rceil$ block cipher invocations. Black and Rogaway proposed XCBC [5] which requires only $\lceil |M|/n \rceil$ block cipher invocations.

XCBC takes three keys: one block cipher key K_1, and two n-bit keys K_2 and K_3. XCBC makes two cases to deal with arbitrary length messages: $M \in (\{0,1\})^+$ and $M \notin (\{0,1\})^+$. If $M \in (\{0,1\})^+$ then XCBC computes exactly the same as the CBC MAC, except XORing an n-bit key K_2 before encrypting the last block. If $M \notin (\{0,1\})^+$ then minimal 10^i padding $(i \geq 0)$ is appended to M so that the length is a multiple of n, and XCBC computes exactly the same as the CBC MAC, except XORing another n-bit key K_3 before encrypting the last block.

1.5 Our Contribution

The key length of XCBC is $(k + 2n)$ bits in total. To reduce the key length, the authors suggested the following solution [6] for $n \leq k \leq 2n$. A secret key is a single key K of E. Then for some distinct constants C_{1a}, C_{1b}, C_2, and C_3, let

$$\begin{aligned} K_1 &= \text{the first } k \text{ bits of } E_K(C_{1a}) || E_K(C_{1b}), \\ K_2 &= E_K(C_2), \\ K_3 &= E_K(C_3). \end{aligned}$$

This key derivation uses one k-bit key, but it has two problems:

1. The number of block cipher invocations is no longer optimal since it requires 3 or 4 additional block cipher invocations.
2. It needs two key schedulings for two block cipher keys K and K_1.

These problems may be significant if one frequently changes the secret key.

In this paper, we propose TMAC, Two-Key CBC Message Authentication Code. TMAC is a refinement of XCBC shown by Black and Rogaway. We use only $(k+n)$-bit key for TMAC while XCBC uses $(k+2n)$-bit key. The cost for reducing the size of secret keys is almost negligible; only one shift and one conditional XOR. Similarly to XCBC, the domain is $\{0,1\}^*$ and it requires $\lceil |M|/n \rceil$ block cipher invocations.

We show a comparison of CBC MAC and its variants in Table 1, where M is the message and E is a block cipher. The third column gives the number of invocations of E, assuming $|M| > 0$. The fourth column gives the number of different keys used for E.

Table 1. Comparison of CBC MAC and Its Variants.

Name	Domain	$\#E$ Invocations	$\#E$ Keys	Key Length
CBC MAC [8, 10, 3]	$(\{0,1\}^n)^m$	$\lvert M\rvert/n$	1	k
EMAC* [4, 12]	$\{0,1\}^*$	$1+\lceil(\lvert M\rvert+1)/n\rceil$	2	$2k$
RMAC [11]	$\{0,1\}^*$	$1+\lceil(\lvert M\rvert+1)/n\rceil$	2	$2k$
XCBC [5, 6]	$\{0,1\}^*$	$\lceil\lvert M\rvert/n\rceil$	1	$k+2n$
TMAC (Our proposal)	$\{0,1\}^*$	$\lceil\lvert M\rvert/n\rceil$	1	$k+n$

1.6 Other Related Works

Recently, some researchers proposed parallelizable MAC algorithms. Bellare, Guérin, and Rogaway proposed XOR MAC [2]. Gligor, and Donescu proposed XECB-MAC [9]. Black and Rogaway proposed PMAC [7].

However, these MAC algorithms have overhead as follows. XOR MAC requires much more invocations of E than the other MAC algorithms. XECB-MAC requires modulo 2^n arithmetic and three more invocations of E than XCBC and TMAC. PMAC needs to generate a sequence of masks.

Therefore, TMAC and XCBC are better than these algorithms in non-parallelizable environment.

2 Mathematical Preliminaries

2.1 Notation

If A is a finite set then $\#A$ denotes the number of elements in A. For a set A, $x \stackrel{R}{\leftarrow} A$ means that x is randomly chosen from A. If $\alpha \in \{0,1\}^*$ is a string then $|\alpha|$ denotes its length in bits. If $\alpha, \beta \in \{0,1\}^*$ are equal-length strings then $\alpha \oplus \beta$ is their bitwise XOR.

For an n-bit string $\alpha = a_{n-1} \cdots a_1 a_0 \in \{0,1\}^n$, let

$$\alpha \ll 1 = a_{n-2} a_{n-3} \cdots a_1 a_0 0 \ .$$

Similarly, let

$$\alpha \gg 1 = 0 a_{n-1} a_{n-2} \cdots a_2 a_1 \ .$$

2.2 The Field with 2^n Points

We interchangeably think of a point a in $\mathrm{GF}(2^n)$ in any of the following ways:

1. as an abstract point in a field;
2. as an n-bit string $a_{n-1} \cdots a_1 a_0 \in \{0,1\}^n$;
3. as a formal polynomial $a(\mathtt{u}) = a_{n-1}\mathtt{u}^{n-1} + \cdots + a_1\mathtt{u} + a_0$ with binary coefficients.

To add two points in $\mathrm{GF}(2^n)$, take their bitwise XOR. We denote this operation by $a \oplus b$.

To multiply two points, fix some irreducible polynomial $f(\mathtt{u})$ having binary coefficients and degree n. To be concrete, choose the lexicographically first polynomial among the irreducible degree n polynomials having a minimum number of coefficients. We list some indicated polynomials.

$$\begin{cases} f(\mathtt{u}) = \mathtt{u}^{64} + \mathtt{u}^4 + \mathtt{u}^3 + \mathtt{u} + 1 & \text{for } n = 64, \\ f(\mathtt{u}) = \mathtt{u}^{128} + \mathtt{u}^7 + \mathtt{u}^2 + \mathtt{u} + 1 & \text{for } n = 128, \text{ and} \\ f(\mathtt{u}) = \mathtt{u}^{256} + \mathtt{u}^{10} + \mathtt{u}^5 + \mathtt{u}^2 + 1 & \text{for } n = 256. \end{cases}$$

To multiply two points $a \in \mathrm{GF}(2^n)$ and $b \in \mathrm{GF}(2^n)$, regard a and b as polynomials $a(\mathtt{u}) = a_{n-1}\mathtt{u}^{n-1} + \cdots + a_1\mathtt{u} + a_0$ and $b(\mathtt{u}) = b_{n-1}\mathtt{u}^{n-1} + \cdots + b_1\mathtt{u} + b_0$, form their product $c(\mathtt{u})$ where one adds and multiplies coefficients in $\mathrm{GF}(2)$, and take the remainder when dividing $c(\mathtt{u})$ by $f(\mathtt{u})$.

Note that it is particularly easy to multiply a point $a \in \{0,1\}^n$ by $\mathtt{u}$. We show a method for $n = 128$, where $f(\mathtt{u}) = \mathtt{u}^{128} + \mathtt{u}^7 + \mathtt{u}^2 + \mathtt{u} + 1$. Then multiplying $a = a_{127} \cdots a_1 a_0$ by $\mathtt{u}$ yields a product $a_{n-1}\mathtt{u}^n + a_{n-2}\mathtt{u}^{n-1} + \cdots + a_1\mathtt{u}^2 + a_0\mathtt{u}$. Thus, if $a_{n-1} = 0$, then $a \cdot \mathtt{u} = a \ll 1$. If $a_{n-1} = 1$, then we must add $\mathtt{u}^{128}$ to $a \ll 1$. Since $\mathtt{u}^{128} + \mathtt{u}^7 + \mathtt{u}^2 + \mathtt{u} + 1 = 0$ we have $\mathtt{u}^{128} = \mathtt{u}^7 + \mathtt{u}^2 + \mathtt{u} + 1$, so adding $\mathtt{u}^{128}$ means to xor by $0^{120}10000111$. In summary, when $n = 128$,

$$a \cdot \mathtt{u} = \begin{cases} a \ll 1 & \text{if } a_{127} = 0, \\ (a \ll 1) \oplus 0^{120}10000111 & \text{otherwise,} \end{cases} \tag{1}$$

where $a \cdot \mathtt{u} = a(\mathtt{u}) \cdot \mathtt{u} \bmod f(\mathtt{u})$.

Also, note that it is easy to devide a point $a \in \{0,1\}^n$ by $\mathtt{u}$, meaning that one multiplies a by the multiplicative inverse of $\mathtt{u}$ in the field: $a \cdot \mathtt{u}^{-1}$. We show a method for $n = 128$. Then multiplying $a = a_{127} \cdots a_1 a_0$ by $\mathtt{u}^{-1}$ yields a product $a_{n-1}\mathtt{u}^{n-2} + a_{n-2}\mathtt{u}^{n-3} + \cdots + a_2\mathtt{u} + a_1 + a_0\mathtt{u}^{-1}$. Thus, if $a_0 = 0$, then $a \cdot \mathtt{u}^{-1} = a \gg 1$. If $a_0 = 1$, then we must add $\mathtt{u}^{-1}$ to $a \gg 1$. Since $\mathtt{u}^{128} + \mathtt{u}^7 + \mathtt{u}^2 + \mathtt{u} + 1 = 0$ we have $\mathtt{u}^{127} = \mathtt{u}^6 + \mathtt{u} + 1 + \mathtt{u}^{-1}$, so adding $\mathtt{u}^{-1} = \mathtt{u}^{127} + \mathtt{u}^6 + \mathtt{u} + 1$ means to xor by $10^{120}1000011$. In summary, whern $n = 128$,

$$a \cdot \mathtt{u}^{-1} = \begin{cases} a \gg 1 & \text{if } a_0 = 0, \\ (a \gg 1) \oplus 10^{120}1000011 & \text{otherwise.} \end{cases} \tag{2}$$

3 Specification

3.1 Basic Specification

To use TMAC, one must specify a block cipher E.

The block cipher E is a function $E : \mathcal{K}_E \times \{0,1\}^n \to \{0,1\}^n$, where each $E(K, \cdot) = E_K(\cdot)$ is a permutation on $\{0,1\}^n$, $\mathcal{K}_E$ is the set of possible keys and n is the block length. The popular block cipher to use with TMAC is likely to be AES, but any other block cipher is fine.

TMAC is a function taking two keys $K_1 \in \mathcal{K}_E$, $K_2 \in \{0,1\}^n$ and a message $M \in \{0,1\}^*$, and returning a string in $\{0,1\}^n$. The key space $\mathcal{K}$ of TMAC is $\mathcal{K} = \mathcal{K}_E \times \{0,1\}^n$. The function is defined in Fig. 1 and illustrated in Fig. 2.

```
Algorithm TMAC_{E_{K_1},K_2}(M)
if M ∈ ({0,1}^n)^+
        then K ← K_2 · u and P ← M
        else K ← K_2 and P ← M||10^i, where i ← n − 1 − |M| mod n
Let P = P_1||P_2|| ··· ||P_m, where |P_1| = |P_2| = ··· = |P_m| = n
C_0 ← 0^n
for i ← 1 to m − 1 do
        C_i ← E_{K_1}(P_i ⊕ C_{i−1})
return T = E_{K_1}(P_m ⊕ C_{m−1} ⊕ K)
```

Fig. 1. Definition of TMAC.

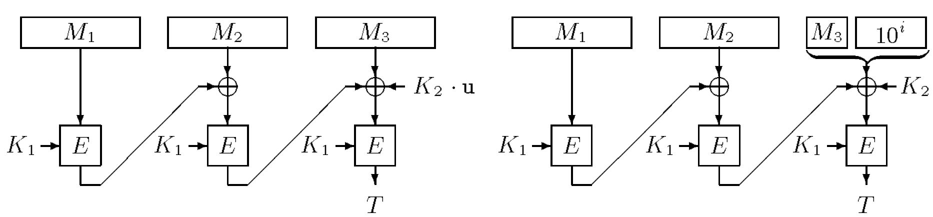

Fig. 2. Illustration of TMAC.

In the third line of Fig. 1 and in the last block of left hand side in Fig. 2, $K_2 \cdot \mathtt{u}$ is a multiplication in $\mathrm{GF}(2^n)$. It can be computed with only one shift and one conditional XOR as shown in (1).

3.2 User Option

We have two options on the computation of $K_2 \cdot \mathtt{u}$. The first option is to keep both K_2 and $K_2 \cdot \mathtt{u}$ in the memory. It uses a memory of $2n$ bits.

The second option uses a memory of only n bits. We first keep K_2 in the memory. When $K_2 \cdot \mathtt{u}$ is needed, we compute $K_2 \cdot \mathtt{u}$ from K_2. We then replace K_2 with $K_2 \cdot \mathtt{u}$ in the memory. Next when K_2 is needed, we compute K_2 from $K_2 \cdot \mathtt{u}$ and replace $K_2 \cdot \mathtt{u}$ with K_2 in the memory. Repeat this process.

Note that it is easy to compute K_2 from $K_2 \cdot \mathtt{u}$ since multiplication by $\mathtt{u}^{-1}$ can be computed with only one shift and one conditional XOR as shown in (2).

3.3 Comparison with XCBC

XCBC is obtained by replacing $K_2 \cdot x$ with K_3 in Fig. 2, where $K_3 \in \{0,1\}^n$ is a random string. In another way around, TMAC is obtained from XCBC by replacing K_3 with $K_2 \cdot x$. The size of keys is reduced from $(k+2n)$ bits to $(k+n)$ bits in this way.

4 Security of TMAC

4.1 Security Definitions

An adversary $\mathcal{A}$ is an algorithm with an oracle (or oracles). The oracle computes some function. Without loss of generality, adversaries are assumed to never ask a query outside the domain of the oracle, and to never repeat a query.

A block cipher is a function $E : \mathcal{K}_E \times \{0,1\}^n \to \{0,1\}^n$ where $\mathcal{K}_E$ is a finite set and each $E_K(\cdot) = E(K, \cdot)$ is a permutation on $\{0,1\}^n$. 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 $\mathrm{Perm}(n)$.

Note that $\{E_K(\cdot) \mid K \in \mathcal{K}_E\}$ should look like $\mathrm{Perm}(n)$. For an adversary $\mathcal{A}$, we define

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

The adversary $\mathcal{A}$ cannot distinguish $\{E_K(\cdot) \mid K \in \mathcal{K}_E\}$ from $\mathrm{Perm}(n)$ if $\mathtt{Adv}_E^{\mathsf{prp}}(\mathcal{A})$ is negligible.

Similarly, a MAC function family from $\{0,1\}^*$ to $\{0,1\}^n$ is a map $F : \mathcal{K}_F \times \{0,1\}^* \to \{0,1\}^n$ where $\mathcal{K}_F$ is a set with an associated distribution. We write $F_K(\cdot)$ for $F(K, \cdot)$. We say that $\mathcal{A}^{F_K(\cdot)}$ *forges* if $\mathcal{A}$ outputs $(x, F_K(x))$ where $\mathcal{A}$ never queried x to its oracle $F_K(\cdot)$. Then we define

$$\mathtt{Adv}_F^{\mathsf{mac}}(\mathcal{A}) \stackrel{\mathrm{def}}{=} \Pr(K \stackrel{R}{\leftarrow} \mathcal{K}_F : \mathcal{A}^{F_K(\cdot)} \text{ forges}) \ .$$

Let $\mathrm{Rand}(*, n)$ denote the set of all functions from $\{0,1\}^*$ to $\{0,1\}^n$. This set is given a probability measure by asserting that a random element R of $\mathrm{Rand}(*, n)$

associates to each string $x \in \{0,1\}^*$ a random string $R(x) \in \{0,1\}^n$. Then we define

$$\mathtt{Adv}_F^{\mathsf{viprf}}(\mathcal{A}) \stackrel{\mathrm{def}}{=} \left|\Pr(K \stackrel{R}{\leftarrow} \mathcal{K}_F : \mathcal{A}^{F_K(\cdot)} = 1) - \Pr(R \stackrel{R}{\leftarrow} \mathrm{Rand}(*, n) : \mathcal{A}^{R(\cdot)} = 1)\right| .$$

Also we write

$$\mathtt{Adv}_E^{\mathsf{prp}}(t, q) \stackrel{\mathrm{def}}{=} \max_{\mathcal{A}} \{\mathtt{Adv}_E^{\mathsf{prp}}(\mathcal{A})\} ,$$

where the maximum is over all adversaries who run in time at most t and make at most q queries. Further we write

$$\mathtt{Adv}_F^{\mathsf{mac}}(t, q, \mu) \stackrel{\mathrm{def}}{=} \max_{\mathcal{A}} \{\mathtt{Adv}_F^{\mathsf{mac}}(\mathcal{A})\} \text{ and } \mathtt{Adv}_F^{\mathsf{viprf}}(t, q, \mu) \stackrel{\mathrm{def}}{=} \max_{\mathcal{A}} \left\{\mathtt{Adv}_F^{\mathsf{viprf}}(\mathcal{A})\right\} ,$$

where the maximum is over all adversaries who run in time at most t, make at most q queries, each of which is at most μ-bits.

4.2 Theorem Statements

We give the following information-theoretic bound on the security of TMAC. A proof of this lemma is given in the next section.

We idealize a block cipher by a random permutation drawn from $\mathrm{Perm}(n)$.

Lemma 4.1. *Let $\mathcal{A}$ be an adversary which asks at most q queries, each of which is at most nm-bits. Assume $m \le 2^n/4$. Then*

$$\Big|\Pr(P_1 \stackrel{R}{\leftarrow} \mathrm{Perm}(n); K_2 \stackrel{R}{\leftarrow} \{0,1\}^n : \mathcal{A}^{\mathrm{TMAC}_{P_1,K_2}(\cdot)} = 1) - \Pr(R \stackrel{R}{\leftarrow} \mathrm{Rand}(*, n) : \mathcal{A}^{R(\cdot)} = 1)\Big| \le \frac{(3m^2+1)q^2}{2^n} .$$

From the above theorem, it is standard to pass to the complexity-theoretic result. (For example, see [3, Section 3.2].) Then we have the following corollary.

Corollary 4.1. *Let $E : \mathcal{K}_E \times \{0,1\}^n \to \{0,1\}^n$ be the underlying block cipher used in TMAC. Then*

$$\mathtt{Adv}_{\mathrm{TMAC}}^{\mathsf{viprf}}(t, q, nm) \le \frac{(3m^2+1)q^2}{2^n} + \mathtt{Adv}_E^{\mathsf{prp}}(t', q') ,$$

where $t' = t + O(mq)$ and $q' = mq$.

The security of MAC is also derived in the usual way. (For example, see [3, Proposition 2.7].) Then we have the following theorem.

Theorem 4.1. *Let $E : \mathcal{K}_E \times \{0,1\}^n \to \{0,1\}^n$ be the underlying block cipher used in TMAC. Then*

$$\mathtt{Adv}_{\mathrm{TMAC}}^{\mathsf{mac}}(t, q, nm) \le \frac{(3m^2+1)q^2+1}{2^n} + \mathtt{Adv}_E^{\mathsf{prp}}(t', q') ,$$

where $t' = t + O(mq)$ and $q' = mq$.

Algorithm $\mathrm{FCBC}_{E_{K_1},E_{K_2},E_{K_3}}(M)$
if $M \in (\{0,1\}^n)^+$
  then $K \leftarrow K_2$, and $P \leftarrow M$
  else $K \leftarrow K_3$, and $P \leftarrow M||10^i$, where $i \leftarrow n - 1 - |M| \bmod n$
Let $P = P_1||P_2||\cdots||P_m$, where $|P_1| = |P_2| = \cdots = |P_m| = n$
$C_0 \leftarrow 0^n$
for $i \leftarrow 1$ **to** $m-1$ **do**
  $C_i \leftarrow E_{K_1}(P_i \oplus C_{i-1})$
return $E_K(P_m \oplus C_{m-1})$

Fig. 3. Definition of FCBC.

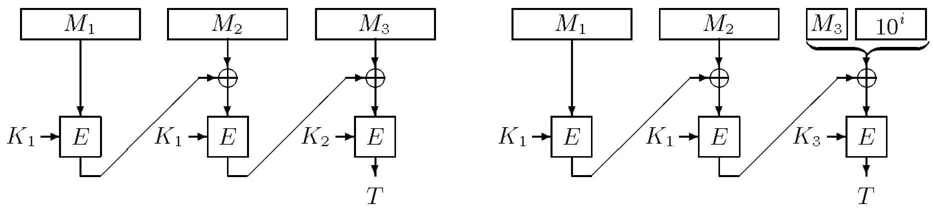

Fig. 4. Illustration of FCBC.

4.3 Proof of Lemma 4.1

For a random permutation P and a random n-bit string K, let

$$Q_1(x) \stackrel{\mathrm{def}}{=} P(K \oplus x),$$
$$Q_2(x) \stackrel{\mathrm{def}}{=} P((K \cdot \mathtt{u}) \oplus x).$$

We first show that $P(\cdot), Q_1(\cdot), Q_2(\cdot)$ are indistinguishable from three independent random permutations $P_1(\cdot), P_2(\cdot), P_3(\cdot)$.

Lemma 4.2. *Let $\mathcal{A}$ be an adversary which asks at most q queries. Then*

$$\Big|\Pr(P \stackrel{R}{\leftarrow} \mathrm{Perm}(n); K \stackrel{R}{\leftarrow} \{0,1\}^n : \mathcal{A}^{P(\cdot),P(K\oplus\cdot),P((K\cdot\mathtt{u})\oplus\cdot)} = 1) - \Pr(P_1, P_2, P_3 \stackrel{R}{\leftarrow} \mathrm{Perm}(n) : \mathcal{A}^{P_1(\cdot),P_2(\cdot),P_3(\cdot)} = 1)\Big| \le \frac{q^2}{2^n} .$$

A proof is given in the appendix.

Next we recall FCBC which appeared in the analysis of XCBC [5]. FCBC is a function taking three keys $K_1, K_2, K_3 \in \mathcal{K}_E$ and a message $M \in \{0,1\}^*$, and returning a string in $\{0,1\}^n$, where E is the underlying block cipher. The function is defined in Fig. 3 and illustrated in Fig. 4. Black and Rogaway showed the following result for FCBC [5].

Proposition 4.1 (Black and Rogaway [5]). *Let $\mathcal{A}$ be an adversary which asks at most q queries, each of which is at most nm-bits. Assume $m \le 2^n/4$.*

Then

$$\Big|\Pr(P_1, P_2, P_3 \stackrel{R}{\leftarrow} \mathrm{Perm}(n) : \mathcal{A}^{\mathrm{FCBC}_{P_1,P_2,P_3}(\cdot)} = 1) - \Pr(R \stackrel{R}{\leftarrow} \mathrm{Rand}(*, n) : \mathcal{A}^{R(\cdot)} = 1)\Big| \le \frac{(2m^2+1)q^2}{2^n} \quad .$$

We finally give a proof of Lemma 4.1.

Proof (of Lemma 4.1). By the triangle inequality,

$$\Big|\Pr(P_1 \stackrel{R}{\leftarrow} \mathrm{Perm}(n); K_2 \stackrel{R}{\leftarrow} \{0,1\}^n : \mathcal{A}^{\mathrm{TMAC}_{P_1,K_2}(\cdot)} = 1) - \Pr(R \stackrel{R}{\leftarrow} \mathrm{Rand}(*, n) : \mathcal{A}^{R(\cdot)} = 1)\Big| \tag{3}$$

is at most

$$\Big|\Pr(P_1, P_2, P_3 \stackrel{R}{\leftarrow} \mathrm{Perm}(n) : \mathcal{A}^{\mathrm{FCBC}_{P_1,P_2,P_3}(\cdot)} = 1) - \Pr(R \stackrel{R}{\leftarrow} \mathrm{Rand}(*, n) : \mathcal{A}^{R(\cdot)} = 1)\Big| \tag{4}$$

$$+\Big|\Pr(P_1 \stackrel{R}{\leftarrow} \mathrm{Perm}(n); K_2 \stackrel{R}{\leftarrow} \{0,1\}^n : \mathcal{A}^{\mathrm{TMAC}_{P_1,K_2}(\cdot)} = 1) - \Pr(P_1, P_2, P_3 \stackrel{R}{\leftarrow} \mathrm{Perm}(n) : \mathcal{A}^{\mathrm{FCBC}_{P_1,P_2,P_3}(\cdot)} = 1)\Big| \quad . \tag{5}$$

Proposition 4.1 [5] gives us an upper bound on (4). We next bound (5). (5) is at most

$$\Big|\Pr(P \stackrel{R}{\leftarrow} \mathrm{Perm}(n); K \stackrel{R}{\leftarrow} \{0,1\}^n : \mathcal{A}^{P(\cdot),P(K\oplus\cdot),P((K\cdot\mathtt{u})\oplus\cdot)} = 1) - \Pr(P_1, P_2, P_3 \stackrel{R}{\leftarrow} \mathrm{Perm}(n) : \mathcal{A}^{P_1(\cdot),P_2(\cdot),P_3(\cdot)} = 1)\Big| \tag{6}$$

since any adversary which does well in the setting (5) could be converted to one which does well in the setting (6), where we assume that $\mathcal{A}$ in (6) makes at most mq total queries to her oracles. By applying Lemma 4.2, (5) is bounded by $m^2q^2/2^n$. Therefore (3) is at most

$$\frac{(2m^2+1)q^2}{2^n} + \frac{m^2q^2}{2^n} = \frac{(3m^2+1)q^2}{2^n} \quad .$$

□

5 Discussion

5.1 Summary of Properties

We give a summary of properties of TMAC in Table 2.

Table 2. Summary of Properties.

Security Function	**Message Authentication Code.** More generally, TMAC is a variable input length ($\{0,1\}^*$) pseudorandom function (VIPRF) with fixed output length ($\{0,1\}^n$).
Error Propagation	**Not applicable.**
Synchronization	**Not applicable.**
Parallelizability	**Sequential.**
Keying Material	**Two keys.** One block cipher key and one n-bit key, where n is the block length of the block cipher.
Ctr/IV/Nonce Requirements	**None.** No counter/IV/nonce is used.
Memory Requirements	**Very modest.** Memory requirements for the CBC MAC plus n bit for key.
Pre-processing Capability	**Limited.** Key-setup of the underlying block cipher and $K_2 \cdot \mathbf{u}$ can be pre-computed. Additional pre-computation is not possible.
Message-Length Requirements	**Arbitrarily length.** Any bit string $M \in \{0,1\}^*$ can be computed, including the empty string. The length of the string need not be known in advance.
Ciphertext Expansion	**Not applicable.**

5.2 Advantages

Short Key. TMAC requires only $(k+n)$-bit keys while XCBC uses $(k+2n)$-bit keys.

Provable Security. We proved that TMAC is a variable input length ($\{0,1\}^*$) pseudorandom function (VIPRF) with fixed output length ($\{0,1\}^n$) by assuming that the underlying block cipher is a pseudorandom permutation.

Efficiency. TMAC uses $\max\{1, \lceil |M|/n \rceil\}$ block cipher calls. The overhead beyond block cipher calls is almost negligible.

Arbitrarily Message Length. Any bit string $M \in \{0,1\}^*$ can be computed, including the empty string. The length of the string need not be known in advance.

No Re-Keying. Whereas some competing schemes (e.g., in [1,4,11]) would require invoking E with two or three different keys, TMAC requires only one key as XCBC. Therefore any key-setup costs are minimized. This enhances efficiency in both software and hardware.

No decryption. As for any CBC MAC variant, TMAC does not use decryption of the block cipher.

Backwards Compatibility. TMAC with $K_2 = 0^n$ is backwards compatible with the CBC MAC.

Simplicity. Because TMAC is simple, it is easily implemented in both software and hardware.

5.3 Limitations

We note the following limitations. They apply to any CBC MAC variants and therefore none of them is specific to TMAC.

Sequential Block Cipher Calls. The CBC MAC and its variants, including TMAC, are not parallelizable.

Limited Pre-processing Capability. Key-setup of the underlying block cipher and $K_2 \cdot \mathtt{u}$ can be pre-computed. Additional pre-computation is not possible without knowing the message.

5.4 Design Rationale

TMAC is generalized to TMAC family as follows. Let C_1 and C_2 in $\{0,1\}^n$ be two distinct constants. Let $H : \mathcal{K}_H \times \{0,1\}^n \to \{0,1\}^n$ be a (universal) hash function as follows, where $\mathcal{K}_H$ is the set of possible keys of H.

$$\text{For any } y \in \{0,1\}^n,\ \#\{K \in \mathcal{K}_H \mid H_K(C_1) = y\} = \tfrac{\#\mathcal{K}_H}{2^n}, \tag{7}$$

$$\text{For any } y \in \{0,1\}^n,\ \#\{K \in \mathcal{K}_H \mid H_K(C_2) = y\} = \tfrac{\#\mathcal{K}_H}{2^n}, \text{ and} \tag{8}$$

$$\text{For any } y \in \{0,1\}^n,\ \#\{K \in \mathcal{K}_H \mid H_K(C_1) \oplus H_K(C_2) = y\} = \tfrac{\#\mathcal{K}_H}{2^n}. \tag{9}$$

By using C_1, C_2 and H, TMAC family is specified in Fig. 5 and Fig. 6.

Algorithm $\mathrm{TMAC}_{E_{K_1}, H_{K_2}, C_1, C_2}(M)$
if $M \in (\{0,1\}^n)^+$
  then $K \leftarrow H_{K_2}(C_1)$ and $P \leftarrow M$
  else $K \leftarrow H_{K_2}(C_2)$ and $P \leftarrow M \| 10^i$, where $i \leftarrow n - 1 - |M| \bmod n$
Let $P = P_1 \| P_2 \| \cdots \| P_m$, where $|P_1| = |P_2| = \cdots = |P_m| = n$
$C_0 \leftarrow 0^n$
for $i \leftarrow 1$ **to** $m - 1$ **do**
  $C_i \leftarrow E_{K_1}(P_i \oplus C_{i-1})$
return $E_{K_1}(P_m \oplus C_{m-1} \oplus K)$

Fig. 5. Definition of TMAC family.

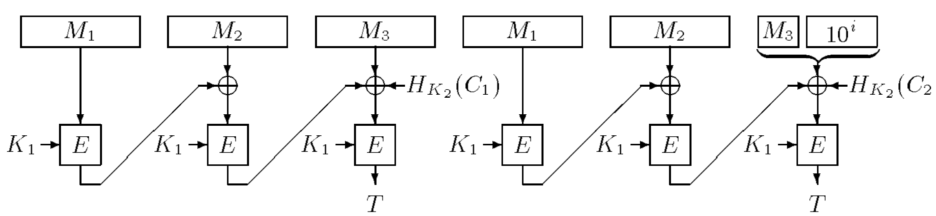

Fig. 6. Illustration of TMAC family.

We can then prove the security of TMAC family similarly to Lemma 4.1, Corollary 4.1 and Theorem 4.1. The security bounds are exactly the same as Lemma 4.1, Corollary 4.1 and Theorem 4.1.

Our choice for TMAC corresponds to $\mathcal{K}_H = \{0,1\}^n$, $H_K(x) = K \cdot x$, $C_1 = \mathtt{u}$, and $C_2 = 1$, or equivalently $H_K(C_1) = K \cdot \mathtt{u}$ and $H_K(C_2) = K$, where $K \in \{0,1\}^n$. It is easy to see that our choice meets the conditions (7),(8), and (9). Below, we list reasons of this choice.

- We adopted multiplications in GF(2^n) since it is simple, easy to understand, and easy to implement for appropriate constants.
- We adopted 1 and $\mathtt{u}$ as constants, since multiplications by 1 and $\mathtt{u}$ are both easy to implement efficiently as we have seen in (1).
- The reason why we let $H_K(C_1) = K \cdot \mathtt{u}$ and $H_K(C_2) = K$ (not $H_K(C_1) = K$ and $H_K(C_2) = K \cdot \mathtt{u}$) is that, most of the case we have $M \notin (\{0,1\}^n)^+$, rather than $M \in (\{0,1\}^n)^+$, if the message is a random string. Therefore we have chosen computationally easier way for the case $M \notin (\{0,1\}^n)^+$.

6 Test Vectors

Test vectors will be provided in a separate paper.

7 Performance Estimation

Similarly to XCBC, TMAC uses $\lceil |M|/n \rceil$ block cipher invocations for any non-empty message M. (The empty string is an exception; it requires one block cipher invocation.) Overhead beyond block cipher calls is almost negligible.

The size of secret keys is n bits smaller than XCBC. The cost for this short key is to use $K_2 \cdot \mathtt{u}$. It is computed with only one shift and one conditional XOR.

8 Intellectual Property Statement

The authors of this paper have no patent related to TMAC. As far as we know, TMAC is covered by no patents.

References

1. ANSI X9.19. American national standard — Financial institution retail message authentication. ASC X9 Secretariat — American Bankers Association, 1986.
2. M. Bellare, R. Guérin, and P. Rogaway. XOR MACs: New methods for message authentication using finite pseudorandom functions. *Advances in Cryptology — CRYPTO '95, LNCS 963,* pp. 15–28, Springer-Verlag, 1995.
3. 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.
4. 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.

5. 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.
6. J. Black and P. Rogaway. Comments to NIST concerning AES modes of operations: A suggestion for handling arbitrary-length messages with the CBC MAC. *Second Modes of Operation Workshop.* Available at `http://www.cs.ucdavis.edu/~rogaway/`.
7. 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.
8. 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.
9. V. Gligor, and P. Donescu. Fast encryption and authentication: XCBC encryption and XECB authentication modes. *Fast Software Encryption, FSE 2001, to appear in LNCS,* pp. 97–111, 2001. Full version is available at `http://csrc.nist.gov/encryption/modes/proposedmodes/`.
10. 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.
11. É. Jaulmes, A. Joux, and F. Valette. On the security of randomized CBC-MAC beyond the birthday paradox limit: A new construction. *Fast Software Encryption, FSE 2002, to appear in LNCS,* pp. 231–245, 2002. Full version is available at `http://eprint.iacr.org/2001/074/`.
12. E. Petrank and C. Rackoff. CBC MAC for real-time data sources. *J.Cryptology,* vol. 13, no. 3, pp. 315–338, Springer-Verlag, 2000.

A Proof of Lemma 4.2

Let $\{A^{(1)},\dots,A^{(q)}\}$ be a set of n-bit strings, that is, $A^{(i)} \in \{0,1\}^n$ for $1 \le \forall i \le q$. We say $\{A^{(1)},\dots,A^{(q)}\}$ *are distinct* as shorthand for $A^{(i)} \ne A^{(j)}$ for $1 \le \forall i < \forall j \le q$.

Before proving Lemma 4.2, we need the following lemma.

Lemma A.1. *Let q, q_1, q_2, q_3 be positive integers such that $q = q_1 + q_2 + q_3$. Let*

$$x_1^{(1)},\dots,x_1^{(q_1)},x_2^{(1)},\dots,x_2^{(q_2)},x_3^{(1)},\dots,x_3^{(q_3)}$$

be fixed n-bit strings such that $\{x_1^{(1)},\dots,x_1^{(q_1)}\}$ are distinct, $\{x_2^{(1)},\dots,x_2^{(q_2)}\}$ are distinct, and $\{x_3^{(1)},\dots,x_3^{(q_3)}\}$ are distinct. Similarly, Let

$$y_1^{(1)},\dots,y_1^{(q_1)},y_2^{(1)},\dots,y_2^{(q_2)},y_3^{(1)},\dots,y_3^{(q_3)}$$

be fixed n-bit strings such that $\{y_1^{(1)},\dots,y_1^{(q_1)},y_2^{(1)},\dots,y_2^{(q_2)},y_3^{(1)},\dots,y_3^{(q_3)}\}$ are distinct. Let $P \in \mathrm{Perm}(n)$ and $K \in \{0,1\}^n$. Then the number of (P,K) which satisfies

$$\begin{cases} P(x_1^{(i)}) = y_1^{(i)} & \text{for } 1 \le \forall i \le q_1, \\ P(K \oplus x_2^{(i)}) = y_2^{(i)} & \text{for } 1 \le \forall i \le q_2, \text{ and} \\ P((K \cdot \mathtt{u}) \oplus x_3^{(i)}) = y_3^{(i)} & \text{for } 1 \le \forall i \le q_3 \end{cases} \quad (10)$$

is at least $(2^n - (q_1+q_2+q_3))! \cdot (2^n - (q_1q_2 + q_1q_3 + q_2q_3))$.

We note that $(2^n - (q_1+q_2+q_3))! \cdot (2^n - (q_1q_2+q_1q_3+q_2q_3)) \geq (2^n - q)! \cdot \left(2^n - \frac{q^2}{2}\right)$ since $q_1q_2 + q_1q_3 + q_2q_3 = \frac{q^2 - q_1^2 - q_2^2 - q_3^2}{2}$.

Proof (of Lemma A.1). We first count the number of K.

Number of K. First, for any fixed i and j such that $1 \leq i \leq q_1$ and $1 \leq j \leq q_2$, we have exactly one K such that $x_1^{(i)} = K \oplus x_2^{(j)}$. Since there are q_1q_2 choice of (i,j), we have

$$\#\{K \mid x_1^{(i)} = K \oplus x_2^{(j)} \text{ for } 1 \leq \exists i \leq q_1 \text{ and } 1 \leq \exists j \leq q_2\} \leq q_1q_2 \ . \tag{11}$$

Next, for any fixed i and j such that $1 \leq i \leq q_1$ and $1 \leq j \leq q_3$, we have exactly one K such that $x_1^{(i)} = (K \cdot \mathtt{u}) \oplus x_3^{(j)}$. Since there are q_1q_3 choice of (i,j), we have

$$\#\{K \mid x_1^{(i)} = (K \cdot \mathtt{u}) \oplus x_3^{(j)} \text{ for } 1 \leq \exists i \leq q_1 \text{ and } 1 \leq \exists j \leq q_3\} \leq q_1q_3 \ . \tag{12}$$

Next, for any fixed i and j such that $1 \leq i \leq q_2$ and $1 \leq j \leq q_3$, we have exactly one K such that $K \oplus x_2^{(i)} = (K \cdot \mathtt{u}) \oplus x_3^{(j)}$. Since there are q_1q_3 choice of (i,j), we have

$$\#\{K \mid K \oplus x_2^{(i)} = (K \cdot \mathtt{u}) \oplus x_3^{(j)} \text{ for } 1 \leq \exists i \leq q_2 \text{ and } 1 \leq \exists j \leq q_3\} \leq q_2q_3 \ . \tag{13}$$

Then from (11), (12) and (13), we have at least $2^n - (q_1q_2 + q_1q_3 + q_2q_3)$ choice of $K \in \{0,1\}^n$ which satisfies the following three conditions:

$$\begin{cases} x_1^{(i)} \neq K \oplus x_2^{(j)} & \text{for } 1 \leq \forall i \leq q_1 \text{ and } 1 \leq \forall j \leq q_2, \\ x_1^{(i)} \neq (K \cdot \mathtt{u}) \oplus x_3^{(j)} & \text{for } 1 \leq \forall i \leq q_1 \text{ and } 1 \leq \forall j \leq q_3, \text{ and} \\ K \oplus x_2^{(i)} \neq (K \cdot \mathtt{u}) \oplus x_3^{(j)} & \text{for } 1 \leq \forall i \leq q_2 \text{ and } 1 \leq \forall j \leq q_3. \end{cases}$$

We now fix any K which satisfies these three conditions.

Number of P. Now K is fixed in such a way that

$$\{x_1^{(1)}, \ldots, x_1^{(q_1)}, K \oplus x_2^{(1)}, \ldots, K \oplus x_2^{(q_2)}, (K \cdot \mathtt{u}) \oplus x_3^{(1)}, \ldots, (K \cdot \mathtt{u}) \oplus x_3^{(q_3)}\}$$

(which are inputs to P) are distinct. Also, the corresponding outputs

$$\{y_1^{(1)}, \ldots, y_1^{(q_1)}, y_2^{(1)}, \ldots, y_2^{(q_2)}, y_3^{(1)}, \ldots, y_3^{(q_3)}\}$$

are distinct. In other words, for P, the above $q_1 + q_2 + q_3$ input-output pairs are determined. The remaining $2^n - (q_1 + q_2 + q_3)$ input-output pairs are undetermined. Therefore we have $(2^n - (q_1+q_2+q_3))!$ possible choice of P for any such fixed K.

Completing the Proof. To summarize, we have:

- at least $2^n - (q_1q_2 + q_1q_3 + q_2q_3)$ choice of K, and
- $(2^n - (q_1 + q_2 + q_3))!$ choice of P when K is fixed.

This concludes the proof of the lemma. $\square$

We now prove Lemma 4.2.

Proof (of Lemma 4.2). Let $\mathcal{O}_1, \mathcal{O}_2, \mathcal{O}_3$ be either $P(\cdot)$, $P(K \oplus \cdot)$, $P((K \cdot \mathtt{u}) \oplus \cdot)$ or $P_1(\cdot)$, $P_2(\cdot)$, $P_3(\cdot)$. The adversary $\mathcal{A}$ has oracle access to $\mathcal{O}_1$, $\mathcal{O}_2$ and $\mathcal{O}_3$.

There are three types of queries $\mathcal{A}$ can make: either $(1, x)$ which denotes the query "what is $\mathcal{O}_1(x)$?," $(2, x)$ which denotes the query "what is $\mathcal{O}_2(x)$?," or $(3, x)$ which denotes the query "what is $\mathcal{O}_3(x)$?." For the i-th query $\mathcal{A}$ makes to $\mathcal{O}_j$, define the query-answer pair $(x_j^{(i)}, y_j^{(i)}) \in \{0,1\}^n \times \{0,1\}^n$, where $\mathcal{A}$'s query was $(j, x_j^{(i)})$ and the answer it got was $y_j^{(i)}$.

Without loss of generality, we assume that $\mathcal{A}$ makes q_1 queries to $\mathcal{O}_1(x)$, q_2 queries to $\mathcal{O}_2(x)$, and q_3 queries to $\mathcal{O}_3(x)$, where $q_1 + q_2 + q_3 = q$. Further, we assume that $\mathcal{A}$ is deterministic (otherwise we consider arbitrarily fixed random tape).

Define view v of $\mathcal{A}$ as

$$\begin{aligned} v = \langle &(x_1^{(1)}, y_1^{(1)}), \ldots, (x_1^{(q_1)}, y_1^{(q_1)}), \\ &(x_2^{(1)}, y_2^{(1)}), \ldots, (x_2^{(q_2)}, y_2^{(q_2)}), \\ &(x_3^{(1)}, y_3^{(1)}), \ldots, (x_3^{(q_3)}, y_3^{(q_3)}) \rangle \ . \end{aligned}$$

We say that v is a *possible view* if the following three conditions are satisfied:

$$\begin{cases} \{y_1^{(1)}, \ldots, y_1^{(q_1)}\} \text{ are distinct,} \\ \{y_2^{(1)}, \ldots, y_2^{(q_2)}\} \text{ are distinct, and} \\ \{y_3^{(1)}, \ldots, y_3^{(q_3)}\} \text{ are distinct.} \end{cases}$$

We note that since $\mathcal{A}$ never repeats a query, we have

$$\begin{cases} \{x_1^{(1)}, \ldots, x_1^{(q_1)}\} \text{ are distinct,} \\ \{x_2^{(1)}, \ldots, x_2^{(q_2)}\} \text{ are distinct, and} \\ \{x_3^{(1)}, \ldots, x_3^{(q_3)}\} \text{ are distinct.} \end{cases}$$

We also note that since $\mathcal{A}$ is deterministic, the i-th query $\mathcal{A}$ makes is fully determined by the first $i-1$ query-answer pairs. Then the number of all possible view N_{all} is $N_{all} = \frac{(2^n)!}{(2^n - q_1)!} \cdot \frac{(2^n)!}{(2^n - q_2)!} \cdot \frac{(2^n)!}{(2^n - q_3)!}$. Similarly, the final output of $\mathcal{A}$ (0 or 1) depends only on v. Hence denote by $\mathcal{C}_{\mathcal{A}}(v)$ the final output of $\mathcal{A}$ as a function of v.

Let $\boldsymbol{v}_{one}$ be a set of all possible view v such that $\mathcal{A}$ outputs 1. That is, $\boldsymbol{v}_{one} \stackrel{\text{def}}{=} \{v \mid \mathcal{C}_{\mathcal{A}}(v) = 1\}$. We let $N_{one} \stackrel{\text{def}}{=} \#\boldsymbol{v}_{one}$. Also, let $\boldsymbol{v}_{good}$ be a set of all possible view v such that $\{y_1^{(1)}, \ldots, y_1^{(q_1)}, y_2^{(1)}, \ldots, y_2^{(q_2)}, y_3^{(1)}, \ldots, y_3^{(q_3)}\}$ are distinct. We let $N_{good} \stackrel{\text{def}}{=} \#\boldsymbol{v}_{good}$, then $N_{good} = \frac{(2^n)!}{(2^n - (q_1+q_2+q_3))!}$. Therefore we have

$$\#\{v \mid v \in (\boldsymbol{v}_{one} \cap \boldsymbol{v}_{good})\} \geq N_{one} - (N_{all} - N_{good}) \ . \tag{14}$$

Evaluation of p_{rand}. We first evaluate

$$p_{rand} \stackrel{\text{def}}{=} \Pr(P_1, P_2, P_3 \stackrel{R}{\leftarrow} \mathrm{Perm}(n) : \mathcal{A}^{P_1(\cdot),P_2(\cdot),P_3(\cdot)} = 1) \ .$$

We have

$$p_{rand} = \frac{\#\{(P_1, P_2, P_3) \mid \mathcal{A}^{P_1(\cdot),P_2(\cdot),P_3(\cdot)} = 1\}}{\{(2^n)!\}^3} \ .$$

For each $v \in \boldsymbol{v}_{one}$, the number of (P_1, P_2, P_3) such that

$$\begin{cases} P_1(x_1^{(i)}) = y_1^{(i)} & \text{for } 1 \le \forall i \le q_1, \\ P_2(x_2^{(i)}) = y_2^{(i)} & \text{for } 1 \le \forall i \le q_2, \text{ and} \\ P_3(x_3^{(i)}) = y_3^{(i)} & \text{for } 1 \le \forall i \le q_3 \end{cases} \tag{15}$$

is exactly $(2^n - q_1)! \cdot (2^n - q_2)! \cdot (2^n - q_3)!$. Therefore, we have

$$\begin{aligned} p_{rand} &= \sum_{v \in \boldsymbol{v}_{one}} \frac{\#\{(P_1, P_2, P_3) \mid (P_1, P_2, P_3) \text{ satisfying } (15)\}}{\{(2^n)!\}^3} \\ &= N_{one} \cdot \frac{(2^n - q_1)! \cdot (2^n - q_2)! \cdot (2^n - q_3)!}{\{(2^n)!\}^3} \\ &= \frac{N_{one}}{N_{all}} \ . \end{aligned}$$

Evaluation of p_{real}. We next evaluate

$$p_{real} \stackrel{\text{def}}{=} \Pr(P \stackrel{R}{\leftarrow} \mathrm{Perm}(n); K \stackrel{R}{\leftarrow} \{0,1\}^n : \mathcal{A}^{P(\cdot),P(K\oplus\cdot),P((K\cdot\mathrm{u})\oplus\cdot)} = 1) \ .$$

We have

$$p_{real} = \frac{\#\{(P, K) \mid \mathcal{A}^{P(\cdot),P(K\oplus\cdot),P((K\cdot\mathrm{u})\oplus\cdot)} = 1\}}{(2^n)! \cdot 2^n} \ .$$

Then from Lemma A.1, we have

$$\begin{aligned} p_{real} &\ge \sum_{v \in (\boldsymbol{v}_{one} \cap \boldsymbol{v}_{good})} \frac{\#\,\{(P, K) \mid (P, K) \text{ satisfying } (10)\}}{(2^n)! \cdot 2^n} \\ &\ge \sum_{v \in (\boldsymbol{v}_{one} \cap \boldsymbol{v}_{good})} \frac{(2^n - q)!}{(2^n)!} \cdot \left(1 - \frac{q^2}{2 \cdot 2^n}\right) \ . \end{aligned}$$

From (14) we have

$$\begin{aligned} p_{real} &\ge (N_{one} - N_{all} + N_{good}) \cdot \frac{(2^n - q)!}{(2^n)!} \cdot \left(1 - \frac{q^2}{2 \cdot 2^n}\right) \\ &= \left(\frac{N_{one}}{N_{all}} - 1 + \frac{N_{good}}{N_{all}}\right) \cdot N_{all} \cdot \frac{(2^n - q)!}{(2^n)!} \cdot \left(1 - \frac{q^2}{2 \cdot 2^n}\right) \ . \end{aligned} \tag{16}$$

Completing the Proof. Now we have

$$\frac{N_{good}}{N_{all}} \geq 1 - \frac{q(q-1)}{2 \cdot 2^n} \quad \text{and} \quad N_{all} \cdot \frac{(2^n - q)!}{(2^n)!} \geq 1 \ .$$

The first inequality follows since

$$\begin{aligned}
\frac{N_{good}}{N_{all}} &= \frac{\prod_{1 \leq i \leq q-1} \left(1 - \frac{i}{2^n}\right)}{\prod_{1 \leq i \leq q_1 - 1} \left(1 - \frac{i}{2^n}\right) \cdot \prod_{1 \leq i \leq q_2 - 1} \left(1 - \frac{i}{2^n}\right) \cdot \prod_{1 \leq i \leq q_3 - 1} \left(1 - \frac{i}{2^n}\right)} \\
&\geq \prod_{1 \leq i \leq q-1} \left(1 - \frac{i}{2^n}\right) \\
&\geq 1 - \frac{1 + 2 + \cdots + (q-1)}{2^n} \ .
\end{aligned}$$

Then from (16) we have

$$\begin{aligned}
p_{real} &\geq \left(p_{rand} - \frac{q(q-1)}{2 \cdot 2^n}\right) \cdot \left(1 - \frac{q^2}{2 \cdot 2^n}\right) \\
&\geq p_{rand} - \frac{q^2}{2^n} \ . \qquad (17)
\end{aligned}$$

Applying the same argument to $1 - p_{real}$ and $1 - p_{rand}$ yields that

$$1 - p_{real} \geq 1 - p_{rand} - \frac{q^2}{2^n} \ . \qquad (18)$$

Finally, (17) and (18) give $|p_{real} - p_{rand}| \leq \frac{q^2}{2^n}$. $\square$