# Fast and Secure CBC-Type MAC Algorithms

Mridul Nandi

National Institute of Standards and Technology
mridul.nandi@gmail.com

**Abstract.** The CBC-MAC or cipher block chaining message authentication code, is a well-known method to generate message authentication codes. Unfortunately, it is not forgery-secure over an arbitrary domain. There are several secure variants of CBC-MAC, among which OMAC is a widely-used candidate. To authenticate an $s$-block message, OMAC costs $(s+1)$ block cipher encryptions (one of these is a zero block encryption), and only one block cipher key is used. In this paper, we propose two secure and efficient variants of CBC-MAC: namely, GCBC1 and GCBC2. Our constructions cost only $s$ block cipher encryptions to authenticate an $s$-block message, for all $s \geq 2$. Moreover, GCBC2 needs only one block cipher encryption for almost all single block messages, and for all other single block messages, it costs two block cipher encryptions. We have also defined a class of generalized CBC-MAC constructions, and proved a sufficient condition for prf-security. In particular, we have provided an unified prf-security analysis of CBC-type constructions, e.g., XCBC, TMAC and our proposals GCBC1 and GCBC2.

**Keywords.** CBC-MAC; OMAC; padding rule; prf-security

## 1 Introduction

In cryptography, a common trend is to design fast and secure algorithms. In this paper, we propose two fast and secure block cipher-based message authentication codes. A message authentication code, or MAC, is useful in those applications where data integrity and authenticity are essential. In terms of security, we want a MAC to be a pseudorandom function, or prf, which means that it is computationally indistinguishable from an ideal random function. Prf-security is a strong security notion, and it also guarantees that the MAC is unforgeable. In this paper, we use the words "secure" and "prf-secure" synonymously. Several secure and fast authentication algorithms are already known. We first broadly classify them into three main categories, based on their underlying building blocks.

Hash-Mac: These are based on hash functions. HMAC [1] is a widely used candidate in this class that has been standardized by the National Institute of Standards and Technology (NIST). The other efficient, popular candidates include the cascaded-PRF [2], sandwich-MAC [21], and KMDP [14].

Universal Hash based Mac: These MACs use universal hash functions and small domain pseudorandom functions. In software, these are very fast for long messages [11, 19]. These generally require field multiplications, key expansions, and invocations of a smaller domain pseudorandom function which may be overhead for short messages. Some popular examples of universal hash based authentications are UMAC [8] and poly1305 MAC [7]. In [17], a 4-round version of AES [12] is used to obtain a universal hash function that eventually produces a very fast MAC computation for long messages (close to two times faster than OMAC). But, this is also slower than OMAC, due to the overhead required for processing short messages.

Block Cipher based Mac: In this paper, we study this category in more detail. These MACs are usually based on several invocations of a block cipher, either in a feedback mode (cipher block chaining or CBC-MAC) or in a parallel mode (e.g., PMAC [9] or XOR-MAC [3]). *A block cipher is a permutation $e_K : \{0,1\}^n \rightarrow \{0,1\}^n$, for each key $K$ chosen from the key space $\{0,1\}^k$, where $n$ (the block size) and $k$ (the key size) are positive integers. We fix these parameters throughout the paper.* Intuitively, a block cipher is called pseudorandom permutation or prp-secure if the keyed block cipher family is computationally indistinguishable from an ideal random permutation. CBC-MAC (cipher block chaining message authentication [4]) is the first construction in this category. Given a message $M = (m_1 \| \cdots \| m_\ell) \in (\{0,1\}^n)^\ell$, the CBC-MAC of the message $M$ based on $e_K$ is computed as follows:

$$\mathsf{CBC\text{-}MAC}_K(M) = e_K(e_K(\cdots e_K(m_1) \oplus m_2 \cdots) \oplus m_\ell).$$

However, CBC-MAC is not secure for variable length messages due to the length extension attack. Many different modifications of it have been proposed so far, among which OMAC [15] or one-key CBC-MAC[1] is efficient (requires one extra zero block encryption compared to CBC-MAC computation), as well as requiring only one key. Another simple modification, called XCBC-MAC [10] or XCBC, is faster in software, but it needs three keys, which may not be suitable in many applications. These keys may be derived from one key at the cost of few block cipher invocations, which causes slower performance for short messages. The TMAC requires only two keys and it is as efficient as CBC-MAC. If the output of zero block encryption of OMAC is stored as a key (to save one block cipher encryption) then eventually, OMAC and TMAC look almost identical.

## 1.1 Our Proposals GCBC1 and GCBC2

Let $M = (m_1 \| \cdots \| m_\ell) \in (\{0,1\}^n)^\ell$ with $\ell \geq 2$. The GCBC1-MAC of the message $M$ based on $e_K$ is computed as follows:

$$\mathsf{GCBC1}_K(M) = e_K\big(e_K(\cdots e_K(e_K(m_1) \oplus m_2) \cdots\big)^{\lll 1} \oplus m_\ell\big).$$

[1] it is also known as CMAC [13] as recommended by the NIST

For all other messages $M$, we pad $10^d$ (for smallest choice of $d$) to make sure that the message size is multiple of $n$ and has at least two blocks. Let $M' = (m_1\|\cdots\|m_\ell) \in (\{0,1\}^n)^\ell$, $\ell \geq 2$, be the padded message. In this case we compute the tag as follows,

$$\mathsf{GCBC1}_K(M) = e_K\big(e_K(\cdots e_K(e_K(m_1) \oplus m_2)\cdots)^{\ll 2} \oplus m_\ell\big).$$

In other words, we apply one or two left shifts to the last intermediate chaining value[2] of $\mathsf{CBC\text{-}MAC}$ before xor-ing the last message block of the padded message. This small tweak eventually helps to avoid length extension attack. Moreover we prove that it is prf-secure (see Section 5). Handling the last intermediate input in two different manners depending on the size of the last message block, is very common in MACs and it is used, e.g., in $\mathsf{XCBC}$, $\mathsf{OMAC}$, $\mathsf{TMAC}$, etc.

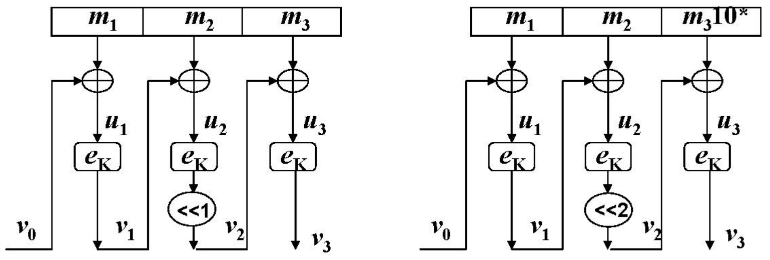

**Fig. 1.** $\mathsf{GCBC1}$, Generalized CBC, which uses a left shift variation operation, an underlying iterative function $e_K$ (block cipher) and a simple padding rule.

**Theorem** (*Security Bound of GCBC1*)

$$\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{GCBC1}}(q,\sigma) \leq \frac{5(\sigma+q)(\sigma+q-1)}{2^{n-4}} + \mathbf{Adv}^{\mathrm{prp}}_{e}(\sigma).$$

Our second construction, called $\mathsf{GCBC2}$, authenticates almost all single block messages by using one block cipher encryption. It considers several cases, depending on the message size. For, $x \in \{0,1\}^n$, define $\overline{x} = x$, and if $|x| \leq n-1$ then define $\overline{x} = x10^{n-1-|x|}$. We define $\delta = 2$ if the message size is multiple of $n$ otherwise, we set $\delta = 1$.

1. Let $|M| \leq n-4$, then $\mathsf{GCBC2}_K(M) = e_K(\overline{M})$.

[2] Note that for a single block padded message, the last intermediate chaining value is nothing but the message block and so, left shift operations on it is a predictable operation. So we can not prevent length extension attack for a single block padded message. Hence, we need to make sure that padded message has at least two blocks.

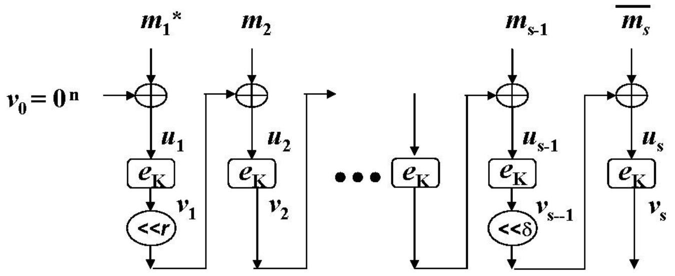

**Fig. 2.** GCBC2 for more than one message block. Let $M = m_1\|m_2\|\cdots\|m_s$, $m_1 = m_1'\|m_1''$, $|m_1''| = 3$. We denote $m_1^* = \overline{m}_1$ if $m_1'' = 000$, o.w., $m_1^* = m_1$. The nonnegative integer $r$, $0 \le r \le 4$, denotes the amounts of left shift applied to the output of the first encryption (the value of $r$ can be determined in the definition of GCBC2).

2. Let $n-3 \le |M| \le n$, then write $M = m_1 := m_1'm_1''$ where $|m_1'| = n-3$ and $|m_1''| \le 3$. We define

$$\mathsf{GCBC2}_K(M) = e_K(e_K(m_1'\|011) \oplus \overline{m_1''}).$$

3. Let $M = m_1\|m_2$, $m_1 = m_1'm_1''$, $|m_1'| = n-3, |m_1''| = 3$ and $|m_2| \le n$. We define

$$\mathsf{GCBC2}_K(M) = \begin{cases} e_K(e_K(m_1)^{\lll \delta+1} \oplus \overline{m_2}) & \text{if } m_1'' \neq 000 \\ e_K(e_K(m_1'\|100)^{\lll \delta-1} \oplus \overline{m_2}) & \text{if } m_1'' = 000 \end{cases}$$

4. Let $M = m_1'm_1''\|m_2\|m_3\|\cdots\|m_s$ where $|m_1'| = n-3, |m_1''| = 3$ and $|m_2| = \cdots = m_{s-1} = n, |m_s| \le n$. We define

$$\begin{aligned}\mathsf{GCBC2}_K(M) &= e_K\big(\big(e_K(\cdots e_K(e_K(\ e_K(m_1'\|m_1'')^{\lll 4} \oplus m_2) \oplus m_3)\cdots)\big)^{\lll \delta} \oplus \overline{m_s}\big) \text{ if } m_1'' \neq 000 \\ &= e_K\big(\big(e_K(\cdots e_K(e_K(\ e_K(m_1'\|100)^{\lll 5} \oplus m_2) \oplus m_3)\cdots)\big)^{\lll \delta} \oplus \overline{m_s}\big) \text{ if } m_1'' = 000\end{aligned}$$

**Theorem** (*Security Bound of GCBC2*, see Section 5)

$$\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{GCBC2}}(q,\sigma) \le \frac{33(\sigma+q)(\sigma+q-1)}{2^n} + \mathbf{Adv}^{\mathrm{prp}}_e(\sigma).$$

OMAC vs Our MACs: For all messages having block sizes at least two, both GCBC1 and GCBC2 have similar performance to CBC-MAC, whereas OMAC costs one extra zero block encryption. Zero block encryption can be computed off line, in which case we have to store the output as a key and hence it has similar performance to TMAC (which has two-key and similar to CBC performance). For single block messages (of size less than or equal to $n$), both OMAC and GCBC1 need two block cipher encryptions (in case of OMAC, one of the encryption is zero block encryption). The GCBC2 costs exactly one block cipher encryption like CBC-MAC for almost all single block messages (except the messages of size in between $n-3$ and $n$, and in which case it costs two encryptions). Thus, GCBC2 is a good choice whenever the short messages are authenticated frequently. In some applications, we might know before hand that message sizes are at least $(n+1)$-bits. In these applications, GCBC1 would be a good choice due to its simplicity and performance (same as CBC-MAC, but secure for arbitrary length messages). In table 1, a comparison of software performances are given.

| Name of MAC | microsec (1-15 bytes) | microsec (16 bytes) | microsec (17 - 32 bytes) | # BC for $s$-block | Total Keysize |
|---|---|---|---|---|---|
| XCBC [10] | 43.7 | 43.7 | 78.46 | $s$ | $k+2n$ |
| TMAC [16] | 43.98 | 44.05 | 78.80 | $s$ | $k+n$ |
| OMAC [15] | 78.72 | 78.80 | 113.80 | $s+1$ | $k$ |
| GCBC1 | 77.9 | 77.92 | 77.95 | $s$ | $k$ |
| GCBC2 | 43.58 | 78.26 | 78.37 | $s$ | $k$ |

**Table 1.** It provides a performance comparison of known CBC-type MACs along with our proposals. The software speed is computed (in the platform Intel(R) Pentium(R) 4 CPU 3.60 GHz, 1GB RAM) with AES-128 as the underlying block cipher. Here, # BC denotes the number of invocations of the block cipher $e : \{0,1\}^k \times \{0,1\}^n \rightarrow \{0,1\}^n$ that is used to authenticate an $s$-block message. Time is computed by taking the average over several executions.

A GENERALIZED CBC-MAC: In this paper, we have also provided a general class of CBC-type constructions, called gcbc, which includes almost all popularly known CBC-type constructions, e.g., XCBC, TMAC, OMAC, and our proposals. We have also given a sufficient condition for prf-secure gcbc constructions and we have shown that almost all known constructions such as XCBC, TMAC, and our proposals satisfy the sufficient condition. So, we have provided that, how an unified way of security analysis of CBC-type MACs can be provided.

**Organization of the paper**. We first provide basic definitions, and notations in Section 2. In Section 3, we propose a generalized CBC-type message authentication algorithms and also show that most of the CBC-type constructions belong to the class. The security analysis has been made by using decorrelation technique. The detailed security analysis of the generalized CBC-type constructions is given in Section 4. Finally, in Section 5, we specify two fast and secure constructions, called GCBC1 and GCBC2, from the generalized class.

## 2 Preliminaries

### 2.1 Definitions and Notations

Given any set $S$, $S^+ = \cup_{i=1}^{\infty} S^i$, and $S^* = \cup_{i=0}^{\infty} S^i = S^+ \cup \{\lambda\}$, where $\lambda$ is the empty string. For example, $\{0,1\}^+$ is the set of all non-empty finite bit-sequences.

Let $|x| = i$ for any $x \in \{0,1\}^i$. Any $X \in S^+$ can be written as $X = (x_1, \cdots, x_i)$ for some $i \geq 1$ and $x_1, \cdots, x_i \in S$. A tuple $Y = (y_1, \cdots, y_j) \in S^*$ is a prefix of $X$ if $j \leq i$ and $y_1 = x_1, \cdots, y_j = x_j$. Trivially, $\lambda$ is a prefix of any $X$, and is called a trivial prefix. Any other prefixes are called non-trivial prefixes. Let $x = x_1 x_2 \cdots x_n \in \{0,1\}^n$, $x_i \in \{0,1\}$. Then for any two integers $i \leq j$, the set $\{i, i+1, \cdots, j\}$ is denoted as $[i..j]$, and $x_i x_{i+1} \cdots x_j$ is denoted as $x[i..j]$ whenever $1 \leq i \leq j \leq n$. If $i > j$, $x[i..j]$ is nothing but $\lambda$. Let $x[i]$ represent the $i$th bit of $x$. The notation $x^{\ll t}$ (or $x^{\gg t}$) is denoted for $t$-bit left shift (or right shift, respectively) of an $n$-bit string $x$. The set $\{0,1\}^n$ is sometimes identified as $\mathrm{GF}(2^n)$ by fixing a primitive polynomial $z^n + c_1 z^{n-1} + \cdots + c_{n-1} z + c_n$, where $c_i \in \{0,1\}$. Let $0^n = \mathbf{0}$ (the additive identity), $0^{n-1}1 = \mathbf{1}$ (the multiplicative identity) and $\alpha = 0^{n-2}10 \in \mathrm{GF}(2^n)$ (known as a primitive element). For any element $x \in \{0,1\}^n$, the field multiplication with $\alpha$ is denoted as $\alpha \cdot x$, and it can be computed as $x^{\ll 1}$ if $x[1] = 0$; otherwise, it is $x^{\ll 1} \oplus c$, where $c = c_1 c_2 \cdots c_n$. We use $x \overset{*}{\leftarrow} S$ to mean that $x$ is chosen uniformly from the set $S$, and it is independently chosen from all other previously described distributions.

**Definition 1.** (ideal random function and ideal random permutation)
*$\rho$ is said to be an ideal random function from $\mathcal{M}$ to $\{0,1\}^n$ if, for any distinct $m_1, \cdots, m_q \in \mathcal{M}$, $(\rho(m_1), \cdots, \rho(m_q))$ is uniformly distributed over $(\{0,1\}^n)^q$ for any $q > 0$. In other words, for any $q$ elements $y_1, \cdots, y_q \in \{0,1\}^n$,*

$$\Pr[\rho(m_1) = y_1, \cdots, \rho(m_q) = y_q] = \frac{1}{2^{nq}}.$$

*Similarly, $\tau$ is said to be an ideal random permutation on $\{0,1\}^n$ if, for any distinct $x_1, \cdots, x_q \in \{0,1\}^n$ and distinct $y_1, \cdots, y_q \in \{0,1\}^n$,*

$$\Pr[\tau(m_1) = y_1, \cdots, \tau(m_q) = y_q] = \frac{1}{2^n(2^n-1)\cdots(2^n-q+1)}.$$

When $\mathcal{M}$ is a finite set, there is an alternative way to view an ideal random function. Let $\mathrm{Func}(\mathcal{M}, \{0,1\}^n)$ denote the set of all functions from $\mathcal{M}$ to $\{0,1\}^n$. The set of all functions from $\{0,1\}^n$ to $\{0,1\}^n$ is denoted as $\mathrm{Func}(n,n)$. An ideal random function from $\mathcal{M}$ to $\{0,1\}^n$ is defined as a function chosen at random (uniformly) from $\mathrm{Func}(\mathcal{M}, \{0,1\}^n)$ (this is not possible when $\mathcal{M}$ is an infinite set). This definition is an equivalent to the previous definition, as one can show that

$$\Pr[\rho(m_1) = y_1, \cdots, \rho(m_q) = y_q : \rho \overset{*}{\leftarrow} \mathrm{Func}(\mathcal{M}, \{0,1\}^n)] = \frac{1}{2^{nq}}$$

for any distinct $m_1, \cdots, m_q \in \mathcal{M}$, and any $y_1, \cdots, y_q \in \{0,1\}^n$. We sometimes write an ideal random function as a keyed function family $\mathsf{rand}_\rho$, where $\mathsf{rand}_\rho(x) = \rho(x)$ and $\rho \in \mathrm{Func}(\mathcal{M}, \{0,1\}^n)$. A block cipher is a function $e : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$, such that for any key $K \in \{0,1\}^k$, $e_K := e(K, \cdot)$ is a permutation on $\{0,1\}^n$. In this paper, we fix $n$, and any element $x \in \{0,1\}^i$ is called a block if $1 \leq i \leq n$. A block is called complete if $i = n$, otherwise, it is called incomplete. For any $x \in \{0,1\}^*$, we denote $\lceil \frac{|x|}{n} \rceil$ as $||x||$ (called the number of blocks of $x$). Let $A$ be an oracle adversary. We say $A$ is a $q$-adversary if it makes at most $q$ queries, and we say it is a $(q, \sigma)$-adversary if it makes at most $q$ queries, and the total number of blocks in all queries is at most $\sigma$. For simplicity, we assume that a $q$-adversary makes exactly $q$ queries, as there is no loss when making some extra dummy queries. We say $q$ is the number of input queries, whereas $\sigma$ as the number of block-queries.

**Definition 2.** (pseudorandom function) *Let $F_{K'}$ be a keyed function family, where $K' \in \mathcal{K}'$ and $F_{K'} : \mathcal{M} \to \{0,1\}^n$ for a message space $\mathcal{M}$. For any probabilistic oracle adversary $A$, we define the prf-advantage of it over the function family $F$ as*

$$\mathbf{Adv}_F^{\mathrm{prf}}(A) = |\Pr[A^{F_{K'}} = 1 : K' \xleftarrow{*} \mathcal{K}'] - \Pr[A^{\rho} = 1]|,$$

*where $\rho$ is an ideal random function from $\mathcal{M}$ to $\{0,1\}^n$, and the probabilities are computed over the internal randomness of $A$, the uniform distribution of $K'$ and randomness of the output behavior of $\rho$. When $\mathcal{M} = \{0,1\}^n$, we can equivalently compute the prf-advantage as*

$$\mathbf{Adv}_F^{\mathrm{prf}}(A) = |\Pr[A^{F_{K'}} = 1 : K' \xleftarrow{*} \mathcal{K}] - \Pr[A^{\mathsf{rand}_\rho} = 1 : \rho \xleftarrow{*} \mathrm{Func}(n,n)]|.$$

*The prf-advantage of $F$ is defined as $\mathbf{Adv}_F^{\mathrm{prf}}(q, \sigma) = \max_A \mathbf{Adv}_F^{\mathrm{prf}}(A)$, where the maximum is taken over all $(q, \sigma)$-adversaries $A$. When $\mathcal{M} = \{0,1\}^n$, we have $\sigma = q$ and hence, we also write $\mathbf{Adv}_F^{\mathrm{prf}}(\sigma)$. We say that a function family $F$ is a $(q, \sigma, \epsilon)$-prf (or $(\sigma, \epsilon)$-prf, for the case that $\mathcal{M} = \{0,1\}^n$) if $\mathbf{Adv}_F^{\mathrm{prf}}(q, \sigma) \le \epsilon$.*

**Definition 3.** (pseudorandom permutation) *The prp-advantage of an oracle adversary $A$ over a block cipher $e : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ is computed as*

$$\mathbf{Adv}_e^{\mathrm{prp}}(A) = |\Pr[A^{e(K,\cdot)} = 1 : K \xleftarrow{*} \{0,1\}^k] - \Pr[A^{\tau} = 1]|,$$

*where the probabilities are computed over internal randomness of $A$, uniform distribution of $K$ and randomness of the ideal random permutation $\tau$ on $\{0,1\}^n$. The prp-advantage of the block cipher $e$ is defined as $\mathbf{Adv}_e^{\mathrm{prp}}(q) = \max_A \mathbf{Adv}_e^{\mathrm{prp}}(A)$, where the maximum is taken over all $q$-adversaries $A$.*

**Lemma 1.** (switching lemma) *For any function family $F = (F_K)_{K \in \mathcal{K}}$, $F_K : \{0,1\}^n \to \{0,1\}^n$, we have*

$$\mathbf{Adv}_F^{\mathrm{prf}}(\sigma) \le \mathbf{Adv}_F^{\mathrm{prp}}(\sigma) + \frac{\sigma(\sigma - 1)}{2^{n+1}}.$$

The proof of the switching lemma can be found in [6], for example.

## 3 Generalized CBC-MAC Class

### 3.1 Building blocks

Every MAC for a message space $\mathcal{M}$ has two main components, namely a randomized key-generation algorithm and a tag-generation algorithm which may be deterministic or probabilistic. The key-generation algorithm returns a key $(K, L)$ at random from it's key space $\mathcal{K} \times \{0,1\}^\ell$. In this paper, we consider deterministic tag-generation algorithms which consist of three main building blocks as described below.

Padding Rule. A padding rule $\mathsf{pad} : \mathcal{M} \to ([0..t] \times \{0,1\}^n)^+$ which ensures that the padded message is in a desired form. The non-negative integer $t$ is said to be the variation number. Given a message $m$, the padded message $\mathsf{pad}(m) = X$ will be written as $((\delta_1, x_1), \cdots, (\delta_s, x_s))$ for some positive integer $s$, $x_i \in \{0,1\}^n$ and $\delta_i \in [0..t]$, where $1 \le i \le s$. We denote the set of all possible $\delta_1$ values as

$$\Delta_{\mathsf{pad}} = \{\delta_1 : \exists m \in \mathcal{M},\ \mathsf{pad}(m) = ((\delta_1, x_1), \cdots)\}.$$

**The role of the $x_i$'s is similar to that of CBC message block, whereas the $\delta_i$ values are used to tweak the intermediate outputs of the block cipher by using a variation operation** (for example, applying $\delta_i$ amounts of left shift, etc.). A padding rule $\mathsf{pad}$ is said to be *prefix-free* if, for any $m \neq m'$, $\mathsf{pad}(m)$ is not a prefix of $\mathsf{pad}(m')$.

Iterative Function. An underlying iterative function $f : \{0,1\}^n \to \{0,1\}^n$ which is determined via a key $K \in \mathcal{K}$. A block cipher $e_K$ or an ideal random function $\mathsf{rand}_\rho$ for $\rho \xleftarrow{*} \mathrm{Func}(n, n)$ (note, $\mathsf{rand}_\rho(x) = \rho(x)$) are different examples of iterative functions.

Variation Operation. A $t$-variate variation operation is a function $h : [0..t] \times \{0,1\}^n \to \{0,1\}^n$, such that $h(0, x) = x$. These operations are defined to be very efficient functions. The variation operation may use a key $L$ (called the *auxiliary key*) and the underlying iterative function $f$ as a subroutine. Thus, the variation operation may be determined by the key $K$ of $f$. In this case, we say the operation is a secret variation operation. If the operation does not use $f$ and any auxiliary key $L$, then $h$ is a publicly computable function, and we say that it is a public variation operation. A simple example of a $t$-variate public variation operation is

$$h(i, x) = x^{\lll i} \text{ for all } 0 \le i \le t,\ x \in \{0,1\}^n.$$

It is called a type-I secret variation operation if it only depends on the auxiliary key and not on the underlying iterative function $f$. All other secret variation operations are called type-II. In this paper, we consider public or type-I secret variation operations when we study the security analysis of the generalized CBC constructions. But, we also see some secure constructions, such as OMAC or CMAC, that use type-II secret variation operations.

## 3.2 Definition of a Generalized CBC-MAC

We define a class of generalized CBC message authentication algorithms (denoted as $\mathcal{C}_{\mathsf{gcbc}}$). Any authentication algorithm from this class for a message space $\mathcal{M}$ has two main functionalities, namely a randomized key-generation algorithm (or Key-Gen) with a key space $\mathcal{K} \times \{0,1\}^\ell$ and a deterministic tag-generation

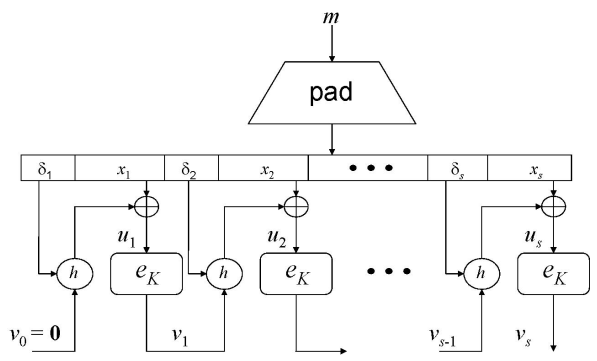

**Fig. 3.** Generalized CBC which uses variation operation $h$, an underlying iterative function $e_K$ (block cipher) and a padding rule $\mathsf{pad}$.

algorithm $\mathsf{gcbc}^{f,h,\mathsf{pad}}$ (defined below). Thus, we need to specify the key space $\mathcal{K} \times \{0,1\}^\ell$, a message space $\mathcal{M}$, the underlying iterative operation $f$, a $t$-variate variation operation $h$ and a padding rule with variation number $t$. The only randomness of the generalized CBC comes from the key $(K, L) \in \mathcal{K} \times \{0,1\}^\ell$ and hence, we denote the authentication algorithm as $\mathsf{gcbc}_{K,L}$ whenever all the above are clear from the context.

1. $\mathsf{Key\text{-}Gen}$ : $(K, L) \overset{*}{\leftarrow} \mathcal{K} \times \{0,1\}^\ell$, where $\mathcal{K} \times \{0,1\}^\ell$ is the key space. Key-generation is parameterized by the key space only.
2. $\mathsf{gcbc}^{f,h,\mathsf{pad}}$: The tag-generation algorithm for a message space $\mathcal{M}$ uses three subroutines viz.,
   - a padding rule $\mathsf{pad} : \mathcal{M} \to ([0..t] \times \{0,1\}^n)^+$, with a variation number $t \geq 0$,
   - a $t$-variate variation operation (public or secret) $h : [0..t] \times \{0,1\}^n \to \{0,1\}^n$, and
   - an underlying iterative function $f : \{0,1\}^n \to \{0,1\}^n$.

   These subroutines, except for the padding rule $\mathsf{pad}$, are specified by the key $(K, L)$ (the output of $\mathsf{Key\text{-}Gen}$), where $K$ is the key for the underlying iterative function $f$, and $L$ is the auxiliary key that is used for the secret variation operation $h$. For the public variation operation, $\ell = 0$. For any message $m$, we define $\mathsf{gcbc}^{f,h,\mathsf{pad}}(m) = v_s$, where $v_s$ is computed as follows (also described in Algorithm 1 and illustrated in Figure 3):

$$v_0 = 0^n, \;\; u_i = h(\delta_i, v_{i-1}) \oplus x_i, \quad v_i = f(u_i), 1 \leq i \leq s \tag{1}$$

   where $\mathsf{pad}(m) = ((\delta_1, x_1), \cdots, (\delta_s, x_s))$, $\delta_i \in [0..t], x_i \in \{0,1\}^n$.

*Remark 1.* Usually, $x_1, \cdots, x_s$ in Algorithm 1 are all different message blocks and $\delta_1, \cdots, \delta_s$ correspond to tweaking of chaining values. If $\delta_i = 0$ then there is no change in the $i^{\text{th}}$ intermediate input (same as CBC chaining for intermediate inputs). For non-zero values of $\delta_i$, we might do left shift operations on intermediate chaining value before xor-ing with the message block. The variation function $h(\delta, \cdot)$ corresponds to these tweaking operations.

*Remark 2.* An efficiency of the tag-generation algorithm $\mathsf{gcbc}^{f,h,\mathsf{pad}}$ depends on the number of invocations of $f$, as the underlying iterative function is the most costly operation (we also desire a strong security notion from it, such as it being a pseudorandom function). Note that the number of invocations of $f$ is at least $s$, and it may be more if we use the type-II secret variation operation. To keep it small, we should carefully design a padding rule so that the value of $s$ is as small as possible. The padding rule and the variation operations (except the computation of $f$, which may be used in $h$) are usually very cheap and hence, we mostly focus on the number of invocations of $f$ when we compare the performance of different constructions.

**Algorithm 1** Generalized Cipher Block Chaining Message Authentication

```
Require:
    key.       $K \| L \in \mathcal{K} \times \{0,1\}^{\ell}$. \\ an output of the key generation algorithm Key-Gen,
                                  \\ which is used in the functions $f$ and $h$
    function.  $f : \{0,1\}^n \to \{0,1\}^n$,
               $h : [0..t] \times \{0,1\}^n \to \{0,1\}^n$,
               $\mathsf{pad} : \mathcal{M} \to ([0..t] \times \{0,1\}^n)^+$ \\$\mathcal{M} = \{0,1\}^*$.
    input.     $m \in \mathcal{M}$.
1: $X = \mathsf{pad}(m)$
2: divide $X$ as $((\delta_1, m_1) \cdots (\delta_s, x_s))$ where $x_i \in \{0,1\}^n, \delta_i \in [0..t], 1 \le i \le s$
3: $v_0 = 0^n$
4: for $j = 1$ to $s$ do
5:     $u_j = h(\delta_j, v_{j-1}) \oplus x_j$
6:     $v_j = f(u_j)$
7: end for
8: return $v_s$
```

### 3.3 Known CBC-type MACs are Generalized CBC-MAC

This class is indeed a generalized class, as it contains almost all CBC-type authentication algorithms. We describe more precisely how XCBC, TMAC and OMAC belong to the class. A common choice of the underlying iterative function is a block cipher $e_K$, $K \in \mathcal{K} = \{0,1\}^k$, and a common choice of padding rule pad is described below. Given a message $m = m_1 \| \cdots \| m_{s-1} \| m_s$ with $m_1, \cdots, m_{s-1} \in \{0,1\}^n$, $m_s \in \{0,1\}^r$, and $1 \le r \le n$, the padding rule (with variation number as two) is defined as

$$\mathsf{pad}(m) = (0, m_1), \cdots, (0, m_{s-1}), (\delta, \overline{m}_s),$$

where $\delta = 1$ if $r < n$; otherwise, we set $\delta = 2$. Also,

$$\overline{m}_s = \begin{cases} m_s \| 10^{n-1-|x|} & \text{if } |m_s| < n \\ m_s & \text{if } |m_s| = n \end{cases}$$

The value of $\ell$ and the 2-variate variation operations $h$ are described below for each construction. Recall that the key generation algorithm returns a key $(K, L) \overset{*}{\leftarrow} \{0,1\}^k \times \{0,1\}^\ell$.

XCBC: Let $\ell = 2n$, and write $L = L_1 \| L_2$, where $L_1, L_2 \in \{0,1\}^n$. Define $h(i,x) = x \oplus L_i \ \ \forall x \in \{0,1\}^n, i = 1, 2$.

TMAC: Let $\ell = n$, and define $h(i,x) = x \oplus (L \cdot \alpha^{i-1}) \ \ \forall x \in \{0,1\}^n$, for $i = 1, 2$, where $\alpha$ is a primitive element and $\alpha^0 = \mathbf{1}$ (multiplicative identity).

OMAC: Let $\ell = 0$, and define $h(0,x) = x$ and $h(i,x) = x \oplus (e_K(\mathbf{0}) \cdot \alpha^i) \ \ \forall x \in \{0,1\}^n, i = 1, 2$.

*Remark 3.* OMAC is an example where the variation operation is a type-II secret. In the case of the other two examples, the variation operations are type-I secret variation operations. In the next section, we propose two different padding rules and two public variation operations. The use of public variation operations helps to keep the key size as low as possible. We also see that these public variation operations are efficiently computable. We should be careful in the security analysis when we choose a public variation operation, since the security analysis is not straightforward.

## 4 Security Analysis

### 4.1 Decorrelation Technique

Vaudeney's Decorrelation Theorem (Lemma 22 of [20][3]) is used in the security analysis. Based on our notations, we state the following version of the Decorrelation Theorem.

**Theorem 1. (Decorrelation Theorem)**
*Let $q$ and $\sigma$ be two fixed integers, and let $F_{K'} : \mathcal{M} \to \{0,1\}^n$ be a family of functions indexed by key $K'$ that is chosen uniformly from the key space $\mathcal{K}'$. Suppose that the following holds for a positive real number $\epsilon$;*

> $\Pr[F_{K'}(x_1) = y_1, \cdots, F_{K'}(x_q) = y_q : K \overset{*}{\leftarrow} \mathcal{K}] \geq 2^{-nq}(1-\epsilon)$ *for any* $(y_1, \cdots, y_q) \in (\{0,1\}^n)^q$ *and distinct* $m_1, \cdots, m_q \in \mathcal{M}$, $\sum_{i=1}^q \|m_i\| \leq \sigma$.

*Then for any distinguisher $A$ which asks $q$ queries with $\sigma$ blocks present in all queries, $\mathbf{Adv}_F^{\mathrm{prf}}(A) \leq \epsilon$.*

[3] it was mentioned in [20] that the decorrelation theorem was freely adapted from Patarin's coefficient H-techniques [18]

**What does the Decorrelation Theorem mean?** The above condition means that the output behavior of the function family is very close to that of an ideal random function for any choices of distinct inputs (note, $2^{-nq} = \Pr[\rho(x_1) = y_1, \cdots, \rho(x_q) = y_q]$). Thus, the prf-advantage of a $(q, \sigma)$-adversary should be small, independent of how the adversary works. Note that the adversary (at the end of the query-responses) has a set of inputs and outputs, and he has to distinguish the function family from an ideal random function based on the query-responses. However, the values of $\epsilon$ can depend on $q$ and $\sigma$.

## 4.2 Security Analysis of Generalized CBC Algorithm

A variation operation can be public or secret. A variation operation is said to be *allowed* if either it is public, or it is a type-I secret (which does not use the underlying iterative as a subroutine and only uses an auxiliary key).

**Definition 4.** *Let $\Delta$ be a subset of $[0..t]$ and $\epsilon$ be a nonnegative real number. An allowed $t$-variate variation operation $h : [0..t] \times \{0,1\}^n \to \{0,1\}^n$ is said to be a $(\epsilon, \Delta)$-xor weak universal operation if, for any $0 \le \delta \ne \delta' \le t$, $c \in \{0,1\}^n$, and for all auxiliary key $L_0 \in \{0,1\}^\ell$, the following conditions are satisfied.*

*W1:* $\Pr[h_{L_0}(i, R) = c : R \xleftarrow{*} \{0,1\}^n] \le \epsilon$, *for any* $0 \le i \le t$,

*W2:* $\Pr[h_L(\delta, 0^n) \oplus h_L(\delta', 0^n) = c : L \xleftarrow{*} \{0,1\}^\ell] \le \epsilon$ *whenever* $\delta, \delta' \in \Delta$

*W3:* $\Pr[h_L(\delta, R) \oplus h_L(\delta', R) = c : (R, L) \xleftarrow{*} \{0,1\}^n \times \{0,1\}^\ell] \le \epsilon$.

Note that when $\Delta$ is a singleton set, then the condition W2 is vacuously true. We provide a sufficient condition for prf-secure of generalized CBC constructions.

### A sufficient condition for prf-secure gcbc

1. Let pad be a prefix-free padding rule with a variation number $t \ge 0$, $h$ be a $(\epsilon, \Delta_{\mathsf{pad}})$-xor weak universal, allowed variation operation, and
2. the underlying iterative function family $(f_K)_{K \in \mathcal{K}}$ is $(\sigma, \mu)$-prf.

The generalized CBC, based on the above such building blocks, is $(q, \sigma, \epsilon')$-prf, where $\epsilon' = \sigma'(\sigma' - 1)\epsilon + \mu$ and where $\sigma'$ denotes the total number of blocks in all queries after padding. (which have been queried by an adversary)[4]. As we are going to apply decorrelation theorem, we want to show the following probability for distinct $m_1, \cdots, m_q \in \mathcal{M}$ and distinct $y_1, \cdots, y_q \in \{0,1\}^n$,

$$p = \Pr_{\rho, L}[\mathsf{gcbc}_{\rho,L}(m_1) = y_1, \cdots, \mathsf{gcbc}_{\rho,L}(m_q) = y_q] \ge \frac{1 - \frac{\sigma'(\sigma'-1)\epsilon}{2}}{2^{nq}} \qquad (2)$$

where the probability is computed over $(\rho, L) \xleftarrow{*} \mathrm{Func}(n, n) \times \{0,1\}^\ell$. Then by applying decorrelation theorem and switching lemma, we know that $\mathsf{gcbc}_{K,L}$

[4] Later, we propose two padding rules where the number of block can only increase by at most one for each message and hence, $\sigma' \le \sigma + q$.

is $(q, \sigma, \epsilon')$-prf-secure. Now, it remains to show the above equation. We first introduce some notations as given below.

**Notations.** Let $\mathsf{pad}(m_i) = X_i = ((\delta_{i,1}, x_{i,1}) \cdots, (\delta_{i,\ell_i}, x_{i,s_i}))$, where $x_{i,1}, \cdots, x_{i,s_i} \in \{0,1\}^n$ and $\delta_{i,1}, \cdots, \delta_{i,s_i} \in [0..t]$. Let $X_{i,j} = ((\delta_{i,1}, x_{i,1}) \cdots, (\delta_{i,j}, x_{i,j}))$ for $0 \le j \le s_i$, where $X_{i,0} = \lambda$ for any $i$. For each $1 \le i \le q$, we have the following sequences of $u_i$'s and $v_i$'s values.

$$\begin{aligned} v_i\text{-sequence} &: 0^n \overset{(\delta_{i,1},x_{i,1})}{\rightarrow} v_{i,1} \overset{(\delta_{i,2},x_{i,2})}{\rightarrow} v_{i,2} \cdots \overset{(\delta_{i,s_i},x_{i,s_i})}{\rightarrow} v_{i,s_i} \\ u_i\text{-sequence} &: \lambda \overset{(\delta_{i,1},x_{i,1})}{\rightarrow} u_{i,1} \overset{(\delta_{i,2},x_{i,2})}{\rightarrow} u_{i,2} \cdots \overset{(\delta_{i,s_i},x_{i,s_i})}{\rightarrow} u_{i,s_i} \end{aligned}$$

Note that all these variables $u_{i,j}, v_{i,j}$ are random variables, whereas $\delta_{i,j}, x_{i,j}$ are fixed constants. Moreover, $u_{i,j} = h_L(\delta_{i,j}, v_{i,j-1}) \oplus x_{i,j}$ and $\mathsf{rand}_\rho(u_{i,j}) = v_{i,j}$. Thus, $u_{i,j}$ corresponds to the $i^{\text{th}}$ input of $\mathsf{rand}_\rho$ while computing $\mathsf{gcbc}_{\rho,L}(m_i)$ and $v_{i,j}$ is its corresponding output. The most common approach is to recognize all trivial collisions of inputs (which are because of the fact that some parts of two messages are identical). Then we try to prove that any other input collision has low probability (in other we would provide an upper bound on all other input collisions). Next, we would define admissible tuples which correspond to all intermediate inputs and outputs which do not have any non-trivial collisions.

**Lemma 2.** *If $X_{i,j} = X_{i',j'}$ then $u_{i,j} = u_{i',j'}$ and $v_{i,j} = v_{i',j'}$ with probability 1.*

**Definition 5.** *Let $L_0 \in \{0,1\}^\ell, V_{i,j} \in \{0,1\}^n$, $1 \le i \le q$, $1 \le j \le s_i$ for some fixed integers $s_1, \cdots, s_q$ and $q$. We say that a tuple $(L_0, V_{i,j})_{i,j}$ is* admissible *if*
*- $V_{i,j} = V_{i',j'}$ whenever $X_{i,j} = X_{i',j'}$,*
*- $V_{i,0} = 0^n, V_{i,s_i} = y_i$ for all $1 \le i \le q$ and*
*- $h_{L_0}(\delta_{i,j}, V_{i,j-1}) \oplus x_{i,j} \ne h_{L_0}(\delta_{i',j'}, V_{i',j'-1}) \oplus x_{i',j'}$ for all $i, j, i', j'$ such that $X_{i,j} \ne X_{i',j'}$.*

Let $\sigma_1$ be the maximum number of all pairs $(i,j)$ with distinct $X_{i,j}$'s. More precisely, $\sigma_1 = |\{X : X = X_{i,j} \text{ for some } i,j\}|$. Clearly, $\sigma_1 \le \sigma' = \sum_{i=1}^q ||X_i||$.

**Lemma 3.** *Given any admissible tuple $(L_0, V_{i,j})_{i,j}$,* $\Pr[L = L_0, v_{i,j} = V_{i,j}] = \frac{1}{2^{n(\sigma_1+q)}} \times \frac{1}{2^\ell}$.

**Proof.** Given any such admissible tuple, $u_{i,j} = u_{i',j'}$ if and only if $X_{i,j} = X_{i',j'}$ and all $u_{i,j} = U_{i,j} = h_{L_0}(\delta_{i,j}, V_{i,j-1}) \oplus x_{i,j}$ values are fixed. Thus,

$$\begin{aligned} \Pr[L = L_0, v_{i,j} = V_{i,j}] &= \Pr[L = L_0, \rho(U_{i,j}) = V_{i,j} \text{ for all } i,j] \\ &= \Pr[\rho(U_{i,j}) = V_{i,j} \text{ for all } i,j] \times \Pr[L = L_0] \\ &= \frac{1}{2^{n(\sigma_1+q)}} \times \frac{1}{2^\ell}. \end{aligned}$$

**Lemma 4.** *The number of admissible tuples is at least $2^{n\sigma_1+\ell}(1 - \frac{\epsilon\sigma_1(\sigma_1-1)}{2})$.*

**Proof.** We have $2^{n\sigma_1+\ell}$ tuples $(L_0,(V_{i,j})_{i,j})$ such that $V_{i,0}=0^n, V_{i,s_i}=y_i$ for all $1\le i\le q$ and $V_{i,j}=V_{i',j'}$ whenever $X_{i,j}=X_{i',j'}$. Now we need to find an estimate of the number of tuples among these such that $h_{L_0}(\delta_{i,j},V_{i,j-1})\oplus x_{i,j}\ne h_{L_0}(\delta_{i',j'},V_{i',j'-1})\oplus x_{i',j'}$ for all $i,j,i',j'$ such that $X_{i,j}\ne X_{i',j'}$. To do so, we count the complement. Suppose that for some $i,j,i',j'$ with $X_{i,j}\ne X_{i',j'}$, $h_{L_0}(\delta_{i,j},V_{i,j-1})\oplus x_{i,j}=h_{L_0}(\delta_{i',j'},V_{i',j'-1})\oplus x_{i',j'}$. The number of such tuples is at most $2^{n\sigma_1+\ell-1}$, since $h$ is a weakly $(\epsilon,\Delta_{\mathsf{pad}_1})$-xor universal variation operation. The total number of possible values of $i,j,i',j'$ such that $X_{i,j}\ne X_{i',j'}$ is $\binom{\sigma_1}{2}$. Subtracting all such non-admissible tuples, we see that there are at least $2^{n\sigma_1+\ell}(1-\frac{\epsilon\sigma_1(\sigma_1-1)}{2})$ admissible tuples. ∎

Combining the above two lemmas, we can prove the following theorem.

**Theorem 2.** *Let $h$ be a weakly $(\epsilon,\Delta_{\mathsf{pad}})$-xor universal operation and* pad *be a prefix-free padding rule. Suppose that the underlying iterative function is an ideal random function $(\mathit{rand}_\rho)_{\rho\in\mathrm{Func}(n,n)}$, and we denote the corresponding generalized CBC authentication algorithm as $\mathsf{gcbc}_{\rho,L}$. Let $\sigma'$ be the largest number of blocks after padding $q$ messages having at most $\sigma$ blocks in total. Then for any distinct $m_1,\cdots,m_q\in\mathcal{M}$ (message space), and any $y_1,\cdots,y_q\in\{0,1\}^n$,*

$$\Pr_{\rho,L}[\mathit{gcbc}_{\rho,L}(m_1)=y_1,\cdots,\mathit{gcbc}_{\rho,L}(m_q)=y_q]\ge\frac{1-\frac{\sigma'(\sigma'-1)\epsilon}{2}}{2^{nq}}.$$

**Theorem 3.** *Based on all notations defined so far, we have,*

$$\begin{aligned}\mathbf{Adv}^{\mathrm{prf}}_{\mathit{gcbc}}(q,\sigma)&\le\frac{\sigma'(\sigma'-1)\epsilon}{2}+\mathbf{Adv}^{\mathrm{prf}}_{f}(\sigma')\\&\le\frac{\sigma'(\sigma'-1)}{2}(\epsilon+\frac{1}{2^n})+\mathbf{Adv}^{\mathrm{prp}}_{f}(\sigma')\end{aligned}$$

**Theorem 4.** *The variation operations defined in XCBC and TMAC are weakly $(\frac{1}{2^n},\Delta_{\mathsf{pad}})$-xor universal operations and $\sigma'=\sigma$. So,*

1. $\mathbf{Adv}^{\mathrm{prf}}_{\mathit{XCBC}}(q,\sigma)\le\frac{\sigma(\sigma-1)}{2^n}+\mathbf{Adv}^{\mathrm{prp}}_{f}(\sigma)$.
2. $\mathbf{Adv}^{\mathrm{prf}}_{\mathit{TMAC}}(q,\sigma)\le\frac{\sigma(\sigma-1)}{2^n}+\mathbf{Adv}^{\mathrm{prp}}_{f}(\sigma)$.

## 5 Two New Efficient Generalized CBC-MAC: GCBC1 and GCBC2

In this section, we propose two secure, generalized CBC constructions, namely GCBC1 and GCBC2 both have message space $\{0,1\}^*$.

### 5.1 GCBC1

We first define a padding rule $\mathsf{pad}_1:\{0,1\}^{>n}\to([0..2]\times\{0,1\}^n)^+$. For any $m=m_1\cdots m_{s-1}m_s\in\{0,1\}^{>n}$, where $m_1,\cdots,m_{s-1}\in\{0,1\}^n$ and $m_s\in\{0,1\}^r$, $1\le r\le n$, we define the padded message as

$$\mathsf{pad}_1(m)=((0,m_1),\cdots,(0,m_{s-1}),(\delta,\overline{m}_s)),$$

**Algorithm 2** GCBC1

```
Require:
    key.       $K \overset{*}{\leftarrow} \{0,1\}^k$. \\ block cipher key
    function.  $e_K : \{0,1\}^n \rightarrow \{0,1\}^n$. \\ block cipher
    input.     $m \in \{0,1\}^*$.
1: divide $m$ as $(m_1, \cdots, m_{s-1}, m_s)$
   where $m_1, \cdots, m_{s-1} \in \{0,1\}^n, m_s \in \{0,1\}^r$, $1 \leq r \leq n$.
2: if $s = 1$ and $r = n$ then
3:     $m_2 = 10^{n-1}$, $s = 2, r = n - 1$.
4: else if $s = 1$ then
5:     $m_1 = \overline{m}_1, m_2 = \mathbf{0}$, $s = 2$.
6: end if
7: $v_0 = \mathbf{0}$
8: for $j = 1$ to $s - 1$ do
9:     $u_j = v_{j-1} \oplus x_j$
10:    $v_j = e_K(u_j)$
11: end for
12: if $r < n$ then
13:    $u_s = v_{s-1}^{\lll 1} \oplus x_s$
14: else
15:    $u_s = v_{s-1}^{\lll 2} \oplus x_s$
16: end if
17: $v_s = e_K(u_s)$
18: return $v_s$
```

where $\delta = 1$ if $r < n$; otherwise, $\delta = 2$. We extend the definition of the padding rule to the message space $\{0,1\}^*$ as follows. Let $m_1 \in \{0,1\}^r$, define

$$\mathsf{pad}_1(m_1) = \begin{cases} ((0, \overline{m}_1), (1, \mathbf{0})) & \text{if } r < n \\ ((0, m_1), (1, 10^{n-1})) & \text{if } r = n. \end{cases}$$

Thus, $s$-block messages have $s$-block padded messages for all $s \geq 2$, and one-block messages have two-block padded messages. It is also easy to observe that $\Delta_{\mathsf{pad}_1} = \{0\}$. Moreover, the padding rule is prefix-free.

**Proposition 1.** *The padding rule* $\mathsf{pad}_1$ *over the message space* $\{0,1\}^*$ *is a prefix-free padding rule.*

**Proof.** Suppose $m = m_1 \cdots m_{s-1} m_s$ and $m' = m'_1 \cdots m'_{s'-1} m'_{s'}$ where $s \leq s'$, $\mathsf{pad}_1(m)$ is a prefix of $\mathsf{pad}_1(m')$ and $m_1, m'_1, \cdots, m_{s-1}, m'_{s'-1} \in \{0,1\}^n, m_s \in \{0,1\}^r, m'_{s'} \in \{0,1\}^{r'}$, $1 \leq r, r' \leq n$.

Case $s \geq 2$: $\mathsf{pad}_1(m) = ((0, m_1), \cdots, (0, m_{s-1}), (\delta, \overline{m}_s))$ is a prefix of $\mathsf{pad}_1(m')$ $= ((0, m'_1), \cdots, (0, m'_{s'-1}), (\delta', \overline{m'}_{s'}))$. Since $\delta, \delta' \neq 0$, $s' = s$ and $\delta = \delta'$. Moreover, $m'_1 = m_1, \cdots, m_{s-1} = m'_{s-1}, \overline{m}_s = \overline{m'}_s$. Now, $\overline{m}_s = \overline{m'}_s$ and $\delta = \delta'$ implies that $m_s = m'_s$. Thus $m = m'$.

Case $s = 1, s' \geq 2$: $\mathsf{pad}_1(m) = ((0, x_1), (1, x_2))$ where $\mathsf{pad}_1(m') = ((0, m'_1), \cdots, (0, m'_{s'-1}), (\delta', \overline{m'}_{s'})$. By comparing $\delta$ values of the second pair, we can see that $s' = 2$, $r' < n$ and $\overline{m'}_2 = x_2$. But $x_2$ is either $\mathbf{0}$ or $10^{n-1}$ which can't be $\overline{m'}_2$ for any $m'_2 \in \{0,1\}^{r'}$, $1 \leq r' < n$. So this case does not arise.

Case $s = 1, s' = 1$: Obviously, if $\mathsf{pad}_1(m_1)$ is a prefix of $\mathsf{pad}_1(m'_1)$, then they should be equal. But, it is easy to see that they can be equal only when $m_1 = m'_1$ and hence, $m = m'$. ∎

Now we define a simple public variation operation $\mathsf{ls}$, which has two variations. For any $x \in \{0,1\}^n$ and $0 \leq \delta \leq 2$, $\mathsf{ls}(\delta, x) = x^{\lll \delta}$.

**Proposition 2.** *The operation* $\mathsf{ls}$ *with 2 variations is a public and weakly* $\frac{1}{2^{n-2}}$*-xor universal for* $\Delta = \Delta_{\mathsf{pad}_1} = \{0\}$*. The same operation* $\mathsf{ls}$ *with 5 variations is a public and weakly* $\frac{1}{2^{n-5}}$*-xor universal for* $\Delta = \Delta_{\mathsf{pad}_1} = \{0\}$*.*

**Proof.** By definition, $\mathsf{ls}(0, x) = x$. It is easy to see that $x^{\lll i} = c$ has at most 4 solutions of $x$ for any $i \leq 2$ and any constant $c \in \{0,1\}^n$. So, condition W1 holds. The condition W2 trivially holds, since $\Delta = \{0\}$. To see the condition W3, we first prove that the number of solutions of $x \oplus x^{\lll 1} = c$ is exactly one for any constant $c \in \{0,1\}^n$. In fact, the solution is $x[n] = c[n], x[n-1] = c[n-1] \oplus c[n-1], \cdots, x[1] = c[1] \oplus \cdots \oplus c[n]$. Similarly, one can see that the number of solutions of $x$ for the equation $x \oplus x^{\lll 2} = c$ is exactly one. Now we want to find the number of solutions of the equation $x^{\lll 1} \oplus x^{\lll 2} = c$. Let $y = x^{\lll^1}$. We have exactly one solution of $y$ and hence, there are exactly two solutions of $x$. Combining all these observations, we can see that condition W3 is true. Thus, $\mathsf{ls}$ is a weakly $\frac{1}{2^{n-2}}$-xor universal. The case for 5 variation operation can be proved similarly. ∎

We define the tag-generation algorithm of $\mathsf{GCBC1}$ as the generalized CBC algorithm (see Figure 1) $\mathsf{gcbc}^{e,\mathsf{ls},\mathsf{pad}_1}$ with a message space of $\{0,1\}^*$ and the padding rule $\mathsf{pad}_1$ (see Algorithm 2).

**Theorem 5.** (Security Bound of $\mathsf{GCBC1}$)

$$\mathbf{Adv}^{\mathrm{prf}}_{\mathit{GCBC1}}(q, \sigma) \leq \tfrac{5(\sigma+q)(\sigma+q-1)}{2^n} + \mathbf{Adv}^{\mathrm{prp}}_{e}(\sigma + q)$$

**Proof.** We apply the result of the above two propositions to the generalized CBC security bound (see Theorem 3 where $sigma' \leq \sigma + q$). ∎

## 5.2 GCBC2

We first define a padding rule $\mathsf{pad}_2$ for the message space $\{0,1\}^*$ with variation number 5. Let $m = m_1 \cdots m_{s-1} m_s \in \{0,1\}^*$, where $m_1, \cdots, m_{s-1} \in \{0,1\}^n$ and $m_s \in \{0,1\}^r$, $0 \leq r \leq n$ ($r = 0$ only when we have an empty message $m$). Let us denote $\delta = 1$ if $r < n$; otherwise, $\delta = 2$. If $|m| \geq n-3$ then denote $m_1 = m'_1 \| m''_1$, where $m'_1 \in \{0,1\}^{n-3}$ and $m''_1 \in \{0,1\}^*$. Define $\mathsf{pad}_2(m)$ to depend on $s$.

$$\text{Case } s = 1, \quad \mathsf{pad}_2(m_1) = \begin{cases} ((0, m'_1\|011), (0, \overline{m''}_1)) & \text{if } r \geq n-3 \\ (0, \overline{m}_1) & \text{if } r \leq n-4 \end{cases}$$

$$\text{Case } s = 2, \quad \mathsf{pad}_2(m_1, m_2) = \begin{cases} ((0, m_1), (\delta+1, \overline{m}_2)) & \text{if } m''_1 \neq 000 \\ ((0, m'_1\|100), (\delta-1, \overline{m}_2)) & \text{if } m''_1 = 000 \end{cases}$$

For all other cases, i.e., $s \geq 3$,

$$\mathsf{pad}_2(m) = \begin{cases} ((0, \overline{m'}_1), (5, m_2), (0, m_3), \cdots, (0, m_{s-1}), (\delta, \overline{m}_s)) \text{ if } m''_1 = 000 \\ ((0, m_1), (4, m_2), (0, m_3), \cdots, (0, m_{s-1}), (\delta, \overline{m}_s)) \text{ if } m''_1 \neq 000 \end{cases}$$

Note, $\Delta_{\mathsf{pad}_2} = \{0\}$ and it increases one block only when the message size is in between $n - 3$ and $n$. All other messages and their padded messages have same number of blocks. Now we prove that it is a prefix-free padding rule.

**Proposition 3.** $\mathsf{pad}_2$ *is prefix-free padding rule over the message space* $\{0, 1\}^*$.

**Proof.** Suppose that $m = m_1 \cdots m_{s-1} m_s$ and $m' = m'_1 \cdots m'_{s'-1} m'_{s'}$ where $s \leq s'$, $\mathsf{pad}_2(m)$ is a prefix of $\mathsf{pad}_2(m')$ and $m_1, m'_1, \cdots, m_{s-1}, m'_{s'-1} \in \{0,1\}^n, m_s \in \{0,1\}^r, m'_{s'} \in \{0,1\}^{r'}$, $1 \leq r, r' \leq n$. Let $\mathsf{pad}_2(m) = ((0, x_1), (\delta, x_2), \cdots)$ if $s \geq 2$; otherwise, $\mathsf{pad}_2(m) = (0, x_1)$. Similarly we denote $\mathsf{pad}_2(m') = ((0, x'_1), (\delta', x'_2), \cdots)$ if $s \geq 2$; otherwise, $\mathsf{pad}_2(m') = (0, x'_1)$. When $s = 1$, $s'$ must be 1; otherwise, for any $s' \geq 2$, we always have $(x_1, \delta) \neq (x'_1, \delta')$. It is also easy to see that if $s = s' = 1$, then $m = m'$. Let $s \geq 2$. By comparing the values of $\delta$ and $\delta'$ we must have $s = s' = 2$ or $s, s' \geq 3$. From the definition of $\mathsf{pad}_2$, one can check that $m = m'$. ■

We define the tag-generation algorithm of GCBC2 as the generalized CBC algorithm (see Figure 2) $\mathsf{gcbc}^{e,\mathsf{ls},\mathsf{pad}_2}$ with a message space of $\{0,1\}^*$ and the padding rule $\mathsf{pad}_2$ (see Algorithm 1 for generalized CBC tag generation algorithm or see a complete description of GCBC in introduction). The proof of the following theorem is immediate from Theorem 3 ($sigma' \leq \sigma + q$).

**Theorem 6.** (Security Bound of GCBC2)

$$\mathbf{Adv}^{\mathrm{prf}}_{\mathit{GCBC2}}(q, \sigma) \leq \frac{33(\sigma + q)(\sigma + q - 1)}{2^n} + \mathbf{Adv}^{\mathrm{prp}}_{e}(\sigma + q).$$

*Remark 4.* Our bound is of the form $\sigma^2/2^n$, whereas in [5], it had been shown that CBC-MAC is prf-secure for prefix-free messages with the security bound of the form $\ell q^2/2^n$, where $\ell$ denotes the number of blocks of the longest query among all $q$ queries. Note that GCBC1 can be viewed as a CBC-MAC, where the last message block is modified by the last intermediate chain value. Because of this modification, it seems hard to obtain prefix queries. If it is so, then we can apply the result from [5] to obtain a bound of the form $\ell q^2/2^n$ for GCBC1. This would be our future research work and we leave this as an open problem.

**An Efficient Variation Operation.** One can choose an efficient, different public variation operation $h = \mathsf{tr}$ for a generalized CBC MAC algorithm, which is defined as follows. Let $n'$ be a divisor of $n$ and $x = x_1 \cdots x_{n'}$, where $x_i \in \{0,1\}^w$. The actual value of $w$ can depend on the underlying block cipher and when using AES, we choose $w = 8$. Define, $\mathsf{tr}(0, x_1 \cdots x_{n'}) = (x_1, \cdots, x_{n'})$, $\mathsf{tr}(1, x_1 \cdots x_{n'}) = \mathsf{tr}(x_1 \cdots x_{n'}) := x_2 \cdots x_{n'} x_1^{\lll 1}$ and inductively define for $i \geq 2$,

$$\mathsf{tr}(i, x_1 \cdots x_{n'}) = \mathsf{tr}(i - 1, \mathsf{tr}(1, (x_1 \cdots x_{n'})) = \mathsf{tr}(i - 1, x_2 \cdots x_{n'} x_1^{\lll 1}).$$

In particular, we have $\mathsf{tr}(2, x_1 \cdots x_{n'}) := x_3 \cdots x_{n'} x_1^{\lll 1} x_2^{\lll 1}$ and $\mathsf{tr}(3, x_1 \cdots x_{n'}) := x_4 \cdots x_{n'} x_1^{\lll 1} x_2^{\lll 1} x_3^{\lll 1}$ and so on. This would be very efficient in software when we use a $w$-bit processor. In case of GCBC2 with the above defined variation operation, it needs at most three 8-bit shift operations. Note, an 8-bit implementation of a single shift on 128 bits needs 16 shift operations. For example, if we use AES, then a single shift on 128 bits (it is partitioned into 16 bytes) requires 16 shift operations and several bitwise-and and bitwise-or operations. The proof is very similar to that of proposition 2 and hence we omit it.

**Proposition 4.** *The operation* $\mathsf{tr}$ *with 5 variations is a public and weakly* $\frac{1}{2^{n-5}}$*-xor universal for* $\Delta = \Delta_{\mathsf{pad}_2} = \{0\}$.

## 6 Conclusion

In this paper, many popular CBC-type message authentication algorithms are viewed in a unified way. In particular, a wide class of authentication algorithms called generalized CBC algorithms is introduced. This class contains almost all known CBC-type secure authentication algorithms. Moreover, we have proposed two secure constructions GCBC1 and GCBC2 from this class which are optimum in key size and the number of block cipher invocations. These constructions may have significant performance compared to OMAC for short messages. We also characterize the prf-secure generalized CBC constructions. We hope the idea of generalizing CBC constructions can also help us to generalize other similar constructions for different security goals.

**Acknowledgement.** We would like to thank Elaine Barker, William Burr, Tetsu Iwata and the anonymous referees of FSE-09 who have made several editorial and technical comments. We would also like to thank Liting Zhang who found a small error on the earlier version of GCBC2 and based on his comments we have made a minor modification of GCBC2.

## References

1. Mihir Bellare, Ran Canetti, and Hugo Krawczyk. Keying Hash Functions for Message Authentication. In Neal Koblitz, editor, *CRYPTO*, volume 1109 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 1996.
2. Mihir Bellare, Ran Canetti, and Hugo Krawczyk. Pseudorandom Functions Revisited: The Cascade Construction and Its Concrete Security. In *FOCS*, pages 514–523, 1996.
3. Mihir Bellare, Roch Guérin, and Phillip Rogaway. XOR MACs: New Methods for Message Authentication Using Finite Pseudorandom Functions. In Don Coppersmith, editor, *CRYPTO*, volume 963 of *Lecture Notes in Computer Science*, pages 15–28. Springer, 1995.
4. Mihir Bellare, Joe Kilian, and Phillip Rogaway. The Security of the Cipher Block Chaining Message Authentication Code. *J. Comput. Syst. Sci.*, 61(3):362–399, 2000.

5. Mihir Bellare, Krzysztof Pietrzak, and Phillip Rogaway. Improved Security Analyses for CBC MACs. In Victor Shoup, editor, *CRYPTO*, volume 3621 of *Lecture Notes in Computer Science*, pages 527–545. Springer, 2005.
6. Mihir Bellare and Phillip Rogaway. The Security of Triple Encryption and a Framework for Code-Based Game-Playing Proofs. In Serge Vaudenay, editor, *EUROCRYPT*, volume 4004 of *Lecture Notes in Computer Science*, pages 409–426. Springer, 2006.
7. Daniel J. Bernstein. The Poly1305-AES Message-Authentication Code. In Henri Gilbert and Helena Handschuh, editors, *FSE*, volume 3557 of *Lecture Notes in Computer Science*, pages 32–49. Springer, 2005.
8. John Black, Shai Halevi, Hugo Krawczyk, Ted Krovetz, and Phillip Rogaway. UMAC: Fast and Secure Message Authentication. In Michael J. Wiener, editor, *CRYPTO*, volume 1666 of *Lecture Notes in Computer Science*, pages 216–233. Springer, 1999.
9. John Black and Phillip Rogaway. A Block-Cipher Mode of Operation for Parallelizable Message Authentication. In Lars R. Knudsen, editor, *EUROCRYPT*, volume 2332 of *Lecture Notes in Computer Science*, pages 384–397. Springer, 2002.
10. John Black and Phillip Rogaway. CBC MACs for Arbitrary-Length Messages: The Three-Key Constructions. *J. Cryptology*, 18(2):111–131, 2005.
11. Larry Carter and Mark N. Wegman. Universal Classes of Hash Functions. *J. Comput. Syst. Sci.*, 18(2):143–154, 1979.
12. Joan Daemen and Vincent Rijmen. The Design of Rijndael: AES - The Advanced Encryption Standard., 2002. http://csrc.nist.gov/CryptoToolkit/aes/rijndael/Rijndael-ammended.pdf.
13. Morris Dworkin. Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication. http://csrc.nist.gov/publications/nistpubs/index.html#sp800-38B.
14. Shoichi Hirose, Je Hong Park, and Aaram Yun. A Simple Variant of the Merkle-Damgård Scheme with a Permutation. In *ASIACRYPT*, pages 113–129, 2007.
15. Tetsu Iwata and Kaoru Kurosawa. OMAC: One-Key CBC MAC. In Thomas Johansson, editor, *FSE*, volume 2887 of *Lecture Notes in Computer Science*, pages 129–153. Springer, 2003.
16. Kaoru Kurosawa and Tetsu Iwata. TMAC: Two-Key CBC MAC. In Marc Joye, editor, *CT-RSA*, volume 2612 of *Lecture Notes in Computer Science*, pages 33–49. Springer, 2003.
17. Kazuhiko Minematsu and Yukiyasu Tsunoo. Provably Secure MACs from Differentially-Uniform Permutations and AES-Based Implementations. In Matthew J. B. Robshaw, editor, *FSE*, volume 4047 of *Lecture Notes in Computer Science*, pages 226–241. Springer, 2006.
18. J. Patarin. Etude des Générateurs de Permutations Basés sur le Schéma du D.E.S. Phd Thèsis de Doctorat de l'Université de Paris 6, 1991.
19. Phillip Rogaway. Bucket Hashing and Its Application to Fast Message Authentication. *J. Cryptology*, 12(2):91–115, 1999.
20. Serge Vaudenay. Decorrelation: A Theory for Block Cipher Security. volume 16, pages 249–286, 2003.
21. Kan Yasuda. "Sandwich" Is Indeed Secure: How to Authenticate a Message with Just One Hashing. In Josef Pieprzyk, Hossein Ghodosi, and Ed Dawson, editors, *ACISP*, volume 4586 of *Lecture Notes in Computer Science*, pages 355–369. Springer, 2007.