Fast and Secure CBC Type MAC Algorithms

Mridul Nandi

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

Abstract. CBC-MAC or cipher block chaining message authentication code is a well known method to generate message authentication code. Unfortunately, it is not forgery secure over arbitrary domain. There are several secure variants of CBC-MAC among which OMAC (or one-key CBC-MAC) is a widely used candidate. A simple variant of it called CMAC also has been recommended by NIST and is also used widely. Both of these cost $(s+1)$ blockcipher invocations to authenticate an s-block message and it takes only one blockcipher key. In this paper we propose two secure and efficient variants of CBC-MAC. Our constructions cost only s blockcipher invocations to authenticate an s-block message (except for few single block messages, in which case it costs two blockcipher invocations like OMAC) and they need only one blockcipher key. If AES is plugged into these new constructions then they are significantly faster than OMAC for short messages, roughly twice faster for messages up to 125 bits and 1.5 times faster for messages up to 256 bits.

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 "so far fastest" and secure blockcipher 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 MAC to be a pseudorandom function or prf which means that it is computationally indistinguishable from an ideal random function. The prf-security is a strong security notion and it guarantees unforgeable securities of MAC. In this paper we use "secure" and "prf-secure" words synonymously. Till now several secure and fast authentication algorithms are known. We first broadly classify them into three main categories based on the underlying building blocks.

Hash-Mac: These are based on hash functions. HMAC [1] is widely used candidate in this class which also has been standardized by National Institute of Standards and Technology or NIST. The other efficient popular candidates are cascaded-PRF [2], sandwich-MAC [24], KMDP [14] etc. The prf security of these are proved under prf assumption of the underlying compression function (in some cases a stronger security assumption such as related key prf may be needed). Thus, it is also known as prf-preserving domain extension since we extend the domain of prf from a fixed size (domain of the keyed compression function) to an arbitrary domain (usually $\{0,1\}^*$).

Universal Hash based Mac: Here universal hash functions and small domain pseudorandom functions are used. In software these are very fast for long messages [10, 21]. These generally cost field multiplications, key expansions, invocations of a smaller domain pseudorandom function etc. Thus, for short messages these may not always give similar performances. In [18], 4 round-AES is used to obtain an universal hash functions which eventually produces a very fast MAC algorithm for long messages (close to two times faster than OMAC). Again it is much slower than OMAC due to overhead when we have one or two block messages.

Blockcipher based Mac: In this paper we study this category in more details. These are usually based on several invocations of a blockcipher either in feedback mode (cipher block chaining or cbc type) or in parallel mode (e.g., PMAC [8], XOR-MAC [3] etc). A blockcipher is a permutation $e_K : \{0,1\}^n \rightarrow \{0,1\}^n$

for each key K chosen from the keyspace $\{0,1\}^k$ where n (block size) and k (key size) are positive integers (we fix these parameters throughout the paper). Recently, blockciphers have been used extensively in many applications. Moreover, several fast and "till now secure" blockciphers are known e.g., AES [11], RC6 [20] etc. Hence an authentication based on only blockcipher could be useful. The prf security of these are based on the "pseudorandom permutation" assumption of the underlying blockciphers meaning that the keyed blockcipher family is computationally indistinguishable from the ideal random permutation. CBC-MAC or cipher block chaining message authentication [4] is the first construction in this category. Now it is well known that CBC-MAC is not secure for variable length messages. Many different modifications of it have been proposed so far, among which OMAC [15] (or one-key CBC-MAC) and its variant CMAC [12] (NIST recommended) are efficient as well as requiring minimum key. Another simple modification called XCBC-MAC [9] or XCBC is faster in software but it needs three keys which may not be desired in many situations. These keys may be derived from one key at the cost of few blockcipher invocations which causes slower performance for short messages.

Our Contribution. In this paper, we first provide a general class of cbc type constructions called gcbc which includes almost all popularly known cbc type constructions. We also characterize prf-secure gcbc constructions and propose two so far fastest and secure candidates namely GCBC1 and GCBC2. We have implemented these with the blockcipher AES-128 in the platform Intel(R) Pentium(R) 4 CPU 3.60 GHz, 1GB RAM and provided a performance comparison in Table 1. One can see that GCBC2 is twice faster than OMAC for almost all single block messages and both GCBC1 and GCBC2 are 1.5 times faster for all two block messages. This is understood from the fact that our constructions need s blockcipher invocations to authenticate s block messages whereas OMAC needs $s+1$ blockcipher invocations. In nutshell, the two new proposed constructions are optimum among all generalized cbc type MAC in both keysize and the number of blockciphers and these are really useful in the applications where short message authentication is needed.

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 and when the underlying blockcipher is AES-128. Here # BC denotes the number of invocations of blockcipher $e : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ to authenticate an s block message. Time is computed by taking average over several executions. GCBC1 and GCBC2 are proposed in this paper.

Name of MAC	microsec (1-15 bytes)	microsec (16 bytes)	microsec (17 - 32 bytes)	# BC for s-block	Total Keysize
XCBC [9]	43.7	43.7	78.46	s	$k+2n$
TMAC [17]	43.98	44.05	78.80	s	$k+n$
OMAC [15]	78.72	78.80	113.80	$s+1$	k
GCBC1	-	-	77.95	s	k
GCBC2	43.58	78.26	78.37	s	k

Organization of the paper. We first provide basic definitions and notations and different cbc type MACs 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 detail security analysis of generalized cbc type constructions is given in section 4. Finally in section 5, we specify two fast and secure construction called GCBC1 and GCBC2 from the generalized class.

2 Preliminaries

2.1 Definitions and Notations

Given any set S, we write $S^+ = \cup_{i=1}^{\infty} S^i$ and $S^* = \cup_{i=0}^{\infty} S^i = S^+ \cup \{\lambda\}$ where λ is the empty string. For example, $\{0,1\}^+$ is the set of all non-empty finite bit-sequences. We write $|x| = i$ 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 $x_1, \cdots, x_i \in S$. We say $Y = (y_1, \cdots, y_j) \in S^*$ is a prefix of X if $j \leq i$ and $y_1 = x_1, \cdots, y_j = x_j$. Trivially λ is prefix of any X and we call it a trivial prefix. Any other prefixes will be called non-trivial prefixes. Let $x = x_1x_2\cdots x_n \in \{0,1\}^n$, $x_i \in \{0,1\}$, then for any two integers $i \leq j$ we denote the set $\{i, i+1, \cdots, j\}$ as $[i..j]$ and we denote $x_ix_{i+1}\cdots x_j$ as $x[i..j]$ whenever $1 \leq i \leq j \leq n$. If $i > j$, $x[i..j]$ is nothing but λ. To represent the ith bit of x we simply write $x[i]$. We use $x^{\ll t}$ (or $x^{\gg t}$) to denote t-bit left shift (or right shift respectively) of an n-bit string x. We identify $\{0,1\}^n$ as $\mathrm{GF}(2^n)$ by fixing a primitive polynomial $z^n + c_1z^{n-1} + \cdots + c_{n-1}z + c_n$ where $c_i \in \{0,1\}$. Let $0^n = \mathbf{0}$ (the additive identity), $0^{n-1}1 = \mathbf{1}$ (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 α 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 $c = c_1c_2\cdots c_n$. We denote $x \overset{*}{\leftarrow} S$ 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)
ρ 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 τ 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 $y_1, \cdots, y_q \in \{0,1\}^n$ we have

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

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 $\{0,1\}^n$. We denote the set of all functions from $\{0,1\}^n$ to $\{0,1\}^n$ as $\mathrm{Func}(n,n)$. Now an ideal random function from $\mathcal{M}$ to $\{0,1\}^n$ is chosen uniformly from $\mathrm{Func}(\mathcal{M}, \{0,1\}^n)$ (it is not possible when $\mathcal{M}$ is an infinite set). It is an equivalent 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 may write an ideal random function as keyed function family rand_ρ where $\mathsf{rand}_\rho(x) = \rho(x)$ and $\rho \in \mathrm{Func}(\mathcal{M}, \{0,1\}^n)$. A blockcipher $e : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ 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 block if $i \leq n$. It is called complete if $i = n$, otherwise called incomplete. For any $x \in \{0,1\}^*$, we denote $\lceil \frac{|x|}{n} \rceil$ as $||x||$ called 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, σ)-adversary if it makes at most q queries and the total number of blocks in all queries is at most σ. For simplicity we assume that a q-adversary makes exactly q queries as there is no loss making some extra dummy queries. We say q as number of input queries whereas σ as number of block-queries.

Definition 2. (pseudo random 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 $\mathcal{M}$. Now for any probabilistic oracle adversary A we define the prf-advantage of it over the function family F as*

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

where ρ is an ideal random function from $\mathcal{M}$ to $\{0,1\}^n$ and the probabilities are computed over internal randomness of A, uniform distribution of K' and randomness of output behavior of ρ. 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 maximum is taken over all (q,σ)-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 a function family F is (q,σ,ϵ)-prf (or (σ,ϵ)-prf in case of $\mathcal{M} = \{0,1\}^n$) if $\mathbf{Adv}_F^{\mathrm{prf}}(q,\sigma) \le \epsilon$.

Definition 3. (pseudo random permutation) *The prp-advantage of an oracle adversary A over a blockcipher $e : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ 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 τ on $\{0,1\}^n$. The prp-advantage of the blockcipher e is defined as $\mathbf{Adv}_e^{\mathrm{prp}}(q) = \max_A \mathbf{Adv}_e^{\mathrm{prp}}(A)$ where 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*

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

The proof of the switching lemma can be found in many literatures, e.g. [6].

2.2 Known Examples of CBC Type MAC Algorithms

For any integer $\ell \ge 1$, an ℓ-block tuple $(x_1, \cdots, x_\ell) \in (\{0,1\}^n)^\ell$, an initial value iv $\in \{0,1\}^n$ and a permutation $\pi \in \mathrm{Perm}(n)$, we define $\pi_{\mathrm{iv}}^{+}(x_1, \cdots, x_\ell) = \mathtt{v}_\ell$ where $\mathtt{v}_\ell$ is computed as follows;

$$\mathtt{v}_0 = \mathrm{iv};\ \mathtt{for\ i = 1\ to}\ \ell \quad \mathtt{v_i} = \pi(\mathtt{v_{i-1}} \oplus \mathtt{x_i}); \quad \mathtt{return}\ \mathtt{v}_\ell; \qquad (1)$$

The initial value iv can be chosen publicly (i.e., a fixed constant), or kept secret, or it can be the internal state.[1] When iv is a fixed constant, we choose iv $= \mathbf{0}$ (a sequence of 0-bits of size n) unless it's value is clearly mentioned. We write π^+ instead of $\pi_{\mathbf{0}}^{+}$. Note that $\pi^+ : (\{0,1\}^n)^+ \to \{0,1\}^n$. For a binary string x of size at most n, we denote

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

So $\overline{x}$ has size exactly n for any x with $0 \le |x| \le n$. Let $m = m_1 \| \cdots \| m_{s-1} \| m_s$ where $m_1, \cdots, m_{s-1} \in \{0,1\}^n$ and $|m_s| = r$ with $1 \le r \le n$. Then each m_i is said to be a block of the message. Note that m_s or the final message block can be either a complete (if $r = n$) or an incomplete block (if $r < n$). Unless we precisely mention we use the above representation of a message m with the variables s and r as stated above.

Informally a construction, which uses e_K^+ function as a major component given a blockcipher e, is called cbc-type construction (see equation 1 with $\pi = e_K$ and iv $= 0$). There are several cbc-type constructions. We study some of these here.

CBC-MAC[4]: Given any message m and a blockcipher e_K with a secret key K, output of CBC-MAC is defined as $\mathsf{cbc\text{-}mac}^{e_K}(m) = e_K^+(m10^d)$ where d is the smallest nonnegative integer such that $|m| + 1 + d$ is multiple of n. In [4, 5], it was shown that CBC-MAC is a prefix-free pseudorandom function (attacker is not

[1] internal state usually is updated in each computation and the currently used state should be transmitted along with other outputs

allowed to choose queries m and m' such that $m' = m\|x$ or m is a prefix of m') provided the underlying blockcipher is pseudorandom permutation. Moreover, there is an efficient distinguisher which can distinguish CBC-MAC from an ideal random function by making only two adaptive queries (definitely one of these has to be a prefix to the other). In particular, the following equation 2 holds with probability one when $f = \mathsf{cbc\text{-}mac}^{e_K}$ for any key K, but with negligible probability for the ideal random function f.

$$m \in \{0,1\}^n,\ f(m) = c \ \Rightarrow\ f(m \parallel c \oplus m) = c. \tag{2}$$

Since we do not have prefixes among fixed size inputs, CBC-MAC restricted on fixed size input is always a pseudorandom function. But for almost all practical purposes we need a pseudorandom function over arbitrary domain. Prepending length of the message is one possible way out [4] in which case, either the length of the the messages should be known in prior or we have to store the complete message in buffer before starting underlying blockcipher invocation.

XCBC[9]: To avoid the above mentioned problem, a prf-secure XCBC[9] was proposed. Let $K \in \{0,1\}^k$ (blockcipher key), $L_1, L_1 \in \{0,1\}^n$ be three independently chosen keys. Now we define

$$\mathsf{xcbc\text{-}mac}^{e_K, L_1, L_2}(m) = \begin{cases} e_K^{+}(m_1 \parallel \cdots \| \overline{m}_s \oplus L_1) & \text{if } |m_s| < n \\ e_K^{+}(m_1 \parallel \cdots \| m_s \oplus L_1)) & \text{if } |m_s| = n. \end{cases}$$

TMAC[17] and OMAC[15]: Note that XCBC needs three keys which may be expensive in some applications where a small key is desired. Keeping it in mind, TMAC or two-key CBC-MAC and OMAC or one-key CBC-MAC have been proposed. Here the two keys L_0 and L_1 are generated from either a single key L or $e_K(\mathbf{0})$. Let α be a primitive element of the Galois field of size 2^n. Then output of these authentication codes are defined as follows:

$$\mathsf{tmac}^{e_K, L}(m) = \begin{cases} e_K^{+}(m_1 \parallel \cdots \| \overline{m}_s \oplus L) & \text{if } |m_s| < n \\ e_K^{+}(m_1 \parallel \cdots \| m_s \oplus L \cdot \alpha) & \text{if } |m_s| = n. \end{cases}$$

$$\mathsf{omac}^{e_K}(m) = \mathsf{tmac}^{e_K, L}(m) mboxwhere L = e_K(\mathbf{0}) \cdot \alpha$$

In [17, 15, 16], they were shown to be prf. Among all these above constructions, OMAC is one of the good choice in terms of efficiency and key-size. It has been used in EAX [7] (a secure authenticated encryption), TET [13] (a length-preserving tweakable strong pseudo random permutation) and many others. A simple variant of it also has been recommended by NIST. In the next section we propose a class of message authentication codes which are significantly faster than OMAC for short messages.

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 which may be deterministic or probabilistic. Key-generation algorithm returns a key (K, L) at random from it's keyspace $\mathcal{K} \times \{0,1\}^{\ell}$. In this paper we consider deterministic tag-generation algorithms which have the following three main building blocks.

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 particular 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 $\delta_i \in [0,t]$, $1 \le i \le s$. We denote the set of all possible δ_1 values as

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

The roll of x_i's is similar to the message block of cbc whereas δ_i values take part to tweak the intermediate outputs of the blockcipher by using a variation operation (which is one of the building blocks). A padding rule pad is said to be *prefix-free* if for any $m \neq m'$, $\mathsf{pad}(m)$ is not 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 blockcipher e_K or an ideal random function rand_ρ for $\rho \overset{*}{\leftarrow} \mathrm{Func}(n,n)$ (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 usually very efficiently computable simple functions. It may be determined via a key L called *auxiliary key* and use the underlying iterative function f as a subroutine (in this way it may be determined by the key K of f). In this case we say it is a secret variation operation. If it does not use f and any auxiliary key L then h is publicly computable function and we say that it is a public variation operation. A simple example of 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 type-1 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-2. In this paper we consider public or type-1 secret variation operations when we study the security analysis of generalized cbc constructions. But we also see some secure constructions such as OMAC or CMAC which use type-2 secret variation operations.

3.2 Definition of a Generalized CBC-MAC

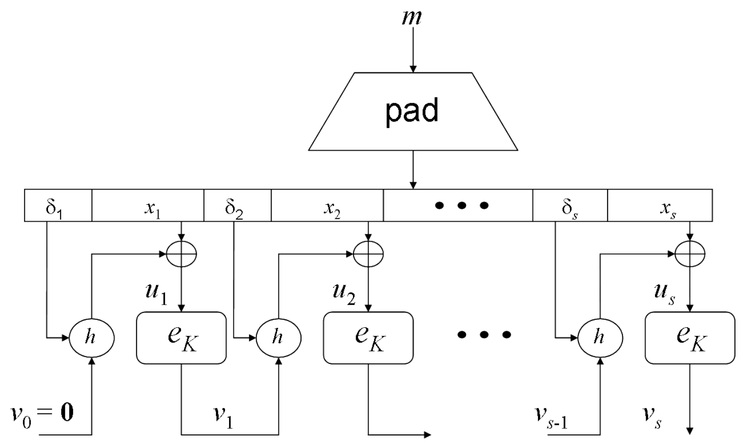

Fig. 1. Generalized CBC which uses variation operation h, an underlying iterative function e_K (blockcipher) and a padding rule pad.

Now we define a class of generalized cbc message authentication algorithms denoted as $\mathcal{C}_{\mathsf{gcbc}}$. Any authentication algorithm from the class for a message space $\mathcal{M}$ has two main functionalities namely a randomized key-generation algorithm or Key-Gen with a keyspace $\mathcal{K} \times \{0,1\}^\ell$ and a deterministic tag-generation algorithm $\mathsf{gcbc}^{f,h,\mathsf{pad}}$ (defined below).

1. Key-Gen : $(K, L) \overset{*}{\leftarrow} \mathcal{K} \times \{0,1\}^\ell$ where $\mathcal{K} \times \{0,1\}^\ell$ is the keyspace. So key-generation is parameterized by the keyspace 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 the padding rule pad are specified by the key (K, L) (output of $\mathsf{Key\text{-}Gen}$) where K is the key for the underlying iterative function f and L is the auxiliary key which is used for secret variation operation h. For public variation operation $\ell = 0$. Now 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 1);

   $$v_0 = 0^n,\ u_i = h(\delta_i, v_{i-1}) \oplus x_i, \quad v_i = f(u_i), 1 \leq i \leq s \tag{3}$$

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

In nutshell, to define an authentication algorithm we need to specify the keyspace $\mathcal{K} \times \{0,1\}^\ell$ (the value of ℓ, may be zero and the set $\mathcal{K}$), 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) and hence we denote the authentication algorithm as $\mathsf{gcbc}_{K,L}$ whenever all the above are clear from the context.

Remark 1. An efficiency of 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 pseudorandom function). Note that the number of invocations of f is at least s and it may be more if we use type-2 secret variation operation. To keep it small, we should carefully design a padding rule which gives the value of s 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 key generation algorithm Key-Gen
                                     \\ which are 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\}^*$ or $\mathcal{M} = \cup_{i>n}\{0,1\}^i$.
    input.    $m \in \mathcal{M}$.
1: $X = \mathsf{pad}(m)$
2: divide $X$ as $((\delta_1, m_1) \cdots (\delta_s, x_s))$ where $x_i \in \{0,1\}^n, \delta_i \in [0,t], 1 \leq i \leq s$
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

This class is indeed a generalized class as it contains almost all cbc type authentication algorithms. Now we describe more precisely how these following popular candidates XCBC, TMAC and OMAC belong to the class. A common choice of the underlying iterative function is a blockcipher 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 $m_1, \cdots, m_{s-1} \in \{0,1\}^n$, $m_s \in \{0,1\}^r$, $1 \leq r \leq n$, the padding rule with two variation number 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 $\delta = 2$. So it remains to define the value of ℓ and 2-variate variation operations h which are described below for each constructions. Recall that key generation algorithm returns a key $(K, L) \xleftarrow{*} \{0,1\}^k \times \{0,1\}^\ell$.

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

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

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

Remark 2. The OMAC is an example where the variation operation is type-2 secret. In the case of the other two examples the variation operations are type-1 secret variation operations. In the next section we propose two different padding rules and two public variation operations. Usage of public variation operations help to keep the key size lowest possible. We also see these public variation operations are efficiently computable. We should be careful in security analysis when we choose a public variation operation since the security analysis is not straightforward.

4 Security Analysis

4.1 Decorrelation Technique

We state Vaudeney's decorrelation theorem (Lemma 22 of [23][2]). Based on our notations, we state the following version of decorrelation theorem.

Theorem 1. (Decorrelation Theorem)
Let q and σ be two fixed integers and let $F_{K'} : \mathcal{M} \to \{0,1\}^n$ be a family of functions indexed by key K' chosen uniformly from the keyspace $\mathcal{K}'$. Suppose the following holds for some positive real numbers ϵ_1 and ϵ_2;

C1. *there exists a subset $\mathcal{Y} \subseteq (\{0,1\}^n)^q$ such that $|\mathcal{Y}| \geq 2^{nq}(1 - \epsilon_1)$ and*

C2. $\Pr[F_{K'}(x_1) = y_1, \cdots, F_{K'}(x_q) = y_q : K \xleftarrow{*} \mathcal{K}] \geq 2^{-nq}(1 - \epsilon_2)$ *for any $(y_1, \cdots, y_q) \in \mathcal{Y}$ and q distinct $m_1, \cdots, m_q \in \mathcal{M}$ with $\sum_{i=1}^{q} \|m_i\| \leq \sigma$.*

Then for any distinguisher A which asks q queries with σ many blocks present in all queries, $\mathbf{Adv}_F^{\mathrm{prf}}(A) \leq \epsilon_1 + \epsilon_2$.

What does Decorrelation Theorem mean? The second condition C2 means that the output behavior of the function family is close to that of an ideal random function when the output-tuple $(y_1, \cdots, y_k)$ are from a set $\mathcal{Y}$. Note, $2^{-nq} = \Pr[\rho(x_1) = y_1, \cdots, \rho(x_q) = y_q]$. The first condition means that with high probability the output-tuple of the ideal random function are from $\mathcal{Y}$. So the given conditions say eventually that the output probability distribution of the function family for any set of q inputs are almost identical with that of an ideal random function. So prf-advantage of a (q, σ)-adversary should be small independent of how adversary works. Note that adversary at the end of query-responses has a set of inputs and outputs and he has to distinguish based on it. However the values of ϵ_1 and ϵ_2 can depend on q and σ.

What is the difference from the game-based approach? Conceptually the decorrelation theorem corresponds to the game-based reductions [22, 6]. In the game-based reductions, the response of function

[2] it was mentioned in [23] that the decorrelation theorem was freely adapted from Patarin's coefficient H-techniques [19]

family are viewed as a game say G_0. The function family game is modified sequentially into the ideal function family game G_t (or in the other direction) via some intermediate games. Now it is usually shown that the games G_0 and G_t are equivalent given that some bad event bad does not occur and hence the distinguishing advantage is bounded by $\Pr[\mathsf{bad}]$. The rest is devoted to compute the probability. In the decorrelation technique, there are two bad events bad_1, the output tuple $(y_1, \cdots, y_q) \notin \mathcal{Y}$ and bad_2 (depending on the definition of $F_{K'}$). It is easy to see that C1 is true with $\epsilon_1 = \Pr_\rho[\mathsf{bad}_1] = \frac{2^{nq} - |\mathcal{Y}|}{2^{nq}}$. The second bad event would help us to find a bound of the form

$$\Pr_K[F_{K'}(m_1) = y_1, \cdots, F_{K'}(m_q) = y_q | \neg\mathsf{bad}_2] = 2^{-nq}$$

which would eventually provide the second condition C2 with $\epsilon_2 = \Pr_K[\mathsf{bad}_2]$. Now we can use decorrelation theorem to bound the prf-advantage. Decorelation technique has advantage over game based technique as it involves only probability computation which can be verified from the basic knowledge of probability theory.

The value of ϵ_1. Now we describe the set $\mathcal{Y}$ which is going to be used throughout the paper. In fact, it is the most common choice. It is the set of all q-tuples which contain all q distinct elements.

$$\mathcal{Y}_{\text{noColl}} := \{(y_1, \cdots, y_q) \in (\{0,1\}^n)^q : y_i = y_j, \forall i = j\}.$$

Now it is easy to see that $|\mathcal{Y}_{\text{noColl}}| \geq (1 - \frac{q(q-1)}{2^{n+1}}) \times 2^{nq}$. So $\epsilon_1 = \frac{q(q-1)}{2^{n+1}}$ in C1. The value of ϵ_1 does not depend on the construction F_K as we compute the probability over random function ρ.

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 type-1 secret (which does not use underlying iterative as a subroutine and only uses an auxiliary key).

Definition 4. *An allowed t-variate variation operation $h : [0..t] \times \{0,1\}^n \to \{0,1\}^n$ is said to be a (ϵ, Δ)-xor weak universal operation if for any $0 \leq \delta = \delta' \leq t$, $c \in \{0,1\}^n$ and for all auxiliary key $L_0 \in \{0,1\}^\ell$ the following conditions are satisfied.*

W1: $\Pr[h_{L_0}(i, R) = c : R \overset{*}{\leftarrow} \{0,1\}^n] \leq \epsilon$

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

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

Note that when Δ is a singleton set then the condition W2 is vacuously true. Now we characterize generalized cbc constructions which are prf-secure. Let pad be a prefix-free padding rule with a variation number $t \geq 0$, h be a $(\epsilon, \Delta_{\mathsf{pad}})$-xor weak universal allowed variation operation and the underlying iterative function family $(f_K)_{K \in \mathcal{K}}$ is (σ, μ)-prf. Then we show that the generalized cbc based on the above such building blocks is (q, σ, ϵ)-prf where $\epsilon = \sigma'(\sigma' - 1) \times (\epsilon + \frac{1}{2^n}) + \frac{q(q-1)}{2^{n+1}}$ where σ' denote the total number of blocks in all q padded messages (which have been queried by an adversary). Later we propose two padding rules where the the number of block can only increase by one for each message and hence $\sigma' \leq \sigma + q$. Now we try to compute ϵ_2 in the condition C2 of the decorrelation theorem for the function family $\mathsf{gcbc}_{\rho,L}$ where the above underlying function family $(f_K)_{K \in \mathcal{K}}$ is replaced by an ideal random function family $(\mathsf{rand}_\rho)_{\rho \in \text{Func}(n,n)}$. More precisely for distinct $m_1, \cdots, m_q \in \mathcal{M}$ and distinct $y_1, \cdots, y_q \in \{0,1\}^n$ we want to compute

$$p = \Pr_{\rho,L}[\mathsf{gcbc}_{\rho,L}(m_1) = y_1, \cdots, \mathsf{gcbc}_{\rho,L}(m_q) = y_q]$$

where probability is computed over $(\rho, L) \overset{*}{\leftarrow} \text{Func}(n,n) \times \{0,1\}^\ell$. We first introduce some notations which are needed to compute the above probability.

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.

$$v_i\text{-sequence} \ : \ \text{iv} \overset{(\delta_{i,1},x_{i,1})}{\longrightarrow} v_{i,1} \overset{(\delta_{i,2},x_{i,2})}{\longrightarrow} v_{i,2} \cdots \overset{(\delta_{i,s_i},x_{i,s_i})}{\longrightarrow} v_{i,s_i}$$
$$u_i\text{-sequence} \ : \ \lambda \overset{(\delta_{i,1},x_{i,1})}{\longrightarrow} u_{i,1} \overset{(\delta_{i,2},x_{i,2})}{\longrightarrow} u_{i,2} \cdots \overset{(\delta_{i,s_i},x_{i,s_i})}{\longrightarrow} u_{i,s_i}$$

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 $\mathsf{rand}_\rho(u_{i,j}) = v_{i,j}$. The following lemma is easy to prove by looking at the above u_i's and v_i's sequences and the definition of $X_{i,j}$.

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. *We say 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*
- *$h_{L_0}(\delta_{i,j}, V_{i,j-1}) \oplus x_{i,j} = h_{L_0}(\delta_{i',j'}, V_{i',j'-1}) \oplus x_{i',j'}$ for all i, j, i', j' such that $X_{i,j} = X_{i',j'}$.*

Let σ_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, $\sigma_1 \le \sigma' = \sum_{i=1}^{q} ||X_i||$ and hence $\sigma_1 \le \sigma + q$.

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. This is true since for 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, $\Pr[L = L_0, v_{i,j} = V_{i,j}] = \Pr[L = L_0, \rho(U_{i,j}) = V_{i,j} \text{ for all } i,j] = \Pr[\rho(U_{i,j}) = V_{i,j} \text{ for all } i,j] \times \Pr[L = L_0] = \frac{1}{2^{n(\sigma_1+q)}} \times \frac{1}{2^\ell}$ ■

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 $2^{n\sigma_1+\ell}$ many tuples $(L_0, (V_{i,j})_{i,j})$ such that $V_{i,j} = V_{i',j'}$ whenever $X_{i,j} = X_{i',j'}$ and $V_{i,0} = 0^n, V_{i,s_i} = y_i$ for all $1 \le i \le q$. Now we need to find estimate of number of tuples among these such that $h_{L_0}(\delta_{i,j}, V_{i,j-1}) \oplus x_{i,j} = h_{L_0}(\delta_{i',j'}, V_{i',j'-1}) \oplus x_{i',j'}$ for all i, j, i', j' such that $X_{i,j} = X_{i',j'}$. To do so we count the complement. Suppose for some i, j, i', j' with $X_{i,j} = X_{i',j'}$, $h_{L_0}(\delta_{i,j}, V_{i,j-1}) \oplus x_{i,j} = h_{L_0}(\delta_{i',j'}, V_{i',j'-1}) \oplus x_{i',j'}$. The number of such tuple 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 $X_{i,j} = X_{i',j'}$ 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 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 the underlying iterative function is an ideal random function $(\mathsf{rand}_\rho)_{\rho \in \text{Func}(n,n)}$ and we denote the corresponding generalized cbc authentication algorithm as $\mathsf{gcbc}_{\rho,L}$. Now let σ' be the largest number of blocks after padding q messages having at most σ many blocks in total. Then for any distinct q inputs $m_1, \cdots, m_q \in \mathcal{M}$ (message space) and any q outputs $y_1, \cdots, y_q \in \{0,1\}^n$,*

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

where $\epsilon' = \sigma'(\sigma'-1) \times (\epsilon + \frac{1}{2^n})$.

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

$$\begin{aligned} \mathbf{Adv}^{\text{prf}}_{\mathsf{gcbc}_{K,L}}(q,\sigma) &\le \sigma'(\sigma'-1) \times (\epsilon + \frac{1}{2^n}) + \frac{q(q+1)}{2^{n+1}} + \mathbf{Adv}^{\text{prf}}_{f_K}(\sigma) \\ &\le \sigma'(\sigma'-1) \times (\epsilon + \frac{1}{2^n}) + \frac{q(q+1)}{2^n} + \mathbf{Adv}^{\text{prp}}_{f_K}(\sigma) \end{aligned}$$

5 Two New Efficient Generalized CBC : GCBC1 and GCBC2

In this section we propose two secure generalized cbc constructions namely GCBC1 and GCBC2. The first construction called GCBC1 which has message space $\{0,1\}^{>n}$ and it's tag-generation algorithm costs only s blockcipher invocations for any s-block messages, $s \geq 2$. If we pad enough bits so that every padded messages has at least two blocks then it can have message space $\{0,1\}^*$. We call this variant $\mathsf{GCBC1}'$. This variant costs 2 blockcipher invocations for one block messages (same as OMAC) and costs s blockcipher invocations for s-block messages (still one less).

The second construction called GCBC2 has message space $\{0,1\}^*$. The padding rule of it has been defined carefully so that it costs one blockcipher for almost all single block messages (for all messages with bit size at most $n-4$). The variation operation is also very fast in software which needs at most three **8-bit shift operations**. Note that a 8-bit implementation of single shift on 128 bit needs 16 shift operations. For example, if we use AES then a single shift on 128 bit (it is partitioned into 16 8-bits) requires 16 shift operations and several bitwise-and, bitwise-or operations.

5.1 GCBC1

Algorithm 2 $\mathsf{GCBC1}'$

```
Require:
    key.       $K \xleftarrow{*} \{0,1\}^k$. \\ blockcipher key
    function.  $e_K : \{0,1\}^n \to \{0,1\}^n$. \\ blockcipher
    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:     $m_1 = \overline{m}_1, m_2 = \mathbf{0}$, $s = 2, r = n-1$.
 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:     $u_s = v_{s-1}^{\ll 1} \oplus x_s$
14: else
15:     $u_s = v_{s-1}^{\ll 2} \oplus x_s$
16: end if
17: $v_s = e_K(u_s)$
18: return $v_s$
```

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 $m_s \in \{0,1\}^r$, $1 \leq r \leq n$, we define the padded message as

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

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

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

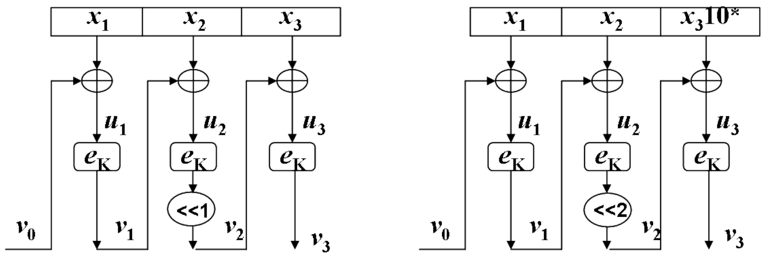

Fig. 2. GCBC1, Generalized CBC, which uses left shift variation operation, an underlying iterative function e_K (blockcipher) and a simple padding rule.

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* 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 $s \geq 2$: Then $\mathsf{pad}_1(m) = ((0, m_1), \cdots, (0, m_{s-1}), (\delta, \overline{m}_s)$ is prefix of $\mathsf{pad}_1(m') = ((0, m'_1), \cdots, (0, m'_{s'-1}), (\delta', \overline{m'}_{s'})$. Since $\delta, \delta' = 0$, $s' = s$ 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 $s = 1, s' \geq 2$: Now $\mathsf{pad}_1(m) = ((0, x_1), (1, x_2))$ where $\mathsf{pad}_1(m') = ((0, m'_1), \cdots, (0, m'_{s'-1}), (\delta', \overline{m'}_{s'})$. By comparing δ values of 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 not 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 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 ls which has two variations. For any $x \in \{0,1\}^n$ and $0 \leq \delta \leq 2$, $\mathsf{ls}(\delta, x) = x^{\ll \delta}$.

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

Proof. By definition, $\mathsf{ls}(0, x) = x$. Now it is easy to see that $x^{\ll i} = c$ 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 $\Delta = \Delta_1 = \{0\}$. To see the condition W3 we first prove that the number of solutions of $x \oplus x^{\ll 1} = c$ 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 $x \oplus x^{\ll 2} = c$ is exactly one. Now we want to find the number of solutions of the equation $x^{\ll 1} \oplus x^{\ll 2} = c$. Let $y = x^{\ll^1}$ then 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, ls is a weakly $\frac{1}{2^{n-2}}$-xor universal. ■

Let e be a blockcipher then we define the tag-generation algorithm of $\mathsf{GCBC1}$ as the generalized cbc algorithm (see Figure 2) $\mathsf{gcbc}^{e,\mathsf{ls},\mathsf{pad}}$ with message space $\{0,1\}^{>n}$. If we consider the message space $\{0,1\}^*$ and choose the extended definition of the padding rule pad_1 then we obtain a variant of $\mathsf{GCBC1}$ called $\mathsf{GCBC1}'$ (see Algorithm 2).

Theorem 4. (Security Bound of GCBC1)

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

Proof. We apply the result of the above two propositions to the generalized cbc security bound (see Theorem 3). ■

5.2 GCBC2

Now we first define a padding rule pad_2 for the message space $\{0,1\}^*$ with variation number 3. Let $m = m_1 \cdots m_{s-1} m_s \in \{0,1\}^{>n}$ where $m_1, \cdots, m_{s-1} \in \{0,1\}^n$ and $m_s \in \{0,1\}^r$, $1 \le r \le n$. Let us denote $\delta = 1$ if $r < n$, otherwise $\delta = 2$. If $|m| \ge n-3$ then denote $m_1 = m_1' \| m_1''$ where $m_1' \in \{0,1\}^{n-3}$ and $m_1'' \in \{0,1\}^*$. Now we define $\mathsf{pad}_2(m)$ depending on the values of s.

$s = 1$,

$$\mathsf{pad}_2(m_1) = \begin{array}{ll} ((0, m_1'\langle 3\rangle_3), (0, \overline{m''}_1)) & \text{if } r \ge n_3 \\ (0, \overline{m}_1) & \text{if } r \le n-4 \end{array}$$

$s = 2$,

$$\mathsf{pad}_2(m_1, m_2) = \begin{array}{ll} ((0, m_1), (\delta, \overline{m}_2)) & \text{if } m_1' = 000 \\ ((0, m_1'\langle \delta\rangle_3), (0, \overline{m}_2)) & \text{if } m_1'' = 000 \end{array}$$

$s \ge 3$,

$$\mathsf{pad}_2(m) = \begin{array}{l} ((0, \overline{m'}_1), (0, m_2), \cdots, (0, m_{s-1}), (\delta, \overline{m}_s)) \text{ if } m_1'' = 000 \\ ((0, m_1), (3, m_2), \cdots, (0, m_{s-1}), (\delta, \overline{m}_s)) \text{ if } m_1'' = 000 \end{array}$$

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. pad_2 *is prefix-free padding rule over the message space* $\{0,1\}^*$.

Proof. Suppose $m = m_1 \cdots m_{s-1} m_s$ and $m' = m_1' \cdots m_{s'-1}' m_{s'}'$ where $s \le s'$, $\mathsf{pad}_1(m)$ is a prefix of $\mathsf{pad}_1(m')$ and $m_1, m_1', \cdots, m_{s-1}, m_{s'-1}' \in \{0,1\}^n$, $m_s \in \{0,1\}^r$, $m_{s'}' \in \{0,1\}^{r'}$, $1 \le r, r' \le n$. Let $\mathsf{pad}_2(m) = ((0, x_1), (\delta, x_2), \cdots)$ (if $s \ge 2$ otherwise $\mathsf{pad}_2(m) = (0, x_1)$). Similarly we denote $\mathsf{pad}_2(m') = (0, x_1'), (\delta', x_2'), \cdots)$ (if $s \ge 2$ otherwise $\mathsf{pad}_2(m') = (0, x_1')$).

Case $s = 1$: Note that s' must be 1, otherwise for any $s' \ge 2$ we always have $(x_1, \delta) = (x_1', \delta')$. Now, it is also easy to see that if $s = s' = 1$ then $m = m'$.

Case $s \ge 2$: By comparing the values of δ and δ' we must have $s = s' = 2$ or $s, s' \ge 3$. One can check from the definition of pad_2 in these two cases, we also have $m = m'$. ■

We choose a public variation operation $h = \mathsf{tr}$ which is defined as follows. Let m be a divisor of n. We write $x = x_1 \cdots x_m$ where $x_i \in \{0,1\}^w$. The actual value of w can depend on the underlying blockcipher and in case of AES we choose $w = 8$. Now we define $\mathsf{tr}(1, x_1 \cdots x_m) = (x_1, \cdots, x_m), \mathsf{tr}(1, x_1 \cdots x_m) = \mathsf{tr}(x_1 \cdots x_m) := x_2 \cdots x_m x_1^{\lll 1}$ and inductively define

$$\mathsf{tr}(i, x_1 \cdots x_m) = \mathsf{tr}(i-1, \mathsf{tr}(1, (x_1 \cdots x_m)) = \mathsf{tr}(i-1, x_2 \cdots x_m x_1^{\lll 1}), \text{ for } i \ge 2.$$

Proposition 4. *The operation* tr *with* 3 *variations is a public and weakly* $\frac{1}{2^{n-3}}$*-xor universal for* $\Delta = \Delta_{\mathsf{pad}_1} = \{0\}$.

Proof. By definition, $\mathsf{tr}(0, x) = x$. It is also easy to see that the number of $x = x_1 \cdots x_m$ such that

$$x_2 \cdots x_m x_1^{\lll 1} = c \text{ or } x_3 \cdots x_m x_1^{\lll 1} x_2^{\lll 1} = c \text{ or } x_4 \cdots x_m x_1^{\lll 1} x_2^{\lll 1} x_3^{\lll 1} = c$$

has at most eight solutions for any fixed constant c. So the condition W1 is true. The condition W2 trivially holds since $\Delta = \Delta_1 = \{0\}$. One can also show that the number of solutions of $\mathsf{tr}(i, x) \oplus \mathsf{tr}(j, x) = c$ is at most four for any fixed constant c and $0 \le i = j \le 3$. Combining all these observations we can see that condition W3 is true. Thus, tr is a weakly $\frac{1}{2^{n-3}}$-xor universal. ■

Theorem 5. (Security Bound of GCBC2)

$$\mathbf{Adv}^{\mathrm{prf}}_{\mathit{GCBC2}}(q,\sigma) \leq \frac{\sigma(\sigma-1)}{2^{n-5}} + \mathbf{Adv}^{\mathrm{prp}}_{e}(\sigma)$$

Proof. We apply the result of the above two propositions to the generalized cbc security bound (see Theorem 3). ■

6 Conclusion

In this paper we view many popular cbc type message authentication algorithms in a unified way. In particular we introduce a wide class of authentication algorithms called generalized cbc algorithms. This class contains almost all known cbc type secure authentication algorithms. Moreover, we have found two more secure constructions GCBC1 and GCBC2 from this class which are optimum in key size and number of blockcipher invocations. These constructions are significantly faster than the widely used candidates OMAC and CMAC for short messages. We also characterize which generalized cbc constructions are prf-secure. We hope the idea of generalizing cbc constructions can also help us to generalize other similar constructions in different security goals.

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 Vaudenay [23], pages 409–426.
7. Mihir Bellare, Phillip Rogaway, and David Wagner. The eax mode of operation. In Bimal K. Roy and Willi Meier, editors, *FSE*, volume 3017 of *Lecture Notes in Computer Science*, pages 389–407. Springer, 2004.
8. 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.
9. John Black and Phillip Rogaway. Cbc macs for arbitrary-length messages: The three-key constructions. *J. Cryptology*, 18(2):111–131, 2005.
10. Larry Carter and Mark N. Wegman. Universal classes of hash functions. *J. Comput. Syst. Sci.*, 18(2):143–154, 1979.
11. Joan Daemen and Vincent Rijmen. The design of rijndael: Aesthe advanced encryption standard., 2002. http://csrc.nist.gov/CryptoToolkit/aes/rijndael/Rijndael-ammended.pdf.
12. Morris Dworkin. Recommendation for block cipher modes of operation: The cmac mode for authentication. http://csrc.nist.gov/publications/nistpubs/index.html#sp800-38B.
13. Shai Halevi. Invertible universal hashing and the tet encryption mode. In Alfred Menezes, editor, *CRYPTO*, volume 4622 of *Lecture Notes in Computer Science*, pages 412–429. Springer, 2007.
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. Tetsu Iwata and Kaoru Kurosawa. Stronger security bounds for omac, tmac, and xcbc. In Thomas Johansson and Subhamoy Maitra, editors, *INDOCRYPT*, volume 2904 of *Lecture Notes in Computer Science*, pages 402–415. Springer, 2003.
17. 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.
18. 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.
19. J. Patarin. Etude des generateurs de permutations bases sur le schema du d.e.s. Phd Thesis de Doctorat de l'Universite de Paris 6, 1991.
20. Ronald L. Rivest, Matthew J. B. Robshaw, and Yiqun Lisa Yin. Rc6 as the aes. In *AES Candidate Conference*, pages 337–342, 2000.
21. Phillip Rogaway. Bucket hashing and its application to fast message authentication. *J. Cryptology*, 12(2):91–115, 1999.
22. Victor Shoup. Sequences of games: a tool for taming complexity in security proofs. Cryptology ePrint Archive, Report 2004/332, 2004. http://eprint.iacr.org/.
23. Serge Vaudenay, editor. *Advances in Cryptology - EUROCRYPT 2006, 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28 - June 1, 2006, Proceedings*, volume 4004 of *Lecture Notes in Computer Science*. Springer, 2006.
24. 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.