Parallelizable MAC Revisited

Wun-She Yap[1,2], Sze Ling Yeo[1], Swee-Huay Heng[2]
& Matt Henricksen[1]
[1]Institute for Infocomm Research, A*STAR, Singapore.
{wsyap,slyeo,mhenricksen}@i2r.a-star.edu.sg
[2]Faculty of Information Science & Technology,
Multimedia University, Malaysia.
shheng@mmu.edu.my

February 1, 2013

Abstract

Message authentication codes (MACs) are widely used in communication networks for authentication purposes. In Eurocrypt 2002, Black and Rogaway proposed a parallelizable MAC (PMAC) which is relatively efficient when a parallel environment is possible. This parallelism is achieved via constant multiplications in the underlying finite field. In order to yield a better solution, Rogaway refined PMAC in Asiacrypt 2004 by using powering-up construction to generate the constants. This is in contrast to the first design that used successive words of the gray code to generate the constants. In this paper, we analyze how some unique characteristics of these constants result in weaknesses of the respective PMAC designs against forgery attacks in different ways. Our analysis highlights some pitfalls that designers should be mindful of when designing schemes which exploit such constants.

Keywords: Communication networks, MAC, authentication, security analysis, forgery attack

1 Introduction

1.1 Background

Message authentication codes (MACs) can be viewed as the symmetric key equivalent of digital signatures. Essentially, MAC is a symmetric key cryptographic scheme that serves to authenticate both the source of a message and its integrity. It is widely used in the mobile, wireless and data communication networks due to its high efficiency.

At present, there exist several approaches to design a MAC scheme. In particular, MACs can be constructed based on cryptographic hash functions

(HMAC [1]), block ciphers (CBC-MAC [11], XCBC [3], TMAC [15], OMAC [12]) or even universal hash functions (MMH [8], UMAC [2]). Among these MAC schemes, CBC-MAC and HMAC are the most popular. However, these two schemes share a common characteristic of being inherently sequential where one can only process the i-th message block after all the previous message blocks have been processed.

To counter this bottleneck, Black and Rogaway [4, 5] proposed a provably secure and parallelizable MAC (PMAC) scheme based on block ciphers. More precisely, Black and Rogaway proved that PMAC approximates a random function as long as the underlying n-bit block cipher is a pseudorandom permutation. In fact, PMAC [19] was proposed in response to NIST's call for contributions for the first mode of operation workshop. In Asiacrypt 2004, Rogaway [18] proposed the use of a tweakable block cipher for MACs and refined PMAC to yield a more efficient PMAC using powering-up construction. We refer to the two different versions as PMAC1 [19, 4, 5] and PMAC2 [18] throughout this paper.

For the design of PMACs, the parallelism is achieved at the expense of an extra constant multiplication in the underlying finite field $GF(2^n)$, where n is the size of block cipher. The distinction between these two variations of PMACs lies in the definition (or generation) of the constant multipliers involved. More precisely, PMAC1 uses successive words in a gray code to construct the constants such that two consecutive constants differ in exactly one bit (so the Hamming distance is 1). On the other hand, PMAC2 uses an easier-to-compute sequence of constants comprising $x^1, x^2, \ldots, x^{2^n-1}$ based on the squaring operation. These constants are then multiplied with a point $L \in GF(2^n)$ to form the corresponding masks for the blocks . Since it is computationally efficient to multiply a point by x (requiring only a shift and/or an addition), PMAC2 gives rise to a relatively more efficient MAC scheme.

1.2 Previous Work

Even though PMAC1 had been proposed since 2001, there was no known cryptanalytic result on the PMACs that quantifies the number of message queries with which a forgery attack can be performed until Lee *et al.* took a first step in analyzing the security of PMAC1 in [16]. They showed how forgery attacks on PMAC1 can be devised, both with truncation (i.e., a tag is truncated to a certain value before being generated) and without truncation (i.e., the length of a tag equals the block size, n).

For PMAC1 without truncation, an attacker first obtains the tags for $2^{n/2+1}$ different messages. Using the birthday paradox arguments, we can expect to find the collision of two tags between two different messages with a probability of 0.63. The attacker can then exploit this collision to find out certain information on the key and thus forge a tag for a new message which does not belong to the earlier set of $2^{n/2+1}$ selected messages.

A similar forgery attack can be carried out in case truncation of the tags is specified. By truncation, we mean that the final tag comprises only the first τ bits of the n-bit output, where $\tau < n$. Once again, an attacker will request

the tags of $2^{n/2+1}$ different messages. We can then expect at least $2^{n-\tau}$ of these messages to share the same tag with a probability of 0.63 according to the birthday paradox. However, since collision only applies to the first τ bits (rather than the entire output), additional computations on these $2^{n-\tau}$ messages need to be performed in order to extract useful information to forge other valid tags.

In both these scenarios, we see that the attacker can launch a successful forgery attack with $2^{n/2+1}$ message queries (i.e., number of (message, tag) pairs observed) with the success rate of 0.63 due to the birthday bound. We remark that this attack does not contradict the security bound shown by Black and Rogaway as the complexity/resource needed by this attack tallies with their bound. Thus, it proves the tightness of their bound. Moreover, the key recovery attack on the underlying block cipher proposed by Lee *et al.* is merely an exhaustive key search attack.

1.3 Motivation and Contributions

Given any MAC, exploiting the birthday paradox arguments will, with a high probability, lead to a collision of the tags of at least two different messages after $O(2^{n/2})$ (the birthday bound) queries. More importantly, the birthday bound is the security benchmark for all of XCBC, TMAC and OMAC, that is, attacks exist for all of these MACs after obtaining around $2^{n/2}$ (message, tag) pairs. Thus, a forgery attack based on birthday paradox arguments cannot be treated as a weakness of such MACs. Users of these MACs can change the secret key before reaching the birthday bound.

In a real world scenario, it is highly impractical to obtain for a large number of (message, tag) pairs, that is, to obtain as many as $O(2^{n/2})$ (message, tag) pairs under the same secret key K. This further invalidates the effectiveness of the above birthday attack. In some environments, it may however be feasible to perform a large number of tag verifications whereby (message,tag) pairs are submitted to test for their validity. For example, an attacker can forward many transmissions to a server in an attempt to authenticate a client using their common key functions as the verification oracle.

In order to achieve parallelism, both PMAC1 and PMAC2 employ two different methods to generate the constant multipliers. In this paper, we seek to analyze the effect of the constants on the resilience of the PMAC schemes against forgery attacks. In particular, we present weaknesses of the two PMAC schemes arising from some special properties of the constants which will in turn result in greater forgeability of the schemes. Thus, some extra cautions need to be considered when different methods in generating the constants are used. We also emphasize that our attacks exploit the structure properties of PMAC schemes.

For PMAC1, we show that the use of the gray code to generate the constant multipliers renders the scheme *less random* in the following sense: There exists a message M such that the probability of forging a different message having the same tag is higher than 2^{-n}. More significantly, this probability is greater for larger number of message blocks. Observe that in the security proof given

in [8, Proof of Lemma 2, Case(D_3, D_5)], the authors claimed that any two random messages, irrespective of their number of message blocks, will have equal tags with a probability of $1/2^n$. However, it follows from our explicit construction that there exist messages sharing the same tag with a much higher probability. Further, by exploiting this observation, we propose a forgery attack which may pose a greater threat in network applications where the number of tag verifications is unrestricted. Our existential forgery attack exploits an internal collision of the final input to a block cipher for two different messages and recovers information on the key. It also applies to messages of varying number of message blocks (namely, 2^k blocks for any $k > 2$). We also present a way to extend the attack to become a more powerful *almost universal forgery attack* using the recovered information on the key.

In addition, we show that three queries for tag generation are sufficient to forge a message for PMAC2. The possibility of such a small number of queries for an attack certainly merits our attention as it requires far fewer queries than that suggested by the birthday attack. We thus further explore the feasibility of our attack by investigating the total number of message blocks in our queries. We show that this number of message blocks is influenced by the choice of the irreducible polynomial used in defining the finite field $GF(2^n)$. In particular, the number of message blocks in the messages may go *below* $2^{n/2}$ for PMAC2 if a *bad* irreducible polynomial is used.

We remark that in [18], the designer had pointed out the need to check for certain parameters when selecting an irreducible polynomial. More precisely, the designer suggested that one can use discrete-log calculations to help choose and verify that a given choice of parameters[1] (i.e., irreducible polynomial, base and indices) provides a unique representation of the constants. In this paper, we emphasize the importance of performing these verifications by showing the existence of irreducible polynomials in which the unique representation fails for number of blocks around $2^{n/2}$. Further, we provide an explicit forgery attack to demonstrate why repetitions of the constants pose a threat. This possible security threat becomes more important especially when block ciphers (i.e., Triple-DES, MISTY1 [17], HIGHT [10] and PRESENT [6]) with a 64-bit block-size remain widely used in practice.

1.4 Organization

In Section 2, we review the notations used throughout the paper and the security definition of a MAC. In Section 3, we present the specifications of PMAC1 and PMAC2 in detail. In Section 4, we exploit the structure of the gray code to increase the forgery probability of a message and thereby, construct some forgery attacks against PMAC1. We then present a chosen message attack on PMAC2 in Section 5 by forging a tag for a specific message that requires only three queries. To study the feasibility of this attack, we investigate the effect of our choice of irreducible polynomials on the total number of message blocks

[1] Refer to [18] for more details

in our queries. In Section 6, we discuss the implications of our observations and results. Finally, we round up the paper with some concluding remarks in Section 7.

2 Preliminaries

2.1 Notation

Throughout this paper, the elements of the finite field $GF(2^n)$ will be represented either by their polynomial representation or by the corresponding n-bit binary string of coefficients. Here, we fix a basis $1, x, \ldots, x^{n-1}$ for the vector space $GF(2^n)$ over $GF(2)$. For example, $x = 000\ldots10$.

In this paper, the following notations are adopted.

$\mathrm{ntz}(i)$	The number of trailing 0-bits in the binary representation of i. For example, $\mathrm{ntz}(7) = 0$ and $\mathrm{ntz}(8) = 3$.
n	The block size.
$\|i\|$	The length of i in bits.
0^i	The bit string that consists of i '0' bits.
$\mathrm{pad}(i)$	Given a string $i \in \{0,1\}^*$ and $\|i\| < n$, the padding of i is the string $i10^{n-\|i\|-1}$. If $\|i\| = n,\ \mathrm{pad}(i) = i$.
$E_K(x)$	The encryption of input message x using a block cipher E under a k-bit K.
τ	The tag length is an integer $\tau \in [1..n]$. For simplicity, we denote $\tau = n$ throughout the paper.
$\lceil i \rceil$	The ceiling function of a real number i.
$\cdot$	The multiplication in a finite field $GF(2^n)$.
$\oplus$	The exclusive-or (xor) operation.
$\lll$	The left shift operation.
$\|\|$	The concatenation operation.

2.2 Security Definition of MAC

A MAC algorithm [9, 7] consists of three algorithms with associated key space $\mathcal{K}$, message space $\mathcal{M}$ and tag space $\mathcal{T}$:

1. Key generation (KGEN): a randomized algorithm that generates a secret key K.

2. Tag generation (TAG): an algorithm $\mathrm{TAG}_K(M)$ which takes the inputs a message M and a key K to generate an n-bit tag Tag_M.

3. Tag verification (VRFY): an algorithm $\mathrm{VRFY}_K(M, Tag_M)$ which takes the inputs a message M, a tag Tag_M and a key K to generate an answer true or false (1/0).

The key K is shared between the sender and the receiver. When the sender wants to send a message M, he computes a tag $Tag_M = \text{TAG}_K(M)$ and transmits the (M, Tag_M) pair to the receiver. Finally, the receiver checks the validity of the (M, Tag_M) pair. If the (M, Tag_M) pair is valid, an answer true (i.e., 1) will be generated; else, an answer false (i.e., 0) will be generated.

Security threats to a MAC algorithm include key recovery attack and forgery attack [13]. For the key recovery attack, an attacker tries to recover the secret key K from a number of (message, tag) pairs. Obviously, a successful key recovery attack leads to the construction of an arbitrary number of forgeries. On the other hand, a forgery attack can be divided into three types, i.e., existential forgery, selective forgery and universal forgery. In this paper, we are primarily concerned with the existential forgery attack and this leads us to the following notion of security called existential unforgeability under chosen message and chosen verification queries attack against MACs [7]. We briefly desribe this notion and readers can refer to [7] for more details.

Existential Unforgeability. Consider a MAC: $\mathcal{K} \times \mathcal{M} \rightarrow \mathcal{T}$. The attacker can make queries to two different oracles, namely $\text{TAG}_K(\cdot)$ and $\text{VRFY}_K(\cdot, \cdot)$. $\text{TAG}_K(\cdot)$ is an oracle that allows the attacker to obtain tags of messages of his choice. We name $\text{TAG}_K(\cdot)$ the tagging oracle and queries to $\text{TAG}_K(\cdot)$ the tagging queries. On the other hand, $\text{VFRY}_K(\cdot, \cdot)$ is an oracle that allows the attacker to verify the validity of (message, tag) pairs of his choice. We name $\text{VFRY}_K(\cdot, \cdot)$ the verification oracle and queries to $\text{VFRY}_K(\cdot, \cdot)$ the verification queries.

The query complexity includes the total number of queries q (i.e., the sum of the tagging queries and the verification queries), the total number of message blocks in queries σ (the number of message blocks in a message M is $\lceil |M|/n \rceil$ where n denotes the block size) and the maximum number of message blocks in queries l.

The security game proceeds as follows:

1. The challenger runs KGEN to select a secret key K randomly.
2. Query Phase: The attacker $\mathcal{A}$ is given access to two oracles, $\text{TAG}_K(\cdot)$ and $\text{VRFY}_K(\cdot, \cdot)$. For $1 \leq i \leq q$, the challenger returns $\text{TAG}_K(M_i) = Tag_{M_i}$ for tagging query or $\text{VFRY}_K(M_i, Tag_{M_i}) = 1/0$ for verification query. The queries are adaptive where $\mathcal{A}$ can view the response of the previous queries before sending the next query.
3. Forgery Attempt: The attacker $\mathcal{A}$ generates a forgery $Tag_{M'}$ on M'.

An attacker wins the game if

- $\text{VFRY}_K(M', Tag_{M'}) = 1$ and
- the attacker had not queried $\text{TAG}_K(\cdot)$ with the message M'.

Remark [13]: Universal forgery is a more powerful/ stronger adversarial goal than existential forgery as an attacker is able to find a tag for any message.

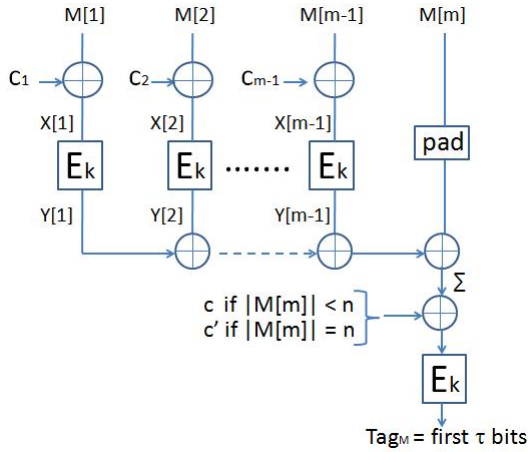

Figure 1: Overall Structure of PMAC

For existential forgery attack, the forgery message may not have any particular meaning. An attack is considered as an almost universal forgery attack when an attacker is able to find a tag for *almost* all the messages.

3 Specification

In the design of both PMAC1 and PMAC2, operations in the Galois field $GF(2^n)$ are involved. These fields are constructed as the quotient of the ring $GF(2)[x]$ modulo the ideal generated by a fixed monic irreducible polynomial $p_n(x)$ of degree n with binary coefficients. By viewing two binary n-bit strings as elements of $GF(2^n)$, the result of their product (as an n-bit binary string) often depends on the polynomial $p_n(x)$ that defines $GF(2^n)$. For the design of PMAC, Rogaway [19] chose the lexicographically first polynomial among the irreducible degree n polynomials having a minimum number of coefficients. Here is the list of irreducible polynomials suggested for some parameters of n:

- For $n = 64$, $p_{64}(x) = x^{64} + x^4 + x^3 + x + 1$.
- For $n = 96$, $p_{96}(x) = x^{96} + x^{10} + x^9 + x^6 + 1$.
- For $n = 128$, $p_{128}(x) = x^{128} + x^7 + x^2 + x + 1$.
- For $n = 160$, $p_{160}(x) = x^{160} + x^5 + x^3 + x^2 + 1$.

The general structure of the PMAC schemes consists of three algorithms as depicted in Figure 1:

1. KGEN:

- Choose a secret key $K \in \mathcal{K}$ randomly for a block cipher E.
- Share the secret key K between a sender and a receiver.
- Both parties compute $L = E_K(0^n)$ where L denotes the encryption of the message with the value 0 using a block cipher E under the key K.

2. TAG: Given a message M, let $m = \lceil |M|/n \rceil$ where m denotes the number of message blocks in message M .
   - If $|M| > n2^n$, return 0^τ.
   - Partition M into $M[1] \dots M[m]$.
   - Set $\sum = 0$.
   - For $i = 1$ to $m - 1$, do:
     - Compute $X[i] = M[i] \oplus c_i \cdot L$.
     - Compute $Y[i] = E_K(X[i])$.
     - Compute $\sum = \sum \oplus Y[i]$.
   - If $|M[m]| < n$, compute $X[m] = \sum \oplus \text{pad}(M[m]) \oplus c \cdot L$.
   - If $|M[m]| = n$, compute $X[m] = \sum \oplus M[m] \oplus c' \cdot L$.
   - Output first τ bits of $Tag_M = E_K(X[m])$.
3. VFRY: Given a (M, Tag_M) pair, do the following:
   - Generate the tag Tag'_M of M using the TAG algorithm.
   - If $Tag'_M = Tag_M$, output 1 to indicate the message M as authentic.
   - If $Tag'_M \neq Tag_M$, output 0 to indicate the message M as inauthentic.

For simplicity, we let $\tau = n$ in this paper, that is, we mainly consider the case where no truncation is performed. Here, $c_1, \dots, c_{m-1}, c$ and c' refer to constant multipliers which we are going to describe in detail.

3.1 The Generation of Constants for PMAC1

For PMAC1, the sequence of constant multipliers is generated by successive nonzero words of the gray code. Specifically, define $c_0 = 0$ and $c_1 = 1$. For $2 \leq i \leq 2^n - 1$, $c_i = c_{i-1} \oplus (0^{n-1}1 \lll \text{ntz}(i))$. We summarize some basic properties of the constants $c_1, \dots, c_{m-1}$ below where m denotes the block length of the message.

(i) $c_1, \dots, c_{2^n-1}$ are all distinct and nonzero.

(ii) The Hamming distance of c_{i-1} and $c_i, 1 \leq i \leq 2^n - 1$ is 1.

(iii) As an integer, $c_i < 2i$ for $1 \leq i \leq 2^n - 1$.

(iv) for $1 \leq i \leq n - 1, 1 \leq j \leq 2^i, c_{2^i-j} = c_{2^i+j-1} \oplus (1 \lll i)$.

It follows easily from induction and property (iv) above that for any positive integer k with $k \leq n$, only the last k bits of $c_0, c_1, \ldots, c_{2^k-1}$ can be nonzero. Hence, when these strings are viewed as the coefficients of the binary representation of integers, each of these integers is less than 2^k. Since there are exactly 2^k nonnegative integers less than 2^k and all these strings are distinct, we conclude that $c_0, c_1, \ldots, c_{2^k-1}$ comprise all the n-bit binary strings with 0 in all the first $n-k$ bits. This leads us to the following theorem.

Theorem 3.1. *For a positive integer $k \leq n$, let $C_k = \{c_0, c_1, \ldots, c_{2^k-1}\}$. Fix an $a \in C_k$. Then the function $f : C_k \to C_k$ given by $f(y) = y \oplus a$ for all $y \in C_k$ is a bijection. In particular, the sequence $c_0 \oplus a, c_1 \oplus a, \ldots, c_{2^k-1} \oplus a$ is a rearrangement of the words in the sequence $c_0, c_1, \ldots, c_{2^k-1}$.*

Proof. Here, it suffices to prove that for each $y \in C_k, y \oplus a \in C_k$. This is clear since both y and a have 0 in the first $n-k$ bits and so $y \oplus a$ must have 0 in the first $n-k$ bits as well. □

Observe that the constant multipliers $c_1, \ldots, c_{m-1}$ are defined as binary strings which are independent of the choice of the irreducible polynomial used in the construction of $GF(2^n)$. However, the binary strings representing the constants $c_1 \cdot L, \ldots, c_{m-1} \cdot L$ may differ according to the polynomial being used. Further details on gray code and $c_i \cdot L$ computations can be found in [19, 4, 5].

Finally, we take $c = 0$ and $c' = x^{-1}$ for PMAC1. It is useful to note that these two constant multipliers do not depend on m.

3.2 The Generation of Constants for PMAC2

The constant multipliers for PMAC2 are generated using the powering-up construction. Specifically, define $c_i = x^i$ where $i \geq 1$. It is easy to see that for each $i, 1 \leq i \leq m-1, X[i] = M[i] \oplus x^i \cdot L$. Thus, if a primitive polynomial $p_n(x)$ is used in constructing $GF(2^n)$, then x will be a generator of the cyclic group of nonzero elements of $GF(2^n)$ so that all the constants $x, \ldots, x^{2^n-1}$ are distinct. Notice that in this case, both the constant multipliers c_i and the constants $c_i \cdot L$ may differ when different irreducible polynomials are used in constructing $GF(2^n)$. For PMAC2, $c = x^m(x^2+1)$ and $c' = x^m(x+1)$. Hence, unlike PMAC1, these constant multipliers are defined based on the block length of the message.

4 On the Security of PMAC1

In this section, we present some weaknesses of PMAC1 against forgery attacks. Indeed, by exploiting the properties of the gray code, especially the result of Theorem 3.1, we prove that the risk of forgery will increase as we increase the block length of the message queries.

In [8, Proof of Lemma 2, Case(D_3, D_5)], the designers claimed in its security proof that given two messages M and M', the probability that $\sum_M = \sum_{M'}$ is 2^{-n}, where $\sum_M$ and $\sum_{M'}$ represent the output $\sum$ in the tag generation

algorithm for M and M', respectively. We show that this is not necessarily the case and in fact, there exist messages M and M' in which the probability is much higher.

To this end, we first observe that the tag generation for a one-block message allows us to find the encryption of any n-bit message $M_0 \notin \{0^n, 10^{n-1}\}$. More precisely, let M_0' be such that $\mathrm{pad}(M_0') = M_0$. Clearly, M_0' exists since $M_0 \neq 0^n$ or 10^{n-1}. Request for the tag for M_0', $Tag_{M_0'}$. We have $Tag_{M_0'} = E_K(\mathrm{pad}(M_0') \oplus 0 \cdot L) = E_K(M_0)$.

The main result of our section is the next theorem.

Theorem 4.1. *Let k be a positive integer with $k \leq n-1$. Fix M_1 to be a one-block n-bit message with $M_1 \notin \{0^n, 10^{n-1}\}$. Consider a message M with 2^k blocks such that each of the first $2^k - 1$ blocks is equal to M_1 and the last block is $E_K(M_1)$. Randomly pick another one-block n-bit message M_2 with $M_2 \notin \{M_1, 0^n, 10^{n-1}\}$ and let M' be a message consisting of 2^k blocks such that each of the first $2^k - 1$ blocks is M_2 and the final block is $E_K(M_2)$. Then the probability of $Tag_M = Tag_{M'}$ is at least $\frac{2^k-3}{2^n}$, where Tag_M and $Tag_{M'}$ refer to the tags of M and M', respectively.*

Proof. Observe that $Tag_M = E_K(\bigoplus_{i=1}^{2^k-1} E_K(M[i] \oplus c_i \cdot L) \oplus E_K(M_1) \oplus c' \cdot L) = E_K(\bigoplus_{i=0}^{2^k-1} E_K(M_1 \oplus c_i \cdot L) \oplus c' \cdot L)$. Similarly, $Tag_{M'} = E_K(\bigoplus_{i=0}^{2^k-1} E_K(M_2 \oplus c_i \cdot L) \oplus c' \cdot L)$. Suppose that $Z = M_1 \oplus M_2 \in \{c_0 \cdot L, c_1 \cdot L, \ldots, c_{2^k-1} \cdot L\}$. Write $Z = a \cdot L$ with $a = c_i$ for some $i = 0, \ldots 2^k - 1$. Thus

$$\begin{aligned} Tag_{M'} &= E_K(\bigoplus_{i=0}^{2^k-1} E_K(M_1 \oplus Z \oplus c_i \cdot L) \oplus c' \cdot L) \\ &= E_K(\bigoplus_{i=0}^{2^k-1} E_K(M_1 \oplus (a \oplus c_i) \cdot L) \oplus c' \cdot L) \\ &= E_K(\bigoplus_{i=0}^{2^k-1} E_K(M_1 \oplus c_i \cdot L) \oplus c' \cdot L). \end{aligned}$$

Here, the last equality follows from Theorem 3.1 since $a + c_0, \ldots, a + c_{2^k-1}$ is just a rearrangement of $c_0, \ldots, c_{2^k-1}$. Since the probability that $M_1 \oplus M_2 \in \{c_0 \cdot L, \ldots, c_{2^k-1} \cdot L\}$ is at most $\frac{2^k-3}{2^n}$, (since $M_2 \neq M_1, 0^n, 10^{n-1}$), the desired result follows. □

Clearly, when $k > 2$, $\frac{2^k-3}{2^n} > \frac{1}{2^n}$. Thus, we have shown that there exist messages for which the probability of the messages sharing the same tag is greater than 2^{-n}, contradicting the claim made in [8, Proof of Lemma 2, Case(D_3, D_5)] which essentially asserts that tags of any two messages collide with a probability of $1/2^n$. Here, we have shown that this probability can be influenced by the block lengths of the messages involved. Notice that this claimed bound is

used in the computation of the MM-collision probability in the specified lemma. However, since the MM-collision probability includes collision probabilities between the blocks of one message with blocks of the second message (apart from collisions of the tags), it is likely that the tags will be random when the internal blocks collide with the highest possible probability. As such, we believe that the proof of the MM-collision bound (and thus, the security bound) remains valid even though the claims made in cases D_3 and D_5 are ambiguous. Nonetheless, we show that this higher tag collision probability can be exploited in various ways as follows.

First, it can be easily deduced that the number of different tags of messages taking the form in the preceding theorem is at most 2^{n-k}. Thus, applying the birthday paradox arguments imply that we can expect to attain a tag collision after around $O(2^{(n-k)/2})$ tagging queries. Note that in this case, the total block length $\sigma = O(2^{(n+k)/2})$. Observe that when a collision occurs, say between M and M' as defined in Theorem 4.1, we may conclude that $M_1 \oplus M_2 = c_i \cdot L$ for some $i = 1, \ldots, 2^k - 1$. By exhaustively trying each of these c_i's, L will be recovered. More importantly, we can construct an almost universal forgery attack using this information of L. More details on almost universal forgery attack will be given below.

In some real world applications, it may be more feasible to perform a large number of tag verifications rather than tagging queries [14]. Under such a scenario, Theorem 4.1 allows us to devise the following forgery attack on PMAC1.

1. Fix a positive integer k so that it is feasible to obtain the tag for a message with block length 2^k.
2. Randomly pick an n-bit block $M_1 \neq 0^n, 10^{n-1}$.
3. Obtain the encryption $E_K(M_1)$ via a tagging query.
4. Construct a 2^k-block message $M = M_1||M_1|| \ldots ||M_1||E_K(M_1)$.
5. Query the oracle for the tag of M, denoted by Tag_M.
6. Randomly pick another n-bit block $M_2 \neq 0^n, 10^{n-1}$.
7. Obtain the encryption $E_K(M_2)$ via a tagging query.
8. Construct a 2^k-block message $M' = M_2||M_2|| \ldots ||M_2||E_K(M_2)$.
9. Send the pair (M', Tag_M) for verification.

Just as before, once a forgery is successful, it follows that $M_1 \oplus M_2 = c_i \cdot L$ for some $i = 1, \ldots, 2^k - 1$. Similarly, testing out each of these possibilities will now reveal the value of L with the complexity of $O(2^k)$.

By Theorem 4.1, in this attack, the number of tag verifications required is expected to be $O(2^{n-k})$. Moreover, we can lower the number of verifications needed by increasing the number of tagging queries. In general, given t tags, we expect that the number of verifications required to obtain a forgery is $O(2^{n-k}/t)$.

This may be too large to be much of a threat in practice but it highlights a potential weakness of PMAC1 in an environment which allows for unlimited tag verifications. For some applications, it is not common to maintain a counter to keep track of the number of verification queries made.

Extension: An Almost Universal Forgery Attack. Next, we discuss how knowledge of L leads to an almost universal forgery attack, namely, apart from a few exceptions, we wish to find the tag of any message M without passing M through the tagging oracle. Here, we describe one such procedure. Let M denote a given message. Decompose M into $M = M[1]||\dots||M[m]$. Suppose that $M[m] \neq x^{-1} \cdot L$ or $10^{n-1} \oplus x^{-1} \cdot L$. We consider two cases.

- **Case 1**: $|M[m]| < n$

  1. Determine $M[m]' = \text{pad}(M[m]) \oplus x^{-1} \cdot L$.
  2. Request the tag of $M' = M[1]||M[2]||\dots||M[m-1]||M[m]'$. Notice that $M' \neq M$.
  3. Output $Tag_{M'}$ as the tag of M.

  We see that this attack works since $Tag_{M'} = E_K(\bigoplus_{i=1}^{m-1} E_K(M[i] \oplus c_i \cdot L) \oplus M[m]') = E_K(\bigoplus_{i=1}^{m-1} E_K(M[i] \oplus c_i \cdot L) \oplus \text{pad}(M[m]) \oplus x^{-1} \cdot L) = Tag_M$.

- **Case 2**: $|M[m]| = n$

  1. Determine $M[m]' = M[m] \oplus x^{-1} \cdot L$.
  2. Find $M[m]''$ where $\text{pad}(M[m]'') = M[m]'$. Notice that $|M[m]''| < n$ and $M[m]''$ exists because $M[m]' \neq 0^n, 10^{n-1}$.
  3. Request the tag of $M' = M[1]||M[2]||\dots||M[m-1]||M[m]''$. Observe that $M' \neq M$.
  4. Output $Tag_{M'}$ as the tag of M.

  Similar to case 1, it is straightforward to verify that this attack works.

We may use more queries to forge the tag of our exceptional cases but we leave interested readers to work out the details. Notice that this procedure can be applied to the birthday attack proposed in [16] as well.

Finally, we remark that all the attacks in this section are general in the sense that they can be applied to messages of varying block lengths, namely 2^k blocks, for any positive integer k. More precisely, observe that the birthday attacks of [16] require $q = O(2^{n/2})$ queries in which the maximum length of each query is $l = O(1)$. On the other hand, in the worst-case scenario when a "bad irreducible polynomial" is used in the construction of the finite field $GF(2^n)$, our analysis in the next section shows that PMAC2 tags can be forged using $q = O(1)$ queries and $l = O(2^{n/2})$. As such, our attacks on PMAC1 can be viewed as a bridge between these two extremes, since they enable us to forge PMAC1 tags with $l = O(2^k)$ and $q = O(2^{(n-k)/2)})$ for any integer k between 1 and $n/2$.

5 On the Security of PMAC2

We now turn our attention to the security of PMAC2 against existential forgery attacks. By considering the tag generation algorithm for PMAC, it is apparent that the security of such construction greatly depends on the constants c_i used to mask the individual blocks. For instance, if there exist two identical masks for two different blocks of message, an attacker may exploit such a scenario to construct a forgery by swapping the messages in these two different blocks as both messages will generate a same tag (resulting in a collision). This holds true since the addition operation in $GF(2^n)$ (or equivalently, the exclusive-or operation) is commutative.

However, the constants $c_i = x^i, 1 \leq i \leq m-1$ will all be distinct for PMAC2 as long as $m \leq 2^n - 1$. Nonetheless, as the constants used in different messages are the same, we now show how a one-block message can be exploited to construct collisions of messages.

Recall that for PMAC2, the last constants (i.e., c or c') are dependent on the block length of the message involved. Unless otherwise specified, we denote this last constant by c_m for a message with block length m.

Let M_1 be any one-block message so that $m = 1$. By our construction, the tag for M_1 is $Tag_{M_1} = E_K(\text{pad}(M_1) \oplus d \cdot L)$, where $d = x^3 + x$ or $x^2 + x$. Now, suppose that there exists some $a, a \geq 1$ for which $c_a = d$. Define a message $M = M[1]||M[2]||\ldots||M[a]||M[a+1]$, where $M[1], \ldots, M[a-1]$ are arbitrary, $M[a] = \text{pad}(M_1), M[a+1] = Tag_{M_1} \oplus R$, for some arbitrary n-bit R. Hence,

$$\begin{aligned} Tag_M &= E_K(\bigoplus_{i=1}^{a} E_K(M[i] \oplus c_i \cdot L) \oplus Tag_{M_1} \oplus R \oplus c_{a+1} \cdot L) \\ &= E_K(\bigoplus_{i=1}^{a-1} E_K(M[i] \oplus c_i \cdot L) \oplus E_K(M[a] \oplus c_a \cdot L) \oplus Tag_{M_1} \\ &\quad \oplus R \oplus c_{a+1} \cdot L) \\ &= E_K(\bigoplus_{i=1}^{a-1} E_K(M[i] \oplus c_i \cdot L) \oplus R \oplus c_{a+1} \cdot L). \end{aligned} \quad (1)$$

Since the constants $c_1, \ldots, c_{2^n-1}$ are all distinct, there must certainly exist some $a, 1 \leq a \leq 2^n - 1$ for which $c_a = d$. Since M_1 is arbitrary in the derivation of Tag_M in Equation 1, we now show that three queries are sufficient to launch a forgery attack on PMAC2.

In this present situation, $d = x^2 + x$ or $x^3 + x$, depending on the length of the block M_1. Denote by a and b the integers for which $x^a = x^2 + x$ and $x^b = x^3 + x$, respectively. (More details on a and b will be given in the next section.)

- Let M_1 and M_1' be two k-bit messages, where $k = n$ if $a < b$ and $k < n$ if $a > b$.

- Query the tagging orcle to obtain the tags Tag_{M_1} and $Tag_{M_1'}$ for the messages M_1 and M_1', respectively.
- Fix R to be an n-bit string. Construct a message $M = M[1]||\ldots||M[a-1]||pad(M_1)||M_2$, where $M_2 = Tag_{M_1} \oplus R$ and $M[1], \ldots, M[a-1]$ are arbitrary.
- Query the tagging oracle to obtain Tag_M, the tag for M.
- Construct another message $M' = M[1]||\ldots||M[a-1]||pad(M_1')||M_2'$, where $M_2' = Tag_{M_1'} \oplus R$.
- Then the tag $Tag_{M'}$ for M' is identical to Tag_M.

It is clear that both Tag_M and $Tag_{M'}$ are identical since M_1 and M_1' are not involved in the expression for Tag_M (and thus $Tag_{M'}$) in Equation 1. Here, 3 queries are required and the total block length of our queries is $\min(a, b) + 3$. Further, we remark that $i \geq 1$ forgeries can be constructed using the above attack with total queries, $q = i+2$ and total number of blocks, $\sigma = i \times \min(a, b) + 3$.

5.1 Determining the Size of a and b

Evidently, the feasibility of our preceding attack is greatly influenced by the size of a or b. In [18], a fix primitive irreducible polynomial was suggested for $n = 128$ and the values of a and b were shown to be extremely large for the specified polynomial. Hence, the authors have proposed parameters to guard against our attack.

In this section, we attempt to determine a and b under different irreducible polynomials in order to investigate how large or small they may be. Our analysis shows that the choice of the underlying irreducible polynomial greatly influences the range of these values. As such, care must be taken to check for the values of a and b when new parameters are set for the scheme.

According to our attack on PMAC2, we need the values of a and b for which $x^a = x^2 + x$ and $x^b = x^3 + x$. Now, let l be such that $x^l = x + 1$. Since $x^{l+1} = x^2 + x$ and $x^{2l+1} = x(x+1)^2 = x^3 + x$, we have $a = l + 1 \mod 2^n - 1$ and $b = 2l + 1 \mod 2^n - 1$. As such, we concentrate on finding l satisfying $x^l = x + 1$.

In fact, this is the well-known discrete logarithm problem in the field $GF(2^n)$ which can be solved using the index-calculus method with a worst-case sub-exponential running time. For $n \leq 256$, these computations can be carried out within hours.

Recall that a primitive irreducible polynomial needs to be specified in constructing the field $GF(2^n)$. Although different polynomials give rise to isomorphic finite fields $GF(2^n)$, the corresponding value of l (satisfying $x^l = x + 1$) may vary over all the primitive irreducible polynomials of degree n with binary coefficients. To illustrate, we list down the values of l for the various primitive irreducible polynomials of degree 8 in Table 1.

Table 1: The values of l for the various primitive irreducible polynomials of degree 8

Primitive Irreducible Polynomial	l
$x^8+x^6+x^5+x^2+1$	233
$x^8+x^7+x^2+x+1$	99
$x^8+x^6+x^4+x^3+x^2+x+1$	122
$x^8+x^6+x^5+x+1$	197
$x^8+x^7+x^6+x^3+x^2+x+1$	141
$x^8+x^6+x^3+x^2+1$	23
$x^8+x^4+x^3+x^2+1$	25
$x^8+x^7+x^3+x^2+1$	59
$x^8+x^6+x^5+x^3+1$	16
$x^8+x^7+x^6+x^5+x^4+x^2+1$	134
$x^8+x^7+x^5+x^3+1$	13
$x^8+x^5+x^3+x^2+1$	240
$x^8+x^7+x^6+x+1$	157
$x^8+x^6+x^5+x^4+1$	231
$x^8+x^7+x^6+x^5+x^2+x+1$	115
$x^8+x^5+x^3+x+1$	243

Observe that the value of l ranges from 13 to 243. For $l = 13, a = 14$ and $b = 27$ which implies that $a < 2^{n/2} = 16$. Our next result shows that for any even n, there exists an irreducible polynomial of degree n such that $l = 2^{n/2}$. For more properties on finite fields which are required for the proof, we refer the reader to [20].

Theorem 5.1. *Let n be even and write $n = 2n_0$. There exists a monic irreducible polynomial of degree n, denoted by $p(x)$, such that $x^{2^{n_0}} = x + 1$.*

Proof. Let $f(x) = x^{2^{n_0}} + x + 1$. We first show that $f(x)|(x^{2^n} + x)$. Since $x^{2^{n_0}} = x+1$, 2^{n_0} repeated squarings then yields $x^{2^n} = x^{2^{n_0}} + 1 = x+1+1 = x$. Hence, it follows that $f(x)|(x^{2^n} + x)$. This means that every root of $f(x)$ is an element in the field $GF(2^n)$. These roots are not elements of $GF(2^{n_0})$ since $\gcd(f(x), x^{2^{n_0}} + x) = 1$. Moreover, as the derivative of $f(x)$ is $1, f(x)$ has no repeated roots, that is, it has 2^{n_0} distinct roots which lie in $GF(2^n)\backslash GF(2^{n_0})$. Hence, there must exist some root β of $f(x)$ which does not lie in any subfield of $GF(2^n)$. Let $p(x)$ be the minimal polynomial of β over $GF(2)$. Then, $p(x)|f(x)$ is an irreducible of degree n and $\beta^{2^{n_0}} = \beta + 1$. Our result now follows. □

We verify our results using the Magma Algebra Computational System for $n \leq 40$ and find that in fact, there exists a primitive irreducible polynomial such that Theorem 3 holds for all these values of n.

Before we conclude this section, we list down the discete log values of a and b for $n = 64, 96, 128, 160$ and their respective polynomials suggested by the

authors in Table 2. Once again, our results are obtained within minutes with the aid of the Magma Algebra Computational System.

Table 2: Discrete log values for different irreducible polynomials

n	$p_n(x)$	$\log_2 a$	$\log_2 b$
64	$p_{64}(x) = x^{64} + x^4 + x^3 + x + 1$	63.071	59.683
96	$p_{96}(x) = x^{96} + x^{10} + x^9 + x^6 + 1$	95.674	95.253
128	$p_{128}(x) = x^{128} + x^7 + x^2 + x + 1$	127.994	127.987
160	$p_{160}(x) = x^{160} + x^5 + x^3 + x^2 + 1$	159.670	159.241

6 Discussion

So far, we have presented some possible forgery attacks on both PMAC schemes. Essentially, our attacks hinge on the following characteristics of the sequence of constants $c_1, \ldots, c_{2^n-1}, c, c'$:

- For some m, the constants $c_1, \ldots, \ldots, c_m, c, c'$ contain a large subset A satisfying the following property: There exist nonzero n-bit strings y such that $A \oplus y = \{z \oplus y : z \in A\} = A$.
- There exists some $i, 1 \leq i \leq m$ such that $c_i = c$ or $c_i = c'$.

We have seen that the gray code satisfies the first property with $A = \{c_0, c_1, \ldots, c_{2^k-1}\}$ for any positive integer k and any $y \in A$. As a result, the probability of a successful forgery will increase proportionately with the block length m. On the other hand, PMAC2 is not as vulnerable to this attack since it is not so straightforward to construct a set A and many strings y satisfying the given property for a given m.

As pointed out previously, this attack will pose a more serious threat in environment which does not limit the number of tag verifications made with a given key. Hence, we recommend that both tagging queries and verifications under the same key should be controlled to maintain the security of the PMAC schemes when the users implement the PMAC schemes.

It can be noted that both sets of constants satisfy the second property. However, as pointed out in [4], the value of i for which $c_i = x^{-1}$ will be huge (bigger than 2^{n-1}) irrespective of the irreducible polynomial being used. In view of this, we focus our attention on PMAC2 and we showed that a "bad" choice of primitive irreducible polynomial will lead to a forgery attack with total block length $\sigma = O(2^{n/2})$. We even gave some counter examples for $GF(2^8)$ in Table 1 that there exists some primitive irreducible polynomials of degree 8 that will lead to a forgery attack with total block length lesser than 2^4 and break the security bound of PMAC2.

Apart from selecting a good primitive irreducible polynomial for which the value of i is exhorbitantly large, it may be useful to restrict the block length of

the messages that can be authenticated as well. Alternatively, the tagging of a one-block message can be separately treated so as to prevent the possibility of inserting the tag into a long message to cancel out an intermediate block.

Finally, we briefly discuss the case when truncation of the tags is performed (i.e., $\tau < n$) on PMAC2. Following the same attack procedure, we can then construct a set of $2^{n-\tau}$ messages that contains a message having the same tag as one of our queries. This gives us a probability of $1/2^{n-\tau}$ of forging a valid tag with three queries for PMAC2. If messages in this set can be queried, then the last $n-\tau$ bits of the encrypted output corresponding to our one-block message can be determined, and thus, additional forgeries can be easily constructed.

7 Concluding Remarks

In this paper, we presented weaknesses of the PMAC schemes arising from two main characteristics of the constant multipliers in the construction. These weaknesses allow us to launch forgery attacks on PMAC1 as well as PMAC2. Our existential forgery attack on PMAC1 is based on a new technique which exploits an internal collision of the final input to a block cipher for two different messages and recovers the information of L (derived from the key). Once L is recovered, we used L to construct an almost universal forgery attack on PMAC1 where we can forge the tags for most of the messsages. Our verification attack and PMAC2's attack are explicit and independent of the birthday paradox arguments. Instead, our attacks exploited the structural properties of PMAC. We also showed a way in extending the attack to a more powerful almost universal forgery attack using the additional information recovered in the attacks. Even though the number of queries may be surprisingly small (especially in the case of PMAC2), the total block length is often too large to pose any direct danger to the schemes. Nonetheless, we pointed out that the attacks remain valid according to the specifications given by the authors of the schemes. More importantly, we showed that the block lengths may vary significantly for PMAC2 (which may be as low as $2^{n/2}$), depending on the choice of the irreducible polynomial used in the construction of the underlying finite field. This indicates the danger of fixing an irreducible polynomial randomly or arbitrarily. However, we emphasize that our attacks did not break the security bound of PMACs. Rather, we provided explicit attacks which achieve these bounds in some instances. Designers should limit the length of the message that can be authenticated while the communication network applications that implement PMAC1 should limit the number of verifications to maintain the validity of the security proof.

References

[1] Mihir Bellare, Ran Canetti and Hugo Krawczyk. Keying Hash Functions for Message Authentication. In *Proceedings of CRYPTO 1996*, LNCS 1109, pp. 1-15, 1996.

[2] John Black, Shai Halevi, Hugo Krawczyk, Ted Krovetz and Phillip Rogaway. UMAC: Fast and Secure Message Authentication. In *Proceedings of CRYPTO 1999*, LNCS 1666, pp. 216-233, 1999.

[3] John Black and Phillip Rogaway. CBC MACs for Arbitrary-Length Messages: The Three-Key Constructions. In *Proceedings of CRYPTO 2000*, LNCS 1880, pp. 197-215, 2000.

[4] John Black and Phillip Rogaway. A Block-Cipher Mode of Operation for Parallelizable Message Authentication. In *Proceedings of EUROCRYPT 2002*, LNCS 2332, pp. 384-397, 2002.

[5] John Black and Phillip Rogaway. Full version: A Block-Cipher Mode of Operation for Parallelizable Message Authentication. Available at http://www.cs.ucdavis.edu/ rogaway/ocb/pmac-doc.htm, 2002.

[6] Andrey Bogdanov, Lars R. Knudsen, Gregor Leander, Christof Paar, Axel Poschmann, Matthew J. B. Robshaw, Yannick Seurin, C. Vikkelsoe. PRESENT: An Ultra-Lightweight Block Cipher. In *Proceedings of CHES 2007*, LNCS 4727, pp. 450-466, 2007.

[7] Yevgeniy Dodis, Eike Kiltz, Krzysztof Pietrzak and Daniel Wichs. Message Authentication, Revisited. In *Proceedings of EUROCRYPT 2012*, LNCS 7237, pp. 355-374, 2012.

[8] Shai Halevi and Hugo Krawczyk. MMH: Software Message Authentication in the Gbit/Second Rates. In *Proceedings of FSE 1997*, LNCS 1997, pp. 172-189, 1997.

[9] Helena Handschuh and Bart Preneel. Key-Recovery Attacks on Universal Hash Function Based MAC Algorithms. In *Proceedings of CRYPTO 2008*, LNCS 5157, pp. 144-161, 2008.

[10] Deukjo Hong, Jaechul Sung, Seokhie Hong, Jongin Lim, Sangjin Lee, Bonseok Koo, Changhoon Lee, Donghoon Chang, Jaesang Lee, Kitae Jeong, Hyun Kim, Jongsung Kim and Seongtaek Chee. HIGHT: A New Block Cipher Suitable for Low-Resource Device. In *Proceedings of CHES 2006*, LNCS 4249, pp. 46-59, 2006.

[11] ISO/IEC 9797-1: Information technology − Security techniques − Message Authentication Codes (MACs) − Part 1: Mechanisms using a block cipher. 1999.

[12] Tetsu Iwata and Kaoru Kurosawa. OMAC: One-Key CBC MAC. In *Proceedings of FSE 2003*, LNCS 2887, pp. 129-153, 2003.

[13] Keting Jia, Xiaoyun Wang, Zheng Yuan and Guangwu Xu. Distinguishing and Second-Preimage Attacks on CBC-Like MACs. In *Proceedings of CANS 2009*, LNCS 5888, pp. 349-361, 2009.

[14] Lars R. Knudsen and Chris J. Mitchell. Partial Key Recovery Attack Against RMAC. In *Journal of Cryptology*, Vol. 18, No. 4, pp. 375-389, 2005.

[15] Kaoru Kurosawa and Tetsu Iwata. TMAC: Two-key CBC MAC. In *Proceedings of CT-RSA 2003*, LNCS 2612, pp. 33-49, 2003.

[16] Changhoon Lee, Jongsung Kim, Jaechul Sung, Seokhie Hong and Sangjin Lee. Forgery and Key Recovery Attacks on PMAC and Mitchell's TMAC Variant. In *Proceedings of ACISP 2006*, LNCS 4058, pp. 421-431, 2006.

[17] Mitsuru Matsui. New Block Encryption Algorithm MISTY. In *Proceedings of FSE 1997*, LNCS 1267, pp. 54-68, 1997.

[18] Phillip Rogaway. Efficient Instantiations of Tweakable Blockciphers and Refinements to Modes OCB and PMAC. In *Proceedings of ASIACRYPT 2004*, LNCS 3329, pp. 16-31, 2004.

[19] Phillip Rogaway. PMAC: Proposal to NIST for a parallelizable message authentication code. Available at http://www.cs.ucdavis.edu/ rogaway/ocb/pmac.pdf, 2001.

[20] Rudolf Lidl and Harald Niederreiter. Finite Fields. Encyclopedia of Math. and Its Appl., Vol. 20, Addison-Wesley Publ. Co., Reading, Mass., 1983, xx + 755 pp; now distributed by Cambridge University Press.