Nomenclature

$\oplus$ Bitwise exclusive-xor operation(XOR)

(C_{fake},T_{fake}) The frame that the adversary used to replace (C_{origin},T_{origin})

(C_{origin}, T_{origin}) The valid memory frame that the adversary wants to attack

(M,T) Message M and its tag T

$\{0,1\}^n$ Set of all n-bit binary strings

Adv_{F1}^{F0} The probability for an adversary to distinguish between two functions F0 and F1

Bit Index The location of a bit in a string. Ex. in big ending the lsb bit has index 0 and n-1 for msb bit

Block Length The number of bits of a block

$E_k(M)$ Encrypt input M with secret key k

F_{MAC} The probability that Forgery(MAC, A)=1 for an adversary A and a MAC scheme

Forgery(MAC, A) Forgery experiment on a MAC scheme conducted by the adversary A. Return 1 if A succeed a forgery attack otherwise 0

$Func_k(M)$ Function F processing input M with secret information k

GF-mult(A,B) Galois Field multiplication with A and B

Len(M) The number of bits that string M consist of

$Len(M)_l$ The number of l-bits blocks that string M consist of

MAC Message Authentication Code: The algorithm to generate tags

Pr[A] The probability that event A happens

PRF Pseudo-random Function

PRP Pseudo-random Permutation

Tag A short message block used to protect the integrity of plaintext or ciphertext

TG(M, ...) Tag generation stage adopting a MAC scheme. The input of TG is message protected and the parameters in the argument list '...'

UF-CMA Unforgeability under adaptive Chosen Message Attack

VF(M,T, ...) Message-Tag pair verification stage. Verify the message M and tag T with the parameters in the arguments list '...'. Return 1 if TG(M,...) = T and 0 if not.

1 Notations and Definitions

In this part we firstly depict the notations of symbols used to describe Cost-Effective Tag Design(CETD) and the MAC scheme adopted, CETD-MAC. Then we present the security definition of MAC schemes in computational scenario.

1.1 Notations of Symbols

Let $\{0,1\}^n$ be the set of all n-bit binary strings. The set of all binary string is expressed as $\{0,1\}^*$. For a string $M \in \{0,1\}^n$, Len(M) is its length in bits, and $\text{Len(M)}_l = \lceil \text{Len(M)/l} \rceil$ is the number of l-bits blocks M consist of. Let 0^l and 1^l denote l-bits strings of all zeros and all ones. For a bit string M and an integer l that $\text{Len(M)} \geq l$, $\text{msb}_l(M)$ denotes the most significant l bits(left most l bits) of M and $\text{lsb}_l(M)$ for least significant l bits(right most l bits) of M. For two bit string X and Y, we denote $X\|Y$ or XY as the their concatenation. For bit string M whose length in bits is multiple of integer l, we denote X parted into l-bit sub-strings as $X = (X[1]X[2]...X[n])_l$, where X[1], X[2], ..., $X[n] \in \{0,1\}^l$.

The block cipher encryption of a string X with a secret key K is denoted as $E_K(X)$. $E_K(X)$ expresses the String mapping of $\{0,1\}^n \rightarrow \{0,1\}^n$ where n is the Len(X) and Len(output).

1.2 Security Definition of Memory Integrity Protection Designs

Threat Model The attacks discussed in this article mainly situated on the probing and tampering the data transferred between chip and external memory. In this scenario, the adversary can conduct types of attacks on the (ciphertext, tag) pair stored on the external memory or in the transformation between chip and memory. We call these two types of integrity attack content modification attack and copy-then-replay attack.

When conducting a content modification attack, then adversary modifies the content of a (ciphertext, tag) pair, (C_{origin}, T_{origin}). At least one of ciphertext and tag in the modified memory frame (C_{fake}, T_{fake}) are distinct to the original one sent from the chip. When the chip reads the modified pair (C_{fake}, T_{fake}), this pair is verified by VF(C_{fake}, T_{fake}, ...). If VF(C_{fake}, T_{fake}, ...)=1, then the modified pair passes and the content modifying attack succeeds.

For a copy-then-replay attack, we make the following denotions first:

- A_{origin}: The address of the external memory frame M attacked
- A_{copy}: The address of the external memory frame M1 which is copied by the adversary as the fake frame
- TS_{origin}: The generation time stamp of the external memory frame M attacked
- TS_{copy}: The generation time stamp of the external memory frame M2. M2 and M has same address while TS_{M2} is previous of TS_M

During a copy-then-replay attack on a memory frame M whose address is A_{origin} and time stamp is TS_{origin}, the adversary A replace M with a copy. This copy can be a frame from other address A_{copy} or one at same address but copied at an old time stamp TS_{copy}. The copy is a valid pair from the chip. When the chip reads the modified frame (C_{fake}, T_{fake}), the fake pair(C_{fake}, T_{fake}) is verified by VF(C_{fake}, T_{fake}, ...) and the copying-then-replying attack succeeds if VF(C_{fake}, T_{fake}, ...)=1.

If the fake pair from the adversary passes the verification, the integrity of the data is broken. To defend content modification attack, the MAC scheme should ensure that when the secret information used in tag generation collides for two distinct input data, the related two tags should have low probability to be identical. For copying-then-replaying attack, identical input data should have low probability to lead to tag collision if the secret information used in tag generation are distinct.

The Security Analysis of Memory Integrity Protection Design

CETD meets the security requirement of CETD-MAC To defend copy-then-replay attack, CETD-MAC requires that when the chip writes the content of a memory frame in an address, the nonce used to generate the tag should be randomly chosen. As the nonce is generated by encrypting nonce-input, the memory authentication system based on CETD-MAC should ensure that for each time of memory writing, the nonce-input is distinct.

In CETD tag design, a input of nonce generation is a tuple(addr,ctr, rnd). The meaning of each element in the tuple is listed below:

- Addr: The address of ciphertext
- Ctr: A incrementing counter
- Rnd: A random number

When the adversary replaces the memory frame with his copy frame from same address but older time point, the counter part in the nonce-input of this copy is smaller than the counter part in the nonce-input of the valid frame. This difference of counter part ensure that the two nonce-inputs are distinct. On the other hand, when the adversary tries to replace the frame with a copy from another memory address, the address part for two nonce-inputs will different and the two nonce-inputs will be distinct. We can see that Cost-Effective Tag Design can meet the security requirements of CETD-MAC.

1.3 Security Definition of MAC Schemes

In this part we depict the security definition of MAC schemes in computational model.

Chosen Message Attack The brute-force attack on the integrity of memory frame can be expressed as the following steps:

- Randomly pick a memory frame
- Keep the ciphertext(or the tag) fixed, then tranverse all the possible tag(or ciphertext) until

Goldwasser et al. introduced the concept of unforgeability of chosen message attack in []. The adversary can select numbers of valid ciphertext-tag pairs, observe the relationship between each ciphertext and its tag, then send a fake ciphertext-tag pair to the verification stags based on the observation.The adversary A1 conducting chosen-message attack is more strategical compare with the one A2 for brute-fore attack in content modification scenario, which means A1 can try less number of ciphertext-tag pairs to pass the verification. For instance, if there is a type of relationship between ciphertext and tag for a MAC scheme, A1 will realise this relationship and conduct two types of attack with high probability of passing: just modifying ciphertext from C1 to C2 and C2 has high probability to generate same tag T1; or send new pair (C2,T2) to the verification where C2 and T2 meet the relationship. In A1‘s eyes, for given ciphetext C and a unknown nonce N, the tags in the tag domain has different probability to pass the verification while the probability is identical to A2.

In this article, we assume the adversary can choose the ciphertext in both content modification and copy-then-replay attack. The adversary aims to find groups of inputs that the two inputs from same group has probability of tag collision that is much larger than $1/2^n$, where n is Len(tag).

MAC Security and proposition 2.7 The security of a MAC scheme is measured by the probability that the forger makes Forgery Experiment returns 1. In this Forgery Experiment, the adversary conducts chosen-message attack on the tag generation oracle and observes the tags. He then made a fake (C,T) pair and sends C to the tag generation oracle, is the results tag T is identical to fake one, the FE returns 0, else return 1.

To make FE return 1, the adversary should be sure that for the given tag generation oracle, the probability that fake tag is identical to T. This fact can be modeled as MAC(C, ...) = T. To ensure that the probability that FE returns 1 is low, the MAC scheme should ensure that Pr[MAC(C,...)=T] is low. This fact make the proposition 2.7 valid, which gives the assertion that if the MAC is prf then it‘s a secure MAC.

In the computational model based security analysis on MAC schemes, a secure MAC scheme should ensure that for any two input messages in the domain, the probability that their tags collide should be no more than an upper bound. This definition of secure MAC scheme is widely accepted and has been adopted in the security analysis of iterated MAC schemes such as CBC-MAC [] and its variants [], and parallel MAC schemes such as PMAC [] and its variants[]. This definition emphasizes that when the key used in the MAC scheme is unaccessible to the adversary, the adversary cannot find any relationship between a given

tag and a given input message. In the eyes of the adversary, a given tag can be generated by any message in the input domain and for a given input message, its tag can be any one value in the tag range.

Distinguish two functions The function in an oracle is chosen from MAC or PRF by coin flipping. If the MAC behaves like a prf, then the outputs from MAC scheme should have all the properties of PRF.

Security Analysis Model In our security analysis of CETD-MAC scheme, the adversary A has access to the tag generation scheme and verification scheme. A can send chosen ciphertexts to the tag generation scheme and determine whether the nonce for his chosen ciphertexts are fixed or not. He can acquire the value of the tag for each his chosen ciphertext while cannot access the value of the related nonce. After analyzing several valid ciphertext-tag pairs, A sends his fake pair (C_{fake}, T_{fake}) to the verification scheme. The verification scheme generates the tag T of C_{fake} and the fake pair passes the verification if T is identical to T_{fake}. Our security analysis on CETD-MAC is modeled like this:

- For content-modification attack, the adversary A fixes the nonce and inputs distinct ciphertexts to the tag generation scheme.
- For copying-then-replaying attack, the value of ciphertexts are fixed to a value C chosen by A. A inputs C continually and as the tag generation scheme to maintain the nonce distinct for each input.

After several rounds of inputing, the adversary analyses the value of tags for his chosen inputs then sends his fake pair(C_{fake}, T_{fake}) to the verification scheme.

Content Modification Scenario In content modification scenario, the nonce value is fixed and kept in the trusted area and is unaccessible to the adversary. The adversary tries to replace the valid memory frame (C_{origin}, T_{origin}) with his fake pair (C_{fake}, T_{fake}). The forgery advantage $F_{CETD-MAC}$ under content modification attack is the probability that the adversary can get the verification oracle to accept his fake cipehrtext-tag pair.

Copy-then-Replay Scenario In copy-then-replay scenario, the encryption key is maintained unchanged while the nonce value is updated for each calling of tag generation oracle. Secret key and nonces are kept in trusted area and cannot be acquired by the adversary. The adversary can copy the message-tag pairs in old time stamp. When conducting the replay attack, the adversary replaces a memory frame containing a data-tag pair in new timestamp with a copy from his storage. The forgery advantage $F_{CETD-MAC}$ under replay attack is the probability that the adversary can get the verification oracle to accept the replace data-tag pair.

Assume the copy with old timestamp from the adversary is expressed as (M,T1) where M is the message and T1 is the corresponding tag, and the nonce

of (M,T1) is denoted as N1. When applying M to the tag generation at a new time point, we mark the corresponding tag T2 and the nonce N2. If the adversary can succeed a replay attack, then T1 is identical to T2 regardless of the equality of N1 and N2. Then $F_{CETD-MAC}$ under replay attack can be denoted as the probability Pr[T1 = T2 | Message value is M & N1,N2 are randomly generated].

2 Security Analysis of CETD-MAC

In this section we analysis the security of cost-effective tag design. Our analysis mainly focus on the CETD-MAC scheme and is based on computational model. We prove that under both content modification and copy-then-replay attack, CETD-MAC does not behave like a PRF as the its tag space for a input is no more than half of the tag domain. Based on this assertion, we prove that CETD-MAC is not a secure MAC scheme based on the security definition in computational model.

2.1 CETD-MAC Definition

In this article, we name the MAC scheme adopted in Cost-Effective Tag Design[] CETD-MAC. We briefly depict the design of CETD-MAC scheme and related notations. CETD-MAC scheme can be expressed as tag = CETD-MAC(M, nonce-input). The input arguments of CETD-MAC are message M and a tuple named nonce-input which is the concatenation of the memory address of M, a counter and a random number. The nonce-input tuple is denoted as (address‖counter‖random). The length of nonce-input, Len(nonce-input), is identical to the length of input to block cipher $E_K(X)$. The output of CETD-MAC is named tag, whose length is optional. We use Sub-BLK-No to express the value of Len(M)/Len(tag). One preliminary of CETD-MAC is that Sub-BLK-No should be no less than 2 to assure that swapping stage can work and 0s will be concatenated to the leftmost of input message M when Sub-BLK-No is not integer. The concept of CETD-MAC is expressed in Figure **??**.

We follow the definition of CETD-MAC in [**?**].

- Message is splitted to sub-blocks each of whose length is tag length
- The first stage is several rounds of bit-segment swapping. For each round , two sub-blocks are randomly chosen. Introduce shuffle parameter V[i]
- The output sub-blocks from swapping stage, named X blocks, are sent to block-level rotate shifting stage. Each X block is rotate shifted for several bits. The concept of rotate shifting is expressed in Fig 2 Introduce shift parameter S[i]
- The output sub-blocks from shifting stage, named Y blocks, are XORed to form the final tag

Assume E_K performance like a ideal random value generator, for two distinct nonce-inputs NI1 and NI2, the corresponding nonce values N1 and N2 should be random and the probability that N1 is identical to N2 should be $1/2^n$, where n is Len(N1).

CETD-MAC is secure under brute-force attack In [], the authors provided a security analysis of CETD-MAC scheme. The adversary is assumed to conduct brute force attack in which he randomly chooses a ciphtertext C and

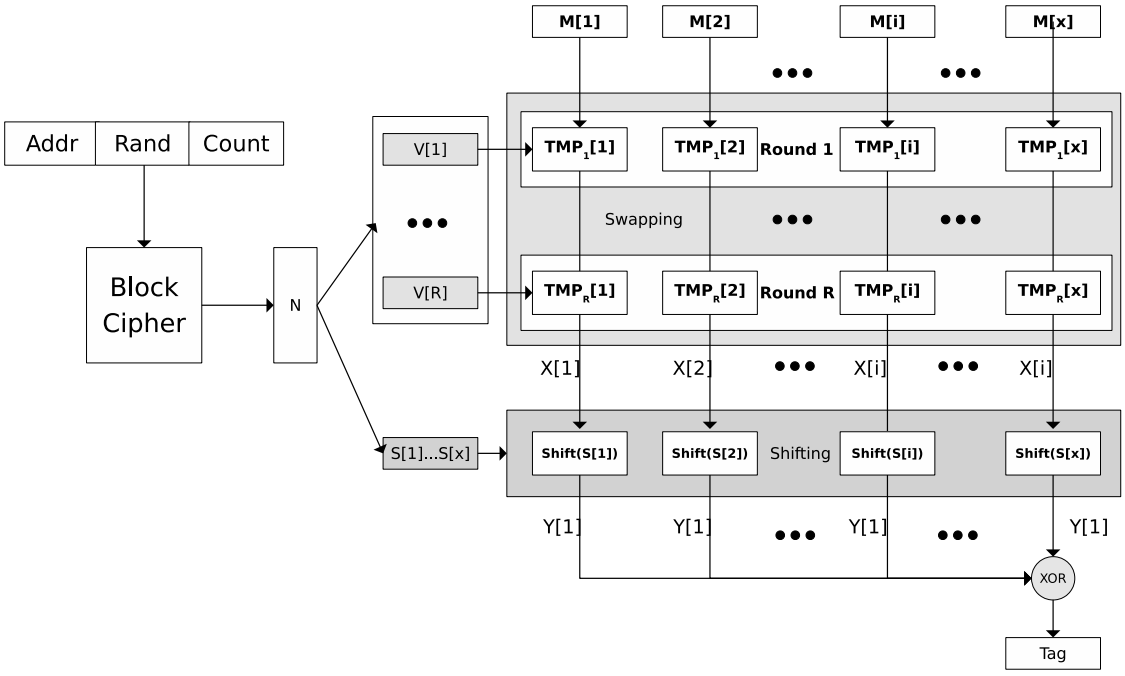


Figure 1: The CETD-MAC Scheme

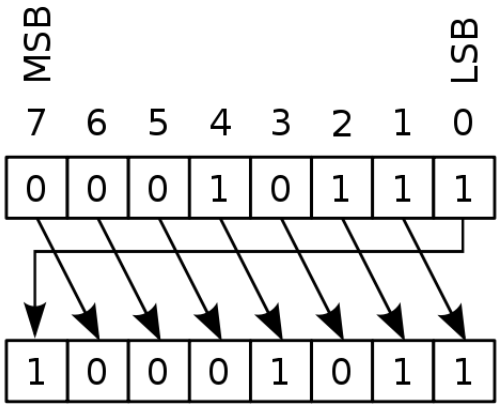


Figure 2: The Concept of Rotate Shifting(Right)

a nonce N. Then he keeps C and N fixed and tries different tags until a tag T can pass the verification with C and N. All the tags tried until passing the verification form a set named tag exploration space. The author assumed a MAC to be secure if the following conditions are met:

- The size of tag exploration space is large
- If the ciphertext and nonce is randomly generated and unaccessible to the adversary, the probability to pass the verification for each tag in the tag exploration space is identical

Based the above two conditions, the author assume CETD-MAC is secure.

2.2 The Randomness of CETD-MAC

In this part we analysis the randomness of CETD-MAC as the randomness of CETD-MAC determines the forgery success probability. The adversary is assumed to know the length of ciphertext and tag on the memory frame he acquired. We prove that the probability that the fake pair passes the verification stage under both two attacks is determined by the proportion of the 1s and 0s in the input ciphertexts and provide the equations quantifying this relationship. This fact indicates that the adversary can choose special ciphertext-tag pairs which have higher probability to pass the verification compared with other pairs. This relationship between input and its tag indicates that CETD-MAC does not behave like PRF.

The Properties of PRF We follow the analysis path in the security analysis of CBC-MAC from Bellare et al []. The adversary is given a black box named oracle. There is a function in the oracle which can be a MAC scheme or PRF and decided by coin flipping. The adversary has infinite computational power and can choose any input sent to the oracle. If no matter what kind of strategy that the adversary choose, he cannot tell which function is in the oracle, we assume that the MAC scheme behaves like a PRF.

As the tag space is limited, the tag value will repeat if the adversary call the oracle too many times. However, the outputs from PRF have the following two properties if the output space is large:

1. All the element in the output space will appear
2. The output values distribute uniformly and have no correlation.
3. For any two input in PRF‘s domain, the probability of tag collision should be $1/2^n$. This property means the input of PRF has no difference and the equality of their outputs has no advantage to predict.

Property 1 is the basic requirement of the functions that behave like PRF. One counterexample that have property 1 but property 2 is a continuous counter. For property 3, a counterexample is that odd input generate odd output and even input generates even output.

2.2.1 The Relationship Between MAC Input and its Tag

In CETD-MAC scheme, the shuffle and shift stage relocate bits in the ciphertext to new index. Even though the new index of each bit in the cihpertext is unpredictable as the nonce is the output of a PRF and unaccessible to the adversary, it is certain the output of shift shuffle and shift stage contain same bits as the ciphertext. Assume ciphertext is consist of m blocks each of whose length is n bits, and there are totally k 1s in the m*n bits in the ciphertext. The output of shuffle and shift stage also have k 1s and m*n - k 0s.

If the MAC scheme is ideal and behaves like a PRF, for any input ciphertext, the probability that its tag equals to a specific value is $1/2^n$, where n is Len(tag). As the both shuffle and shift subroutines in CETD-MAC scheme are diffusion operations, for each input ciphertext, its 0-1 proportion will be maintained until the xor subroutine. This fact means that for a specific input ciphertext, its tag space cannot reach the full tag domain. For example, assume the input ciphertext is (0x0, 0x01) and the Len(tag) is 8 bits, we can see that there are only 8 possible values that CETD-MAC can produce with different nonce, which are 0x01, 0x02, 0x04, 0x08,0x10, 0x20, 0x40, 0x80. The reason for this fact is that there are only a single '1' in the input ciphertext, which means the Y blocks that serve as the input of xor stage contain only a single '1', so it is impossible to produce tag values such as 0xFF or ohter ones containing more than one '1'. From this instance we can see that for a input ciphertext, as the shuffle and shift stage can not introduce new bits, so for any nonce value, the number of 0s and 1s in the blocks doing xor operation is always same as the proportion in the input ciphertext, which means the possible number of 1sx in the tag after the xor operation is limited. We summarize this fact to the Theorem 2.1 and the proof is expressed in the appendix.

Theorem 2.1. *Assume the input ciphertext of CETD-MAC scheme, marked as C, is consist of m n-bits blocks and the nonce is a output from PRF. The number of possible distinct tag values, marked as num_tag, is determined by the number of 1s, marked as k, in C. The relationship between num_tag and k can be expressed as:*

$$\begin{aligned} num\_tag &= \sum_{i=0}^{\lfloor k/2 \rfloor} \binom{n}{k-2*i} if k \in [0, n) \\ &= \sum_{i=0}^{\lfloor n/2 \rfloor} \binom{n}{n-2*i} if k \in [n, m*(n-1)] \\ &= \sum_{i=0}^{(m*n-k)/2} \binom{n}{(m*n-k)-2*i} if k \in (m*(n-1), m*n] \end{aligned} \tag{1}$$

Based on Theorem 2.1 we get the Colloery 2.2 and 2.3:

Corollary 2.2. *if there are even number of 1s in the input ciphertext of cetd-mac, then its tag will also contains even number of 1s; if there are odd number of 1s in the ciphertext, then its tag has odd number of 1s.*

Corollary provides a counterexample of Property 3 of PRF. In Section **??** we introduce a forgery attack strategy based on Corollary **??**.

Corollary 2.3. *For any input ciphertext, the size of its possible tag rage is no more than* 2^{n-1} *where n is Len(tag).*

Corollary indicates that the Property 1 of PRF cannot be met by CETD-MAC scheme as for any input the related tag space is no more than half of the tag domain. We also adopt this weakness in constructing our forgery attack strategy on CETD-MAC. The proofs of Corollary 2.2, **??** and Theorem **??** are provided in the appendix.

From Theorem 2.1 and Corollary **??**,**??** we can conclude the following assertions about the randomness of CETD-MAC scheme:

- For a chosen input ciphertext C, the probability that its tag from CETD-MAC is identical to a given value T is not $O(1/2^n)$ for any T value. T must be in the tag space of C.
- The size of tag space of any input ciphertext C is no more than 2^{n-1} and the size value is determined by the number of 1s in C.

The above assertions indicates that CETD-MAC does not behave like a PRF.

From Theorem 2.1 we can see that the domain of input ciphertext can be splitted into three groups based on the number of 1s. The rage size is identical to num_tag. We can draw the following conclusion based on this fact:

- the maximum value of num_tag is 2^{n-1} if k $\in$[n, m*(n-1)].
- The probability of tag collision is affected by the value of num_tag

2.3 CETD-MAC is not a Secure MAC

In this part we prove that CETD-MAC is not a secure MAC. This assertion is made based on the assertion that CETD-MAC is not a PRF. As the tag space for a given input ciphertext is predictable, we found strategy to succeed the forgery on CETD-MAC in both content modification and copy-then-replay scenarios with high success probability.

Forgery Attack in Content Modification Scenario From Corollary **??** we have known that the odd-even property of a tag from CETD-MAC is same as its input ciphertext. This fact means the adversary should ensure that if he maintains the tag fixed and changes the value of ciphertext part on the memory frame, he should eusure that the odd-even property of his chosen fake ciphertext C_{fake} should be same as the original tag T_{origin}. On the other hand, if the adversary want to completely change the content of a memory frame, he should ensure that his fake pair (C_{fake},T_{fake}) has same odd-even property. By taking this strategry, the probability of succeeding a forgery attack in content modification scenario can be enhanced from $O(1/2^n)$ to $O(1/2^{n-1})$.

We can enhance the success rate of forgery attack in content modification scenario based on Theorem**??**. After acquiring the value of ciphertext and tag from a memory frame, the adversary can check the number of 1s in the tag, denoted as $\mathrm{N}_{1s}(\mathrm{T}_{origin})$. If the adversary wants to maintain the tag part of the frame unchanged, the best strategy to choose fake ciphertext C_{fake} is ensuring the number of 1s in C_{fake} is identical to $\mathrm{N}_{1s}(\mathrm{T}_{origin})$. This assertion is depicted in Corollary **??**.

Corollary 2.4. *Given a original memory frame (C_{origin},T_{origin}) and keep T_{origin} unmodificed, the best strategy for the adversary to forge with fake ciphertext C_{fake} is to ensure that C_{fake} has same number of 1s as T_{origin}. Assume the number of 1s in T_{origin} is N1s, the highest probability of successful forgery is acquired with this strategy, reaching O(1/2^{N1s-1}) is N1s is even or O(1/2^{N1s}).*

If the adversary wants to modify both the ciphertext and the tag, strategy dipicted in Corollary **??**. Based on the probability in Corollary **??** we can see that the less number of 1s C_{fake} and T_{fake}, the higher probability that the forgery succeed.

Forgery Attack on CETD-MAC in Copy-then-Replay Scenario When conducting copy-then-replay, the adversary replace a valid memory frame with his stored copy of a frame from other address or same address but older time point. The probability that the adversary succeeds the copy-then-replay attack is determined by the probability that for a fixed input C, the two tags T1 and T2 are identical when their nonce N1 and N2 are randomly generated.

As depicted in Section 2.2.1, the 0-1 proportion of a ciphertext is maintained to the input blocks(Y blocks) of xor stage. This fact means for a fixed input, the Y block set Y1 and Y2 for two randomly generated nonce N1 and N2 have same 0-1 proportion. This fact means that when the adversary conducts chosen-message attack in copy-then-replay scenario, the tag space is fixed and its size is determined by the chosen input ciphtertext. Based on Theorem **??** we can see that the tag space for a fixed input ciphertext C is determined by the number of 1s in C. A good strategy for the adversary to succeed the forgery in copy-then-replay scenario is to choose valid memory frame whose ciphertext part C has too many (more than m*(n-1)) or too few(fewer than n) 1s. Assume the tag space size for such C is denoted as Size(C), then Size(C) is smaller than 2^{n-1} and the forgery success probabilityis bigger than O($1/2^{n-1}$), which represent unsecure for short tags.

3 Improve the Security of CETD-MAC

We have shown that the original CETD-MAC cannot effectively defend two types of integrity attacks as the number of distinct tags is determined by the proportion of 1s and 0s. In this section, we propose several approaches to improve the security of CETD-MAC. Firstly we tried to utilize the nonce and the operation existing in the original CETD-MAC design and then prove the

reason that this attempt cannot succeed. Then we proposed our rationale on improvement based on the instruction from [].

3.1 Secure CETD-MAC Without Additional Operation

In this part we try to enhance the security of CETD-MAC without introducing new operations or data. We analyze the functionality of each operation first and discuss whether it is necessary to maintain each operation in the new design.

3.1.1 Design Rationale of CETD-MAC Scheme

CETD-MAC scheme is consist of three stages: bit-segment shuffle with user chosen shuffle round, then cycle shift for each output lf shuffle stage with randomly chosen shift length and finally xor all output block of shift stage to the tag. It is insecure to construct with only one of the three stages.

XOR stage The xor operation is a common choice for parallel MAC schemes to convert m n-bits block to a n-bits tag. It has been analyzed by Bellare in [] that it is insecure to construct a MAC scheme with xor operation only as simply swapping the order of two blocks in the input can lead to tag collision. The input blocks must be processed before sent to xor stage to ensure that simply changing the order of two input blocks in the input will change the input set of xor stage.

Cycle Shift Stage In original CETD-MAC scheme, cycle shift stage, short for shift stage, locates before the xor stage. For each input block of shift stage, the number of bits shifted is determined by the value of a nonce segment. Shift stage can defend the attacks such as changing the order of two input blocks as the adversary has no access to the nonce value and then can not predict the number of bits shifted of each block, then has no idea that changing the order of which two blocks lead to tag collision with high probability. However, it is still insecure to construct a MAC scheme by adopting shit stage then xor stage, as for some input blocks which are consist of one kind of patterns, shifting different number of bits can still lead to same output block. This fact means that reordering the blocks formed by a pattern have higher probability to lead to tag collision than other blocks. This fact is depicted in detail in Appendix B.2.

Bit-Segment Shuffle Stage Shuffle stage swaps two bit-segments in each round. As the index and offset of swapping in each round is determined by the nonce, the adversary cannot predict which blocks participate the shuffle. One problem to use shuffle without shift is that if the number of blocks are large while the number of shuffle round is small, the number of blocks participating the shuffle is small. This fact is serious in copy-then-replay attack as majority of blocks may not participate shuffle and the content of these unshuffled blocks will be maintained until the next stage.

In the original CETD-MAC, the shift stage will help re-ordering the bits in the output blocks from shuffle stage to ensure that the input blocks of xor stage is hard for the adversary to predict.

3.1.2 Injecting Random Bits

XOR Plus Nonce Segment

Shift Plus Nonce Segment

Shuffle Plus Nonce Segment

Diffusion is insufficient to construct a secure MAC scheme We can see what shuffle and shift do is reordering the bits in the input message blocks. We give each input bit a index from 1 to m*n. When the nonce is fixed and unknown, we are certain for two inputs M1 and M2 the bits with same index will be relocated to same new index. The adversary does not know the exact bit distribution of the input blocks to xor stage but he knows that the 0-1 proportion is same as the input of CETD-MAC. As depicted in Section 2.2, the odd-even property of the tag is identical to its input, and the number of possible distinct tag values can be predict based on the number of 1s in the input. When realized this fact, the adversary just needs to choose a tag value that has same odd-even property and the number of 1s is no more than min(n,k), there k is the number of 1s in the input.

Content Modification Scenario In content modification scenario, the adversary keeps modifying a memory frame in an address until the frame passes verification. The modification will be previous to the next write operation from the chip the same address.

Copy-then-Replay Scenario To eliminate this weakness, we need to add a operation to make sure that if the adversary does not know the value of the fixed nonce, he cannot predict the 0-1 proportion and bit distribution of the input blocks of xor stage.

3.2 New MAC Schemes Based On CETD-MAC

In this part we introduce our improved version of CETD-MAC. In our first attempt of improvement, we do galois field multiplication, short for gf-mult, to each input block with a segment extracted from nonce. We call this first attempt Fully-GFM CETD-MAC as each input block will participate the galois multiplication. We also introduce its variant named Selected-GFM CETD-MAC in which the number of input blocks doing gf-mult is determined by user choice, just as what shuffle stage does.

No-linearity Operation In cite,R and J depicted the design rationale of each operation in Rij AES. Their should be three types of operations in a block cipher

design serving as no-linearity, random-bit injection and diffusion. In original CETD-MAC scheme, both shuffle and shift operation achieve the diffusion, while no operation can inject random bits for each input and all the operation in the CETD-MAC is linear.

Galois Field Multiplication As mentioned in Section 3.1.2, it is insufficient to use linear operation only to construct a secure MAC scheme. The reason is that linear operations can diffuse the bit distribution of the input while can not introduce new bits. On the other hand, the input and the output of linear operation has linear relationship and the adversary can find the relationship and conduct successful forgery attack since then without acquiring the nonce or the key.

The we need to add no-linearity random bit injection operation. For the MAC schemes constructed with block cipher, such as CBC-MAC(and its variants) and PMAC(and its variants), the block cipher usually contains these two operations. R and J mentioned in [] the galois field multiplication and its inverse operation can serve as no-linearity operation. For example in Rij AES, the ByteSub provide no-linearity to the input by adopting the galois multiplication inverse to each input byte then affline transformation with a constant matrix to the input. Random bits that stored in a roundkey, which is acquired by expanding the encryption key, is injected to the input by xoring the roundkey with processed input.

In GMAC, the author utilized the galios field multiplication other than block cipher to process each input block serially. Each input block does galois field multiplication with H, which is acquired by encrypting constant 0 with a secret key. The output of multiplication is xored with the next input block to serve as the multiplicand for H. The advantage of multipling each xored input block with H compared with calling the encryption function to the input block directly is that the no-linearity and random bit injection is combined to one round operation, which reduces the space cost and enhances the processing speed.

The application of GF multiplication in GMAC inspired us on the security optimization of original CETD-MAC scheme as for each ciphertext to be processed, a nonce is generated with block cipher. The nonce or its segment can serve as the H in the GF multiplication in GMAC.

Adding gf-mult before shuffle can inject random bits to the inputs and the adversary cannot predict the tag based on the input value according to the attack strategy depicted in Section 2.2. This improvement works for all the input value except for all 0 input as the output of 0 gf-mult with any value will be 0. To address this problem, we introduce our second version of improved CETD-MAC scheme named Flip-then-GFM CETD-MAC scheme. We randomly flip each input block and then do selecte gf-mult as we did in the variant of first version. We will show that our second version of improved CETD-MAC is secure under two types of integrity attacks in this part.

3.2.1 GFM CETD-MAC

Our first design draft based on the original CETD-MAC is named as Fully GF-Mult CETD-MAC, which mean each input will do galois field multiplication with a nonce segment. The advantage of this draft is that if there is at least one bit value is identical to one in a input block, then the gf-mult on this block lead to an output block whose bit distribution cannot be predicted if the nonce segment is unaccessible. The are two disadvantages of this draft: firstly if a input block is all 0, then the gf-mult will generate all 0 output block no matter what nonce segment is applied; secondly a calling of gf-mult is much time costing compared with the shuffle or shift operation.

Selected-GFM CETD-MAC As gf-mult is a time costing operation, for some embedded platform the high security brought the Fully GF-MULT CETD-MAC is less important as the update speed of the memory frame is high and the adversary has no enough time to finish the whole attack procedure. Base on this fact we modify the draft from conducting gf-mult on each input block to the new draft, which is named as Selected-GFM CETD-MAC. IN Selected-GFM, there is another input parameter r_blk determine the number of input blocks which will be conducted gf-mult. Then r_blk continuous input blocks will be selected and the index of first input block is determined by a segment on the nonce.

Like Fully-GFM CETD-MAC, Selected-GFM CETD-MAC cannot process the all 0 input blocks either.

3.2.2 Flip-then-GFM CETD-MAC

As no matter fully or selected block version of GFM CETD-MAC, the input blocks consist of all 0s can not be processed as 0 gf-mult with any value will lead to 0. To address this problem, we add another stage before selected gf-mult.

Block Flipping

Flipping-GFM CETD-MAC

A The Security Weakness of CETD-MAC

In this part we prove the theorems and related corollaries about the relationship between 0-1 proportion in the input ciphertext and the probability of tag collision.

A.1 Proof of Assertions in Section 2.2

We proof the theorems and corollaries in Section 2.2. These assertions reveal the security weakness that the size of tag range for a input C is determimined by the 0-1 proportion of C, which means the adversary can use input with small tag range for the forgery.

Proof of Theorem 2.1 As mentioned before, the 0-1 proportion is maintained during the shuffle and shift operation. That means if two distinct ciphertext C1 and C2 have same proportion, marked as k/m*n-k, the 0-1 proportion in their corresponding shift outputs Y1 and Y2 is also k/m*n-k. If we fix a nonce and keep querying ciphertext with same 0-1 proportion, the outputs of shift stage can be regarded as a set of m*n bits messages with same 0-1 proportion but uncertain bit distribution. If a ciphertext has k 1s, then the distinct ciphertexts have k 1s form a set S_n^k and the set size is $\binom{n}{k}$, marked as C_n^k. The ith element in set S_n^k is marked as C[i] and the tag of C[i] is T[i]. We fix a nonce N for the set S_n^k and compute T[1]. We discuss possible distribution of k 1s in m Y blocks according to the value of k compared with m*n to see the number of possible distinct values of T1.

Firstly, we assume k is no larger than n. When xoring Y blocks to generate T[1], the jth bit of T[1], marked as T[1][j], is computed as T[1][j] = Y1[1][j]⊕Y1[2][j]⊕Y1[m][j], where Y1[i][j] is the jth bit in the ith Y block. We call the jth bit in all Y blocks a jth columm. The jth bit of T1 is formed by xoring all the bits in the jth column of Y block. It is obvious that T1 can have at most k 1s and the number of distinct T1 with k 1s is $\binom{n}{k}$. In this case all the k 1s distribute in distinct k columne. If we want new value of T1, we need to move one of 1 to another column. This relocation reduce the number of 1s in T1 from k to k-2, leading to $\binom{n}{k-2}$ new distinct values of T1. If we want more new values, we need to keep move 1s to another column like this way and each time of relocation reduce 2 1s in T1, which means the possible number of 1s in T1 canbe k, k-2, k-4,. . ., 0(if k is even) or 1(if k is odd). When k is identical to n or n-1, the number of distinct tags is 2^{n-1}.

If k is larger than n and no larger than m*(n-1), for any k in this domain, it is possible to generate a tag containing 0 to n 1s. That means for any k ∈ [n, m*(n-1)], the number of possible distinct tags for a ciphertext is 2^{n-1}.

When k is larger than m*(n-1), the way to compute number of possible distinct tags is simular to the case that k is no larger than n. In this case, the tag can has at most n - (k mod n) 0s the value with less number of 0s can be acquired by overlapping 0s in one column. The number of possible distinct

tag values can be computed with the following equation: No of distinct tags is $\sum_{i=0}^{(m*n-k)/2} \binom{n}{m*-k-2*i}$ if k is even or $\sum_{i=0}^{(m*n-k+1)/2} \binom{n}{k-2*i}$ if k is odd.

Proof of Corollary 2.2 As we depicted in the proof of Theorem 2.1 that if we want to generate a new tag value, we need to move a 1 to another column then there will be two 1s eliminated. As subtracting 2 can not change the odd-even property of a number, then the corollary is proved.

Proof of Corollary2.3 When k is in the domain [n, m*(n-1)], the number of tag range num_tag is identical to the equation: $\sum_{i=0}^{n/2} \binom{n}{n-2*i}$ if k is even and $\sum_{i=0}^{(n-1)/2} \binom{n}{n-1-2*i}$. As $\sum_{i=0}^{n} \binom{n}{n-i}$ is identical to 2^n, $\sum_{i=0}^{(n-1)/2} \binom{n}{n-1-2*i}$ is identical to $\sum_{i=0}^{n/2} \binom{n}{n-2*i}$, which is 2^{n-1}. When k is in other domains, num_tag can be expressed as $\sum_{i=0}^{\lfloor x/2 \rfloor} \binom{n}{x-2*i}$, where x $\in$ [0,n). As x is smaller than n, then num_tag is smaller than $2^(n-1)$. The corollary is proved.

B Design Rationale of CETD-MAC

In this section, we prove the theorems and corollaries indicating the design rationale of original CETD-MAC scheme. These assertions reveal the fact that it is

Exposing the Weakness of CETD-MAC As the security of CETD-MAC is not provided in [], we do not know the randomness of tags under replay attack. To begin with, we conduct a simulation of replay attack on inputs formed by two 8-bit blocks concatenated. The hexadecimal value of the input various from 0x0000 to 0xFFFF. For each input, we static the Tag Distinction Rate(TDR) under 1000 times replay attack. TDR is acquired by dividing the number of distinct values of 1000 tags with 2^8. If TDR is low, the tag collision probability is high. Figure **??** expresses the TDR of all inputs. We can see that for some specific inputs the TDR is extremely low which means the collision probability is high. The experiment results expressed in Figure **??** introduce the motivation of systematically analyzing the security of CETD-MAC under replay attack. We examined the behaviour of each stage in CETD-MAC scheme and found that each stage have a set of inputs with high probability leading output collision.

B.1 Weakness of Shuffle Stage

Under replay attack, the input data M is identical for the verificaiton and the replayed data-tag pair from the adversary. The basic requirement of shuffle stage is diordering the bits in M and ensure the distinction of the outputs, X blocks. If for some specific M, the X block set X1 for the verificaiton is identical to the set X2 for the adversary‘s data, shuffle stage is assumed lose the effect.

According to the original design of CETD-MAC, a segment of nonce is extracted as the control parameter to a round of shuffle. As nonce is the output of block cipher encryption, we assume the control parameter random. We denote this segment R[i] where is expressed the ith round of shuffle. R[i] is splitted to five bit segment components as R[i] = (index1‖offset1‖index2‖offset2‖segment length). Index determine which two blocks are selected for shuffle and offset determines the beginning bit of bit segment. The length of bit segment in block1 and block2 is determined by segment length.

We initialize the analysis of shuffle stage at 1 round shuffle. During the replay attack, when the shuffle parameters $R_1[1]$ and $R_2[1]$ is identical, the outputs of shuffle stage X1 and X2 are identical. If $R_1[1]$ and $R_2[1]$ are distinct, the equality of X1 and X2 are determined by the input M, and the two R parameters.

Assume the number of blocks in input M is x and the shuffle parameters $R_1[1]$ and $R_2[1]$ are distinct. Due to the randomness of block cipher encryption, the bit segment from nonce is assumed to be random and the value distributed uniformly. This property lead to the fact that when $R_1[1]$ and $R_2[1]$ are distinct there is at least one component pair is distinct. The analysis of the behaviour of one round shuffle is classified with each component pair. The following paragraphs in shuffle analysis is not finished, needs further discussion.

When Index Are Distinct Assume the $R_1[1]$ and $R_2[1]$ are distinct in the index components. If one pair of index is identical and the other is distinct, there are three pairs of M blocks involved in the one round shuffle, marked as ($M_1[i]$, $M_2[i]$), ($M_1[j]$,$M_2[j]$) and ($M_1[k]$,$M_2[k]$).

If both two pairs of index are distinct, there are four pairs involved in the one round shuffle, marked as ($M_1[i]$, $M_2[i]$), ($M_1[j]$,$M_2[j]$), ($M_1[k]$,$M_2[k]$) and ($M_1[h]$,$M_2[h]$).

We can see that if the blocks in a input data M involved in the shuffle has common bit segment, there is possibility to form identical X blocks. The longer the big segment is, the higher the probability that X collision occurs.

When Offset Are Distinct If the $R_1[1]$ and $R_2[1]$ are distinct in the offset components, there are two M block pairs involved, marked as ($M_1[i]$, $M_2[i]$), ($M_1[j]$,$M_2[j]$).

When Segment Length Are Distinct If the $R_1[1]$ and $R_2[1]$ are distinct in the offset components, there are two M block pairs involved, marked as ($M_1[i]$, $M_2[i]$), ($M_1[j]$,$M_2[j]$).

We can see that when the segement-length components is distinct, the X collision can be triggered with M blocks with comon bit segment.

Combination of Cases

B.2 Weakness of Shift Stage

In this part we depict the weakness of shift stage and prove that it is not secure to construct a MAC scheme with only shift and xor stage.

XOR Operation and Tag Collision The tag T from CETD-MAC can be expressed as the following Equation 2:

$$\mathrm{T} = \mathrm{Y[1]} \oplus \mathrm{Y[2]} \oplus \ldots \oplus \mathrm{Y[x]} \tag{2}$$

Then Pr[T1 = T2] can be expressed as the Equation 3:

$$\begin{aligned} Pr[T1 = T2] = Y_1[1] \oplus Y_1[2] \oplus \ldots \oplus Y_1[x] \\ \oplus Y_2[1] \oplus Y_2[2] \oplus \ldots \oplus Y_2[x] \end{aligned} \tag{3}$$

We define the concept "equivalent set" as the set whose elements are identical. We denote Y_{all} as Y1∪Y2, and Y_{sub} as a sub-set of Y_{all}. $\sum_{i=1}^{x} \oplus\ \mathrm{Y}_{sub}[\mathrm{i}]$ is denoted as $\mathrm{Y}_{sub}[1] \oplus \ldots \oplus \mathrm{Y}_{sub}[\mathrm{x}]$. Assume Y_{sub} is an equivalent set, then $\sum_{i=1}^{x} \oplus\ \mathrm{Y}_{sub}[\mathrm{i}]$ is identical to 0 if x is an even number and to $\mathrm{Y}_{sub}[1]$ if x is an odd number.

Assume Y_{all} can be splitted to several distinct equivalent Y_{sub} sets and the number of distinct Y_{sub} sets each of whom contains odd number of elements is k, then the result of $\sum_{i=1}^{2} x \oplus\ \mathrm{Y}_{all}[\mathrm{i}]$ can be expressed as $\sum_{i=1}^{k} \oplus\ \mathrm{Y}_{dist}[\mathrm{i}]$, where Y_{dist} is denoted as the set of k distinct values. The Pr[T1 = T2] is identical to $\sum_{i=1}^{k} \oplus\ \mathrm{Y}_{dist}[\mathrm{i}]$.

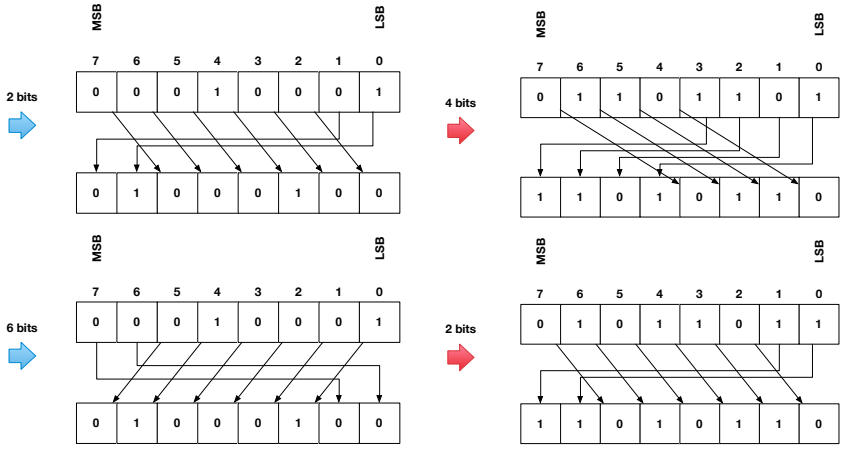
We can see that the probability of tag collision is determined by the number of distinct values in the union of Y1 and Y2, which are the output sets from rotate shifting stage. If the shifting stage can not ensure the randomness of Y sets if X set collision occurs , the related tags have high probability to collide. Figure **??** gives the examples of the outputs from shifting stage that lead tag collision. There are two types of repeated X sets leading to high probability of tag collision, denoted as sequence-level pattern and set-level pattern.

The Sequence-level Pattern Collision Firstly, we treat blocks as a block sequences, which means in a block sequence, each block has a unique index. The definition of block sequence equality is expressed below:

Definition B.1. *Sequence Equality: Two block sequences X1 and X2 are equal in sequence-level if $X_1[i]$ is identical to $X_2[i]$ for any i∈[1,x].*

Assume the shuffle stage does not work, then the following fact exist:

- X1 and X2 are two identical block sequences in sequence-level



(a) Same Block Shifted Different Bits (b) Different Blocks Shifted Different Bits

Figure 3: The Examples of Y Block Collision

In block-level rotate shifting stage, a segment from nonce, marked as S, is adopted as the parameter of shifting bits. S is a concatenation of sub-blocks and is denoted as S=(S[1]‖S[2]...S[x]), where x is the number of blocks in X sequence. The value of S[i] the bits shifted for X[i] block. For two identical blocks X_1[i] and X_2[i], the output blocks Y_1[i] and Y_2[i] are identical if S_1[i] is identical to S_2[i]. We found that Y_1[i] and Y_2[i] still have probability to be identical if X_1[i] is formed by a repeated short bit segment named pattern. This property is depicted in Proposition 1.1:

Theorem B.1. *Let $X_1[i]$ and $X_2[i]$ be two identical block and len($X_1[i]$)=N, $N=2^n$. Let $S_1[i]$ and $S_2[i]$ be two distinct shifting bit parameters for $X_1[i]$ and $X_2[i]$. Then $Y_1[i]$ and $Y_2[i]$ can be identical only when $X_1[i]$ is formed by repeating a binary bit segment named pattern and marked as P. The length of P len(P) has the following format: $P\_L = 2^p$, $p \in [0,n-1]$*

Base on Theorem $B.1$, we got the following corollaries:

Corollary B.2. *If a pattern P is not formed by a sub-pattern with shorter length, we call P a distinct pattern with length P_L. The No. of distinct patterns with length $P\_L=2^p$ is 2^{2^p}-$2^{2^{p-1}}$*

Corollary B.3. *Assume the length of a pattern-forming X block is $N=2^n$, then the No. of all possible distinct patterns is $2^{N/2}$*

When the adversary conducts the replay attack on the data concatenated by pattern-formed blocks, the probability that Y block collision is high when shuffle stage does not work. The proof of Theorem $B.1$, Corollary $B.2$ and $B.3$ is expressed in the appendix.

The Probability of Tag Collision Under Sequence Pattern Attack For two identical X blocks X_A[i] and X_B[i], the probability that Y_A[i]=Y_B[i] when R_A[i] and R_B[i] are randomly generated is marked as Pr[Y block collision]. Pr[Y block collision] is expressed in Theorem $B.4$:

Theorem B.4. *Assume for two identical blocks $X_1[i]$ and $X_2[i]$ the length is $N=2^n$, and the pattern length $P_l=2^p$ where $p\in[0,n-1]$. If the pattern contains no internal sub-pattern, then :*

$$Pr[Y_1[i]=Y_2[i]] = 1/2^p \tag{4}$$

The Set-level Pattern Collision Secondly we treat blocks as a set allowing the existence of identical elements. We give the definition of block set equality below:

Definition B.2. *Set Equality: Two block sets X1 and X2 are identical in set-level if the following properties are met:*

- *X1 and X2 have same number of elements*
- *The number of elements for each distinct value in X1 is identical to the number in X2*

When none of block in a X block sequence is formed by repeated pattern, it is impossible to make two Y sequences with block sequence equality by using two distinct s sequences. This assertion can be proved based on the proof of Proposition 1.1. However, we found that it is possible to form two Y sets with block set equality by using two distinct s sequences. The set-level identical Y sets can lead to tag collision directly. This attack is expressed in Theorem $B.5$:

Theorem B.5. *Let $X_1[i]$ and $X_2[j]$ be two distinct block and $len(X_1[i])=len(X_2[j])=N$, $N=2^n$. Let $S_1[i]$ and $S_2[j]$ be two distinct blocks of shifting bit parameters for $X_1[i]$ and $X_2[j]$. Then $Y_1[i]$ and $Y_2[j]$ can be identical if and only if $X_1[i]$ can be formed by rotate shifting $X_2[j]$.*

Based on Theorem $B.5$, we got the following corollary:

Corollary B.6. *For the rotate shifting stage, when the distinct input blocks $X_1[i]$ and $X_2[j]$ have the property that $X_1[i]$ can be formed by rotate shifting $X_2[j]$,and the shifting bits parameter $S_1[i]$ and $S_2[j]$ are distinct, the probability that the output blocks $Y_1[i]$ and $Y_2[j]$ are identical is:*

$$Pr[Y_1[i]=Y_2[j]]= 1/n \tag{5}$$

When the adversary conducts replay attack on the memory frame whose data part is concatenated by blocks that are formated by shifting a common base block, each element in X1 sequence can be formed by rotate shifting another element and the probability of forming identical Y sets is high, which will lead to tag collision. This fact is obvious if the shuffle stage cannot change the content of input data M.

The Probability of Tag Collision Under Set Pattern Attack Assume X1 and X2 are identical in sequence level and each block in X1 can be formed by rotate shifting another block in X1. From Corollary $B.6$ we know that for a N-bit block, the size of domain is 2^N and half of the domain is formed by repeated pattern. This fact also indicates that half of the domain of a block is not formed with a pattern, totally $2^{N/2}$ distinct values.

Assume the X sets under replay attack, marked as X1 and X2, are identical in sequence level and none of the blocks is formed by a repeated pattern. If all the X blocks are formed by rotate shifting a base block which is not consist of a pattern, we denoted such X1 and X2 sets same base sets, short for SBS sets.

When X1 and X2 are SBS sets, the probability that Y1 and Y2 are identical in set level if shifting control paramter sets S1 and S2 are randomly generated is denoted as Pr[Y1 and Y2 are set-level equal | X1 and X2 are SBS sets] i, short for Pr[Y Sets Collision]

The lower bound of Pr[Y1 and Y2 are set-level equal | X1 and X2 are SBS sets] is expressed in Theorem $B.7$.

Theorem B.7.

$$Pr[Y\ Sets\ Collision] \geq (\sum_{K=1}^{min(N,x)} (P_n^k * 1/2 * S(x,k)^2))/(N^{2x}) \tag{6}$$

S(x,k) is the Stirling Number with x and k as inputs.

The Vulnerable Inputs Set of CETD-MAC Under Replay Attack As summarized before, the shuffle stage is vulnerable to inputs that contains long all-0 or all-1 bit segments. After splitting a input to blocks, shuffle stage cannot effectively change the bit distribution of input blocks and the output of shuffle stage, X blocks, have big probability to be identical to the input. On the other hand, shifting stage if vulnurable to the input formed with a type of patterns. The patterns can exist in sequence level or set level. The pattern in the input of shifting stage have good chance to induce identical Y sets and directly produce tag collisions. We can see that if the input chosen by the adversary for replay attacks is the in the intersection of vulunrable inputs of shuffle and shifting stage, none of the stages can effectively randomnize the input data M under replay attack, then there is good chance to form identical Y sets even if the nonce is generated from secure block cipher encryption.

B.3 Proof of Theorem $B.1$

This part proves that if two identical block X_A[i] and X_B[i] are shifted different bits and the result blocks remain identical, X_A[i] is formed by pattern. The pattern length and δ=|R_A[i]-R_B[i] | has such correlation:

- If Δ = P_L then Y_A[i] = Y_B[i] where P_L the length of pattern

Assume the length of X_A[i] is N bits, where N= 2^n. When X_A[i] is formed by a pattern whose length is P_L = 2^p bits, then the domain of Y_A[i] has P_L distinct values. There are N/P_L distinct R_A[i] values that shift X_A[i] to a Y_A[i].

B.4 Proof of Theorem $B.4$

For identical each block pair X_A[i] and X_B[i], the No. of combination of R_A[i] and R_B[i] = N*N, where N is the length a X block. If X_A[i] is formed by pattern, then Y_A[i]=Y_B[i] if |R_A[i]-R_B[i]| = P_L * K, where P_L is the pattern length and K is a positive integer. Then for two random R_A[i] and R_B[i], Pr[Y_A[i]=Y_B[i] | X_A[i] = X_B[i] & pattern = P_L](shoft for Pr[Y_A[i] = Y_B[i]]) can be expressed in the following way:

- Pr[Y_A[i]=Y_B[i]] = N * (N/P_L)/(N*N) = 1/P_L

B.5 Proof of Theorem $B.5$

If two distinct X blocks result two identical Y block. Assume the shift bits parameter blocks are R_A[i] and R_B[i]. Then X_A[i] can be formed by rotate shifting Y_A[i] for (N-R_A[i]) mod N bits. X_B[i] can be formed for (N-R_B[i]) mod N bits. X_B[i] can be formed by rotate shifting X_A[i] for (R_A[i] + N-R_B[i]) mod N bits. Theorem 1.2 proved.

B.6 Proof of Theorem $B.7$

As the input sets of rotate shifting stage, X1 and X2, contain the following properties:

- X1 and X2 are identical in sequence level
- Each block in X1 can be formed by rotate shifint another block in X1 for several bits

After the shifting stage, the number of distinct block value in Y1 set, denoted as k, fall in the domain [1,min[N,x]]. For each k, Y1 and Y2 set are identical in set level if the following properties are met:

- Y2 blocks have k distinct values

- Assume the k distinct values in Y1 is formed as: value1 has v1 blocks, value2 has v2 blocks, ..., valuek has vk block and $\sum_{i=1}^{k}$ vi =x. Then the distribution of k values amount x blocks in Y2 set is same as the distribution in Y1 sets.

For example, assume there is 2 distinct values among 4 blocks in Y1 set, then there are two types of distribution of blocks can be (1,3) or (2,2). For type (1,3) there is four different ways to distribute 4 blocks and for type (2,2) there is three ways. The number of ways to distribute 2 values to 4 blocks is 4+3=7. If the distribution in Y1 is (1,3), then Y2 set is identical to Y1 set if the distribution of blocks in Y2 is (1,3). When there are k distinct values among x blocks, the number of ways to distribute k values to x blocks is know as Stirling Number of Second Type, marked as S(x,k). Assume there are k_x types of distribution and the number of ways to distribute k values for each distribution type is denoted as V_1,V_2,...,V_{k_x}. Then $\sum_{i=1}^{k\_x}$ V_i = S(x,k)

For each type of Y1 blocks distribution whose distribution ways number is V_i, the number of Y2 sets that are identical to Y1 is V_i. Then for each type of distribution, the probability that Y1 and Y2 set are identical in set level can be expressed as:

$$V_i^2/(N^x)^2 \tag{7}$$

Then the probability that if there are k distinct values in Y1 set Y1 and Y2 are identical in set level can be expressed as:

$$P_N^k * \sum_{i=1}^{k\_x} V_i^2/(N^x)^2 \tag{8}$$

As S(x,k) = $\sum_{i=1}^{k\_x} V_i$, $\sum_{i=1}^{k\_x} V_i^2$ is no less than 1/2 * S(x,k)^2. Then the following inequation stands:

$$P_N^k * \sum_{i=1}^{k\_x} V_i^2/(N^x)^2 \geq P_N^k * 1/2 * S(x,k)^2/(N^x)^2 \tag{9}$$

As the domain of k is [1,min(x,N)], then:

$$Pr[Y1 = Y2 \mid S1 \text{ and } S2 \text{ are random}] = (\sum_{k=1}^{min(x,N)} (P_N^k * 1/2 * S(x,k)^2))/(N^x)^2 \tag{10}$$

Theorem $B.7$ is proved.