1 The Security Analysis of Cost-Effective Tag Design

In this section we analysis the security of cost-effective tag design. We firstly introduce our security evaluation model of tag design. This model splits the security analysis into two steps: scheme security and implementation security. We analysis

1.1 Threat Model of Tag Designs

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 modifying attack and copying-then-replaying attack.

When conducting a content modifying attack, then adversary modifies the content of a (ciphertext, tag) pair. At least one of ciphertext and tag are distinct to the original one sent from the chip. When the chip reads the modified pair (C1, T1), this pair is verified by VF(C1, T1, ...). If VF(C1, T1, ...)=1, then the modified pair passes and the content modifying attack succeeds.

For a copying-then-replaying attack, we make the following denotion 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 copying-then-replaying attack on a memory frame M whoes 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, the fake pair(C2, T2) is verified by VF(C2, T2, ...) and the copying-then-replying attack succeeds if VF(C2, T2, ...)=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.

1.2 Implementation Security of Cost-Effective Tag Design

As the MAC scheme adopted in cost-effective tag design(CETD) is a statefule scheme, the input of nonce generation should be distinct for any two write-read period. This is the security requirement in implementation level.

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

1.3 Computational Based Security Analysis of CETD-MAC

In this section we discuss the security of CETD-MAC under two tpyes of integrity attack. 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 scheme 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.

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 input 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.

Chosen Message 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 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 unaccessable 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.

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, is 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).

Design Rationale of CETD-MAC CETD-MAC scheme is consist of three stages: bit-segment shuffle, cycle shift and xor. The purpose of bit-segment shuffle is to diffuse the distribution of bits in the input. The number of shuffle rounds effects the condition of diffusion. The cycle shift stage diffuses the distribution of input in each block. Another purpose of shift stage is to diffuse the blocks that do not participate the shuffle stage, which is common if the number of blocks is large while the number of shuffle rounds is small. Finally, the output blocks of shift stage are xored to form the final tag.

1.3.1 Security of CETD-MAC under Content Modification Attack

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.

In this part we prove that for CETD-MAC, the input ciphertexts have relationship between their tags. For a given tag, the number of possible inputs is limited and can be computed based on the tag value. On the other hand, for a give input ciphertext, the number of possible tag value is also limited and can be computed based on the ciphertext value. This fact indicates that when modifying the ciphertext part C1 of a memory frame, the number of possible ciphertexts C2 to replace C1 is limted. On the other hand, the adversary can manifast the fake ciphertext-tag pair C_{fake}-T_{fake} based on the relanship between a ciphertext and its tag quantified in thei part and win a much high probability to pass the verification stage compared with randomly choosing a ciphertext and a tag.

When conducting content modification attack, the adversary modifies the content of a memory frame from (C, T) to (C1, T1). When the verification stage VF read the modified (C1, T1) pair, VF computes C1's tag, marked as T_{tmp}, using the nonce N which is also used in computing T. If T_{tmp} is identical to T1, then the content modification attack succeed. We can see that in content modification attack, for two pairs (C_{origin}, T_{origin}) and (C2, T1), their nonce N and N1 are identical. To succeed the content modification attack, the adversary will choose the C1 that has high probability to get tag value T

Diffusion Operations Do Not Introduce New Bits 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 unpredictble as the nonce is the output of a PRF and unaccessable 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 the output of shuffle and shift stage also have k 1s and m*n - k 0s.

When the modified ciphertext has same 0-1 proportion 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 seg 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 possbile 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). Then we draw the following Theorem:

Theorem 1.1. *Assume a ciphertext consist of m n-bits blocks has k 1s and k is no larger than n, then the the number of possible distinct value of its tag when the nonce is unaccessible is compute with the equation: No of distinct tags is* $\sum_{i=0}^{k/2} \binom{n}{k-2*i}$ *if k is even or* $\sum_{i=0}^{(k-1)/2} \binom{n}{k-2*i}$ *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 $\sum_{i=0}^{(m*n-k)/2} \binom{n}{()k-2*i}$ if k is even or $\sum_{i=0}^{(m*n-k+1)/2} \binom{n}{k-2*i}$ if k is odd

Forgery Attack on CETD-MAC in Content Modification Scenario In content modification attack, the adversary acquires a valid memory frame and try to modify the content. If the adversry knows the length of ciphertext and the tag, he will know the value of the ciphertext and the tag. According to the previous analysis on the relationship between bit proportion and possible tag values, the adversary can conduct the following two types of modification on the frame:

1. Select a new ciphertext that has T_{origin} in its possible tag domain. A better choice is to select a ciphertext with same number of 1s as the tag T_{origin}
2. Replace the frame with a new pair (C2, T2) that T2 is in the possible tag domain of C2

From Theorem **??** we know that if the number of 1s in m*n bits is to large(more than m*(n-1)) or too small(less than n), the possible tag domain size is small. This fact

1.3.2 The Security of CETD-MAC under Copy-then-Replay Attack

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 succeed 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 **??**, the 0-1 proportion of a ciphertext is maintained to the output blocks(Y blocks) of shift 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.

Forgery Attack on CETD-MAC in Copy-then-Replay Scenario As the probability that two tags collide when their inputs are identical and their nonce are randomly generated is determined by the 0-1 proportion of their inputs, the adversary can just attack the frame with the copy whose ciphertext has many 1s(more than m*(n-1)) or few 1s(fewer than n). Two extrame case is all 0 and all 1 ciphertexts, no matter what nonce is adopted, their tags will always be 0.

1.4 Security Enhancement of CETD-MAC with Its Operations

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

1.4.1 Design Rationale

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 finaly xor all output block of shift stage to the tag.

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 benn analyzed by Bellare in [] that it is unsecure to construct a MAC scheme with xor operation only as simpliy swaping the order of two blocks in the input can lead to tag collision. The input blocks must be processed to defend change-order attack before sent to xor stage.

Cycle Shift Stage Cycle shift operation can defend change-order attack to an extent as for some combination of input block value and its shift length, change any one of theme will lead to new result. However, it is unsecure to us cycle shift only before xor stage, as for some specific input value, changing the input value or the shift length will still maintain the output unchanged with some probability. This fact is proved in Theorem **??**. Lema **??** quantifys the probability based on the input value and shift length.

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. On 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 paticipating the shuflle is small and the adversary can repeat some blocks in the input as he wish to enhance the collision probability of the output from shuffle. 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.

The Reason of Unsecurity of Original CETD-MAC 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 reloacted 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 **??**, 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.

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.

The Reason of Security of Previous MAC Schemes 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.

To enhance the security of

2 Security Improved 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 [].

Galois Field Multiplication As mentioned in Section **??**, 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 ther hand, the input and the output of linear operation has linear relationship and the adversary can find the relationship and conduct successfule 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.

2.1 Improved CETD-MAC with higher security level

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

mulatiplication. 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.

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 **??**. 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.

2.1.1 Fully-GFM CETD-MAC

Our first design draft based on the original CETD-MAC are described in Algorithme **??**. We name this design Fully GF-Mult CETD-MAC, which mean each input will do galois field multiplication with a nonce segment.

Selected-GFM CETD-MAC

Security Analysis of

2.1.2 Flip-then-GFM CETD-MAC

Block Flipping

2.2 Our Optimization of Original CETD-MAC

Defending Content Modification Attack Hence the adversary can know the possible input set for a given tag , he can modify the content of ciphertext on the frame to a fake one that has same number of 1s as the tag and this way of modification has highest passing probability. On the other hand, as he knows the possible tag set for a given ciphertext, he can choose the tag for his fake ciphertext‘s set to evade the tag values that can not be generated with C_{fake} with any nonce.

To eliminate these advantage that left to the adversary by the original CETD-MAC, one idea is to make the relationship of 0-1 proportin between cipehrtext and its tag unpredictable if the nonce is unknown.

As the 0-1 proportion of the tag is determined by its input ciphertext, we need to ensure that no matter how we modify the input, the value of the tag for the modified input is unpredictable. Th

.1 Security Theorem Proof

.1.1 Proof of Theorem ??

Algorithm 1 Fully Galois Field Multiplication

Input:

The set of positive samples for current batch, P_n;
The set of unlabelled samples for current batch, U_n;
Ensemble of classifiers on former batches, E_{n-1};

Output:

Ensemble of classifiers on the current batch, E_n;

Extracting the set of reliable negative and/or positive samples T_n from U_n with help of P_n;
Training ensemble of classifiers E on $T_n \cup P_n$, with help of data in former batches;
$E_n = E_{n-1} \cup E$;
Classifying samples in $U_n - T_n$ by E_n;
Deleting some weak classifiers in E_n so as to keep the capacity of E_n;
return E_n;
for each $i \in [1, 9]$ **do**
    initialize a tree T_i with only a leaf (the root);
    $T = T \cup T_i$;
end for
for all c such that $c \in RecentMBatch(E_{n-1})$ **do**
    $T = T \cup PosSample(c)$;
end for
for $i = 1; i < n; i++$ **do**
    // Your source here;
end for
for $i = 1$ to n **do**
    // Your source here;
end for
// Reusing recent base classifiers.
while $(|E_n| \leq L_1) and (D \neq \phi)$ **do**
    Selecting the most recent classifier c_i from D;
    $D = D - c_i$;
    $E_n = E_n + c_i$;
end while