1 Implement Issues

2 Experiments and Results

This section represents the simulation of two types of integrity attacks on the original CETD-MAC design and its improved versions. We use short ciphertext-tag pair to demonstrate our security analysis results.

2.1 Integrity Attacks Simulation

Our simulations of the content modification and copying-then-replaying attack follow the attack definitions depicted in Section 2. We developed the simulator of original CETD-MAC scheme and its improved variants with C programming language. We designed a simulation to evaluation the behaviour of tags from each simulator under integrity attacks and demestracte our theoretical analysis with the records of tag behaviours.

Simulation Platform and Parameters Our simulator of original CETD-MAC and its improved version are developed with C programming language under C99 standard. The simulations were conducted on Linux x64 platforms. Due to the limitation of computation resources, we choose short cihpertext and tag. A ciphertext is consist of two 8-bit blocks. The tag length is 8 bits. We use 128-bits key to generate the 128-bits nonce. The block cipher we choose is the Rijndael AES implemented in PolarSSL [].

2.1.1 Content Modification Attack Simulation

When conducting a content modification attack, the adversary modifies the content of a memory frame. The verification stage compute the tag T for the ciphertext $C_f$on the modified frame $Frame_f$ using the same nonce generating the original ciphertext and tag. This procesure can be modeled as querying the MAC scheme with distinct inputs while maintain the nonce unchanged. A secure MAC scheme should ensure that for any two distint input in the domain, the probability that the two tags collide when the nonce is identical should be low.

In the simulation of content modification attack, totally 17 input sets are created and each input set is consist of all possible 16-bit two-block inputs with same number of 1s. The size of a input set is computed with the equation: $\binom{n}{k}$, where n is the Len(input) which is 16 in our simulation and k is the number of 1s in a input. We divided all 16-bits binary numbers to 17 distinct sets and each set consist of the number contains k 1s and 16-k 0s, where k $\in$ [0,16]. For each set, we use its elements as inputs to tag generations. A base nonce is generated before the test with AES and the nonce for each input in the same set is a copy of the base nonce. Table **??** expressed the relationship between the number of 1s in a 16-bits input and the set size. In our simulation of content modification attack, each round is expressed in the following steps:

- Step 1: Generate a base nonce N_{base} with a tuple(address, counter, random number) for a set S_i of inputs. Any element in S_i contains i 1s and 16-i 0s.
- Step 2: Generate two tags with a copy of the base nonce N_{base}. The first tag T1 uses a element from set S_i and the second tag T2 uses a 16-bits random number as input.
- Step 3: Repeat step 2 for all the elements in set S_i, compute the number of distinct T1s and T2s, marked as $Round_r$_num_T1_S_i and $Round_r$_num_T2_S_i. $Round_r$ is the index of current test round.
- Step 4: Repeat step 3 for all the sets where i $\in$ [0,16]

We repeat the simulation 1000 rounds and compute the average of $Round_r$_num_T1_S_is and $Round_r$_num_T2_S_is seperately. The average results Aver_num_T1_S_i and Aver_num_T2_S_i for each set S_i serves as the indicator of the security of a tested scheme under content modification attack. The concept of our simulation of content modification attack is expressed in Figure **??**.

2.1.2 Copying-then-Replaying Attack Simulation

If the adversary conducts a copy-then-replay attack, he will maintain a copy of valid fram written in an older time point of on different memory address and then replace the vailid frame with this copy. This procedure can be modeled as querying the MAC scheme with a fixed chosen input while the input of nonce generation is distinct for each query. It is possible that two tags generated with two distinct nonce but one identical input collide. The high probability of this collision indicate that the MAC scheme is unsecure under copy-then-replay attack.