On Cost Effective Tag Design for Processor Memory Data Integrity Protection in Embedded Systems

July 7, 2014

Abstract

Security becomes increasingly important to computing systems. Effective evaluation to ensure the security of a design is even critical. This paper presents a study on an existing memory data authentication design for embedded systems. We identify the security loopholes of the design and present an improved design solution.

1 Introduction

Security becomes increasingly critical in embedded systems due to ubiquitous eServices. One vulnerable component in the embedded system is the processor off-chip memory that may be accessible to adversaries. If the memory data are tampered, the security of the system will be greatly affected. To protect the system from potential attacks, data need to be encrypted before sent off-chip and authenticated when required by the processor. Figure 1 shows a typical processor-memory system, where a unit of encrypted data in the memory is attached with a tag and the tag value is checked each time the data is fetched and used by the processor; if the tag value is changed, the data is deemed as invalid. Usually the tag size indicates a level of protection for the data integrity. For high security, a large tag size is often required. However, designs using large tags will sustain considerable memory consumption, which is often unbearable to embedded systems.

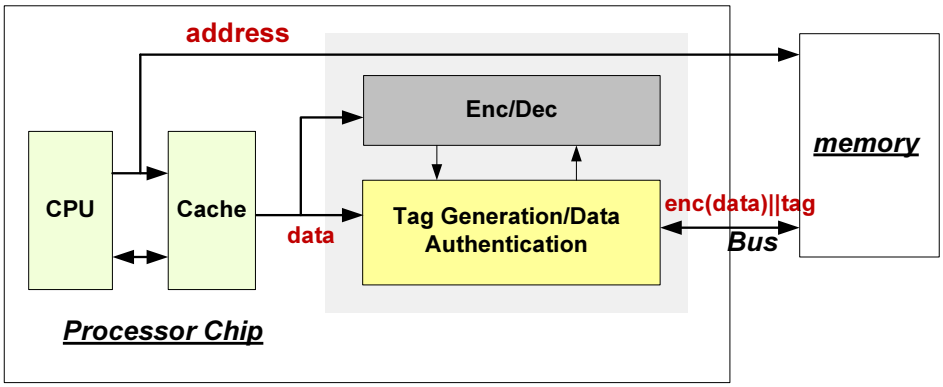

Figure 1: Typical Processor System Structure

In [1], the authors presented a cost effective tag design. Unlike other existing designs, where the tag generation logic is fixed and the related high cost cannot be reduced, this design offers design flexibility for varied security levels (which often is the case with embedded systems where different applications may require different levels of security) and enables possibility for reducing hardware implementation costs if the tag size (hence security level) is downgraded. However, the security in their design is mainly focused on the attacks with random data and tag values; attacks with chosen values are not adequately addressed. Nevertheless, the chosen value attack exploits the weakness of design and can be much stronger than the random attack.

We tested the tag generation design proposed in [1] with some chosen input data. Figure 2 gives the number of distinct tag values collected in the experiment for each input (based on sufficient samples). The number of distinct tag values shows the size of effective tag domain. From the experiment, we can see that some data have smaller tag domains than the others.This invariance of tag domain sizes indicates that the system is not equally secure for different data. For an attack with carefully chosen data, a high success probability exists.

In this paper, we investigate this problem in the proposed design and present a possible solution that can counter chosen-value attacks while retaining main features of the original design – namely flexible, cost effective, yet more securer.

The rest of the paper is organized as follows. Section 2 reviews some existing works related to data authentication and design security evaluation.

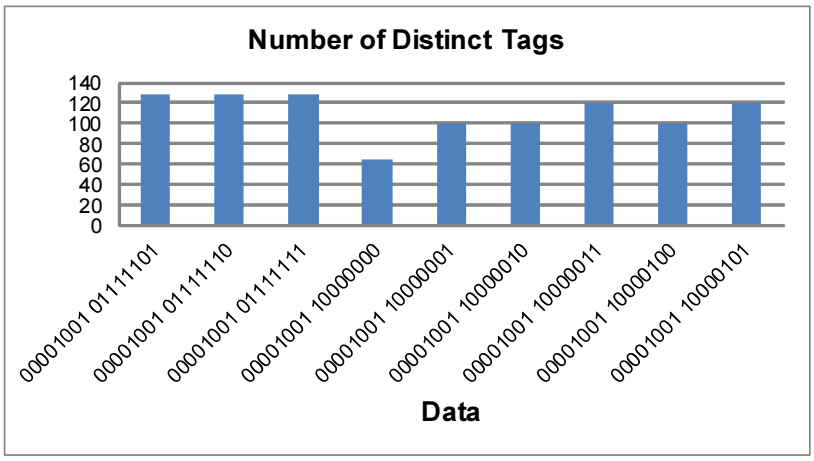

Figure 2: Experiment Data: Number of Distinct Tag Values of Different Data

The memory authentication design proposed in [1] and its security risks are discussed in Section 3. Section 4 presents the designs for improvement. The paper is concluded in Section 6.

2 Related work

Authentication can be applied to different components in a system, for example, authenticating users, communication parties, messages, and code executed. Here, we focus on data received by a processor from its memory, hence message authentications are relevant. We first survey the message authentication, then review typical designs for memory data protection; the security evaluation will be briefly discussed at the end of this section.

2.1 Message Authentication

For the message authentication, the common approach is to use a MAC (short for message authentication code, also called tag), to identify a message. The main issue of authentication is how to design MAC to make the success of tampering data extremely difficult, which often requires the secrecy in the MAC generation.

Existing MAC schemes can be stateless or stateful. The stateless MAC

schemes use a single and static key as secret information in MAC generation; for a given message, the MAC value is fixed; therefore, some works also refer to the stateless MAC as a deterministic scheme. With the stateful MAC designs, on the other hand, the secret key changes and carries information from the system previous state; the MAC code of a message can be different from time to time.

Most MAC schemes divide messages into blocks and the MAC generation is made of operations on blocks. Depending on how those operations are structured between blocks, MAC schemes can be divided into iterated MAC and parallel MAC.

In the iterated MAC design, the operations on blocks are performed in sequence. The result of one block relies on the output of previous blocks.

The cipher block chaining message authentication code design (CBC-MAC in short) is a typical iterated MAC scheme, where encryption with a fixed key is used and block encryptions are concatenated; a block can be processed only after its previous blocks have finished.

An initial CBC-MAC design can be found in [2]. One known attack on this design is that given a message (M) of a single block, a forged message which is formed by two blocks (M and (M$\oplus$T)) will have a same tag (T) as the one-block message. Therefore, this CBC-MAC design is not secure for messages with a varied size. It can be considered secure only if every message has a fixed number of blocks. Bellare et al. in [3] presented a quantitative security evaluation based on the success probability of a forgery with a given number of tries and the probability was estimated under the assumption that the block cipher used in the design was random.

To address the problem of the initial CBC-MAC design with arbitrary length inputs, Petrank and Rackof proposed EMAC (Encrypted CBC MAC)[4] It can handle messages of arbitrary number of blocks. The design applies CBC-MAC on message blocks with one encryption key and the result is then encrypted with a different key. The authors showed that EMAC was secure if the block cipher used in the design is random.

CMAC (Cipher-based MAC) [5] is another CBC-MAC design for the arbitrary-length messages. Compared to EMAC, it has two further improvements: 1) it saves one cipher operation, and 2) messages in this design are allowed to have arbitrary number of bits, not restricted to number of blocks. Therefore, a message handled by CMAC consists of varied number of blocks and the last block can be incomplete. If incomplete, the last block is then padded with constant bits for a full block length required by the block

operation.

In [6], Black and Rogaway introduced a CMAC called XCBC that uses three keys. One key is used by the block cipher. The last block is processed by an XOR operation with the rest two keys for two different cases: complete block and incomplete block. The authors analyzed the security of their design and showed an upper bound of success probability for the random attack; the upper bound was later tightened by Minematsu and Matsushima in [7].

Using three keys in XCBC is expensive due to key generation and storage required. Kurosawa and Iwata in [8] presented a two-key CMAC, named TMAC. The two keys used in XCBC are replaced by two hashed values generated with one key. They showed that TMAC had a same level of security as XCBC.

To further reduce keys, Iwata and Kurosawa proposed OMAC [9] where only one key is used (by the block cipher) and the last block is marked with a Galois field multiplication, which is easy to implement in hardware and has performance benefit if implemented in software.

The above MAC schemes are all stateless designs. McGrew and Viega in [10] introduced a stateful MAC design, called Galois/Counter Mode authenticated encryption (abbreviated as GCM, or GMAC). In the design, the Galois field multiplication is applied to process blocks; the counter value is used for the system sequential operation state and the counter mode of operation is incorporated into the MAC generation to identify messages of different time periods. GCM is designed for both confidentiality and integrity of the message. Its security requires that the encryption in GCM be secure and the collision probability of the ciphertext blocks be low. Iwata, Ohashi and Minematsu later identified the weakness in the security evaluation given in the original paper and presented an improved evaluation result in [11].

Handschuh and Preneel in [**?**] introduced key-recovery attacks on universal hash function used in GMAC. They introduced two types of attacks: weak key finding and partial information leaks.

The iterated MAC designs incur long latency due to their sequential block operations. To reduce the delay, parallel MAC designs were proposed. In the parallel MAC scheme, all blocks in a message are processed in parallel.

In [13], Bellare, Guerin and Rogaway presented a parallel MAC scheme named XOR MAC, where all blocks are precessed in parallel by a pseudorandom function and the results of block operations are XORed for the final tag value. To ensure the order of individual blocks in the message, each block is associated with a sequence number. The block data and its sequence number

each take the half size of the pseudorandom function input; therefore more cipher operations are incurred.

To reduce the number of cipher operations, Gligor and Donescu introduced XECB-MAC [14], where a full block size of data are processed by each block cipher and the order of each block is marked by the operation $M_i + i * nonce$ (i is the input block index).

In [15], PMAC was introduced.Compared to XOR MAC and XECB MAC, PMAC requires less block cipher operations and accepts messages of varied lengths. In PMAC, each block is xored with a Gray code (for the block ordering). The padding of the last message block is marked with the GF function. The authors showed that PMAC was secure if the block cipher used behaved like a pseudorandom permutation; the probability that an adversary can distinguish the PMAC from a pseudorandom function is the sum of the output collision probability of internal block cipher operations and the probability of tag collision.

The security evaluation on the PMAC was also performed by Lee et al. [16]. Their research indicated that the PMAC design has the same level of security as the iterated MACs.

In [17], Rogaway presented another PMAC using the tweakable block cipher [18] proposed by Liskov and Rivest. With the tweakable block cipher, the PMAC is easy to implement for high performance and is also structurally simple that facilitates security analysis. The author showed that the design is secure under the security notion of message authentication system (namely, low success forgery probability).

2.2 Memory Data Authentication

For the processor memory data authentication, the data is often of a fixed length (determined by the processor cache line size).

Typical authentication designs includes cryptographic hash function based, MAC scheme based, and Added Redundancy Explicit Authentication ((AREA) based[19]. The hash-based design is the stateless scheme that cannot counter replay attacks. The tag size in the AREA design is restricted by the input size of the encryption function, which imposes the limit on the design security level. The MAC based scheme allows for stateful designs and large range of tag size, hence we focus on the MAC-based design.

In [20], Yan et al. proposed an authentication and encryption design (**AE**) to protect the confidential and integrity of memory data with Galois/Counter

Mode(GCM) operations.A data block to be protected is into chunks each of the same size as the block cipher. Each data chunk is encrypted and the tag of the whole encrypted data block is generated with a nonce that is based on the data address and its counter value. The counter for a data block is formed by two parts: a long-length major counter(64 bits) concatenated with a short-length minor counter(no more than 8 bits). When the minor counter overflow, the major counter updates. The design is proved to have the same security as GCM.

In [21] Rogers and Milenkovic proposed another AE design aiming to protect both code and data in the memory. A PMAC scheme is used in the design for memory data authentication. The tag for a memory data is associated with the data address and the sequence number, and the sequence number is unique to each tag generation for this memory location. Bellare and Namprempre in [22] pointed out this design is not as secure as [20]. One weakness of the design from Rogers et al. is that Encrypt-and-MAC structure is less secure according to the analysis from Bellare and Namprempre in [22]. Moreover, there need two distinct secret keys in the tag generation, which introduces additional on-chip cost.

In [23], Vaslin et al. designed an AE system using one-time pad(OTP) for encryption and CRC checksum module for integrity checking. The checksum is associated with the plaintext and the size of the checksum is small. Therefore, this design is less secure than other the authentication designs with hash functions (such MD5 and SHA-1), as has been pointed out by Elbaz et al. in [19].

2.3 Security Evaluation

So far, the security evaluations proposed basically follow a two-step approach: setting up the thread/attack model for a given design and investigating how the security measures implemented in the design can against the modelled attacks.

The model can be an internationally-approved set of security standards such as Common Criteria (CC) [24], NIST FIPS [25], or a theoretical math model, often used in a formal method approach.

Formal methods are usually used in communication protocol analysis to guarantee certain security properties even if a malicious party has access to the communication channel[26]. Issues in formal methods for cryptographic protocol analysis were discussed in [27].

The assessment of a design security against the modelled attacks can be theoretical induction [28], or statistical experiments, for example, testing randomness of a random generator [29].

Another types of evaluation schemes that have been found in the cryptographic area is the success probability of attacks. An attack is effective if it can succeed with a non-negligible probability [12][22].

In [30] Goldreich et al. modeled a security design as a pseudorandom function (PRF) and used the distinguishability between the PRF and a random function (RF) for evaluating security against random attacks. If the value produced by the PRF could not be distinguished from the result from the RF, the design is deemed as secure.

Ludy and Rackoff in [31] treated block ciphers as a pseudorandom permutation(PRP) and the design was evaluated based on the probability of output collisions from the PRP.

The success probability has mostly adopted in the MAC design for random attacks [3][6][7]. Also the distinshability is also popular [15].

In this paper, we evaluate the CETD design based on the tag collisions and apply a twist cipher block to patch the loopholes of CETD.

3 CETD Design

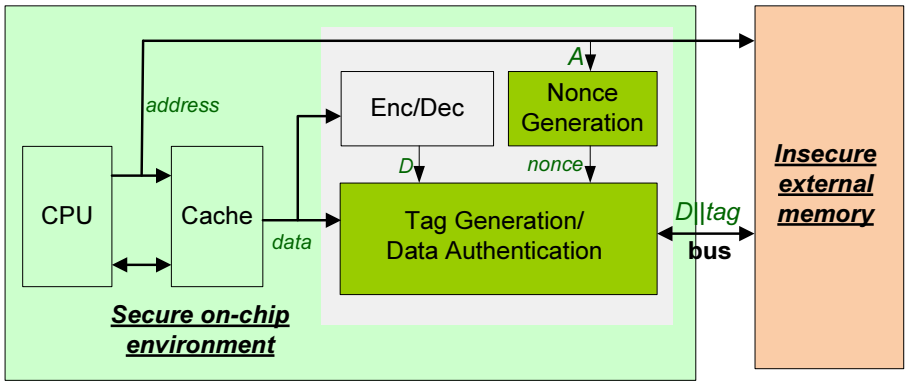

Figure 3: CETD System Overview

Figure 3 shows an overview of the system that has been proposed in [32]. It targets a system that has a secure processor chip and insecure off-chip memory. The off-chip memory can be accessed by attackers and the memory

data can be read and altered. Therefore, the plaintext data is encrypted and the encrypted data is tagged. The encrypted data D and its tag are stored in the memory. When a data is required, its encrypted value and the related tag are fetched from the memory into the processor chip, then the tag value of the fetched data is re-calculated and compared with the tag obtained from the memory. If both values are the same, the data is authenticated and can be further decrypted for use. Otherwise, the data should be discarded.

3.1 Design Overview

The tag generation process is given in Figure 4(a), where the input encrypted data is divided into blocks with the size same as that of the tag. The process consists of three steps: 1) block segment shuffle 2) block bit rotation, and 3) block XOR.

The block segment shuffle is randomly choosing a pair of blocks and swapping bit-segments between the two blocks, as demonstrated in Figure 4(b), where 2-bit segments in blocks D_i and D_j are swapped. There can be many rounds of swaps. For each round, the swap can be performed on a different block pair and with a different segment size, and the segment position in the block can also be different. The block shuffle operation breaks the block boundaries, helping to counter the slice attack [32]. The second step, block rotation, further randomizes the input. The rotation is performed on individual blocks. An example is given in Figure 4(c), where a block is left-rotate-shifted 3 bits. For each rotation, the shift distance can be varied. In the final XOR step, blocks from the rotation are merged into one block as a tag value.

The operations in both the block shuffle and block rotation steps are controlled by a nonce value, which is explained below.

For each memory location, there are a dedicated counter and an associated random number. The nonce is generated by a block cipher that takes the input the memory address A, the associated random number R, and the counter value C, as shown in Figure 5(a). The counter value is incremented for each update operation to the same location; when the counter reaches to its maximum, it will wrap around and a new random number is used, as demonstrated in Figure 5(b); the table in Figure 5(b) shows how the counter value and random number are changed for a sequence of memory update operations to memory location $A1$. Therefore, the nonce value varies with the memory location and its counter value, as specified in Figure 5(c). The

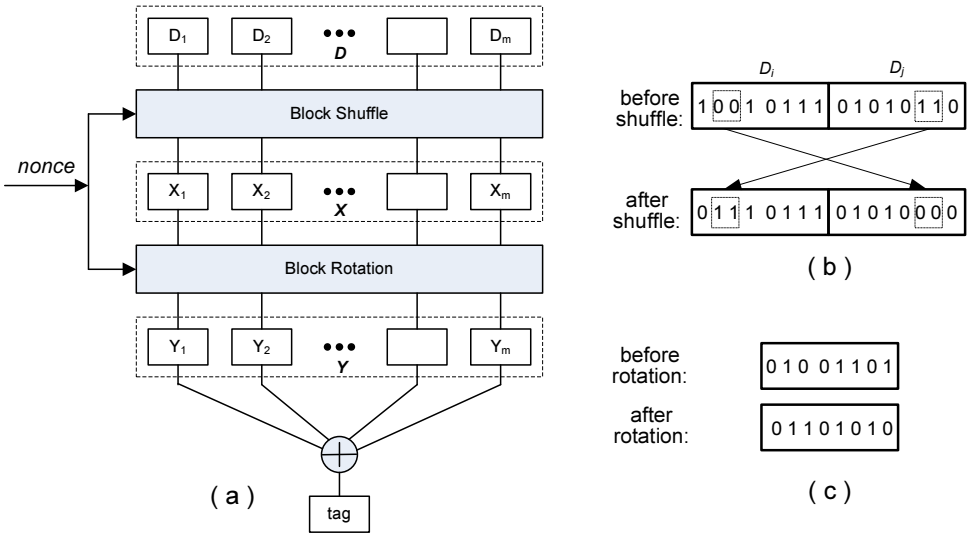

Figure 4: Tag Generation Process (a) diagram (b) shuffle example (c) rotation example

nonce stays unchanged for a given memory data and the processor will use the nonce to authenticate the data each time it is fetched from memory.

Assume the data-tag pair stored on the off-chip memory at a location, A, is marked as D-T, the possible attacks on the design can be grouped as follows:

- Modify the content of D-T at A;
- Insert new data-tag pair D'-T' to the location, where D' can be forged based on current data in the memory location.
- Replace the content of D-T with a copy of data-tag pair from another location B;
- Replay an earlier data pair D-T at A

The chosen-value attack can be performed on any of the four cases. But unlike the random attacks, the chosen-value attack exploits the weakness of the design.

Here, we analyse the CETD design and identify the data the adversary can choose for effective attacks, which are elaborated in the next section.

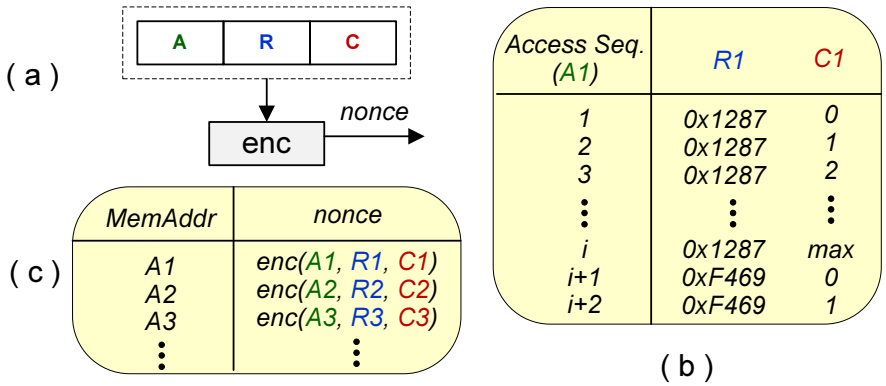

Figure 5: Nonce Generation (a) structural overview (b) change of random number (c) dedicated nonce values

3.2 Security Against Chosen-value Attacks

We evaluate the security of the design against a chosen data attack based on the tag collision rate of the data. Ideally, the design is highly secure if the tag values for any given data are uniformly distributed over the whole tag value space.

For CETD, let us consider data D, which is divided into m blocks. Each block is of n bits (namely, the tag size). We define bits of value 1 as **set bits**, bits of value 0 **unset bits**, and the number of bits of a same bit value as **bit frequency**.

The big weakness of the CETD design is that the block shuffle and shift operations in the design cannot camouflage the bit frequency of the input data. The XOR operation in the design takes value that has the same bit frequency as in the input data. Assume D consists of k set bits, then the unset bit frequency in D is $nm - k$. Assume input blocks to the XOR operation are $Y(1)$, $cdots$, $Y(m)$ and the input data D consists of k_0 set bits. The XOR operation will reduce m blocks to 2 blocks, with the number of remaining set bits, k, $k \leq k_0$ The possible output values (i.e. the tag values) from XOR for D can be derived based on a number of cases.

`WHEN` $k \leq n$`:`

One special case (case 1) is that the k set bits are in one block, say block Y_i. The number of possible tag values in this case is $\binom{n}{k}$.

If one set bit is in other block (say Y_j), the XOR operation may result in two possible situations: 1) the set bit is cancelled if it is in the same bit location as a set bit in block Y_i, or 2) the set bit is not cancelled. The result from the second situation is the same as that in case 1. But for the situation 1 when the set bit is canceled, only $k-2$ set bits left in the XOR output and the number of possible tag values is $\binom{n}{k-2}$. In the same way, we can form a general case where i set bits ($i = 0, \cdots, k/2$) are in other blocks and they are cancelled by the set bits in block Y_i, which leaves $k - 2 * i$ set bits in the XOR output. Therefore, when $k \leq n$, the total number of possible tag values, N_1, is

$$N_1 = \sum_{i=0}^{\lfloor k/2 \rfloor} \binom{n}{k-2*i} + (k\%2) * n, \tag{1}$$

N_1 increases with the number of set bits in the data, when $k = n$ (i.e., the block size), $N_1 = 2^{n-1}$.

`WHEN` $k > n$`:`

$k - n$ set bits will be cancelled and $2n - k$ bits left in the final tag value, and the number of possible tag values is

$$N_2 = \sum_{i=0}^{\lfloor n-k/2 \rfloor} \binom{n}{n-2*i} + (k\%n) * n. \tag{2}$$

$N_2 \leq 2^{n-1}$.

The above analysis shows that the effective tag value domain is smaller than the full tag value space and it varies with different data. The maximal tag domain size is 2^{n-1} – half of the full tag value space. When the frequency is unbalanced between the set bits and unset bits, N is low, and tag collisions are high. The extreme case is when the input are all set bits or unset bits, where the tag domain contains only one value and the tag collision 100%, hence 100% success when attacking with such a value.

If the tag generation algorithm is known to adversaries. They can find effective tag domain for a given data; if data with smaller tag domain size is used in the attack, the forged tag chosen from the domain may be likely collide with the tag value calculated by the processor based on current nonce (that is unknown to the adversary).

Therefore, to improve the design security, we want the effective tag value domain for each individual data large and close to the full tag value space as possible.

4 Design Improvements

In search for possible solutions to the security loopholes identified in the above section, we start with a simple and cheap design to ensure the effective tag value domain of a given data to cover the full tag value space: just XORing input data blockswith a random value block from nonce, as shown in Figure 6 (a). However, with this design,

$$tag = R \oplus (D(1) \otimes \cdots \otimes D(m)),$$

which means for a given m-block data, any data formed by the m blocks will have a same tag value. Attacks with a data modified in such a way can go undetected.

A second solution is adding randomness on top of the original design, for example injecting random components to the input or to the tag, as shown in Figure 6 (b), (c).

For (b), the tag can be written as

$$tag = R \oplus (\bigoplus(SP(D))),$$

where $SP(.)$ represent the combined shuffle and permutation operations and $\bigoplus$ represents the XOR of all blocks.

Since the block shuffle and bit shift cannot change bit frequency, any data with a same bit frequency will have a same effective tag domain. The collision rate of $\bigoplus(SP(D))$ is high. Though the effective tag value domain can be enlarged to the full tag value space (due to the random R), the resulting tag values still have high collision rate for a given memory location.

Similarly for design (c), where both the input data and tag are randomized, the tag is calculated by

$$tag = R \oplus (\bigoplus(SP(D \oplus n))).$$

Randomization of the input D ($SP(D \oplus n)$) will result in same parity for two different data with the same parity. For example if D has an odd number

of 1 bits, any data with odd number of 1 bits when XORed with the same nonce value, the resulting in parity should be same, hence the parity of the related tag values, which effectively reduces the data space by half, and the tag space by half as well, hence increasing the tag collision rate for different value in the group.

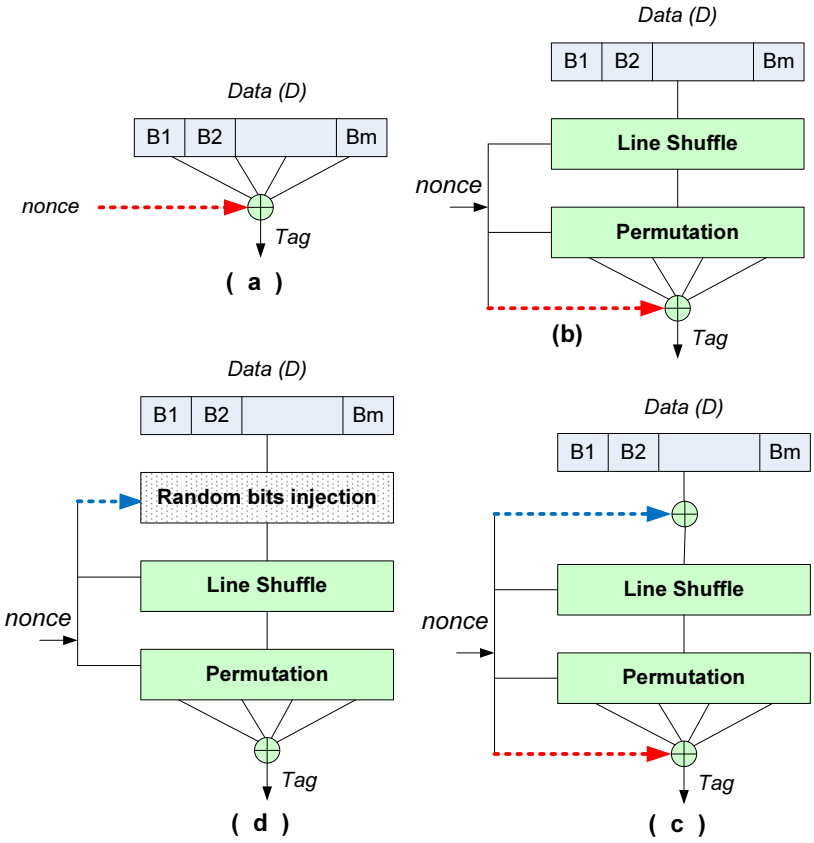

Figure 6: Design Options (a) inject random bit to tag (SR operations removed) (b) inject random bits to tag (SR operations retained) (c) inject random bits to input with XOR (d) inject random bits with substitution

Knowing that the above design solutions are not effective, we finally come to a design, with a non-linear random bit inject to disturb input data. For non-linear random bit injection, finite field multiplication (FFM) is common technique. However, it does not function for input with all reset bits. To handle this problem, we add the random bit flip operation to the input data.

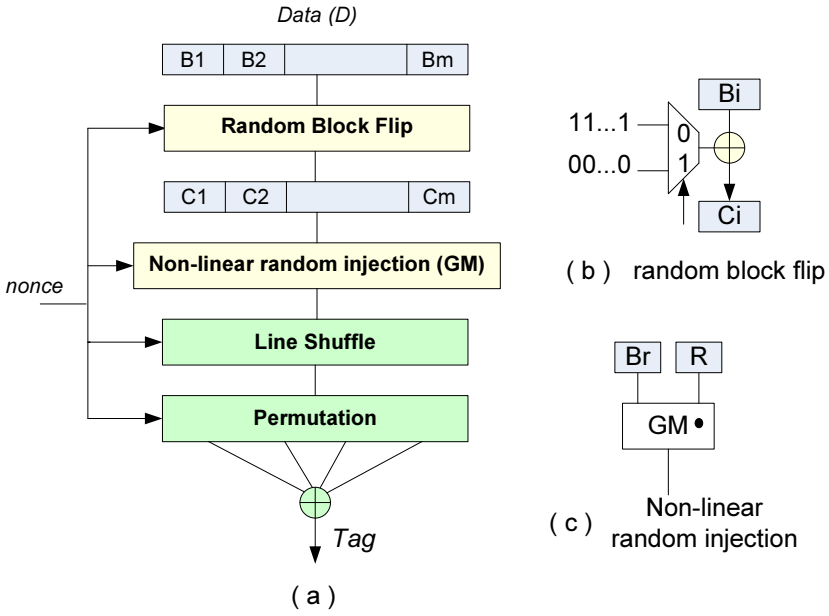

Figure 7: Final Design (a) overview (b) random block flip (c) non-linear random injection

The whole design is shown in Figure 7(a), where random block flip operation and block FFM are added on the top of the original design. A block can be randomly flipped based on the control of the nonce value bits, and the design for block flip is shown in Figure 7 (b), where 2-to-1 multplixoer is used to select all 0's and all 1's to be XORed with the related block; if the control bit is 1, the block is XOR with 1's and the it is flipped; otherwise, the block is intact. Block flip will camouflage the input value of straight set bits or reset bits.

It must be pointed out that the original design adopts a parallel MAC scheme. For a parallel MAC, block ordering should be maintained. Rather than the order is maintained by a fixed index as found in other PMAC design [][], we use randomized block flip to reenforce the block order. In case some blocks swapped by a forgery, the change of the block order will be reflected through the block flip. Also block flip will camouflage the input value of straight set bits or reset bits.

The non-linear random bit injection is realized by FF multiplcating a random value with a randomly picked block, which further confuses the input data. The confusion are then diffused over all blocks through the block shuffle

and bit rotate shift operations existing in the original design. Without the FF multiplication, the design cannot break the correlation between the input data and the out tag.

The features of the design can be summarized as follows:

- maintenance of the block order is improved,
- confusion with non-linear random operation is included.
- confusion is spread by random block shuffle and random bit shift operation
- tag value is sensitive to a change of input data,
- tag value is sensitive to a change in nonce

Playing data from different memory location will face a different nonce, resulting in different tag generation process, hence tag value; Similarly true is the replay attack. For the case of data modification for a same location, the nonce will be unchanged, hence the same tag generation process will applied to both the original and and forged data. Since the design is sensitive to the change of the input data, the tag values generated from the two different inputs are very unlikely same.

We will shown through the experiment that random distributed over the whole tag value space, which is given the next section.

5 Simulations and Discussions

Based on the study shown in the previous section. We calculate the probability of replay attacks under different cases. The results are given below.

5.1 Simulation Setup and Results

A simulation system written in C has been built to test the randomness of tag values of different design choices discussed in the above section. It basically consists of three parts: input data set generation, tag generation, and effective tag domain size for a given or a set of given input data. The tag generation uses one of the give designs: CETD, D1(improved design 1), D2(improved design 2), D3, D4, D5, D6, they are summarized in Table **??**.

Table 1: Designs Tested

name	descryption
CETD	original design proposed in [1]
D1	with GL multiplication of each block with random value
D2	with GL multiplication of each block with either random value or constant
D3	with GL multiplication of one block with random value
D4	with GL multiplication of two blocks with random values
D5	D3 plus block flip
D6	D4 plus block flip

Table 2: Number of Distinct Tags of Data Sets

set-bit count	input data set size	max tag domain size	random data set	fix bit frequency data set
0	1	1		
1	16	8		
2	120	29		
3	560	64		
4	1820	99		
5	4368	120		
6	8008	127		
7	11440	128		
8	12870	128		
9	11440	128		
10	8008	127		
11	4368	120		
12	1820	99		
13	560	64		
14	120	29		
15	16	8		
16	1	1		

Table **??** shows the number of distinct tags for two different input data sets of same size: one is formed by all data of the fixed set bit count and another is formed with randomly generated data. Since the first set is deterministic, each value in the set is unique; while data in the second set is randomly picked from the whole tag value space. any two distinct inputs in the domain, the probability that the two tags collide when the nonce is identical is low for two random numbers, but higher for data from the same set of a fixed set bit frequency.

The meaningful data is the sample size sufficiently large than the domain size. Here we use the data for the chosen data sets of set bit count in the range of 6-11, and compare CETD with the different designs, as shown in Figure 8.

Figure 8 gives the relative tag domain size of different input sets; each input set contains with all input data with a given number set bits, as given in the x-axis. The y-axis gives the ratio of the number of distinct tags for

each set compared to that of same size of data set of random input data. Each plot represents 7 different designs.

As can be seen, CETD has much smaller effective tag domain size (hence higher tag collision rate) than the proposed design for the data with a fixed set frequency. Among all the designs, D6 is the best with the effective tag domain size close to the full tag value space for all different input data sets. The two troughs represent the case,

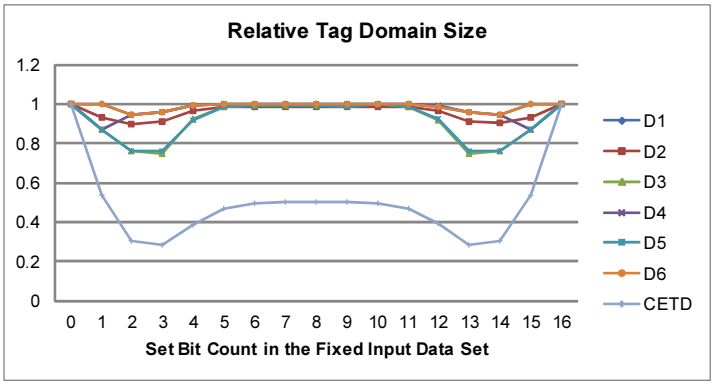

Figure 8: Relative Tag Domain Size

6 Conclusions

References

[1] Mei Hong, Hui Guo, and Sharon X Hu. A cost-effective tag design for memory data authentication in embedded systems. In *Proceedings of the 2012 International Conference on Compilers, Architectures and Synthesis for Embedded Systems (CASES 2012)*, pages 17–26, 2012.

[2] ANSI X9.9. American national standard for financial institution message authentication (wholesale). 1981.

[3] M Bellare, J Kilian, and P Rogaway. The security of cipher block chaining. In Yvo Desmedt, editor, *Advances in Cryptology Crypto 94*, volume 839 of *Lecture Notes in Computer Science*, pages 341–358. International Association for Cryptologic Research, Springer-Verlag, 1994.

[4] A Berendschot, Jean-Paul Boly, Antoon Bosselaers, J Brandt, David Chaum, Ivan Damgård, Peter de Rooij, Markus Dichtl, Walter Fumy, Cees Jansen, et al. Integrity primitives for secure information systems. final report of race integrity primitives evaluation (ripe-race 1040). *Lecture Notes in Computer Science*, 1007, 1995.

[5] NIST. Recommendation for block cipher modes of operation: The cmac mode for authentication. http://csrc.nist.gov/publications/nistpubs/800-38B/SP_800-38B.pdf.

[6] J Black and P Rogaway. CBC MACs for arbitrary-length messages: the three-key constructions. In *Advances in Cryptology - CRYPTO 2000. 20th Annual International Cryptology Conference. Proceedings (Lecture Notes in Computer Science Vol.1880)*, pages 197–215, Berlin, Germany, 2000.

[7] K Minematsu and T Matsushima. New bounds for PMAC, TMAC, and XCBC. In *Fast Software Encryption. 14th International Workshop, FSE 2007. Revised Selected Papers. (Lecture Notes in Computer Science vol. 4593)*, pages 434 —- 51, 2007.

[8] K Kurosawa and T Iwata. TMAC: two-key CBC MAC. In *Topics in Cryptology - CT-RSA 2003. Cryptoghraphers' Track at the RSA Conference 2003. Proceedings (Lecture Notes in Computer Science Vol.2612)*, pages 33–49, Berlin, Germany, 2003.

[9] T Iwata and K Kurosawa. OMAC: one-key CBC MAC. In *Fast Software Encryption. 10th International Workshop, FSE 2003. Revised Papers (Lecture Notes in Comput. Sci. Vol.2887)*, pages 129 —- 53, 2003.

[10] D A McGrew and J Viega. The security and performance of the Galois/counter mode (GCM) of operation. In *Progress in Cryptology - INDOCRYPT 2004. 5th International Conference on Cryptology in India. Proceedings (Lecture Notes in Computer Science Vol.3348)*, pages 343 – 55, Berlin, Germany, 2004.

[11] T Iwata, K Ohashi, and K Minematsu. Breaking and Repairing GCM Security Proofs. In *32nd Annual Cryptology Conference. Advances in Cryptology - CRYPTO 2012*, pages 31–49, Berlin, Germany, 2012.

[12] P Rogaway. Evaluation of some blockcipher modes of operation. 2011.

[13] M Bellare, R Guerin, and P Rogaway. XOR MACs: New methods for message authentication using finite pseudorandom functions. In *Advances in Cryptology - CRYPTO '95. 15th Annual International Cryptology Conference. Proceedings*, pages 15–28, Berlin, Germany, 1995.

[14] Virgil D Gligor and Pompiliu Donescu. Fast encryption and authentication: Xcbc encryption and xecb authentication modes. In *Fast Software Encryption*, pages 92–108. Springer, 2002.

[15] J Black and P Rogaway. A block-cipher mode of operation for parallelizable message authentication. In *Advances in Cryptology - EUROCRYPT 2002. International Conference on the Theory and Applications of Cryptographic Techniques. Proceedings (Lecture Notes in Computer Science Vol.2332)*, pages 384 —- 97, 2002.

[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 *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 4058 LNCS, pages 421–431, Melbourne, Australia, 2006.

[17] P Rogaway. Efficient instantiations of tweakable blockciphers and refinements to modes OCB and PMAC. In *Advances in Cryptology-ASIACRYPT 2004. 10th International Conference on the Theory and Application of Cryptology and Information Security. Proceedings (Lecture Notes in Computer Science Vol.3329)*, pages 16–31, Berlin, Germany, 2004.

[18] Moses Liskov, Ronald L Rivest, and David Wagner. Tweakable block ciphers. In *Advances in CryptologyCRYPTO 2002*, pages 31–46. Springer, 2002.

[19] Reouven Elbaz, Lionel Torres, Gilles Sassatelli, Pierre Guillemin, Michel Bardouillet, and Albert Martinez. Block-Level Added Redundancy Explicit Authentication for Parallelized Encryption and Integrity Checking of processor-memory transactions. *Transactions on Computational Science X. Special Issue on Security in Computing, Part I*, 6340(PART 1):231–260, 2010.

[20] Chenyu Yan, B Rogers, D Englender, D Solihin, and M Prvulovic. Improving cost, performance, and security of memory encryption and authentication. In *Proceedings. 33rd International Symposium on Computer Architecture*, pages 12 pp. —-, 2006.

[21] A Rogers and A Milenkovic. Security extensions for integrity and confidentiality in embedded processors. *Microprocess. Microsyst. (Netherlands)*, 33(5-6):398–414, 2009.

[22] M Bellare and C Namprempre. Authenticated encryption: relations among notions and analysis of the generic composition paradigm. In *Advances in Cryptology - ASIACRYPT 2000. 6th International Conference on the Theory and Application of Cryptology and Information Security. Proceedings (Lecture Notes in Computer Science Vol.1976)*, pages 531 – 45, Berlin, Germany, 2000.

[23] R Vaslin, G Gogniat, J-P Diguet, E Wanderley, R Tessier, and W Burleson. A security approach for off-chip memory in embedded microprocessor systems. *Microprocess. Microsyst. (Netherlands)*, 33(1):37–45, 2009.

[24] Common criteria. https://www.commoncriteriaportal.org.

[25] Nist computer security. http://csrc.nist.gov, Visited on May 10, 2014.

[26] Hubert Comon and Vitaly Shmatikov. Is it possible to decide whether a cryptographic protocol is secure or not?, 2001.

[27] Catherine Meadows. Open issues in formal methods for cryptographic protocol analysis. In *In Proceedings of DISCEX 2000*, pages 237–250. IEEE Computer Society Press, 2000.

[28] Lawrence C. Paulson. The inductive approach to verifying cryptographic protocols. *Journal of Computer Security*, 1998.

[29] Juan Soto. Statistical testing of random number generators. http://csrc.nist.gov/groups/st/toolkit/rng/documents/nissc-paper.pdf, Retrieved on May 30, 2014.

[30] S. Goldwasser O. Goldreich and S. Micali. How to construct random functions. *Journal of the ACM*, 33(4):792–807, 1986.

[31] M. Luby and C. Racko. How to construct pseudorandom permutations from pseudorandom functions. *Journal of Computing*, 17(2):373–386, 1988.

[32] M. Hong, H. Guo, and X. Hu. A cost-effective tag design for memory data authentication in embedded systems. In *Proceedings of the 2012 international conference on Compilers, architectures and synthesis for embedded systems*, 2012.