On Design of Memory Data Authentication For Embedded Processor Systems

by

Tao Liu

School of Computer Science and Engineering

The University of New South Wales

Sydney, Australia

A THESIS SUBMITTED IN ACCORDANCE WITH THE REQUIREMENTS
FOR THE DEGREE OF MASTER BY RESEARCH

February 2015

PLEASE TYPE

THE UNIVERSITY OF NEW SOUTH WALES
Thesis/Dissertation Sheet

Surname or Family name: LIU

First name: Tao

Other name/s:

Abbreviation for degree as given in the University calendar: MSc

School: School of Computer Science and Engineering

Faculty: Faculty of Engineering

Title: On Design of Memory Data Authentication For Embedded Processor Systems

Abstract 350 words maximum: (PLEASE TYPE)

Many designs for memory data protection are based on the cryptographic primitives that have been systematically analysed and extensively evaluated, and often provide a guaranteed level of security. However, such cryptographic primitives usually come with significant processing and resource costs and may not be suitable to embedded systems.
This thesis studies an existing design for protecting the integrity of memory data in an embedded processor system, where tag is used for data authentication. The design is highly cost efficient, consumes small on-chip resources and low off-chip memory, and offers flexibility for good trade-off between the design security and its implementation cost.
However, the design assumes that the data to be protected are random and have the uniform distribution, and the security of the design is mainly focused on the attacks with random data and tag values. Attacks with chosen values have merely been addressed. Nevertheless, the attack with chosen values can exploit the design weakness, is much stronger than the random attack, and determines the true security level of a design.
We have identified three pitfalls in this design: 1) there are some correlations between data and tag, 2) for a given data, its tag value is not distributed over the whole tag value space; the effective tag domain size for a given data is reduced and is less than the half the tag value space, and 3) the effective tag domain size varies for different data. Those weaknesses lead to the low security of the design.
To patch the loopholes, we improve the design by implementing a series of random flip function and the non-linear Galois field multiplication on the data block. We show, through the theoretical analysis and experimental demonstration, that with the design modifications the tag generated bears no correlation to its data and the tag is uniformly random over the full tag value space. The improved design has the same capability to counter attacks with chosen values as to counter attacks with the random data. Therefore, the design is much secure yet still carrying the cost effective feature of the original design.

Declaration relating to disposition of project thesis/dissertation

Tao Liu .... Signature   [illegible] .... Witness   05/02/2015 .... Date

FOR OFFICE USE ONLY   Date of completion of requirements for Award:

THIS SHEET IS TO BE GLUED TO THE INSIDE FRONT COVER OF THE THESIS

COPYRIGHT STATEMENT

Signed ....Tao Liu........

Date ....05 / 02 / 2015........

AUTHENTICITY STATEMENT

Signed ....Tao Liu........

Date ....05 / 02 / 2015........

ORIGINALITY STATEMENT

Signed ....Tao Liu..........

Date ....05 / 02 / 2015..........

Abstract

the boom of embedded systems and their wide applications, especially in the area of e-business and -service, have raised increasing concerns about their security. one of the vulnerable components in most embedded systems is memory. protecting memory data is essential to the embedded system.

many designs for memory data protection are based on the cryptographic primitives that have been systematically analysed and extensively evaluated, and often provide a guaranteed level of security. however, such cryptographic primitives usually come with significant processing and resource costs and may not be suitable to embedded systems, where resources are extremely restricted.

this thesis studies an existing design for protecting the integrity of memory data in an embedded processor system, where tag is used for data authentication. the design is highly cost efficient, consumes small on-chip resources and low off-chip memory, and offers flexibility for good trade-off between the design security and its implementation cost.

however, the design assumes that the data to be protected are random and fit the uniform distribution, and the security of the design is mainly focused on the attacks with random data and tag values. attacks with chosen values have merely been addressed. nevertheless, the chosen-value attacks can exploit the design weakness, is much stronger than the random attack, and determines the true security level of a design.

we have identified three pitfalls in this design: 1) there are some correlations between data and the tag, 2) for a given data, its tag value is not distributed over the whole tag value space; the effective tag space size for a given data is reduced and is less than the half of the tag value space, and 3) the effective tag space size varies for different data. those weaknesses lead to the low security of the design.

to patch the loopholes, we improve the design by implementing a series of random flip functions and non-linear galois field multiplication on the data blocks. we show, through the theoretical analysis and experimental demonstration, that with the design modifications the tag generated bears no correlation to its data

and the tag is uniformly random over the full tag value space. the improved design has the same capability to counter attacks with chosen values as to counter attacks with the random data. therefore, the design is much secure yet still retaining the cost effective feature of the original design.

Acknowledgements

First of all, I would like to thank my supervisor Dr. Annie Hui Guo, for her insightful and inspiring advice, kind guidance and consistent support. Without her generous help, I would not have been able to finish the research project and write this thesis.

I would like to thank my girlfriend, Miss. Mengti Sun, for her unconditional love, her support in my life and sacrificing her time to help me quickly get familiar with data processing technique. My thanks also go to Dr. Ihor Kuz and Mr. Siwei Zhuang for their invaluable feedback and advice given on my Linux programming study. These feedback and advice greatly promote my development of the experiment platform and results analysis.

It has been a great experience for me to work with some of the great minds during my candidature in CSE, UNSW. I would like to thank Mr. Mahanama Wickramasinghe, for being such a good partner to work with and a good friend of mine; Mr. Kai(Lukas) Li, for his kindly instructions on thesis preparation.

At last, I would like to thank my parents for their unconditional love and supports in my life.

Acronyms

CETD	Cost-Effective Tag Design	3
iCETD	improved CETD design	4
NIST	National Institute of Standards and Technology	4
AE	**Authenticated Encryption**	7
MAC	Message Authentication Code	8
CBC-MAC	Cipher Block Chaining MAC	8
EMAC	Encrypted CBC-MAC	9
CMAC	Cipher-based MAC	9
XCBC	Three-key XOR CBC-MAC	11
TMAC	Two-key CBC-MAC	11
OMAC	One-key CBC-MAC	12
GCM	Galois/Counter Mode	13
IV	Initialization Vector	13
XECB-MAC	eXtended Electronic Codebook MAC	15
PMAC	Parallel MAC	16
OCB	Offset Codebook	16
AES	Advanced Encryption Standard	22
PRF	Pseudorandom Function	22
PRP	Pseudorandom Permutation	23
CAR Attack	**Copy-And-Replay Attack**	34
FT Attack	**Forged Tag Attack**	34

RT Attack	**Resused Tag Attack**	34
BFP	**Bit-frequency Parity**	35
GF	Galois Field	61
GFM	Galois Field Multiplication	61
RBF	Random Block Bit Flip	62

List of Publications

- Tao Liu, Hui Guo. Dynamic Encryption Key Design and Its Security Evaluation for Memory Data Protection in Embedded Systems, in *International Conference on IT Convergence and Security* , Page 1-4, Beijing, China, 2014
- Tao Liu, Hui Guo. On Cost Effective Tag Design for Processor Memory Data Integrity in Embedded Systems , in *University of New South Wales Technical Report*, Sydney, Australia, 2014

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The last a few decades have seen great expansion of the embedded system market. From personal smart devices to industrial control equipment, from government services to military arms, the utilization of embedded systems can be found everywhere. The boom of embedded systems, especially their growing involvements in storing and processing sensitive data, has also drawn attention of adversaries. Each year, significant losses due to the system security issues have been reported, which raises increasing concerns about the embedded systems security and the importance of secure designs. Security joins performance, cost and power consumption, and forms a new dimension in embedded system design[36].

Most embedded systems use processors for computing and memory for data storage. With the increasing scale of the embedded system, the large off-chip memory to store software and data becomes indispensable. However, the off-chip memory and its link to the processor chip can be easily probed and the data stored in the memory or transferred on the communication link can be tempered, which leaves the system vulnerable to attacks.

This thesis targets an embedded system that has a single processor and memory, as shown in Figure 1.1, where the processor and memory are not on the same chip. The components on the processor chip are secure but the data on the off-chip memory and its buses are accessible to adversaries. Therefore, the confidentiality and integrity of the memory data should be protected.

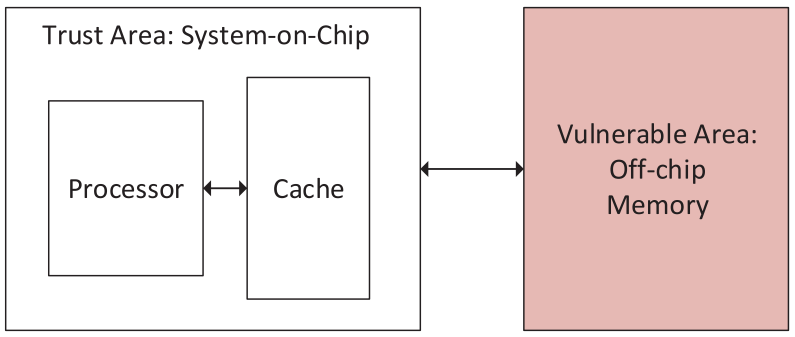


Figure 1.1: Single-Processor and Memory System

Attacks on embedded systems can be software based and hardware based. Software attacks mainly target on the operating system and application software; while hardware attacks, by physical accessing the system components (hence also known as physical attacks), can be more towards to the data stored and processed in the system.

To protect the memory data, there have been many approaches. Most of them are designed, based on the cryptographic primitives that have been systematically evaluated and their attributes and capabilities are well studied. In [53] the authors pointed out that those primitives offer a strong level of security but at a cost of significant resources or extremely degraded execution performance. Therefore, designs primarily based on the cryptographic primitives are often not suitable for embedded systems, where resources are stringent.

To reduce the resource cost and also meet the security and performance requirements, researchers have investigated customized designs. One of the related works is a cost-effective tag generation design (we name it as **Cost-Effective Tag Design (CETD)** for short in this thesis) recently proposed by Hong, Guo, and Hu [30], for memory data integrity protection in the embedded processor system. The CETD design is built upon the operations that can carry the randomness of their input data and it uses nonce to dynamically control the tag generation process. The tag generated changes with the memory data location and access time. The design allows customizations for design efficiency: it is customizable for different applications, and the on-chip cost can be minimized for a given tag size.

However, compared to the extensive cost evaluation that has been performed, the evaluation of the design security is very brief and insufficient, which motivates the work of this thesis. In this thesis, we apply an innovative approach that is based on the collisions of secret values in the system. The security of the CETD design is closely related to the two secret values, nonce and tag. The designs for the tag and nonce in CETD are therefore focused.

We evaluate the nonce generation design based on the uniqueness of its input and the randomness of its output. We found that the security of the design is slightly lower than the security against the random attack. For tags, we evaluate the tag generation design based on the tag collisions of individual data. And we identify the weaknesses of the design that can be exploited by adversaries for chosen value attacks. We propose an improved design to patch the loopholes in the system. Experiments demonstrate the improved design is much securer than the original design.

The rest of the thesis is organized as follows.

Chapter 2 lays the general background about the security and security designs, and reviews the related work on data authentication design and evaluation.

Our evaluation target system, CETD, proposed in [30] is presented in Chapter 3, where its designs on the nonce and tag generation are detailed.

Chapter 4 evaluates the tag generation in CETD. A few pitfalls in the system are identified. Both analysis and experiment show that the CETD is not secure against attacks with chosen values.

The security evaluation of the nonce design in CETD is given in Chapter 5, where security of the nonce design is modeled with two types of attacks, random attack and attack with used nonce. The security of the CETD against the attacks is analyzed. It is shown that the security of the nonce design in CETD is similar to other existing designs and is highly secure.

Chapter 6 discusses a few possible modifications to CETD, and proposes an improved CETD design (iCETD). Analytical discussion and experiments based on the National Institute of Standards and Technology (NIST) random tests, demonstrate that with the improved design, the tags generated for individual data are random and distributed over the full tag space, and the design is much securer than the original CETD design.

Chapter 7 summarizes the contributions and limitations of the thesis work, and points the possible research directions related to the memory data authentication and design evaluation for embedded systems.

Chapter 2

Background and Literature Review

This chapter provides the background and related work on design and evaluation of data integrity protection systems. Section 2.1 gives an overview of the security issues in computing systems. Section 2.2 presents research works on data integrity protection design and security evaluation .

2.1 Security and Security Design

Broadly speaking, the security of a system is its ability to protect the information and resources it possesses. For a computing system, attacks can be launched at different design levels: circuit level, micro-architecture level, operating system level, application level, network level, and information system level. Each level has different issues and requirements, and the related security designs have different focuses.

At the circuit level, where circuits are integrated onto chips, protecting the chip from environment damage and unlawful exploration is the main focus. The physical security mechanisms include the use of strong enclosures and tamper detection circuitry [38]. For example, a secure coprocessor [71] that is used to protect the encryption key in the cryptographic module zeros the plain key when the removable covers of the hardware module are opened. Such coprocessor can also be used for nonce design protection.

At the operating system level, password and access control are typically implemented. In [42], the authors presented an architectural design to isolate software applications in the system to protect them from copying and tampering each other. In the system, execution processes are restricted in separate internal compartments, a process in one compartment cannot read data from another compartment. All data that leaves the machine is encrypted.

At the networking level, many communication protocols (IPsec, TLS/SSL and SSH) [2] are proposed to ensure the data transferred secret and genuine, and the communication parties authentic.

At the micro-architecture level, attacks can target on the insecure components and execution behavior. Non-random execution behaviors are exploited by the side-channel attacks. To conceal the execution information, injecting redundant or random operations has been considered [4]. In [48], Moore et al. presented a design technique for constructing smart card functions to counter side channel attacks. To protect data on the insecure system buses and memories, encryption and authentication are utilized [21, 26, 31, 62, 67, 68].

In [9], Bellare and Namprempre introduced the concept of **Authenticated Encryption** (AE) system that combines the encryption and data authentication in a single scheme to ensure both confidentiality and integrity of the data. According to the categorization from Bellare and Namprempre, there are three ways to construct an AE scheme: Encrypt-and-Authenticate, Authenticate-then-Encrypt and Encrypt-then-Authenticate. They claimed that Encrypt-then-Authenticate has the strongest security among the three.

The review of the data integrity protection that is related to our research topic is given in the next section.

2.2 Data Integrity Protection and Security Evaluation

Data integrity is about maintaining and assuring the accuracy and consistency of data over their entire life-time. A common approach for data integrity protection is authentication. 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, commonly applied at varied communication levels, are relevant. We first survey the message authentication, then review typical designs for memory data protection; the security evaluation will be discussed at the end of this section.

2.2.1 Message Authentication

For the message authentication, the common approach is to use a Message Authentication Code (MAC) to identify a message. MAC is also called “tag” in the data authentication. 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 the MAC generation; for a given message, the MAC value is fixed; therefore, some works also refer to the stateless MAC as the 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, therefore, 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 the iterated MAC and the 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 MAC (CBC-MAC) 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. The concept of CBC-MAC is expressed in Figure 2.1, where the message consists of i blocks, with each block of a same size and the last block is padded. For each block, its value is XORed with the encryption result from its preceding block (except the first block which is directly encrypted). The encryption from the last block makes the MAC for the whole message.

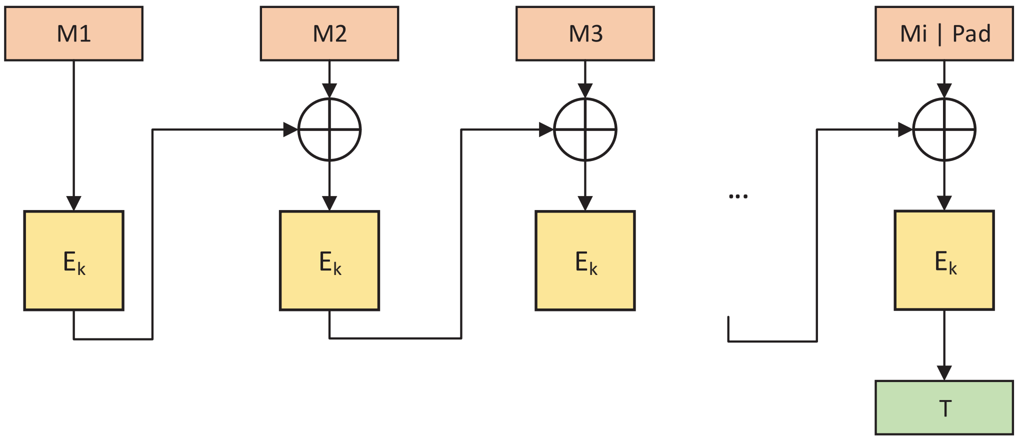


FIGURE 2.1: CBC-MAC

An initial CBC-MAC design can be found in [66]. One known attack on this design is that given a message (M) with a tag T, 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 varied sizes.

To address this issue, Petrank and Rackof proposed Encrypted CBC-MAC (EMAC) [11], which can handle messages with an arbitrary number of blocks. The design applies CBC-MAC on message blocks with one encryption key and the resulting value is then encrypted with a different key, as illustrated in Figure 2.2, where two keys, K andn $K2$ are used in the encyption.

Cipher-based MAC (CMAC) [49], shown in Figure 2.3, is another type of MAC designs for the arbitrary-length messages. Compared with EMAC, it has two further improvements: 1) it saves one cipher operation with no extra encryption

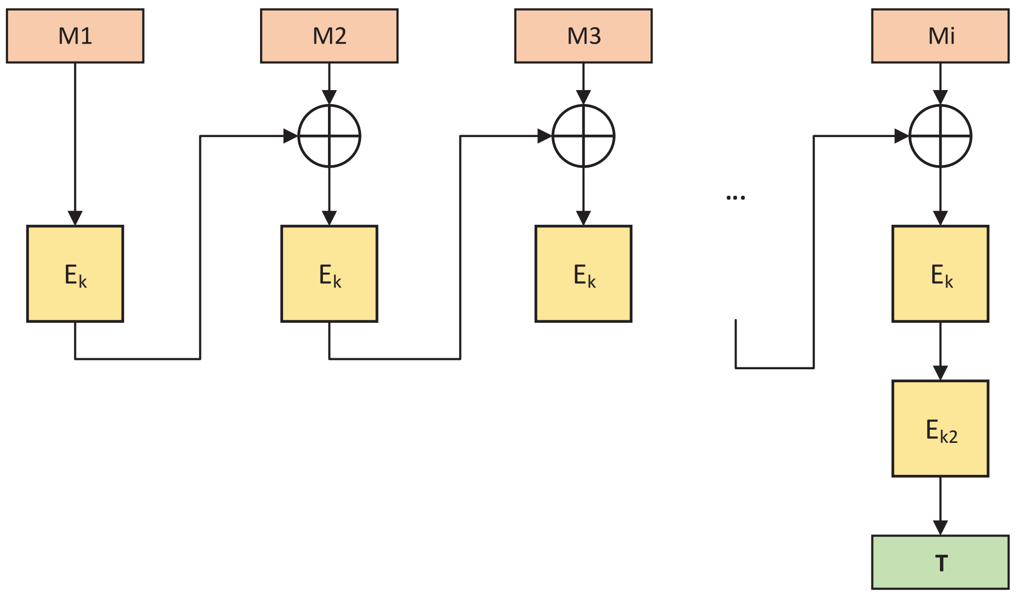


FIGURE 2.2: EMAC

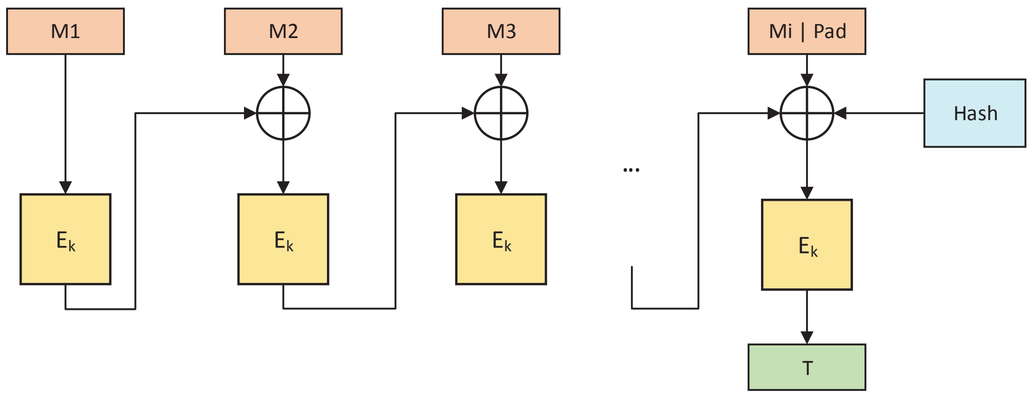


FIGURE 2.3: CMAC Class

for the last block, 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 with a shorter block size. If incomplete, the last block is then padded with constant bits for a full block length required by the block operation.

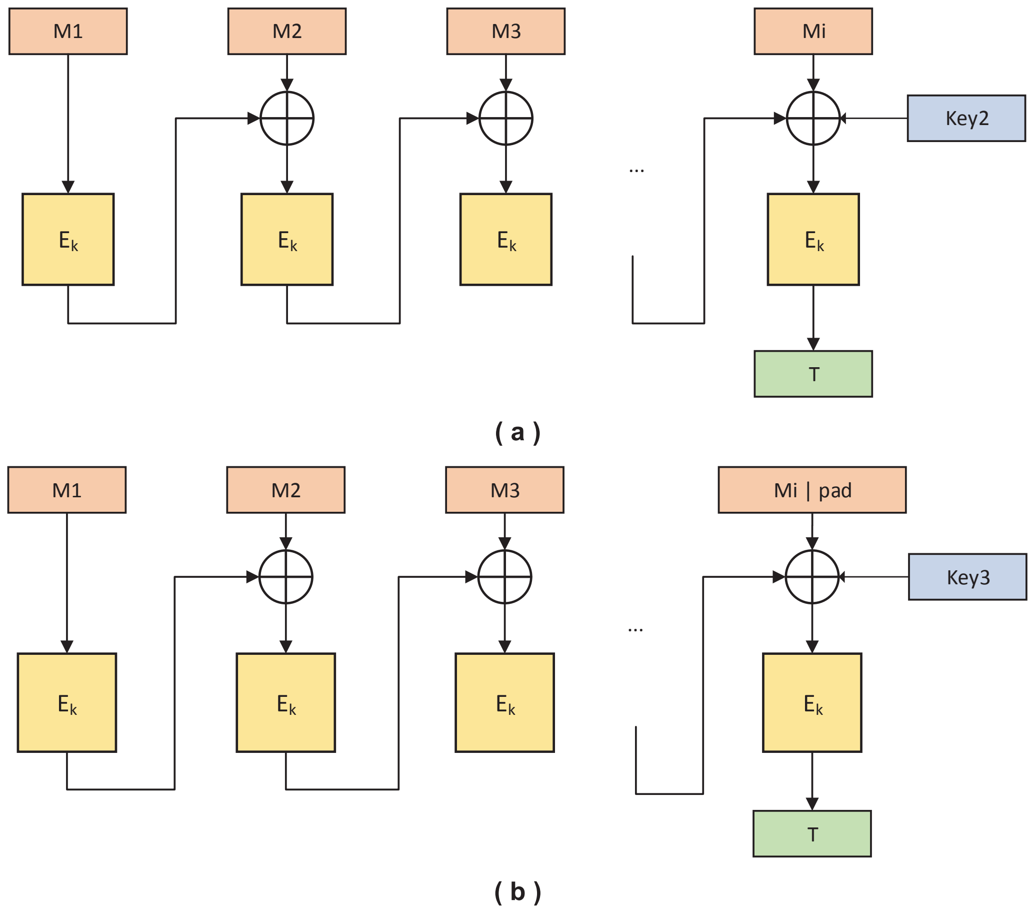


Figure 2.4: XCBC MAC Scheme: (a) last block without padding (b) last block with padding

In [12], Black and Rogaway introduced a CMAC called Three-key XOR CBC-MAC (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, as shown in Figure 2.4 (a) and (b). The authors analyzed the security of their design and provided an upper bound of success probability for the random attack; the upper bound was later tightened by Minematsu and Matsushima in [47].

Using three keys in XCBC is expensive due to key generation and storage required. Kurosawa and Iwata in [37] presented a Two-key CBC-MAC (TMAC)(Figure 2.5).

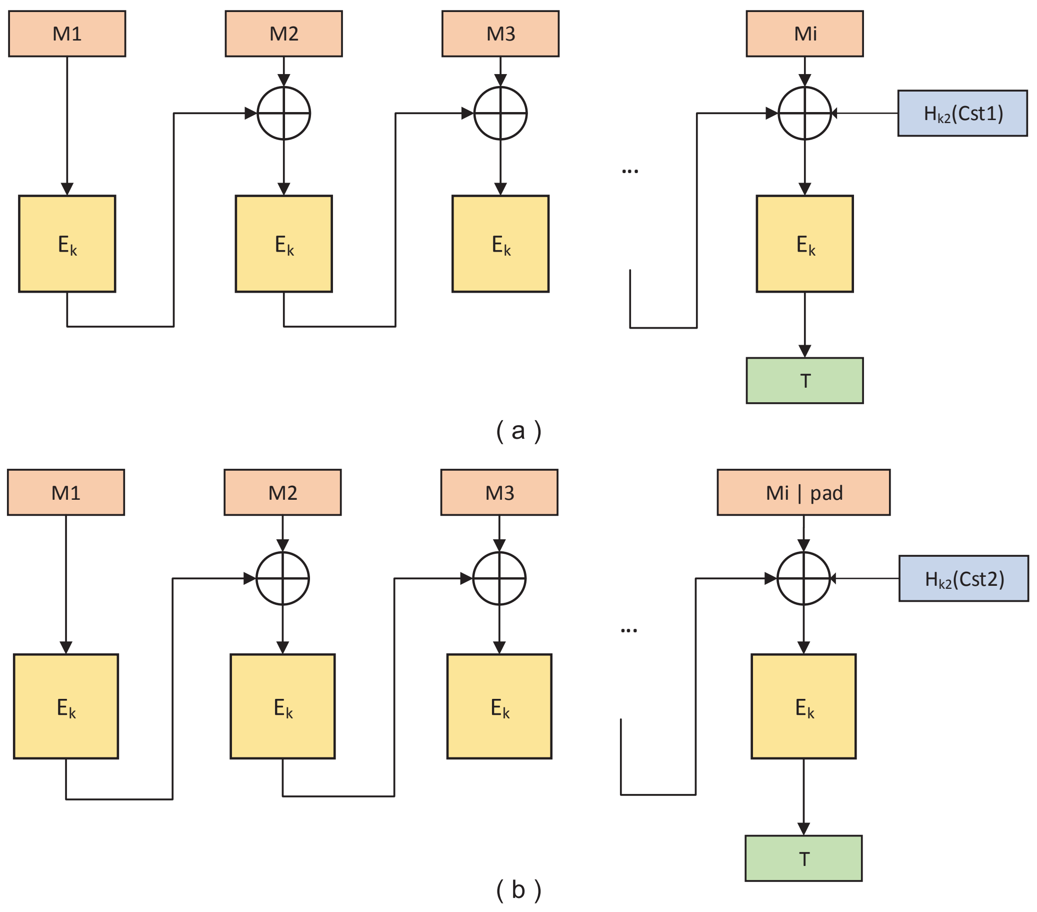


FIGURE 2.5: TMAC Scheme: (a) last block without padding (b) last block with padding

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 One-key CBC-MAC (OMAC) [32] where only one key is used (by the block cipher) and the last block is marked with a Galois Field [41] multiplication, which is easy to implement in hardware and has performance benefit if implemented in software. The structure of OMAC scheme is expressed in Figure 2.6.

The above MAC schemes are all stateless designs. McGrew and Viega in [45]

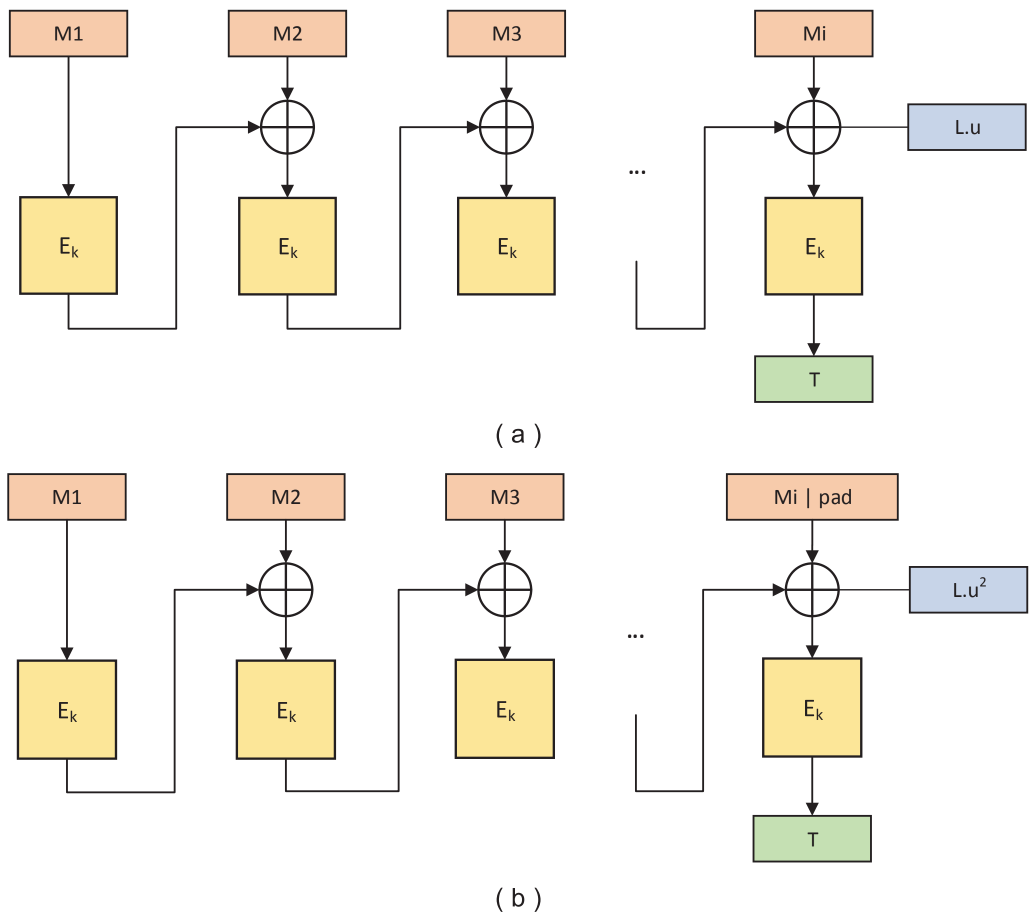


FIGURE 2.6: OMAC Scheme: (a) last block without padding (b) last block with padding. L is $E_k(0)$ and u is a constant value

introduced a stateful MAC design in an authenticated-encryption scheme, called Galois/Counter Mode (GCM) authenticated encryption. The MAC scheme of GCM uses a nonce as a state. The nonce is generated with a block cipher that takes a value (named as Initialization Vector (IV)) as input. 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 [33].

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 of a message are processed in parallel.

A simple parallel MAC scheme is XORing all the encrypted input data blocks to form the tag. However, as pointed out in [5], this scheme will generate the same result for different messages if they are made of a same set of data blocks.

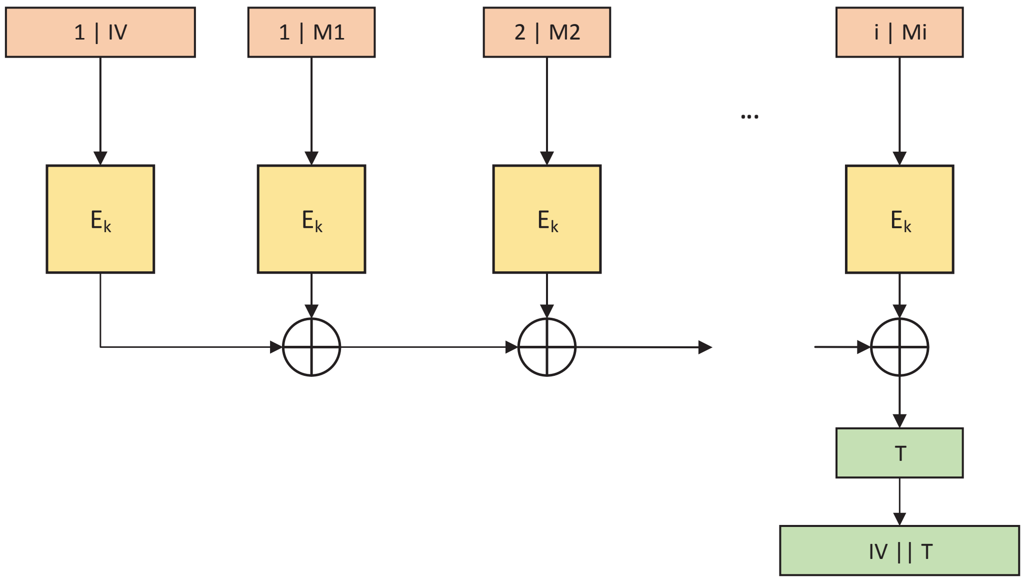


FIGURE 2.7: XOR-MAC Scheme: IV can be a counter or a random number

In [7], Bellare, Guerin and Rogaway presented a parallel MAC scheme named XOR MAC, where each block is encrypted and all encrypted blocks are XORed for the tag value, as shown in Figure 2.7. A parameter IV is used as the state in XOR-MAC and IV can be a secret counter value or random number. 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 encryption function input; therefore, the number of blocks for the message are doubled and more cipher operations are incurred.

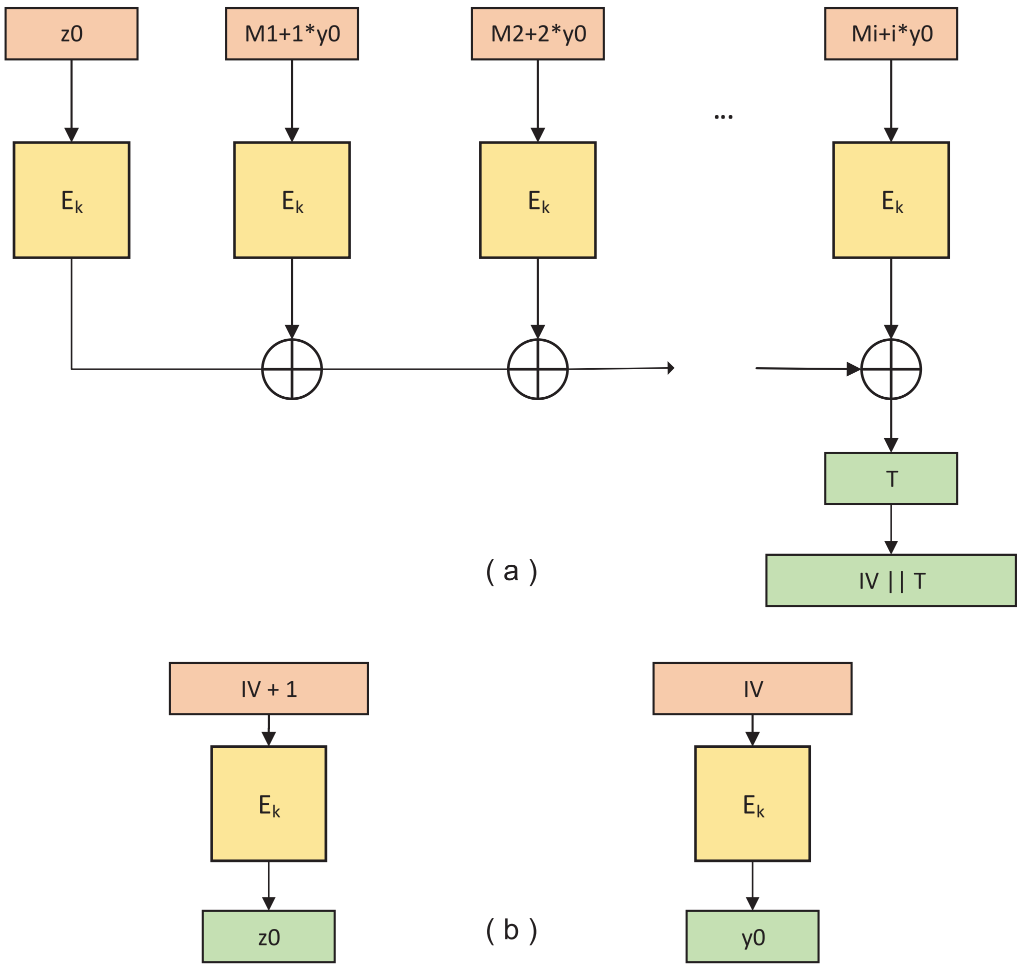


FIGURE 2.8: XECB-MAC Scheme: (a) XECB-MAC and (b) z0 and y0: IV can be a counter or a random number

To reduce the number of cipher operations, Gligor and Donescu introduced eXtended Electronic Codebook MAC (XECB-MAC) (depicted in Figure 2.8) [27], where a full block size of data are processed by each block cipher and the order of each block M_i is marked by the operation $M_i + i * y0$ (i is the input block index and $y0$ the secret one-time value).

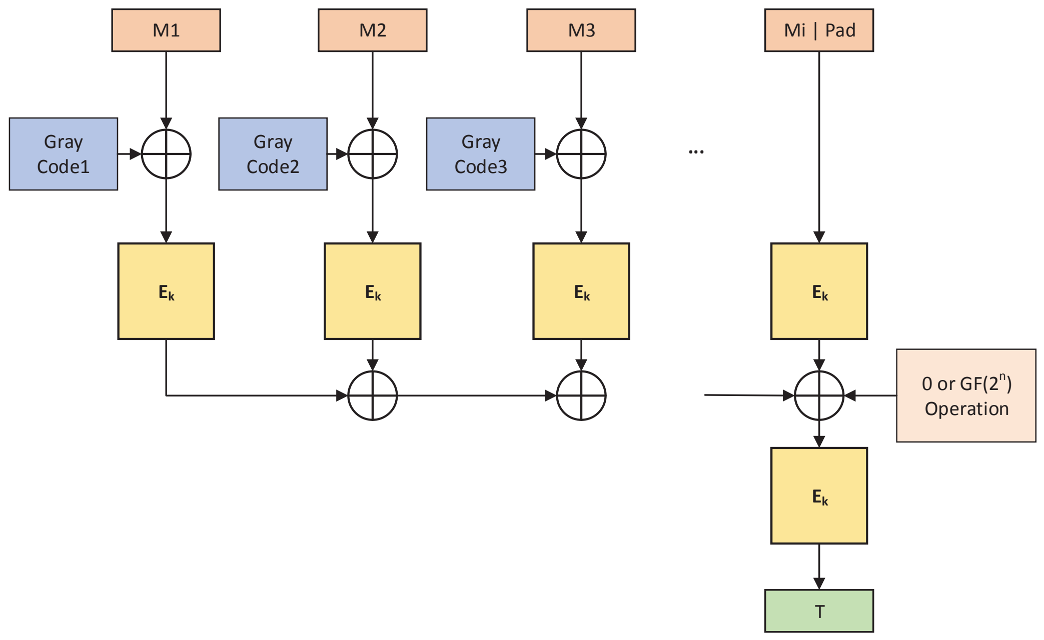


FIGURE 2.9: PMAC Scheme

In [13], Parallel MAC (PMAC) (as illustrated in Figure 2.9) was introduced. Compared with the 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 a GF (Galois Field) 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 MAC collision.

The concept of tweakable block cipher was introduced by Liskov, Rivest, and Wagner in [43]. Rogaway presented two efficient implementations of tweakable block cipher (see Figure 2.11(a)) in [54] and applied this implementation to replace the block cipher used in the original PMAC and Offset Codebook (OCB) authenticated encryption mode [56]. The advantages of constructing PMAC

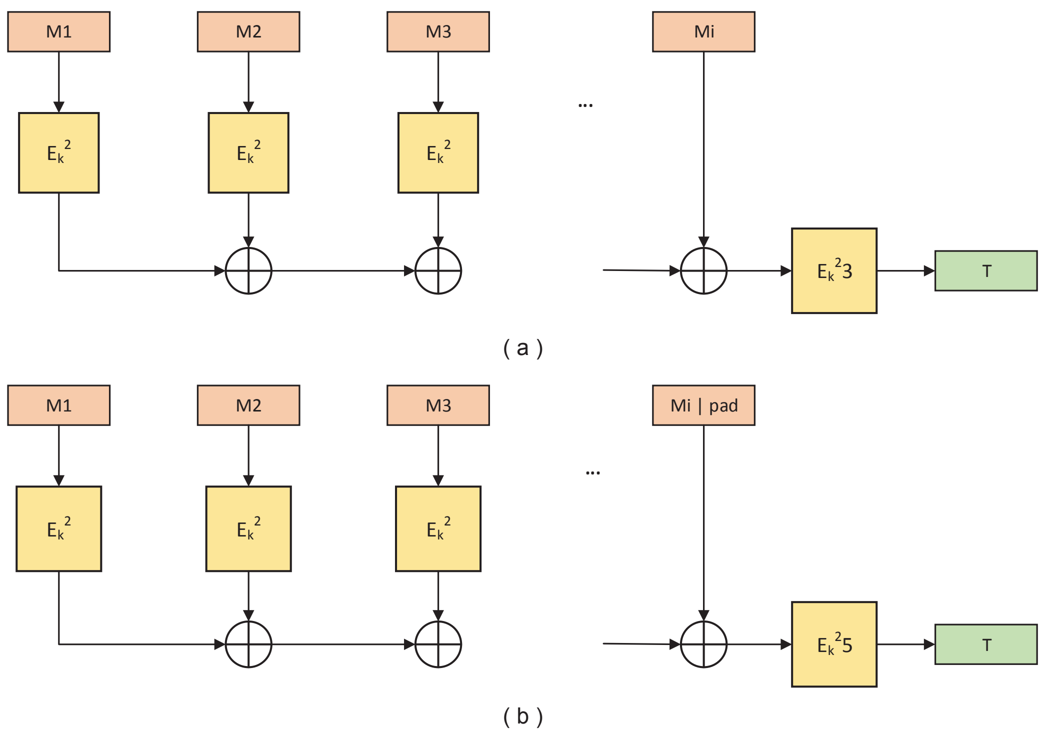


FIGURE 2.10: Tweakable Block Cipher Based PMAC Scheme

with tweakable block cipher include easier implementation and simpler security analysis. In tweakable based PMAC, input blocks processing is packaged into a tweakable block cipher E_k^2 to form the temperary output blocks. All the temparary output blocks are XORed then encrypted with another tweakable block $E_k^2 3$ for no-padding case and $E_k^2 5$ for padding case. The difference between E_k^2 and $E_k^2 3$ is expressed in Figure 2.11(b). This packaged processing simplifies the structure of the original PMAC where an input block is first XORed with $r_i \cdot L$ then encrypted. This simplification of structure also reduces the complexity of computational based security analysis on PMAC.

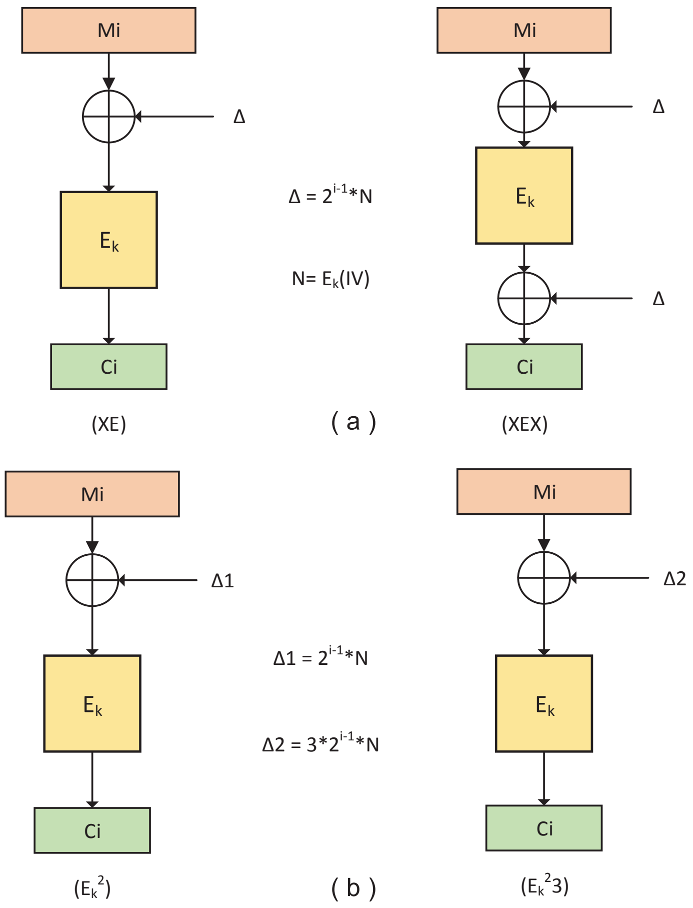


FIGURE 2.11: Tweakable Block Cipher in PMAC: (a) Tweakable Block Cipher XE and XEX; (b) E_k^2 and $E_k^2 3$

2.2.2 Memory Data Authentication

For the processor memory data authentication, the data is often of a fixed length (determined by the processor cache line size). There are two typical authentication designs: cryptographic hash function based, and MAC scheme based.

The hash-based design is the stateless scheme, which cannot effectively counter the replay attack. Therefore, memory protection systems that use hash function as integrity protection component, such as [25, 40, 62, 63], usually need to include additional components to defend replay attacks.

The MAC based scheme allows for stateful designs and tags in a large value range. Therefore, the MAC based scheme is suitable for data memory protection.

In [67], Yan et al. proposed an AE design to protect the confidential and integrity of memory data with Galois/Counter Mode (GCM) operations. A data block to be protected is divided into blocks, each of the same size as the block cipher. Each data 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 the concatenation of two parts: a major counter (64 bits) and a minor counter (no more than 8 bits). The major counter will be updated when the minor counter is overflowed.

In [57] 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. It is an Encrypt-and-Authenticate scheme, where both the tag and the ciphertext are generated on the plaintext data. Bellare and Namprempre in [9] pointed out the Encrypt-and-Authenticate design is not as secure as Encrypt-then-Authenticate design, where the data is first encrypted and the encrypted data is then tagged.

The above memory data protections are built on the crypto-primitives and have a fixed design complexity for tag generation; the design cost is high and the cost stays same even with small-sized tags. However, for many embedded systems, where resources are stringent, the cost-effectiveness is essential. In [30], the authors proposed a tag generation design that is based on a sequence of random bit-operations; the design aims for the low tag storage consumption and on-chip area cost, and can be optimized for a given tag size. We will analyse this design and improve it for a better memory data protection in this thesis.

There are also other protection designs that are not based on MAC or hash. In [22], Elbaz et al. introduced a hardware design aiming to protect both the read-only and read-write data. The structure adopted in this design is similar to Authenticate-then-Encrypt. The tag for read-only data is the address of the data, while the tag for read-write data is the address of data concatenated with a random number. Data protected is concatenated with the tag and encrypted with block cipher. The ciphertext block is sent to off-chip memory. According to the analysis in [9] by Bellare and Namprempre, this design is less secure than the design with an Encrypt-then-Authenticate scheme. Another weakness of the design is the restrict tag size; the short tags of different data can easily collide, especially for large amount of data, which increases the attack success probability.

In [64], Vaslin et al. designed a memory protection system using one-time pad (OTP) for encryption and CRC checksum module for integrity checking. For each data protected, a tuple (address, timestamp (TS), padding value (PV)) is encrypted to form an OTP. A checksum is computed with CRC (Cyclic Redundancy Check) [52] on plaintext data. This checksum is stored on-chip. The timestamp for each data is also saved on the chip. The OTP is XORed with plaintext data to form the ciphertext. The ciphertext is stored on an off-chip memory. When the ciphertext is retrieved, it is XORed with the OTP obtained with tuple (address, timestamp (TS), PV). The output of XOR is used to compute a checksum (CS2) with CRC and compared with the checksum (CS1) on-chip. The weakness of this design is that CRC is less secure compared to the hash functions such as MD5 or SHA-1, as pointed by the authors in this paper and by Elbaz et al. in [23]. To address the weakness, Crenne et al. introduced a configurable memory protection system in [16]. Instead of using OTP and CRC, the improved design uses AES-GCM to protect the data integrity and confidentiality.

Most stateful authentication designs utilize an arbitrary number called nonce.

Nonce can be static or dynamic. Being constantly changing in value, dynamic nonces provide a much stronger security protection than static nonce, and have drawn increasing attention in memory protection systems [21, 26, 31, 62, 67, 68].

2.2.3 Security Evaluation

Compared with the security designs, which are abundant and have been studied for decades, works on security evaluation are relative lacking. So far, the security evaluations basically follow a two-step approach: setting up the threat/attack model for a given design and investigating how the security measures implemented in the design can fight against the modeled attacks.

The model can be an internationally-approved set of security standards such as the Common Criteria (CC) [1], the NIST FIPS [3], or a theoretical model [19], often used in a formal 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 [15]. Issues in formal methods for cryptographic protocol analysis were discussed in [46].

The assessment of a design security against the modelled attacks can be theoretical induction [51], or statistical experiments, for example, testing randomness of a random generator [60]. In [59] and [61], NIST conducted randomness tests on the Advanced Encryption Standard (AES) candidates with the randomness examination tests introduced in [60].

The evaluation can also base on the feasibility of attacks. For example, proving that a known attack scheme is infeasible in terms of its computational complexity. If the attacking scheme is computationally intractable, the design is regarded as secure. Another evaluation scheme that has 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 [9, 55].

In [28] Goldreich et al. modelled a security design as a Pseudorandom Function (PRF) and used the distinguishability between the PRF and a random function (RF) for evaluation of 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 [44] 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 [8, 12, 47]. The distinshability metrics is also of some popularity [13].

Because the computation of success probability of a MAC design is usually effected by its design details, some researchers think that such evaluation mechanism is “complex and error-prone” [14].

To relieve the difficulties and improve the evaluation effectiveness, some researchers have attempted to abstract the computation of success probability in such a way that it is not affected by the details of the evaluated systems.

One such an attempt is the “game-playing proof” first introduced by Goldwasser, Micali [29] and Yao [69]. The approach was then adopted in the security analysis of various encryption systems like [58] and authentication systems [6]. In [35], Kilian and Rogaway improved the game-playing proof by modeling the evaluated system with program code. This improvement approach, named as the code-based game-plaing proof, motivated the development of automated security evaluation of cryptographic systems [14]. In the area of data protection, Bellare and Rogaway are the first to apply this code-based game-playing proof in the security evaluation of CBC-MAC in [10]. The approach was later used in the security evaluation of encryption and authentication systems [70].

One issue with the game-playing proof is that the evaluated system should be constructed by operations whose behaviors have been systematically analyzed, for example the cryptographic primitives, which prohibits the wide applications of the evaluation approach.

In this thesis, we evaluate the memory data authentication design proposed in [30]. Our evaluation is based on the collisions of the random values in the system, which will be discussed in the following chapters.

Chapter 3

CETD: A Cost Effective Data Authentication Design

The cost effective data authentication design was proposed by Hong et al. in [30]. The design aims to reduce the design implementation cost so that it is feasible to memory data protection in embedded systems.

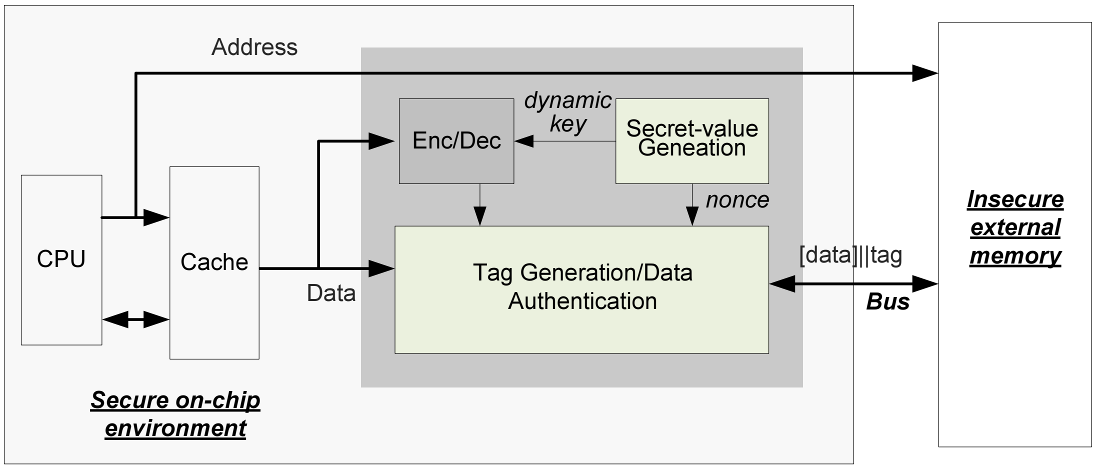


FIGURE 3.1: System Overview

3.1 Design Overview

The design is based on a stateful scheme and targets a system that has a secure processor chip and insecure off-chip memory, as shown in Figure 3.1.

The off-chip memory and its buses are accessible to attackers and the memory data can be snooped and altered by the attacker. Therefore, the plaintext data is encrypted and the encrypted data is tagged. The encrypted data 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.

The data protection design consists of three components: 1) encryption/decryption unit, 2) secret value generation unit, and 3) tag generation/data authentication unit. The block cipher is used in the encryption/decryption and tag generation process.

The secret value is stateful and carries the information of memory access time and access location. The value serves as the dynamic key to the data encryption and decryption, and is also used as the nonce in the tag generation.

The encryption/decryption uses the symmetric block cipher with the Output FeedBack (OFB) [24] mode operations and the indirect-memory encryption scheme [65] is used, as shown in Figure 3.2. For encryption (see Figure 3.2(a)), each block data is XORed with a random value and the XORs on all blocks are performed in parallel (which is fast), but the generations of the random value for each block are chained. The decryption (Figure 3.2(b)) is similar to the encryption, the only difference is that the result of XOR is the deciphered plaintext.

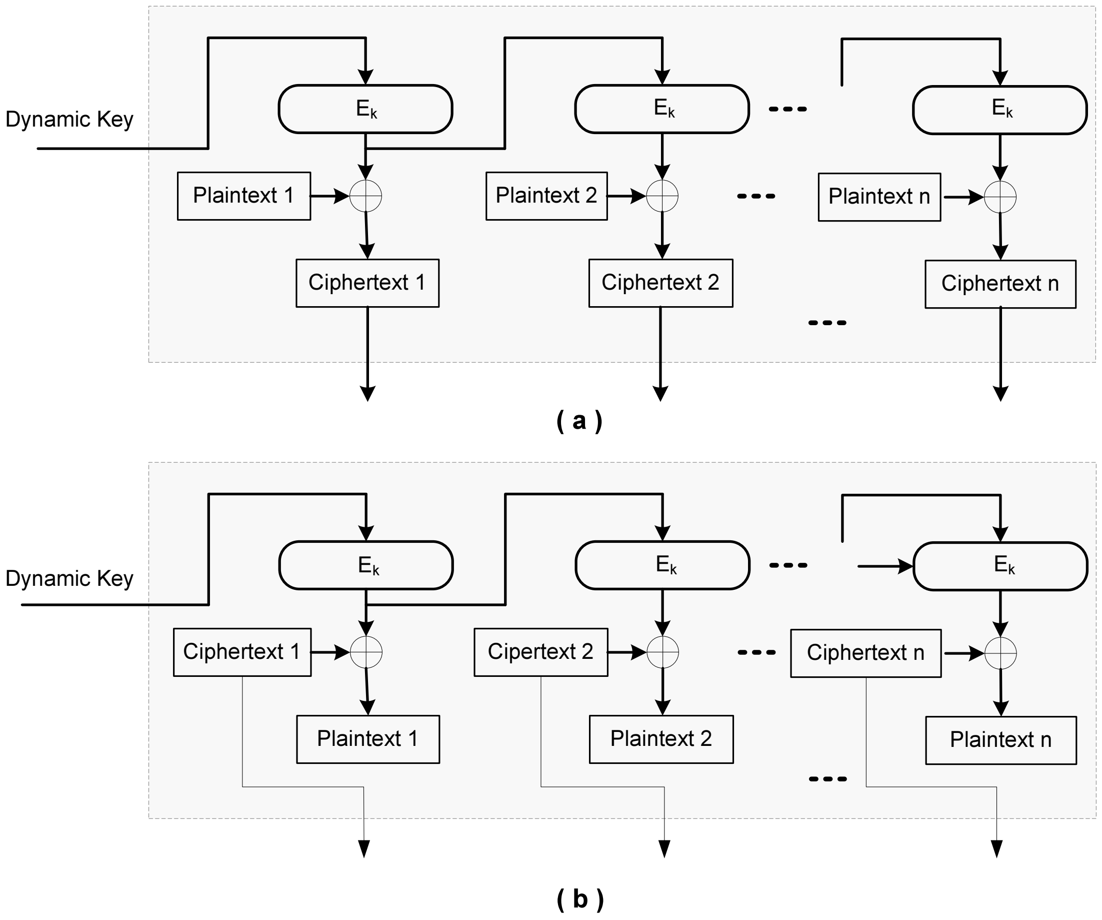


FIGURE 3.2: (a) Encryption (b) Decryption

In the enc/dec design, only one hardware encryption component is required, and it is used sequentially for all data blocks, which therefore saves the hardware cost. In addition, the design allows the overlap of the random value generation with the memory access so that the impact of the decryption on the overall system performance can be reduced.

The CETD design for the data authentication is composed by the nonce generation and tag generation. They are detailed in the following subsections.

3.2 Tag Generation

The tag generation process is given in Figure 3.3(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 block pair and swapping bit-segments between two blocks, as demonstrated in Figure 3.3(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 bit rotation in the second step is performed on individual blocks. An example is given in Figure 3.3(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 design associates the tag generation complexity with the tag size. Big tag means big block, hence the small number of blocks involved in the process, which in turn, changes the design space of the tag generation. Hong et al., the authors of the CETD design in [30], showed that for a given tag size there is an optimal design with a minimal cost. Hong et al has also demonstrated that the tag value generated is uniformly distributed over the tag value space if the encrypted data is uniformly random.

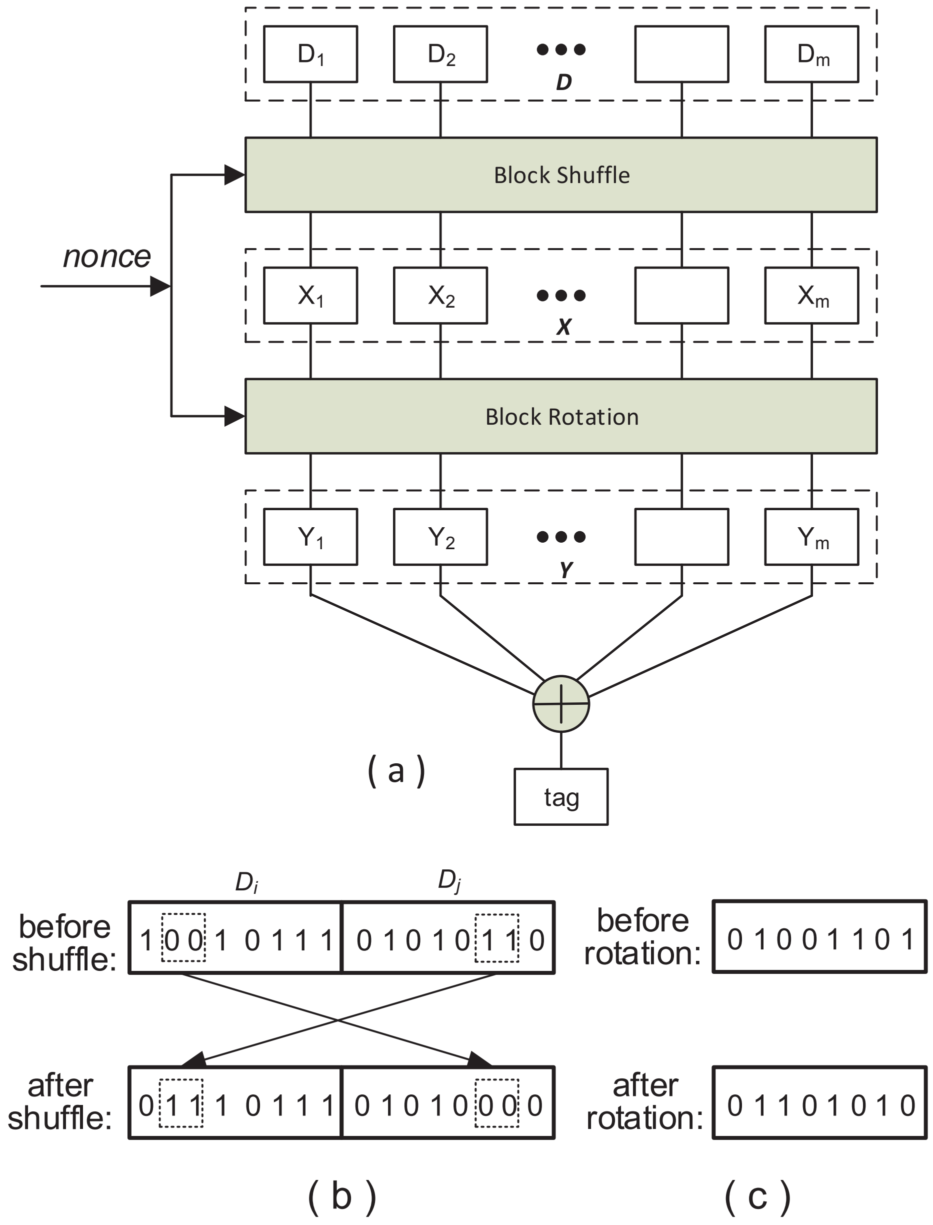


Figure 3.3: Tag Generation Process (a) diagram (b) shuffle example (c) rotation example

3.3 Nonce Design

The operations of both the block shuffle and block rotation steps in the tag generation are controlled by a nonce value. The nonce is designed to change with the memory data access time and location.

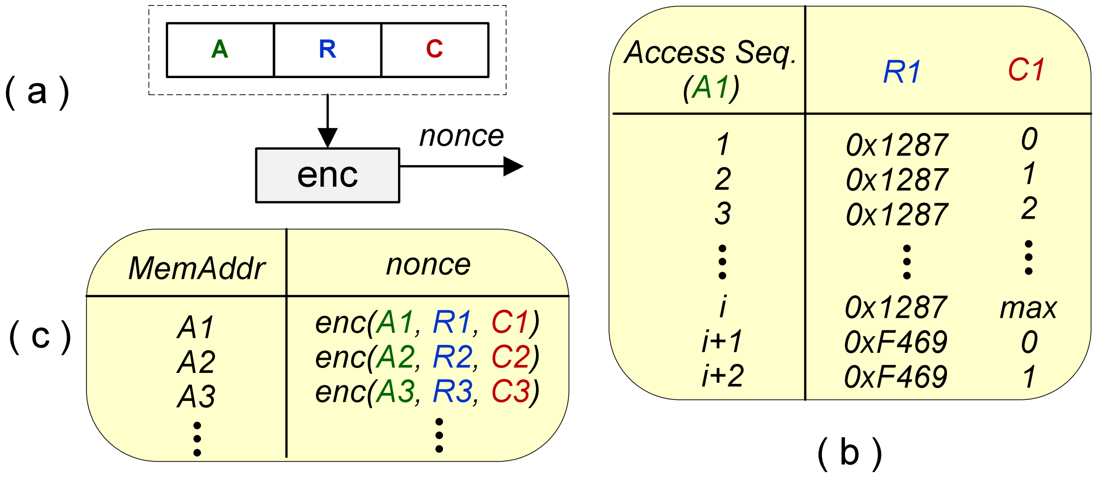


FIGURE 3.4: Nonce Generation (a) structural overview (b) change of random number (c) dedicated nonce values

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 memory address A, the associated random number R, and the counter value C as the input, as shown in Figure 3.4(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. The table in Figure 3.4(b) demonstrates how the counter value and random number are changed for a sequence of memory update operations to the memory location $A1$. Therefore, the nonce value varies with the memory location and its counter value, as specified in Figure 3.4(c). The 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.

Hong et al. proposed an implementation for such nonce generation and management on-chip. The design greatly reduces the on-chip cost, achieving up to 90% savings that would be required by a state-of-the-art design.

This thesis focuses on the security of the CETD design, which is discussed in the following two chapters (Chapter 4 and Chapter 5).

Chapter 4

Security Evaluation of the Tag Design in CETD

In this chapter, we investigate the security of of the tag design in CETD. Attacks on the tag design can be launched with random values or with chosen values.

It has been demonstrated in [30] that the big-size tag offers a high security to counter the random attack. However, no work has been done on attacks with a chosen value.

We tested the CETD design with some input data. Figure 4.1 gives the number of distinct tag values collected in the experiment for each input. The number of distinct tag values shows the size of effective tag space. From the experiment, we can see that some data have smaller effective tag space sizes than the others. This invariance of effective tag space sizes indicates that the system is not equally secure for different data. For an attack with some judiciously-chosen data, a high success probability exists. Therefore, we focus on the chosen-value attacks.

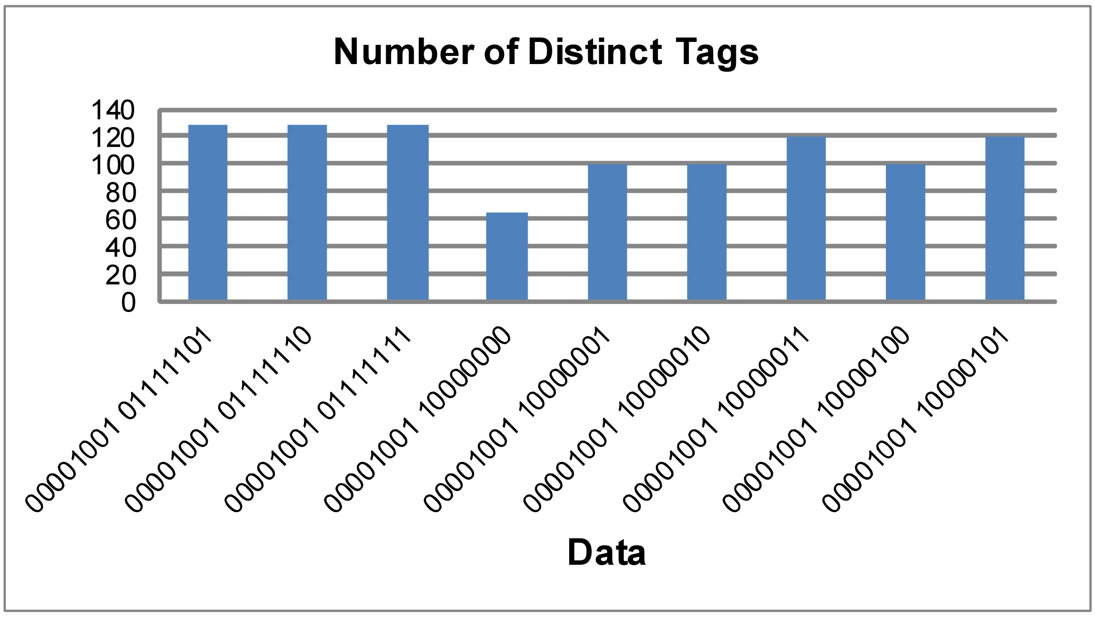


Figure 4.1: Experiment Data: Number of Distinct Tag Values of Different Data

As a contribution, we identify three pitfalls in this design:

- There are some correlations between data and tag.
- For a given data, its tag is not distributed over the whole tag value space; the effective tag space size for a given data is reduced.
- The effective tag space size varies from data and data.

We show that CETD design is not as secure against the chose-value attacks as against the random attacks.

The chapter is organized as follows.

The security weakness of CETD is discussed in Section 4.1. Our experimental verification is expressed in Section 4.2. The chapter is concluded in Section 4.3.

4.1 Analysis

In [22], Elbaz et al. proposed a data-tampering model that consists of three types of attacks: replay, splicing and spoofing. With the replay attack, the current data is replaced with an old valid data of the same memory location; while in the splicing attack, a copy of valid data from other location is used; for the spoofing attack, instead of using data obtained from the system, it fakes the data with a new one. This model was also mentioned in [20, 21, 23] and adopted in the security evaluation of data integrity protection systems in [30, 39, 57, 64].

Here we view attacks from the data authentication perspective and categorize attacks into different three groups:

- **Copy-And-Replay Attack** (CAR Attack) , where the attacker replays a copy of a valid data/tag pair from a different memory location or the same memory location but of a previous time spot[1];
- **Forged Tag Attack** (FT Attack) , where the data used by the attack may not be an existing or old data and the related tag is forged;
- **Resused Tag Attack** (RT Attack) , where the data used by the attack may not be an existing or old data and the tag of the current valid data is reused.

The attacks can escape the integrity checking if 1) the true data in the memory have a high tag collision rate for different nonce values (CAR and RT attacks); and/or 2) the tag of the true data has a high collision with a tag from a fake data for a same nonce (RT attack).

[1]Replaying previous data from the same location is commonly called replay attack, which is part of CAR attack we defined here.

Therefore, 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.

To facilitate the description, 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**, $F_{set}(D)$ for **set-bit frequency of data** D and $F_{unset}(D)$ for **unset-bit frequency**. The frequency is either odd or even, which is defined as the **Bit-frequency Parity** (BFP).

Let us consider an m-block input data D; each block is of n bits (namely, the tag size). If there are k set-bits in D, then the *unset-bit frequency* of D is $F_{unset}(D) = mn - k$.

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 block XOR operation in the design takes the value that has the same bit frequency as the input data. That is

$$F_{set}(SR(D)) = F_{set}(D), \tag{4.1}$$

where $SR(.)$ represents the block shuffle and bit rotation operations in the CETD.

We use $\bigoplus()$ to denote the XOR of m blocks. The tag of CETD is

$$tag = \bigoplus(SR(D)). \tag{4.2}$$

For a D of k set-bits, its possible tag values can be derived as follows.

When $k = 0$, the effective tag space (τ_0) of D only contains one value that consists of all unset-bits, and the size of the effective tag space τ_0 is

$$|\tau_0| = 1.$$

Here we use symbol τ_k for effective tag space and $|\tau_k|$ for its size; k is the number of set bits in the input data D.

When $k = 1$, there is one set-bit in the tag. The size of the effective tag space is

$$|\tau_1| = \binom{n}{1}.$$

When $k = 2$, the two set-bits can be part of the tag, or they can be canceled after block shuffle, bit rotation and XOR operations, resulting in zero set-bits in the tag, which is τ_0. Therefore, the size of the effective tag space is

$$|\tau_2| = \binom{n}{2} + |\tau_0|.$$

Similarly, when $k = 3$, the three set-bits can be part of the tag, or two of the set-bits are canceled and only one set-bit is left in the tag (equivalent to the case of one set-bit tag). The size of the effective tag space is

$$|\tau_3| = \binom{n}{3} + |\tau_1|.$$

In the same way, we can obtain the effective tag space size for $k \leq n$:

$$|\tau_k| = \sum_{i=0}^{\lfloor k/2 \rfloor} \binom{n}{k - 2 * i}, \qquad \text{k} \leq \text{n} \tag{4.3}$$

$|\tau_k|$ increases with k.

When $k = n - 1$,

$$|\tau_{n-1}| = \binom{n}{n-1} + |\tau_{n-3}| + \cdots + |\tau_1| = 2^{n-1}, \tag{4.4}$$

and when $k = n$,

$$|\tau_n| = \binom{n}{n} + |\tau_{n-2}| + \cdots + |\tau_0| = 2^{n-1}. \tag{4.5}$$

When $k = n + 1$, some set-bits in the m-blocks to the XOR operation will have a same bit position, and they will be cancelled by XOR operation and the tag will have $n - 1$ set-bits; therefore, the effective tag space size is

$$|\tau_{n+1}| = |\tau_{n-1}| = 2^{n-1}.$$

When $k = n + 2$, two set-bits should be cancelled, and

$$|\tau_{n+2}| = |\tau_n| = 2^{n-1}.$$

Following the same trend, we have the effective tag space size for $n \leq k \leq n(m-1)$:

$$|\tau_k| = 2^{n-1}, \qquad n \leq k \leq n(m-1). \tag{4.6}$$

But when $k > n(m-1)$, there are $mn-k$ unset-bits, which is equivalent to the case $n-(mn-k)$ set-bit cases and can be written

$$|\tau_k| = |\tau_{k-n(m-1)}|.$$

Putting together, we have the effective tag space size for a data D with k:

$$|\tau_k| = \begin{cases} \sum_{i=0}^{\lfloor k/2 \rfloor} \binom{n}{k-2*i}, & \text{if } 0 \le k \le n; \\ 2^{n-1} & \text{if } n \le k \le (m-1)n; \\ \sum_{i=0}^{\lfloor (mn-k)/2 \rfloor} \binom{n}{n-2*i} & \text{if } (m-1)n \le k \le mn. \end{cases} \tag{4.7}$$

From the above analysis, we can observe the following weaknesses of the CETD design:

Observation 1: *The effective tag value space for a given data input is smaller than the full tag value domain and the effective tag space varies with different input data. The maximal effective tag space size is just half of the full tag value space.*

When set-bits or unset-bits are especially dominant in the data, the related number of distinct tags is even much lower than the half of the full tag value space, hence the related tag collision probability is high, as illustrated in Figure 4.2 for the 16-bit data and 8-bit tag. In the extreme case, when the data is of all 0's or all 1's, the effective tag space is reduced to only one value and there is a 100% success if attacking with such a known value/tag pair.

Based on Observation 1, one can launch a CAR attack (Copy-And-Replay attack) by choosing an existing data-tag pair or instrument an attack with a new data (RT attack) that has a small effective tag space for a high attack success probability.

It must be pointed out that some data, such as all-set-bit (or all-unset-bit) value, though offering high success probabilities, may not be the choice of the attacker.

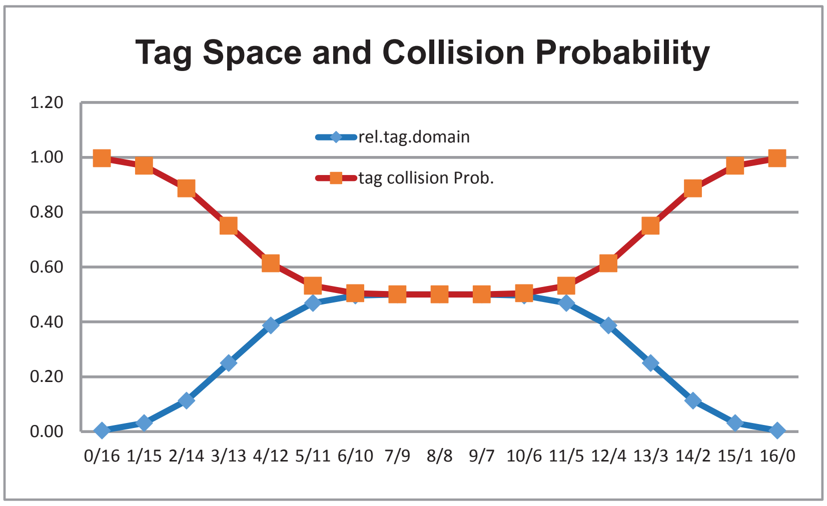


FIGURE 4.2: Reduced Effective Tag Space and Increased Tag Collision Probability

Observation 2: *For CETD, a data and its tag have a same bit frequency parity.*

If we divide the input data set into two groups: one group, D_{odd}, with an odd bit-frequency and another group, D_{even}, with an even bit-frequency, the tag set, τ_{odd}, for D_{odd} will never collide with the tag set, τ_{even}, for D_{even}, which can be expressed as

$$D_{odd} \cup D_{even} = 1 \quad D_{odd} \cap D_{even} = \Phi$$

$$\tau_{odd} \cup \tau_{even} = 1 \quad \tau_{odd} \cap \tau_{even} = \Phi.$$

We use the 16-bit data and 4-bit tag to demonstrate. Data $D1 = 0000100100001000$ will have a tag that consists of an odd number of set-bits since D has 3 (odd number) 1's, and data $D2 = 1000100100001000$ will have a tag of even number of 1's. Both tags of $D1$ and $D2$ will be certainly different. The Observation 2 shows that when forging a tag for a given data, the attacker only needs to consider half of the full tag domain value and this half tag domain consists of values that have the same BFP (bit frequency parity) as the data. By choosing a value from the space, the attack success probability is doubled as compared with the random attack.

As can be seen, the CETD design leaves some pitfalls for chosen-value attacks.

4.2 Experiments and Results

We have built a simulator to simulate the replay and splicing attacks on CETD. The simulator is written in C. Simulations are conducted on UNIX x64 platforms. Due to the resource restriction, the experiments are only performed for two limited input data sets: *full 16-bit data set* that covers all 16-bit values, and *partial 32-bit data set* that only includes values ranging from 0 to 65535 in the 32-bit data space. The tag length for both two input sets is also set to a small 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. Figure 4.3 shows the experimental setup.

We first examine the number of distinct tags (namely the effective tag space size) of a data in the 16-bit input data set for the replay and splicing attacks. For demonstration, Figure 4.4 shows the effective tag space size of the first 100 input

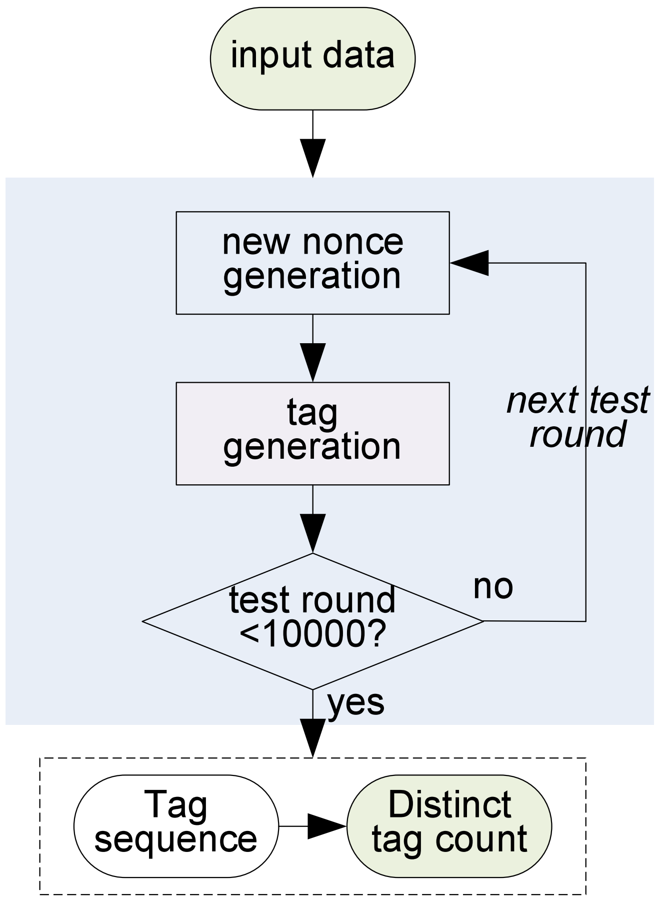


FIGURE 4.3: Experimental Setup

values in 1000-round tests, as shown in Figure 4.4 (a) and the first 600 input values in 10000-round tests (Figure 4.4 (b)). In the figure, the x-axis represents the input data in the form of decimal value and the y-axis is the effective tag space size. As can be seen from Figure 4.4(a), the effective tag space size varies for different input data, and this situation is not changed with different test rounds (as demonstrated in Figure 4.4(b)), which demonstrates a convergent result.

Table 4.1 shows the statistic information of the effective tag space size on all data values with the test rounds of 1000 and 10000. The maximum, minimum, variance and the average effective tag space size are, respectively, given in the last four columns.

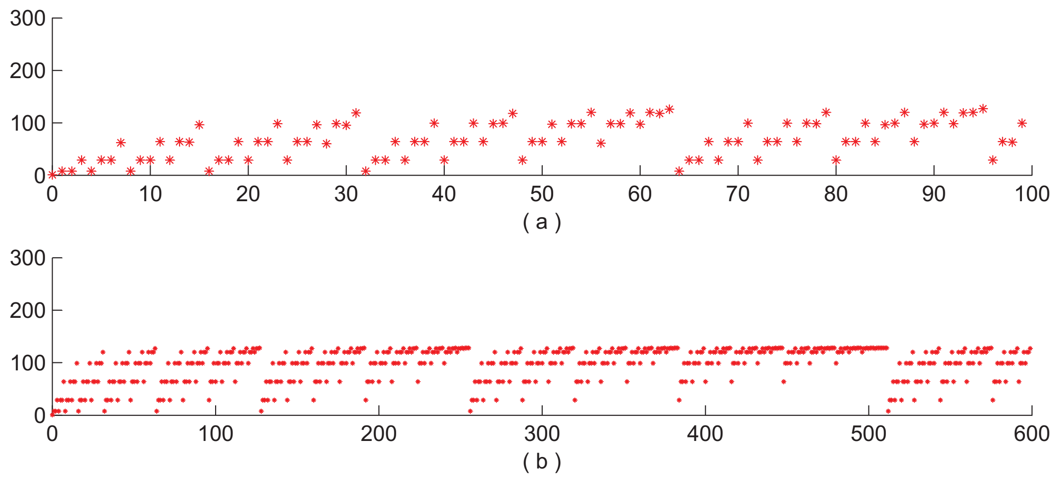


FIGURE 4.4: Effective Tag Space Size for Individual Input Data(the 16-bit input case): (a) first 100 values of 1000-round case (b) first 600 values of 10000-round case

TABLE 4.1: Effective Tag Space Size: 16-bit data case

test rounds	maximum	minimum	variance	average
1000	128	1	148.91	123.2
10000	128	1	149.22	123.56

From Figure 4.4 and Table 4.1 we can see that

- the effective tag space size of an input data is effected by the data value to be tagged;
- the tag of an input data is not distributed uniformly (variance is as big as 148.19 in 1000-round case and 149.22 in 10000-round case), and
- the maximum tag space size among all input data is no more than half of the full tag domain size (256).

To see the observations are still true for different data size, we also simulate replay and splicing attacks on 32-bit input data set. Figure 4.5 shows the effective tag space size of first 600 input values in 1000-round tests. The statistic information

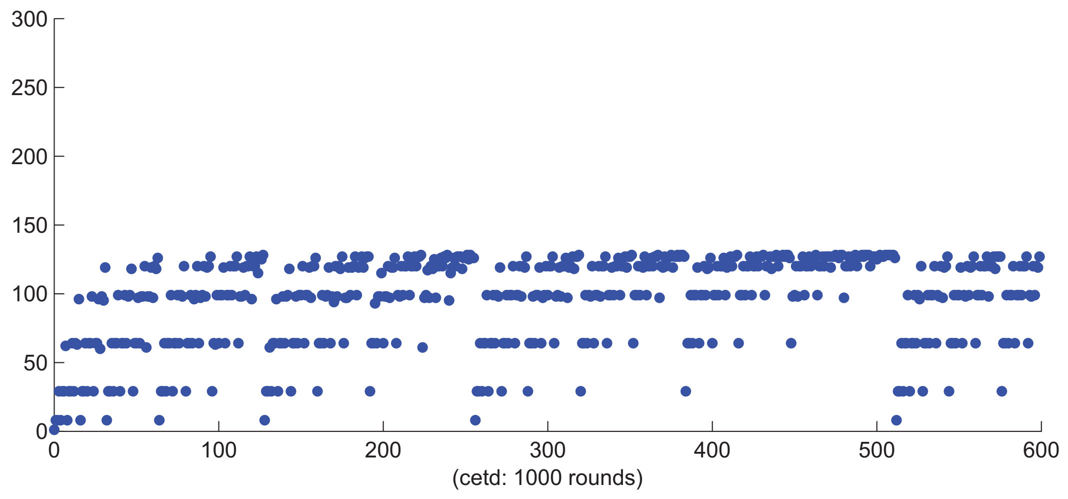


FIGURE 4.5: Effective Tag Space Size for Individual Input Data: 32-bit data case

of the effective tag space size of all input data in the 32-bit input set is given in Table 4.2. From Figure 4.5 and Table 4.2 we can see that the results of 32-bit data is same as the results of 16-bit data.

TABLE 4.2: Effective Tag Space Size: 32-bit data case

test rounds	max	min	var	aver
1000	128	1	148.48	123.2

The experiment verifies the security weakness of the CETD design that we analyzed in Section 3.

4.3 Summary

In this chapter, we studied the tag generation design for memory data integrity protection in the embedded systems. We evaluated the CETD design that is cost effective and can be used for cost-restrained embedded systems. Both our analysis and experiments showed that for CETD the effective tag space size for each data is no more than half of the full tag domain size and this size is effected by the data value. Therefore, the design has high tag collision rates for some data values and exposes a great risk if such data were used in the attacks.

Chapter 5

Security Evaluation of the Nonce Design in CETD

For the whole system secure, it is required the nonce used in the system secure. In this chapter, we evaluate the nonce design in CETD and compare it with a state-of-the-art design.

Our main contributions are:

- we presented a novel quantitative security evaluation approach that is based on both the randomness and the uniqueness of nonce values, and
- we evaluated the security of the nonce design in CETD [30] and compare its security with a design adopted in [67]; we found that the design by [67] is more secure than that by CETD for a limit number of nonces to be used; but when the number of nonces used exceeds this limit, the design in CETD becomes better.

The rest of the chapter is arranged as follows.

The security of the nonce design in CETD is evaluated in Section 5.1. The experimental verification is given in Section 5.2. The chapter is concluded in Section 5.3.

5.1 Analysis

Nonce is used as the state in the tag generation in CETD. The nonce is secure if it is random and unique.

Attacks on the nonce design can be basically divided into two types: 1) **random attacks**, where for a forged data, the attacker randomly chooses a nonce value to generate its tag and try to pass the data authentication; and 2) **used-nonce attacks**, where the attacker has cracked a nonce based on previously observed tag generation with some techniques, such as side-channel analysis of system internal operations, and uses the nonce as the current nonce for a data block.

For the random attack, the design security is determined by the randomness of the nonce. We use the maximum probability of all nonce values for its randomness. If the nonce is purely random, its values should be uniformly distributed and all values have a same probability.

Here we assume a typical block cipher, e.g. AES [17], is used in the design.

A block cipher is a deterministic algorithm that takes a fixed-length input (i.e., the plaintext data) and generates, with a secret key, an output (ciphertext data) of the size same as that of the input. The size of the cipher input and output is also regarded as the **block cipher size**.

The block cipher (here generally denoted as $\mathrm{E}_k()$) has two main features: bijective transformation and output uniformly random. With the bijective transformation, each input of the cipher maps uniquely to one output; namely, $\mathrm{E}_k(\mathrm{a}) \neq \mathrm{E}_k(\mathrm{b})$, if $a \neq b$; if $\mathrm{E}_k(\mathrm{a})$=$\mathrm{E}_k(\mathrm{b})$, a and b must be the same. The output uniform randomness means all values in the output domain have an equal chance to be the cipher result.

When the block cipher is used in the nonce generation, the nonce size is the block cipher size, as shown in Figure 5.1, where K is the block cipher size, and A, RN and C are the address, random number and the counter value associated with a memory data. In CETD, the counter size and random number size can be customized (one of the main features in the CETD design), but the total input size is fixed, namely

$$K = S_A + S_{RN} + S_C,$$

where S_A, S_{RN}, and S_C are, respectively, the size of A, RN, and C.

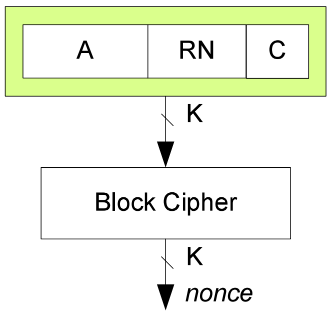


FIGURE 5.1: Nonce Generation in CETD

With the uniform distribution of the cipher output, the probability of a randomly-picked value that happens to be the nonce of the targeted cipher-data, $P_{(r-attack)}$, is

$$P_{(r-attack)} = \frac{1}{2^K}, \tag{5.1}$$

where K is the nonce size. As can be seen, the larger the nonce size, the lower the success probability of the random attack.

For the used-nonce attack, the success of the attack is relying on whether the current nonce is same as the used nonce, which is not only determined by the randomness of the cipher output but also decided by the output uniqueness. Since the output of the block cipher is unique if its input is unique, the uniqueness of nonce is related to the uniqueness of the cipher input. The success of a used-nonce attack is only possible when the input is not unique. Therefore, **we evaluate the security of the nonce design against the used-nonce attack based on two factors: the input uniqueness and the output randomness of the block cipher**.

We use the probability of input collision, $P_{(i-colli)}$, to measure the uniqueness of inputs. Lower $P_{(i-colli)}$ means higher uniqueness. According to the Pigeonhole principle [34], when the length of input sequence exceeds the size of the input value space, there will always exist collisions and $P_{(i-colli)} = 1$.

For the randomness of cipher output, we similarly use the maximum probability, $P_{(o-rand)}$, of all possible values. The success likelihood of a used-nonce attack can be defined as

$$P_{(u-attack)} = P_{(i-colli)} * P_{(o-rand)}, \tag{5.2}$$

where $P_{(i-colli)}$ serves as a “gating function” to relate the success possibility of the used-nonce attack to the nonce randomness. If $P_{(i-colli)} = 0$ (namely, no input collisions), there are zero chances of attack success even though the outputs are random.

There are three input components, address A, random number RN, and counter value C to the block cipher; the input values are unique if any one of the three is unique. In our target system (as shown in Figure 3.1), the memory address is observable to the attacker. Therefore, we assume **the attacker will apply the used-nonce attack only on a same memory location** since nonces for different locations are always different and attacks with a nonce of different memory location will never succeed. For a fixed memory location, we derive $P_{(o-rand)}$ in Formula (5.3) and (5.4) as below.

Given the output uniform distribution of the block cipher, we have

$$P_{(o-rand)} = 1/\psi, \tag{5.3}$$

where ψ is the total number of possible nonce values that can be used for the used-nonce attack. For a given memory location, the number of unique inputs is $2^{(K-S_A)}$ (K is the block cipher size, also the nonce size, and S_A the size of memory address); hence, there are $2^{(K-S_A)}$ possible unique nonce values the attacker can use. Therefore,

$$P_{(o-rand)} = 1/2^{(K-S_A)}. \tag{5.4}$$

With a fixed A, for two cipher inputs, if the related random numbers are identical and the counter values are the same, the inputs will collide. Therefore, the input collisions are closely related to whether there are random number collisions and counter value collisions, which are in turn related to the length of nonce sequence generated, hence the sequence of random numbers and the sequence of counter values used. Figure 5.2 shows simulation results on a design of 128-bit nonce and 2^{18}-nonce space. The two plots in Figure 5.2 show the percentage of nonce collisions over the nonce sequence length for two cases: with and without random number in the cipher input. In the case of without random number (denoted by "w.o. RN" in the figure), when the nonce sequence length is smaller than the maximum counter value 2^{18}, there are no collisions. However, when the random number comes into play (labelled as "with RN"), nonce collide and the percentage of nonce collision exceeds 50% even for a relatively short sequence of nonce.

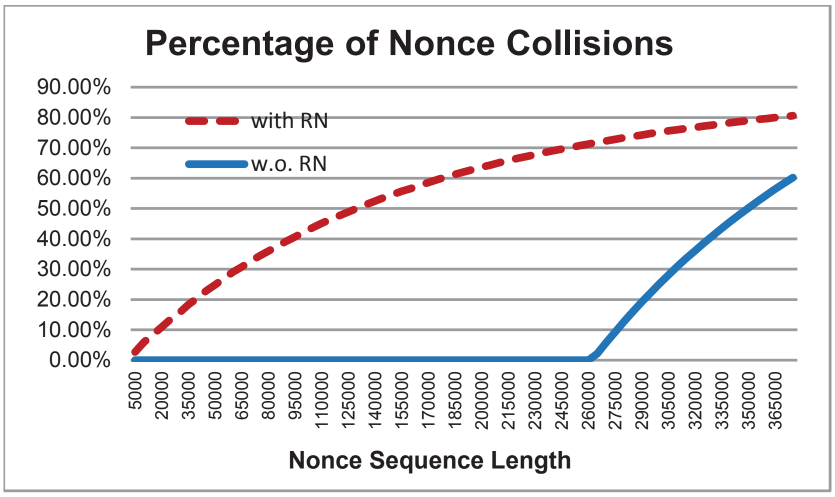


FIGURE 5.2: Experiment Data: percentage nonce collisions over nonce sequence length

Therefore, Formula (5.2) can be written as

$$P_{u-attack} = \begin{cases} 0 & S_{rn} = 0, \quad L \leq 2^{S_{ct}} \\ P_{(i-colli)} * 1/2^{(K-S_A)} & otherwise, \end{cases} \tag{5.5}$$

where $0 < P_{(i-colli)} \leq 1$ and L the length of nonce sequence.

Formula (5.5) shows that without the random number, there are no input collisions if the nonce sequence length is less than $2^{S_{ct}}$ (the counter value space). In this case, the success probability of the used-nonce attack is 0.

It is worthy to note that, with the presence of input random number, in contrast to the random attack, the success probability of the used-nonce attack changes with the memory space size, as is plotted in Figure (5.3). On most of cases, it is higher than the probability of the random attack, and the bigger the memory address space, the higher the success probability of the used-nonce attack.

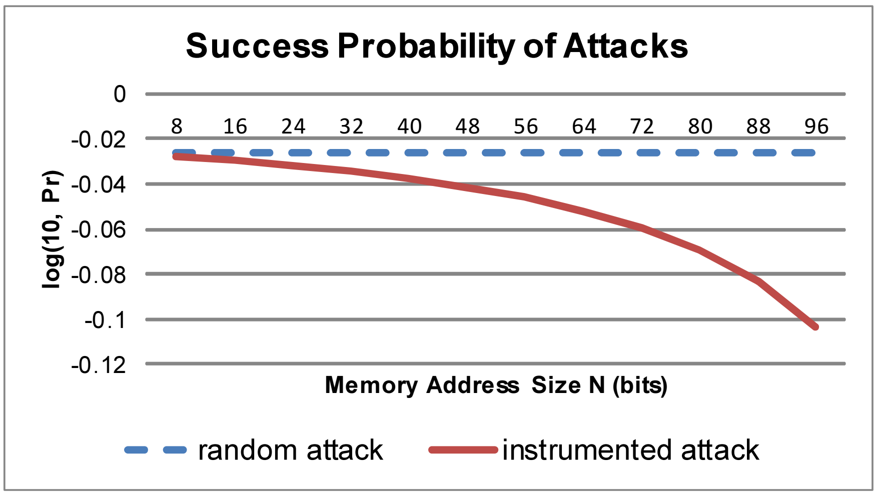


FIGURE 5.3: Success Probability of Two Attacks with Varying Memory Size

5.2 Experiments and Results

For the security evaluation, we built a software platform, where AES is used to generate nonce and the design can be easily changed for different configurations (namely, different combination of the address size, random number size and counter size).

We test nonce sequences with varied memory address spaces; for a given memory location, we generate a set of nonce sequences of different lengths. Each sequence is generated for a fixed counter size (hence a fixed random number size), as shown in the shaded blocks in Figure 5.4. An in-house tool, written in Perl, is used to analyse each nonce sequence and extract the related nonce collision information.

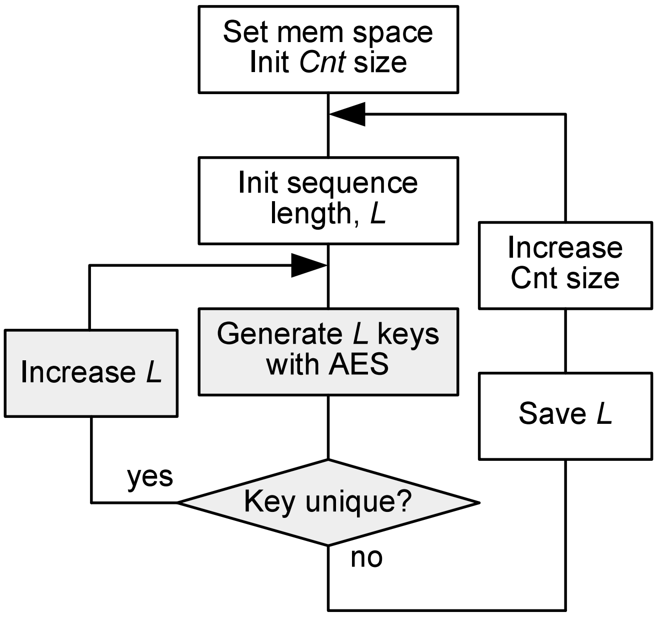
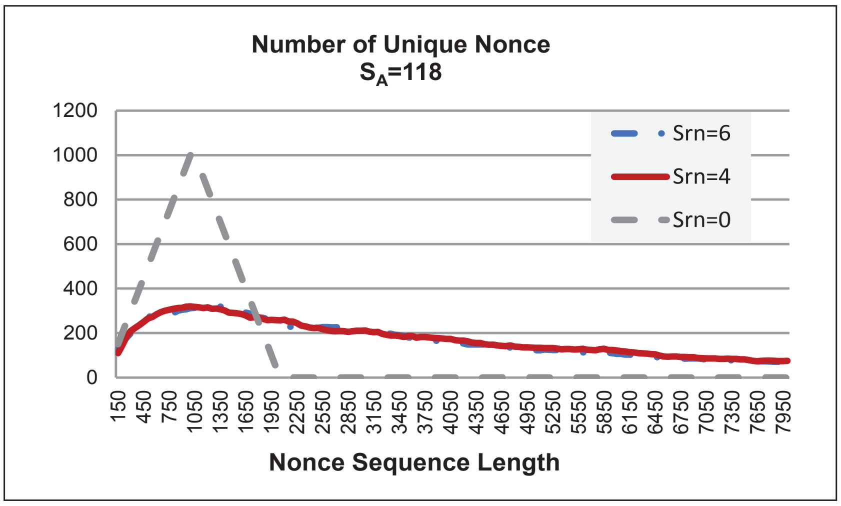


FIGURE 5.4: Software Experiment Platform

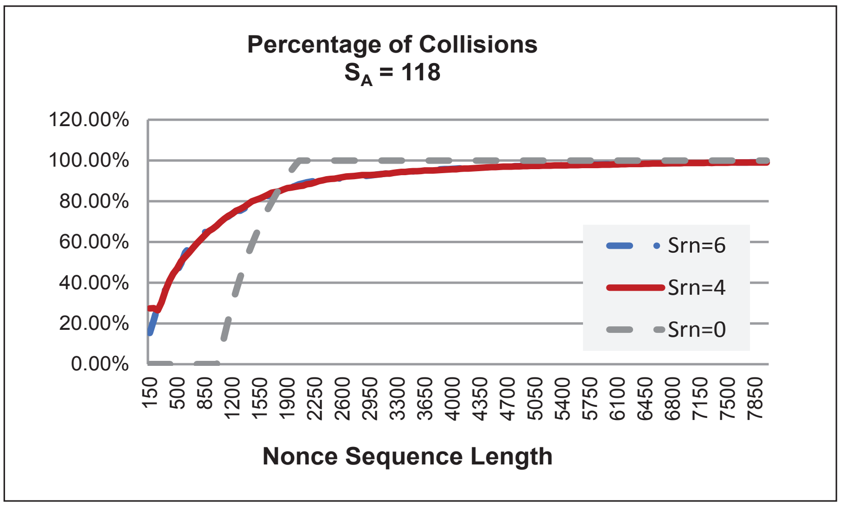
We first examined how the random number size (hence the counter size) affects the nonce collisions and whether the AES block cipher generates a unique nonce for a different input.

Given the limited host machine resources, we generated nonce sequences each with a length below 800000. Therefore, we set the memory address size in the range between 110 to 118 bits so that obtained sample data are sufficient for the nonce collision investigation.

With a fixed memory address, for each given random number size (S_{RN}), we generate a set of nonce sequences of varying lengths. Figure 5.5(a) shows the number of unique nonces in each sequence when the address size is set to 118 bits (namely, $S_A = 118$). For the case of $S_{RN} = 0$, there are no random numbers



(a)



(b)

FIGURE 5.5: Nonce Collisions (S_A=118)

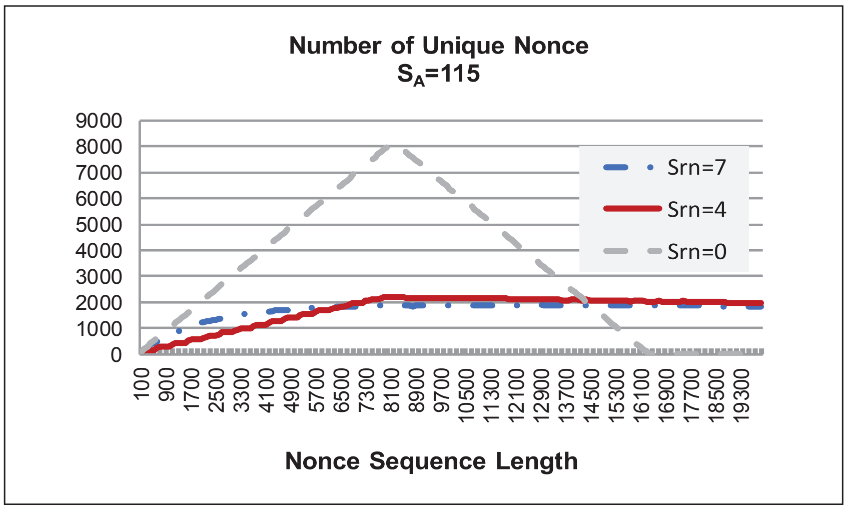
in the input and the counter size is 10 bits, the first 2^{10} nonces are always unique, hence the number of unique nonces linearly increases with the nonce sequence length; but further increases of the sequence length lead to the collision for each new nonce, therefore the number of unique nonce linearly decreases as the nonce sequence grows; after $2 * 2^{10}$ nonces, all nonces are re-used and the number of unique nonce is reduced to 0.

However with the existence of input random number ($S_{RN} > 0$), the collision can be observed within a short nonce sequence; the collision frequency increases with the nonce sequence length and approaches to saturation, as shown in Figure 5.5(b), where the collision percentage is calculated by $(L - U)/L$ (L is the nonce sequence length and U the number of unique nonces in the sequence). The similar feature can also be observed for smaller memories, as presented in Figure 5.6 and Figure 5.7 for $S_A = 115$ and 113. Also from the figure (b) of Figure 5.5,5.6, and 5.7, we can see that the large random number size helps reducing nonce collisions. The percentage nonce collision is over 10% when about 5% nonce in the nonce space is used.

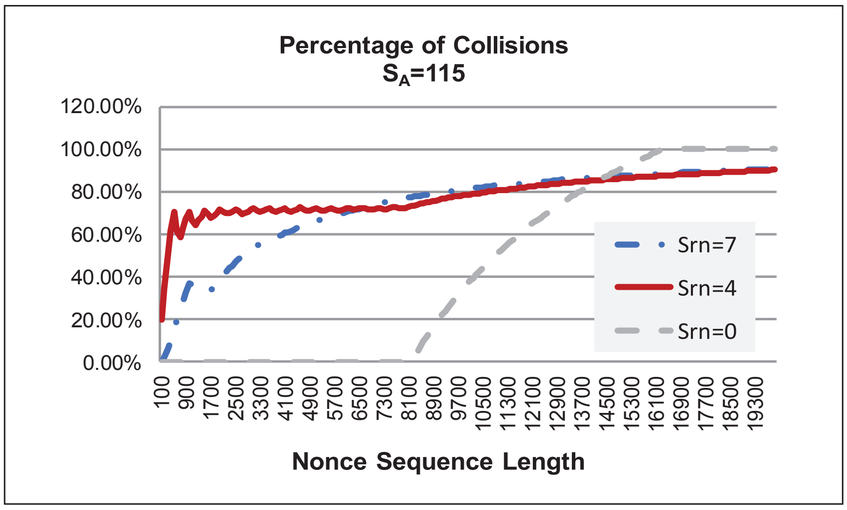
With the given security evaluation Formulas (5.1) and (5.5), we evaluate the security of design proposed in [67] and the design in CETD. Both designs use AES as the block cipher, and the block cipher size is set to 128 bits.

Attack success probabilities of the two designs against the random and used-nonce attacks are summarized in Table 5.1. For a general view, Row 2 gives the nonce generation function of each design; with the CETD design, address A, random number RN, and counter C are used as the AES input, while in the Yan's design only the address and counter value C' are considered. The size of C' in Yan's design is the sum of sizes of RN and C in CETD. The success probability of the random attack is given in Row 3 and the probabilities of the used-nonce attack under two cases ($L <= 2^{(128-S_A)}$, and $L > 2^{(128-S_A)}$) are given in the last two rows, where the value is approximated with the input collision probability as 1.

As can be seen from Table 5.1, both designs have the equal security against the random attack. For the used-nonce attack, CETD has a same or higher attack success probability than the Yan's design. But it is still limited by $1/2^{(128-S_A)}$.



(a)

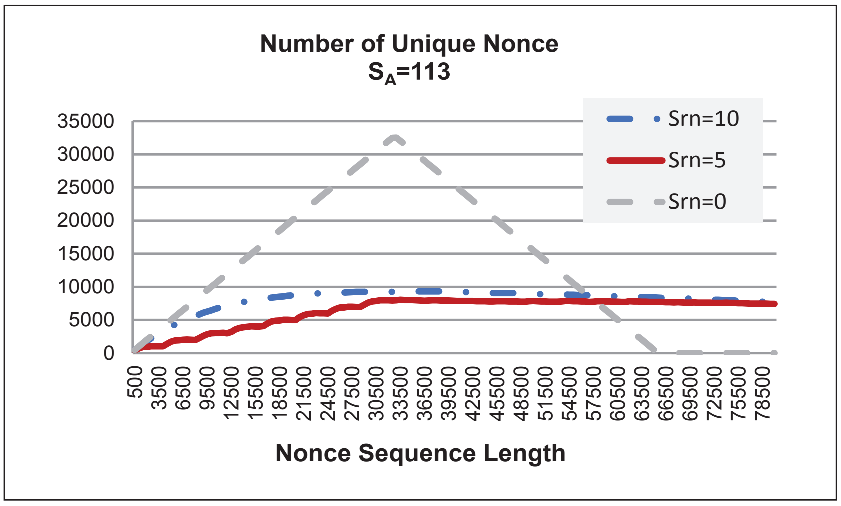


(b)

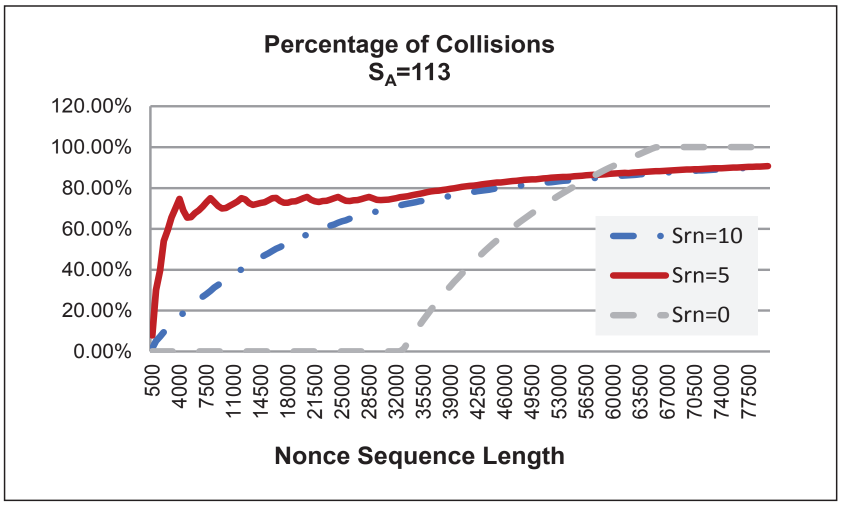
FIGURE 5.6: Nonce Collisions (S_A=115)

TABLE 5.1: Attack Success Probability of Two Existing Designs

		CETD [30]	Yan's Design [67]
nonce generation formula		AES(A,RN,C)	AES(A,C')
random attack		$1/2^{128}$	$1/2^{128}$
used-nonce attack	$L <= 2^{(128-S_A)}$	$\sim 1/2^{128-S_A}$	0
	$L > 2^{(128-S_A)}$	$1/2^{128-S_A}$	$1/2^{128-S_A}$



(a)



(b)

FIGURE 5.7: Nonce Collisions (S_A=113)

With a common 32-bit memory address space system and the memory block size of 2^7 bytes, the probability is $1/2^{128-32+7} = 1/2^{103}$, which is equivalent to resisting brutal-force attack on 103-bit nonce; but this brutal-force attack is under the restriction of the much short nonce life as compared with the normal static nonce.

5.3 Summary

We evaluated the security of the nonce design in CETD and compared its security with another existing nonce design. In these designs, a nonce is generated with a block cipher, and the nonce changes with different memory block and access time. Two types of attacks have been identified: the random attack with arbitrarily chosen nonce values, and the use-nonce attack with uncovered old nonce values. The success probabilities of the both attacks were analyzed.

It was found that the random attack basically has a lower success probability than the used-nonce attack. For the used-nonce attack, the security of the CETD design is degraded due to the possible input random collisions as compared with the design without the random number in the block cipher input. The success probability of the used-nonce attack is exponentially limited by the difference of (nonce size and memory address size). For a typical 128-bits nonce, 32-bit memory address, and 2^7-byte data block, this success probability is $1/2^{103}$, which is equivalent to resisting brutal-force attack on 103-bit nonce; but this brutal-force attack is under the restriction of the much short nonce life as compared with the normal static nonce. Therefore, we regard that the nonce design in the CETD is sufficient secure.

In the next chapter, we will focus on the security improvement of the tag design.

Chapter 6

Security Improvement on CETD

We have known the design weaknesses for the tag generation in the CETD. In this chapter, we investigate the possible solutions to improve the security of the CETD design, which are discussed in Sections 6.1. The analysis of final improved CETD, the iCETD, is given in Section 6.2. The experimental verification of our improved design is presented in Section 6.3. The chapter is concluded in Section 6.4.

6.1 iCETD: An Improved Design

We want to improve the security of CETD at low overhead while keeping the trade off feature (cost vs tag size) of the original design. A straightforward idea is applying randomization functions to its input or output. Figure 6.1 shows two examples of the designs we initially considered with the low cost XOR operation.

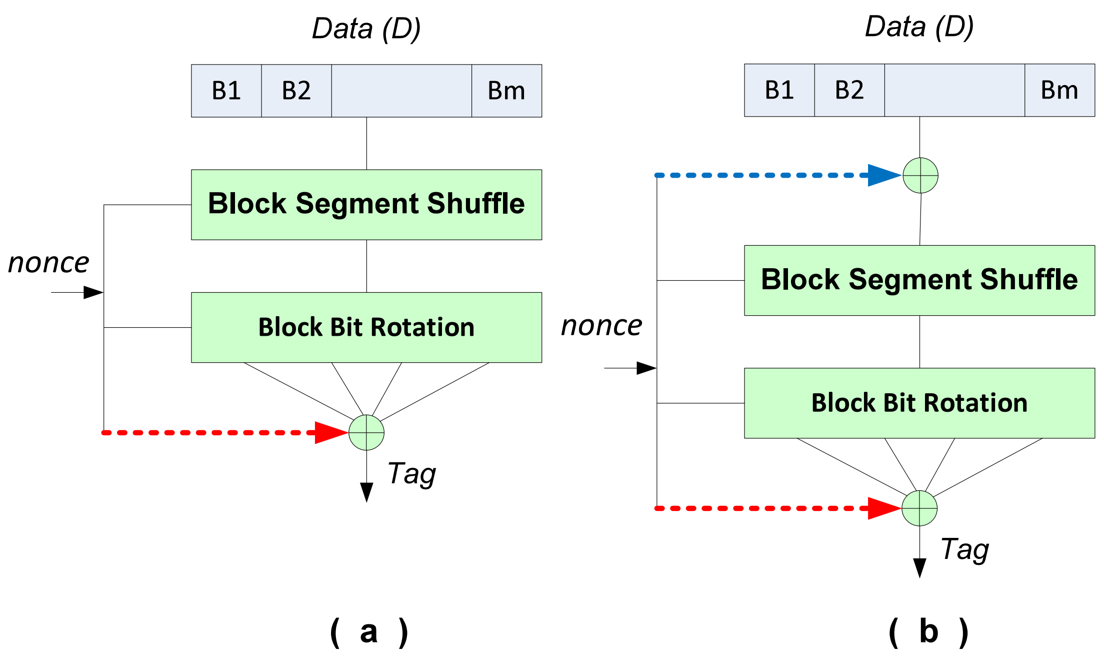


FIGURE 6.1: Two Design Options with XOR (a) injecting random bits to the CETD output (b) injecting random bits to the CETD input

In the design of Figure 6.1(a), the random value from the nonce is injected to the output of the CETD. The tag generated from the design can be written as

$$tag = r \oplus (\bigoplus(SR(D))), \tag{6.1}$$

where $SR(.)$ represents the combined shuffle and rotation operations in CETD and $\bigoplus$ the operation that XORs all blocks and the random value r is a segment of the nonce used in the shuffle and rotation operations.

If the design is targeted by a spoofing attack, where the nonce will be unchanged, the r in Formula (6.1) for the forged data is the same as the r used for the true data.

Assume the current valid data $D1$ with tag $T1$ was updated at time $t1$ and the spoofing attack was lunched with a forged data $D2$ at time $t2$ ($t2 > t1$) during a memory read operation. In order for the forged data to pass the authentication, the tag value, $T2$, calculated for $D2$ should be the same as $T1$. The probability of the two tags to be identical can be written as:

$$\begin{aligned} Pr[T1 = T2] &= Pr[(r \oplus (\bigoplus(SR(D1)))) \oplus (r \oplus (\bigoplus(SR(D2)))) = 0] \\ &= Pr[(\bigoplus(SR(D1))) \oplus (\bigoplus(SR(D2))) = 0]. \end{aligned} \tag{6.2}$$

Formula (6.2) indicates that the tag collision probability of the design shown in Figure 6.1(a) is the same as that of the CETD design for the spoofing (with reused nonce) attacks. If the attacker modifies the memory data by taking advantages of the CETD weaknesses, the attack will have at least a double success probability as compared to the random attack.

For the design shown in Figure 6.1 (b), where both the input data and tag are randomized with the nonce value, the tag is calculated by

$$tag = r \oplus (\bigoplus(SR(D \oplus nonce))). \tag{6.3}$$

In the similar way, we can find that randomization of the input D with $(SR(D \oplus nounce))$ will still result in the same bit frequency parity for two different data values if they have an identical bit frequency parity. For example, if D has an odd number of 1s, any data with odd number of 1s when XORed with the same nonce value, the resulting in parity is same; therefore, the parity of the related tag values are the same, which effectively reduces the data space, as well as the tag space, by half.

As we can see, the above designs with XORing random value from nonce still carry some correlations between their input and output, which can be exploited by the attacker. To break the input/output correlation, in our final design, we include a non-linear random bit injection operation. For non-linear random bit injections, the multiplication on the Galois Field [41] is a common technique.

Galois Field (Galois Field (GF)) , also called Finite Field, is a set that contains finite number of elements. The results of arithmetics on a GF should also be in the same GF. In [50] Nyberg proved the non-linear property of Galois Field Multiplication (Galois Field Multiplication (GFM)). Daemen and Rijmen later in [17] indicated that GFM and its inverse operation can be used in cryptographic systems. GFM has been used in block cipher construction ([18]) and data protection systems (GCM [45]).

With GFM, the multiplicand and multiplier have the same length. They are expressed as polynomials. To ensure that the result of GFM is still in the same Galois field, the product polynomial of GFM is divided by an irreducible polynomial. A polynomial is irreducible if it has no divisors other than 1 and itself. We show the procedure of GFM by using 0x57 and 0x83 as multiplicand and multiplier. The computation is given below:

$$\begin{aligned} 0x57 \cdot 0x83 &= 01010111 \cdot 10000011 \\ &= (x^6 + x^4 + x^2 + x + 1) \cdot (x^7 + x + 1) \quad \text{mod } (x^8 + x^4 + x^3 + x + 1) \\ &= x^7 + x^6 + 1 \\ &= 0xC1, \end{aligned} \tag{6.4}$$

where $(\mathrm{x}^8 + \mathrm{x}^4 + \mathrm{x}^3 + \mathrm{x} + 1)$ is the irreducible polynomial.

One issue with GFM is that it cannot effectively inject randomness to the data of all 0's. To deal with this problem, we add the random block bit flip (Random Block Bit Flip (RBF)) operation to disturb the input data. It must pointed out that the block flip is a linear operation, using it without GFM will not break the correlation the design input and output.

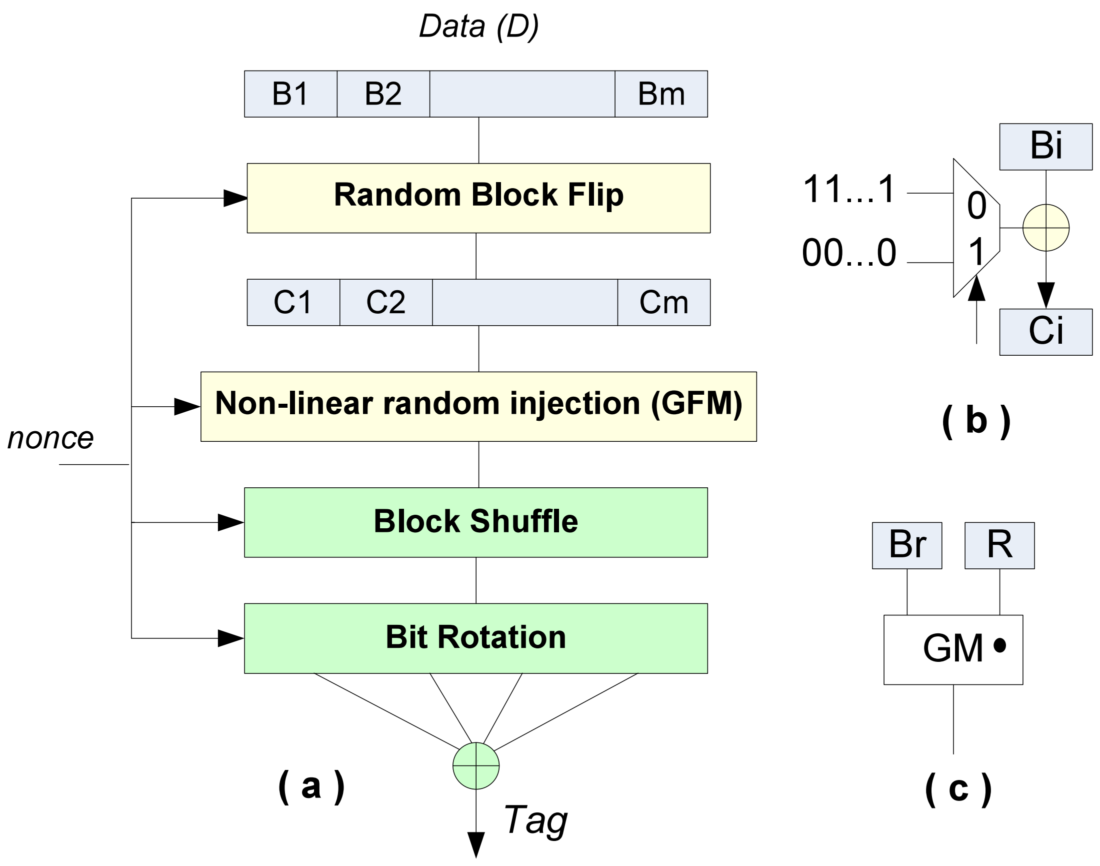


FIGURE 6.2: Final Design (iCETD) (a) overview (b) random block flip (c) non-linear random injection

The whole structure of the improved design (here we name it as **iCETD**) is shown in Figure 6.2(a), where the random block flip operation and block GFM are built on top of the original CETD design. A block can be randomly flipped based on the control of the nonce value bits; the design for block flip is shown in Figure 6.2(b), where a 2-to-1 multiplexor is used to select all 0's or all 1's for XORing with the related block; if the control bit is 1, the block is XORed with all 1's and it is flipped; otherwise, the block is copied through. The block flip operation camouflages the input value of straight 1 bits or 0 bits. The function can be implemented by bit-wise XORing each block with its corresponding control bit.

The input data is further confused by the non-linear random bit injection, which is realized by GFM a random value with a randomly picked block. The confusion injected by the block flip and GFM operation is then diffused over all blocks through the block shuffle and bit rotate operations existing in the original design.

It must be pointed out that the CETD design is a parallel MAC scheme. For a parallel MAC, block ordering should be maintained. Rather than ordering blocks by a fixed index as found in other parallel MAC designs [7, 13, 27, 54], we use the randomized block flip, GFM, segment shuffle and bit rotation operation to enforce the block order. In case some blocks swapped by a forgery, the change of the block order will be reflected through these random block operations.

In summary, the iCETD design proposed here has the following features:

- injection of confusion with non-linear random operations is included;
- the number of input blocks participating GFM operation is adjustable;
- the confusion is spread by the random block shuffle and random bit rotate operation;
- the data block ordering is better maintained; as a result,
- the tag value generated from the design is sensitive to a change to the input data and sensitive to a change to nonce, and

- the tag value is random over the full tag value domain.

Playing data from different memory locations or different times will face a different nonce, resulting in different tag generation process, hence different tag value; the same is true for the attack with a forged tag. In the case of the used tag attack, the nonce is unchanged, hence the same tag generation process will applied to both the original and the forged data; however, since the design is sensitive to the change of the input data, the tag values generated from the two different data inputs will be random and has a collision probability same as random numbers.

The security analysis of the iCETD design is elaborated in the next section.

6.2 Analysis

In 4.1 we categorized the attacks from the data authentication perspective into three groups:

- CAR Attack, where the attacker replays a copy of a valid data/tag pair from a different memory location or the same memory location but of a previous time spot;
- FT Attack, where the data used by the attack may not be an existing or old data and the related tag is forged;
- RT Attack, where the data used by the attack may not be an existing or old data and the tag of the current valid data is reused.

We evaluated the security of CETD against a chosen data attack based on the tag collision rate of the data. We found that the CETD design leaves some pitfalls for chosen-value attacks:

- The effective tag value space for a given data input is smaller than the full tag value domain and the effective tag space varies with different input data. The maximal effective tag space size is just half of the full tag value space;
- For CETD, a data and its tag have a same bit frequency parity.

To break the input/output correlation as shown in CETD, in iCETD we process the input data with Random Block Flip(RBF) and selected Galois Field Multiplication(GFM) operation before the Block Shuffle and Bit Rotation stages.

Assume there are two distinct values A and C doing GFM with non-zero value B. Further more A, B and C come from the same Galois field. If the products are identical, the following equation should be met:

$$B \cdot A = B \cdot C, \tag{6.5}$$

where $\cdot$ represents GFM operation. According to the basic properties of Galois field multiplication, if B is not zero then there is an another value B^{-1} that makes the following equation valid:

$$B^{-1} \cdot B = 1.$$

This means Equation 6.5 can be converted to the following format:

$$B^{-1} \cdot B \cdot A = B^{-1} \cdot B \cdot C \tag{6.6}$$

$$1 \cdot A = 1 \cdot C. \tag{6.7}$$

It is a basic property in Galois field that any value A in the field doing GFM with will get A. This fact means

$$A = C. \tag{6.8}$$

From Equation 6.8 we can see that given a non-zero value B, for any distinct values A and C, the related two products are distinct. This fact means given a non-zero block value B, the nonce piece r and the output value of GFM G is bijective. Further more, it is also a basic property in the Galois field that if two operators of multiplication come from same field, their product will also be in the same field.

For summary we can see when doing single block GFM on a data D, if the block selected by GFM is not all-zero and the nonce is random, then the output block of GFM is also random and uniformly distributed in the block domain.

Because any data block doing GFM with all-zero block will lead to all-zero output, any data containing all-zero blocks has possibility to be unchanged after GFM. If all bits in the data is 0, then the data will not be changed by GFM. To address this issue, we add Random Block Flip(RBF) before GFM to randomly flip the bits in some blocks in the data.

Assume there are K set-bits in data D and there are i blocks flipped containing totally $k1$ set-bits. The output of RBF stage is marked as D_{RBF}. The set-bits in D_{RBF}, marked as $K2$, is $K - k1 + i * n - k1$. The difference between $K1$ and $K2$ is $i * n - 2 * k1$, which is a even number. This means RBF can inject set-bits to all-zero blocks, but cannot change the parity of the number of set-bits. Further more, if the data D containing all-zero or all-one blocks then the output data of GFM, marked as D_{GFM}, still has possibility to have same bit frequency parity(BFP) as D. This is because during RBF stage, all-zero can be unchanged and all-one block can be flipped to all-zero block. If GFM processes such an all-zero block in D_{RBF}, then the GFM has no effect on D_{RBF} and D_{GFM} will be same as D_{RBF}.

Assume there are i all-zero blocks and j all-one blocks in the data D sent to RBF. The sum of i and j is marked as X. The probability that there are k all-zero blocks is expressed as:

$$\binom{X}{k} * 1 * 1/2^m. \tag{6.9}$$

First we discuss the single block GFM case. The probability that GFM selected an all-zero block in D_{RBF} is:

$$k/m. \tag{6.10}$$

The probability that D and D_{GFM} have same BFP is:

$$\sum_{k=1}^{X}(\binom{X}{k} * k/m * 2^m). \tag{6.11}$$

Simplifying Equation 6.11 the probability can be expressed as:

$$X/m * 2^{(m-X+1)}. \tag{6.12}$$

Because $X\in[0,m]$, Equation 6.12 gets maximum value when X is m. The maximum probability is 1/2.

If there are i blocks in D_{RBF} doing GFM, the probability that these i blocks are both all-zero, denoted as Pr[i], is:

$$\binom{k}{i}/\binom{m}{i}. \tag{6.13}$$

Comparing Pr[i] and Pr[i+1] we can get the following equation:

$$Pr[i]/Pr[i+1] = m - i/k - i. \tag{6.14}$$

Because m≥k, then Pr[i] ≥ Pr[i+1]. This means if we increase the number of blocks doing GFM, then the probability that all blocks selected by GFM is all-zero blocks will be reduced.

On the other hand, the probability that D and D_{GFM} have same BFP if i blocks do GFM is:

$$\sum_{k=i}^{X}(\binom{X}{k}*\binom{k}{i}/(2^m*\binom{m}{i})). \tag{6.15}$$

We can see that if i is increased, then the number of addends in Equation 6.15 is reduced. On the other hand when i is bigger than X, GFM will process at least one block in D_{RBF}, which means Equation 6.15 will be zero.

Assume the input data D of iCETD contains some all-zero or all-one blocks. We denote the GFM on i blocks as GFM$_i$. Based on the above analysis we can see that if we increase i then the result of Equation 6.15 will be decreased. This means the probability that GFM gets invalidated is decreased. If doing GFM on all blocks then no data except the one formed with only all-zero or all-one blocks can invalidate GFM stage. The value X of data that has probability to invalidate GFM$_m$ is m.

We will show through the experiments that the tag values generated from iCETD for a given input are random and uniformly distributed over the whole tag value space, which is given in the next section.

6.3 Simulations and Discussions

A simulation system written in C has been built to test the randomness of tag values. Figure 6.3 shows the experimental setup.

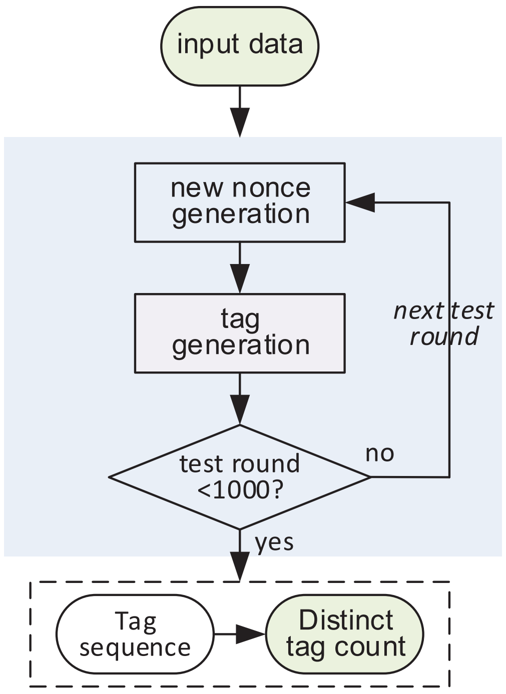


FIGURE 6.3: Experimental Setup

For a given input, we simulate copy-and-replay attack by generating tags with randomly generated nonce values (1000 random nonce values used for each input), as given in Figure 6.3. All the experiments are based on the 32-bit data and 8-bit tag settings.

To check the effectiveness of the Galois Field Multiplication (GFM) operations, we applied GFM to variable k data blocks in iCETD and the related design is denoted by iCETDk. In our experiment, k is in the range of 0 to 4; k=0 means no GFM operation in the design, while k=4 means all data blocks participate the GFM operations.

Figure 6.4 depicts the number of distinct tags for individual input data in the small range from 0 to 300 (for visibility) in the decimal format for three different designs (a) CETD, (b)iCETD without GFM (iCETD0), and (c) iCETD with GFM applied to a random one block (iCETD1). From Figure 6.4, we can see that the effective tag space size from the CETD design varies greatly for different input data, the minimum size is 1 (that is associated with the data of all 0's) and the maximal effective tag space size is about 125 – half of the full tag value domain. With the use of the block flip operation (the design of iCETD0), the value distribution becomes more uniform but still on the half of the full tag value domain. By adding the GFM operation (ref. iCETD1), the tag distribution is not only uniform but also towards over the full tag value space.

The maximum, minimum, average and the standard variance of the effective tag space size of all input data for the two designs are given in the last four rows in Table 6.1. For iCETD designs, all different GFM operation cases (namely, $k = [0, \cdots, 4]$) are examined.

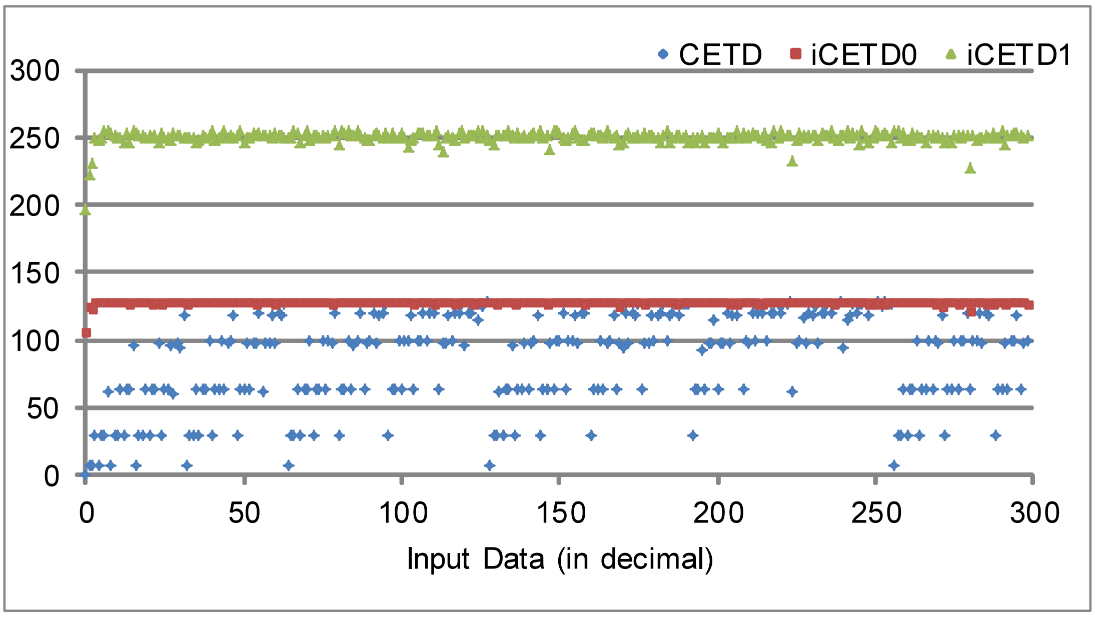


FIGURE 6.4: Number of Distinct Tag for Individual Input Data (a) CETD (b) iCETD0 (c) iCETD1

Table 6.1 shows that iCETD designs with GFM operations (last four columns in the table) have about the same effective tag space size (with a small standard variance, ranging from 4.6 to 7.6), for all individual data, and the tag values are distributed almost all over the full tag domain (256). But for the CETD design (Column 2), the maximum effective tag space size (namely, 128) is the half of the full tag domain and the effective tag space sizes are distributed over the large range from 0 to 128 with a significant deviation (the standard variance equals to 148.91).

From the Column 3 of Table 6.1, we can also see that the block flip operation in iCETD0 only partially improves the design in that all the input data have about the same effective tag space size (close to 128) and the tag values are uniformly distributed (with a small standard variance, 0.133), but the effective tag space size still remains below the half of the full tag domain.

Therefore, the experiment results demonstrate that the tags from both the CETD and the iCETD0 designs are not uniformly random over the full tag value domain for individual data.

TABLE 6.1: Statistics of Effective Tag Space Size

Design	CETD	iCETD				
		(0)	(1)	(2)	(3)	(4)
max	128	128	256	256	256	256
min	1	106	197	232	230	239
var	148.91	0.13317	7.6289	4.7062	4.6309	4.593
avg	123.2	127.9	250.36	250.87	250.87	250.89

To verify whether the iCETD design is random and sensitive to the input data and nonce, we run NIST tests on the tag sequences collected from the experiments. For each input data, D, we concatenated its 1000 8-bits tags to form a 8000-bit sequence, as shown in an example in Figure 6.5 where three tags (01101101, 11001011, 00110001) form a 24-bit sequence 011011011100101100110001. For the data size of 16 bits, there are 2^{16} different data values, therefore, a total of 2^{16} sequences, each of 8000 bits, were formed.

Table 6.2: Chosen NIST Tests

Test Name(Test No.)	Parameter	Description	Min Len(n)
Approximate Entropy(1)	m=3	compare freq of m and m+1 bits pattern	100
Block Frequency(1)	M=128	Proportion of 1 in M-bits blk	100
Cum-sum(2)	NaN	unbalanced value of cumsum of the sub-seq	100
Frequency(1)	NaN	Proportion of 0 and 1 in seq	100
Long Run of Ones(1)	NaN	Longest runs of 1 in the seq	128
Runs(1)	NaN	total number of uninterrupted identical sub-seq in a seq	100
Serial(2)	m=8	frequence of all m-bit pattern in the seq	100
Spectral DFT(1)	NaN	peak heights in the DFT of the seq	1000

Table 6.3: NIST Tests Results

NIST Test Name	CETD	iCETD				
		(0)	(1)	(2)	(3)	(4)
Approximate Entropy	0.52594	0.896988	0.974777	0.985641	0.988739	0.989304
Block Frequency	0.678101	0.973984	0.98671	0.988815	0.989609	0.989899
Cumulative Sums: Reverse	0.5009	0.949203	0.978027	0.985748	0.988861	0.989182
Cumulative Sums:Forward	0.50058	0.949997	0.978409	0.985184	0.988846	0.989075
Discrete Fourier Transform	0.945313	0.988342	0.988754	0.988953	0.989258	0.989319
Frequency	0.494202	0.946671	0.97728	0.985397	0.988693	0.988861
Longest Run	0.694946	0.956848	0.988419	0.989197	0.989365	0.989532
Runs	0.547684	0.886246	0.974625	0.985703	0.988693	0.989624
Serial 1	6.10E-05	0.000153	0.972397	0.987961	0.988159	0.989502
Serial 2	0	0	0.978058	0.989319	0.989746	0.990143
AVG	0.444339	0.686221	0.980445	0.98723	0.988839	0.989317

TABLE 6.4: NIST Tests with Large Data Set Requirements

Test Name(Test No.)	Parameter	Min Len(n)
Universal Statistical(1)	NaN	387840
Linear Complexity(1)	m	1000000
Non-Overlapping(148)	m	1000000
Overlapping(1)	m	1000000
Random Excursions(8)	NaN	1000000
Random Excursions Variant(18)	NaN	1000000
Rank(1)	M=Q=32(default)	38912

The NIST suite offers fifteen tests. Some tests, as listed in Table 6.4, require large input data sets, with the minimum data size of at least 38912 bits (see the last column in the table) that is much larger than the 8000-bit tag sequences. Due to the limited resources (especially storage capacities) we currently have, those tests are not fulfilled and only eight types of tests, as given in Table 6.2 are performed in our experiment. The description of each test is given in the third column and the minimum data size required is presented in the last column. Each test has several sub-tests. The Test No. following the test name in the first column is the number of sub-tests given by NIST. Some of these tests have parameters; we apply the tag size (8 bits) as the pattern length for the Serial test to examine whether the tag value is randomly distributed over the whole tag value space; other parameters were set based on the data sequence length as required by the NIST. We run NIST tests on each bit sequence and the statistic pass rate of each NIST test is obtained based on all 2^{16} tag sequences.

Table 6.3 shows the pass rate of the chosen eight NIST tests for the tag sequences generated by the CETD and the iCETD designs with different number of GFM block operations (shown in the second row for iCETD). For each design, the average pass rate is given in the last row.

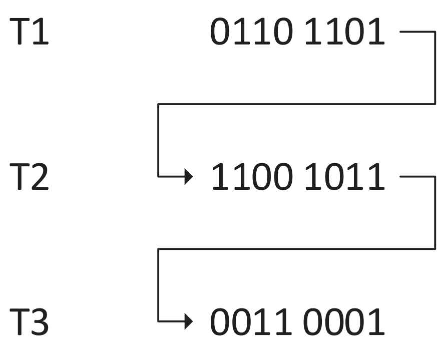


FIGURE 6.5: Tag NIST Test Sequence Formation

From Table 6.3, we can see that the pass rate of the CETD tags is low, especially for the Serial test which are closely relevant to the tag value distribution over a 8-bit value space; hence the CETD tag is not random for individual input data. On the other hand, the iCETD designs with the GFM operation (last four columns in the table) have straight high pass rates across all tests, which demonstrates that the GFM operation greatly improves the random distribution; without the GFM operations, the randomness of the tag is considerable degraded (see column 3).

To see how GFM affects the tag distribution over the tag domain, we further ran two Serial tests (Serial 1 and Serial 2) for a number of typical input data. The results are shown in Table 6.5, where the first column lists the individual data in hexadecimal format and the pass rate of their tag sequence for four designs (with 0-block-GFM operation to 4-block-GFM operation) are given in the last 5 columns.

As can be seen from the table, the pass rate increases as the number of blocks applied by GFM operation increases.

TABLE 6.5: NIST Serial Tests of Tags for Individual Data

Data	NIST Random Distribution Test	iCETD (0)	(1)	(2)	(3)	(4)
0x00,0x00,0x00,0x00	Serial 1	0	0	0	0.269	0.934
	Serial 2	0	0	0.331	0.926	0.994
0x00, 0x00, 0x00, 0xff	Serial 1	0	0	0	0.264	0.917
	Serial 2	0	0	0.317	0.925	0.987
0xFF,0x00,0xFF,0x00	Serial 1	0	0	0	0.271	0.921
	Serial 2	0	0	0.309	0.921	0.99
0xFF,0xFF,0xFF,0xFF	Serial 1	0	0	0	0.286	0.931
	Serial 2	0	0.149	0.587	0.987	0.995
0x00, 0x00, 0x00, 0x5A	Serial 1	0	0	0.234	0.913	0.918
	Serial 2	0	0.122	0.925	0.987	0.985
0x00,0xFF,0x00, 0x5C	Serial 1	0	0	0.287	0.884	0.931
	Serial 2	0	0.205	0.928	0.986	0.99
0x00, 0xff, 0x34, 0xCF	Serial 1	0	0.962	0.987	0.986	0.993
	Serial 2	0	0.952	0.986	0.989	0.993
0x00, 0x12, 0x34, 0x56	Serial 1	0	0.982	0.989	0.989	0.99
	Serial 2	0	0.986	0.987	0.988	0.995
0x12,0x34,0xAB,0xCD	Serial 1	0	0.983	0.995	0.989	0.993
	Serial 2	0	0.986	0.991	0.989	0.991
0x56, 0x78, 0x9A, 0xBC	Serial 1	0	0.988	0.997	0.99	0.99
	Serial 2	0	0.987	0.996	0.992	0.995

The experiment results verify the randomness of the iCETD tag, hence shows the security improvement on the original CETD design.

6.4 Conclusions

In this chapter, we investigated a few possible solutions to improve the security of CETD design. It is found that further randomizing CETD with XOR operation is not effective. We presented an improved design that includes the random block flip operation and the non-linear Galois Field multiplication. Our analysis and experiments show that the tag of any individual data from the design is random and distributed over the full tag domain. Therefore, the improved design has same security against the chosen-value attack as against the random attack.

Chapter 7

Conclusions and Future Work

Security has becomes an critical issue in embedded systems. To effectively protect data, that are stored and processed in the embedded system, is important. The thesis has investigated an existing data authentication design (CETD) for protecting the integrity of memory data in the embedded processor system. As a summary, this chapter highlights the contributions of the thesis work and discusses the possible research directions on the memory data protection and security design evaluation for embedded systems.

7.1 Conclusions

In this thesis, we target the data protection design for the system that has a secure processor chip and insecure off-chip memory and the system is embedded in a bigger system to perform a dedicated or a set of dedicated functions.

The CETD design is designed for the embedded system. It utilizes the customization techniques to reduce the resource overhead, which has been extensively studied in the original work. In this thesis, we have focused on the security of the design.

Existing security designs are mainly based on the standard cryptographic primitives and the evaluation approaches used for those designs are not suitable to the security evaluation of CETD which is mainly constructed on a group of special operations (They can carry the randomness of the data). Therefore, we have applied a different evaluation strategy.

We model the security of the memory data protection design as the security of the secret values in the design: nonce and tags, and we evaluate the security of the design related to each secret value based on the its collision probability. The higher collision probability means the lower security of the design.

In CETD, the nonce is generated with a block cipher that takes the input the random number, address, and counter value associated with a memory location. Use of the random number is cost-effective; but with our study, the random number in the input of the block cipher will slightly degrade the design security against the used-nonce attack. However, the degradation can be acceptable due to the dynamic nature of the nonce used in the CETD design.

For the tag generation design, we have identified three loopholes:

- There are some correlations between data and its tag. The tag generated by CETD has the same parity of 1s as its data (namely, either odd or even number of 1s).

- For a given data, its tag value is not distributed over the whole tag value space; the effective tag space size for a given data is reduced and is less than the half the full tag value space.
- The effective tag space size varies for different data.

Those loopholes lead to the low security of the design against the chosen-value attack on the data integrity.

To improve the security of the design, we proposed a design modification that includes the random block flip function to the input data and random Galois field multiplication on the data block. The Galois field multiplication is a finite-field operation that is non-linear. Those operations added to the original CETD design break the correlation of the data and tag and ensure that the tag generated is random and uniformly distributed on the full tag value space.

7.2 Future Work

The research work focuses on the memory data protection and design evaluation for embedded systems. The work is yet fully completed and some extensions are possible, which are given below.

In the nonce design evaluation, for the used-nonce attack, our work does not consider the time the attack is applied, which may have different security implications on the designs with and without the random number. With the use of the random number in the nonce generation, the security can be increased as compared to the design without the random number, which can be further studied.

Due to the time limit, we only focused on the security of CETD and its improvement. The random block flip (RBF) and Galois Field Multiplication (GFM) stages introduced in iCETD should incur some overheads. Both RBF and GFM can be implemented in hardware and/or software. Their implementation costs and performance overheads need to be studied.

In terms of security design evaluation, though many approaches have been proposed, a systematically approach is still lacking. A contribution to this area is ultimately important. The same is for design. A systematical design approach for memory data protection is also a much worth topic for future work.

Bibliography

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

[2] Network security protocols. http://www.k2esec.com/secure-communications/network-security-protocols-ipsec-vs-tlsssl-vs-ssh-part-ii, Retrieved on Jan 10, 2014.

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

[4] J. A. Ambrose, R Ragel, and S. Parameswaran. Rijid: Random code injection to mask power analysis based side channel attacks. In *Design Automation Conference (DAC '07)*, page 6pp, San Diego, Ca, USA, 2007.

[5] M. Bellare. Introduction to modern cryptography: Message authentication. *https://cseweb.ucsd.edu/ mihir/cse207/w-mac.pdf.*

[6] M. Bellare and S. Goldwasser. New paradigms for digital signatures and message authentication based on non-interactive zero knowledge proofs. In *Advances in CryptologyCRYPTO89 Proceedings*, pages 194–211. Springer, 1990.

[7] 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.

[8] 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.

[9] 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.

[10] M. Bellare and P. Rogaway. Code-Based Game-Playing Proofs and the Security of Triple Encryption. Cryptology ePrint Archive, Report 2004/331, 2004.

[11] A. Berendschot, J-P. Boly, A. Bosselaers, J. Brandt, D. Chaum, I. Damgård, P. de Rooij, M. Dichtl, W. Fumy, C. 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.

[12] 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.

[13] 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.

[14] B. Blanchet and D. Pointcheval. Automated security proofs with sequences of games. In *Advances in Cryptology-CRYPTO 2006*, pages 537–554. Springer, 2006.

[15] H. Comon and V. Shmatikov. Is it possible to decide whether a cryptographic protocol is secure or not?, 2001.

[16] J. Crenne, R. Vaslin, G. Gogniat, J-P. Diguet, R. Tessier, and D. Unnikrishnan. Configurable Memory Security in Embedded Systems. *ACM Trans. Embed. Comput. Syst.*, 12(3):71:1—-71:23, April 2013.

[17] J. Daemen and V. Rijmen. AES proposal: Rijndael. 1999.

[18] J. Daemen and V. Rijmen. The block cipher rijndael. In *Smart Card Research and Applications*, pages 277–284. Springer, 2000.

[19] D. Dolev and A. C. Yao. On the security of public key protocols. In *22nd Annual Symposium on Foundations of Computer Science*, pages 350 – 7, 1981.

[20] R. Elbaz, D. Champagne, C. Gebotys, R. B. Lee, N. Potlapally, and L. Torres. Hardware mechanisms for memory authentication: A survey of existing techniques and engines. In *Transactions on Computational Science IV: Special Issue on Security in Computing*, volume 5430 LNCS, pages 1–22, 2009.

[21] R. Elbaz, D. Champagne, R.B. Lee, and L. Torres. Tec-tree: a low-cost, parallelizable tree for efficient defense against memory replay attacks. In *9th International Workshop, Cryptographic Hardware and Embedded Systems*, 2007.

[22] R. Elbaz, L. Torres, G. Sassatelli, P. Guillemin, M. Bardouillet, and A. Martinez. A parallelized way to provide data encryption and integrity checking on a processor-memory bus. In *Proceedings - Design Automation Conference*, pages 506–509, 2006.

[23] R. Elbaz, L. Torres, G. Sassatelli, P. Guillemin, M. Bardouillet, and A. 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.

[24] PUB FIPS. 81: Des modes of operation. *Issued December*, 2:63, 1980.

[25] B. Gassend, G. E. Suh, D. Clarke, M. Van Dijk, and S. Devadas. Caches and hash trees for efficient memory integrity verification. In *High-Performance Computer Architecture, 2003. HPCA-9 2003. Proceedings. The Ninth International Symposium on*, pages 295–306. IEEE, 2003.

[26] O. Gelbart, E. Leontie, B. Narahari, and R. Simha. A compiler-hardware approach to software protection for embedded systems. *Computers and Electrical Engineering*, pages 315–328, 2009.

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

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

[29] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of computer and system sciences*, 28(2):270–299, 1984.

[30] M. Hong, H. Guo, and S. 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.

[31] M. Hong, H. Guo, and S. Parameswaran. Dynamic encryption key design and management for memory data encryption in embedded systems. In *Proceedings. IEEE Annual Symposium on VLSI*, pages 70–75, 2013.

[32] 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.

[33] 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.

[34] M. Jeff, F. Peter, B. Gunnar, and G. C. Julio. Earliest known uses of some of the words of mathematics, Retrieved on March 28, 2014.

[35] J. Kilian and P. Rogaway. How to protect des against exhaustive key search. In *Advances in CryptologyCRYPTO96*, pages 252–267. Springer, 1996.

[36] P. Kocher, R. Lee, G. McGraw, A. Raghunathan, and S. Ravi. Security as a new dimension in embedded system design. In *Proceedings - Design Automation Conference*, pages 753–760, 2004.

[37] 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.

[38] O. Kmmerling and M. G. Kuhn. Design principles for tamper-resistant smartcard processors. 1999.

[39] M. Lee, M. Ahn, and E. J. Kim. Fast Secure Communications in Shared Memory Multiprocessor Systems. *IEEE Trans. Parallel Distrib. Syst. (USA)*, 22(10):1714 —- 21, 2011.

[40] R. B. Lee, P. CS. Kwan, J. P McGregor, J. Dwoskin, and Z. Wang. Architecture for protecting critical secrets in microprocessors. In *Computer Architecture, 2005. ISCA'05. Proceedings. 32nd International Symposium on*, pages 2–13. IEEE, 2005.

[41] R. Lidl. *Introduction to finite fields and their applications.* Cambridge university press, 1994.

[42] D. Lie, C. Thekkath, M. Mitchell, P. Lincoln, D. Boneh, J. Mitchell, and M. Horowitz. Architectural support for copy and tamper resistant software. *ACM SIGPLAN Notices*, 35(11):168–177, 2000.

[43] M. Liskov, R. L. Rivest, and D. Wagner. Tweakable block ciphers. In *Advances in CryptologyCRYPTO 2002*, pages 31–46. Springer, 2002.

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

[45] 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.

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

[47] 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.

[48] S. Moore, R. Anderson, P. Cunningham, R. Mullins, and G. Taylor. Improving smart card security using self-timed circuits. In *Technology, Fourth AciD-WG Workshop, Grenoble, ISBN*, pages 211–218, 2002.

[49] 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.

[50] K. Nyberg. Differentially uniform mappings for cryptography. In *Advances in cryptologyEurocrypt93*, pages 55–64. Springer, 1994.

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

[52] W. W. Peterson and D. T. Brown. Cyclic codes for error detection. *Proceedings of the IRE*, 49(1):228–235, 1961.

[53] S. Ravi, A. Raghunathan, P. Kocher, and S. Hattangady. Security in Embedded Systems: Design Challenges. *ACM Trans. Embed. Comput. Syst.*, 3(3):461–491, August 2004.

[54] 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.

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

[56] P. Rogaway, M. Bellare, J. Black, and T. Krovetz. OCB: a block-cipher mode of operation for efficient authenticated encryption. In *ACM conference on Computer and communications Security*, 2001.

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

[58] V. Shoup. Using hash functions as a hedge against chosen ciphertext attack. In *Advances in CryptologyEUROCRYPT 2000*, pages 275–288. Springer, 2000.

[59] J. Soto. Randomness Testing of the Randomness Testing of the Advanced Encryption Standard Candidate Algorithms. pages 0–9, 1999.

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

[61] J. Soto and L. Bassham. Randomness Testing of the Advanced Encryption Standard Finalist Candidates Randomness Testing of the Advanced Encryption. 2000.

[62] G. E. Suh, D. Clarke, B. Gasend, M. Van Dijk, and S. Devadas. Efficient memory integrity verification and encryption for secure processor. In *36th International Symposium on Microarchitecture*, 2003.

[63] G. E. Suh, D. Clarke, B. Gassend, M. Van Dijk, and S. Devadas. Aegis: architecture for tamper-evident and tamper-resistant processing. In *Proceedings of the 17th annual international conference on Supercomputing*, pages 160–171. ACM, 2003.

[64] 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.

[65] S. William and W. G. Stallings. *Cryptography and Network Security, 4th Edition*. Pearson Education India, 2006.

[66] ANSI X9.9. American national standard for financial institution message authentication. 1981.

[67] C. Yan, B. Rogers, D. Englender, D. Solihin, and M. Prvulovic. Improving cost, performance, and security of memory encryption and authentication. In *33rd International Symposium on Computer Architecture*, 2006.

[68] J. Yang, L. Gao, and Y. Zhang. Improving memory encryption performance in secure processors. *IEEE Transactions on Computers*, 54(5):630–640, 2005.

[69] A. C. Yao. Theory and application of trapdoor functions. In *Foundations of Computer Science, 1982. SFCS'08. 23rd Annual Symposium on*, pages 80–91. IEEE, 1982.

[70] K. Yasuda. A new variant of pmac: beyond the birthday bound. In *Advances in Cryptology–CRYPTO 2011*, pages 596–609. Springer, 2011.

[71] B. Yee and J. D. Tygar. Secure coprocessors in electronic commerce applications. In *In Proceedings of The First USENIX Workshop on Electronic Commerce*, pages 155–170, 1995.