JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 36, 709-726 (2020)
DOI: 10.6688/JISE.202007_36(4).0002

A Memory Confidentiality Protection Scheme Based on Dynamic Keys and Compression Counter

PENGFEI YANG[1], JIAYANG HUANG, XIN MI AND BO WAN[+]
School of Computer Science and Technology
Xidian University
Xi'an, 710071 P.R. China
E-mail: pfyang@xidian.edu.cn[1]

More and more smart mobile devices with storing and processing capabilities have been designed to adapt to the development of wireless mobile networks. Different from data processing by network center node previously, edge devices can also process initial data to produce effective information, which yields new security requirements and challenges. The current confidentiality protection methods which consume excessive memory space and reduce the system performance sharply are not applicable for this kind of devices which have limited resources and power dissipation. This paper presents a memory confidentiality protection method based on optimized dynamic keys and compression counters, which makes improvements from two aspects: the algorithm structure and counter overflow update algorithm. The design of dynamic mapping table and partition of counter area reduce the on-chip memory occupation. The optimized update algorithm reduces the amount of calculations when the counter overflows. The proposed method is compared with the split counter and block encryption mode. The experiment results show that the resource occupation is less and the average reduction rate of 11.84% is significantly better in the aspect of system performance.

Keywords: memory security, data confidentiality, dynamic keys, compression counter, edge devices

1. INTRODUCTION

The emergence of wireless mobile networks was firstly applied for military purposes. As various expanding researches on wireless mobile networks rapidly generated, especially with the advent of the 5th generation wireless system, wireless mobile networks could provide more use cases to connect society at large. As Fig. 1 illustrated, different wireless mobile networks use cases are specified, such as vehicle-to-vehicle and vehicle-to-infrastructure communications, industrial automation, health services, smart cities, smart homes and so on. The new architecture, new technologies, and new use cases in the wireless mobile networks will bring challenges to security protection.

Along with wireless mobile networks applying in varieties of scenarios, the demands of today's consumers for information acquisition are significant. Thereinto, edge devices mainly based on embedded systems in wireless mobile networks should contribute to the data collecting and processing, which requires edge devices have certain storage capacity and computation power. But the current edge devices have limited capacities in terms of data processing or storage. This raises a security challenge, because cryptographic op-

Received August 27, 2019; revised October 3 & November 12, 2019; accepted November 19, 2019.
Communicated by Xiaohong Jiang.
[+] Corresponding author.

erations would be energy or memory consuming. To sum up, storage burden and security concerns are the most challenging issues of the edge devices in the wireless mobile networks [1].

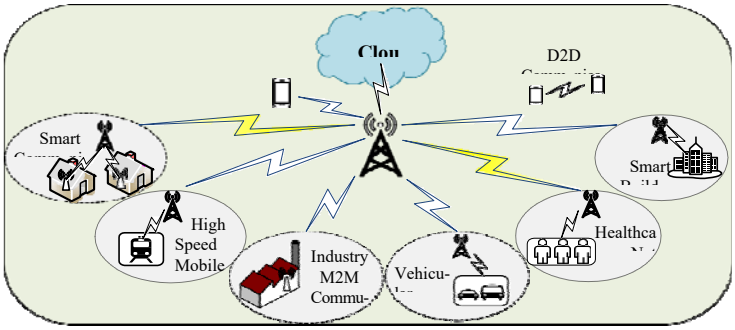


Fig. 1. Different wireless mobile networks use cases.

Moreover, edge devices are easily exposed to physical attackers for their easy access [2]. For example, the hacker organization Equation Group, through the back door of hard disk firmware, stole data from Samsung, Seagate, Toshiba and the other well-known hard disk company. By attacking storage devices and stealing database information, called 'drag database', hackers caused an annual loss of billions of dollars so it is necessary to construct a secure system for data security. Due to the broadcast nature and the limited bandwidth of wireless communications, it is possible but difficult to provide security features such as authentication, integrity and confidentiality. Chief among them is the security of edge devices memory, which is the bottom layer of the security architecture of system. The security of memory data affects or even determines the security of application protocols, algorithms, and the whole system.

Attacks against memory can be categorized into two types, namely, passive attacks and active attacks [3, 4]. For active attacks, attackers attempt to modify the data or interrupt of legitimate communications. Unlike active attacks, passive attacks are difficult to detect, because they are intended to gather data or execute transactions, but not alter the data value or affect the normal operation. Confidentiality protection methods are effective ones to prevent the passive attacks, it protects data transmission from passive attacks by limiting the data access to intended users only and preventing the access from or disclosure to unauthorized users.

However, the existing confidentiality protection methods consume excessive memory space and reduce the system performance sharply, so they are not applicable for wireless mobile networks edge devices which has limited resources and power dissipation. This paper presents a memory confidentiality protection scheme (CPS) based on optimized dynamic keys and compression counters, which makes improvements from two aspects: the algorithm structure and the counter overflow update algorithm. The main contributions of this article are as follows:

(1) Reducing the on-chip memory occupation. The counters in the method are divided into two parts: block counters and compression counters. They are part of seed in-

formation which can be saved in the off-chip memory, so the on-chip memory occupation is reduced. At the same time, a dynamic mapping table structure is designed to match keys with the block counters and compression counters respectively, which could eliminate the statistical relationship between the ciphertext and plaintext, making the security level higher.

(2) Improving the efficiency of overflow update algorithm. Based on the optimal structure, an update algorithm is proposed to solve the problem of ciphertext re-encryption when the counter overflows, which reduces the amount of calculations and improves the system efficiency.

The proposed method is quantitatively evaluated with the SimpleScalar architecture simulation tool. The experimental results show that, the resource occupation is less, and the average reduction rate of 11.84% is significantly better than either the split counter mode method or the block encryption method in the aspect of system performance, indicating that it can effectively reduce the impact of the encryption and decryption process on the system performance.

The remainder of paper is organized as follows. Section 2 introduces the related research work on confidentiality protection methods. Section 3 presents the structure and algorithm flow of the memory confidentiality protection scheme of dynamic keys and compression counters. The experiments with analysis are presented in Section 4. Finally, the conclusion is presented in Section 5.

2. RELATED WORK

The traditional hardware-based security model is shown in Fig. 2, the whole system can be divided into two areas: the trusted area and the untrusted area [5, 6]. The processor and its registers and cache are considered secure because it is impossible to implement the physical attacks to them for high cost. The outside parts of processor chip, such as the system bus, memory, disk, I/O devices are considered vulnerable to attack. To prevent physical attacks against off-chip memory, the memory confidentiality should be guaranteed. The schemes based on the block encryption mode [7] and the counter encryption mode [8] are the major methods of memory confidentiality protection. For the block encryption mode, plenty of system delays were caused by the encryption and decryption operations executed on the critical path of memory read and write [9].

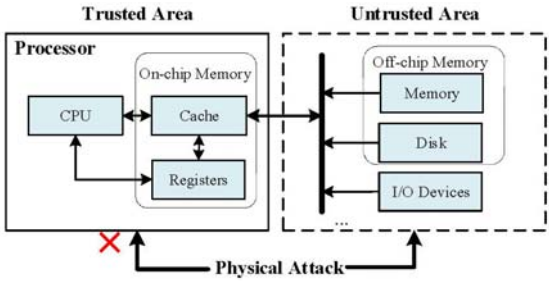


Fig. 2. The hardware security model.

The advantage of the counter encryption mode is that the key stream generation process and the memory read/write are performed in parallel so that the process of encrypting the keystream can be covered with the memory read/write delay. And the complexity of the keystream and XOR operation of the plaintext is constant, making the efficiency higher [10]. Simultaneously, the counter encryption mode, a one-time-pad (OTP) method, can eliminate the statistical relationship between ciphertext and plaintext and get the higher security level.

However, the counter encryption mode has obvious drawbacks. The counter encryption mode required that the seed of the encryption/decryption key stream was globally unique, but the storage space for the seeds was limited. Once the counter seed overflowed, the counter key should be updated, and the current ciphertext would be re-encrypted by the new keys again, which is called 'system freeze'. If the system were in the state of 'system freeze', it would consume heavy loads of system computing and storing resources [11]. To solve these problems, the improvements mainly contain three aspects: counter structure, encryption resources prediction, and data caching.

The modifications of the counter structure include pseudo-random numbers mode, split counter mode (SPLIT), variable-length counters mode and dynamic encryption streams mode [12, 13]. The seed encrypted by pseudo-random number method was comprised of an address and a global random counter [14]. The only condition for guaranteeing the encryption seed was that the address should be assigned to the same random number during the machine activity period, which is literally simple but difficult to implement. The uniqueness of seeds cannot be guaranteed, and the security level was not high enough. SPLIT, considered as a safe and efficient method at present, allocate a master counter and slave counter with a segment-like management scheme. Once the slave counter overflowed, the master counter would be incremented by one, then reset from the counter, and re-encrypt the counter-protected memory blocks only, thereby reducing the amount of calculations and encryption delays. According to the principle of memory access locality, the memory was divided into hot and cold areas, and the variable-length counters were utilized [15]. Shorter counters were allocated in the cold area, while longer counters in the hot area, which reduced the probability of the hot area overflow, so that the spillover would be less and the overall system resource overhead smaller. Dynamic random number mode [13] adopted the idea of split counter structure, replacing the master counters with random numbers, and the random master counters were allocated dynamically. When a counter overflowed, a new random master counter was allocated. When there was no data block reference to random master counter, the random master counter was released, so that multiple data blocks could share the same master counter. This scheme could hide the consumption of re-encryption, making the efficiency higher.

The efficiency of the memory confidentiality protection methods can also be improved by caching the information for the counter encryption ahead, such as the cache key stream value [16]. All these methods attempted to eliminate intermediate calculations, so the data were directly calculated in the cache, but causing excessive memory occupation. For the new characteristics of NVM memory, Jalili *et al.* [17] incorporated the counter encryption mode into worn balance to reduce encryption overhead and extend service length. As encryption affected the data diffusion, it brought higher power and energy consumption. Swami *et al.* [18] proposed a method of reducing diffusion by guaranteeing confidentiality, thereby reducing the energy consumption and prolonging the life.

For the dynamic random number method, when a certain counter overflowed, it was not necessary to re-encrypt other related data blocks, and only a new master counter was asked to be allocated to this overflow counter because it could map each data block to the master counter. This method would occupy part of the on-chip memory space, but could solve the problem of efficiency reduction caused by the re-encryption of the split counter mode.

Both the split counter method and the dynamic random method attempted to make the master counters large enough to guarantee the counter not to overflow during the algorithm execution. The overall resources requested by the dynamic random number method are smaller than those by the split counter method. The former method consumed excessive on-chip resources which was more precious, causing the reduction of the cache hit ratio. In this paper, we make the improvements based on the dynamic random number method of the counter encryption mode, ensuring that the on-chip memory occupation is reduced and the efficiency of the encryption and the re-encryption is improved.

3. ALGORITHM OPTIMIZATION

The methods based on the counter encryption mode have become the mainstream of memory confidentiality protection methods because the encryption process and the memory read/write performed in parallel, the process of encrypting operation can be covered with the memory read/write delay. The counter encryption method, an OTP encryption method, which guarantees the uniqueness of the seeds for encryption. It means the uniqueness not only in the temporal dimension but also spatial dimension. The address of data blocks can be considered as a part of the seeds to guarantee the uniqueness in the spatial dimension, and the counter values at different time for the data blocks with same address can guarantee the uniqueness in the temporal dimension.

The seeds should be long enough to guarantee the uniqueness, but causing excessive memory space occupation in the on-chip memory, thereby reducing the system performance. If the length of seeds decreases, as does the length of the counters, which could save the memory space but would increase the counter update frequency because of the counter overflow. In this paper, firstly, the counters are divided into two parts: block counters and compression counters, storing in the off-chip memory of the untrusted area, thus the on-chip memory occupation is reduced. At the same time, a dynamic mapping table saving in the trusted area is to map the keys to block counters and compression counters respectively. Secondly, the counter encryption mode applied in this paper is an OTP encryption method which could eliminate the statistical relationship between the ciphertext and plaintext, which makes the security level higher. Finally, based on the optimal structure, an update algorithm is designed to solve the problem of ciphertext re-encryption when the counter overflows, which reduces the amount of calculations and improves the system efficiency.

3.1 Algorithm Structure

As is shown in Fig. 3, four parts constitute the main structure of the confidentiality protection scheme: block counter mapping table, key table, compression counters, and

block counters.

The block counter mapping table stored in the trusted area is the kernel to dynamic mapping, which is to store and manage addresses of the compression counters and keys that block counters map to. Instead of the actual physical addresses, it stored the offset addresses, which could reduce the memory occupation in the trusted area because the numbers of keys and compression counters are smaller. The key table is a circular list structure stored in the trusted area, in charge of maintaining and updating keys and producing a new key when the compression counters overflow.

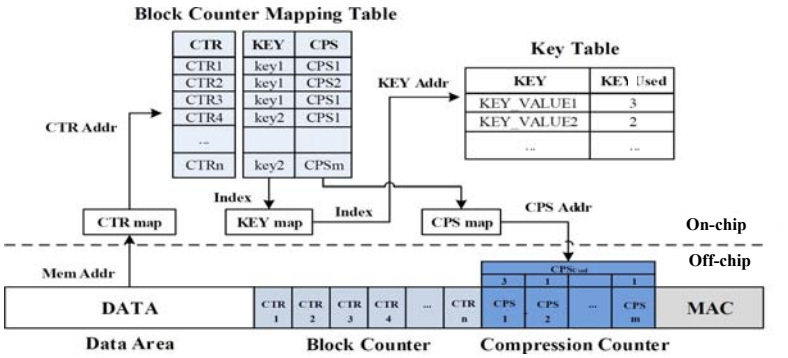


Fig. 3. Algorithm structure.

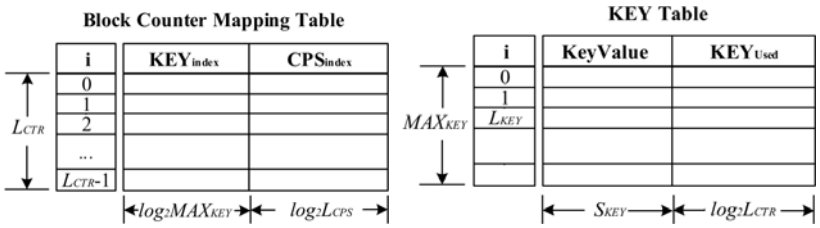


Fig. 4. The structures of the block counter mapping table and the key table.

The structures of block counter mapping table and key table are shown in Fig. 4. $KEY_{index}(i)$ represents the address of the key_i corresponding to the ith block counter, $CPS_{index}(i)$ represents the address of the compression counter of the ith block counter, and L_{CTR} is the number of block counters. The key table is a circular linked list structure stored in the on-chip memory of the trusted area, L_{KEY} indicates the length of the key table that has been used, MAX_{KEY} indicates the maximum length of the key table, S_{KEY} is the length of the key, KEY_{used} represents the number of KEY mapped to the block counter, $KEY_{usedMAX}$ represents the maximum number of keys that can be mapped and $KEY_{usedMAX} = L_{CTR}$. For a record, with the index value i in the key table, if KEY_{used} is m, there are m blocks counter corresponding to the key with the address i, and the value is $Key(i)$.

The compression counter is stored as circular linked list in continuous storage space in the off-chip memory. The number of the compression counters keeps constant, which

is different from the block counters. The block counter is the basic unit of the counter structure, also stored in the untrusted area. The structure of the compression counters and block counters are shown in Fig. 5. L_{CPS} indicates the number of compression counters, $CPS_{used}(i)$ represents the number of compression counters whose index value i is mapped to the block counters, $CPS_{usedMAX}$ indicates the max number of compression counters mapped to the block counters, which is equal to the number of block counters L_{CTR}. For a compression counter with index i, if $CPS_{used}(i)$ is m, there are m block counters corresponding to this compression counter whose value is $CPS(i)$. In the block counters, $CTR(i)$ indicates the counter corresponding to the ith encrypted data block. The lengths of all the block counters are the same and expressed by S_{CTR}, the maximum among which is MAX_{CTR}.

Based on the structure, the corresponding relationship of data, block counters and compression counters are not established by their location-relation but the block counter mapping table. When the system executes the decryption operation, firstly, check whether the counter or compression counter overflows or not when updating the block counter. If overflows, execute the update algorithm and then use the updated key, otherwise, use the current key. The logical separation of the encryption and decryption keys could reduce the amount of calculations of the ciphertext re-encryption. When the counter overflows, only are the decryption keys utilized to decrypt ciphertext. Meanwhile, the data block corresponding to the overflow counter is encrypted with the encryption keys. The decryption keys are still saved and for future decryption of other data blocks that have not updated the counter yet, so the problem of re-encrypting the ciphertext brought by updating the key in the counter mode is solved.

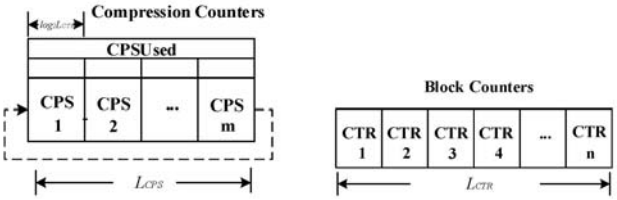


Fig. 5. The structures of compression counter and block counter.

3.2 Seed Structure

Due to the dynamic key and compression counter mapping scheme, the seeds applied in this method will be different from those in the traditional methods. The selection of seeds and the generation of the keystream for the encryption and decryption will be described in details.

In this paper, a new seed is introduced to generate the keystream *Pad* for encryption and decryption:

$$Pad = Encrypt_{key}(seed). \tag{1}$$

$$seed = CPS||CTR||CPS_{addr}||DATA_{addr}. \tag{2}$$

As is shown in Eqs. (1) and (2), *key* is the encryption key, *seed* is to generate the

keystream, *CPS* is the value of compression counter, *CTR* is the value of block counter, CPS_{addr} is the address of compression counter, and $DATA_{addr}$ is the address of the data block for encrypting. Different from the split counter mode, the CPS_{addr} is added as part of the seed in our method to gurantee the uniqueness and security of the key stream.

Although the status of the compression counter in our method is similar to that in SPLIT, the block counter and compression counter establish a dynamic mapping relationship, which is fixed in SPLIT. A compression counter can be mapped by multi-block counters. If the compression counter address is not as a part of the seed, seed duplication would occur when executing mapping operations. For example, as is shown in Fig. 6, the compression counter CPS_{addr1} and CPS_{addr2} are mutually independent, their values CPS_1 and CPS_2 may be the same. At the time of $T1$, the block counter corresponding to the $DATA_3$ data block is CTR_3, and the compression counter corresponding to the block counter CTR_3 is CPS_{addr1}. If not consider the CPS_{addr}, the encryption and decryption seed is $seed_1 = CPS_1\ CTR_3\ DATA_{addr3}$. At the time of $T2$, the mapping relationship is converted to the compression counter CPS_{addr2}, and $seed_2 = CPS_2\ CTR_3\ DATA_{addr3}$. Due to $CPS_1 = CPS_2$, $seed_1 = seed_2$, the encryption key stream is duplicated. This situation should be avoided, so compression counter address CPS_{addr} is as a part of the seed, which solves the problem of seed repetition and meets the security requirements without increasing the extra storage space.

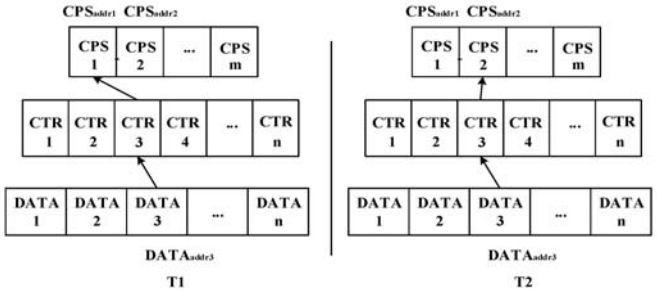


Fig. 6. Example of seed value repetition.

3.3 Algorithm Flow

Based on the modifications above, the framework of the confidentiality protection scheme proposed in this paper is shown in Fig. 7. It contains three basic algorithms: the decryption algorithm when reading data from the off-chip memory in the untrusted area, the encryption algorithm when writing data to the off-chip memory, and the update algorithm when block counters overflow. The overflow update algorithm is the key to improve the encryption efficiency. The specific descriptions of all these algorithms are presented as follows.

Algorithm 1: The decryption algorithm

1: /*Generate decryption Pad*/
2: $key = KEY\ (KEY_{index}(i))$
3: /* Read the decryption key corresponding to data block *i* from the key table */

4: $Pad = Encrypt_{key}(CTR(i)\ CPS(CPS_{index}(i))$
5: $CPS_{addr}(i)\ DATA\ [i]_{addr}$);
6: /*Encryption *Pad* is generated using the block counter value $CTR(i)$, the compression counter value $CPS(CPS_{index}(i))$, the compression counter address $CPS_{addr}(i)$ and the data block address $DATA\ [i]_{addr}$*/
8: /*Encrypt plaintext data using Pad*/

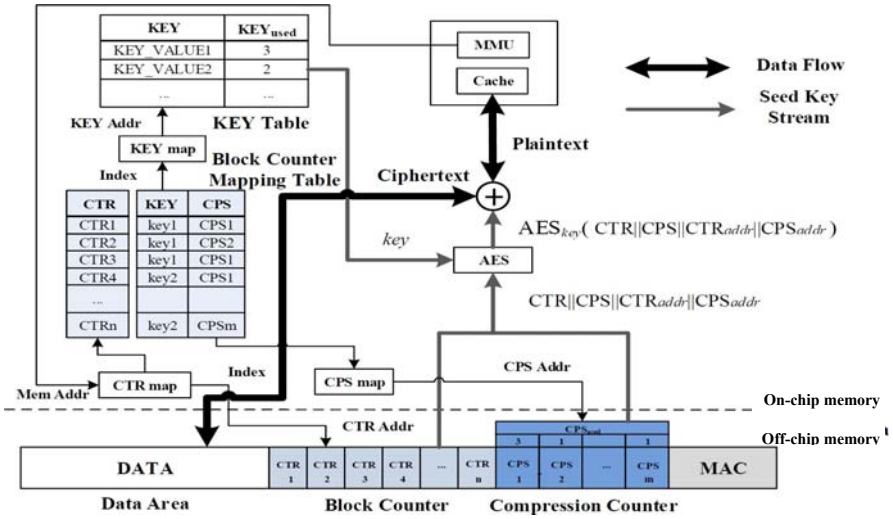


Fig. 7. The framework of the confidentiality protection scheme.

Assume that the compression counter and block counter values have been initialized and there is an existing initial key in the key table. When reading the data from the untrusted memory, the system judge whether the access address is in the cache. If not, the decryption algorithm is executed. Firstly, the block counter, the compression counter, and the key corresponding to the ciphertext block should be obtained to generate a seed. If the seed parameter is already cached, it can be executed in parallel with the read memory operation. Then, the collected seed is encrypted by the key to generate a decryption pad, and finally the ciphertext can be decrypted by the decryption pad. The algorithm flow is shown in Algorithm 1. When writing data to the off-chip memory, the encryption algorithm is executed, as is shown in Algorithm 2. Firstly, update the block counter corresponding to the data block and determine whether the counter overflows. If does, execute the overflow update algorithm. Then, similarly to the decryption algorithm, the seed is generated by combining the block counter, the compression counter, and its address to get the keystream by encrypting the seed with the key. Finally, the ciphertext is re-encrypted to get the new one and write data into memory.

In the memory read/write process, only can the seed parameter corresponding to the data block be obtained through the block counter dynamic mapping table, which simplifies the seed generation process. Algorithm 3 is the block counter overflow update algorithm. When the block counter overflows, the algorithm can update the block counter mapping table, the key table, and the compression counter.

Algorithm 2: The encryption algorithm

1: /*Encrypt the plaintext data block and write it to the data block DATA [i]*/
2: /*Generate decryption Pad*/
3: **if** ($CTR(i)$ + 1) mod L_{CTR} = 0 **then**
4: /*Execute the block counter overflow update algorithm (Algorithm 3) if counter overflows*/
5: **end if**
6: $CTR(i) = (CTR(i) + 1) \bmod L_{CTR}$;
7: $key = KEY\ (KEY_{index}(i))$;
8: $Pad = Encrypt_{key}(CTR(i)\ CPS(CPS_{index}(i))$
9: $CPS_{addr}\ DATA\ [i]_{addr}$;
10: /*Encryption *Pad* is generated using the block counter value $CTR(i)$, the compression counter value $CPS(CPS_{index}(i))$, the compression counter address CPS_{addr} and the data block address $DATA\ [i]_{addr}$*/
11:
12: /*Encrypt plaintext data using Pad*/
13: $DATA\ [i] = Ciphertext$;

The memory confidentiality protection scheme proposed in this paper adopts a two-level dynamic mapping method. The block counters dynamically map compression counters and keys respectively. The key table and the block counter mapping table are stored in the on-chip memory, while the compression counter and the block counter are stored in the off-chip memory. When the block counter overflows, the update are performed from two aspects. Firstly, the compression counters are updated. Since the compression counter applies a circular linked list structure, the value is incremented by one without changing the mapping relationship when it is only mapped to current block counter. Otherwise, the mapping relationship between the compression counter and its corresponding block counters should be adjusted in advance. Moreover, keys should be updated after the update of compression counters. The data block should be mapped to the next key if the mapped key is not the last one in the key table. Otherwise, a new key is generated to allocate, and the unmapped key is reclaimed and released.

4. DISCUSSION AND EXPERIMENTS

In this section, the advantages of the optimized confidentiality protection scheme are presented from four aspects: cache hit ratio, resource occupation, the security and expansibility, and algorithm performance. The SimpleScalar simulator is applied to verify the proposed method, and the MiBench basic assemble test is applied as the test program.

4.1 Cache Hit Ratio

SPLIT stores the master and slave counter in the continuous off-chip memory space. The relationship between the master and slave counter is established by their location relation. The counter storage and master-slave storage structures in this paper are shown in Fig. 8. Each update operation needs to update all data blocks corresponding to the

slave counter, which causes a large amount of calculations. For the algorithm proposed in this paper, the compression counters and the block counters are stored in different addresses of the off-chip memory and the relationship of the data, block counter and compression counter is established not by their location-relation but the block counter mapping table. Only is the table saved in the on-chip memory of trusted area, which occupies less memory space than any other algorithms mentioned above. In addition, the dynamic mapping method is applied to improve the cache hit ratio so that the parallelism of the *pad* generation process and reading cache data process increases, thereby making the efficiency of the decryption higher.

Algorithm 3: The update algorithm

1: /* When the block counter CTR [i] of data block i overflows, generates or updates the compression counters and keys that it mapped */
2: /* Update compression counter value, if only one block counter is mapped by this compression counter */
3: **if** CPS_{used} ($CPS_{index}(i)$) = 1 **then**
4: **else**
5: /* If there are multiple block counters are mapped by this compression counter, then allocate a new compression counter to the block counter whose data block i overflows */
6: /* Determine if this compression counter is the last compression counter in the com pression counter table */
7: **if** $CPS_{index}(i) = L_{CPS}$ 1 **then**
8: /* If it is the last compression counter, update the compression counter and map the key */
9: **if** $KEY_{index}(i) = L_{KEY}$ 1 **then**
10: /* Mapping the data block to the next key if it is not the last key */
11: $KEY_{used}(KEY_{index}(i))$;
12: $KEY_{index}(i) = (KEY_{index}(i)+1) \mod L_{KEY}$;
13: $KEY_{used}(KEY_{index}(i))$++;
14: **else**
15: /*Generate the new keys*/
16: *key = new key number*();
17: **if** $KEY_{used}(KEY_{index}(i)) = 1$ **then**
18: /*The current key is mapped by only one data block, update the key and map the data block to the new key */
19: $KEY(KEY_{index}(i)) = key$;
20: **else**
21: /* Otherwise, allocate new storage space for the new key and add it to the key table */
22: **if** L_{KEY} MAX_{KEY} **then**
23: $Key(L_{key}) = key$;
24: $KEY_{index}(i) = L_{KEY}$;
25: $L_{KEY} = L_{KEY} + 1$;
26: **else**
27: *trow execution*();

28: **end if**
29: **end if**
30: **end if**
31: **end if**
32: $CPS_{used}(CPS_{index}(i))$;
33: $CPS_{index}(i)=(CPS_{index}(i)+1) \bmod L_{CPS}$;
34: $CPS_{used}(CPS_{index}(i))$++;

The master counter of the split counter mode is stored as a page block. When a specified period continuously accesses a logical page data block, the number of cache replacements is reduced, and the cache hit ratio increases. However, it may cause an increase of the cache miss rate when the memory is accessed across different pages. In the dynamic mapping method of this paper, the compression counter can be mapped by more data blocks. In other words, the compression counter would be accessed more frequently than the main counter. For the common cache replacement strategy least recently used (LRU), the data blocks accessed more frequently would stay longer in the cache memory, so the cache hit ratio of the compression counter increases. Moreover, compared with SPLIT, the same size of cache space could save more block counters, and the cache hit ratio of the block counter increases, thus achieving a higher cache hit ratio with this scheme.

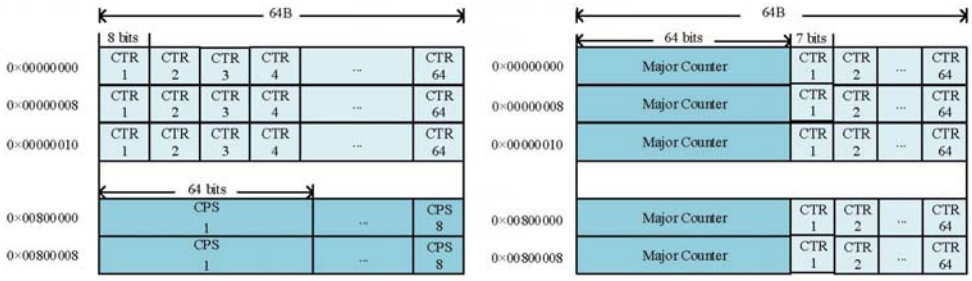


Fig. 8. CPS counter storage and master-slave storage structures.

4.2 Resource Occupation

The off-chip memory occupation with SPLIT method and the on-chip memory occupation with the dynamic random number method are compared. Assume that the key lengths of SPLIT and CPS method equal S_{KEY}, the length of the seeds generated during the encryption and decryption is K bits, the number of block counters is L_{CTR} whose length is S_{CTR}. Firstly, the off-chip memory occupation is estimated. For SPLIT method, the number of master counters is L_{MAJ}. The number of both slave counters and data blocks are equal to L_{CTR}, and the relationship between them is L_{MAJ}: L_{CTR} = 1 : 64. For CPS method, the number of block counters and data blocks are also equal to L_{CTR}. Because with the dynamic mapping method, fewer compression counters are requested: $L_{CTR} > L_{CPS} \times 64$, so $L_{CTR} < L_{MAJ}$. In addition, assume that the same seed length is for the encryption and decryption, $S_{seed} = S_{CTR} + S_{MAJ} + S_{DATA_{addr}} = S_{CTR} + S_{CPS} + S_{CPS_{addr}} + S_{DATA_{addr}}$.

Thus, $S_{MAJ} = S_{CPS} + S_{CPS_{addr}}$. Then, if the seed length is the same, the number of compression counter S_{CPS} is shorter than master counter number S_{MAJ}, and the difference is the number of bits of a compressed counter address $S_{CPS_{addr}}$. In cases where the block counter has the same number of bits and length as the slave counter, the compression counter has a shorter number of bits and length than the master counter. Therefore, the off-chip memory occupation of CPS meth CPS method is less than SPLIT method.

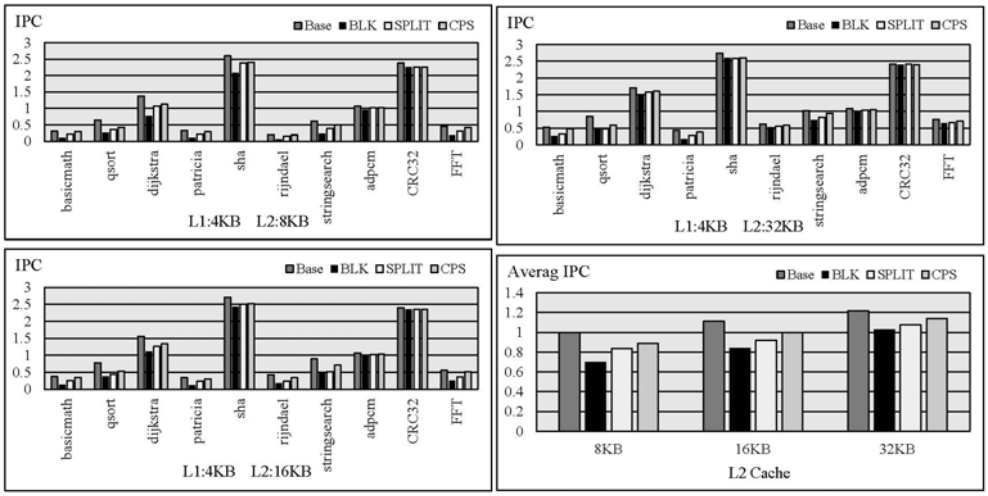


Fig. 9. The IPC of typical applications based on different L2 cache.

Besides, the on-chip memory occupation is estimated. The resource occupation of the dynamic random number method can be divided into block counter table and the random number table. The block counter table contains block address, counter value, and index value of the random number table. L_{CTR} represents the number of data blocks, S_{DATA} denotes the block address bits, and S_{CTR} means the number of counter bits. The random number table contains the index value of the random number table, random number and the number of random number mapped. L_{RANDOM} means the number of random number tables, S_{RANDOM} represents the length of the random number, $RANDOM_{used}$ is the number of random number mapped, $RANDOM_{usedMAX}$ indicates the maximum number of random number mapped. The range of $RANDOM_{used}$ values is [0, $RANDOM_{usedMAX}$], which depends on the number of block counters, and $RANDOM_{usedMAX} = L_{CTR}$.

Based on the definitions above, the on-chip memory occupation of the random number method SP_{DR} is the sum of the random number table SP_R and the block counter table SP_{CTR}: $SP_{DR} = SP_R + SP_{CTR}$, and $SP_R = L_{Random} \times (S_{Random} + Log_2 L_{CTR})$. So $SP_{CTR} = L_{CTR} \times (S_{Random} + Log_2 L_{CTR})$.

The on-chip memory occupation of CPS method proposed in this paper SP_{DK} is the sum of the block counter mapping table SP_{MAP} and a key table SP_{KEY} : $SP_{DR} = SP_{MAP} + SP_{KEY}$, in which $SP_{MAP} = L_{CTR} \times (Log_2 L_{CPS} + Log_2 L_{KEY})$ and $SP_{KEY} = L_{KEY}\ (S_{KEY} + Log_2 L_{CPS})$. Assume that $L_{Random} = L_{KEY}$ and $S_{Random} = S_{KEY}$, $SP_{DR} - SP_{DK} = 6(L_{CTR} + L_{KEY}) > 0$, which means the on-chip memory occupation of the CPS proposed in this paper is less than that of the random number method.

4.3 The Security and Expansibility

In the aspect of security, the traditional memory confidentiality protection methods, as SPLIT method and dynamic random number method, only update the seeds, but the keys keep constant. The CPS method proposed in this paper not only dynamically maps and updates the counters as seeds, but also dynamically updates the keys for encryption and decryption, which makes the security level of CPS method higher. The extensibility of data security protection needs both confidentiality and integrity protection. The separation design of data blocks and counter blocks can improve the scalability and data locality of the system, so as to improve the cache hit ratio in the process of confidentiality and integrity verification. At the same time, as is shown in Fig. 7, when constructing an integrity checking tree with counter mode, it is not necessary to construct the tree for the entire memory data space, but for the counter space corresponding to the data block. As a result, the storage space occupied by the checking tree and the amount of calculations are both reduced.

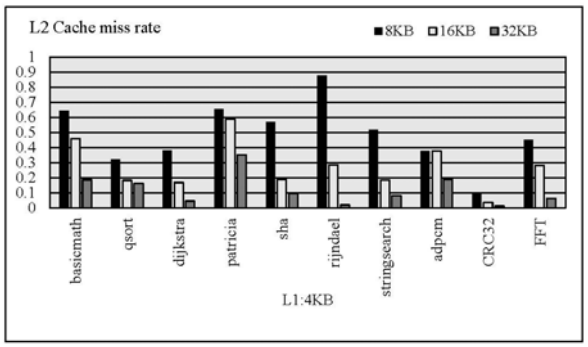


Fig. 10. The cache miss rates of typical applications based on different L2 cache.

4.4 Algorithm Performance

The CPS method proposed in this paper is compared with other three different confidentiality protection methods: the method without any confidentiality protection (Base), the direct block method (BLK) and SPLIT method, and its advantages have been verified. The L1 cache block size is 32B, the L2 cache are 8KB, 16KB, and 32KB respectively. The instructions per cycle (IPC) of ten typical applications based on different confidentiality protection methods are shown as Fig. 9. And the cache miss rates of typical applications based on different L2 cache are also compared, the results are shown in Fig. 10. The experimental results show that all the confidentiality protection methods would reduce the IPC of the system, and the large L2 cache can alleviate this reduction. Anyway, the IPC of CPS is the highest except for the Base method with no confidentiality protection. The performance advantage of CPS is prominent when the L2 cache is small. And when the L2 cache is large enough, the cache miss rate is low. And when methods perform encryption and decryption operations, the data exchange frequency with external memory decreases, so does the advantage of CPS.

To further illustrate the system performance, the decline rate of system performance $r = (base\ c)/base$ describes the effect of different confidentiality protection methods. In Fig. 11, the experimental results show that, compared with the Base method, the average decline rate of system performance based on CPS method is 10.75%, and the values based on BLK method and SPLIT method are 36.06% and 22.7% respectively, indicating that CPS method is the best one, especially when the L2 cache are 8KB and 16KB. The average decline rate of system performance based on CPS method are 11.6% and 12.08%, while the values based on BLK method are 46.36% and 38.71%, and SPLIT method are 24.32% and 26.09%. The proposed memory confidentiality protection scheme in this paper could significantly reduce the impact on system performance.

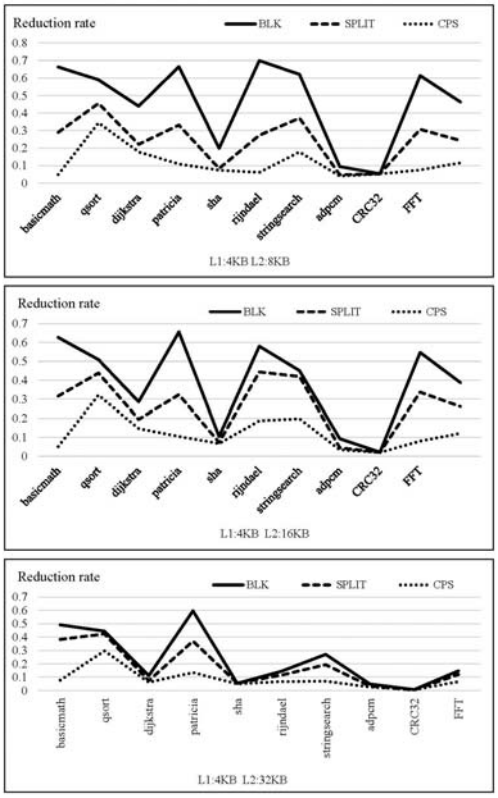


Fig. 11. Performance reduction rate of different confidentiality protection methods.

5. CONCLUDSION

It is necessary to construct a secure system to solve storage burden and security concerns which are the most challenging problems of smart edge devices in wireless mobile networks. This paper presents a memory confidentiality protection scheme based on optimized dynamic keys and compression counters and makes the improvements from two aspects: the algorithm structure and the counter overflow update algorithm. The

proposed memory confidentiality protection scheme is quantitatively evaluated with the SimpleScalar architecture simulation tool. The experimental results show that, the modified method can effectively reduce the impact of the encryption and decryption process on the system operation.

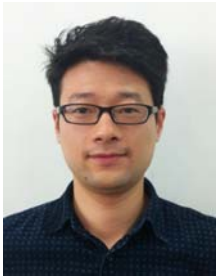
ACKNOWLEDGMENT

The authors would thank the support of National Key R&D Program of China (2017 YFB1102900) and the National Natural Science Foundation of China (61572385, 61702 395, 61972302).

REFERENCES

1. K. Huang, X.-S. Zhang, and X.-F. Wang, "Block-level message-locked encryption with polynomial commitment for IoT data," *Journal of Information Science and Engineering*, Vol. 33, 2017, pp. 891-905.
2. Y. Li, M. Chen, and J. Wang, "Another security evaluation of SPA counter measures for AES key expansion in IoT devices," *Journal of Information Science and Engineering*, Vol. 33, 2017, pp. 1085-1100.
3. I. Kwangseok, Y. Jung-Yeon, H.-J. Lee, and H.-N. Jeong, "Memory system monitoring data integrity and related method of operation," US Patent 9,542,264, 2017,
4. T. Wang, X. Cui, Y. Ni, D. Yu, X. Cui, and G. Qu, "A practical cold boot attack on rsa private keys," in *Proceedings of IEEE Asian Hardware Oriented Security and Trust Symposium*, 2017, pp. 55-60.
5. T. S. Lehman, A. D. Hilton, and B. C. Lee, "Poisonivy: Safe speculation for se cure memory," in *Proceedings of the 49th Annual IEEE/ACM International Symposium on Microarchitecture*, 2016, pp. 1-13.
6. P. Maene, J. Götzfried, R. de Clercq, T. Müller, F. Freiling, and I. Verbauwhede, "Hardware-based trusted computing architectures for isolation and attestation," *IEEE Transactions on Computers*, Vol. 67, 2018, pp. 361-374.
7. L. Guan, C. Cao, P. Liu, X. Xing, X. Ge, S. Zhang, M. Yu, and T. Jaeger, "Building a trustworthy execution environment to defeat exploits from both cyber space and physical space for arm," *IEEE Transactions on Dependable and Secure Computing*, Vol. 16, 2019, pp. 438-453.
8. 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 ACM International Conference on Supercomputing*, 2014, pp. 357-368.
9. Y.-S. Yeh, T.-Y. Huang, and H.-Y. Lin, "Structural binary CBC encryption mode," *Journal of Information Science and Engineering*, Vol. 25, 2009, pp. 937-944.
10. T. Cheng, D.-W. Gu, F.-Y. Hou, and Y.-Y. Zhang, "Survey of research on physical attack-resistant secure memory techniques," *Application Research of Computers*, Vol. 5, 2010, pp. 1601-1605.
11. M. Haifeng, Y. Nianmin, C. Shaobin, and H. Qilong, "Memory confidentiality and integrity protection method based on variable length counter," *Journal of Algorithms & Computational Technology*, Vol. 8, 2014, pp. 421-439.

12. M. Hong, H. Guo, and S. Parameswaran, "Dynamic encryption key design and management for memory data encryption in embedded systems," in *Proceedings of IEEE Computer Society Annual Symposium on Very Large Scale Integration*, 2013, pp. 70-75.
13. T. Liu and H. Guo, "Dynamic encryption key design and its security evaluation for memory data protection in embedded systems," in *Proceedings of IEEE International Conference on IT Convergence and Security*, 2014, pp. 1-4.
14. S. Yin, Z. Xie, C. Meng, P. Ouyang, L. Liu, and S. Wei, "Memory partitioning for parallel multipattern data access in multiple data arrays," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 37, 2018, pp. 431-444.
15. W. Walston and M. S. Diggs, "Systems and methods for improving the performance of non-volatile memory operations," Nov. 3 2015, US Patent 9,176,859.
16. M. T. Kurdziel, M. Lukowiak, and M. A. Sanfilippo, "Minimizing performance overhead in memory encryption," *Journal of Cryptographic Engineering*, Vol. 3, 2013, pp. 129-138.
17. M. Jalili and H. Sarbazi-Azad, "Endurance-aware security enhancement in nonvolatile memories using compression and selective encryption," *IEEE Transactions on Computers*, Vol. 66, 2017, pp. 1132-1144.
18. S. Swami, J. Rakshit, and K. Mohanram, "Secret: Smartly encrypted energy efficient non-volatile memories," in *Proceedings of the 53rd ACM/EDAC/IEEE Design Automation Conference*, 2016, pp. 1-6.



Pengfei Yang was born in 1985. He received the B.Sc., M.Sc. and Ph.D. degrees from Xidian University, Xi'an, China. He has been an academic visitor for one year in the University of Leeds, UK. He is currently a Lecturer in Xidian University. His current research interests include embedded system architecture, memory security and heterogeneous parallel computing.

Jiayang Huang was born in 1995. She received the B.Sc degree from Shandong University, Jinan, China. She is currently a Ph.D. candidate in Xidian University, Xi'an, China. Her current research interests include embedded systems and wearable sensing.

Xin Mi was born in 1992. He received the B.Sc.from Shanxi University, Shanxi, China, and the M.Sc. degrees from Xidian University, Xi'an, China. He is currently an Engineer in the Shenzhen City Tencent Computer System Co. Ltd., China. His current research interests include embedded system architecture and parallel computing.

Bo Wan was born in Leshan, Sichuan Province, China in 1976. He received the B.S. MS, and Ph.D. degrees from Xidian University, Xi'an, Shaanxi province. He is now an Associate Professor in the School of Computer Science and Technology at Xidian University, China. His current research interests include input/output technologies and systems, human-computer interaction and cloud computing.