A Strategy of Encryption and Decryption based in a Low Memory Environment

A Strategy of Encryption and Decryption based in a Low Memory Environment

Danzhi Wang, Zepeng Wu* and Yansong Cui

School of Electronic Engineering, Beijing University of Posts and Telecommunications, No. 10 of Xitucheng road, Haidian , Beijing, China
Email: sunroson@126.com; *18601277657@163.com; cuiys@bupt.edu.cn

Abstract. This paper proposes an information encryption and decryption Strategy that can run smoothly in a low memory environment. It divides the information into blocks, which consumes less memory in encryption. And then it adds the identifiers into each block of cipher text. When cipher text is decrypted, it reads identifiers, and divides the cipher text into groups and decrypt it based on the group identifier to obtain the plaintext. In addition, it analyses the root causes of garbled plaintext after decryption, proposes a strategy of increasing cipher text disorder, and provide a method to make a difference between all identifiers and cipher text content. The test results show that the proposed strategy can implement more secure encryption in low memory system than normal strategy, and reduce the memory requirement for encryption and decryption, and increase the complexity of encrypted cipher text. Practices has proved that this strategy can be used in the field of mobile phone encryption and decryption [2], and has certain use value

1. Introduction

Currently, although the cell phone has the memory of 1G to 2G, or even 6G to 8G, the memory which applications (APP) can use is still limited. The reason is that mobile phones support multiprocessing, which makes each APP can only use limited memory no more than 512Mb when it is running. So low memory running is more and more normal for today’s cell phone running. Meanwhile, the encryption and decryption technique, which is the most important way to guarantee the safety of information in cell phone, is becoming more and more important. So how to realize encryption and decryption in such a low memory environment is very important.

The traditional encryption strategy is loading all the information that needs to be encrypted into the memory at the same time, to perform the encryption operation to obtain the cipher text. When it is decrypted, all cipher text is loaded into memory and decrypted to obtain plaintext. When the amount of information that needs to be encrypted exceeds the threshold of app’s memory or mobile phones’ memory, the cipher text cannot be decrypted. The major problem of the traditional encryption strategy is that it need too much calculation complexity when the cipher text is very large, so traditional encryption and decryption strategy cannot be directly used in the lower memory device.

In this paper, it proposes a low-complex encryption and decryption strategy, which can adapt the low memory environment. It divides the cipher into different length of blocks, and then we get the encrypted and decrypted text by blocks. This strategy makes the encrypted and decrypted text be smaller, which reduce the complexity of the encryption and decryption. Furthermore, in order to make the rule of encryption and decryption not get found, it use non-uniform division.

The rest of the paper is organized as following: the section 2 gives a brief introduction of encryption and decryption; Section 3 gives the low-complex strategy of encryption and decryption;

section 4 gives the test results of the proposed encryption and decryption algorithm. Section 5 is the conclusion of the whole paper.

2. A Brief Introduction of encryption and decryption

In the field of encryption, symmetric-key encryption [3] is faster than public-key encryption [4]. Data encryption standard [5] and Advanced Encryption Standard [6] belongs to symmetric key encryption.

Data encryption standard, due to the 56-bit key, is not a secure encryption method. In addition, data encryption standard works slowly. As an alternative encryption algorithm to data encryption standard, the key's length of Advanced Encryption Standard is optional, which can be 128, 192 or 256 bits. In addition, since Advanced Encryption Standard is based on Substitution-permutation network architecture [7], it can achieve faster speed in software and hardware than data encryption standard.

Advanced Encryption Standard is used widely in the field of encryption and decryption in mobile phone. It is a kind of block cipher algorithm [8]. When the length of information needs to be encrypted exceeds 16 bytes, the information will be divided into blocks based on the length of 16 bytes, and the encryption and decryption is performed on the block basis. If the length is not an integer multiple of 16 bytes, there are a variety of ways to fill, two of which are more common:

A. Filling the remaining characters with number of missing bytes on each group whose length is less than 16 bytes.

B. Filling the last byte with the number of missing bytes on each group whose length is less than 16 bytes and fill the remaining characters with random numbers [9]. If the length of plaintext block is 16 bytes, it will add an additional 16-byte plaintext block.

Instead of converting the blocks of plaintext into cipher text at one time, Advanced Encryption Standard takes many rounds of operations to encrypt each block, and the rounds is determined by the length of cipher key.

3. The Low-complexed Strategy of Complex Encryption And Decryption.

This paper has proposed an encryption and decryption strategy based on limited memory environment like mobile phones. And the Advanced Encryption Standard is also suitable for the proposed strategy. Compared with advanced encryption standard, the text will occupy less memory space for encryption and decryption. So it enhances the disorder of cipher text as well as decrease the difficulty and cost of cracking.

The main strategy is to encrypt block plaintext and decrypt block cipher. When the system execute the encryption and decryption task, it will neither occupy a large number of memory nor increase the complexity of cipher text. Complex encryption and decryption means that the requirements of encryption and decryption is complexed. For example, in some time, not all the information need to be encrypted and decrypted, only part of the information need to be encrypted and decrypted. The encryption and decryption of books of ePub [10] is a typical example. In general, only the content of the chapters needs to be encrypted and decrypted in books ePub. And the information of books including the title of the book, author, chapter, press name, cover the message shows plaintext. The benefits are obvious that it can reduce the demand of memory for encryption and decryption of the content, reducing the memory pressure.

3.1 Complex encryption steps

To achieve complex encryption and decryption, the distinction between plaintext and cipher text in the encrypted information should be clear. In decryption, only the cipher text needs to be decrypted, and the plaintext will not be processed. A cipher text starts with an identifier in the front of block cipher named cipher text start identifier indicating the start of encryption and a cipher text ends with an identifier in the end of the block cipher named cipher text end identifier indicating end of encryption It is the most direct and effective way to distinguish plaintext and cipher text.

Therefore, achieving complex encryption requires total four steps. First of all, it will distinguish which parts of information needs to be encrypted and which parts of information should be plaintext; Secondly it will divide all parts which need to be encrypted into small blocks; Thirdly it will encrypt each block independently and add block start identifiers at the beginning and block end identifiers at the end of each encryption part. It will add no encryption identifiers for the text without encryption demand. Finally it will connect each blocks in accordance with the original order. Encryption processes is shown in Figure 1.

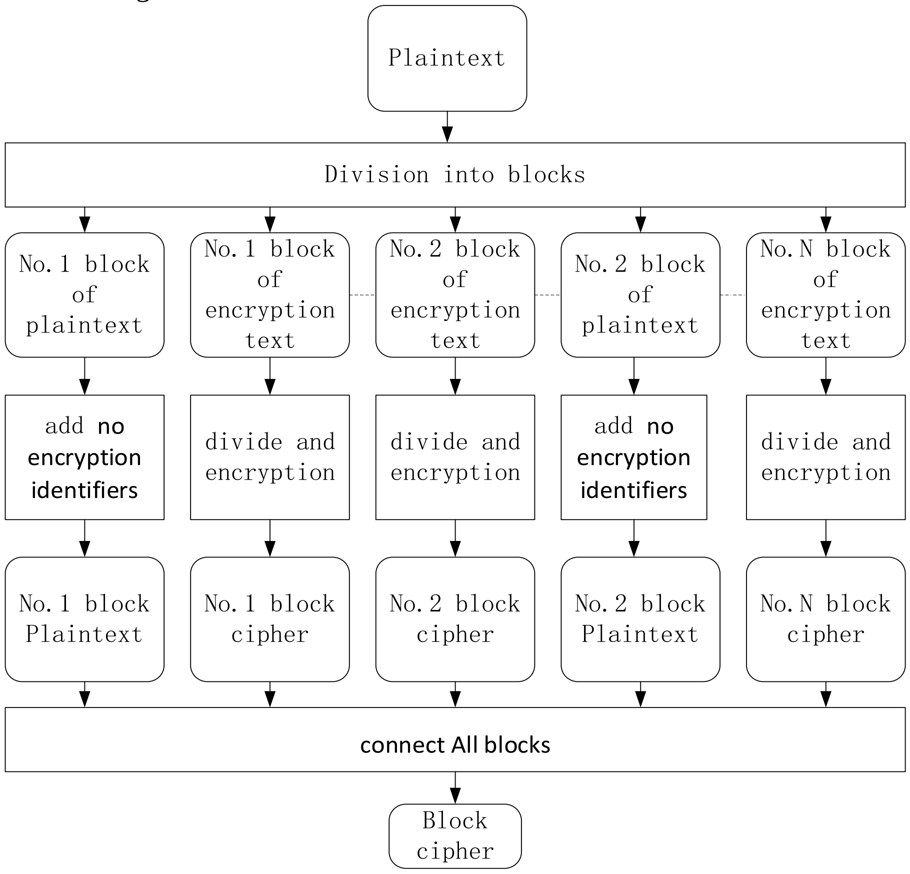

Figure 1: Encryption processes diagram

3.2 The Block Encryption Tool

After the information that needs to be encrypted obtained, it will be sent to the Block Encrypt Tool (BET). The BET executes a total of two tasks, dividing and encrypting. In order to ensure data security, taking into account the fixed-length blocks will increase the risk that blocks are deciphered. Therefore, the length of the blocks is not a fixed value. Only enumerating the AES padding and mode of operation with rightful the key is difficult to find the correct cipher text, greatly enhancing the security of the cipher text. Block cipher working principle shown in Figure 2.

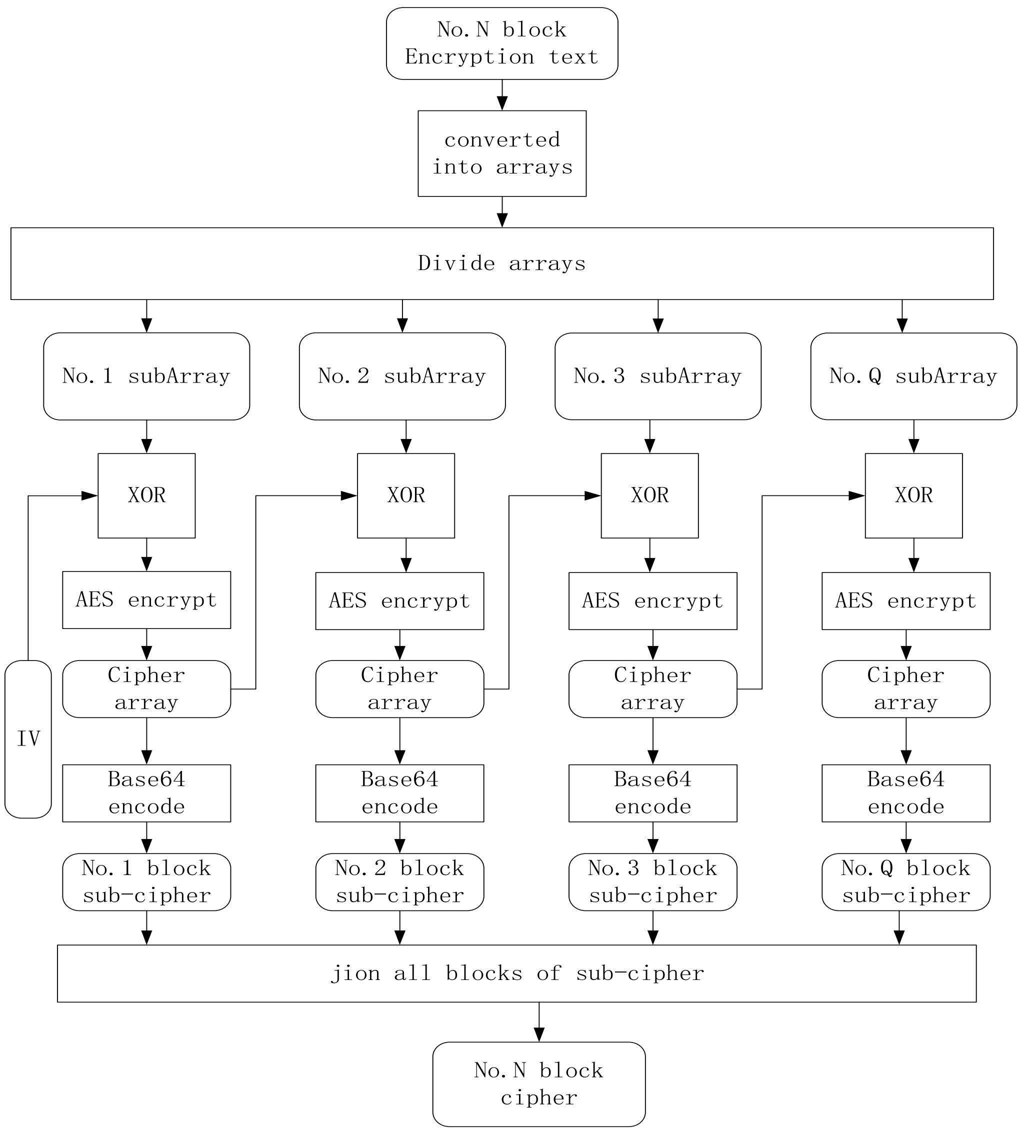

Figure 2: the workflow of the Block Encrypt Tool

3.3 Complex Decryption steps

The steps of decryption and encryption are opposite. Firstly the Cipher Verifier Tool (CVT) obtain blocks cipher from plaintext according to cipher text start identifiers and cipher text end identifier. And then the CVT decrypts the encrypted content by blocks, and removes all identifiers. Finally all contents including the plaintext and content from decryption are concatenated in appropriate order .Then we get the final plaintext. Work flow is shown in Figure 3.

3.3.1 The Cipher Verifier Tool. The Cipher Verifier Tool (CVT) will read a group of the text of the information and judge whether the current group contains cipher text start identifiers and cipher text end identifiers. Due to the complexity of reading the characters into the cipher verifier from memory, the following situations exist:

a) It reads both cipher text start identifiers and the corresponding cipher text end identifiers at the same time;

b) It reads neither cipher text start identifiers nor the corresponding cipher text end identifier at the same time; there are two cases: b1) the current group is a cipher text; b2) the current group is plaintext

c) It reads a cipher text start identifier but does not read the corresponding cipher text end identifier;

d) It only reads a cipher text end identifier but does not read the cipher text start identifier.

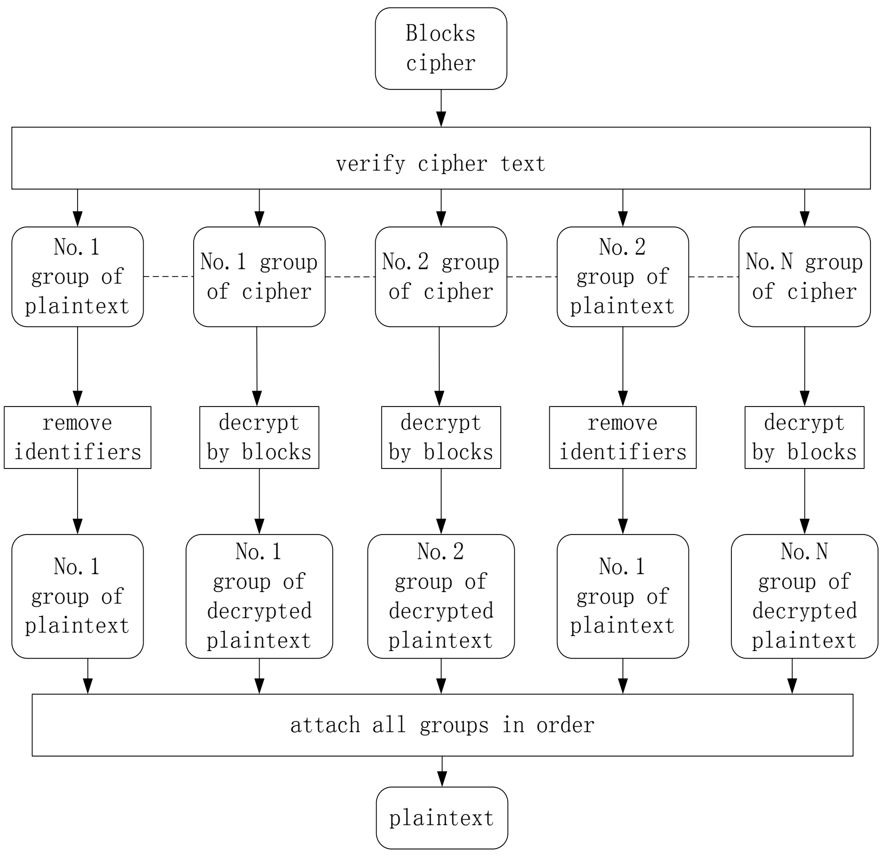

Figure 3 the workflow of cipher Verifier

So, the CVT needs a field named decryption status to indicate the current state of current text, whether in decryption or not. First, the CVT places the decryption status as not in decryption. The textual information mixed by plaintext and cipher text in the encrypted text is read by a decryption buffer in CVT, which can read at most L characters in the cipher verifier tool every time.

If the string read by the decryption buffer only contains a cipher text start identifier, the decryption status is set as in decryption, and the substrings after the cipher text start identifier are the cipher text information to be decrypted The CVT sends the cipher text to the block decryption executor (BDE), and the BDE decrypts the cipher text. And the corresponding plaintext information is returned to the CVT. The decryption status of CVT is set as in decrypted.

If the string read by the decryption buffer only contains the cipher text end identifier, the substring before cipher text end identifier will be sent to the BDE which will return the corresponding plaintext. The decryption status is set as not in decryption.

If the decryption buffer reads the string that contains both a cipher text start identifier and the corresponding cipher text end identifier. If the cipher text start identifier precedes the cipher text end identifier, substring between the two identifiers is the real cipher text. It will be sent to the BDE and the corresponding plaintext is returned.

In some times, the text may contain both short plain text and long cipher text. In this situation, the cipher text ending identifier may precedes the cipher text starting identifier. Then the substring before the cipher text ending identifier and the substring after starting the cipher text identifier are respectively sent to the block decryption executor according to the natural order and corresponding plaintext is returned in order.

If the decryption buffer reads the string contains neither the cipher text start identifier nor the cipher text end identifier. It depends on the decryption status. If the decryption status is in decryption, all content of the buffer will be send to the BDE, which will return the decrypted result. If the decryption status is not in decryption, the CVT considers the string in the decryption buffer to be plaintext, and returns the plaintext directly.

3.3.2 Block Decryption Executor. The Block Decryption Executor (BDE) executes the two kinds of tasks. The first one is to divide the cipher text into groups. Cipher text carries block identifiers including block start identifiers and block end identifiers. Each group is divide into packets according to block identifiers. There will be a variety of situations about each group for the BDE received:

a) The group consists of several complete packets;
b) The group is composed of a number of packets preceding an incomplete packet;
c) The group is composed of an incomplete packet preceding a number of complete packets;
d) The group is composed of an incomplete packet at the beginning plus a few complete packets and an incomplete packet in the end.

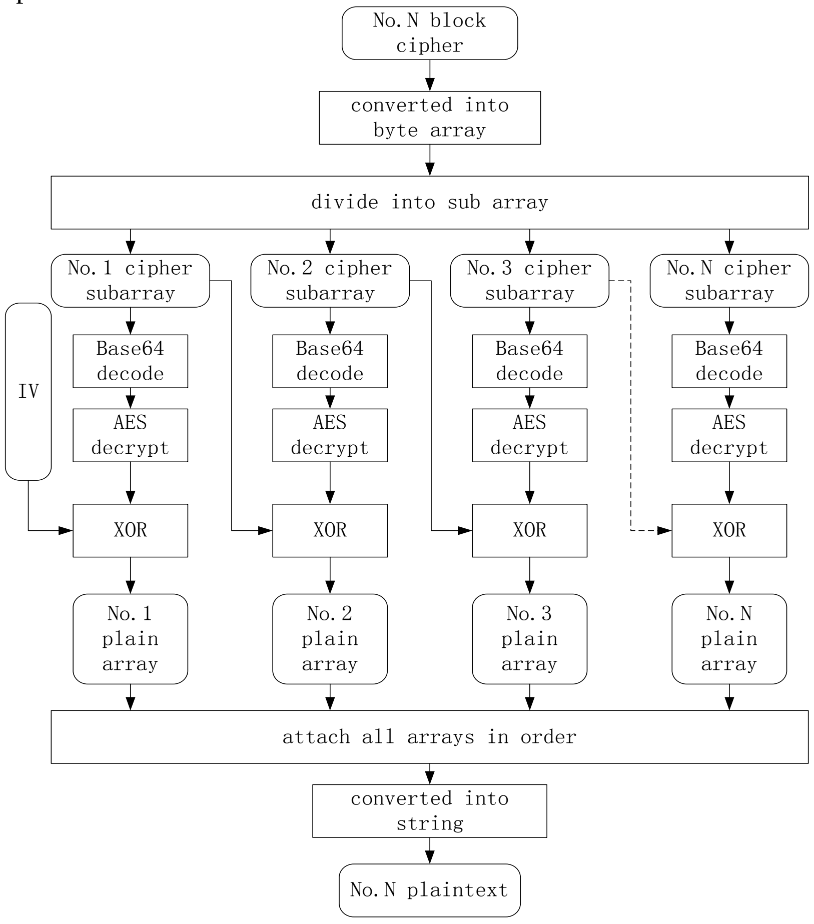

Figure 4 Workflow of Block Decryption Executor

For a), the BDE directly divides the group into packets, and then decrypts the packets according to the order in the cipher text.

For b), firstly, the BDE divides the group into packets, and decrypts all packets in accordance with the order besides the last packet. The last packet is not a complete packet. And only the other missing information is spliced in the next decryption can it be properly decrypted. Then it is recorded in the BDE with another field call last incomplete packet.

For c), the BDE firstly judges whether the field last incomplete packet has an incomplete packet that was not used in the last decryption. If so, splice this incomplete packet in front of the group and then divide it into packets and decrypt them according to a).

For d), the BDE first judges whether the field last incomplete packet has an incomplete packet which is not used in the previous decryption. If any, incomplete packets are concatenated before decrypting the message, then all packets are operated as case b).

The second kind of task is to decrypt the cipher text. At the time of decryption, the first cipher packet needs to be decrypted first, and then it will be done the operator of XOR with the initialization vector to obtain the first plaintext packet. Each packet to be decrypted thereafter needs to be decrypted firstly, then be done the operator of XOR with the previous packet to get the correct plaintext. The workflow of BDE is shown in Figure 4.

4.Test results

In order to validate the encryption and decryption strategy, this paper uses Android test phone to get the following test by running an application (APP):

Test 1: memory occupation of different sizes of files in the same memory environment. Each text in Table 2 is encrypted 10 times in the test device 1 and decrypted 10 times to obtain the data of all the parameters in Table 3.

Test 2: memory occupation of the same files in a different memory environment. No. 1, No. 3, No. 5 documents of Table 2 were encrypted and decrypted in device 1, device 2 device 3 10 times to obtain the data of No.1 and No.4 parameters in Table 3.

In the test, this paper sets the default values of the key parameters as follows. In the encryption process, length of each block is different, is limited to [512, 640]. The length of the block unit is 128 bytes. When decrypted, 1024 characters are read at a time. The encryption and decryption tasks will run in a separate process. The memory occupation refers to the memory of this independent process [11].

Table 1 shows the specific configuration of the Android devices for the test. The files used in the test are shown in Table 2. And the parameters are shown in Table 3.

Table 1 Configuration information about test devices.

	device 1	device 2	device 3
Model Number	ZTE BA510	MI2	Google Nexus S
Android Version	5.1	5.0	4.1
RAM	1 GB	2 GB	512 MB
ROM	8 GB	16 GB	16GB
Rom available	1.82GB	7.9 GB	10 GB

Table 2 Information about test documents

No.	Name	Size(in Mb)	format
1	Document 1	1.44	txt
2	Document 2	5.68	txt
3	Document 3	9.88	txt
4	Document 4	59.31	txt
5	Document 5	273.22	txt

Table 3 Information about test parameters

Name	Meaning
Parameter 1	The average value of 10 encryptions' maximum memory[12]
Parameter 2	Maximum value of 10 encryptions' maximum memory
Parameter 3	The average value of 10 decryptions' maximum memory
Parameter 4	Maximum value of 10 decryptions' maximum memory

Figure 5 and Figure 6 show the results of test1.

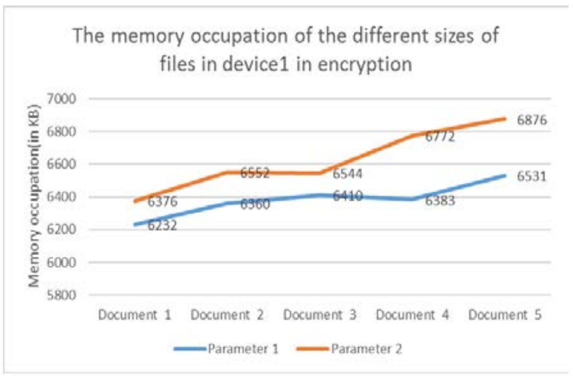

Figure 5 the memory occupation of the different sizes of files in device1 in encryption

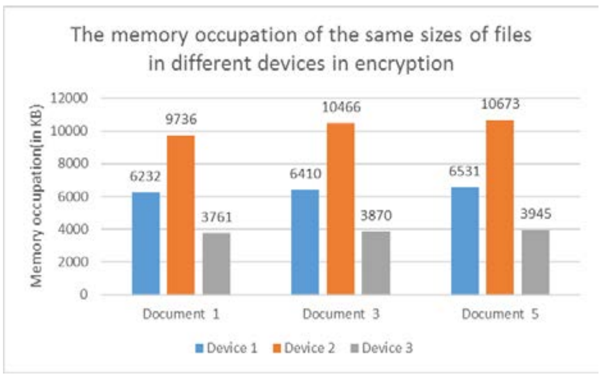

Figure 6 the memory occupation of the different sizes of files in device1 in decryption

As is seen from Figure. 5, with the same memory size, the maximum memory in encryption does not increase significantly with the increase of files' size. Although the size of files increases rapidly, but the maximum memory increases no more than 1 Mb.

As is seen from Figure. 6, with the same size of memory, the maximum of memory in decryption increases slowly. It increases no more than 500kb while the size of files increases nearly 200 times.

Figure. 7 and Figure. 8 show the results of test 2.

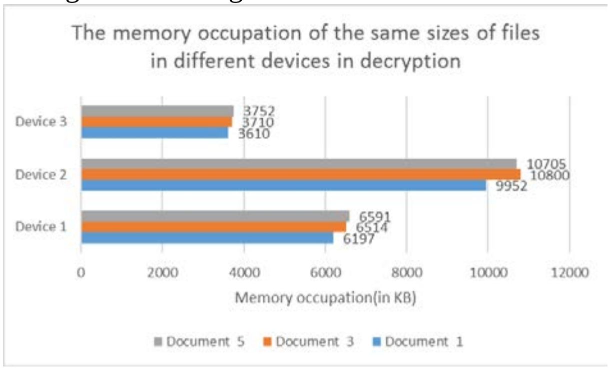

Figure 7 the memory occupation of the same sizes of files in different devices in encryption.

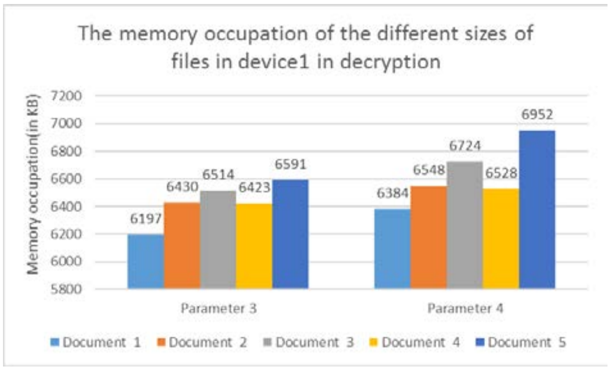

Figure 8 the memory occupation of the same sizes of files in different devices in decryption.

As can be seen from Figure 7, in the case of the same sizes of files and devices with different size of memory, the maximum memory in encryption may change. It has nothing to do with the file size, but is related to the memory size of device.

As can be seen from Figure 8, in the case of the same sizes of files and the device with different size of memory, the maximum memory in decryption varies. However, the magnitude of the change has nothing to do with the file size, but is related to the memory size of the device.

Two conclusions can be drawn from the test result:

1. In the same memory size environment, with the increase of text content, the maximum memory in the encryption or decryption can be maintained in a low range, and the change is not significant.

2. With the same text, as the environment's memory size increases, the maximum amount of memory in the encryption and decryption can increase. However, the magnitude of this change has nothing to do with the size of the file, and is related to the system's memory and memory allocation strategy. The system allocates a suitable amount of running memory to an application according to the system's configuration information. With device memory increasing, the system allocates more operating memory to the application [13, 14].

Through the above tests, it is proved that the low-complex strategy of complex encryption and decryption can be operated normally in device of low memory.

From the test results, the program can complete complex encryption and decryption of text under low memory, and it occupies small size of memory than, and works fast. It can execute encryption and decryption task in a variety size of memory environments, and it has strong compatibility.

5. Conclusions

In this paper, a block division encryption and decryption strategy is proposed to reduce the complexity of the encryption and decryption calculation. It is used for the low memory environment such as cell phone. And the paper uses non-uniform division to guarantee the safety of the cipher text. And the test results shows that the proposed strategy spend less time in encryption and decryption.

Furthermore, In the case of large memory, you can increase the length of blocks to increase the speed of encryption. You can speed up decryption by increasing the length of group. At the same time, the strategy can effectively control the encrypted cipher text repetition and increase the disorder of cipher text. In addition, it increases uncertainty of length of blocks and enhances the security of the cipher text.

6. Acknowledgments

This research is supported in part by the National Key R&D Program of China (No.2017YFB0503700).

7. References

[1] C. Li, J. Bao and H. Wang, "Optimizing low memory killers for mobile devices using reinforcement learning," 2017 13th International Wireless Communications and Mobile Computing Conference (IWCMC), Valencia, 2017, pp. 2169-2174.
[2] C. Wanpeng and B. Wei, "Adaptive and dynamic mobile phone data encryption method," in China Communications, vol. 11, no. 1, pp. 103-109, Jan. 2014.
[3] C. Biswas, U. Das Gupta and M. M. Haque, "A hierarchical key derivative symmetric key algorithm using digital logic," 2017 International Conference on Electrical, Computer and Communication Engineering (ECCE), Cox's Bazar, 2017, pp. 604-609.
[4] K. Huang, R. Tso and Y. C. Chen, "One-time-commutative public key encryption," 2017 Computing Conference, London, 2017, pp. 814-818.
[5] P. M. Chabukswar, M. Kumar and P. Balaramudu, "An efficient implementation of enhanced key generation technique in data encryption standard (DES)algorithm using VHDL," 2017 International Conference on Computing Methodologies and Communication (ICCMC), Erode, 2017, pp. 917-921.
[6] M. Panda and A. Nag, "Plain Text Encryption Using AES, DES and SALSA20 by Java Based Bouncy Castle API on Windows and Linux," 2015 Second International Conference on Advances in Computing and Communication Engineering, Dehradun, 2015, pp. 541-548
[7] R. J. Nayaka and R. C. Biradar, "Key based S-box selection and key expansion algorithm for substitution-permutation network cryptography," 2013 Annual International Conference on Emerging Research Areas and 2013 International Conference on Microelectronics, Communications and Renewable Energy, Kanjirapally, 2013, pp. 1-6.
[8] C. U. Bhaskar and C. Rupa, "An advanced symmetric block cipher based on chaotic systems," 2017 Innovations in Power and Advanced Computing Technologies (i-PACT), Vellore, 2017, pp. 1-4.
[9] T. Miyano and K. Cho, "Chaos-based one-time pad cryptography," 2016 International Symposium on Information Theory and Its Applications (ISITA), Monterey, CA, 2016, pp. 156-160.
[10] M. Ebner, C. Prettenthaler and M. Hamada, "Cloud-Based Service for eBooks Using EPUB under the Aspect of Learning Analytics," 2014 IEEE 8th International Symposium on Embedded Multicore/Manycore SoCs, Aizu-Wakamatsu, 2014, pp. 116-122.
[11] L. Yuqing, X. Hao and L. Xiaohui, "The Research of Performance Test Method for Linux Process Scheduling," 2012 Fourth International Symposium on Information Science and Engineering, Shanghai, 2012, pp. 216-219.

[12] Arkadiusz Socała,Michael Cohen. Automatic profile generation for live Linux Memory analysis[J]. Digital Investigation,2016,16.
[13] K. Vimal and A. Trivedi, "A memory management scheme for enhancing performance of applications on Android," 2015 IEEE Recent Advances in Intelligent Computational Systems (RAICS), Trivandrum, 2015, pp. 162-166.
[14] Z. Zhu et al., "A Thread Behavior-Based Memory Management Framework on Multi-core Smartphone," 2014 19th International Conference on Engineering of Complex Computer Systems, Tianjin, 2014, pp. 91-97.