Mr.Shish Ahmad, Dr. Rizwan Beg, Ms. Raziqa Masood / International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622 www.ijera.com
Vol. 3, Issue 3, May-Jun 2013, pp.995-999

Contribution of Transparent cryptography in prevention of information leakage

(protecting digital content in transparent cryptography)

Mr.Shish Ahmad, Dr. Rizwan Beg, Ms. Raziqa Masood, Mr Shahid Hussain

*Computer Science and engineering Integral University Lucknow, India.
**Computer Science and engineering Integral University Lucknow, India.
***Computer Science and engineering Integral University Lucknow, India.

Abstract

The aim of this paper is to analyze how transparent cryptography protected digital content of mobile in an effective way. In this chapter we will look for possibilities to use transparent cryptography for AES in a secure way. We discuss an application of transparent cryptography in which we split the set of white-box tables into a dynamic part and a static part . The result is that whenever a key needs to be updated, no longer the whole set of tables needs to be updated. In §6.3 different possibilities for using external encodings are described. We discuss what the best possibility .we discuss the problem of the storage space and we look for possibilities to use less tables to save storage. Finally, we look for possibilities to use transparent cryptography for OMA-DRM.

1. INTRODUCTION :

In this paper we looked for possibilities to use transparent cryptography for AES in a secure way. We recommended to split the set of white-box tables into a static part and a dynamic part. In this way each client has a unique set of static tables which can only be used in combination with a unique set of dynamic tables. The security also increases by sending a different ciphertext to each client.

2. We suggested different possibilities for applying transparent cryptography for OMA-DRM. Because of the total size of the tables and the slowdown we recommended using white-box cryptography only for keys which are fixed over a longer period of time. For example, white- box cryptography can be used to update the private key. White-box cryptography can also be used to store keys on the client's device.

3. Overall, we can say that transparent cryptography ensures that the keys are no longer visible. Nevertheless, we are still able to publish the decrypted content. For example we can put a lot of effort in hiding a key which can be used to decrypt an encrypted ringtone, but as soon as the ringtone is decrypted on our mobile phone we could tap it and distribute it on the internet. The same can be said in the case of broadcasting an encrypted movie or a soccer match via a satellite. However, for instance in case of a soccer match, people want to see it live. In this situation people would be more interested in obtaining the key. If they have the key, they could tap the encrypted soccer match and use the key for decryption and watch the soccer match live. In this case it is more valuable to have the key, which implies that it is very

4. important to protect the key securely by for example transparent cryptography.

Breaking the tables into dynamic and static Tables

We want to be able to update the key which is hidden in the white-box tables. This can be done by splitting the tables. The part of the tables which is dependent on the key is sent to the client and the other part of the tables which is not dependent on the key is stored on the client's device. When we want to update the key, only part of the tables needs to be sent to the client. Therefore, less data needs to be transmitted. The tables that are sent by the server to the client can be updated and are called the dynamic tables. The tables that are stored on a clients's device cannot be updated and are called the static tables. If an attacker taps the ciphertext plus part of the tables, he is not able to decrypt the ciphertext, because he also needs the other tables.The following needs to be considered:

• It is important that each client receives different static tables to ensure that each client uses a unique combination of static and dynamic tables. If this is not ensured, then someone could tap the dynamic tables which were sent to another client and use these dynamic tables incombination with his own static tables to decrypt the content.

• It is important that the static tables cannot be copied. Otherwise a client could publish his static tables and his dynamic tables which together could be used for decrypting content. This can be done by locking the static tables on the device (nodelocking).

However, the question remains which tables need

to be transmitted and which tables can be stored. There are five types of tables: type Ia, II, III, IV, and Ib. The tables that are dependent on the key are the type II and the type Ib tables. We want to be able to update this key, therefore these tables cannot be fixed on the client's device. The least amount of data a server needs to send are the tables which are dependent on the keys and those are the type II and type Ib tables.

There are several possibilities for partitioning the set of tables into a set of dynamic tables and a set of static tables:

1. Dynamic tables: II, Ib (208 KB) Static tables: Ia, III, IV(544 KB)
2. Dynamic tables: Ia, II, Ib (272 KB) Static tables: III, IV (480 KB)
3. Dynamic tables: II, III, Ib (352 KB) Static tables: Ia, IV (400 KB)
4. Dynamic tables: II, IV, Ib (544 KB) Static tables: Ia, III (208 KB)
5. Dynamic tables: Ia, II, III, Ib (416 KB) Static tables: IV (336 KB)
6. Dynamic tables: II, III, IV, Ib (688 KB) Static tables: Ia (64 KB)
7. Dynamic tables: Ia, II, IV, Ib (608 KB) Static tables: III (144 KB)
8. Dynamic tables: Ia, II, III, IV, Ib (752 KB) Static tables: -

Partition 8 is the original situation in which all the tables are sent to the client. If an attacker has access to all the tables, the attack can be executed. Therefore, it is not recommended to transmit all the tables over the line.

The server wants to send the least amount of data. Therefore, the server only wants to send tables which it wants to update, like the tables which are dependent on the key or the tables which represent the external encodings. Two partitions remain:

Dynamic tables: II, Ib (208 KB) Static tables: Ia, ,III, IV (544 KB)
(see Figure 15)

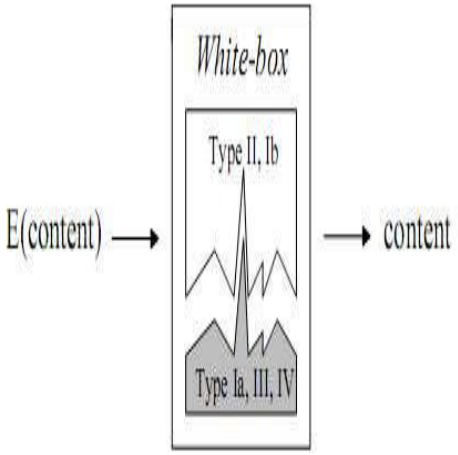

Figure 15 one of the four mapping for a single AES round

Dynamic tables: Ia, II, Ib (272 KB) Static tables: III, IV (480 KB)(see Figure 16)

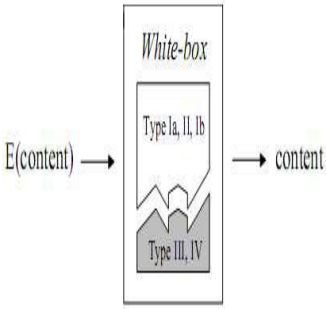

Figure 16 one of the four mappings for a single AES round

The static keys can be seen as personalization keys, which are unique for each client. The choice between the two possible partitions depends on the choice for the external encodings.

Bijection Encodings :

	Plain text	Cipher text	Avalanche effect
Round 1	11110000 01111123 43111123 11111113	79 2c 44 24 01 82 dd 7f 2d f9 87 e7 78 b7 ee 30	.4375 (44)
Round 2	79 f8 cc 24 01 82 dd 7f 2d f9 87 e7 78 b7 ee 30	4a a9 16 11 e2 8a 9f 67 35 30 1f 80 16 c5 b7 cd	.5153 (51)
Round 3	4a a9 16 11 e2 8a 9f 67 35 30 1f 80 16 c5 b7 cd	C6 a1 3c 37 65 8b 29 20 45 4a 3c 36 472c 2b b7	.4453(44)
Round 10	21 a1 3c 37 65 8b 29 20 45 4a 3c 36 472c 2b b7	0d 38 274c 2a 1b 34 27 87 68 3c 2b 1d 4a 34 23	.4215(42)
Applying C= F. AESe. G^{-1} (P)	0d 38 274c 2a 1b 34 27 87 68 3c 2b 1d 4a 34 23	27 b1 3c 37 65 8b 29 20 45 4a 3c 36 472c 2c b8	.4567(46)

First case

- In the first case the server sends the ciphertext C = F · AESe · G^{-1}(P) plus the white-box tables G · AESd · F−1 to the client. The client can use the dynamic tables to decrypt the ciphertext to obtain the plaintext P.
- Let F and G is the bijection of 128 bit data so this cipher text will give the data that will vary as make the changes in F & G ,which make the cipher text complicated,time taking and strongly secure.
- Let us take an example of this cipher text :

F is the bijection of plain text i.e of 128 bit and G is the bijection of key data i.e also of 128 bit data here,

P=ffeeddccbbaa99887766554433221100

- Key: 000102030405060708090a0b0c0d0e0f
- G^{-1}(P)=00112233445566778899aabbccddeeff

After performing 10 rounds of AES :

- R0 (Key = 000102030405060708090a0b0c0d0e0f)

P= 00102030405060708090a0b0c0d0e0f0

- R1 (Key = d6aa74fdd2af72fadaa678f1d6ab76fe)

P = 89d810e8855ace682d1843d8cb128fe4

R10(Key = 13111d7fe3944a17f307a78b4d2b30c5)

- P = 69c4e0d86a7b0430d8cdb78070b4c55a)

C=Ciphertext: 69c4e0d86a7b0430d8cdb78070b4c55a

C= F.AESe.G^{-1} (P)+ G.AESd.F^{-1} (look up tables)

- Similarly , we can get different cipher text by varying the value of F and G.
- After doing XOR with this data with f , it'll give the huge amount of data and that data can vary,which will make the cipher text complicated.
- So this case gets the maximum complexity as compare to other cases.

- **Second case**
- In the second case the server sends the ciphertext C = F ·AESe(P) plus the white-box tables G · AESd · F−1 to the client. The client can use the white-box tables to decrypt the ciphertext .
- The advantage of this method is that F can be varied. The disadvantage is the assumption that the renderer is completely secure.

Third case

- In the third method the server sends the ciphertext

C = AESe · G^{-1}(P) plus the white-box tables G·AESd. F−1 to the client. The client can use the white-box tables to decrypt the ciphertext to obtain the plaintext P.

- The advantage of this method is that the

server has to send less data. However, it is not a good idea to use the local encryption because it weakens the security.

- Thus, if local encryption is used, then it is not safe to assume that F can be kept secret. If the client knows F, then also the mixing bijections of the type Ia tables and the type II tables can be determined. This will probably make it easier to extract the keys from the tables. More research is needed to determine if knowledge of the bijections will make it easier to extract the keys from the tables.

- **Fourth case**
- In the fourth method the server sends the ciphertext C = AESe(P) plus the white-box tables G · AESd · F−1 to the client. The client encrypts the ciphertext with a stored F. The new ciphertext serves as input to the white-box tables.
- The advantage of this method is that the server has to send less data. However, it is not a good idea to use the local encryption because it weakens the security (see method three). Another advantage is that the server sends the same ciphertext C = AESe(P) to each client. This is also a disadvantage because once an attacker has found the AES key, he could publish it and everybody could use that key to decrypt the ciphertext. In the other methods the ciphertext is personalized for each user which prevents this from happening.
-
- **Comparison of all cases**

So on the basis of avalanche effect we can see over here the avalanche effect of case 1 and 2 is much better then othe cases.

- Method one and two are the best methods and three and four are weaker variants which should not be used. The choice between the first and the second method depends on the assumption of security. If a secure renderer can exist the second method is better because P is never available to the client. However, the idea of a secure renderer contradicts the idea of a white- box attack model which we use throughout this document. Thus, using the first method is recommended.

Conclusion : We discussed two drawbacks of transparent cryptography. The first drawback is that whenever the key needs to be updated, the whole set of white-box tables needs to be updated too. We solved this problem by splitting the set of tables into a dynamic part and a static part. Each client has a unique set of static tables which can only be used in combination with a unique set of dynamic tables which are sent to him. Because only the dynamic tables are dependent on the key, the server only has to update the dynamic tables when it wants to update the key. This is also a way to obtain software diversity, because each client needs a unique combination of static and dynamic tables for decryption. A second drawback is that the whole white-box implementation can be used as a key. If an attacker knows to the complete white-box implementation, he can use the white-box tables to decrypt the content. Therefore, it is important that the static tables cannot be copied. This can be done by locking the static tables on the hardware (nodelocking). More research is needed on the possibility of locking the static tables on hardware.

Reference:

[1] D. Aucsmith, Tamper Resistant Software and Implementation, Proc. 1st International Information Hiding Workshop (IHW), Cambridge, U.K. 1996, Springer LNCS 1174, pp. 317-333 (1997).

[2] B. Barak, Can We Obfusacate Programs?, http://www.math.ias.edu/ boaz/Papers/obf informal.html.

[3] B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. Vadhan and K. Yang, On the (Im)possibility of Obfuscating Programs,pp 1-18, Advances in Cryptology - Crypto 2001, Springer LNCS 2139 (2001)

[4] O. Billet, H. Gilbert, C. Ech-Chatbi, Cryptanalysis of a White-box AES Implementation, SAC 2004.

[5] H. Chang, M. Atallah, Protecting Software Code by Guards, Proc. 1st ACM Workshop on Digital Management (DRM 2001), Springer LNCS 2320, pp.160-175 (2002).

[6] Y. Chen, R. Venkatesan, M. Cary, R. Pang, S. Sinha, M. Jacubowski, Oblivious Hashing: A stealthy Software Integrity Verification Primitive, Proc. 5st Information Hiding Workshop (IHW), Netherlands (October 2002), Springer LNCS 2578, pp.400-414.

[7] S. Chow, P. Eisen, H. Johnson, P.C. van Oorschot, AWhite-Box DES implementation for DRM Applications, pp. 1-15, Proceedings of DRM 2002 - 2nd ACM Workshop on Digital Rights Management (DRM 2002), Springer LNCS 2696 (2003).

[8] S. Chow, P. Eisen, H. Johnson, P.C. van Oorschot, White-Box Cryptography and an AES implementation, pp. 250-270, Proceedings of the NinthWorkshop on Selected Areas in Cryptography (SAC 2002), Springer LNCS 2595 (2003).

[9] F. Cohen, Operating System Protection Through Program Evolution, Computers and Security 12(6), 1 Oct. 1993, pp. 565-584.

[10] C. Collberg, C. Thomborson, and D. Low. A Taxonomy of Obfusacting Transformations. Technical Report 148, Department of Computer Science, University of Auckland, July 1997. [11] J. Daemen, V. Rijmen, AES Proposal: Rijndael, http://csrc.nist.gov/encryption/aes/rijndael / Rijndael.pdf, 1999.

[12] S. Forest, A. Somayaji, D. H. Ackley, Building Diverse Computer Systems, pp.67-72, Proc. 6th Workshop on Hot Topics in Operating Systems, IEEE Computer Society Press, 1997.

[13] P. Gorissen, J. Trescher, Key Distribution in Unsafe Evironments, Philips Research Laboratories Eindhoven, to be published.

[14] B. Horne, L. Matheson, C. Sheehan, R. Tarjan, Dynamic Self-Chacking Techniques for Improved Tamper Resistance, Proc, 1st ACM Workshop on Digital Rights Management (DRM2001), Springer LNCS 2320, pp.141-159 (2002).

[15] National Institute of Standards and Technology (NIST). AES Key Wrap Specification, November 2001. Available at csrc.nist.gov/encryption/kms/key-wrap.pdf

[16] P.C. van Oorschot, Revisiting Software Protection, In Proc. of 6th International Information Security Conference (ISC 2003), pages 1-13. Springer-Verlag LNCS 2851, 2003. Bristol, UK, October 2003.

[17] Open Mobile Alliance, DRM Specification V2.0, Open Mobile Alliance Ltd, 2004, La Jolla (CA), USA.

[18] C. Wang, A security Architecture for Sirvivability Mechanisms, Ph. D. thesis, University of Virginia (Oct. 2000).