International Journal of Electrical and Computer Engineering (IJECE)
Vol. 5, No. 4, August 2015, pp. 759~764
ISSN: 2088-8708

Security against Timing Analysis Attack

Deevi Radha Rani[1], S. Venkateswarlu[2]
[1]Department of Computer Science and Engineering, VFSTR University, Vadlamudi, India
[2]Departement of Computer Science and Engineering, KL University, Vaddeswaram, India

Article Info

Article history:

Received Mar 18, 2015
Revised Apr 28, 2015
Accepted May 20, 2015

Keyword:

Data Encryption Standard
Hamming Weight
Side Channel Attack
Timing Attack

ABSTRACT

Timing attack is the type of side-channel attack involves the time taken to complete critical operations. Securing crypto processor from timing attack is critical issue. This paper implements the Bernstein's Timing Attack and timing attack based on hamming weight. The countermeasures of Bernstein's Timing attack are implemented in our experimental test bed and their performance is compared. This paper also proposes the key recovery method based on timing attack using hamming weight of the key.

Corresponding Author:

Deevi Radha Rani,
Departement of Computer Science and Engineering,
VFSTR University,
Vadlamudi, Guntur Dt, AP, India.
Email: dradharani29@gmail.com

1. INTRODUCTION

The use of computers and communication systems increases the need for securing the information kept in devices or sent between them. In real world all the sensitive information is controlled and distributed via computer networks. Cryptographic algorithms protect the information by protecting cryptographic keys but there are still many issues for systems in which the physical implementations can be accessed. Today, cryptographic algorithms are increasingly applied or embedded in devices such as smart cards and cell phones. Attackers can retrieve key using the timing measurements when a cryptographic algorithm is being implemented in any of the emebedded systems. This paper proposes the countermeasures for Bernstein's Timing Attack and proposes the efficient countermeasure that can be implemented in any embedded systems against timing analysis attack. This paper also gives the timing analysis attack using hamming weight and proposes the key recovery algorithm.

2. BACKGROUND AND RELEVANT TOPICS

This section covers the necessary topics required to develop and explain the proposed countermeasures and algorithm.

2.1. Implementation Attacks

Implementation attacks are a type of cryptanalysis attacks that do not target cryptographic algorithms and protocols directly. These attacks rather aim at implementations of cryptographic systems (e.g. smart cards) to gain knowledge about secret information. These attacks can be Active attacks, which target the physical security of the device. Another class of attacks acts in a passive way, just by observing the inherent leakage of the cryptographic device.

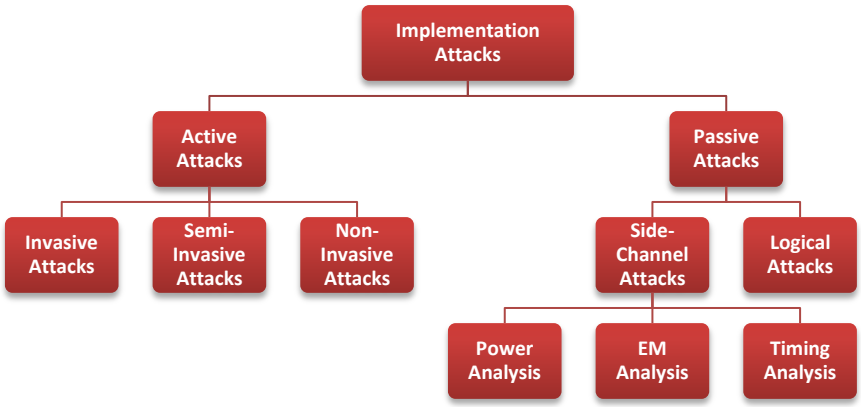

Figure 1. Classification of Implementation Attacks.

Passive Attacks are even more dangerous as they do not leave any damage to the cryptographic device that can be recognized later on. Passive Attacks just use the cryptographic device in its intended environment and can obtain cryptographic keys by leaked information. Side Channel Attacks can retrieve the secret information (key) inside those devices by collecting and analyzing the leakage information from side channels. Classification of implementation attacks are shown in Figure 1.

2.2. Side Channel Attacks

Kocher introduced the use of side channels to break a cryptosystem [1], [2]. Attacks involving passive observation of external characteristics of a device are termed *eavesdropping attacks*, also sometimes called *side-channel attacks*. When a cryptographic device perform encryption or decryption, secret parameters correlated to the intermediate data being processed can be leaked via operating times, power dissipation, or electromagnetic radiation as side channel information. Cryptanalysis based on side channel information is called side-channel attack. Figure 2 shows the scenario of side channel attack.

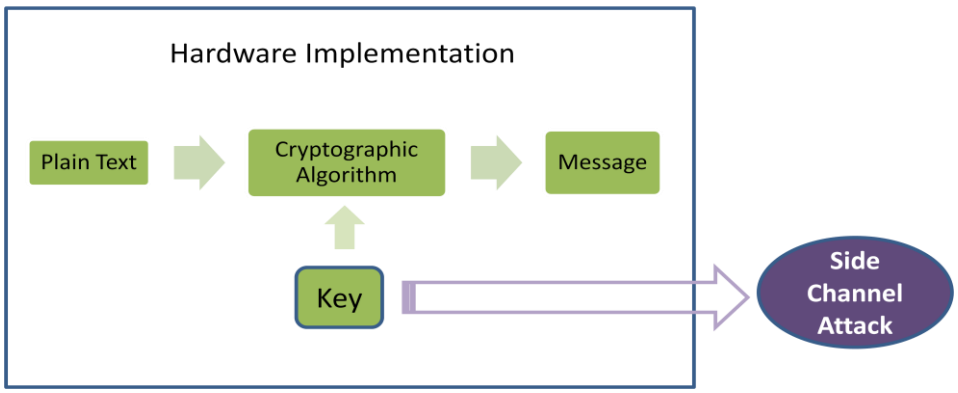

Figure 2. Scenario for Side Channel Attack.

Power analysis attacks [2] exploit the dependence between the instantaneous power consumption of a cryptographic device and the data it processes and/or the operation it performs. The overall power consumption of a cryptographic device can be divided into a static and dynamic part. Since the dynamic power consumption is connected directly with the processed data, it is a potential target to detect the dependency between these two parameters. For that reason, power traces can be used to obtain secret information. There are mainly two attacks using this approach, the simple power analysis and the differential power analysis. In a *simple power-analysis*, the attacker uses detailed knowledge of the device to identify which instructions are being executed based on their power signatures. In a *differential power analysis*, the

attacker uses a hypothetical model of the device, and refines this model with statistical analysis of the power usage of the device as shown in the Figure 3.

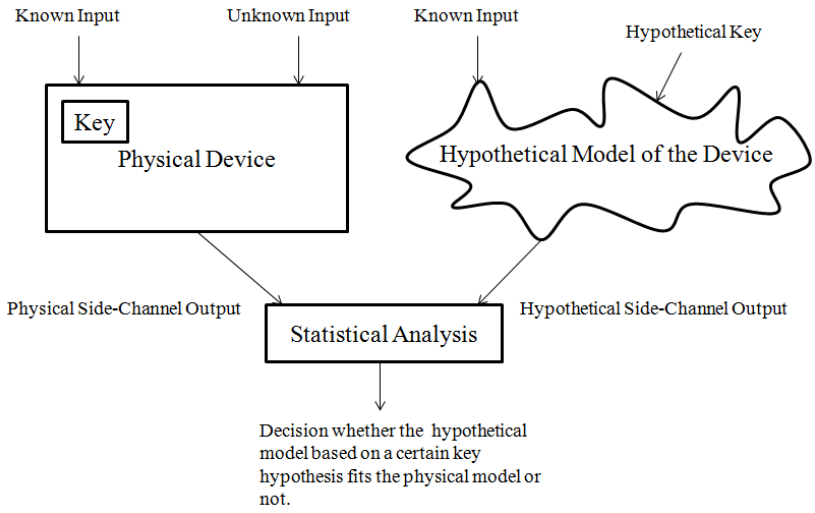

Figure 3. Differential Power Analysis Attack.

Electromagnetic analysis [3] exploits information that leaks through the electromagnetic field that is produced by a device. EM emanation can also exploit local information and, although more noisy, the measurements can be performed from a distance. There are 2 types of emanations: intentional and unintentional. The first type results from direct current flows. The second type is caused by various couplings, modulations etc.

Timing attack is the type of side-channel attack involves the time taken to complete critical operations. Kocher [1] provides a detailed attack strategy for timing crypto-analysis of several commonly used algorithms. He notes that by measuring the time taken to perform private key operations, attackers can recover the input to those operations, thereby determining the private key. Figure 4 present the timing attack principle.

Implementations of cryptographic algorithms often perform computations in non-constant time, due to performance optimizations. If such operations involve secret parameters, these timing variations can leak some information and, provided enough knowledge of the implementation is at hand, a careful statistical analysis could even lead to the total recovery of these secret parameters.

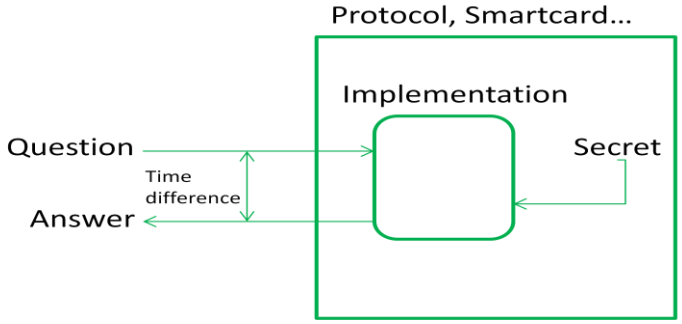

Figure 4. The Timing Attack Principle.

2.3. Advanced Encryption Standard

Advanced Encryption Standard (AES) was announced by Vincent Rijmen and Joan Daemen under FIPS 197 by the NIST [4]. Today AES is widely deployed in both software and hardware and is expected to be the world's predominant block cipher over the coming years. AES is an iterated block cipher, which uses a

fixed block size of 128 bits and a key which is 128, 192 or 256 bits in length. Different transformations operate on the intermediate results, called states. After an initial round key addition, the state array is transformed by implementing a round function10, 12, or 14 times depending on the key length. Each round except the last consists of four stages: SubBytes, ShiftRows, MixColumns and AddRoundKey. Two of these stages involve transformations over Galois Field (GF - 2^8). Generally in software implementations, the multiplicative inverse over GF (2^8) is pre-computed and stored in memory in a table named SBOX. In order to speed up execution of the cipher, software implementations may further combine the SubBytes and ShiftRows with MixColumns, transforming them into a sequence of table lookups. These tables store pre-computed values avoiding time consuming computations. AES algorithm of key length 128/192/256 was well developed in FPGA [5] and throughput and area comparison is done in hardware implementation [6]. During the AES selection process, it was believed that timing attacks were only applicable to software with a data dependent execution path (i.e., branch statements, data dependent shifts, etc.). In the final evaluation of AES candidates, NIST stated that table lookup operations are not vulnerable to timing attacks' and declared Rijndael as capable of averting side-channel attacks. Despite the previous optimistic claims by the NIST, recent research has proven some implementations of AES to be vulnerable to several forms of side channel attacks [7].

2.4. Bernstein's Timing Attack

Two servers are used to implement Bernstein's Timing Attack. The progression of implementation: Clients send packets to servers for encryption, server add time stamp to the packet, encrypts using server's key, another time stamp is added, ciphertext and the time for encrypting the packets are recorded. The original packet and the two time stamps is padded with ciphertext is sent back to the client. The number of cycles that have been taken by the encryption process is calculated using the two time stamp values. Only the packets that have consumed more than 10,000 cycles are considered for the attack to reduce the effect of noise. For each plaintext byte, the average number of cycles, deviation and the estimated deviation for the encryption is calculated. After collecting sufficient amount of timing data for both the known key and the unknown key, a set of key possibilities for each key byte is identified by comparing the two sets of timing data. Finally a packet having all zeros is encrypted with the different key combinations from the set of identified key possibilities. By comparing the resulting cipher text and the ciphertext received of the server, the key combination that would encrypt the zeros in the same way as done by the server is identified as the secret key.

3. RESULTS AND ANALYSIS

3.1. Proposed Countermeasures for Bernstein's Attack

Countermeasures for Bernstein's attack [8] would be Eliminating T tables, Masking timing data from the cache, using smaller tables for calculations, adding random delay in execution of algorithm, placing lookup tables in registerfile, performing encryption using hardware and OS support for partitioning locking and disabling cache. These countermeasures are implemented in our experimental test bed [9] and the performance of the countermeasures is evaluated. Adding random delay would evolve as best countermeasure to Bernstein's timing attack. Figure 5 shows the performance of countermeasures.

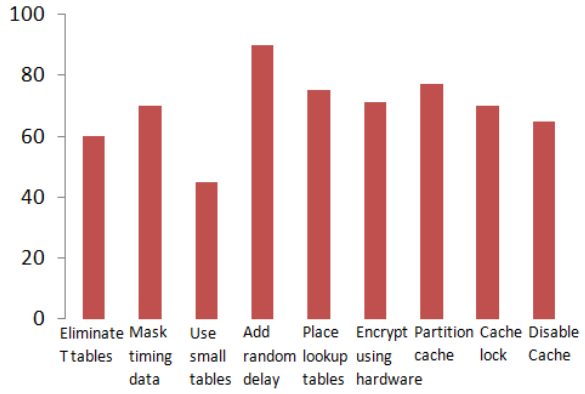

Figure 5. Performance of Countermeasures

3.2. Proposed Key Recovery Algorithm for Timing Attack Based on Hamming Weight

To implement timing attack, a large number of timing measurements is required. In this method we find that timing attack which could reveal hamming weight of the key. The estimation of the hamming weight of key can be achieved with two conditions i) accurate timing measurements can be obtained ii) the time variation by keys with one more or one less set bit is large compared to the variations in the encryption and key schedule generation time produced by different keys with identical hamming weight. The estimation of hamming weight of the key can be done if timing measurements of several encryptions of the same plaintext is obtained. Random plaintext is given as a input. Same plaintext is encrypted for 1000 times using 64 bit key and stores all ciphertext with the corresponding time taken to encrypt. The method collects the plaintext, ciphertexts with its timing measurements. The input message is not fixed, we choose message randomly at start of each encryption. We build a table of average encryption time versus hamming weight of the key. It allows an attacker to determine the hamming weight of the key. Using all generated traces and its timing measurements average timing measurements are calculated with hamming weight of the key. Our implementation is more efficient in revealing the cryptographic key than compared to brute force key search. The pseudocode described below in Figure 6 shows the key recovery method based on timing attack using hamming weight of the key.

```
Input:
M: set of 64-bit plaintexts,
C: set of 64-bit ciphertexts,
t is time it takes AES to generated ciphertext C from Message M
Pseudocode:
for i=0 to 64
Let l be such that |{ j : |Tj-t| < |Tl-t|}| = i^3
Let kl = { K ε {0,1}^64 : wt(K) = l}
Choose random m in {0,….,|Kl| - 1}
For j=0 to |Kl| - 1
Let K be the (m+j) mod |Kl|
If (AES encryption of M under key K yields C)
then return (K)
```

Figure 6. Key recovery method based on timing attack using hamming weight of the key

4. CONCLUSION

We study the implementation Bernstein's Timing Attack against AES cryptosystem and timing attack based on hamming weight. The countermeasures against Bernstein's Timing Attack are presented and the performance of the countermeasures is compared. Adding random delay would evolve as best countermeasure to Bernstein's timing attack. The proposed key recovery method in this paper can recover the key more efficiently using the hamming weight.

ACKNOWLEDGEMENTS

I would like to thank DST WOS-A for sponsoring me to do this research work and publish. I would also thank VFSTR University and KL University for their support and facilities to carry out my research work.

REFERENCES

[1] P. Kocher, *Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS and Other Systems*, in the Proceedings of Crypto 1996, LNCS, vol 1109, pp 104–113, Santa Barbara, CA, USA, August 1996.
[2] P. Kocher, J. Jaffe, B. Jun, *Differential Power Analysis*, in the Proceedings of Crypto 1999, LNCS, vol 1666, pp 398–412, Santa-Barbara, CA, USA, August 1999.
[3] D. Agrawal, B. Archambeault, J. Rao, P. Rohatgi, *The EMSide-Channel(s)*, in the proceedings of CHES 2002, LNCS, vol 2523, pp 29–45, Redwood City, CA, USA, August 2002.
[4] Advanced Encryption Standard, Federal Information Processing Standards Publications 197, 26 November 2001.
[5] Kinge, Pravin V., S.J. Honale, and C.M. Bobade, "Design of AES Algorithm for 128/192/256 Key Length in FPGA", International Journal of REconvigurable and Embedded Systems, Vol. 3, No. 2, 2014.

[6] EI Adib, Samir, Naoufal Raissouni, "AES Encryption Algorithm Hardware Implementation: Throughput and Area Comparison of 128, 192, and 256-bits Key", International Journal of Reconfigurable and Embedded Systems, Vol. 1, No. 2, pp. 67-74, 2012.
[7] F. Koeune and J. Quisquater. A timing attack against Rijndael. Technical Report CG-1999/1, June 1999.
[8] Jayasinghe, D.; Fernando, J.; Herath, R.; Ragel, R., "Remote Cache Timing Attack on Advanced Encryption Standard and countermeasures", *Information and Automation for Sustainability (ICIAFs), 2010 5th International Conference on*, vol., no., pp.177, 182, 17-19 Dec. 2010.
[9] Deevi Radha Rani, S. Venkateswarlu, "Timing Analysis Attack based on Hamming Weight", in International Journal of Applied Engineering Research, Vol. 9, No. 18 (2014) pp. 5161-5169.