Timing Attack: An Analysis of Preliminary Data

Yasin Fitri Alias, Mohd Anuar Mat Isa, Habibah Hashim
Faculty of Electrical Engineering,
Universiti Teknologi Mara, Selangor, Malaysia.
yasinalias90@gmail.com

***Abstract*—Timing attacks have caused an unprecedented evolution in the present cryptographic era where more and more cryptographic applications are running on embedded systems in a wireless environment. Paul Kocher, a well-known cryptanalyst was the first to successfully implement a timing attack on a cryptosystem. Subsequently many other timing attacks have been recorded while cryptographers tirelessly work on making the schemes more resistant to these and other side channel attacks. In this work, we examine timing information leaked from the computation of $g^a \pmod p$ and observe the timing variations of modular exponential computations by varying the key length as well as the operating frequency of our experimental Raspberry Pi board. We have chosen to compute the algebraic expression on a U-Boot Bare Metal platforms our platform and use the GMP bignum library to compute the numbers which are greater than 64 bit. We believe that the timing variations and patterns can lead to the early extraction of secret information in systems based on modular exponentiation. From our observations, there is a strong correlation of timing patterns when computing keys of the same length while the operating frequency used in the computation only affects the computational delay.**

***Index Terms*—Side Channel Attack; Cryptographic; Timing Attack; Modular Exponentiation; Raspberry Pi; Embedded Device; U-Boot Bare Metal; GMP Bignum Library.**

I. INTRODUCTION

Exhaustive key search attack [1] such as Brute Force can be an efficient tool for a finite key space and simple random key search also might be efficient. These kind of methods will no longer acceptable if the key space increases. 20 years ago, Paul Kocher implemented the first timing attack [2]. Timing attack is a side channel attack first presented by Paul Kocher whose work has become a major reference for timing and power analysis attacks. Side channel attack or side channel analysis is an attack on the implementation of a cryptosystem commonly dealing with the leakage of privacy data through side channel information [3]. In timing analysis based attack, timing channel can leak data or keys across a hardware implementation. An attacker will attack the cryptosystem by analyzing the time taken to execute the cryptographic algorithm. To solve the mathematical problems that embedded in the cryptosystem is very difficult [4]. So, the attack only focuses on the hardware or software that executes the schemes. Cryptanalysts or attackers only analyzed information that might lead to secret key disclosure or data compromise.

The author had implemented timing attacks on several cryptosystems based on their modular exponential [5] (e.g. Diffie-Hellman Key Exchange, RSA and El-Gamal). In this research we will implement timing attack on Diffie-Hellman Key Exchange (DHKE). The meaning of DHKE will be explained in section below. This paper will present how the initial data been collected in order to implement this timing attack. Based on our previous work [6], we decided to use U-Boot [7] bare metal in ARM based embedded devices to do this experiment.

A. Difiie-Hellman Key Exchange (DHKE)

DHKE [8] is used for exchanging cryptographic keys in unsecure network communication. The earliest idea about public key infrastructure is to do encryption and decryption algorithms using different keys. The DHKE method allows two entities that have no prior knowledge of each other to jointly establish a shared secret key over an insecure communications channel based on discrete log hard problems (DLP) as cryptographic strength [9][10]. The section below, we present how the DHKE protocol occur between two parties.

Let p be a prime and g be a generator of the multiplicative group z_p common for all participants [11]. Each party; Along and Busu, chooses randomly a secret number such as, a and compute the public parameter $A = g^a \pmod p$. A is made public. The public numbers are sufficient to establish a secret and common key for each pair of participants. Let us assume that Along has secret a and has computed:

$$A = g^a \pmod p \tag{1}$$

and similarly Busu has secret b and computed:

$$B = g^b \pmod p \tag{2}$$

Along then sends g^a to Busu and Busu sends g^b to Along. The shared secret key is $K = g^{ab}$. Along knowing a and g^b, can easily calculated g^{ab}. Busu can determine the secret in similar manner by computing g^{ba}. Thus the both parties obtain:

$$K = g^{ab} \pmod p \tag{3}$$

They can use as a common secret key. For passive eavesdropper to determine this key, he must find key satisfying the (3) with suitable a and b satisfying (1) and (2) where the gathered data is just:

$$(p, g, A, B) \tag{4}$$

This problem called Diffie-Hellman Problem. An obvious way to solve it is to solve the Discrete Log Problem. With input (p, g, A) and then use the value of a to compute K by:

$$K = B^a \quad (5)$$

The critical question is can K be found without obtaining much information about $a\ or\ b$ in the process. In [8][11][12]the authors stated that solving $a\ or\ b$ is basically the only way to solve the Diffie-Hellman Problem. More precisely they assume that the Diffie-Hellman Problem is hard if the Discrete Log Problem is hard.

II. Related Work

In this section, we review several related methods in implementing the timing attack effectively. Kocher (1996) [2] had implemented the timing attack on modular exponentiations and was able to break the cryptosystem. Dhem et al. (2000) [3] proposed several improvements based on Kocher's idea and was able to break a 512-bit key in a few hours. The authors state that the most of the smart cards did not change their design to prevent timing attack against them. Schindler (2000) [13] presented a new technique of timing attack. Using very limited knowledge, the author was able to obtain a secret key from an RSA with the Montgomery multiplication. However, he did not mention the tools that were used to attack the cryptosystem. Song et al. (2001) [14] had devised a timing attack to gain information from SSH. The weakness from the SSH enable attackers to gain secret information from the server. The authors also explained the method to prevent such attacks.

Hamza (2004) [15] had developed an attack based on genetic algorithm to attack the RSA cryptosystem. Based on the authors, this attack will not succeed if the cryptosystem used blinding technique and Montgomery algorithm. Brumley et al. (2005) [4] show that timing attacks from an OpenSSL-based web server over a local network can reveal RSA private keys. They modified the Kocher's attack [2] and successfully demonstrated that the attack can be conducted remotely against the server which is running OpenSSL. Bernstein (2005) [16] performed a timing attack to recover the AES key from the OpenSSL network server from another computer. The author claimed that the implementation of AES to write constant-time high-speed for common general-purpose CPU is extremely difficult and this made the attack successful. Maired et al. (2005) [17] explained the method used by Bernstein to attack the AES using timing attack. The authors had stated that Bernstein's attack can be prevented by several methods.

Scott et al. (2009) [18] had proposed a remote timing attack by analyzing jitter and network response time over internet and local network. The authors mentioned that by filtering the jitter; they can reduce the time measurement to get the critical information from the server. Strenzke (2010) [19] performed how timing attack can used to steal information from browser. We will use these case studies of timing attack to further our research in implementation of timing attack.

III. Proposed Method

We wish to eventually propose a method to reduce the time to search the private key ($a\ or\ b$) in the DHKE. Instead of using the brute force method, we will use timing attack as our method [6]. Timing information that leaked from the computation of the $g^a\ (mod\ p)$ will be used to extract the secret key or secret information unlike the man-in-the-middle attack which needs to know the values of $a, p\ and\ g$. Therefore, we need to record the computational time, t_a which compute the value of $(g^a\ mod\ p)$.

All the values of $p, g, a\ and\ b$ must be within the same bit size as the key length (e.g. 2048 bit) used. Assume that the most significant bit (MSB) for the minimum value of the key length is one (1) for the private key to be secure. Therefore, we need to find the lower and upper bound for the key length to limit the search for the value of a. To summarize this method, we state below the information that an attacker needs in order to launch the timing attack and the information the attacker wishes to uncover in the finality of the attack:

Attacker's knowledge: A, g, p, t_a
Attacker's goal: a

IV. Methodology

This section present the sequence of activities performed towards the completion of this experiment. For the experiments, two key lengths will be used, namely 1024 bits and 2048 bits and computation will be conducted on the Raspberry Pi board at two different frequencies, which are 700 MHz and 100MHz. Firstly, some random numbers $p, g\ and\ a$ are generated to produce keys of length 1024 bit and 2048 bit. The values of $p,\ g\ and\ a$ must be primes of the multiplicative group z_P respectively [11]. For each experiment of different key length, the value for p is the largest prime number of 1024 bit and 2048 bit depending on the key length in use. Meanwhile the value of g is the second largest prime number in the key length bit size. As for the secret value a, the random value must be in the range of the lower bound and the upper bound based on the key length. The source code must be compiled into U-Boot Bare Metal by using the ARM toolchain. Debian 7 is the operating system that been used to compile the U-Boot by using the ARM tool chain on the Raspberry Pi device. Besides, MobaXterm software is used in this experiment to take control of the raspberry pi board through the serial line. With it, we can access the U-Boot without physically connecting additional peripheral screen or keyboard. We connected the raspberry pi using the general purpose input/output (GPIO) to the computer through the serial port and the configuration for the serial port is shown as below:

Baud rate	: 115200
Parity	: none
Bits	: 8

The objective of this experiment is to observe the time taken to compute modular exponentiation $g^a\ (mod\ p)$ between the lower bound and upper bound of value a by using different frequency (e.g. 100MHz and 700 MHz). The frequency of the raspberry pi can be modified in the config.txt [7] as shown in Figure 1. This computation will be using same value of $p\ and\ g$ but different value of a for one thousand iterations. While, the value of a will be incremented from the lower bound by ten thousand for one thousand times. The tabulation of data will be explained in the next section.

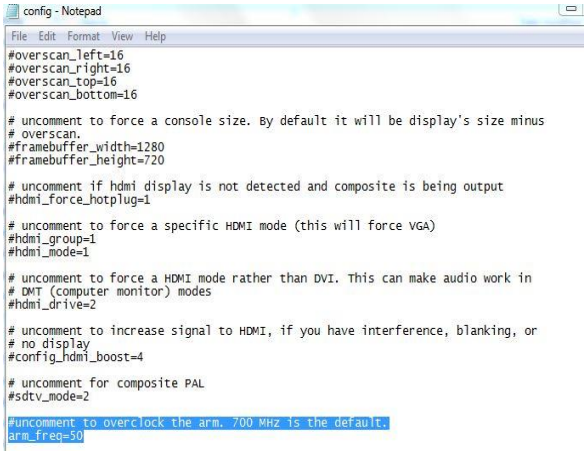

Figure 1: Config.txt

V. Result and Discussion

This section analyze the results of the experiment that conducted by using Raspberry Pi. The results have been taken for one thousand iterations of modular exponential $(g^a \ mod \ p)$ computations by using two different key lengths and frequencies. The value of p and g are same for the whole experiment, while the random value of a is different as it is incremented by ten thousand for each iteration.

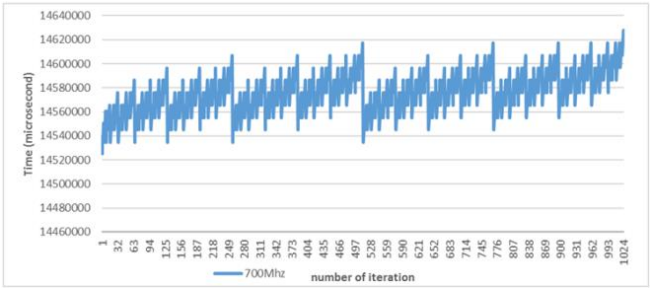

Figure 2: 700 MHz (2048 bit)

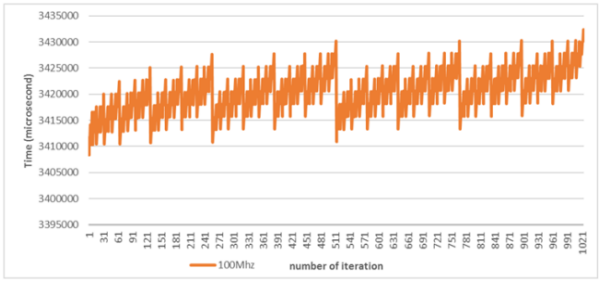

Figure 1: 100 MHz (2048 bit)

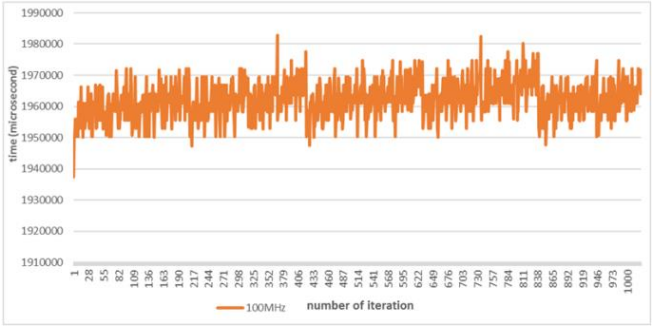

Figure 5: 100 MHz (1024 bit)

Figure 2 and 3 show the time taken for 2048-bit running in 700 MHz and 100 MHz raspberry pi operating frequency respectively. While Figure 4 and 5 show the time taken for 1024-bit running in 700 MHz and 100 MHz raspberry pi operating frequency respectively. For 2048-bit keys, the timing pattern remains the same even though the frequency is changed. Similarly, the same is observed for the 1024-bit keys. The difference in frequency is used to analyse the effect of the computational time of the modular exponentiation. Undoubtedly, we can identify the key length based on the patterns even when the frequency has changed. The most unexpected finding is the results for each key length have the similarity at certain points.

VI. Conclusion

In this paper, we present our observations on the timing data relating to the computation of modular exponential expression. The results obtained showed that different key length exhibit similar timing patterns when computing the modular exponentiation ( ) even though the frequency is changed. We believe the findings will help us to devise a timing attack to uncover the secret parameters by reducing the area of key-search. We have considered to implement this attack on embedded devices (e.g. raspberry pi) as more and more embedded devices are running cryptographic applications and are more susceptible to timing attacks. In the next stage of our study, we will expand our experiment to recover the secret key or information using our method in timing attack.

Acknowledgment

The authors would like to thank Universiti Teknologi MARA (UiTM) for providing research grant "600-RMI/FRGS 5/3 (160/2013)" for this research work.

References

[1] .Zhou Y. and Feng D., "Side-Channel Attacks: Ten Years After Its Publication and the Impacts on Cryptographic Module Security Testing," IACR Cryptol. ePrint Arch. no. 60503014, pp. 1–34, 2005.
[2] Kocher P., "Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems," Adv. Cryptology—CRYPTO' 1996.
[3] Dhem J., Koeune F., and Leroux P., "A practical implementation of the timing attack," Smart Card Res, 2000.
[4] Brumley D. and Boneh D., "Remote timing attacks are practical" Comput. Networks, 2005.
[5] key exchange Images W., "Diffie–Hellman key exchange," cinqueterreliguria.net.
[6] Alias M., Anuar Y.F., Isa M., Hashim H., " Sieving Technique to Solve the Discrete Log Hard Problem in Diffie-Hellman Key Exchange, pp. 129–133, 2015.
[7] Engineering D. S., 2014. http://www.denx.de/wiki/U-Boot.
[8] Diffie W. and Hellman M., "New directions in cryptography," Inf. Theory, IEEE Trans, 1976.
[9] Bellare M. and Rogaway P., "Introduction to modern cryptography," UCSD CSE, pp. 1–10, 2005.
[10] Goldwasser S., "New Directions in Cryptography: Twenty Some Years Later," in Proceedings 38th Annual Symposium on Foundations of Computer Science, pp. 314–324, 1997.
[11] den Boer B., "Diffie-Hellman is as strong as discrete log for certain primes," Adv. cryptology—CRYPTO'88, pp. 530–539, 1990.
[12] Boneh D., "The decision diffie-hellman problem," Algorithmic number theory, pp. 1–14, 1998.
[13] Schindler W., "A timing attack against RSA with the chinese remainder theorem," Hardw. Embed. Syst, pp. 109–124, 2000.
[14] Song D., Wagner D., and Tian X., "Timing Analysis of Keystrokes and Timing Attacks on SSH," USENIX Secur. Symp.

[15] Ali H. and Al-Salami M.,"Timing attack prospect for RSA cryptanalysis using genetic algorithm technique," Int. Arab J. Inf. vol. 1, no.1, pp. 80–84, 2004.
[16] Bernstein D., "Cache-timing attacks on AES," Compute, pp. 37, 2005.
[17] O'Hanlon M. and Tonge A., Investigation of cache timing attacks on AES. Sch. Comput. Dublin City Univ, 2005.
[18] Crosby S., Wallach D., and Riedi R., "Opportunities and limits of remote timing attacks," ACM Trans. Inf, vol. 12, no. 3, 2009.
[19] Strenzke F., "A timing attack against the secret permutation in the McEliece PKC," Post-Quantum Cryptogr. no. July. 1–29, 2010.