Journal of Computers Vol. 30 No. 1, 2019, pp. 87-95
doi:10.3966/199115992019023001009

An Improved Data Cache Timing Attack against RSA Based on Hidden Markov Model

Cai-sen Chen[1*], Yang-xia Xiang[2], Jia-xing Du[2], Zhiwei Cheng[2]

[1] Department of Training Center, Academy of Army Armored Force, Beijing 100072, China
caisenchen@163.com

[2] Department of Information Engineering, Academy of Army Armored Force, Beijing 100072, China
elephant_187@163.com,dhtl2004@163.com,cheng.zw@mail.scut.edu.cn

Received 9 January 2018; Revised 06 November 2018; Accepted 06 November 2018

Abstract. Considering the former attack algorithm was impacted by noise easily, a trace driven data Cache timing attack model on RSA was built, and the analysis algorithm of the power exponent was provided based on the Cache hit or miss side channel information. This paper proposes a new algorithm for automated analysis of the abundant Cache timing data based on these, which combines the vector quantization and Hidden Markov Model (HMM) cryptanalysis. We built a template for Cache timing analysis, and get the operation sequence using the Viterbi algorithm with HMM, to improve the precise of analysis. Finally, under the practical environment, we realized a Cache timing attack on RSA of OpenSSL. The experiment results showed that: the proposed analysis algorithm can effectively reduce the influence of noises from other processes, make the analysis of Cache timing data automatically, and the effectiveness and feasibility of Cache timing attack on RSA also can be improved.

Keywords: Cache timing attack, Hidden Markov Model, k-means clustering, RSA cryptosystem, vector quantization

1 Introduction

Cryptology is very important for information security, which consists of cryptography and cryptanalysis. From traditional cryptanalysis, cryptosystem is considered as one kind of mathematical abstraction, which is used to attack by input and output. With the increase of cryptosystem complexity and key length, the probability of success is reducing. In recent years, side-channel attack is becoming more and more popular, which is an attack based on information gained from the physical implementation of a cryptosystem, rather than brute force or theoretical weaknesses in the algorithms [1]. For example, timing attack, power analysis, electromagnetic leaks or even sound can provide an extra source of information, which can be exploited to break the system.

Cache access behavior information is used for side-channel attack, which was firstly proposed by Kocher [1]. Micro-architectural Attack is one kind of side channel attacks, which takes the different execution time via Cache hits and misses. Cache attack is designed for Block Cipher at the earliest stage, for example, DES and AES need to find S-box during execution. Cache model attack method on DES was proposed firstly by Page in 2002 [5]. Based on index of DES and characteristic of Cache access, the first trial of cache attacking on DES was implemented by Tsunoo in 2003 [6]. Remote timing driven timing attacks on AES algorithm for OpenSSL was accomplished by Bernstein in 2004 [7]. However, Cache timing attack on public key cryptosystem started relatively late. The idea of data Cache timing attack on RSA algorithm was firstly presented by Percival in 2005 [8]. Cache timing attack on RSA algorithm was designed and implemented based on multi-threaded cache access method and cache access behavior on malicious threads monitor password thread, A new kind of side-channel attack was proposed by Aciicmez in 2007 [11], which is Micro-architecture attack. Microprocessors can be attacked, which

* Corresponding Author

are instruction cache [12] and branch prediction unit [13]. Based on cache template timing attack about ECC, which was presented by Brumley [14], an improved I-Cache attack on ECDSA have been introduced by Aciicmez [15]. Shared Cache attack was proposed by B.Brumley, and data cache attack on ECDSA was implemented by OpenSSL code library [16]. The latest results show that security vulnerabilities of remote timing attack exists in the ECDSA algorithm on OpenSSL [17].

Theoretically, the complete power exponent can be achieved from the attack algorithms which was presented by Percival [8]. However, there are two problems during execution: Firstly, it is liable to various external factors, and it is very difficult to distinguish between elements in pre-calculated table and temporary variable in cache group. If it is not elements in the pre-calculated table, the original analysis algorithm becomes inefficient. Secondly, as the pre-calculated tables are small, which is in the sliding window of RSA algorithm, many elements may be mapped to one Cache group. For example, when key length is 1024-bit, the elements in 16 pre-calculated tables are mapped to 5~6 different cache groups, so we will not know visited elements by judging the cache hit and miss. This paper presents data analysis algorithm based on hidden markov model (HMM), which is used to improve the analysis accuracy.

Oswald have done related research about side channel attack on HMM, Karlof, Wagner [3] and Green [4] have built formalization analysis model on key, which is used to abstract side channel. Karlof and Wagner [3] have introduced the HMM model that single observation can be corresponding to one bit. But Green et al. [4] have proposed model that multiple observations can be corresponding to the same state, one system state may lead to different lengths of the side-channel information. This paper applies the HMM to actual cache timing attack and introduces the vector quantization, which can make many observations in the same state.

Based on RSA public-key cryptosystems in OpenSSL 0, RSA algorithm and Data Cache access behavior are analyzed, Cache hit and miss with sliding window algorithm are used, data cache attack model on RSA algorithm are built, vector quantization and HMM during data analysis are combined, K-means clustering analysis method are classified, a large number of Cache timing data are analyzed automatically, the accuracy of cache timing data and execution efficiency are improved, and Cache timing attack are applied to cryptosystem attack.

2 Data Cache Timing Attack Model on RSA

2.1 Trace Driven Data Cache Timing Attack model Based on RSA Algorithm

The sliding window method is used to improve the efficiency of modular multiplication in RSA algorithm, so it uses a pre-calculate table. This method increases possibility of Cache attack [8]. Attackers can get power exponent bit by Cache timing information.

Cache is a kind of small cache, which is between memory and central processing unit. It provides ways to access data and instructions which are accessed frequently. If data or instructions are not present in the cache, the extra delay time will happen. Data or instructions are fetched from CPU faster than memory, so the attacker can calculate the key bit by bit according the time difference.

On the simultaneous multithreading processor, cache timing attack on RSA algorithm can keep spy process synchronized with key process, and share one cache. In [8], it is described how the two processes were shared in one Cache. If the Spy Process (SP) and the Cipher Process (CP) share one cache, they have part of the same address maps. The new attack model is described in Fig. 1.

The structure of memory and Cache have been simplify further in Fig. 1. If a Cipher Table Data (CTD) is designed for memory blocks of CP instruction and a Spy Process Data (SPD) is designed for SP instruction, The size of the CTD is size of pre-calculated table, SPD and the whole Cache have the same size (Fig. 1). Follow these steps:

(1) Before CP instructions start, SPD will be accessed by SP process and loaded into the Cache (Fig. 1(b)).

(2) Start CP process, part of SPD, which have been loaded, will be unload from Cache, and elements are loaded into Cache (Fig. 1(c)).

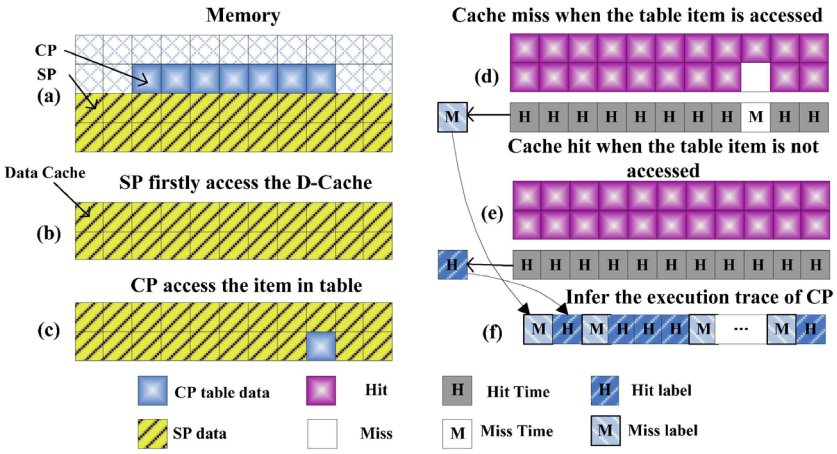

Fig. 1. Trace driven timing attack model based on the D-Cache

(3) When SP and CP are performed synchronously, SPD are accessed continually and measured accessing time of cache row. If SP access all of SPD, cache miss will be happened. It is cost much time to access higher-level memory. If CP is not lookup table, cache hit will be happened (Fig. 1(e)). Accessing time for each cache group is sum of all cache rows.

(4) SP loops through some steps at a certain frequency, which are filling Cache, execution time of each Cache group are measured, and Cache access trace of CP are got. After that, Cache track timing information of CP will be got.

(5) Based on cache track timing information, the bit information of power-exponent was deduced by the side-channel information and power-exponent algorithm.

The former attack algorithm was impacted by the noise easily, this paper proposed an improved method with random process.

2.2 The Analysis Algorithm of Exponent Based on the Cache Timing

Modular multiplications are implemented by sliding window algorithm, $x=C^{d_p} \bmod p$ is disassembled into sequence arithmetic ($x=x^2 \bmod p$) and multiplication ($x=x*C^{2k+1} \bmod p$), where C^{2k+1} is pre-calculated multiplier : $\{C, C^3, C^5,\ldots, C^{31}\} \bmod p$. In this paper, A 512-bit index (d_p) is studied, the sliding window size is 5.

Based on the Cache time attack model, if operation of lookup table is happen, square and multiplication can be calculated with cache access hit and miss. Based on sliding window algorithm, if multiplication is happen, power-exponent is 1. If square is greater than 5, power-exponent is 0. Window values of power exponent are got by index value [8].

3 Vector Quantization of Cache Timing Data

3.1 Vector Quantization Using K-means Clustering

Vector quantization (VQ) is designed for automated analysis of cache timing data. If there is only one trace, data are analyzed annually; Otherwise, VQ is used for analysis.

$R^n \rightarrow C$ *map V by VQ, where* $C=\{c_1,c_2,\ldots,c_a\}$ is codebook, $D(v,c)$ is represent for Euclidean distance of v and c. L is used for marker of vector C in codebook. To build codebook, K-means clustering algorithm will be used. Starting from training vector and predefined label value set($T=\{(t_1,l_1), (t_2,l_2),\ldots, (t_j,l_j)\}$), if t_i have same features, which are corresponding to the same labels. Based on K-means clustering, specified data in large sample size can make cluster analysis gradually.

The steps to *K*-means clustering are as follows.

(1) Initializing cluster center. Based on our experience, 3 samples are selected from cache timing data, which are multiplication, square and start/end, and labeling them with M, S, E respectively.

(2) Based on the nearby principle, each sample is classified by the Euclidean distance, calculating the sample mean again, updating the cluster center and classing all samples.

(3) Estimating the rationality of clustering results. If clustering result is irrational, classification will be modified until the termination of the algorithm. The final cluster centers are adapted to Cache timing data temple at the end of the loop.

Data analysis contains training and identification, which are undertaking VQ. When training is on, VQ is used for cluster analysis, that is to say, reference codes are built by a number of representative vectors, when identification is on, and samples are replaced with one code in codebook, which is on the principle of minimum distortion.

3.2 Building a Template of Cache Timing Data

Based on the Cache timing attack model, we get the cache timing data on Pentium 4 L1 Cache which have 32 cache sets, so the 32-d trace data is $t_j \circ$ Assumed that L = {M, S, E}, which label square, multiplication and start/end. Then the initial data T are training. During the training process, attackers can generate a specific private key and signature which are used to generate training data. If one spy is executed many times, memory of cache timing data does not have to be loaded. After that, there are quite dramatic differences among each vector.

In order to build the data analysis template, we modified software which perform a single operation. For modular multiplication implementation on RSA, square or multiplication is executed one time by building special power exponents and synchronization with spy process, and cache timing data is got, which is used to building the data analysis template and VQ codebook.

To reduce the influence of the error vector, dividing-and-dealing strategies are used. Assumed that $T = \bigcup_{l \in L} \{(t_i, l_i) : l_i = l\}$ is collection for single training vector(l_i), The results of clustering centers are added to the code book (C) by K-means clustering analysis. C are redefined by Learning Vector Quantization (LVQ) algorithm, and the code book of reasonable error rate is got, which is used to Cache timing attack module.

4 An Improved Cache Timing Attack Based on HMM

4.1 HMM of Sliding Window Algorithm Operating State

HMM is a well-studied model for finite-state stochastic processes [19]. An execution of a HMM contains a sequence of hidden, unobserved states and a corresponding sequence of related, observable outputs. So HMM is a double stochastic process, one stochastic process is move from one state condition to another state condition, the other is observation of state. HMM is used mainly to solve some solutions those are evaluation, decoding and learning. Parameters of HMM consist of State transition probability matrix (A=$P(S_i|S_j)$), observation probability matrix (B=P($S_i|O^j$)) and initial state probability distribution vector (Π=$\{\Pi_1, \Pi_2, \ldots, \Pi_N\}$) [19].

The hidden states of HMM corresponds to the operation sequence of the algorithm in this paper, square and multiplication are implemented with sliding window algorithm. The state cannot be observed directly, but it can be deduced from vector sequence, and each observation show the various states by some prior estimation of density distribution, State set is defined as S(S=$\{S_1, S_2, \ldots, S_7\}$). Sliding window length is different, operation sequence is also different, state-transition diagrams are shown in Fig. 2.

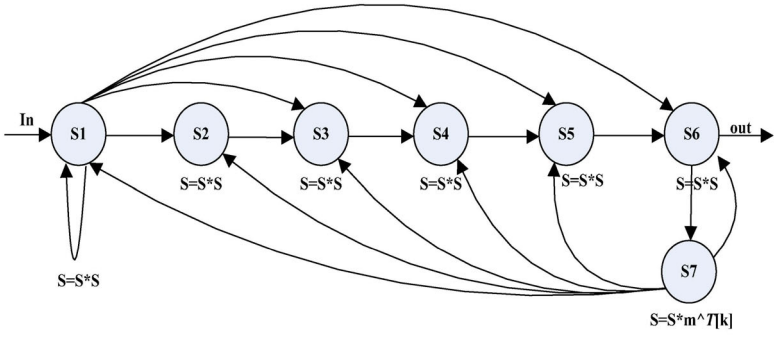

Fig. 2. The state conversion diagram for sliding window algorithm

S_1 denotes square operation if key-bit is 0, otherwise S_2 to S_6 denotes square operation. Obviously, the length of the non-zero window is 5. Based on the size of the sliding window, S_7 represent square operation that it is look-up table.

4.2 Recover the Operation Sequence of RSA Based on HMM

HMM is often used to eliminate the noise signal in signal processing. This paper attains internal operation sequence of cipher algorithm from VQ.

Through above analysis, side-channel information can be divided into three classifications: multiplication operation, square operation and begin/end operation. Based on the literature [14], the input of HMM is label sequence, operation sequence is deduced by cipher algorithm, and the power exponents are calculated by operation sequence. Recover steps are as follows:

(1) For the square and multiplication operation, cache access behavior are represented by some time vectors, which is called "template".

(2) To eliminate noise and reduce the size of the code book, a corresponding vector quantization codebook is created for each operation by above model.

(3) To enable hidden Markov models reflecting control flow of cipher algorithm, HMM parameters (A, B and Π) are confirmed by training, output of VQ is used as input of HMM.

(4) Based on the results of VQ for cache timing data, possible operations sequences are deduced by Viterbi algorithm [20].

Power exponents are calculated with square and multiplication operation sequence, and the complete key is recovered by the idea of lattice attack.

4.3 Recover the Private Key by Lattice Attack

In an ideal world, the complete bits of exponent d_p and d_q can be obtained based on the proposed algorithm. In fact, only partial scattered bits of d_p and d_q could be got. On the one hand, the temporary variables can be mapped to the pre-calculate table in a same Cache set, SP cannot collect the complete information effectively. On the other hand, the same cache set is mapped to several indexes of the pre-calculate table, one table index values may also correspond to two adjacent Cache set. Although the literature [9] have shown that, modulus N can be decomposed by lattice reduction methods [10], which are completed based on continuous key bit and bit information of prime factor. So this method is not suitable for recovering key in this paper, we have to find a new method for discomposing modulus N.

Assume that the modulus N and the public exponent e are known, and some information about the private exponents d_p and d_q have been obtained. Then from the RSA algorithm, we note that: $ed_p \equiv 1(\bmod p-1)$, $ed_q \equiv 1(\bmod q-1)$. and so for some $k_p, k_q \in Z_e^*$, $ed_p \equiv k_p(p-1)+1$, $ed_q \equiv k_q(q-1)+1$.

And after some simple algebraic manipulation, a new equation can be described as follows:

$$Nk_pk_q = (pk_p).(qk_q) = (ed_p + k_p - 1).(ed_q + k_q - 1) \quad \textbf{(1)}$$

We consider the sets S_n containing all ordered 4-tuples (k_p, k_q, d_p, d_q) which satisfy Eq.(1) modulo e.2^n, match the scattered bits of d_p *and* d_q, and have $0 \le d_p, d_q < 2^n$. The set S_0 simply contains tuples (k_p,k_q,0,0) where $Nk_pk_q=(k_p-1).(k_q-1)$ mod e. So, k_q is uniquely determined by N and k_p, and $|S_0|<$e, where $|S_0|$ defines the size of sets S_0. Further, if we have got the set S_0, we can computer the set S_{n+1} by lifting each 4-tuple in S_n into two 4-tuples. Because the n^{th} bit of can be either 0 or 1, once it has been chosen, Eq.(1) will determine the n^{th} bit of d_p, and discarding any 4-tuples which do not match our observations.

If we know neither the n^{th} bit of d_p nor the n^{th} bit of d_q, then $|S_{n+1}|=2|S_n|$. If we know one of the bits, then each 4-tuple in maps to a unique 4-tuple in S_{n+1}, and we have $|S_{n+1}|=|S_n|$. If the both bits are known, then the size of S_{n+1} will be approximately half the size of S_n. $S_n \approx e*(1/2)^k*2^l$, where k is defined as known amount for the n^{th} bit of d_p and d_q, l is defined as unknown amount. Consequently, the size of S_n as n increases follows a random walk starting from e. If we can obtain more known bits, the search space of the unknown d_p and d_q will be smaller. If known bits are more than unknown bits, the overall trend of the size of sets is downwards, and the sets are unlikely to ever become so large as to be unwieldy. Once the set S_{512} has been computed, it is a simple matter to test the remaining candidate exponents and retrieve

the factorization of N, recover the private key d finally.

5 Experimental Results and Discussion

5.1 Environment Configuration and Cache Timing Collection

According to the attack principle and proposed improved Cache timing data algorithm, the Cache attacking algorithm on RSA is programmed, the cache set access during the RSA decryption is timed accurately by CPU's RDTSC, and the size of cache S is got by CPUID. In Linux, if the hyper-threading of CPU is on, code process and spy process are performed synchronously by the *fork* function. Before operation, spy process need to clear cache, the pre-calculate table elements in Cache are replaced with accessing continuous area in memory for S after the code process ends, and minimize the operation of other processes. For the convenience of research, we concentrate on Hyper Threading CPUs processor featuring hyper threading technology. The detail environment configuration of our attack is shown in Table 1:

Table 1. Environment configuration of Cache attack on RSA

Configuration item	Parameter	
Operating system	Linux Fedora 8	
Simultaneous multi-threading	Turn on	
OpenSSL	OpenSSL v.1.0.1	
CPU	Intel Pentium 4 (R) 3.0 GHz	
Memory	1 GB	
$L1$ Cache	Cache size: 16KB	associative size: 8 way
	Cache line size: 64B	
$L2$ Cache	Cache size: 2048KB	associative size: 8 way
	Cache line size: 64B	

In experiments, the priv.key is randomly generated, and during the spy process is executed, a character array b[2^14] which size is equal to L1 is constructed, so the Cache can be cleared, and $b[k]$ is sequentially accessed, where k is $64i+2048j$ ($0 \le i < 32$, $0 \le j < 8$), access time of cache set is the sum of access time for 8 caches, 32 cache timing data are used as initial data. Above steps are repeated until code process is ended, and cache access state for code process are monitored. There are about 1024-bit 25*10^7 CPU clock cycles when RSA key performs one signature operation. There are 8230 CPU clock cycles when SP process accesses all cache sets. Therefore, assumed that the number of cycles for SP is 32768, 32768*32 samples are gathered for one RSA signature. To process a number of samples, VQ and HMM are combined in this paper.

5.2 Combining VQ with HMM for Data Analysis

Experiment results show that the number of clock cycles for cache hit is about 120, and the number of clock cycles for cache miss is about 150. In fact, if the square operation is not look-up table, the number of clock cycles is 165~220. If the multiplication operation is look-up table, the number of clock cycles from L2 is 280 and 330 from memory. Based on a 32-d data-collection, we need to filter data for sliding window algorithm and delete data at beginning and ending. Based on Cache timing data model and VQ, different categories are identified as $L\{M, S\}$, those result are used as the output of HMM. The partial analytical results are shown in Fig. 3.

In Fig. 3, the X-axis represents the time sequence of the implementation, and the Y-axis represents the timing data for 32 cache sets, the color depth for each grid represents the length of time for cache set. The darker the color depth is, the shorter time is.

Because number of elements in the pre-calculate table are 16, the size of BIGNUM is 20 byte [0]. If the row size of cache is 64 byte, 320 byte for pre-calculate table is correspond to 5~6 cache sets continuously. So, as shown in Fig. 3, we can conclude that the pre-calculate table is mapped to the 11^{th}~16^{th} Cache sets whose timing data are always low.

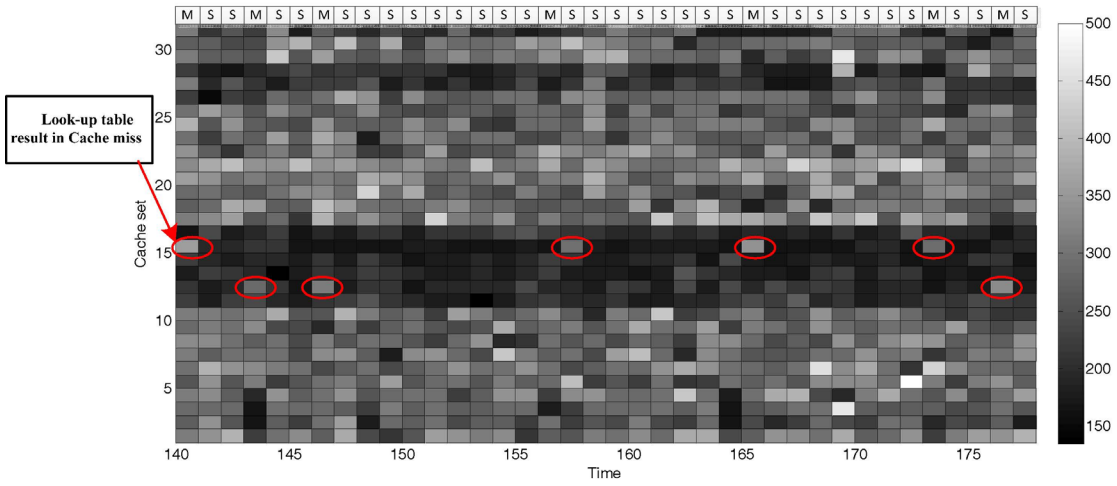

Fig. 3. Pseudo-color diagram of Cache timing data

As shown in Fig. 3, the 7 red circles are for labeling the look-up pre-calculation table operation, which also denote multiplication, and the black area is represent the square or other operation, such as Montgomery's operation. The operation label sequence (M and S) on the top line in Fig. 3 are got by using VQ.

The parameter of HMM can be trained using the operation of square and multiplication, which has been got When d_p is known. Based on the given observation-sequence, HMM is trained with *esthmm* function and Baum-Welch algorithm, the training results are shown in Table 2.

Table 2. HMM training results

Times of repetitive training	74	
The number of observable states *M*	2	
The number of hidden states *N*	2	
Probability matrix of state transferring *A*	0.818703	0.182297
	0.999930	0.001070
Probability matrix of output symbols *B*	0.003104	0.997896
	0.972795	0.028205
Initial state probability distribution *pi*	0.001003	0.999997

As shown in Table II, *A* denotes the probability matrix of state transferring that is the transition probability between the square and the multiplication. The results show that the probability of square to square is 0.818703 and to multiplication is 0.182297, but the probability of multiplication to square is 0.999930 and to multiplication is 0.999930.

B denotes the state transition matrix between the observe state and the hidden state. There are obvious difference between the square and the multiplication for characteristics of Cache timing information. Those are consistent with operation sequence of sliding window algorithm.

Pi denotes the initial state probability that defines the probability for each state at the start moment. As the power exponents bit begin with 1, the probability of square is 0 by default, and we can find that the probability of square is 0.001003 from the learning result.

Finally, the highest probability of operation sequence can be deduced by HMM and Viterbi algorithm, and the power exponents bit information can be recovered by the operation sequences of power exponents.

5.3 Experimental Results and Comparison

Based on the our experimental data, we need to define two arrays V and H. V is used for holding the output of VQ, and the output is also used as the input of HMM; H is used for holding the predicted state sequences of HMM. Based on the sliding window algorithm, the key segment can be deduced by the left side of the operation sequence and each power exponent segment can be corresponded to one sliding

window bit. We can conclude that there are one square operation and 3 multiplication operations, as the first candidate values for the power exponents segment are {1001,1011,1101,1111}, and the second one is {11,01}. If the window size is 5 and the square operation which is continuously executed for more than 5 times, the key bit is 0. So power the exponents fragment is: 1*XXX*1*X*10XXXX1, where x denotes 0 or 1. Using the same method, the other candidate values also can be deduced, and the complete RSA key could be got using the method in 4.3.

In experiments, the square and multiplication operations are performed for 10000 times, each time we get 32768*32 data samples, so the analysis model is built in a large sample size. If a RSA signature operation is executed in attack, spy process could collect 32768*32 cache timing data. Ideally, at the beginning of the module and power operation, the RSA signature operation can get about 310 bits of d_p. It may cost much time for attacking in data analysis phrase and key deducing phrase. The results are compared with experiments in literature [8], as shown below.

(1) In the previous traditional attack, during the Cache timing information collection, the execution opportunity for spy process and code process must be controlled, so a time delay had been inserted into the square operation. That is to say, it is still only a simulation experiment. However, in our experiment, we use the simultaneous multithreading processor, so the SP and CP can be executed synchronously; the attack experiments are implemented in practical environmental situations.

(2) In the practical experiment, the noise is very heavy in Cache timing information, so it is very difficult to deduce the sequence of square and multiplication operation. However, using the proposed data analysis algorithm based on the HMM and VQ, the Cache timing data be mingled with some noise can be analyzed and partial information of power exponent can be obtained automatically. So, the new proposed attack scheme can improve the feasibility of trace-driven D-Cache timing attack against RSA.

(3) The previous studies only provided some methods about how to get partial information of power exponent using Cache attack, but further studies about how to recover the complete private key from the partial power exponents have not been done. Based on idea of lattice reduction attack, the method of recovering the complete private key using discrete power exponents is deduced.

6 Discussion and Future Work

This paper focuses on the trace-driven Cache timing attack on RSA algorithm which uses the sliding window algorithm for modular exponentiation, we study the principle of Cache timing attack, analyze the difficulty in the previous trace-driven Cache timing attack on RSA. Based on the trace-driven Cache timing attack, we proposed an advanced analysis algorithm of data analysis to deduce the operation sequence of square and multiplication, using the *k*-means clustering, VQ and HMM, by analyzing the correlativity between the power exponent bits and the operation sequence, and the characteristic of the window size and the correspondence relationship between the pre-calculated table indexes and the window values. We also provide the key during the attack and some solutions for the potential problems of the real attack, including how to build the data template, perform the state transition on sliding window algorithm, build the HMM model, classify the Cache timing data effectively, and deduce the highest probability of operation sequence with HMM, and so on, finally recover the complete key with Lattice attack. We realize a Cache timing attack on RSA of OpenSSL under the Linux operating system which ran on the simultaneous multithreading processor, the trace timing data is collected by the spy process which can run with the cipher process in parallel, and the power exponents are analyzed. Using the discrete known bits of d_p and d_q, the whole private key can be recovered using lattice reduction methods. The experiment results demonstrate: the new proposed analysis algorithm of the operation sequence using VQ and HMM can improve the precision and efficiency of analyzing the Cache timing data adequately; it is possible to obtain approximately 310 bits out of each 512-bit exponent d_p or d_q, further reduce the search space of the key bits. So, the correctness and feasibility of trace-driven Cache timing attack are verified by the attack results.

Although the new proposed attack scheme based on VQ and HMM can improve the feasibility of D-Cache timing attacks against RSA, however in the future research, there are still some problems needed to be considered: Firstly, how to further reduce the affect of the noise effectively, especially the attack in a multi-core CPU processor; Secondly, we need to improve the intelligent of automated analysis, especially use vast cache-timing data.

Acknowledgments

The authors would like to thank anonymous reviewers for their valuable comments. This research was supported by the National Natural Science Foundation of China under Grant No. 1836101 and 61402528.

References

[1] P.C. Kocher, Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems, in: Proc. 16th Annual International Cryptology Conference Santa Barbara, 1996.

[2] E. Oswald, Enhancing simple power-analysis attacks on elliptic curve cryptosystems, in: Proc. the 4th International Workshop on Cryptographic and Embedded System, 2002.

[3] C. Karlof, D. Wagner, Hidden Markov model cryptanalysis, in: C.D. Walter, C.K. Koç, C. Paar (Eds.), CHES 2003, Springer, Heidelberg, pp. 17-34.

[4] P.J Green, R. Noad, N.P. Smart, Further hidden Markov model cryptanalysis, in: J.R. Rao, B. Sunar (Eds.), CHES 2005, Springer, Heidelberg, 2005, pp. 61-74.

[5] D. Page, Theoretical use of cache memory as a cryptanalytic side-channel. <https://eprint.iacr.org/2002/169.pdf>, 2002 (accessed 06.06.18).

[6] T. Yukiyasu, S. Teruo, S. Tomoyasu, S. Maki, M. Hiroshi, Cryptanalysis of DES implemented on computers with cache, in: C.D. Walter, C.K. Koç, C. Paar (Eds.), CHES 2003, Springer, Heidelberg, pp. 62-76.

[7] D.J. Bernstein, Cache-timing attacks on AES. <http://cr.yp.to/papers.html\#cachetiming>, 2005 (accessed 06.06.18).

[8] C. Percival, Cache missing for fun and profit. <http://www.daemonology.net/hyperthreading-considered-harmful>, 2005 (accessed 06.06.18).

[9] D. Brumley, D. Boneh, Remote timing attacks are practical, in: Proc. the 12th Usenix Security Symposium, 2003.

[10] D. Coppersmith, Finding a small root of a bivariate integer equation; factoring with high bits known, in: U. Maurer (Ed.), EUROCRYPT'96, Springer, Heidelberg, 1996, pp. 178-189.

[11] O. Aciicmez, W. Schindler, A major vulnerability in RSA implementations due to microarchitectural analysis threat, in: Proc. 14th ACM Conference on Computer and Communications Security (ACM CCS'07), 2007.

[12] O. Acıiicmez, W. Schindler, A vulnerability in RSA implementations due to instruction cache analysis and its demonstration on openSSL, in: T.G. Malkin (Ed.), CT-RSA 2008, Springer, Heidelberg, 2008, pp. 256-273.

[13] O. Acıiicmez, C.K. Koc, J.P. Seifert, On the power of simple branch prediction analysis. in: Proc. the 2nd ACM Symposium on Information, Computer and Communications Security (ASIACCS 2007), 2007.

[14] B.B. Brumley, R.M. Hakala, Cache-timing template attacks, in: Proc. ASIACRYPT 2009, 2009.

[15] O. Acıiicmez, B.B. Brumley, P. Grabher, New results on instruction cache attacks, in: Proc. CHES 2010, 2010.

[16] B.B. Brumley, N. Tuveri, Cache-timing attacks and shared contexts, in: Proc. COSADE 2011, 2011.

[17] B.B. Brumley, N. Tuveri, Remote timing attacks are still practical. <http://eprint.iacr.org/2011/232.pdf>, 2011 (accessed 06.06.18).

[18] T. Kohonen, Self-organizing maps, Proceedings IEEE 78(9)(1990) 1464-1480.

[19] L.R. Rabiner, A tutorial on hidden Markov models and selected applications in speech recognition, Proceedings IEEE 77(2)(1989) 257-286.

[20] A.J. Viterbi, Error bounds for convolutional codes and an asymptotically optimum decoding algorithm, IEEE Transactions on Information Theory 13(2)(1967) 260-269.