Timing Attack Prospect for RSA Cryptanalysts Using Genetic Algorithm Technique

Hamza Ali and Mikdam Al-Salami
Computer Science Department, Zarka Private University, Jordan

Abstract: *This paper presents an approach to cryptanalysis of RSA cryptosystem based on the application of genetic algorithm. The search utilizes the idea of timing attack as computation time information may leak due to different modular operations throughout the RSA encoding. This approach suggests a speed up process, aiming at reducing the required number of plaintext-ciphertext samples needed for a successful timing attack. The proposed notion of timing attack outlined in this work with its preliminary implementation, have given encouraging results on RSA cryptosystem samples. Further work carried on to implement the idea of genetic algorithm technique to practical RSA system has demonstrated encouraging results.*

Keywords: *Cryptography, cryptanalysis, genetic algorithms, timing attacks, RSA, key search.*

Received February 16, 2003; accepted June 26, 2003

1. Introduction

Any exhaustive key search attack, such as “Brute Force” would be an efficient tool for a finite key space problem. Also, simple random key search might be effective. However, as the key space increases, neither Brute Force search nor random search would be practical nor acceptable, but other means may be sought.

Cryptanalysts look for any specific information that when analyzed, might lead to data compromise or secret key disclosure. Cryptosystem can be viewed as either a mathematical objects, or as a concrete implementation of that mathematical object. Cryptanalysis has been directed against mathematical objects (i.e. differential and linear cryptanalysis), regardless of the implementation considerations. Most of such attacks have a theoretical consideration and may not work against cryptosystem in practice. Another kind of cryptanalysis targets specific implementation details. Timing attack [4] based on time measurement with respect to particular implementation of cryptosystem. Cryptographic algorithms often perform computation in non-constant time, if such operation involves secret parameters, these timing variations can leak some information. With enough knowledge of the implementation at hand, statistical analysis could even lead to the total recovery of these secret parameters [1].

A number of possible applications of timing attack against various cryptographic algorithms were introduced. The amount of time required to perform private key operations against Deffie-Helman, RSA, DSS were outlined by Kocher [5, 10, 11], the timing attack against Rijndael (i.e. AES system) by Koeune and Quisquater [6], simple and differential power consumption analysis by Kocher *et al.* [5]. And a practical implementation of timing attack of CASCADE smart card against RSA algorithm presented by Dhem *et al.* [2]. This implementation has shown an efficient performance in key disclosure, where error detection technique was introduced, to guide the attack and to reduce the number of messages samples that were required to activate the attack. On the other hand, exploiting regularities in the cipher system or the language have led to the implementation of genetic algorithms in cryptosystems, see for example Spillman *et al.* [12] and Metthews [9] for breaking simple substitution ciphers and classic cryptographic systems (substitution and transposition), respectively.

The present work introduces the idea of using genetic algorithm to enhance the timing attack. This idea is based on the concept of error detection to distinguish the partial correct secret key from completely wrong one. The principles of designing fitness function along with other genetic operations are reported here. In section 2, Genetic algorithms technique is defined and reviewed with the aim of using it in collaboration with the timing attack process on RSA, and error detection is outlined in section 3. The proposed principle for cryptanalysis attack is formally defined and discussed in section 4 and finally the paper is concluded in section 5.

2. Genetic Algorithms

The main feature of generic algorithms (GA’s) is their ability to add direction to what seems to be random search. This might produce powerful and efficient search technique. These algorithms attempt to solve complicated mathematical problems by simulation, to certain extent, the way in which biological genetic processes operate, see Liepins and Hilliard [7, 8]. They use the concept drawn from the theory of evolution to “breed” progressively better solutions to problems with very large solution space. Therefore, for a correct key selection, the goal is to search through the key space (each represented by some binary string) in order to find the one that maximize the possibility of a correct answer. A genetic algorithm begins with a randomly selected population of function inputs represented by strings (chromosomes), which are in the form of guessed key strings. GA uses the current population of strings to create a new population such that the string in the new population, on average gives better fitting results than those in the current population. This is achieved by assessing the fitness of each attempt. The more successful keys would then be used to create new successive generations of fitter key combinations. This process is repeated until an optimal key emerges, that produces substantial decryption. GA follows three processes cycle for transition from one population to the next one in a similar way to that found in living creatures. This cycle consists of *selection, crossover,* and *mutation*, as shown in Figure 1 [12].

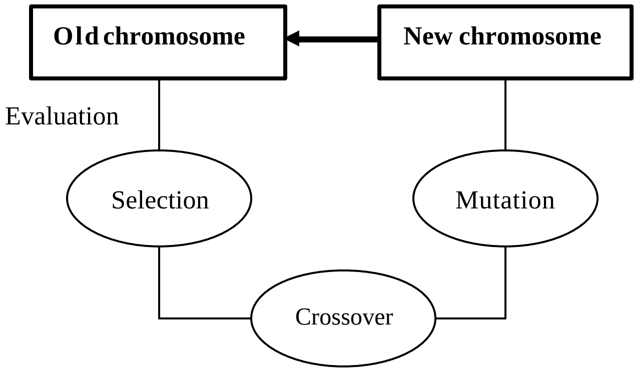


Figure 1. The basic generic algorithm cycle.

The selection process determines which of string (chromosome) in the current generation will be used to breed the next generation. This is based on biased or *directed* random selection strategy. The crossover process determines the expected string of the next generation, which will result into a pair of parents. The final step is mutation that follows some mutation probability, which is set at the start of the algorithm. Some changes will take place, producing the generations of strings. This cycle is repeated, creating new generation of strings every round, until certain criterion is reached.

3. Timing Attack and Error Detection

RSA cryptosystem uses exponentiation operations to perform both encryption and decryption. An exponentiation algorithm works by scanning the key from left to right, as shown in Figure 2. The computations are done through a sequence of square and multiply operation in which key bits are involved. Timing attack against this algorithm could be aimed at either the square or the multiply operations. The result of Dhem *et al.* [2] have shown that attacking the square operation is more efficient than attacking the multiply operation. For each bit of the secret key captured message sample is categorized into four sets. At first, the most significant bit is taken as 1, so both multiplication and squaring operations will be performed. Then, when the next most significant bit is considered, it is required to determine whether an additional reduction is necessary or not, depending on its value being 1 or 0. If it is taken as 1, then multiplication is performed and deciding if an additional reduction is required or not followed by squaring. Doing this process for every message, the samples are divided into two subsets, M$_1$ contains messages that require reduction and M$_2$ contains messages that require no reduction. If this bit is considered as 0, which means only square operation will be done, the samples are divided into two subsets, M$_3$ and M$_4$ based on the necessity of additional reduction or not. Further details can be found in [2].

```
$x = m$
For $i = n – 2$ down to 0
  $x = x^2$      //square
  If $k_i = 1$ then
     $x = x . m$    // multiply
  end for
return $x$
```

Figure 2. Left and right exponentiation.

For each set, the measured mean time is computed and compared to make a decision about the key bit to be either 1 or 0. For error detection, the difference between mean times of M$_1$ and M$_2$ sets and the difference between mean times of M$_3$ and M$_4$ sets are subtracted from each other in order to compute a difference value as a figure of merit. This value should be significant enough to give an indication that the bit value is guessed correctly. The values of this figure of merit will show a noticeable decrease if the computation was done using wrong key bits. In this work, this feature will be used to evaluate the correctness of key.

4. The Principle of the Attack

Timing attack against RSA cryptosystem shows an efficient performance in revealing the decryption key [2]. However, the number of required sample messages is still rather large. In this work we tried a method that aims at reducing the number of these samples in order to increase the possibility of a successful attack. This enhanced method is based on using genetic algorithm to search the large key space and utilize the feature of GA to produce and improve a set of possible keys (population), for generation over generation. A single key with a good feature (correct key bits) will survive at the end of the search. The crucial idea enabled the attack on RSA is the possibility of evaluating the suggested keys (population individuals) and distinguishing the wrong key from the partially correct key. The evaluation process depends on the concept of error detection. In the proposed system, the key population is a set of chromosomes, each consists of a sequence of bits represents an estimated key. The length of each chromosome equals to the maximum expected key length. The most important idea to discuss here is the way of evaluating a given key in order to assign different fitness values for each individual key in the population. This will enable the GA to evolve into good keys over a sequence of generations. A good key refers to a key that has a large number of correct bits or it may rather be completely correct. In other words we should have a method to evaluate the fitness (or correctness) of each key.

In the following, we demonstrate the way of building a fitness function that uses error detection technique to assign a fitness value for each individual key. An efficient evaluation and crossover operation will help to have a direct search through a huge search space. Then a description of crossover operation is presented in addition to other genetic operation related to the process of producing new population from an old one.

4.1. Key Evaluation

For the basic formal definition of fitness function, we first define the following key, k, as follows:

$$k = \{ k_1, k_2, k_3, \ldots, k_n \} \quad (1)$$

Where k represents an individual key or chromosome, which consists of n bits, each could be either 0 or 1. And suppose we have:

$$M = \{M_1, M_2, M_3, \ldots, M_s \} \text{ and}$$
$$T = \{T_1, T_2, T_3, \ldots, T_s \} \quad (2)$$

Where M is a set of captured messages (plain text), and T represents a sequence of time amount needed to compute a signature of each of these message using decryption key.

For these sequences, we can define the measured time parameter T_{ij}, as follows:

$T_{1j} = \{r_i \in T \mid k_j=1$ and M_i required reduction at bit j of the individual key k$\}$

$T_{2j} = \{r_i \in T \mid k_j=1$ and M_i required no reduction at bit j of the individual key k$\}$

$T_{3j} = \{r_i \in T \mid k_j=0$ and M_i required reduction at bit j of the individual key k$\}$

$T_{4j} = \{r_i \in T \mid k_j=0$ and M_i required no reduction at bit j of the individual key k$\}$ (3)

The function $\mu(T_{ij})$ is calculated as the average computation time for the message samples. Also, we can define the computation average time differences as:

$$D = \{D_1, D_2, \ldots, D_n\} \quad (4)$$

Where

$$D_i = \left| \, |\mu(T_{1i}) - \mu(T_{2i})| - |\mu(T_{3i}) - \mu(T_{4i})| \, \right| \quad (5)$$

The error detection for the timing attack can be of the form shown in Figure 3, where the computation average time difference is plotted as a function of the suggested key length (bits).

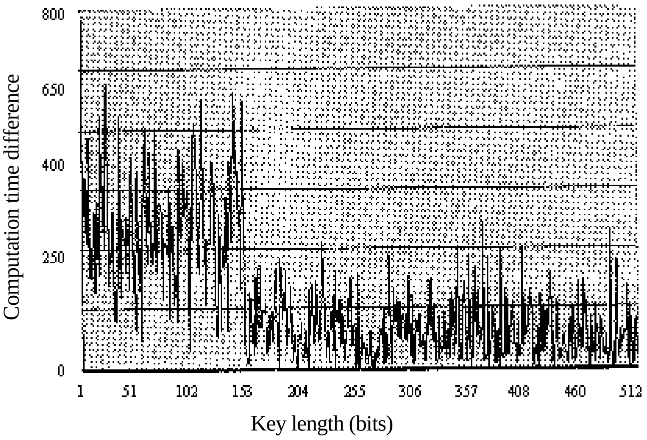


Figure 3. Error detection for 512-bit key [3].

Finally, the fitness value for key, k can be computed by the following formula:

$$\text{Fitness}(k)=\{\sum_{i=1}^{n} D_i*(n-i+1)\}/\{(n*(n+1))/2\} \quad (6)$$

Each element in the sequence D represents a figure of merit which is the difference between two differences of mean times. These differences should have stable and significant values if the computation was done using correct key bits [2]. However, if the key contains wrong bit(s), the elements of the sequence D that correspond to the wrong bits in the key will apparently have lower values as compared with that corresponds to correct bits. In other words, the element D_i gives a sort of evaluation to the key bit k_i. D_i should maintain a stable high value for i=1, 2, .., n if its corresponding key bit is correct. To explain this in more details, let us suppose we have a wrong bit k_e at position e. this implies that the average of D_1, D_2, .., D_{e-1} must be greater than the average values D_e, D_{e+1}, .., D_n. This means that, the error bit position clearly effects the over all key evaluation. It is possible to use the value μ(D) to give a good indication about the key fitness. In equation (6) further improvement is applied. In this improvement we give a weight value for each key bit evaluation according to its position within the key (i.e. according to it's effect on the over all key evaluation). If the error occurred at most significant bits of the key, this will be resulting in completely wrong key, so that the error in most significant bits must have higher weight than the least significant bits. In equation (6), D_1 has the weight n, D_2 has the weight n-1, and so on until D_n which has the weight of 1.

4.2. Crossover and Other Operation

In order to have a good application of genetic algorithm, the crossover operation should be able to produce highly evaluated individual keys from good evaluated ones, with high probability. In present work, two individual keys are selected by using roulette wheel selection method and subject to crossover. In this process, a random swap point is generated. Swap point is a number whose value between 1 and the maximum length of individual key (number of bits). Then, a swapping operation is done, in which the first part of the first key (before the swap point) is swapped with first part of second key. By this way new two individual keys are created.

As the average of population fitness goes greater, the position of swap point must be shifted toward the least significant bits. The idea behind this is that the growth of population fitness means that each individual should has a considerable number of correct bits. So, that shifting of swap point is to avoid changing the correct part of individual key and also to spread it among other keys.

Falling in local minimum is expected but this can be detected by observing the population fitness or individuals evaluation through a number of successive populations. To cope with such situations, mutation rate must be increased automatically. This will help to create new features (new bits segments) that may get the population through this local area search space. In normal situation, we should have a small mutation value, while the high mutation rate may result in disfiguring the good individuals.

5. Practical Results

The proposed attack system was implemented by using Visual C++ ver. 6.0 language on P III, 700 MHz PC working under windows 2000 operating system.

Samples of messages were generated and all time measurements were done by using VC++ program. These messages were signed by number of keys of 64 bits length. Initial experiments have shown encouraging results for key disclosure. Figure 4 shows the average of population fitness over number of generations. Current results have led to 96% correct key disclosure by using only 1500 messages as maximum and 100 individuals in population. These results can be considered as a serious improvement when compared with previously published results.

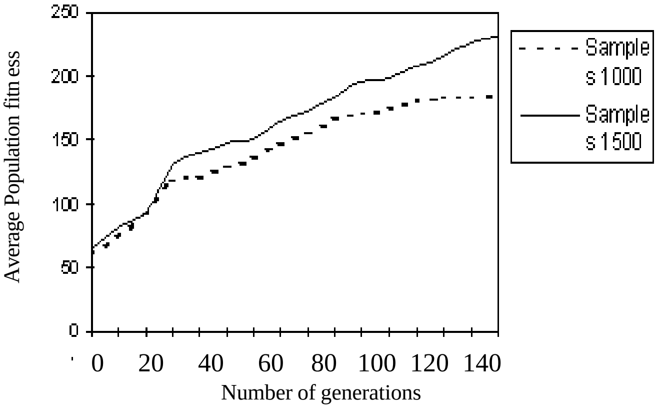


Figure 4. Population fitness

6. Discussion and Conclusion

The threat against cryptosystem implementation using timing attack is greatly increased when less message samples are required. Besides not many details about the system implementation are urgently required to be known, except the knowledge that it uses modular exponentiation process.

This process might fail in attacking systems having some kind of blinding techniques, where some operations are added, such as subtraction or multiplication operations that upsets the Montgomery algorithm for modular exponentiation. The timing attack has also a weakness of not being able to work against the Chinese Remainder Theorem for modular exponentiation.

However, the timing attack suggested here is of general nature in the sense that it may be used efficiently for attacking other similar systems, that implement modular multiplication and exponentiation schemes, such as digital signature standard, DSS or any variant of DSA. More investigation will be pursued is required to implement this speed up error searching technique on a practically existing system.

References

[1] Dhem J. F., "Design of an Efficient Public-Kcryptographic Library for RISC-Based Smart Cards," PhD Thesis, *Universite Catholique de Louvain-UCL Crypto Group-Laboratoire de Microelectronique* (DICE), May 1998.

[2] Dhem J. F., Koeune F., Leroux P. A., Mestre P., Quisquater J. J., and Willems J. L., "A Practical Implementation of Timing Attack," *Technical Report Series, Universite Catholique de Louvain-ULC Crypto Group,* 1998.

[3] Haches G., Koeume F., and Quisquater J. J., “Timing Attack: What Can be Achieved by a Powerful Adversary,” *Technical Report*, *Universite Catholique de Louvain-UCL Crypto Group*, URL: http://www.dice.ucl.ac.be/crypto, 1999.

[4] Kocher P. C., “Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems,” in Koblitz N. (Ed), *Advances in Cryptology (CRYPTO’96)*, Santa Barbara, California, Springer, vol. 1109 of *LNCS*, pp. 104-113, 1996.

[5] Kocher P. C., Jaffe J., and Jun B., “Introduction to Differential Power Analysis and Related Attacks,” http://www.cryptography.com/ dpa/, 1998.

[6] Koeune F. and Quisquater J. J., “A Timing Attack Against Rijndael,” *Technical Report*, *Universite Catholique de Louvain-UCL Crypto Group*, 1999.

[7] Kolodziejczyk J., “The Application of Genetic Algorithm in Cryptoanalysis of Knapsack Cipher,” *http:// www.cryptography. com/ dpa…/, 1998*.

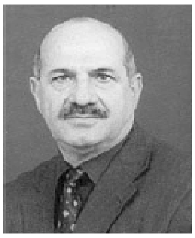
[8] Liepins G. E. and Hillard M. R., “Genetic Algorithms: Foundations and Applications,” *Annals of Operations Research*, vol. 21, pp. 31-58, 1989.

[9] Matthew R. J., “The Use of Genetic Algorithms in Cryptanalysis,” *Cryptologia*, vol. 17, no. 2, pp. 187-201, 1993.

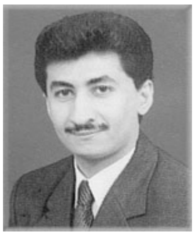
[10] Rivest R., Shamir A., and Adlemen L., “Method for Obtaining Digital Signature and Public Key Cryptosystem,” *Communications of ACM*, vol. 21, no. 2, pp. 120-126, 1978.

[11] Schneier B., “Applied Cryptography,” John Wiley & Sons, 1996.

[12] Spillman R., Janssen M., Nelson B., and Kepner M., “Use of a Genetic Algorithm in the Cryptanalysis of Simple Substitution Cipher,” *Cryptologia*, vol. 17, no.1, 1993.



Hamza Ali BSc in physics from Basrah University 1968, MSc in electronics from University of London 1973 and PhD in computer engineering from University of London, 1977. Previously he worked in Basrah University, Iraq, Shatt Alarab University College, Iraq, Aizu University, Japan, and currently associate professor at the Computer Science Department, Zarka Private University, Jordan. His research interest includes computer data and networks security, authentication and digital signature, recognition of Arabic script and neural networks.



Mikdam Al-Salami obtained his MSc and BSc in computer science from Basrah University, Iraq, 1996 and 1999, respectively. Previously he worked at Ar-Rafidain University College, Iraq. Currently, he is a lecturer at the Computer Science Department, Zarka Private University, Jordan. His research areas include cryptography in computer systems, authentication and management of data protection.