TRANSACTIONS ON SCIENCE AND TECHNOLOGY

Cracking of the Merkle–Hellman Cryptosystem Using Genetic Algorithm

Zurab Kochladze[1]* & Lali Beselia[2]

[1]Ivane Javakhishvili Tbilisi State University, 1, I.Chavchavadze av 1, 0128, Tbilisi, Georgia
[2]Sokhumi State University, 9, Anna Politkovskaya Street, 0186, Tbilisi, Georgia
*Corresponding author. E-Mail: zurab.kochladze62@gmail.com; Tel: +995 32 2 224712; Fax: +995 32 2 223874

Received: *30 March 2016*
Revised: *1 June 2016*
Accepted: *21 June 2016*
Online: *30 June 2016*

Keywords:
Genetic algorithm; cryptanalysis; Merkle-Hellman cryptosystem; Shamir's algorithm; Cryptanalytic attack

Abstract
The article considers the possibility of using genetic algorithms in cryptanalysis, namely for cracking the Merkle-Hellman cryptosystem. The obtained analysis results lead us to conclusion that the use of genetic algorithms in cryptanalysis may be effective. For example, the genetic algorithm described in the article finds a cipher key faster than the well-known Shamir algorithm.

Introduction

In 1978 the famous Merkle & Hellman (1978) article was published, which described an open key (asymmetric) cryptosystem based on a concrete case of the knapsack problem (Martello & Toth, 1990). We can formulate it as follows: there is a knapsack of V volume and a set of $B=\{b_1,b_2,...b_n\}$ subjects, which have certain volumes. Our goal is to find such $B_i \subseteq B$ subset of B set, for the elements of which the following equation is worked out:

$$V=\sum_{i=1}^{n} b_i \cdot x_i \qquad (1)$$

where $x_i \in \{0,1\}$, $i=1,2,...,n$ in case $x_i=1$. This means that we should place i subject into the knapsack, and in case $x_i=0$-, then we should not place the subject into the knapsack. As it is known, the knapsack problem belongs to the NP grade task groups (Martello & Toth, 1990). However, in this concrete case, if B set is an extremely increasing sequence, i.e. each b_i member of the sequence fulfils the condition.

$$b_i > \sum_{j=1}^{i-1} b_j, \qquad (2)$$

Then there is a linear algorithm for the solution of the problem (Martello & Toth, 1990).

Using this feature, Merkle and Hellman (1978) developed an open key cryptosystem, in which the open cipher key is $A=\{a_1,a_2,...a_n\}$ non-extremely increasing sequence, where each a_i member of A sequence is obtained by the following rule:

$$a_i = b_i \cdot t(\mathrm{mod}\, m), \tag{3}$$

where $m,t \in Z$ and the following conditions are fulfilled:

$$m > \sum_{i=1}^{n} b_i\ ,\ (t,m)=1. \tag{4}$$

The key of the cipher is the (B,m,t) three. The open text, which represents a sequence of zeroes and ones, during encryption is divided into a block of n length and L quantity and acts as $x_i \in \{0,1\}$ set. The encrypted text is represented with $S_1,S_2,...,S_L$ sums, which are calculated by the formula:

$$S_j = \sum_{i=1}^{n} x_{ij} \cdot a_i \tag{5}$$

For restoring the open text it is necessary to solve the above mentioned version of the knapsack problem by a linear algorithm when B extremely increasing sequence and m and t parameters are known. For this purpose each sum is multiplied by t^{-1} modulo m

$$S_j^{/} = S_j \cdot t^{-1}(\mathrm{mod}\, m) \tag{6}$$

and the knapsack problem is solved by the above mentioned linear algorithm for each $S_j^{/}$ sum separately, when B extremely increasing sequence is known. In case an opponent desires to crack the system and find the open text he/she will have to solve NP problem that is quite impossible as far as the quantity of the members in B sequence changes from two hundred to three hundred elements.

At a glance this system seemed to be protected from any cyber attacks and was the fastest open key system, the use of which was possible for encryption of vast texts. However, it turned out to have certain trapdoors (Salomaa, 1996), by use of which famous cryptologist A. Shamir (1984) created the polynomial algorithm and cracked the system.

Among the weak points the most noteworthy is that unlike other open key cryptosystems, the open key is obtained from the cipher key not by means of a single-direction function. Moreover, seemingly, it is not at all necessary to find exactly the (t_0,m_0) pair, by which the open key – A non-extremely increasing sequence is obtained from B extremely increasing sequence. As it appears, any B extremely increasing sequence, from which it is possible to obtain the given A non-extremely increasing sequence, may be used as a cipher key, i.e. it is possible to make an attack on the key. Using these trapdoors A. Shamir (1984) created a two-stage algorithm to attack against cryptosystems. In the first stage the algorithm looks for such integers, for which the following

condition is fulfilled: u/m value for several a_i-s is located in the common minimal space of these functions. After finding such numbers by the Diophantine approximation method the algorithm looks for the (u,m) pair, by means of which it becomes possible to calculate the cipher key from the open key.

Using the Genetic algorithm to Crack the Merkle-Hellman cryptosystem

Genetic algorithms were first used for the solution of optimization problems (Goldberg, 1989). After some time they were also used in different fields of sciences. Genetic algorithms are based on one of the basic principles of biological evolution: struggle for population saving by maximal adjustment to the environment that is reached by improvement and development of the best features in new generations. One of the most significant advantages of the genetic algorithms compared to other search algorithms is the possibility of their parallelization. This significantly decreases the attack time. Use of genetic algorithms for cryptanalysis of cryptographic algorithms is a new trend, which has not yet developed in practical cryptology. There are several hundreds of works, the authors of which make their efforts to show that this approach may have advantages compared to other ones (Garg & Shastri, 2006; Garg, *et al.*, 2007; Muthuregunathan, 2009; Tadors *et al.*, 2010; Ramani & Balasubramanian, 2011; Mishra & Bali, 2013). In regard to this we tried to crack the already cracked Merkle-Hellman cryptosystem by means of genetic algorithms. Then we compared the results obtained by us to the results obtained by the Shamir algorithm. There are works where the Merkle-Hellman cryptosystem is cracked by genetic algorithms, though in all cases the attack is carried out on the basis of ciphertext (Spilman, 1993; Spilman *et al.*, 1993; Garg & Shastri, 2006; Garg *et al.*, 2007; Muthuregunathan, 2009; Ramani & Balasubramanian, 2011; Ramani, 2011). Unlike these works and like the Shamir method, we search for the cipher key by attacking against the open key. We elaborated new heuristic methods, by which made the use of genetic algorithms more precise and faster. The research results and software given in this article may be used for cryptanalysis of other asymmetric cryptosystems as well.

The problem formulation and results

Our method of attack is quite different from the methods used in the above mentioned works. Besides, we created a genetic algorithm quite different from other genetic algorithms (different in the selection criterion and crossover process).

Our genetic algorithm is described in file “genetic2.h” created by us. In “genetic” class of the file four functions are described: the fitness function (*bool fitness (vector<populatcia>&v)*), the crossover function (*void crossover (vector<populatcia>&v)*), the mutation function (*void fitness(vector<populatcia>&v*)) and the selection function (*void selektcia(vector<populatcia>&v*)). algorithm is as follows:

1. The fitness function determines the extreme increase in each member (solution-candidate) of the population transmitted to it. The fitness value of the solution-candidate in the sequence is equal to the quantity of the extremely increasing members.
2. The selection function chooses the selection-candidates, which the most fulfill the fitness function, i.e. their fitness values are higher than those of others. In case the population size is L we choose only L/5 solution-candidates. Exactly these solution-candidates form new generations.
3. The crossover function receives the population of the solution-candidates. From this population we choose solution-candidates with t1 and t2 numbers in pairs by means of a random generator taking into account that t1 and t2 do not coincide with each other and the used pair is not repeated. Each solution-candidate is divided in two parts (at the mid point).
4. The mutation function changes one byte of each solution-candidate. We choose the index by the random generator and change the relevant bit, i.e. if the bit value is zero it is changed in 1, and on the contrary, if it is 1, then it is changed in zero.

Our goal is, using the above described algorithm, to find such (u,m) pair, by which we will be able to find the extremely increasing sequence by the following formula:

$$b_i = a_i u (\mathrm{mod}\, m), \text{ where } u=t^{-1} \text{ modulo } m \quad (7)$$

The algorithm is realized on C++ language base. It consists of the preparation and main parts. In the preparation part the information-to-be-transmitted is ciphered by the Merkle-Hellman algorithm. We took $\{b_1, b_2, ...b_n\}$ extremely increasing sequence, m modulo root and selected t multiplier, by means of which we calculated open key $a_i = b_i \cdot t (\mathrm{mod}\, m)$ and ciphered the information-to-be-transmitted by (3) formula algorithm is as follows:

1. The initial population is represented by m root, which is initialized by random generator (it is represented in binary system). The size of each member (solution-candidate) of the population is d*n, where n is equal to the length of the open key and d=2;
   The solution-candidates are transformed into binary system.
2. Like the Shamir algorithm we take the first four members of the open key and calculate the inverse of t multiplier by m root $u = p*m/a_i$, where $u = t^{-1} (\mathrm{mod}\, m),\ 1 \le p \le a_{i,}$ $0 \le i < 4$.Thus, we receive the population all probable multipliers. We set limits for selecting u multiplier. Besides (u,m)=1 and u<m, u multiplier multiplied by the third member of the open key must exceed m root. By adding this limit we reduce the solution-candidates of u multiplier, i.e. make the algorithm more purposeful. We find the relevant closed key by (4) formula for all probable candidates of (u,m) pair.
3. We determine the criterion for selection by the fitness function. In this case the criterion for selection is the extreme increase in the closed key obtained as a result of the fourth phase. In case

the sequence is extremely increasing, i.e. the value of the fitness function is less than n, we pass to the following phase.

4. By means of the crossover function we carry out the crossover operation for the chosen solution-candidates.
5. For the received solution-candidates the second, third and fourth phases are repeated. In case the fitness function of any solution-candidate is equal to n, it means the desired result is obtained and the program stops functioning. Otherwise, we pass to the following phase.
6. The selection function chooses the L/5 (L is the size of the initial population) number solution-candidates, the fitness functions of which are higher.
7. We have indicated that the process will repeat 10 times. If this process is repeated, 10 times and we don't get the desired result, only in this case we use a mutation, or change the function of the gene, and then repeat the 2nd, 3rd and 4th steps. When we get the desired results, we stop working. But tests showed that non f the mutations feature is not needed, and the hybridization of a maximum of 5 times using we get the desired result.

Table 1 shows the result obtained in the experiment.

Table 1. The results of the experiments.

No	Knapsack sequence size	Population size	The number of experiments carried out	Repeating the average number of genetic operators (crossover operation)	Average execution time
1.	8	10	10	2	3.41st
2.	16	20	5	3	4.52st
3.	16	30	5	4	7.41 st
4.	20	50	5	5	8.71 st

Discussion and Conclusion

According to the experiment results, it is obvious that by using genetic algorithm, the Merkle-Hellman cryptosystem is cracked quite quickly. Experiments show that our algorithm is almost twice faster than Shamir (1984) algorithm. Therefore, we can conclude that attacking of the open key cryptographic systems through the genetic algorithms can be done only on the basis of public key, even in cases when the polynomial algorithms of the attack are unknown.

Reference

[1] Goldberg, D. E. (1989). *Genetic Algorithm in Search, Optimization and Machine Learning*. Boston: Addison-Wesley Longman Publishing.
[2] Garg, P. & Shastri, A. (2006). An Improved Cryptanalytic Attack on Knapsack Cipher using Genetic Algorithm. *International Journal of Information Technology*, **3**(3), 145-152.
[3] Garg, P., Shastri, A. & Agarwal, D. C. (2007). An Enhanced Cryptanalytic Attack on Knapsack Cipher Using Genetic Algorithm. *Word Academy of Science, Engineering and Technology*, **12**, 829-832.
[4] Li, T., Li, J., & Zhang, J. (2014). A Cryptanalysis Method Based on Niche Genetic Algorithm. *Applied Mathematics & information Sciences. An International Journal*, **8**(1), 279-285.
[5] Martello, S. & Toth, P. (1990). *Knapsack Problems: Algorithms and Computer Implementations.* New York: John Wiley & Sons.
[6] Merkle, R. & Hellman, M. (1978). Hidding information and signatures in trapdoor Knapsak. *IEEE Transactions on Information Theory*, **24**(5), 535-530.

[7] Mishra, S. & Bali, S. (2013). Public Key Cryptography Using Genetic Algorithm. *International Journal of Recent Technology and Engineering*. **2**(2), 150-154.
[8] Muthuregunathan, R., Vekataraman, D. & Rajasekaran, P. (2009). Cryptanalysis of Knapsack Cipher Using Parrallel Evolutionary Computing. *International Journal of Recent Trends in Engineering*, **1**(1), 260-263.
[9] Ramani, G. & Balasubramanian, L. (2011). Genetic Algorithm Solution for Cryptanalysis of Knapsack Cipher with Knapsack Sequence of Size 16. *International Journal of Computer Applications*, **35**(11), 17-23.
[10] Ramani, G. (2011). Genetic Algorithm solution for Cryptanalysis of Knapsack Cipher with Knapsack Sequence of Size 16. *International Journal of Computer Applications*, **35**(11), 17-23.
[11] Salomaa, A. (1996). *Public-Key Cryptography* (2nd Enlarged Edition). Berlin Heidelberg: Springer-Verlag.
[12] Shamir, A. (1984). A Polinomial-Time Algorithm for Breaking the Basic Merkle-Hellman Cryptosystem. *IEEE Transactions on Information Theory*, **30**(5), 699-704.
[13] Spilman, R. (1993). Cryptanalysis of Knapsack Cipher Using Genetic Algorithms. *Cryptologia*, **17**(4), 367-377.
[14] Spilman, R., Janssen, M., Nelson, B. & Kepner, M. (1993). Use of a Cenetic algorithm in the Cryptanalysis of Simple Substitution Ciphers. *Cryptologia*, **17**(1), 31-44.
[15] Tadors, T., Hegazy, A. E. F. & Bard, A. (2010). Genetic Algorithm for DES Cryptanalysis. *International Journal of Computer Science and Network Security*, **10**(5), 5-11.