Cryptanalytic Results on Knapsack Cryptosystem Using Binary Particle Swarm Optimization

Conference Paper *in* Advances in Intelligent Systems and Computing · June 2014

DOI: 10.1007/978-3-319-07995-0_37

1 author:

Ashish Jain
Manipal University Jaipur
20 PUBLICATIONS **149** CITATIONS

SEE PROFILE

Cryptanalytic Results on Knapsack Cryptosystem Using Binary Particle Swarm Optimization

Ashish Jain and Narendra S. Chaudhari

Discipline of Computer Science and Engineering
Indian Institute of Technology Indore, India
{phd11120101,nsc}@iiti.ac.in, ashishjn.research@gmail.com

Abstract. The security of most Public Key Cryptosystem (PKC) proposed in literature relies on the difficulty of the integer factorization problem or discrete logarithm problem. However, using shor's [19] algorithm the problems can be solved in acceptable amount of time via 'quantum computers'. Therefore in this context knapsack (more accurately subset sum problem(SSP)) based PKC is reconsidered as a viable option by the cryptography community. However, before considering the practicability of this cryptosystem, there is a growing need to cryptanalyze it using all possible present techniques, in order to guarantee their security. We believe that modern Computation Intelligence (CI) techniques can provide efficient cryptanalytic results (because of the new aspects have been incorporated in CI techniques). In this paper, we use two different binary particle swarm optimization techniques to cryptanalyze knapsack PKC. The results obtained via extensive testing are promising and proficient. We present, discuss and compare the effectiveness of the proposed work in the result section.

Keywords: Cryptanalysis of Knapsack Cryptosystem, Binary Particle Swarm Optimization (BPSO), Modified Binary Particle Swarm Optimization (MBPSO), CI, Merkle-Hellman (MH).

1 Introduction

The trapdoor knapsack used for hiding information and signature is a knapsack-based cryptosystem, first proposed by Merkle and Hellman [14] in 1978. This public key encryption proposal has been thoroughly investigated owing to a high computational efficiency (at that time). The motivation of its design is to convert a superincreasing knapsack sequence into a computationally hard sequence. Though, the basic version was broken by Shamir in 1984 [17] by exploiting the special structure in the sequence of the knapsack. The basic tool for analysis was sawtooth curves (function of $Va_i(mod\ m)$). At present, the most significant challenge towards knapsack cryptosystems are lattice attacks. Lagarias-Odlyzko [12] attack can solve knapsacks with density d $<$ 0.64. The Lagarias-Odlyzko attack was further improved by Coster et al. [2] to knapsack densities of d $<$ 0.94. However, an acceptably dense knapsack cryptosystem is also broken using

J.G. de la Puerta et al. (eds.), *International Joint Conference SOCO'14-CISIS'14-ICEUTE'14*, Advances in Intelligent Systems and Computing 299,
DOI: 10.1007/978-3-319-07995-0_37,

lattice attack, recently proposed by Shang-Ming Jen et al. [9]. *'They showed (experimentally) that only density consideration is not the sole criteria to assess knapsack cryptosystem security'*. In last few years several extended and improved SSP-based cryptosystem have been reported in literature by Wang et al. [22] in 2007, Murakami et al. [16] in 2008, lyubashevsky et al. [13] in 2010, Wang and Yupu [21] in 2010, kate et al. [10] in 2011 and recently by murakami et al. [15] in 2012. However, security hardness of such cryptosystems must be analyzed with new and improved techniques. In this paper, we are using binary particle swarm optimization techniques (swarm intelligence based CI techniques) for verifying the security of the *'Basic MH Cryptosystem'*[1]. CI methods are very powerful for searching and optimization that also has a huge potential for cryptanalysis. We believe that CI techniques can be used to break cryptosystem or to improve parts of the attack done by other method.

1.1 Motivation

Computational intelligence is a well-established paradigm, where new theories with a sound nature-inspired metaphors have been proposed. The current experimental systems have many of the characteristics of intelligent system to perform a variety of tasks, often considered difficult with conventional algorithms. An effective use of computational intelligence can be seen in [7] and [8] which addresses real time intrusion detection for network security and neural-visualization ids for honeynet data respectively. In addition of this recently in 2012, Danziger et al. [3] reported in there captioned paper "Computational Intelligence Applied on Cryptology: A Brief Review" that only a few work (quite initial) has been done on modern cryptographic systems using CI techniques. In last ten years (2000 to 2010) the number of CI applications in cryptology[2] decreased. The reason is difficulties in representing the problem in terms of CI and poor interaction between researchers of cryptology and CI. But, new concepts and ideas emerged in cryptology that can be used with CI tools owing to the availability of computational and processing capabilities. Typically, Artificial Neural Networks (ANN), Cellular Automata (CA) and (DNA) are applied to develop new cryptographic systems. While, Evolutionary Computation (EC), Swarm Intelligence (SI) and (DNA) are applied to analyze cryptographic systems [3].

1.2 Our Contribution

In this paper, we compare the results of cryptanalysis found via rigorous experimentation (recovering plaintext from given ciphertext) using BPSO technique proposed by Kennedy et al. [11] and MBPSO technique proposed by Bansal et al.[1]. An appropriate fitness function is introduced to achieve optimal results. In addition to this, during experimentation we fine tunned the inertia weight and maximum velocity of the particles. To the best of our knowledge, this is the

[1] Since the core concept behind SSP-based cryptosystems is MH cryptosystem.
[2] Cryptology: A study of two fields - cryptography and cryptanalysis.

first paper, reporting cryptanalysis on the original parameter of basic MH cryptosystem using particle swarm optimization techniques. The remainder of the paper is organized as follows: In the next section, knapsack cryptosystem is described. BPSO and MBPSO methods with respect to cryptanalysis of knapsack PKC is described in Sec. 3. Comparison of Genetic Algorithm (GA) and Particle Swarm Optimization (PSO) methods is discussed in Sec. 4. Experimental setup and results obtained via BPSO and MBPSO is presented in Sec. 5. Followed by conclusion and future work in Sec. 6.

2 Description of the Knapsack PKC

In any PKC system, public key is publicized by the designer (e.g. by Alice), so that using public key sender (e.g. Bob) encrypts the plaintext and sends it over a (insecure) communication channel. Upon receiving the encrypted message (i.e. ciphertext 'b'), Alice decrypts b using her own private key (a key that is used by the designer to generate the public key). In case of MH knapsack-based PKC, the hard knapsack $A = a_1, a_2, ..., a_n$ (set of natural numbers) is publicized as the public key (a typical value of n is 100 and length of a_i increases from 100 bits to 200 bits with the increase in i). Let, the sender have a message of length n as a bit string or simply $X = x_1, x_2, ..., x_n$ ($x_i \in \{0, 1\}$). The sender first compute the sum b ($b = \sum_{i=1}^{n} a_i * x_i$) and then sends it via the public channel. Let, both the receiver and the potential eavesdropper knows vector A and b. Their task is to find which subsets of the a_i sums up to b. This is an instance of the knapsack problem, which is known to be nondeterministic polynomial time complete (NP-Complete). This problem is difficult for the eavesdropper but easy for the receiver because she have the private key ($A^{'}$, m, and $w^{'}$) i.e. easy knapsack, modular m and inverse of multiplier (w) in mod m arithmetic respectively. Now, we take an example; Let $A^{'}$=[1, 3, 7, 13, 26, 65, 119, 267], m=65423, w=21031 and $w^{'}$=5363. Then A=[21031, 63093, 16371, 11711, 23422, 58555, 16615, 54322], where $a_i = a_i^{'} \times w(mod\ m)$. If Bob want to send a letter 'M' (ASCII CODE: 10110010, LSB...MSB). He will encrypt '10110010' using A, generate b=65728 ('21031+16371+11711+16615') and then send to Alice.To decrypt, Alice multiplies b by $w^{'} mod\ m$ (65528*5363 mod 65423 = 140). Now, Alice decomposes 140 using $A^{'}$ as an example 140-119=21, 21-13=8, 8-7=1, 1-1=0. The elements we selected from $A^{'}$ corrospondes to the 1 bits in the message i.e. '10110010' recovered.

3 PSO Methods and Cryptanalysis of Knapsack PKC

3.1 Binary Particle Swarm Optimization (BPSO)

A swarm is initialized with a population of particles. At each iteration all particles move in a search space to find the optimal solution (in case of cryptanalysis exact solution). The position vector $X_i = (x_{i1}, x_{i2}, ..., x_{in})$[3] and the velocity

[3] Each generated position vector is logically equivalent to X(defined in Sec. 2) as a possible solution.

CHARACTER	PLAINTEXT (ASCII CODE)	CIPHERTEXT (or *TARGET*)
M	10110010	65728
A	10000010	37646
C	11000010	100739
R	01001010	103130
O	11110010	128821

Fig. 1. Plaintext and Encrypted Ciphertext

vector $V_i = (v_{i1}, v_{i2}, ..., v_{in})$ is associated with each particle for guiding their movements. Here, $x_{ij} \in (0,1)$, $i = 1, 2, ..., m$ (m represents, number of particles) and $j = 1, 2, ..., n$ (each particle represents a potential solution in the n-dimensional space); v_{ij} represents the velocity of the j^{th} element of the i^{th} particle, constrained by V_{max}. Since in contrast to the real-PSO in binary-PSO small V_{max} allows a higher mutation rate [11]. In the experiments, we fixed V_{max}=2 (for details see Sec. 3.3). Now, for the cryptanalysis of 0/1 knapsack cryptosystem, particle position vector and velocity vector is represented by a binary string of length n. where, n is the total number of elements in published public key. During each iteration the following equations are employed to update particle's position and velocity.

$$v_{ij}(t+1) = w \times v_{ij}(t) + c_1 r_1 (lbest_{ij}(t) - x_{ij}(t)) + c_2 r_2 (gbest_j(t) - x_{ij}(t)) \quad (1)$$

$$Sig(v_{ij}(t+1)) = \frac{1}{1 + e^{-v_{ij}(t+1)}} \quad (2)$$

$$x_{ij}(t+1) = \begin{cases} 1, & \text{if } Sig(v_{ij}(t+1)) > U(0,1) \\ 0, & \text{otherwise} \end{cases} \quad (3)$$

where, w is the inertia weight initialized with 0.9 and decreased after each iteration using equation-5 (for details see Sec. 3.3). c_1 and c_2 are the acceleration constants, we set them to a standard value (2.05). r_1 and r_2 are random variables in the interval [0,1] obtained from a uniform distribution U(0,1). The best previous position of the i^{th} particle and the global best position of all particles (i.e. swarm) is represented by $Lbest_i(t) = (lbest_{i1}, lbest_{i2}, ..., lbest_{in})$ and $Gbest(t) = (gbest_1(t), gbest_2(t), ..., gbest_n(t))$, respectively. A fitness function is used to assess both, $Lbest_i$(t) and $Gbest$(t). $Sig(v_{ij}(t+1)$ is a sigmoid function, that is used to transform velocity in the interval [0,1].[4]

3.2 Modified Binary Particle Swarm Optimization (MBPSO)

In this method, a significant improvement to update the position of each particle is proposed by Bansal et al. [1]. The motivation came from the position update

[4] The importance of velocity in binary space can be better understand as changes in probability of a particle position. If v_{ij}=0.4 it means there is a 40% chance that x_{ij} will take a value 'one' and 60% chance that it will take a value 'zero'.

Table 1. Comparison of Maximum Velocity to Select Efficient V_{max} for Cryptanalysis

Value of Vmax	*BPSO*		*MBPSO*	
	ASS(%)	**SR**	**ASS(%)**	**SR**
1	19.61	75%	19.34	75%
2	13.03	80%	10.51	85%
3	18.17	70%	11.58	75%
4	17.80	60%	15.50	80%

equation ($x_{ij}(t+1) = x_{ij}(t) + v_{ij}(t+1)$) used in real-PSO. Since the velocity is limited in the range [-V_{max}, V_{max}] and x_{ij} can take either 0 or 1. Therefore the term $x_{ij}(t) + v_{ij}(t+1)$ can be bounded between (0 - V_{max}) and (1 + V_{max}) by transforming it into ($\frac{x_{ij}(t)+v_{ij}(t+1)+V_{max}}{1+2V_{max}}$). As a result, the position update equation is given as follows. For details interested reader is referred to [1].

$$x_{ij}(t+1) = \begin{cases} 1, & \text{if } \frac{x_{ij}(t)+v_{ij}(t+1)+V_{max}}{1+2V_{max}} > U(0,1) \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

3.3 Maximum Velocity and Inertia Weight

The granularity of the search can be controlled by clamping escalating velocities, in this context, the value of V_{max} is extremely important. The global exploration is possible by allowing large values of V_{max}, although local exploitation is encouraged using small values. If V_{max} is too large, there is the possibility that swarm may skip good regions and continue to search in fruitless regions of the state space. On the other hand, if value of V_{max} is too small, all the particles may not explore outside locally good regions and may become trapped in local optimum [4]. Therefore, by limiting the value of V_{max}, BPSO and MBPSO has more chance to find the optimal solution. We conducted multiple experiments with different settings of knapsack size(=10,20,30 and 40) and V_{max}(=1,2,3 and 4). An experimental result (average of 20 runs) with knapsack size=20 is shown in Table 1. We can observe that with the setting of V_{max}=2, the percentage of Average Search Space(ASS) is less and Success Rate(SR) is high. In this way we found that V_{max} value should be 2 for gaining the advantage during cryptanalysis of the knapsack cryptosystem. So, we set V_{max}=2 and limited the velocity in the range [-2,2]. The concept of the inertia weight was introduced by Shi and Eberhart [18] to control the exploration and exploitation abilities of the swarm. In our experiments, w is initialized with 0.9 and deceases linearly[5] after each iter-

[5] In starting 60% of total iterations, w has been decreased with delta(=0.9-0.4/$MaxIterations$) and then for remaining iterations we kept it remain constant.

ation (using equation-5). So that the particles spent less time in exploration and more time in exploiting the solution. In equation-5, $w(init)$=0.9, $w(fin)$=0.4.

$$w = \frac{(w(init) - w(fin)) \times (MaxIterations - IterationNum(t))}{(MaxIterations)} + w(fin) \quad (5)$$

3.4 Fitness Function

Since the attacker can reveal the public key and the ciphertext (man in the middle attack). Therefore in such type of automatic attacks, the fitness function can only be designed using $Target$(ciphertext) and $TotalSum$ (sum of all elements of the public key). In this context, we analyzed the fitness function that was suggested by spillman[20] and realize that the concept of normal probability distribution (since the maximum fitness value can be 1) among the fitness of all particles can be employed. The main concern in the design of fitness function is, at what amount, the term '$Difference$' (see equation-6) should be normalized so that it best describes the fitness of investigated Sum to the expected $Target$. After several experimental runs on small relevant data, we found that the 'power term' to be used over '$Difference$' should be either 0.5 or 0.6 to normalize it appropriately.[6] Finally, we chose the following fitness function because it is giving optimal results for cryptanalysis of the knapsack cryptosystem.

$$Difference = \frac{(|Target - Sum|)}{max(TotalSum - Target, Target)} \quad (6)$$

$$Fitness = 1 - \sqrt{Difference} \quad (7)$$

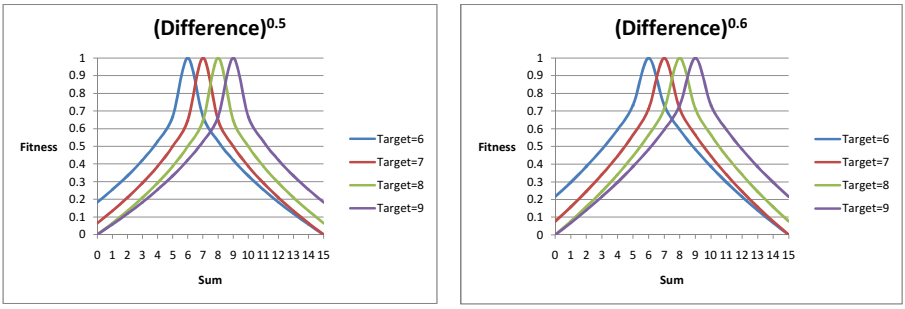

Fig. 2. Analysis of Fitness Function

[6] Analytical result is shown in Fig.2 (we tested the 'power term'(0.4 to 1) but normally distributed curve is obtained via 0.5 and 0.6, therefore presented in Fig.2), we can observed that the obtained curve provides normal distribution of probability.

4 Related Work: GA Applied to Knapsack Cryptosystem and Comparison with BPSO and MBPSO

During literature survey we found few works that has been reported in concern with cryptanalysis of basic MH cryptosystem using GA. In 1993, Spillman [20] has given a GA for the cryptanalysis of the knapsack cryptosystem. His method was further enhanced in [6] and improved in [5] by Garg et al. in 2006. In brief the functionality of Spillman algorithm and Garg et al. algorithm is as follows: The input to the algorithm (see Algorithm 1) is the ciphertext b and the public key vector A. The task of the algorithm is to recover the plaintext (i.e. ASCII code of message) from the known ciphertext with the help of metaheuristic evolutionary approach. Since it is not always possible to recover exact plaintext. Therefore the termination condition to the algorithm is set to a fixed number of iterations (generations) or a best fitness (i.e. '1') is achieved. As a result, the output to the algorithm is either an exact solution (a single chromosome) or a set of best solutions (chromosomes with fitness nearly '1'). Spillman suggested to exploit the set of best chromosomes manually to decipher the ciphertext. While Garg et al. proposed an Improved GA (IGA) in which the genetic operators (Crossover and Mutation) are fine tunned so that the best fittest chromosomes explore properly in the search space and the high probability to get the exact solution. However there are some limitations of GA methods used in cryptanalysis of knapsack cryptosystem. In general, GA methods taking more computational time while the success rate is lower than PSO methods. Moreover, none of the authors presented results on the original parameter[7] of the knapsack cryptosystem that was proposed by Merkle and Hellman to maintain the security and speed over a public channel. The fitness function used in [20,6,5] are also not appropriate. However, for the comparison purpose we ran our proposed BPSO and MBPSO algorithm on the same example (see Fig. 1). As shown in Table 2, results shows that PSO methods perform much better than IGA. Because Average Number of Iterations (ANI) and Average Searched Space (ASS) taken by BPSO and MBPSO is very less than as required by IGA.

Algorithm 1. Pseudocode of GA

repeat
A random population of chromosomes (represented as binary string) is generated.

Evaluate the fitness of each chromosome.
Select the best fitness chromosomes.
Apply crossover operator to each pair of selected chromosomes.
Apply mutation operator to candidates found in previous step.
Next generation population is scanned to update a list of "best" chromosomes.
until "stopping condition is true"

[7] Original Parameters: number of elements (a_i) in a public vector=100 and the length of a_i increases from 100 bits to 200 bits with the increase in i.

Table 2. Comparison of IGA with proposed BPSO and MBPSO Methods

CHARACTER	*IGA*			*BPSO*			*MBPSO*		
	ANI	**ASS(%)**	**SR**	**ANI**	**ASS(%)**	**SR**	**ANI**	**ASS(%)**	**SR**
M	17.00	35.3	100%	36.6	20.94	100%	40.7	14.76	100%
A	112.00	49.1	100%	9.4	19.62	100%	14.3	21.21	100%
C	271.00	63.6	100%	59.5	50.45	100%	42.6	38.9	100%
R	89.40	50.7	100%	65.1	37.13	100%	46.9	24.6	100%
O	87.20	40.6	100%	96.3	57.39	100%	78.4	50.69	100%
Avg. Sum	115.32	47.86	100%	53.38	37.106	100%	44.58	30.032	100%

Table 3. Comparison of Cryptanalytic Results obtained by BPSO and MBPSO Method

BPSO				*MBPSO*			
NOE	**AFE**	**ASS**	**SR**	**NOE**	**AFE**	**ASS**	**SR**
10	3053	$2^{8.51}$	100%	10	1987	$2^{8.43}$	100%
15	19135	$2^{12.18}$	98%	15	18478	$2^{11.87}$	96%
20	536443	$2^{16.33}$	78%	20	496560	$2^{15.98}$	84%
25	1596694	$2^{19.59}$	72%	25	1287587	$2^{18.47}$	78%
30	12562002	$2^{23.35}$	48%	30	1999439	$2^{20.83}$	66%
35	18481231	$2^{23.83}$	34%	35	2186184	$2^{21.03}$	48%
40	19656721	$2^{25.98}$	26%	40	17536068	$2^{23.57}$	44%

5 Experimental Setup and Results

Methods described in Sec. 3 are implemented in Java 2.0 (an equivalent algorithm is shown in Fig. 3) on a Intel Quad-Core processor *i*7 (@3.40Ghz with 32 GB RAM) and tested on $n(=40)$ with increasing length (100 bits-200 bits) of elements of public key, as described in Sec. 2. In the experiments, NoI (Number of Iterations) are limited to 35 times to the sizes of public key. However, the limitation on NoP (Number of Particles) is different for each of the different range of public key sizes. The choices on NoP are; 50 times for (10,15), 100 times for (20), 400 times for (25,30) and 800 times for (35,40). Afterwards, both algorithms ran 50 times for 10 different subset sums[8] and obtained results are presented in the Table 3. MBPSO achieves higher Success Rate (SR) than BPSO while the Average Searched Space (ASS) is less (except NOE (Number of Elements) or knapsack size = 15). Moreover Average Function Evaluation (AFE) in case of MBPSO is much less than BPSO. This study shows that the MBPSO performs better than the BPSO. In this way, if a significant amount of the plaintext is recovered from ciphertext, the whole message can be revealed.

[8] We ran the both algorithm on 10 different subset sums for each size of public-key. However for comparison we chose an average case of cryptanalysis.

Input: Public-Key Vector and Ciphertext
Initialize: *m* particles with *n*-dimensions. Here n is the size of Public-Key Vector
repeat
for (each particle *i = 1, 2, . . ., m*) **do**
'Evaluate' fitness value using equation-7;
if (the fitness value is better than '*Lbest'* (the best fitness value in history))
then 'Set' the corresponding particle position as the new '*Lbest* ;
end for
'Select' the particle with the best fitness value of the swarm as *Gbest*;
for (each particle *i = 1, 2, . . ., m*) **do**
'Update' particle velocity according to equation-1;
if (BPSO) **then** 'Update' particle position according to equation-3;
if (MBPSO) **then** 'Update' particle position according to equation-4;
end for
until ("stopping condition is true"**)**

Fig. 3. Pseudocode of BPSO and MBPSO

6 Conclusion and Future Work

In Sec. 4, we show that BPSO and MBPSO performs much better than IGA. Afterwards, we applied BPSO and MBPSO on MH cryptosystem (with original parameter setting) and obtained cryptanalytic results presented, discussed and compared. Experimental result shows that the MBPSO method perform better than the BPSO method. The real time taken by MBPSO, in case of 40 NOE $\approx$ 1.5 hour while BPSO taking $\approx$ 2 hour. Hence, the conclusion of the investigation is that the knapsack cryptosystem is not secure (when n=40) with this configuration (the computational machine used for cryptanalysis) using BPSO and MBPSO. Moreover the real time attack on the knapsack cryptosystem can become much faster with the use of a much faster computational machine e.g. Intel X eon processor *E*7 (v2, @3.40Ghz with 37.5M cache and 512 GB RAM). Future work includes the improvement to automatic attacks (with the aid of incorporating hybrid approach with presented method e.g. crossover operator with significant probability) in order to achieve higher convergence speed and success probability for n>40. Another future work might be interesting to cryptanalyze RSA algorithm with the help of proposed attack.

References

1. Bansal, J.C., Deep, K.: A modified binary particle swarm optimization for knapsack problems. Applied Mathematics and Computation 218(22), 11,042–11,061 (2012)
2. Coster, M.J., Joux, A., LaMacchia, B.A., Odlyzko, A.M., Schnorr, C.P., Stern, J.: Improved low-density subset sum algorithms. Computational Complexity 2(2), 111–128 (1992)
3. Danziger, M., Henriques, A.: Computational intelligence applied on cryptology: a brief review. IEEE Latin America Transactions (Revista IEEE America Latina) 10(3), 1798–1810 (2012)

4. Engelbrecht, A.P.: Computational intelligence: an introduction. John Wiley & Sons (2007)
5. Garg, P., Shastri, A.: An improved cryptanalytic attack on knapsack cipher using genetic algorithm. International Journal of Information Technology 3(3) (2006)
6. Garg, P., Shastri, A., Agarwal, D.: An enhanced cryptanalytic attack on knapsack cipher using genetic algorithm. Transaction on Engineering, Computing and Technology 12 (2006)
7. Herrero, Á., Navarro, M., Corchado, E., Julián, V.: Rt-movicab-ids: Addressing real-time intrusion detection. Future Generation Computer Systems 29(1), 250–261 (2013)
8. Herrero, A., Zurutuza, U., Corchado, E.: A neural-visualization ids for honeynet data. International Journal of Neural Systems 22(2) (2012)
9. Jen, S.M., Lai, T.L., Lu, C.Y., Yang, J.F.: Knapsack cryptosystems and unreliable reliance on density. In: 2012 IEEE 26th International Conference on Advanced Information Networking and Applications (AINA), pp. 748–754. IEEE (2012)
10. Kate, A., Goldberg, I.: Generalizing cryptosystems based on the subset sum problem. International Journal of Information Security 10(3), 189–199 (2011)
11. Kennedy, J., Eberhart, R.C.: A discrete binary version of the particle swarm algorithm. In: 1997 IEEE International Conference on Systems, Man, and Cybernetics, 1997. Computational Cybernetics and Simulation, vol. 5, pp. 4104–4108. IEEE (1997)
12. Lagarias, J.C., Odlyzko, A.M.: Solving low-density subset sum problems. Journal of the ACM (JACM) 32(1), 229–246 (1985)
13. Lyubashevsky, V., Palacio, A., Segev, G.: Public-key cryptographic primitives provably as secure as subset sum. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 382–400. Springer, Heidelberg (2010)
14. Merkle, R., Hellman, M.: Hiding information and signatures in trapdoor knapsacks. IEEE Transactions on Information Theory 24(5), 525–530 (1978)
15. Murakami, Y., Hamasho, S., Kasahara, M.: A public-key cryptosystem based on decision version of subset sum problem. In: 2012 International Symposium on Information Theory and its Applications (ISITA), pp. 735–739. IEEE (2012)
16. Murakami, Y., Katayanagi, K., Kasahara, M.: A new class of cryptosystems based on chinese remainder theorem. In: International Symposium on Information Theory and Its Applications, ISITA 2008, pp. 1–6. IEEE (2008)
17. Shamir, A.: A polynomial-time algorithm for breaking the basic merkle-hellman cryptosystem. IEEE Transactions on Information Theory 30(5), 699–704 (1984)
18. Shi, Y., Eberhart, R.: A modified particle swarm optimizer. In: The 1998 IEEE International Conference on Evolutionary Computation Proceedings, IEEE World Congress on Computational Intelligence, pp. 69–73. IEEE (1998)
19. Shor, P.W.: Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM Journal on Computing 26(5), 1484–1509 (1997)
20. Spillman, R.: Cryptanalysis of knapsack ciphers using genetic algorithms. Cryptologia 17(4), 367–377 (1993)
21. Wang, B., Hu, Y.: Quadratic compact knapsack public-key cryptosystem. Computers & Mathematics with Applications 59(1), 194–206 (2010)
22. Wang, B., Wu, Q., Hu, Y.: A knapsack-based probabilistic encryption scheme. Information Sciences 177(19), 3981–3994 (2007)