Defence Science Journal, Vol. 62, No. 2, March 2012, pp. 117-121, DOI: 10.14429/dsj.62.778

Cryptanalysis of Simplified-AES using Particle Swarm Optimisation

Vimalathithan R.[*] and M.L.Valarmathi[#]

[*]*KPR Institute of Engineering and Technology, Coimbatore–641 404, India*
[#]*Government College of Technology, Coimbatore–641 013, India*
[*]*E-mail: athivimal@gmail.com*

ABSTRACT

Particle swarm optimisation (PSO) based cryptanalysis has gained much attention due to its fast convergence rate. This paper proposes a PSO-based cryptanalysis scheme for breaking the key employed in simplified-advance encryption standard (S-AES). The cost function is derived using letter frequency analysis. The novelty in our approach is to apply ciphertext-only attack for an S-AES encryption system, where we obtained the key in a minimum search space compared to the Brute-Force attack. Experimental results prove that PSO can be used as an effective tool to attack the key used in S-AES.

Keywords: Cryptanalysis, ciphertext, ciphertext-only attack, cost, particle swarm optimisation, plain text, simplified-AES

1. INTRODUCTION

Cryptography is the study of methods for sending messages in disguised forms, so that only intended recipients can remove the disguise and read the message. The message we want to send is called plaintext and the disguised message is called ciphertext. The process of converting a plaintext into ciphertext is called encryption and the reverse process is called decryption. Cryptography is the art of making cipher text while cryptanalysis is the art of breaking ciphertext. Cryptanalysis is a study of mathematical techniques to defeat cryptographic techniques and attack the cipher text, without accessing the secret key[1].

Cryptanalysis is a challenging task in cryptology. There are several types of attacks that a cryptanalyser may use to break a cipher, depending upon how much information is available to the attacker. The goal is to derive the key, so that the ciphertext can be easily recovered. A brute-force attack is one way of doing so. In this type of attack, the cryptanalyser tries every possible combination of key until the correct key is identified. For lengthy keys, using a network of computers and combining their computational strength and their cumulative power, brute-force attack feasible at increased cost[2-3].

Swarm intelligence is an innovative paradigm used for solving the complicated problems. PSO is a population based optimisation tool which simulates the social behavior of schools of fish or swarms of bee. PSO is used to solve complicated problems. The main strength of PSO is its fast convergence, which compares favorably with many global optimisations like genetic algorithm and simulated annealing.

Simplified-advance encryption standard (S-AES) is a non-feistel cipher that takes a 16 bit plaintext, 16 bit key and generates 16 bit ciphertext. S-AES is used in embedded systems like mobile phones, GPS receivers, which require low memory and low processor capacity[4]. Musa, attacked S-AES for the first time using linear and differential cryptanalysis[5]. Linear and differential cryptanalysis is applied for one round S-AES in this paper. From their results, 109 plaintext and corresponding ciphertext pair were required to apply linear cryptanalysis. This is large in number. If a second round is included in S-AES, even more number of plaintext and corresponding ciphertext pair is required for cryptanalysis. Simmons attacked S-AES using Algebraic cryptanalysis[6]. Here known plaintext attack is used which is easy to attack. More number of plaintext ciphertext pair is required for this type of attack. Davood attacked S-AES against linear cryptanalysis[7]. Linear cryptanalysis is applied using plaintext attack in this paper. To break the first round 116 plain texts were required and 548 plain texts were required to break the second round. Uddin used PSO to attack simple substitution ciphers using ciphertext -only attack[8]. Their results say that PSO is an effective tool for attacking substitution ciphers, though the simple substitution cipher is easy to attack. Shahzad presented a PSO algorithm based approach for the cryptanalysis of DES reduced to four rounds[9]. They compared the results for DES reduced to four rounds and eight rounds using PSO and GA. Known plain text attack is used which is easier to implement because more information is available for analysis. Also the fitness function is suitable only for known plain text attack.

The authors proposed an approach using PSO to attack S-AES for ciphertext-only attack. In case of linear cryptanalysis, number of ciphertext and its corresponding plaintext is known to the attacker. And, this linear cryptanalysis is easy to attack. Our ultimate goal is to take more complicated problem and effectively solve the same by PSO. Hence we take ciphertext-only attack, which is the most complicated task in cryptanalysis. Cryptanalysis of S-AES using PSO have not been reported earlier.

Received 8 March 2011, revised 21 February 2012, online published 13 March 2012

2. SIMPLIFIED–ADVANCE ENCRYPTION STANDARD

2.1 Simplified–Advance Encryption Standard

2.1.1 S-AES Encryption

Simplified–advance encryption standard (S-AES) is a Non-Feistel Cipher that takes a 16 bit plaintext and 16 bit key as input and generates 16 bit ciphertext as output. It uses one pre-round transformation and two round transformations. The 16 bit input plaintext is divided into two by two arrays of nibbles, called state. Each round takes a state and creates another state to be used for the next round. To provide high security, S-AES uses four transformations: Substitution, shift row, mix columns and Add round key. Figure 1 shows encryption algorithm, key generation and decryption algorithm for S-AES.

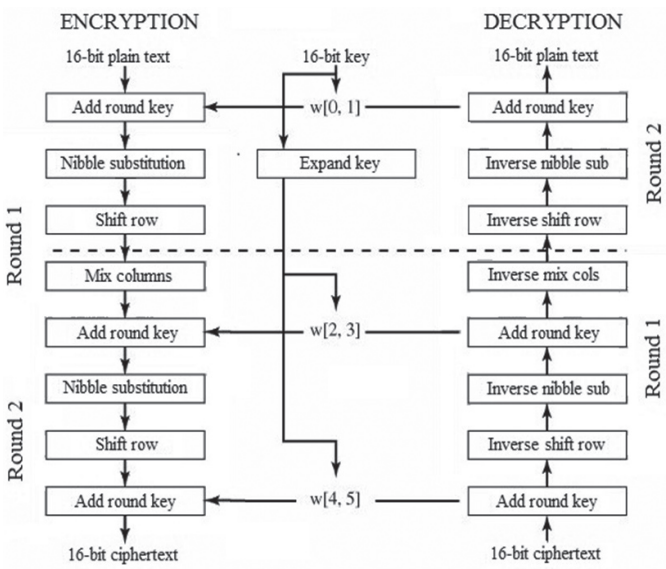

Figure 1. Encryption, key generation and decryption algorithm for simplified-AES.

2.1.2. S-AES Decryption

The decryption algorithm is the reverse process of the encryption. The decryption takes a 16 bit ciphertext and 16 bit key as input and generates 16 bit plaintext as output. This decryption uses one pre-round and two round transformations. Each round takes a state and creates another state to be used for the next round. The pre-round section uses only one transformation called AddRound Key. Round one uses three transformations: Inverse Shift Rows, Inverse Substitution and AddRound Key. The inverse shift row in decryption is the inverse of Shift row which is used in encryption and similarly for inverse substitution. Round two is same as that of round one but includes one additional transformation called inverse mix column which is the inverse of mix column used in the encryption. The output of the Round two is the 16 bit plaintext.

2.1.3. Key Generation

The key expansion algorithm creates three round keys from 16 bit Cipher key. Each round key is also 16 bits. The first key is used for pre-round and the remaining two keys are used for Addroundkey in each Round, one and two. The order of the key is reversed for decryption. More detail about the S-AES encryption; decryption and key expansion algorithm can be found in Stallings[2] and Forouzan[3].

2.2 Particle Swarm Optimisation

Swarm intelligence is an innovative paradigm used for solving the complicated problems. Particle swarm optimisation is a population based optimisation tool which could be implemented and applied to solve various optimisation problems[11-15].

The canonical PSO model consists of a swarm of particles, which are initialised with a population of random candidate solutions. They move iteratively through the d-dimension problem space to search the new solutions, where the cost, C_k, can be calculated as the certain quality measure. Each particle has a position represented by a position-vector x_i (i is the index of the particle), and a velocity represented by a velocity-vector v_i. Each particle remembers its own best position so far in the vector $x_i^{\#}$ (pbest), and its j-th dimensional value is $x_{ij}^{\#}$. The best position-vector among the swarm so far is then stored in a vector x^*(gbest), and its j^{th} dimensional value is x_j^*. During the iteration time t, previous velocity ($v_{ij}(t)$) is updated to the new velocity ($v_{ij}(t+1)$), determined by Eqn (1). The new position is then determined by the sum of the previous position and the new velocity by Eqn (2).

$$v_{ij}(t+1) = w.v_{ij}(t) + c_1 r_1(x_{ij}^{\#}(t) - x_{ij}(t)) + c_2 r_2(x_j^*(t) - x_{ij}(t)) \quad (1)$$

$$x_{ij}(t+1) = x_{ij}(t) + v_{ij}(t+1). \quad (2)$$

where w is called as the inertia factor, r_1 and r_2 are the random numbers, which are used to maintain the diversity of the population and are uniformly distributed in the interval [0,1] for the j^{th} dimension of the i^{th} particle. c_1 is a positive constant, called as coefficient of the self-recognition component, c_2 is a positive constant, called as coefficient of the social component. From Eqn (1), a particle decides where to move next, considering its own experience, which is the memory of its best past position, and the experience of its most successful particle in the swarm.

2.2.1 Binary PSO

The canonical PSO is basically developed for continuous optimisation problems. In case of cryptanalysis, the problem deals with binary information. Hence the canonical PSO cannot be applied directly. In our problem, the variable x_{ij} represents the key in binary form, hence $x_{ij}(t)$ should take 0 or 1, but from the Eqn (1), we can see that the results of $v_{ij}(t+1)$ may not be an integral, and from Eqn (2) we can see $x_{ij}(t+1)$ may take numerical other than 0,1 after iteration. The equations have to be adjusted in such a way that the velocity and position are to be in binary form. In the binary PSO, we can define a particle's position and velocity in terms of changes of probabilities that will be in one state or the other[13].

At each time step, each particle updates its velocity and moves to a new position according to the Eqns (3) and (4):

$$v_{ij}(t+1) = wv_{ij}(t) + c_1 r_1(x_{ij}^{\#}(t) - x_{ij}(t)) + c_2 r_2(x_j^*(t) - x_{ij}(t)) \quad (3)$$

$x_i(t+1) = 1$ if $\rho \leq s(v_i(t))$, 0 otherwise (4)

where ρ is a random function in the closed interval [0, 1]. The Velocity update equation is similar to that of canonical PSO. The only difference is in the position update as given in the Eqn (4).

3. COST FUNCTION

The main task in formulating the cryptanalysis problem using PSO is to find out the effective cost function. Finding a cost function is a difficult task. Once the effective cost function is defined, then the problem can be solved easily. The cost function used for the considered problem is given by Eqn (5), which represents the ngram[1] statistics of the decrypted message, where the language is assumed to be known.

$$C_K = \alpha\Sigma(i \varepsilon \tilde{A}) |K(i)^u - D(i)^u| + \beta \Sigma(i, j \varepsilon \tilde{A}) | K(i, j)^b - D(i, j)^b| + \gamma\Sigma(i, j, k \varepsilon \tilde{A}) |K(i, j, k)^t - D(i, j, k)^t| \quad (5)$$

In Eqn (5), Ã denotes the language alphabet, i.e., {A,B…Z, _} for English where _ represents the space symbol; *K* and *D* are the known language statistics and decrypted message statistics, respectively; *u*, *b*, and *t* denote the unigram, digram and trigram statistics, respectively. Calculating the frequency of each character represents unigram. Bigram is the extension of unigram that calculates the frequency for pair of letters rather calculating the frequency of an individual character. In the same way, three letter combinations represent trigram statistics. Among the possible 27^2 digrams and 27^3 trigrams, only few were useful. They are reported in Table 1.

Finally α, β and γ are the weights assigning different priorities to each of the three statistics where α+ β + γ = 1. α, β and γ takes 0.2, 0.4 and 0.4, respectively. Since very few digrams and trigrams are considered, more weight is assigned to β and γ.

To compute the cost function, the randomly generated key is used to decrypt the cipher text, i.e., to obtain the plaintext.

Table1. Useful digrams and trigrams

Useful digrams	Useful trigrams
TH, HE, ME, IN, ER, AN, RE, ED, ON, ES, AT, TO, NT, ND,HA, EA, OU, IS, IT, TI, ET, AR, TE, SE, HI, OF, AS, OR.	HER, MEN, ION, THE, ING, ERE, AND, THA, WAS, FOR, HAS.

Once the plaintext is obtained, the cost function counts the occurrence of individual alphabets {A, B…Z, _} and the above said digram combinations and trigram combinations alphabets. These frequency statistics are compared with the known language statistics. For example, the known statistics for the frequency of occurrence of the letter *e* in an english text is 12.5 per cent while for *t* it is 9.25 per cent, the digram frequency for AN is 1.81 per cent while ER it is 2.13 per cent. The frequency for remaining pairs can be found in[10].

While applying PSO for attacking the ciphertext, initially the minimum cost value, calculated by computing the average cost for standard text files with same size is to be defined. A plaintext file is taken from standard english novel with different sizes and encrypted using S-AES, is used as a known ciphertext file.

4. PROPOSED APPROACH FOR ATTACKING S-AES USING PSO

Proposed approach and how PSO can be applied to break the cipher key in the field of cryptanalysis are described here. The problem is to find the key effectively. In a swarm of particles, each particle represents a key, which is a 16 bit binary key. Initialise the swarm particles X_i. Using the generated particles, decrypt the known cipher text to obtain the plaintext and evaluate the cost function from the obtained plain text, by computing the letter frequencies. The best position is associated with the minimum cost value i.e., cost (P_{ibest}) of the particle P_{ibest} and global best P_{gbest} is the best position among all particles in the swarm which is achieved so far. The global position is associated with the global cost value, cost (P_{gbest}) of the particle P_{gbest}.

Velocity and particle's position are updated according to the Eqns (3) and (4). The cost is computed from the updated particle's position in order to update the position of P_{ibest}, P_{gbest}. The process is continued either until the cost function is minimised or maximum number of iterations is reached. If there is no improvement in the cost for some iteration continuously then the algorithm is stopped. An algorithm for finding the key using PSO is shown in Table 2.

Table 2. Algorithm for cryptanalysis of S-AES using PSO

1	Set: (i) Number of iterations (ii) Minimum cost – $Cost_{min}$
2	Initialise the swarm particles
3	Decrypt known cipher text using generated particles.
4	Compute cost value
5	Update the velocity and particle's position according to Eqns (3) and (4)
6	Update the position of gbests and pbests. If Cost ($X_i(t)$) < Cost(P_{ibest}) Then $P_{ibest} = X_i(t)$ and if Cost ($X_i(t)$) < Cost(P_{gbest}) then $P_{gbest} = X_i(t)$
7	Check for stopping criteria. Repeat steps 3-6 until the stopping criteria is satisfied.
8	Display key found: Key = P_{gbest}

5. EXPERIMENTAL SET UP AND RESULTS

The proposed algorithm was implemented using Matlab on an Intel PIV processor. The performances of PSO in attacking the cipher key were analysed. Table 3 shows the PSO parameters to be initialised.

For ciphertext-only attack, the initial population of size 10 is taken with a swarm size of 16 bit i.e., 10 sets of 16 bit keys were taken randomly. The total number of iterations is taken as 2000, i.e. the key search space is set to a maximum of 20,000. The Known cipher text is decrypted using the initial keys. The cost function is calculated using Eqn (5) for each key. A final solution is generated for each and every iteration. The optimum key is found based on the minimum cost value. The convergence of the algorithm depends upon the size of the

Table 3. The PSO parameters

Self-recognition parameter c1	1
Social parameter c2	4-c1
Constriction parameter C	1
Inertia weight	0.999 < w < 0
Initial population	10
Number of iterations	2000

ciphertext file considered. The algorithm converges fast since the more frequency information is available in the decrypted plaintext. Figure 2 shows how the convergence of the cost function depends on the size of the ciphertext.

Table 4 shows the key used for encryption, number of keys searched, and attacked key using PSO. If the size of the ciphertext is large then the algorithm converges fast, thereby the number of keys searched is also less in number. For instance when the number of ciphertexts is 100 the number of key searches was nearly 13,000 whereas for 1000 ciphertext the key search was only 6,000. In all cases, our approach

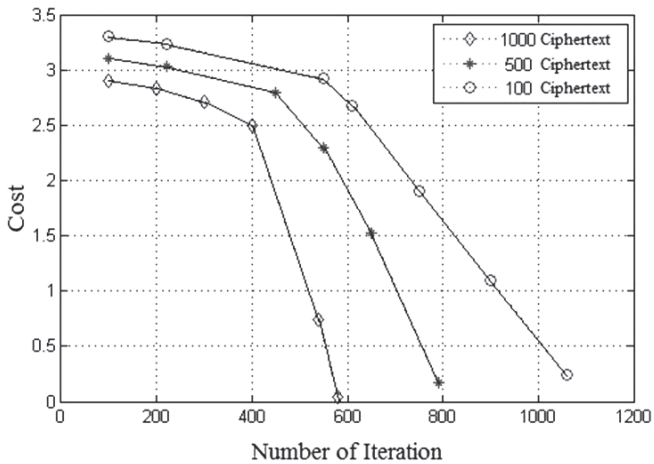

Figure 2. Cost *vs* number of iterations.

breaks the key successfully but a minimum 100 number of ciphertexts were required in order to compute the unigram statistics effectively. In case of Brute force attack the keys searched will be 2^16 (65,536), whereas in case of PSO, in the best case, the key searched is nearly 6000, there is a reduction of the factor of 10 which is a very good factor in cryptanalysis. Even in case of 100 ciphertexts, the key searched is nearly 13,000 and the reduction factor is 5. This is a good reduction factor in cryptanalysis. The result shows that the convergence of the algorithm is independent of the initial population size and initial seed (initial keys).

6. CONCLUSIONS

A new approach has been proposed in this paper for attacking simplified-AES using PSO. The experimental result shows that the PSO is well suited for attacking the ciphers. Proposed algorithm breaks the key successfully using ciphertext-only attack. Though simplified-AES is easier to attack than

Table 4. Simulation results for attacking S-AES using PSO

Cipher text	Key used	Keys searched	Key obtained	Successive bits
100	1010 0111 0011 1011	12,820	1010 0111 0011 1011	16
500	1010 0111 0011 1011	9,164	1010 0111 0011 1011	16
1000	1010 0111 0011 1011	6,904	1010 0111 0011 1011	16
1000	1010 0100 0101 1111	5,817	1010 0100 0101 1111	16

AES which uses 128 bit keys, this approach paves the way to attack AES. The cost function used for ciphertext-only attack method in simplified-AES is not appropriate for AES, as the compilation of frequency statistics becomes infeasible when the number of bits is increased to 128 bit. Hence the future work is to extend the proposal to attack AES by constructing an effective cost function. Also our work extends to attack other complex block ciphers using evolutionary computation.

REFERENCES

1. Koblitz, Neal. A course in number theory and cryptography. Springer International Edition, 2008.
2. Stallings, William. Cryptography and network security principles and practices. Pearson Education, 2004.
3. Forouzan, Behrouz A. Cryptography and network security. Tata McGraw hill Education, Ed 2nd. 2008.
4. Manangi, S.J.; Chaurasia, P. & Singh, M.P. Simplified AES for low memory embedded processors. *Global J.Comp. Comp. Sci. Technol.*, 2010, **10**(14), 7-11.
5. Musa, Mohammad A.; Schaefer, E.F. & Wedig, S. A simplified AES algorithm and its linear and differential cryptanalysis. *Cryptologia*, 200, **27**(2), 148-77.
6. Simmons, S. Algebraic cryptanalysis of simplified AES. *Cryptologia*. 2009, **33**(4), 305-14.
7. Mansoori, S. Davood & Bizaki, H. Khaleghei. On the vulnerability of simplified AES algorithm against linear cryptanalysis. *Int. J. Comp. Sci. Network Security*, 2007, **7**(7), 257-63.
9. Uddin, M.F. & Yousssef, Amr M. Cryptanalysis of simple substitution ciphers using particle swarm optimisation. *In* IEEE Congress on Evolutionary Computation, Canada, 2006. pp. 677-80.
10. Shahzad, Waseem; Siddiqui, A.B. & Khan, F. Aslam. Cryptanalysis of four round DES using binary particle swarm optimisation. *In* GECCO'09. pp. 2161-166.
11. Menezes, A.; Vanoorschot, P. & Vanstone, S. Handbook of applied cryptography. CRC Press, 1996.
12. Kennedy, J. & Eberhart, R. A discrete binary version of the particle swarm algorithm. *In* International Conference on Neural network, Australia,1997. pp. 4104-108.
13. Kennedy, J. & Eberhart, R. Particle Swarm Optimisation. *In* IEEE international Conference on Neural Networks, Australia, 1995. pp.1942-948.
14. Nedjah, Nadia; Abraham, Ajith & Mourelle, Luzia de Macedo. Swarm Intelligent systems. *In* Studies

in Computational Intelligence, 2006, **26**,
15. Nedjah, Nadia; Abraham, Ajith & Mourelle, Luzia de Macedo. Computational intelligence in information assurance and s ecurity. *In* Studies in Computational Intelligence, 2007, **57**.
16. Haupt, R.L. & Haupt, S.E. Practical genetic algorithms, Ed. 2nd, Wiley, 2004.

Contributors

Mr Vimalathithan R. received his ME from Government College of Technology, Coimbatore. Currently pursuing his PhD in Electronics and Communication Engineering from Anna University, Coimbatore. He is currently working as Assistant Professor in the Department of Electronics and Communication, KPR Institute of Engineering and Technology, Coimbatore, Tamilnadu, India. He is a member of Cryptology Research Society of India. His area of interests include: Cryptography, cryptanalysis, electromagnetic interference, and compatibility.

Dr M.L. Valarmathi received her ME (Comp. Sci. Engg.) from Government College of Technology, Coimbatore and PhD (Comp. Sci. Engg.) from Bharathiar University, Coimbatore. She is working as an Assistant Professor in the Department of Computer Science and Engineering, Government College of Technology, Coimbatore, Tamilnadu, India. Her research interests includes: Optimisation techniques, image processing, algorithm design, compilers and network Security. She is a member of ISTE. She has published more than 40 technical papers in National and International conferences and Journals.