ORIGINAL RESEARCH ARTICLE UHD JOURNAL OF SCIENCE AND TECHNOLOGY

Intelligent Techniques in Cryptanalysis: Review and Future Directions

Sufyan T. Al-Janabi[1,2], Belal Al-Khateeb[3] and Ahmed J. Abd[3]

[1]*Department of Information Systems, College of CS and IT, University of Anbar, Ramadi, Anbar - Iraq*
[2]*Department of Computer Science, College of Sceince and Technology, University of Human Development, Sulaimaniya, Kurdistan Region - Iraq*
[3]*Department of Computer Science, College of CS and IT, University of Anbar, Ramadi, Anbar - Iraq*

ABSTRACT

In this paper, we consider the use of some intelligent techniques such as artificial neural networks (ANNs) and genetic algorithms (GAs) in solving various cryptanalysis problems. We review various applications of these techniques in different cryptanalysis areas. An emphasis is given to the use of GAs in cryptanalysis of classical ciphers. Another important cryptanalysis issue to be considered is cipher type detection or identification. This can be a real obstacle to cryptanalysts, and it is a basic step for any automated cryptanalysis system. We specifically report on the possible future research direction of using spiking ANNs for cipher type identification and some other cryptanalysis tasks.

Index Terms: Artificial Neural Networks, Cipher Identification, Classical Ciphers, Cryptanalysis, Genetic Algorithms

1. INTRODUCTION

The basic aim of cryptography is to transmit messages from one place to another in a secure manner. To satisfy this, the original message called "plaintext" is encrypted and sent to the receiver as "ciphertext." The receiver decrypts the ciphertext to get the plaintext. This can be done using a cipher which is a tool that hides the plaintext and converts it to the ciphertext (and also can return back the plaintext from the ciphertext). Ciphers make use of (cryptographic) keys that determine the relationship between the plaintext and the ciphertext. Cryptography can be considered as assemble from security and mathematics. It is used to protect important information and ensure that this information arrives to its destination in peace without violations. Ciphers gradually evolved from simple ones which are currently considered to be easily breakable such as Caesar cipher through more complex cipher algorithms such as the data encryption standard (DES) and the advanced encryption standard (AES) [1], [2].

On the other hand, cryptanalysis means trying to break any security system (or cipher) using unauthorized ways to access the information in that system. Thus, cryptanalysis works against cryptography. The cryptanalyst tries to find any weakness in the cryptographic system to get either the source of information (plaintext) or the key used in the encryption algorithm. This process is called an attack. If this attack is successfully applied, then the cryptographic system is said to be broken. Cryptography and cryptanalysis together form the field of cryptology [3], [4].

In the recent decades, cryptography developed quickly because of the development in computational resources which increased the speed and decreased the time of encryption and decryption processes. This moved cryptography from solving by hand to more and more complex computer programs that need considerably long time and sophisticated attack

Access this article online

DOI: 10.21928/uhdjst.v1n1y2017.pp1-10

E-ISSN: 2521-4217
P-ISSN: 2521-4209

Corresponding author's e-mail: saljanabi@fulbrightmail.org

Received: 10-03-2017 Accepted: 25-03-2017 Published: 12-04-2017

techniques to solve. Hence, instead of using the simple Caesar cipher which needs no more than few minutes (or seconds) to be broken using brute force attack (trying every possible solution), we are using now more complex ciphers (AES, triple DES, etc.) that might need hundreds (or thousands) years to break using brute force attack with the current technology.

One important issue to mention is that despite the technological and mathematical complexity, the modern versions of cryptosystems still follow the same classical concepts. Thus, it is still prudent to apply certain attacks on classical ciphers and study their evolution aspects before using them with more complex modern ciphers. This is quite justifiable considering the nature of intelligent techniques such as GAs, artificial neural networks (ANNs), and evolutionary algorithms (EA).

Although several survey works can be found in earlier literature [5]-[7], more work is needed in this direction to shed the light on various aspects of this kind of interdisciplinary research. The aim of this paper is to review various applications of intelligent techniques in cryptanalysis problems and to investigate some possible future research directions.

The remaining of this paper is organized as follows: Section 2 summarizes various types of ciphers and cryptanalysis attacks in a generic way. The intelligent techniques of ANNs, GAs, and evolutionary computation are reviewed and compared to each other in Section 3. Then, Section 4 reviews the application of GAs in cryptanalysis of classical ciphers. The issue of classification or identification of cipher type is considered in Section 5. Next, we present some insights regarding the future direction of using spiking ANNs in cipher classification in Section 6. Finally, the paper is concluded in Section 7.

2. CLASSIFICATION OF CIPHERS AND ATTACKS

Cryptosystems can be classified in multiple approaches depending on various criteria. This can simplify the study of cryptography science and make it easier to understand and implement. At first, if we take in consideration the amount of data that can be encrypted at a time, we can then classify cryptosystems in two classes:[3]

1. Block ciphers, which encrypt block of data at time like DES
2. Stream ciphers, which encrypt single datum (symbol, byte, or bit) at a time like Caesar cipher.

Second, it is also possible to classify cryptosystems according to the key used in encryption and decryption processes. In this case, we can put a cryptosystem under one of the following:

1. Symmetric key ciphers, where the same key is used for encryption and decryption, for example, Vigenere cipher.
2. Public key ciphers, where one key is used for encryption and another one for decryption, for example, Rivest-Shamir-Adleman system.

Third, we can classify cryptosystems depending on the history and time of invention. Thus, we can put cryptosystems under one of the following:

1. Classical ciphers, which are those ciphers used in the past and can be solved by hand. They became now breakable, for example, Caesar cipher
2. Modern ciphers, which are those complex (computerized) ciphers widely used currently and cannot be solved by hand, for example, AES.

Finally, another classification approach is to classify ciphers according to their building blocks. This approach is typically applied for classical ciphers to divide it into:[3]

1. Substitution systems, where every character is replaced by another one, for example, monoalphabetic ciphers
2. Transposition systems, where characters are rearranged rather than replaced, for example, columnar cipher.

It is also possible to further classify both of the main two categories of classical ciphers: Substitution and transposition ciphers. Transposition ciphers can be classified into sub classes:[3], [8]

- Single transposition: This type transposes one letter at a time, for example, the columnar transposition, route transposition, and grille transposition ciphers
- Double transposition: This type transposes more than one letter at a time.

Substitution ciphers can be classified into sub classes as follows:[3], [9]

- Monoalphabetic substitution ciphers: In this type of encryption techniques, one letter of plaintext is represented by one letter in ciphertext, and one ciphertext letter represents one and only one plaintext letter, so it is the simplest form of substitution techniques. Monoalphabetic substitution includes direct monoalphabetic, reversed monoalphabetic, decimated monoalphabetic, and mixed monoalphabetic ciphers
- Polyalphabetic substitution ciphers: In this type of

encryption, one letter of plaintext is represented by multiple ciphertext letters, and one ciphertext letter represents multiple plaintext letters. There are two types of polyalphabetic substitution ciphers: Periodic (where there is a keyword repeating along plaintext like the Vigenere cipher) and non-periodic (where there is no repeating key, e.g., the running key cipher)

- Polygraphic substitution ciphers: In this type of substitution, more than one plaintext letters are encrypted at a time by more than one ciphertext letters. This includes digraphic, trigraphic, and tetragraphic ciphers. Examples of these ciphers are the Playfair cipher and Hill cipher
- Homophonic substitution ciphers: In this type of substitution, one plaintext letter is represented by multiple ciphertext letters or characters, and every ciphertext letters or characters can only represent one plaintext letter, for example, the nomenclator cipher.

Furthermore, it is possible to define combinations of transposition and substitution ciphers to produce more secure systems. Such combinations are used to avoid the weaknesses in pure transposition and pure substitution systems. A classical example of such combined ciphers is when we combine simple substitution with a columnar transposition. In modern cryptography, ciphers are designed around substitution and transposition principles simultaneously. Fig. 1 depicts various types of classical systems.

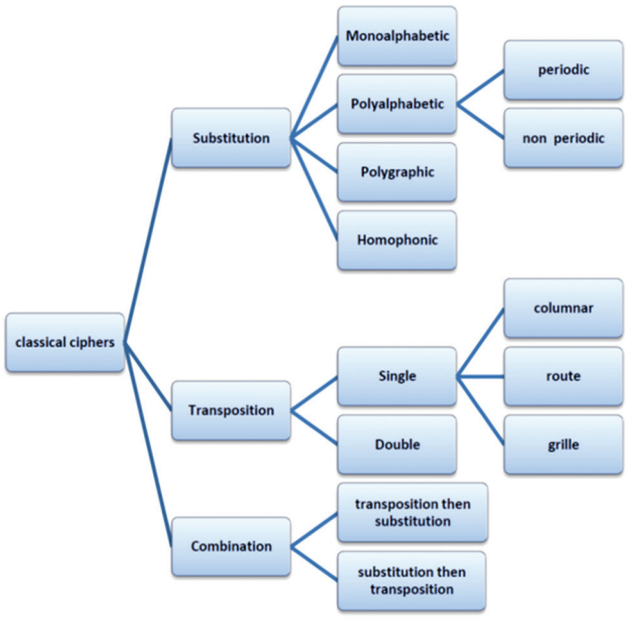

Fig. 1. Most important classical cipher types

Similarly, we can also classify cryptanalysis attacks. Actually, there are many types of such attacks. Some of them can be considered as general types, while others are specific for certain ciphers, protocols, or implementations. Here, we are not going to try to list all attack types rather we are only interested in some generic ways for classifying attacks. It is possible to generically classify attacks based on the amount of information available to the attacker. The amount of information that attacker have is important to make any attack so the cryptanalyst should determine what is available in his hand. Accordingly, we are going to have cipher text only, known plaintext, chosen ciphertext, chosen plaintext, adaptive chosen plaintext, adaptive chosen ciphertext, and related key attacks. Alternatively, we might generically classify attack according to the computational resources (time, memory, and data) required by these attacks [3], [10].

3. INTELLIGENT TECHNIQUES

In this section, we review the relevant intelligent techniques of ANNs, genetic algorithms (GAs), and evolutionary computation. We also give a brief comparison on their characteristics an application scope.

A. ANNs

ANNs are numerical models that use a gathering of basic computational units called neurons that connect with each other to build a network. There are many types of ANNs; each type is suitable for one or more problems depending on the problems itself. Hence, the important thing in ANNs is how to design the topology of ANN that can better describe the problem then solving it using very simple principles to obtain very complex behavior [5], [11]. ANNs can model human brains and use nervous system to solve the problems by learning it with true examples and giving a chance to generalize all solutions. Since the nature of ANNs that simulate the brain and use parallel processing rather than serial computation, we can put ANNs in multiple fields according to the huge capabilities that ANNs can introduce. These fields include classification, approximation, prediction, control, pattern recognition, estimation, optimization, and others.

When using ANN for solving a problem, the following steps should be chosen carefully to make ANN works in an effective way: Design of ANN topology, choosing suitable learning way, and setting the inputs. There are many ANN topologies such as:[12]

- Feed-forward ANNs
- Recurrent ANNs
- Hopfield ANN

- Elman and Jordan ANNs
- Long short-term memory
- Bi-directional ANNs
- Self-organizing map
- Stochastic ANN
- Physical ANN.

There are three generations of neuron models [13]. The first generation of ANNs also called perceptrons, which are composed each of two sections: Sum and threshold. The sum part receives input from a set of weighted synapses. Then, it performs a threshold function on the result of the sum. The input and the output have values that may be equal to either 0 or 1, as shown in Fig. 2.

The second generation of ANNs is composed by two stages:

- Sum of values that are received through weighted synapses
- Sigmoid function evaluator whose input is the result of the sum previously computed. In this generation, the inputs can be any real-valued number, and the output is defined by the transfer function. For example, the sigmoid unit limits outputs to [0; 1], whereas the hyperbolic function produces outputs in the range [1; 1], as shown in Fig. 3.

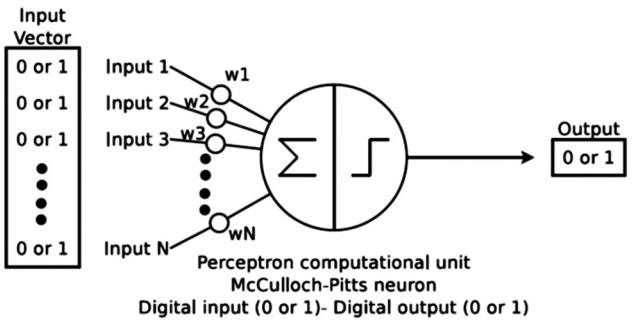

Fig. 2. The first generation of artificial neural networks[13]

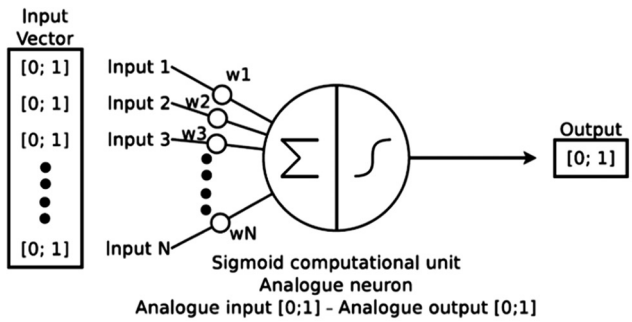

Fig. 3. The second generation of artificial neural networks[13]

The third generation of ANNs is composed by spiking neurons: Neurons which communicate through short signals called spikes. This generation has two main differences when compared with the previous two generation. At first, this generation introduces the concept of time in the simulation, while earlier, the neural networks were based on abstract steps of simulation. Second, such neurons present similarities to biological neurons, as they both communicate using short signals, which in biology are electric pulses (spikes), also known as action potentials, as shown in Fig. 4. The spike train generation can be Gaussian receptive fields [14], Poisson distribution [15], or directed spike generation [16]. Indeed, the applied training algorithm for ANNs is usually the backpropagation [17], while spiking ANNs use Spikeprop [18].

B. GAs

GAs are considered to be one of the best ways to solve a problem, for which there is only a little knowledge. Hence, they work well in any search space. All that is required know is what the solution is needed to be able to do well, and a GA will be able to create a high-quality solution. GAs apply the both principles of selection and evolution to produce several solutions to a given problem [19].

GAs are better applied in an environment in which there is a very large set of candidate solutions and in which the search space is uneven and has many hills and valleys. Although GAs will do well in any environment, they will be greatly outclassed by more situation-specific algorithms in the simpler search spaces. Therefore, GAs are not always the best choice. Sometimes, they can take quite a while to run and are therefore not always feasible for real-time use. However, they are considered to be among the most powerful methods with which to (relatively) quickly create high-quality solutions to a problem. The proper selection of appropriate mutation operators and fitness functions is necessary for implementing a successful attack [19], [20].

In fact, GAs are adaptive heuristic search algorithms based on the evolutionary ideas of natural selection and genetics.

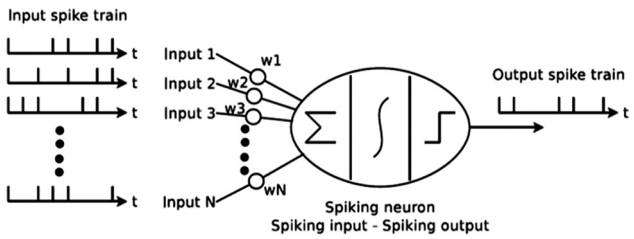

Fig. 4. The third generation of artificial neural networks[13]

Thus, they represent an intelligent exploitation of a random search used to solve optimization problems. They exploit historical information to direct the search into the region of better performance within the search space. The basic techniques of the GAs are designed to simulate processes in natural systems necessary for evolution, especially those follow the principle of "survival of the fittest." This is based on our understanding of nature where competition among individuals for scanty resources results in the fittest individuals dominating over the weaker ones [19].

C. Evolutionary Computation

Simply, evolutionary computation simulates evolution on a computer. The result of such a simulation is a series of optimization algorithms. These are usually based on a simple set of characteristics. Optimization iteratively can improve the quality of solutions to some problem until an optimal (or at least feasible) solution is found. Evolutionary computation is an umbrella term that includes GAs, evolution strategies, and genetic programing [21].

D. Differences Between ANNs, Gas, and Evolutionary Computation

An ANN is a function approximator. To approximate a function, you needs an optimization algorithm to adjust the weights. An ANN can be used for supervised learning (classification and regression) or reinforcement learning and some can even be used for unsupervised learning.

GAs are an optimization algorithm, in supervised learning, a derivative-free optimization algorithm like a GA is slower than most of the optimization algorithms that use gradient information. Thus, it only makes sense to evolve neural networks with GAs in reinforcement learning. This is known as "neuroevolution." The advantage of neural networks like multilayer perceptrons in this setup is that they can approximate any function with arbitrary precision when they have a sufficient number of hidden nodes.

An EA deploys a randomized beam search, which means your evolutionary operators develop candidates to be tested and compared by their fitness. Those operators are usually non-deterministic and you can design them, so they can both find candidates in close proximity and candidates that are further away in the parameter space to overcome the problem of getting stuck in local optima.

EAs are slow because they rely on unsupervised learning: EAs are told that some solutions are better than others but not how to improve them. Neural networks are generally faster, being an instance of supervised learning: They know how to make a solution better using gradient descent within a function space over certain parameters; this allows them to reach a valid solution faster. Neural networks are often used when there is not enough knowledge about the problem for other methods to work.

4. CRYPTANALYSIS OF CLASSICAL CIPHERS USING GAS

There are many approaches and tools that are used in the field of cryptanalysis. One of the successful approaches that achieved promising results is based on GAs. This is mainly due to the nature of GAs that allow reducing the big size of solutions, leading to optimal or likely best solution from this group of solutions. GAs use fitness function to evaluate each solution then select the best one or best group of solutions to generate other children solutions and so on until the cipher is broken. In this section, we report on some interesting aspects of applying GAs in cryptanalyzing classical ciphers.

A. Cryptanalysis of Monoalphabetic Substitution Ciphers

The GA attack on such cipher can be implemented by generated initial keys consisting of permutation of the set of letters. These keys are generated randomly, and after encrypting using each generated key, we can measure the value of fitness of each key. Then, pairs of these keys which have a high fitness value are selected and crossover operation then is used between selected keys to produce new enhancement child keys. After crossover operation is completed, some keys are selected to mutation to enhance the attributes of it by the choice of a random point in a selected key and replacing it with another point. After the two operations are completed, the loop is repeated until the end with suitable stopping [22].

B. Cryptanalysis of Playfair Cipher

For attacking the Playfair cipher using GAs, we should determine the individuals which contain one possible key of the cipher and each individual has its fitness value. One individual is represented as a matrix of 5*5 positions that contain the characters of alphabets distributed randomly. After the generation of the individuals is completed, the selection operation begins according to each individual fitness, so the individual has a highest fitness value that is put in the beginning of the rank. After selection process is completed, the reproduction or crossover operation will begin to produce new children key that may has attribute better than its parents. The crossover operation is implemented by filling

the positions of the child with character of the parents or mutating the child by replacing characters positions locally. The loop continues until meeting the stopping condition.

However, the recovery of the plaintext is not easy to implement usually, for several reasons. One is that words that have double letters may not be counted correctly, due to the fact that the double letters might be split up. Second, because *I* and *J* share a position in the key (typically), all the words that have *I*s and *J*s in them have to be checked using both letters, if the dictionary is fully implemented. Third, the plaintext has no white space to delimit words so being able to tell where words end and begin can be difficult [23].

C. Cryptanalysis of Vernam Cipher

GAs can be used for attacking the Vernam cipher by building a dictionary of words that consist of words that are frequently used in English (e.g., they, the, and when). Then, the fitness value is calculated according to the following steps:[24]

1. Initialize the parameters of the GA and maximum number of iteration
2. Generate random keys which are the population of chromosomes as the 0^{th} generation; each key is a vector with size equal to ciphertext size
3. Decrypt the ciphertext by all generated keys
4. Calculate the fitness function for each chromosome by adding the square value of repeated three letters and four letters which are available in built dictionary. The calculation of fitness function deals with the probability of existing of the three and four letter words in normal English
5. Sort the keys based on decreased fitness values
6. Apply the crossover operator to the parent keys and produce a new generation. Here, a simple two-point crossover can be performed. Furthermore, apply mutation operation by generating two random positions and replace the two letters in these positions by others letters randomly
7. The best key is used to decrypt ciphertext to get the best-decrypted text.

D. Cryptanalysis of Vigenere Cipher

To attack Vigenere cipher using GAs, we should determine the number of attributes that the GA takes as parameters or inputs such as population size, number of individuals tenured per generation, number of random immigrants per generation, number of generations, key length, maximum key length, ciphertext length, known text length, and number of runs per mutation operator combination. These parameters may be used together or some of them might be ignored. The key length parameter is very important, so it must be firstly identified [25].

E. Cryptanalysis of Transposition Ciphers

GAs are very useful to break classical transposition ciphers by finding the sequence of characters that the transposition cipher used. This particular class of algorithms can be used because the automated breaking of such ciphers is very difficult. In spite of that, a number of statistical tools aiding automated breaking have been developed for substitution ciphers, cryptanalysis of transpositions is usually considered to be highly interventionist and demands some knowledge of the likely contents of the ciphertext to give an insight into the order of rearrangement used. Thus, genetic cryptanalyst enables a known plaintext attack to be successfully made, based on only small portion of some plaintext/ciphertext [26].

5. IDENTIFICATION OF CLASSICAL CIPHER TYPE

The typical sequence of steps needs to be followed by cryptanalyst to break any cryptosystems is:[27]

1. The cryptanalyst should determine if the text encrypted by any cipher or it is compressed or generated randomly
2. The cryptanalyst should determine the language of the text
3. The cryptanalyst should determine the type of cipher used in encryption process
4. The cryptanalyst should determine the key used in encryption process
5. The cryptanalyst then uses the key with encrypted data to extract the original data.

When the cryptanalyst wants to identify the cipher type (having just a ciphertext), he/she should extract some features that can lead to estimating the type of cipher. The list below shows a group of features that may help the cryptanalyst in the estimation process:

1. Frequency analysis: Every language has frequency characteristics for its characters such that each character has repeating ratio recognizing it from other characters in normal texts. In English, for example, the letter "*e*" has the greatest frequency ratio (12.70), but the letter "*x*" has the lowest (0.15) [8]. Frequency analysis can be done based on single letter frequency and/or multiple letter frequency (double, triple, etc.). Fig. 5 depicts the typical frequency distribution of single letters in normal English text. Frequency analysis is very useful in differentiating between transposition ciphers and

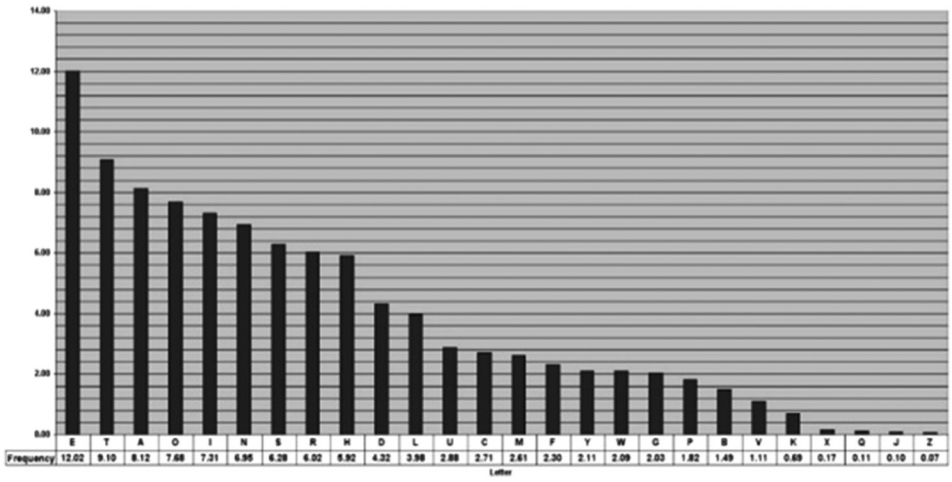

Fig. 5. Frequency distribution of single letters in normal English text[28]

substitution ciphers. Frequency analysis can be used in three main directions:[28]

- The first one is to compute the frequency of ciphertext letters and compare it with the frequency of the original data such that compare the frequency of the letter in ciphertext and natural text and compute the changing in two texts
- The second direction is to compute the frequency of ciphertext letters and find which letters in normal text have the same repeating ratio such that if the letter "*f*" in ciphertext has the same repeating ratio of the letter "*a*" in the original text, we can say the letter "*a*" is encrypted by the letter "*f*."
- Third one is to use frequency analysis to compute if there is any shifting occurs in ciphertext characters such that when the letter "*x*" gives the same ratio of letter "*a*," this indicates that possibly the Caesar cipher which encrypts "*a*" by "*x*" has been used

2. Ciphertext length: The length of ciphertext plays an important role in identification of cipher type where some ciphertext length is exactly divisible by 2 like the Playfair cipher case. Other ciphers (e.g., Hill cipher) can produce ciphertext length divisible by 3, etc.
3. Ciphertext characters number: Some ciphers employee few number of characters such the Baconian cipher which uses just two letters "*a*" and "*b*" in encryption process and the Playfair cipher that uses 25 letters
4. Repeating sections: Periodic polyalphabetic substitution ciphertext has repeating sections with a constant period. This feature can help to identify this type of ciphers [29], [30]
5. AB-BA feature: Ciphertext may contain double sections with its reverse such as "*xy*" and "*yx*." This feature appears in ciphertext produced from Playfair cipher [31]
6. Ciphertext characters type: Some ciphers employee just letters in encryption process another cipher employee letters and numbers [9]
7. Adjacent characters: It can be useful to check if there are any adjacent characters have the same value [28].

6. FUTURE RESEARCH DIRECTIONS

This work lies within a larger team project aiming to design and implement a general cryptanalysis platform for pedagogical purposes. Considering the architectural design of the proposed general cryptanalysis platform, the platform has a number of components or modules including the supervisory module, the crypto-classifier, parallel cryptanalysis modules, feedback and reporting module, graphical analyzer, and the steganography module. Here, we are mainly interested in the crypto-classifier module that is responsible for the identification and classification of the ciphertext type. At least, two levels of classification need to be implemented:[32]

1. Level 1 crypto-classifier: In this module, a first level classification of the considered ciphertext needs to be done so as to decide the general cryptographic category (e.g., classical cipher, block cipher, and public-key cipher) of it. Information obtained from various resource need to be used, and some intelligent classification techniques (such as artificial intelligence, genetics, and neural networks) have to be developed
2. Level 2 crypto-classifier: In the second level of classification, specific algorithm(s) or cipher(s) should

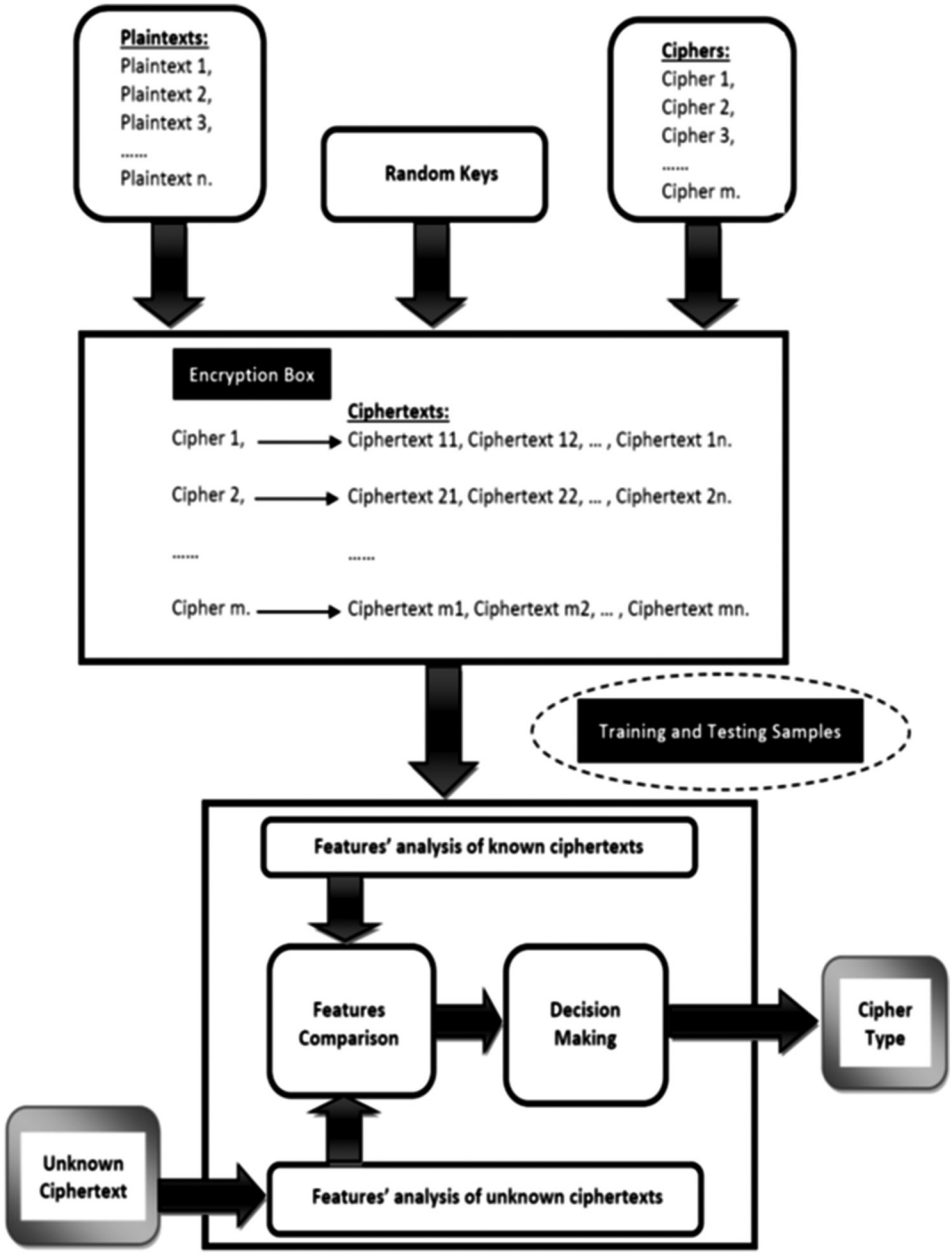

Fig. 6. Data flow of the proposed artificial neural network-based cipher identification process

be assigned for the ciphertext in accordance with the classification done at the first level. For example, if the classifier of level 1 deduced that the ciphertext belongs to the category of block ciphers; level 2 classifier job is to decide which specific block cipher has been used (e.g., DES, AES, and Twofish). Besides the information deduced by different means, some distinguishing characteristics for different ciphers must be known.

Concerning the future research, we are specifically interested in using the estimation capabilities of ANNs to identify the ciphers type. As mentioned previously, ANNs use parallel processing rather than serial computation. This behavior may enable us to move from typical statistical techniques of analyzing any cipher to more powerful generations that provide many solutions at a time. Thus, the analyzing process will depend on how to model ANN

in the correct way and manage the training processes rather than spend the time in mathematical computation of the cipher. Fig. 6 shows the data flow of the proposed estimation process.

The ANN box will have two types of inputs; the first one is a group of training data and the second is a group of testing data. These two groups are managed by ANNs to correct errors produced from estimation process. ANNs would use supervised learning to estimate the cipher type. The number of neurons in the input, hidden, and output layers depend on the number of ciphers used and how much the analyst can extract features from ciphertext.

Several previous works on using ANNs and other techniques for cipher type classification can be found [33]-[37]. However, to the best of authors' knowledge, we could not see specific previous work on using spiking ANNs for this task. Hence, our focus will be directed to this specific application of spiking ANNs. In the first stage, classification of classical ciphers will be considered. In the next stages, other modern cipher types will be taken into consideration also.

7. CONCLUSION

This work is mainly concerned in building automatic tools for various cryptanalysis tasks. This definitely requires the use of suitable intelligent techniques such as GAs and ANNs. The focus here has been on using GAs for cryptanalysis of classical ciphers and adoption of ANNs for cipher type identification. More specific results of cipher classification based on spiking ANNs are going to be presented in a subsequent paper.

REFERENCES

[1] B. Carter, and T. Magoc. "Introduction to classical ciphers and cryptanalysis." A technical report, 11 Sep. 2007. Available: http://www.citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.125.8165. [Feb. 5, 2017].

[2] R. J. Anderson. *Security Engineering: A Guide to Building Dependable Distributed Systems*, USA: John Wiley & Sons, 2010.

[3] W. Stallings. *Cryptography and Network Security Principles and Practice*, 6th ed, Upper Saddle: Pearson Education, Inc., 2014.

[4] M. J. Banks. "A Search-Based Tool for the Automated Cryptanalysis of Classical Cipher." MEng. thesis, Department of Computer Science, The University of York, 2008.

[5] S. Ibrahim, and M. A. Maarof. "A review on biological inspired computation in cryptology." Journal Teknologi Maklumat, vol. 17, no. 1, pp. 90-98, 2005.

[6] S. R. Baragada, and S. Reddy. "A survey of cryptanalytic works based on genetic algorithms." *International Journal of Emerging Trends and Technology in Computer Science (IJETTCS)*, vol. 2, no. 5, pp. 18-22, Sep. Oct. 2013.

[7] H. Bhasin, and A. H. Khan. "Cryptanalysis using soft computing techniques." *Journal of Computer Sciences and Applications*, vol. 3, no. 2, pp. 52-55, 2015.

[8] Department of the Army. *Basic Cryptanalysis*: *Field Manual No. 34-40-2*, *Headquarters*, Washington, DC: Department of the Army, 1990.

[9] F. A. Stahl. "A homophonic cipher for computational cryptography." AFIPS '73 Proceedings of the National Computer Conference and Exposition, New York, pp. 565-568, 4-8 Jun. 1973.

[10] A. K. Kendhe, and H. Agrawal. "A survey report on various cryptanalysis techniques." *International Journal of Soft Computing and Engineering (IJSCE)*, vol. 3, no. 2, May. 2013.

[11] S. Haykin. *Neural Networks and Learning Machines*, 3rd ed, Upper Saddle River, New Jersey: Pearson Education, Inc., 2009.

[12] K. Suzuki. *Artificial Neural Networks-Methodological Advances and Biomedical Applications*, Rijeka, Croatia: InTech, 2014.

[13] S. Davies. "Learning in Spiking Neural Networks." Ph.D. thesis, School of Computer Science, University of Manchester, UK, 2012.

[14] S. M. Bohte, H. La Poutré, and J. N. Kok. "Unsupervised clustering with spiking neurons by sparse temporal coding and multilayer RBF networks." *IEEE Transactions on Neural Networks*, vol. 13, no. 2, pp. 426-435, Mar. 2002.

[15] M. Fatahi, M. Ahmadi, M. Shahsavari, A. Ahmadi, and P. Devienne. "evt_MNIST: A spike based version of traditional MNIST." The 1st International Conference on New Research Achievements in Electrical and Computer Engineering, 2016.

[16] A. Tavanaei, and A. S. Maida. "A minimal spiking neural network to rapidly train and classify handwritten digits in binary and 10-digit tasks." *(IJARAI) International Journal of Advanced Research in Artificial Intelligence*, vol. 4, no.7, pp. 1-8, 2015.

[17] R. Rojas. *Neural Networks*, Berlin: Springer-Verlag, 1996.

[18] S. M. Bohtea, J. N. Koka, and H. La Poutre. "Error-backpropagation in temporally encoded networks of spiking neurons." *Neurocomputing*, vol. 48, no. 1, pp. 17-37, 2002.

[19] D. Goldberg. *Genetic Algorithms*, New Delhi: Pearson Education, 2006.

[20] K. P. Bergmann, R. Scheidler, and C. Jacob. "Cryptanalysis using genetic algorithms." Genetic and Evolutionary Computation Conference GECCO'08, ACM, Atlanta, Georgia, USA, pp. 1099-1100, 12-16 Jul. 2008.

[21] D. B. Fogel. *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*, 3rd ed, New York: John Wily & Sons, Inc., Publication, 2006.

[22] S. S. Omran, A. S. Al-Khalid, and D. M. Al-Saady. "Using genetic algorithm to break a mono-alphabetic substitution cipher." IEEE Conference on Open Systems, Malaysia, pp. 63-68, 5-7 Dec. 2010.

[23] B. Rhew. "Cryptanalyzing the Playfair cipher using evolutionary algorithms." 9 Dec. 2003. Available: http://www.citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.129.4325. [Jul. 15, 2016].

[24] F. T. Lin, and C. Y. Kao. "A genetic algorithm for ciphertext-only attack in cryptanalysis." *IEEE International Conference on Systems, Man and Cybernetics*, vol. 1, pp. 650-654, 1995.

[25] K. P. Bergmann. "Cryptanalysis Using Nature-Inspired Optimization Algorithms." M.Sc. thesis, Department of Computer Science, The University of Calgary, Alberta, 2007.

[26] R. A. Muhajjar. "Use of genetic algorithm in the cryptanalysis of

transposition ciphers." *Basrah Journal of Scienec A*, vol. 28, no.1, pp. 49-57, 2010.

[27] K. N. Haizel. "Development of an Automated Cryptanalysis Emulator (ACE) for Classical Cryptogram." M.Sc. thesis, Faculty of Computer Science, University of New Brunswick, New Brunswick, 1996.

[28] P. Maheshwari. "Classification of Ciphers." master of technology thesis, Department of Computer Science and Engineering, Indian Institute of Technology, Kanpur, 2001.

[29] M. Nuhn, and K. Knight. "Cipher type detection." Information Sciences Institute, University of Southern California, EMNLP, 2014. Available: https://www.semanticscholar.org/paper/Cipher-Type-Detection-Nuhn-Knight/81e5e15afba9301558a7aaca1400b69e0ddaa027#paperDetail. [Jun. 10, 2016].

[30] K. Pommerening. "Polyalphabetic substitutions." Fachbereich Physik, Mathematik, Informatik der Johannes-Gutenberg-Universit at Saarstraße, Mainz, 25 Aug. 2014. Available: http://www.staff.uni-mainz.de/pommeren/Cryptology/Classic/2_Polyalph/Polyalph.pdf. [Jul. 5, 2016].

[31] G. Sivagurunathan, V. Rajendran, and T. Purusothaman. "Classification of substitution ciphers using neural networks." *IJCSNS International Journal of Computer Science and Network Security*, vol. 10, no. 3, pp. 274-279. Mar. 2010.

[32] S. Al-Janabi, and W. A. Hussien. "Architectural design of general cryptanalysis platform for pedagogical purposes, i-manager's." *Journal on Software Engineering*, vol. 11, no. 1, pp. 1-12, Jul. Sep. 2016.

[33] A. D. Dileep, and C. C. Sekhar. "Identification of block ciphers using support vector machines." International Joint Conference on Neural Networks, Vancouver, BC, Canada, pp. 2696-2701, 16-21 Jul. 2006.

[34] J.G. Dunham, M. T. Sun, and J. C. R. Tseng. "Classifying file type of stream ciphers in depth using neural networks." The 3rd ACS/IEEE International Conference on Computer Systems and Applications, pp. 97, 2005.

[35] S. O. Sharif, L. I. Kuncheva, and S. P. Mansoor. "Classifying encryption algorithms using pattern recognition techniques." IEEE International Conference on Information Theory and Information Security (ICITIS), pp. 1196-1172, 17-19 Dec. 2010.

[36] C. Tan, and Q. Ji. "An approach to identifying cryptographic algorithm from ciphertext." 8th IEEE International Conference on Communication Software and Networks, pp. 19-23, 2016.

[37] W. A. R. de Souza, and A. Tomlinson. "A distinguishing attack with a neural network." IEEE 13th International Conference on Data Mining Workshops, pp. 154-161, 2013.