SECURITY AND COMMUNICATION NETWORKS
Security Comm. Networks (2016)
Published online in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/sec.1431

RESEARCH ARTICLE

Mapping private keys into one public key using binary matrices and masonic cipher: Caesar cipher as a case study

Ahmad M. Manasrah[1]* and Basim Najim Al-Din[2]

[1] Computer Sciences Department, Information Technology and Computer Sciences, Yarmouk University, Irbid, Jordan
[2] Computer Sciences, Information Technology and Computer Sciences, Diyala University, Baqubah, Iraq

ABSTRACT

Caesar cipher is a mono alphabetic cipher. It is also a type of substitution cipher in which each letter in the plaintext is "shifted" a certain number of places down the alphabet. However, Caesar cipher method did not last long because of its simplicity and lack of communication security. Therefore, we believe that strengthen the key mechanism should increase its complexity against the various cryptanalysis attacks. This paper proposes an enhanced Caesar cipher method through adopting two private keys that are tied to the character positions (i.e. odd and even) for encryption and/or decryption. The two private keys are mapped into one public key to be transferred to the recipient. At the end, the results show that the new cryptosystem is inevitable to cryptanalysis attack. And the cipher text is reduced in size and thus, memory space. The public key generation process is proven to be a one-way function utilizing binary matrices that are generated and shared between the two communicating parties. Copyright © 2016 John Wiley & Sons, Ltd.

KEYWORDS
cryptography; substitution cipher; Caesar cipher

***Correspondence**
Ahmad M. Manasrah, Computer Sciences Department, Information Technology and Computer Sciences, Yarmouk University, Irbid, Jordan.
E-mail: ahmad.a@yu.edu.jo

1. INTRODUCTION

Because of the invention of computer systems in the twentieth century and the introduction of the communication systems (i.e. networks), protecting computer networks and communication channels became vital because unauthorized users start to exploit various types of information from the cyberspace. Nevertheless, unauthorized users should not otain access to the information on the move neither at rest. Hence, in addition to the various security measures that are deployed to the Internet and within the organizations, enforcing cryptography should harden and increase the security of the information over the communication channels [1,2].

The Egyptians used hieroglyphics to decorate tombs to tell the life story of the deceased since 2000 BC [3]. The Hebrew alphabetic is another cryptography method that is derived from the Aramaic alphabet and variously known by scholars as the Jewish script, square script, block script, and it is used in the Hebrew and Jewish languages [3,4]. Another type of cryptographic methods is the pigpen cipher (or Masonic cipher, Freemason's cipher, Rosicrucian cipher, Tic-tac-toe cipher). This type is a substitution cipher, which replaces letters by special symbols [5]. Julius Caesar developed a simple method of shifting letters of the alphabet in which, each letter is shifted three position back/forward to obtain the cipher text [6]. The encryption process in the Caesar cipher is carried out through the following function [7,8].

$$E(x) = (x + n) \bmod 26 \quad (1)$$

However, the drawback of the Caesar cipher resides in its simple encryption and decryption algorithm [6]; one can decrypt the message without knowing the encryption key. However, this can be easily broken by reversing the encryption process through the following function [9].

$$D(x) = (x - n) \bmod 26 \quad (2)$$

The Caesar cipher algorithm encrypts /decrypts a message in less time compared with other methods [11]. Moreover, Caesar cipher algorithm does not deal with the spaces, and it encrypts each letter by another letter from the alphabet. Further, because it is a matter of shifting letters, it is easy to guess the key as it is a single private key that has the space of 1 to 25 different combinations. As a result, various methods and algorithms are being researched and developed for securing data against unauthorized individuals. For instance, Ochoche, Abraham, and Shefiu Ganiyu O. (2012) proposed some novel improvements to the traditional Caesar cipher algorithm by eliminating the spaces (i.e. blanks) from the cipher text [9]. The blanks elimination is carried out over two steps (1) encryption and (2) scrambling the letters in the cipher text. The authors used a single key technique in the encryption/decryption process. Therefore, it is easy to guess the characters with some extra efforts and time [10]. Singh, Ajit, Aarti Nandal*, et al.* (2012) proposed a combination of substitution and transposition ciphers that combine Caesar cipher with the Rail fence technique [11]. This approach was quite promising as they remove the spaces between words and utilize a stack of words instead. The authors also reduce the size of the cipher text to be transmitted securely [11,12]. Saroha, Vinod, Suman Mor*, et al.* (2012) proposed a double columnar transposition method to strengthen the Caesar cipher encryption process. They also adopt a single key technique but columnar [13]. Their approach encrypt the plain text raw by raw and then column by column. On the other hand, they found that the combination of these two classic techniques (i.e. Caesar and columnar) strengthen the encryption process and make it more secure compared with the one proposed in [11].

On the other hand, Padmapriya, A. and P. Subhasri (2013) proposed a new level of data security solution using the reverse Caesar cipher encryption using ASCII full 256 characters to add a sort of complexity to the encryption process [6]. However, the proposed solution also uses a single key technique for encryption and decryption process. Mishra, Anupama (2013) adopted two level transposition methods with the Caesar cipher encryption/decryption process [2]. The transposition method employs a two-level encryption with one key for each level, and similarly for the decryption. However, brute force attack is not possible because it uses different key levels during the encryption process. This makes it much more secure. However, the proposed approach never been evaluated under the frequency attack, and the size of the cipher text is still the same as the original plain text.

In conclusion, we notice that the aforementioned approaches have common drawbacks that can be summarized as: (i) single key adoption as well as the (ii) size and (iii) space of the cipher text [14]. Consequently, this paper proposes a new mechanism to eliminate the aforementioned drawbacks in the Caesar cipher method through employing a double private key mechanism along with the pigpen/ Masonic cipher. The enhanced Caesar cipher method should yield a compressed cipher text by merging any two identical and adjacent characters from the cipher text into one symbol from the Masonic cipher. The proposed two private keys are then mapped into one public key to be transferred to the other end along with the cipher text over any communication channel. Fortunately, because of the Caesar cipher simplicity and speed, various applications can utilize it if it is enhanced further. For instance, Caesar cipher can be used like any other known cryptosystems. In fact, a privacy preserving methods can utilize an enhanced Caesar algorithm in their cloud computing infrastructure to increase the data security and preserve the data privacy from service providers or cloud consumers.

The rest of the paper is organized as follows. Section 2 explains our methodology of the enhanced Caesar cipher method encryption, decryption process, and key generation mechanism. Section 3 describes the experiments that were carried out to prove the inevitability of the enhanced cryptosystem against various types of cryptanalysis attacks. Our conclusion is in Section 4 . The future work and the research direction can be found at Section 5.

2. THE PROPOSED ENCRYPTION, DECRYPTION, AND KEY GENERATION METHOD

As stated earlier, the challenges in the traditional algorithm are the weakness in the key generation technique and using a single key in this algorithm to encrypt/decrypt the message. However, the proposed algorithm uses the binary matrices as a key generation technique with a double key to encrypt/decrypt the message. Such that the cipher text became more secure than the cipher text in the traditional algorithm and the keys became more difficult to be guessed.

2.1. Phase I: the encryption phase

The encryption process starts by parsing the plain text character by character and encrypt each character in its place (i.e position) using the even $E_{even} = (X + k_1)\ \%\ 26$ or odd $E_{odd} = (X + k_2)\ \%\ 26$ encryption private key. The output at this stage (i.e. *first stage cipher text*) is then parsed again character by character looking for similar adjacent characters to be replaced by one symbol from the Masonic cipher as illustrated by Algorithm A. The output at this stage (i.e. *second stage cipher text*) is a cipher text that is ready to be transferred to the other party along with the one public key in a binary format as discussed in Section 2.2.

Algorithm A: Encryption Algorithm

```
Cipher(Inp_Msg, Inp_Msg_Len, Pvt_key_1, Pvt_key_2)
Initialize orig_Alpha [size] // 1-D array with the characters
from the original text.
Initialize out_Msg[size] // 1-D array with blank characters.
For i = 1 to Inp_Msg_Len – 1 // Traverse the Inp_Msg
for j = 1 to size // Traverse orig_Alpha array
if (Inp_Msg[i] == orig_Alpha[j]) then
pos = j; //character position
if (pos is odd) then
out_Msg[i] = (j + Pvt_key_1)%26
else
out_Msg[i] = (j + Pvt_key_2)%26
```

```
endif
endif
endfor
endfor
for i = 1 to Inp_Msg_Len − 1 // Traverse the out_Msg
if (out_Msg[i]==out_Msg[i + 1]) then
swap(out_Msg[i]&out_Msg[i + 1],get_Masonic(out_Msg[i]))
endif
endfor
return out_Msg;
```

2.2. Phase II: the key generation phase

The process discussed earlier (Phase I) uses two private keys for the encryption process. Each key encrypts certain characters based on their locations. However, sending the two private keys to the receiver is of concern here. Therefore, we are proposing a mechanism to merge the two private keys into one public key to increase its security and add a sort of complexity to the whole process. The Key generation process starts by creating (n) matrices of order (LxM) as illustrated by Algorithm B. Where each matrix has the following structure:

$$A_0 = \begin{bmatrix} 2^i & \dots & x \\ y & \dots & z \end{bmatrix},$$
$$A_2 = \begin{bmatrix} 2^{i+1} & \dots & m \\ n & \dots & q \end{bmatrix}, \dots\dots\dots\dots\dots\dots\dots,$$
$$A_n = \begin{bmatrix} 2^{i+j} & \dots & q \\ r & \dots & s \end{bmatrix}$$

where 2^i is the binary weight that is normally used in the binary number conversion. The purpose of doing so is to arrange the key according to its binary representation $\forall\, i \in N$. After the matrices are generated, the two private keys are then encoded and combined as one key to be sent to the receiver. For this purpose, we generate a unique identifier for each matrix based on Equation (3)

$$ident(A_i) = \left|a_{(m,l)} - a_{(1,l)}\right| + a_{(1,1)}, \forall i \in N \qquad (3)$$

Algorithm B: Binary Matrix Generation Algorithm
```
matrix_gen(matrix_no)
for No =1 to $2^{matrix\_no}$ // No is an iterator
# A = matrix_no //number of matrices used in key generation
L = int($2^{matrix\_no -1}$/matrix_no)-1 //# rows
M = 2 matrix_no -1/L //# columns
for i = 1 to L
for j = 1 to M
for A = 1 to matrix_no
A[1,1] = $2^{matrix\_no}$ // Each first element is $2^{matrix\_no}$ in every matrix
end for
end for
end for
for i = 1 to L
for j = 1 to M
for A = 1 to matrix_no
A[i,$2^{j+1}$] = A[i,$2^{j}$] + $2^{j}$ + 1 // all other elements in the matrices
end for
end for
end for
end for
```

These identifiers are then used to encode the two private keys to be mapped into one public key as in Algorithm.

Algorithm C: Binary Matrix Generation Algorithm
```
For each private key,
Find $a_{(1,1)}$, $\forall A_i$
Convert the private key into 7-bit binary number as follows:
$B_i = \sum_{i=0}^{n} a_{(1,1)} = \text{Pvt\_key\_1}$
Calculate the $key = \sum_{i=0} ident(A_i) \forall A_i \in B_i, i \in N$
```

For instance, the private key 6 (i.e. *number of shifts*) is equal to 0000110 in binary format, which indicates that the key existed in matrix A_1 *and* A_2, which is also equivalent to $ident(A_1) + ident(A_2) = 4 + 8 = 12$ as illustrated in Figure 2. Hence, the final code is a 7-bit binary number that is equal to 0001100. Note that the private key number is $\sum a_{(1,1)}, \forall\, a \in A_1, A_2$. At this stage, the two encoded private keys are then merged into one public key to be transferred to the receiver as the structure portrayed in Figure 1. Note that the said structure is appended to the cipher text generated from Phase I.

On the other hand, the decoding stage decodes the public key code into two private keys in a reverse process. Consequently, the private keys code is then converted to the original values in a reverse process as portrayed in Figure 2.

2.3. Phase III: decryption process

At this phase, a transmitted frame is received from phase II (i.e. the sender). The first step in this phase starts by separating the public key code from the cipher text, which is represented by the first 14-bit character. The one public key is decoded into two private keys as discussed in Phase II and illustrated in Figure 2.

For instance, the aforementioned binary representation indicates that the private key (key_1) fall into (n) matrices where $ident(A_i) = 4$, *and* $ident(A_i) = 8$. Solving for Equation (3), we conclude that $ident(A_i) = 4$, belongs to matrix A1 and $ident(A_i) = 8$ belongs to matrix A2, respectively. After that, mapping the symbols of the Masonic cipher in the cipher text to its two identical characters that are corresponds to this symbol. The decryption phase (Algorithm D) involves getting the two private keys from the binary matrices using the keys $D_{odd} = (x - Pvt\_key\_1)\,\%\,26$ and $D_{even} = (x - Pvt\_key\_2)\,\%\,26$.

Algorithm D: Decryption algorithm
```
Decipher(Inp_Msg,Inp_Msg_Len, Pvt_key_1, Pvt_key_2)
Extract_Key(key)
For i = 1 to inp_Msg_Len-1
swap(Masonic character, out_Msg[i]&out_Msg[i + 1])
if pos==odd
pos = (j- Pvt_key_1)%26
else
pos = (j- Pvt_key_2)%26
endif
endfor
return inp_Msg
```

3. PERFORMANCE EVALUATION

For the purpose of evaluating the enhanced method, we applied different cryptanalysis methods [3,15] to ensure the

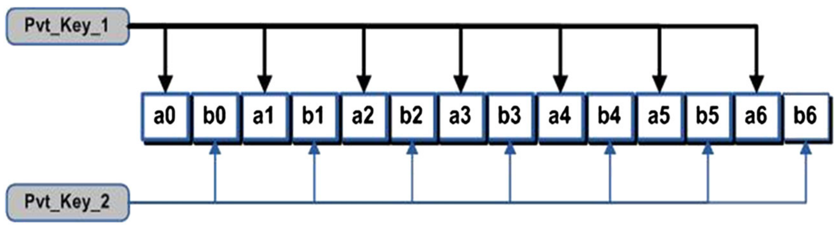


Figure 1. The public key code structure.

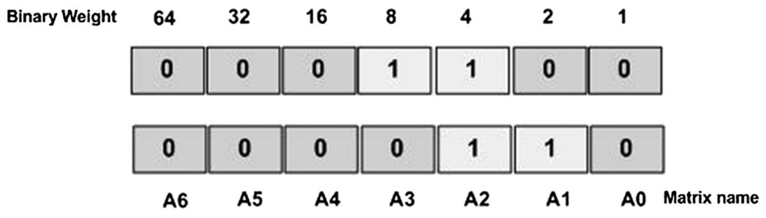


Figure 2. The key decoding process.

security of the enhanced method over different plain texts of various lengths and keys, as summarized in Figure 3.

Table I shows a portion of the plain text and its corresponding first and second stage cipher text. Note that the second cipher text is the final cipher text to be sent over to the receiver.

3.1. Cryptanalysis using the brute force attack

The brute force attack is one of the most common methods that are used by the attackers to break cipher texts [11]. This depends on finding all the probabilities for the correct plain text through trying all possible key combinations to determine the correct plain text [15,16]. The brute force attack is evaluated through the entropy measure [17,18] over the three samples of the two cipher text stages. The entropy measure is the average amount of information that is contained in each message received. The entropy measure can be calculated as

$$H(x) = -\sum_i P(x_i)\log_2(P(x_i)) \qquad (4)$$

where $H(X)$ is the entropy of (x) value, (i) is the number of the repeated probability of each letter, and (P) is the probability of (x_i). The results of applying the brute force attack are summarized in Table II. The minimum entropy per letter is corresponding to the encryption key.

The results of applying the brute force attack for the best guess of the 26 letters that represent the English letters were evaluated for each number of shifting letter (key) (i.e. entropy per letter) as illustrated in Figure 4.

For instance, for the second stage cipher text of sample_2, the entropy value was 5.045, which corresponds to a key of 19 with a reduced number of character of 2%. Consequently, we can conclude that the brute force attack failed to break the cipher text that is obtained from the proposed method because of employing two private key at the encryption phase. This indicates that the proposed approach is inevitable to the brute force cryptanalysis method.

3.2. Cryptanalysis using the frequency attack

The basic use of the frequency analysis is to count the frequencies of the cipher text letters and associates the guessed plaintext letters to them. Moreover, the frequency attack cryptanalysis was also used to calculate the

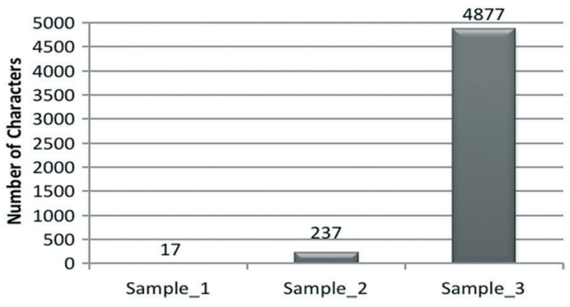


Figure 3. Plain text samples character distribution.

Table I. Sample plain text and cipher text (1st and 2nd stage cipher text) for the three samples.

Sample	Sample plain text
Sample_1	Missing messessipe
1st cipher text	*sjytoom sfytktyjvf*
2nd cipher text	*sjyt⊓ m sfytktyjvf*
Sample_2	Because of the invention of computer systems in the twentieth century …
1st cipher text	*hvi xp ulfjrwioxjso pj gpqqyuis tctxfqt jr xii xxioxjiul gfruysc eoh…*
2nd cipher text	*hvi xp ulf jrwioxjso pj gp⌐yuis tctxfqt jrx□ <ioxjiul gfruysc eoh…*
Sample_3	The stream ciphers encrypt plain text one byte or bit at a time …
1st cipher text	*wnh vzukds ilvkkuy kqiuesz voglt zhdw rth eewk uu eow dz g zlsh vafn …*
2nd cipher text	*wnh vzukds ilv⊏uy kqiuesz voglt zhdw rth ∟wk > eow dz g zlsh vafn gv …*

Table II. Brute force attack summary.

Sample/stage	Plain text no. of characters	Private keys	Public key (decimal)	Guessed key	Entropy/letter	Cipher text no. of character
Sample_1/1st stage	17	K1 = 6, K2 = 1	921	5	4.755	15
Sample_1/2nd stage		K1 = 6, K2 = 1	921	5	4.142	
Sample_2/1st stage	237	K1 = 4, K2 = 1	611	17	5.011	221
Sample_2/2nd stage		K1 = 4, K2 = 1	611	19	5.045	
Sample_3/1st stage	4877	K1 = 3, K2 = 6	2422	1	4.943	4665
Sample_3/2nd stage		K1 = 3, K2 = 6	2422	2	4.379	

frequency of the letters, digraphs, Trigraph, and the most common double letters in the cipher text and compare the results with that of the English language (Table IV) to conclude the differences between them. To evaluate the strength of the proposed method, we also applied the frequency attack on the two stages of the encryption process over the three samples; the results are reported in Table III.

We can notice from Table III that the frequencies of the English language letters are different from that of the one obtained from the proposed method for the three samples. For instance, the most frequent letter in sample_2 cipher text was (I), which is different from that of the English language that is (E). Similarly for the most common Digraph, Trigraph, and double letters of the cipher text that do not

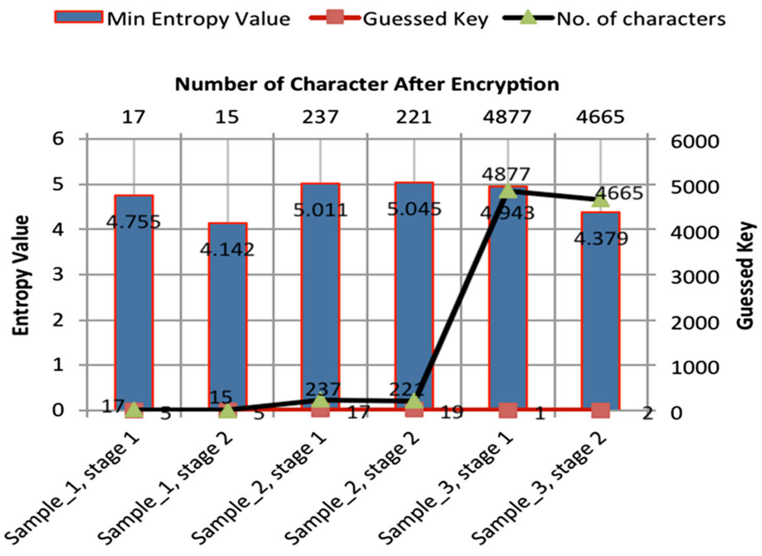


Figure 4. The details of the brute force attack on the three samples.

Table IV. English common digraph, Trigraph and double letters.

English common digraph	TH, HE, AN, IN, ER, ON, RE, ED, HA, AT, EN
English common trigraph	THE, AND, THA, ENT, ION, TIO, FOR, NDE, HAS, NCE, TIS, OFT, MEN
English common double letters	SS, EE, TT, FF, LL, MM, OO

match with those in the English language. As a result, we concluded that the proposed method is inevitable to the frequency attack because it hides the characteristics of the language itself (i.e. cipher text) (Table IV).

3.3. Cryptanalysis using the autocorrelation attack

In order to calculate the similitude between the plain text with itself and the cipher text with itself and to determine the secret key length as well as discovering the non-randomness in a cipher text, the autocorrelation attack was applied to measure the correlation of the two values in the same cipher text at different time steps [19–21]. The correlation coefficient between the time series x (i.e. cipher text/plain text) and y (i.e. delayed cipher text/ delayed plain text) is given by Equation (5)

$$r = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\left[\sum (x_i - \bar{x})^2\right]^{1/2} \left[\sum (y_i - \bar{y})\right]^{1/2}} \tag{5}$$

Table III. Frequency attack detailed results.

Sample/stage	No. of characters			Private keys				Final no. of characters		
Sample_1	17			K1 = 6, K2 = 1				15		
English letter frequencies	E	T	A	O	I	N	S	H	R	D
	12.7	9.1	8.2	7.5	7.0	6.7	6.3	6.1	6.0	4.3
Cipher text letter frequencies (1st stage)	T	Y	F	J	O	S	K	M	V	A
	17.6	17.6	11.7	11.7	11.7	11.7	5.88	5.88	5.88	0
Cipher text letter frequencies (2nd stage)	13.2	13.2	9.1	9.0	1.1	9.0	4.6	4.5	4.5	0
Cipher text common digraph	YT, SJ, JY, TO, OO, OM, MS, SF, FY, TK, KT, TY, YJ									
Cipher text trigraph	SJY, JYT, YTO, TOO, OOM, OMS, MSF, SFY, FYT, YTK, TKT, KTY, TYJ									
Cipher text double letters	OO									
Sample_2	237			K1 = 6, K2 = 1				221		
English letter frequencies	E	T	A	O	I	N	S	H	R	D
	12.7	9.1	8.2	7.5	7.0	6.7	6.3	6.1	6.0	4.3
Cipher text letter frequencies (1st stage)	I	X	S	J	U	O	P	R	F	W
	9.7	9.7	8.86	6.32	5.9	5.48	5.06	5.06	4.64	4.21
Cipher text letter frequencies (2nd stage)	4.5	6.5	4.3	5.3	4.4	4.4	4.2	4.1	3.5	3.0
Cipher text common Digraph	XJ,XI,SO,II,UL,JR,JS,IS,SS,XP,OX,OP,PJ									
Cipher text Trigraph	XII, XJS, JSO, SOP, XPU, PUL, ULF, LFJ, FJR, IOX, OXJ, OPJ, GPQ									
Cipher text double letters	II, SS, XX, WW, QQ									
Sample_3	4877			K1 = 3, K2 = 6				4665		
English letter frequencies	E	T	A	O	I	N	S	H	R	D
	12.7	9.1	8.2	7.5	7.0	6.7	6.3	6.1	6.0	4.3
Cipher text letter frequencies (1st stage)	E	T	I	A	R	S	O	N	H	C
	14.0	10.6	7.4	7.25	7.17	6.97	6.37	5.94	5.86	5.33
Cipher text letter frequencies (2nd stage)	1.3	3.1	3.8	2.4	4.8	3.0	4.6	3.6	8.1	0.6
Cipher text common digraph	HE,TH, ER, TE, IN, TI, RE, ES, EC, ON, AR									
Cipher text trigraph	THE, HER, ION, TIO, EXT, INT, TEX, CIP, IPH, PHE, PRO, CRY, RYP									
Cipher text double letters	SS, TT, EE, LL, PP, OO, DD, FF, HH, RR, MM, II, AA									

The result of applying this attack on sample_1 plain text and cipher text displaced (t) position is reported in Figure 5.

Figure 5 (a) shows the autocorrelation of the plain text and the plain text displaced (t) position. From which, there is a matching between the characters in lags greater than (1) and less than (9). On the other hand, Figure 5 (**b**) shows the autocorrelation of the cipher text to itself displaced (t) positions. We can notice a match between the characters in lag (6). Comparing the two-autocorrelation figures earlier, we can notice that the autocorrelation of the cipher text is reduced compared with its uniform with the autocorrelation of the plain text. Hence, the autocorrelation attack failed to break the cipher text neither guess the key length.

Similarly, the same experiment was evaluated for sample_2, and the results are reported in Figure 6.

Figure 6 (**a**) earlier demonstrates a match between the characters in lag (50) and lag (90). This match indicates a correlation in these lags of the plain text. Similarly, Figure 6 (b) shows the autocorrelation of the cipher text with a match between the characters in the lags between (20) and (30). So it is noticed that the autocorrelation of the cipher text is reduced compared with its uniform with the autocorrelation of the plain text. Therefore, the proposed approach resists the autocorrelation attack till this stage.

Finally, Figure 7 illustrates the results obtained from applying the autocorrelation attack on sample_3.

With a match between the characters in lag (1) and lag (20) in Figure 7 (a), a correlation in these lags is existed. Similarly, there is a match between the characters in lag (140) and (160). The autocorrelation of the cipher text is reduced compared with the autocorrelation of the plain text. Hence, the autocorrelation attack failed to break the cipher text of the proposed method.

We can conclude that the proposed approach is also resistant to the auto correlation attack. Especially that, the cipher text is reduced compared with its uniform with the auto correlation of the plain text. Thus, guessing the characteristics of the letters of the cipher text is not possible.

3.4. Cryptanalysis using differential power analysis attack

Differential power analysis attack is one of the most common types of power analysis attacks on cryptographic system as part of the side channel attack [22]. DPA attacks employ statistical techniques to extract secret information from the cipher texts even when the information is mixed with any type of noises [23]. The successful attack on cryptographic algorithms through using DPA allows the attackers to extract the secret cryptographic keys used by the algorithm through extracting any differences in the power consumption [21,24–26]. The

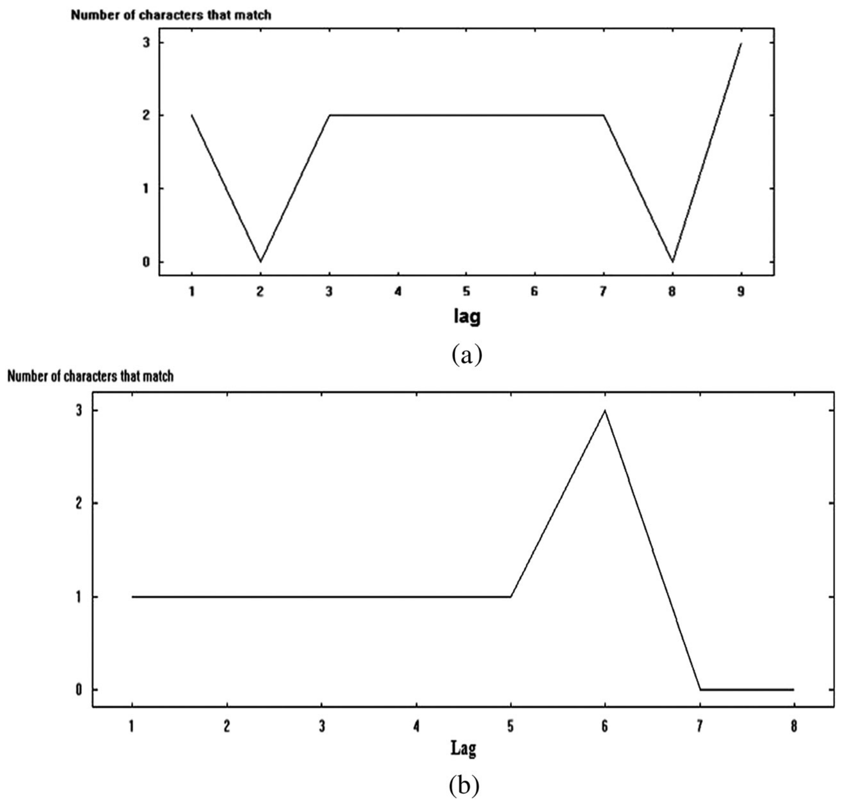
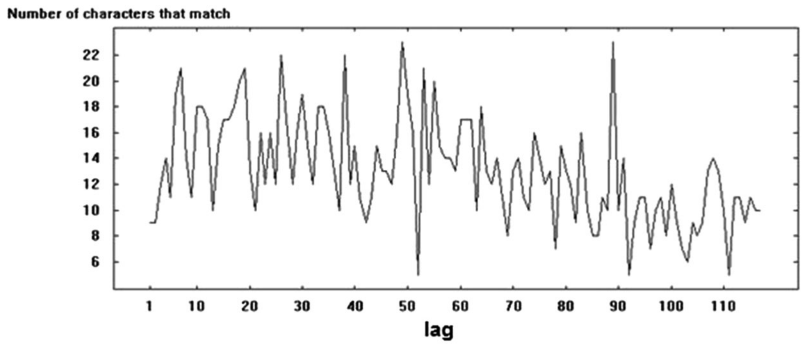
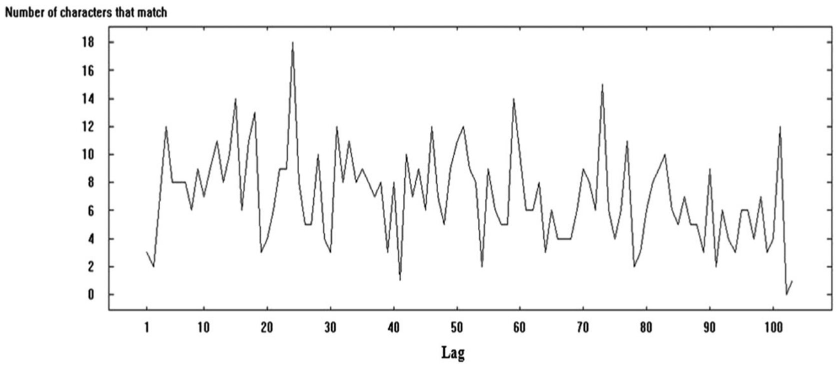


Figure 5. Autocorrelation of the plain text (a) and the cipher text (b) of sample_1.



(a)



(b)

Figure 6. Autocorrelation of the plain text (a) and the cipher text (b) of sample_2.

idea of the DPA attack is to gather many different power consumption curves and then assuming the key value, after that dividing the data into two groups (0 and 1 for chosen bit), and calculating the mean value curve of each group. For the purpose of applying the DPA attack, let us define the DPA selection function $D(C_i, b, K_s)$, which is defined as computing the value of the (b th) output bit, and the cipher text (C), where (Ks) is the guessed key. The attacker then computes a k-sample of traces $\Delta_D[1..k]$ through finding the differences between the average of the traces for (D = 1) and the average of the traces for (D = 0) through Equation (6).

$$\Delta_D = \frac{\sum_{i=1}^{m} D(C_i, b, , K_s)T_i[j]}{\sum_{i=1}^{m} D(C_i, b, , K_s)} - \frac{\sum_{i=1}^{m} (1 - D(C_i, b, , K_s))T_i[j]}{\sum_{i=1}^{m} (1 - D(C_i, b, , K_s))} \tag{6}$$

where (Ti) is the set of power traces containing (*k*) samples such that, the traces are a set of power consumption measurements that taken during the cryptographic operations. From Equation (6), if the (*K*s) is incorrect then $\lim_{m\to\infty} \Delta D[j] \approx 0$, which means the bit computed using the selection function (*D*) will be different from the actual target bit for half of the cipher texts (*C*i). Therefore, the components of the trace are uncorrelated to the selection function (*D*) [27,28]. Therefore, we evaluated the affect of applying side channel attack/DPA cryptanalysis on the cipher text of the three samples, and the results are illustrated in Figure 8.

Figure 8 (a) shows one maximum peak at position 235 with a high correlation value of 0.096465. This indicates that the guessed key is 235 where the actual keys, which were used in this sample, were k1 = 6 and k2 = 1. Similarly, Figure 8 (b) shows one maximum peak at position 238 with a high correlation value of 0.083795. This also indicates that the guessed key is 238 where the actual keys are k1 = 4 and k2 = 1. Figure 8 (c) shows that there is one maximum peak at position 1 with high correlation value of 0.033725, which indicates that the guessed key is 1, where the actual keys are k1 = 3 and k2 = 6. As a result, the DPA attack failed to guess the correct key (key hypothesis). Thus, guessing the characteristics of the letters of the cipher text is not possible as well.

3.5. Mathematical proof

The most basic primitive for cryptographic applications is the one-way function property, which is easy to compute but hard to invert. The concept of the one-way function is considered as a central importance in cryptography

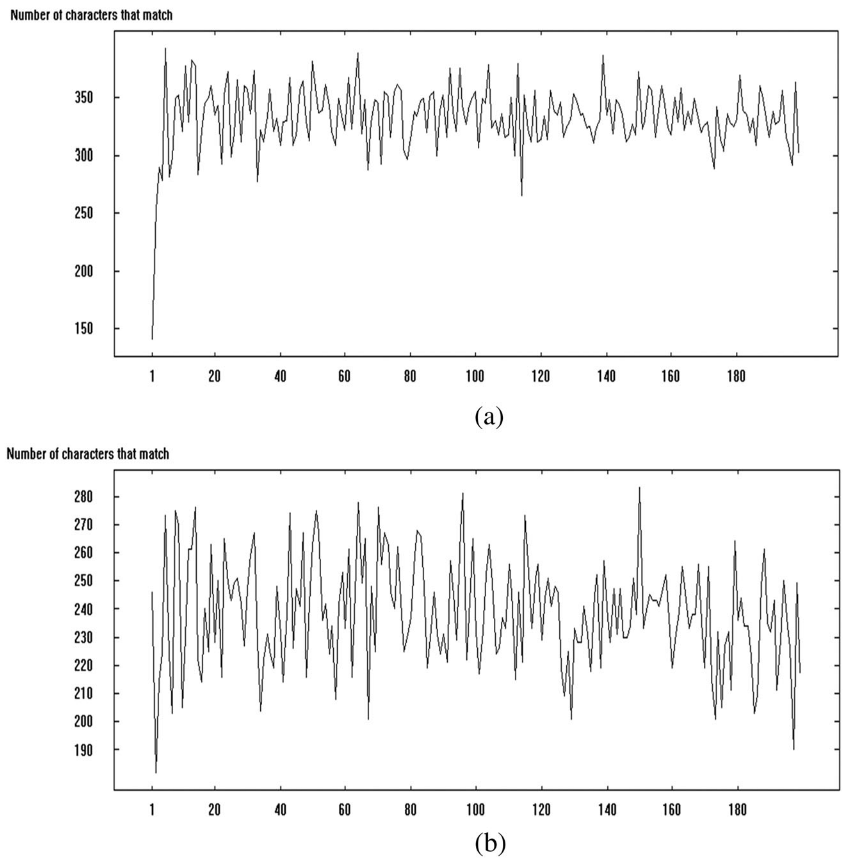


Figure 7. Autocorrelation of the plain text (a) and the cipher text (b) of sample_3.

[29,30]. For the purpose of proofing that the proposed method is an NP-hard, the following definitions are needed.

Definition 1. A function $\sum : N \rightarrow R$ is negligible if $\forall\, c, \exists\, n_0$ such that [31].

$$\sum(n) \left\langle \frac{1}{n^c} \right. \quad \forall n \geq n_0 \tag{7}$$

Definition 2. A strong one-way function is a simple one-way function, and the definition for the complexity (i.e. hardness) of the strong/simple one way function is defined as $\forall\, nuPPTA, \exists\, \varepsilon$ such that $\forall\, n \in N$

$$p_r\left[x \leftarrow \{0,1\}^n, A\left(1^n, f(x) \in f^{-1}(f(x))\right)\right] \leq \in (n) \tag{8}$$

And ε is a negligible value [31].

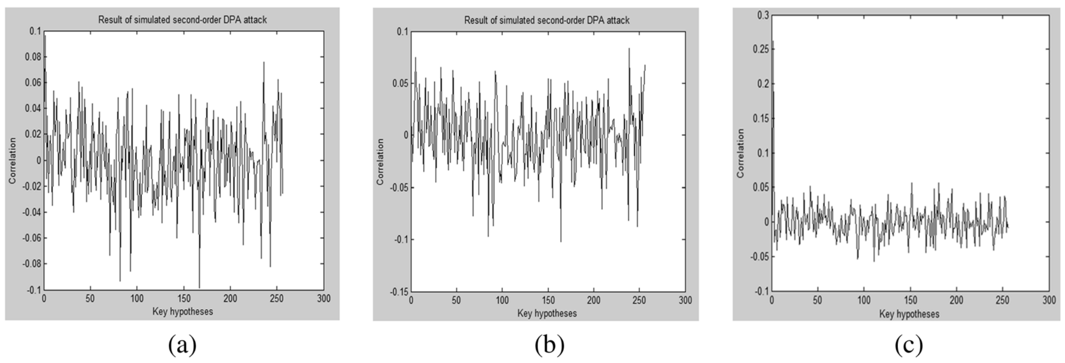


Figure 8. Differential power analysis (DPA) attack for sample_1 (a), sample_2 (b), and sample_3 (c).

Definition 3. A function is a weak one-way function if it is easy to compute and $\exists\, q(x), \forall\, nuPPTA, \forall\, n \in N$ such that [31].

$$p_r\left[x \leftarrow \{0,1\}^n, A(1^n, f(x)) \in f^{-1}(f(x))\right] \leq 1 - \frac{1}{q(n)} \quad (9)$$

Theorem 1. "If there exists a weak one-way function, then there exists a strong one-way function" [29–31].

The equation for the key generation used in this research is in the interval of the key range (x) as shown later:

$$\left|\left|\left|a_{(1,1)} + a_{(2,1)} - a_{(n,m)}\right| - \left(a_{(1,1)} + a_{(2,1)}\right)\right| - 6\right| \leq x \quad (10)$$

From the definition of the absolute value,

$$|x| \leq b \rightarrow -b \leq x \leq b. \quad (11)$$

Solving Equation (11) according to the absolute value definition is not valid, because there is more than one absolute value in this equation. To clarify that, taking the part when the equation equals (x), which is observed through the definition of the absolute value of x

$$|x| = a \rightarrow x = a \quad \text{or} \quad x = -a \quad (12)$$

And the definition of the absolute value of $(x - a)$ is

$$|x - a| = b \rightarrow (x - a) = b \ \text{ or } \ (x - a) = -b \quad (13)$$

Then, the equation of the proposed key generation method becomes

$$\pm\left(\pm\left(\pm\left(\left(\left(a_{(1,1)} + a_{(2,1)} - a_{(n,m)}\right) - \left(a_{(1,1)} + a_{(2,1)}\right) - 6\right)\right)\right)\right) \quad (14)$$

That means the entire absolute value can be rewritten as

$$\left|a_{(1,1)} + a_{(2,1)} - a_{(n,m)}\right| = x_1 \rightarrow a_{(1,1)} + a_{(2,1)} - a_{(n,m)} = x_1 \text{ or } a_{(1,1)} + a_{(2,1)} - a_{(n,m)} = -x_1 \quad (15)$$

In the same manner, the second entire absolute value

$$\left|\left|\left(a_{(1,1)} + a_{(2,1)} - a_{(n,m)}\right)\right| - \left(a_{(1,1)} + a_{(2,1)}\right)\right| = x_2 \rightarrow \left(a_{(1,1)} + a_{(2,1)} - a_{(n,m)}\right) - \left(a_{(1,1)} + a_{(2,1)}\right) = x_2 \ \text{ or } \left(a_{(1,1)} + a_{(2,1)} - a_{(n,m)}\right) - \left(a_{(1,1)} + a_{(2,1)}\right) = -x_2 \quad (16)$$

So solving for the absolute value becomes

$$\left|\left|\left|a_{(1,1)} + a_{(2,1)} - a_{(n,m)}\right| - \left(a_{(1,1)} + a_{(2,1)}\right)\right| - 6\right| = x \rightarrow \left(\left(\left(a_{(1,1)} + a_{(2,1)} - a_{(n,m)}\right) - \left(a_{(1,1)} + a_{(2,1)}\right) - (6)\right)\right) = x \text{ or } \left(\left(\left(a_{(1,1)} + a_{(2,1)} - a_{(n,m)}\right) - \left(a_{(1,1)} + a_{(2,1)}\right) - (6)\right)\right) = -x \quad (17)$$

All the result earlier are for one case that, all absolute values take the positive direction, but there are another many cases when positive and negative directions are taken into account because it is a part of the solution. However, to find the inverse of the proposed method equation

$$f(a) = \left|\left|\left|a_{(1,1)} + a_{(2,1)} - a_{(n,m)}\right| - \left(a_{(1,1)} + a_{(2,1)}\right)\right| - 6\right| \quad (18)$$

Let $f(a) = b$ then

$$a = \left|\left|\left|b_{(1,1)} + b_{(2,1)} - b_{(n,m)}\right| - \left(b_{(1,1)} + b_{(2,1)}\right)\right| - 6\right|. \quad (19)$$

Because there are three absolute values, it is difficult to take all the probabilities of the positive and negative directions to solve this equation. In addition, there are three unknown variables, which are, $a(1,2), a(2,1), a(n.m)$, but there is one equation. This makes it impossible to solve the equation and find the three variables. In order to find the three variables, we must have more than one equation to solve simultaneously.

So from the previous discussion, it is clear that there is no inverse function for the proposed method function. Moreover, the function $f(x) = |x|$ is hard because it will take 2^c time to write a valid inverse to something such that $f(x) = c$.

The hardness definition is [29,30,32]

$$\forall x, \{0,1\}^* p_r\left[\left(1^{|x|}, f(x)\right) \in f^{-1}(f(x))\right] = 1 \quad (20)$$

where $1^{|x|}$ is the string concatenation of 1 repeated $|x|$ times. This is just to ensure that the output of (Λ) does not shrink the size of its output too much. Therefore, the hardness of the proposed function is

$$\forall x, \{0,1\}^* p_r[A(1^{\left|\left|\left|a_{(1,1)} + a_{(2,1)} - a_{(n,m)}\right| - \left(a_{(1,1)} + a_{(2,1)}\right)\right| - 6\right|}, f(x)) \in f^{-1}(f(x))] = 1 \quad (21)$$

where $1^{||a(1,1) + a(2,1) - a(n,m)| - a(1,1) + a(2,1)|}$ is the string concatenation of 1 repeated (M) times where $M = ||a(1,1) + a(2,1) - a(n,m)| - a(1,1) + a(2,1)|$

Another problem with this definition is the quantification of inputs (x) to (f). We do not exactly need that because the hardness equation holds for all (x). From definition (2), it is concluded that the proposed method is a one-way function and NP-hard problem ($P \neq NP$) that is easy to compute but difficult to invert.

4. CONCLUSION

This paper presents an enhanced Caesar cipher method through the use of a double private key technique and mapping the two private keys into one public key using binary matrices. In order to evaluate the strength of the proposed method, the brute force attack, frequency attacks, autocorrelation attack, and the DPA attack methods were applied into different sizes of messages/plain texts. After applying the attacks on the messages, it was observed that the proposed method is difficult to break by the brute force attack and frequency attack.

In this paper, we use double key technique rather than one key to increase the complexity of the proposed method. Moreover, we reduced the number of characters in the cipher text through mapping any to identical concurrent characters in one Masonic symbol. Converting the two private keys into one public key using the binary matrices increases the hardness of guessing the correct keys and to add more complexity to the cipher text when sending it across the Internet. Finally, merging the public key with cipher text and adding more complexity to know the position of the key in the cipher text that have been send across a communication channel.

Finally, the evaluation of the proposed method through using different types of cryptanalysis techniques proved that the cipher text obtained from the proposed method was very difficult to break. The autocorrelation attacks computed the autocorrelation of the plain text with itself and the cipher text with itself in terms of the number of characters that match. And it is failed to guess the correct key because there was no correlation between the text and itself. The cross correlation also failed in breaking the cipher text. Similarly, the DPA attack failed to guess the correct key. In this research, the one-way function has been proven for the proposed method, which means that the proposed method is an $P \neq NP$, and this leads to the fact that this problem is an NP-hard problem.

5. FUTURE WORK

This work can be improved by increasing the number of keys to obtain more secure cryptosystem in addition to reduce the number of characters and the size of the message. Also, the binary matrices can also be used as a key generation technique in modern cryptosystem. However, this cryptosystem is designed to work with English text. Therefore, considering this method to work with different languages text is left open as a future work. Also, the key generation technique used in this research can be parallelized and can be applied to block cipher cryptosystems.

REFERENCES

1. Luciano D, Prichett G. Cryptology: from caesar ciphers to public-key cryptosystems. *The College Mathematics Journal* 1987; **18**(1):2–17.
2. Mishra A. Enhancing security of caesar cipher using different methods. *IJRET: International Journal of Research in Engineering and Technology* 2013; **2** (9):327–332.
3. Harris S. CISSP All-in-One Exam Guide. 2012; McGraw-Hill Global.
4. Scripts A.Old Hebrew. 2012 [cited 2014; Available from: http://www.ancientscripts.com/old_hebrew.html.
5. Sanchez J. The Masonic Trowel. 2012 March 22, 2014 [cited 2014; Available from: www.themasonictrowel.com/education/others_files/the_masonic_cipher/the_masonic_cipher.htm.
6. Padmapriya A, Subhasri P. Cloud computing: reverse caesar cipher algorithm to increase data security. *International Journal of Engineering Trends and Technology* 2013; **4**(4).
7. Govinda K, Sathiyamoorth E. Multilevel cryptography technique using graceful codes. *Journal of Global Research in Computer Science (JGRCS)* 2011; **2**(7):1–5.
8. Bruen AA, Forcinito MA. *Cryptography, Information Theory, and Error-Correction: A Handbook For The 21st Century*, Vol. **68**. John Wiley & Sons, 2011.
9. Ochoche A, Ganiyu OS. An improved caesar cipher (ICC) algorithm. *International Journal Of Engineering Science & Advanced Technology (IJESAT)* 2012; **2** (5):1198–1202.
10. Singh A, Nandal A, Malik S. Implementation of caesar cipher with rail fence for enhancing data security. *International Journal of Advanced Research in Computer Science and Software Engineering* 2012; **2**(12):78–82.
11. Carter B, Magoc T. Classical ciphers and cryptanalysis. *Space* 2007; **1000**:1–5.
12. Dhavare A, Low RM, Stamp M. Efficient cryptanalysis of homophonic substitution ciphers. *Cryptologia* 2013; **37** (3):250–281.
13. Saroha V, Mor S, Dagar A. Enhancing security of caesar cipher by double columnar transposition method. *International Journal of Advanced Research in Computer Science and Software Engineering* 2012; **2**(10):86–88.
14. Al-Obaidi BNA-D. Mapping private keys into one public key using binary matrices and masonic cipher: applying caesar cipher as a use case. In *Computer Sciences*. Yarmouk University: Jordan, 2014.
15. Schneier B. *Applied Cryptography: Protocols, Algorithms, and Source Code In C* (Second edn). John Wiley & Sons, 2007.
16. Agency NS. Cryptanalysis/Signals Analysis. 2009, 2014 [cited 2014; Available from: https://www.nsa.gov/careers/career_fields/cryptsiganalysis.shtml.
17. Ratnaparkhi A. A Simple Introduction to Maximum Entropy Models for Natural Language Processing. 1997, University of Pennsylvania.
18. Rajesh R, Nisha Y, Mayank V *et al.* Entropy, the Indus Script, and language: a reply to R. Sproat. *Computational Linguistics* 2010; **36**(4):795–805.

19. Brier E, Clavier C, Olivier F. Correlation power analysis with a leakage model. In *Cryptographic Hardware and Embedded Systems-CHES 2004*. Springer, 2004; 16–29.
20. Garcia E. Evaluation of the Single Keybit Template Attack. 2011, Air force Institute of Technology.
21. Wong C. Analysis of DPA and DEMA attacks. In *Analysis*. San José State University, 2012; 1–2012.
22. Maximov A. Some Words On Cryptanalysis Of Stream Ciphers. 2006; Lund University.
23. Jaffe J, Rohatgi P, Riscure MW. Efficient Sidechannel Testing For Public Key Algorithms: RSA Case Study. 2011.
24. Clavier C, Coron J-S, Dabbous N. Differential power analysis in the presence of hardware countermeasures. In *Cryptographic Hardware and Embedded Systems—CHES 2000*. Springer, 2000.
25. David H, Kris T, Alireza H *et al.* AES-based security coprocessor IC in 0.18-CMOS With resistance to differential power analysis side-channel attacks. *Solid-State Circuits, IEEE Journal of* 2006; **41**(4): 781–792.
26. Joshi MK, Pandey A. Trend and spectral analysis of rainfall over India during 1901–2000. *Journal of Geophysical Research: Atmospheres (1984–2012)* 2011; **116**(D6).
27. Monjur A, Santosh G, Dipanwita C, Indranil S. First-order DPA vulnerability of Rijndael: security and area-delay optimization trade-off. *IJ Network Security* 2013; **15**(3):219–230.
28. Choudary O. Breaking Smartcards Using Power Analysis. University of Cambridge, 2005.
29. Attila B, Ködmön J, Attila P. A one-way function based on norm form equations. *Periodica Mathematica Hungarica* 2004; **49**(1):1–13.
30. Alina B, Lance H, Christopher H, Jörg R. One-Way Functions in Worst-Case Cryptography: Algebraic and Security Properties. Algebraic and security properties, 1999.
31. Merkle RC. One way hash functions and DES. In *Advances in Cryptology—CRYPTO'89*. Springer: New York, 1990.
32. Goldreich O. *P, NP, And NP-Completeness: The Basics of Computational Complexity*. Cambridge University Press, 2010.