Vaccine: A Block Cipher Method for Masking and Unmasking of Ciphertexts' Features

Ray R. Hashemi
Amar Rasheed
Jeffrey Young
Department of Computer Science
Armstrong State University,
Savannah, GA, USA
e-mails: {rayhashemi, amarrasheed, alanyoung7}@gmail.com

Azita A. Bahrami
IT Consultation
Savannah, GA, USA
e-mail: Azita.G.Bahrami@gmail.com

Abstract— **A ciphertext inherits some properties of the plaintext, which is considered as a source of vulnerability and, therefore, it may be decrypted through a vigorous datamining process. Masking the ciphertext is the solution to the problem. In this paper, we have developed a new block cipher technique named *Vaccine* for which the block size is random and each block is further divided into segments of random size. Each byte within a segment is instantiated using a dynamic multi-instantiation approach, which means (i) the use of Vaccine does not produce the same masked outcome for the same given ciphertext and key and (ii) the options for masking different occurrences of a byte is extremely high. Two sets (100 members in each) of 1K long plaintexts of *natural* (borrowed from natural texts) and *synthesized* (randomly generated from 10 characters to increase the frequency of characters in the plaintext) are built. For each plaintext, two ciphertexts are generated using Advanced Encryption System (AES-128) and Data Encryption Standard (DES) algorithms. Vaccine and two well-known masking approaches of Cipher Block Chaining (CBC), and Cipher Feedback (CFB) are applied separately on each ciphertext. On average: (a) the Hamming distance between masked and unmasked occurrences of a byte using Vaccine is 0.72 bits higher than using the CBC, and CFB, and (b) Vaccine throughput is also 3.4 times and 1.8 times higher than the throughput for CBC and CFB, correspondingly, and (c) Vaccine *masking strength* is 1.5% and 1.8% higher than the masking strength for CBC and CFB, respectively.**

Keywords- Cyber Security; Masking and Unmasking Ciphertext; Variable-Block Cipher Vaccination; Masking Strength

I. INTRODUCTION

Protecting sensitive electronic documents and electronic messages from unintended eyes is a critical task. Such protections are often provided by applying encryption. However, the encrypted text (ciphertext) is often vulnerable to datamining. For example, let us consider the plaintext message of: "*The center is under an imminent attack".* The plaintext may be converted into the following ciphertext using, for instance, a simple *displacement* encryption algorithm: "xligirxvmwyrhivermqqmrirxexxego". The features of the plaintext are also inherited by the ciphertext—a point of vulnerability.

To explain it further, word "attack" is among the key words related to security. The characteristics of the word are: (i) length is six, (ii) the first and the fourth characters are the same, and (iii) the second and the third characters are the same. Using these characteristics, one can mine the given ciphertext and isolate the subtext of "exxego" that stands for "attack" which, in turn, may lead to decryption of the entire message.

More sophisticated encryption modes, such as CBC and CFB [1][2][3] are not exempt from the inherited-features problem. The Block cipher techniques that employ CBC/CFB encryption mode to produce distinct ciphertexts are vulnerable to information leakage. In the case of CBC/CFB employing the same Initial text Vector (IV) with the same encryption key for multiple encryption operations could reveal information about the first block of plaintext, and about any common prefix shared by two plaintext messages. In CBC mode, the IV must, in addition, be unpredictable at encryption time; in particular, the (previously) common practice of re-using the last ciphertext block of a message as the IV for the next message is insecure (for example, this method was used by SSL 2.0). If an attacker knows the IV (or the previous block of ciphertext) before he specifies the next plaintext, he can check his guess about plaintext of some block that was encrypted with the same key before (this is known as the TLS CBC IV attack) [4].

The logical solution for inherited-features problem is to mask the ciphertext using a masking mechanism that is *dynamic* and supports a high degree of *multi-instantiations* for each byte. A dynamic masking mechanism does not produce the same masked outcome for the same given ciphertext and the same key. The high degree of multi-instantiation masking mechanism replaces the *n* occurrences of a given byte in the ciphertext with *m* new bytes such that *m* is either equal to *n* or extremely close to *n*. The literature addresses many of these masking techniques [5][6].

Our goal is to introduce a masking mechanism named Vaccine that can mask the inherited features of a ciphertext in the eye of a data miner while providing for transformation of masked ciphertext into its original form, when needed. Vaccine will be dynamic and support a high degree of multi-instantiations for each byte of data, and has the following three unique traits, which makes it a powerful masking mechanism: It (1) divides the ciphertext into random size blocks, (2) divides each block into random size segments, and (3) every byte within each segment is randomly instantiated into another byte. All three traits are major departures from the norm of masking mechanisms.

The rest of the paper is organized as follows. The Previous Works is the subject of Section 2. The Methodology is presented in Section 3. The Empirical Results are discussed in Section 4. The Conclusions and Future Research are covered in Section 5.

II. PREVIOUS WORKS

Masking the features of a ciphertext that are either inherited from the plaintext or generated by the encryption scheme itself is the essential step in protecting a ciphertext. The block cipher and stream cipher mode of operations provides for such a step. We are specifically interested in CBC [7][8][9] and CFB [10] as samples of the block cipher and stream cipher mode of operations. They are to some degree comparable to the proposed Vaccine.

CBC divides the ciphertext into fixed–length blocks and masks each block separately. The use of fixed-length block demands padding for the last partial block of the ciphertext, if the latter exist. The CBC avoids generating the same ciphertext when the input text and key remain the same by employing an Initial text Vector (IV). CFB eliminates the need for possible padding of the last block (that is considered vulnerability for CBC [11]) by assuming the unit of transmission is 8-bits. However, CFB also uses IV for the same purpose that it was used by CBC. In contrast, Vaccine splits the ciphertext into the random size blocks and then divides each block into segments of random size. Masking each pair of segments is done by using a pair of randomly generated patterns. As a result, Vaccine needs neither padding nor IV. The randomness of the block size, segment size, and patterns used for instantiation of a given character are the major departure points of Vaccine from the other block and stream cipher approaches.

III. METHODOLOGY

We first present our methodology for instantiation of a byte, which contributes into dynamicity of Vaccine and then introduce our methodology for building Vaccine. The details of the two methodologies are the subjects of the following two subsections.

A. *Instantiation*

Instantiation is the replacement of a byte, c, by another one, c', such that c' is created by some modifications in c. To perform the instantiation, we present our two methods of *Self-substitution* and *Mixed-Substitution.* Through these methods, a number of parameters are introduced that are referred to as the *masking* parameters. At the end of this subsection, we present the masking parameters as a profile for the patterns suggested by the substitution methods.

1) Self-Substitution: Consider byte 10011101 and let us (i) pick two bits in positions p_1 and p_2 such that $p_1 \neq p_2$, (ii) flip the bit in position p_1, and (iii) swap its place with the bit in position p_2—Two-Bit-One-Flip-Circular-Swap technique.

It is clear that the pairs (p_1=1, p_2=7) and (p_1=7, p_2=1) create different instances for the byte. Therefore, the order of p_1 and p_2 is important. The number of possible ways selecting a pair (p_1, p_2) from the byte is 7*8=56, which means a byte may be instantiated by 56 possible different ways using Two-Bit-One-Flip-Circular Swap technique. The technique name may be generalized as *K-R-Bit-M-Flip-Circular-Swap*. For the above example K=2 and M=1, as shown in Figure 1. (We introduce the parameter R shortly.)

One may pick 3-bits (K=3) to instantiate the byte. Let us assume 3 bits randomly selected that are located in the positions p_1, p_2, and p_3. There are many ways that M-Flip-Circular-Swap technique can be applied:

a. (One-Flip-Circular-Swap) Flip one of the three bits and then make a circular swap among p_1, p_2, and p_3.
b. (Two-Flip-Circular-Swap) Flip two out of the three bits and then apply circular swapping.
c. (Three-Flip-Circular-Swap) Flip all three bits and then apply circular swapping.

The number of possible combinations grows to 5040.

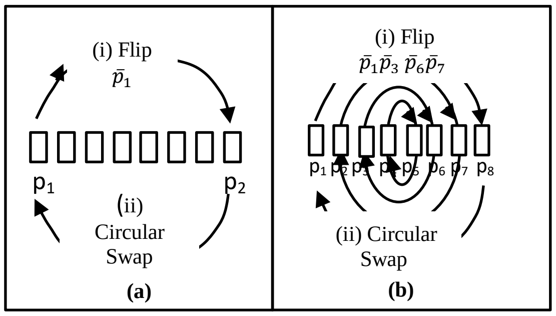
(i) Flip
$\bar{p}_1$
p_1
p_2
(ii) Circular Swap
(a)
(i) Flip
$\bar{p}_1\bar{p}_3\ \bar{p}_6\bar{p}_7$
p_1 p_2 p_3 p_4 p_5 p_6 p_7 p_8
(ii) Circular Swap
(b)

Figure 1. K-Bit-M-Flip-Circular-Swap Technique: (a) K=2 and M=1 and (b) K=8 and M=4

Using K-R-Bit-M-Flip-Circular-Swap for all possible values of K (K=2 to 8) and M (M=1 to K-1) generates the total of (X=1,643,448) possible substitutes for a given byte. If either K or M is equal to zero then, the self-substitution has not been enforced and in this case X=1 (the byte itself). Now, we explain the role of parameter, R (where, R is a byte long).

Let us refer to the case of K=2 and M=1 one more time that is able to facilitate the generation of 56 possible number of instantiations of a given byte using all the possible pairs of (p_1=•, p_2=•). That is, the two positions of p_1 and p_2 could have any value from 1 to 8 as long as $p_1 \neq p_2$. What if one is only interested in those instantiations resulting from the pairs of (p_1=3, p_2=•), which by definition also includes instantiations resulting from the pairs of (p_1=•, p_2=3). The chosen value (bit) of interest for p_1 is a value from 1 to 8 that is expressed by setting the bit of interest in R. The number of bits that are set to "1" in R is always equal to M. For our example, R="00000100".

The pairs represented by (p_1=v, p_2=•) are the set of seven pairs of {(p_1=v, p_2=1), . . ., (p_1=v, p_2=8)}. The seven pairs are named the *primary set* for the *primary signature* of (p_1=v, p_2=•). The (p_1=*, p_2=v), which is a tweaked version of (p_1=•, p_2=v) is the *Complementary signature* of (p_1=v, p_2=•) and stands for the other set of seven pairs {(p_1=8, p_2=v), . . ., (p_1=1, p_2=v)}. These seven pairs make the *complementary set* for (p_1=*, p_2=v). (Values of p_1, in the complementary set, are in reverse order of values of p_2 in the primary set.)

The primary and complementary sets also referred to as the *primary sub-pattern* and *complementary sub-pattern*, respectively. The two sub-patterns collectively make a *pattern* and (K=2, M=1, R="00000100") is the *pattern's profile*.

The profile of (K=4, M=3, R="00001011") means four bits are chosen from the byte out of which three bits (M=3) in positions 1, 2, and 4 are the positions of interest (p_1=1, p_2=2, p_3=4.) Therefore, the primary signature and the Complementary signatures are, respectively, defined as (p_1=1, p_2=2, p_3=4, p_4=•) and (p_1= *, p_2=2, p_3=4, p_4=1). It is clear that M cannot be equal to K, because, when M= K, the primary and complementary sets are the same and they have only one member.

When none of the bits in R is set to "1", it means R has not been enforced. In this case, we have one pattern. However, to apply Vaccine, we need to determine the primary and complementary sets for this pattern, which is provided by default value of R (i.e., R with its M least significant bits set to "1".)

2) Mixed-Substitution: In a nutshell, the instantiation of the given byte, c, and each key byte are done separately. One of the instantiated key bytes is selected as the *key image* and the final instance of c is generated by XORing the key image and the instantiated c. The details are cited below.

Application of self-substitution with masking parameters of (K, M, and R) on a given byte generates the primary and the complementary sub-patterns of (u_p^1 . . . u_p^n) and (u_c^m . . . u_c^1). The subscripts p and c stand for these two sub-patterns and there are *n* and *m* members in the p and c sub-patterns, respectively. The key byte B_j is instantiated into another byte using the self-substitution with masking parameters of (K_j, M_j, and R_j, for j=1 to 4). Application of self-substitution on the individual four bytes of the key (B_1 . . . B_4) generates the primary and the complementary sub-pattern for each byte as follows:

($u_p^{1B_1}$ . . . $u_p^{n1B_1}$) and ($u_c^{m1B_1}$ . . . $u_c^{1B_1}$),
($u_p^{1B_2}$ . . . $u_p^{n2B_2}$) and ($u_c^{m2B_2}$ . . . $u_c^{1B_2}$),
($u_p^{1B_3}$ . . . $u_p^{n3B_3}$) and ($u_c^{m3B_3}$ . . . $u_c^{1B_3}$), and
($u_p^{1B_4}$ . . . $u_p^{n4B_4}$) and ($u_c^{m4B_4}$ . . . $u_c^{1B_4}$).

A byte, say c_1, using the first member of the primary sub-pattern, u_p^1, is instantiated to c_1'. The first byte of key, B_1, using its first member of the primary sub-pattern, $u_p^{1B_1}$, is instantiated to B_1'. The other three bytes are also instantiated into B_2', B_3', and B_4' using their first member of the primary sub-patterns, $u_p^{1B_2}, u_p^{1B_3}, and\ u_p^{1B_4}$, respectively. The Hamming distance of *HD*(*c*', B_j'), for j=1 to 4, are measured and *B*'=Argmax[*HD*(*c*', B_j'), for j=1 to 4] is the key image. In the case that there are ties, the priority is given to the instantiated byte of B_1, B_2, B_3, and B_4 (and in that order.) The final substitution for c_1 is:

$$c_1''=(c_1' \oplus B') \tag{1}$$

The next byte, c_2, within a given segment of ciphertext is instantiated to c_2' using u_p^2, and key bytes of B_1, B_2, B_3, and B_4 are instantiated to B_1', B_2', B_3', and B_4' using $u_p^{2B_1}, u_p^{2B_2}, u_p^{2B_3}, and\ u_p^{2B_4}$, respectively.

B'=Argmax[*HD*(*c*', B_j'), for j=1 to 4] and c_2''=(c_2'⊕ *B*'). The process continues until the segment of the ciphertext is exhausted. The bytes of the next sub-list and the key bytes are instantiated using the complementary sub-patterns. Therefore, the sub-patterns are alternatively used for consecutive segments of the ciphertext.

Using the mixed substitution, the number of possible combinations for each key byte is equal to X and for the key of four bytes is X^4 (>1.19*10^{31} combinations.) Reader needs to be reminded that the four-byte key may be expanded to the length of N bytes for which the outcome of XOR is one of the X^{N+1} possible combinations. For N=16 (128-bit key) The XOR is one of the X^{17} possible combinations (>4.65*10^{105}.)

3) Patterns' Profile: Considering both self and mixed substitutions, the masking parameters grow to fifteen: (K, M, and R) for the instantiation of a byte of segment and (K_j, M_j, and R_j, for j=1 to 4) for instantiation of the four bytes of the key. Therefore, a pattern profile includes the fifteen parameters, which are accommodated by a 96-bit long binary string as described below.

Since the possible values for each of the parameters K and K_j is nine (0 through 8), the value of each parameter can be accommodated by 4 bits (the total of 20 bits). The parameters M and M_j have eight possible values (1 through 8) and each parameter can be accommodated by 3 bits (the total of 15 bits). The parameters R and R_j need eight bits each (the total of 40 bits). In addition, we use sixteen bits as the *Flag bits* and another five bits as the *Preference bits*.

The flag bits represent a decimal number (Δ) in the range of (0: 65,535). Let us assume that the length of the ciphertext that is ready to be masked is L_{ct}. Three bytes of f_1, f_2, and f_3 of the ciphertext are flagged which are in locations: δ_1= δ, $\delta_2=\lfloor L_{ct}/2+\delta/2 \rfloor$, and $\delta_3=L_{ct}$ - δ, where, δ is calculated using formula (2)

$$\delta=\begin{cases} \Delta\ Mod\ L_{ct}, & \Delta > L_{ct} \\ L_{ct}\ Mod\ \Delta, & \Delta \leq L_{ct} \end{cases} \tag{2}$$

The flagged bytes will not be masked during the vaccination process and they collectively make the *native byte* of F=(f_1⊕f_2⊕f_3). Since the length of the ciphertext and the length of its masked version remain the same there is no need for including the length of the ciphertext in the profile. The question of why the flagged bytes are of interest will be answered shortly.

The purpose of preference bits is to build a *model* which is influenced by both the key and flagged bytes. The model is used to create variable length blocks and segments. To build the model, a desired byte number (z) of the key is identified by the four least significant bits of the preference bits. That is, one can select any byte from a a maximum of 16-byte long key. (If a longer than 16-byte key is used, the number of bits for the preference bits needs to be increased.) The key is treated as circular and the two pairs of bytes of $A_1=(z+1||z)$ and $A_2=(z+2||z-1)$ are selected from key. A new pair of bytes of $A_3=A_1\oplus A_2\oplus(F||F)$ is built. If the most significant bit of the preference bits is set to zero then, model is A_3; otherwise, the model is $a_1\oplus a_2$, where, a_1 and a_2 are the pair of bytes in A_3.

Let us assume that there are two similar ciphertexts of CT_1 and CT_2 and we are using the same key and the same profile to mask the two ciphertexts, separately, using Vaccine. As long as one of the three flagged bytes in CT_1 and CT_2 is different the native bytes and, therefore, the models of the two ciphertexts are different and so their masked versions. This is one of the major advantages of Vaccine.

To summarize, the number of bits needed for the pattern profile is 96 bits (or 24 hex digits.) Dissection of a pattern profile is shown in Figure 2. The 24 hex digits representing the pattern profile along with eight hex digits representing the key may be sent to the receiver in advance or they may hide in the masked ciphertext itself:

a. In a predefined location/locations,
b. In location/locations determined by the internal representation of the key following some formula(s), or
c. A mixture of (a) and (b).

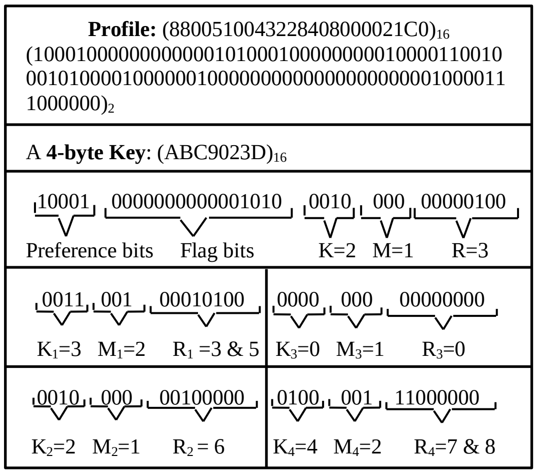

Figure 2. Dissection of the pattern's Profile of Interest

B. Vaccine

Vaccine is a variable-block cipher methodology capable of masking and unmasking a ciphertext. The details of masking and unmasking of Vaccine are presented in the following next two subsections.

1) Masking of the Ciphertext: Vaccine as a masking mechanism is able to mask the features of a ciphertext in the eye of a text miner. Vaccine: (1) divides the ciphertext into random size blocks, (2) each block, in turn, is divided into a number of segments such that the length of each segment is random, and (3) every byte within each segment is randomly instantiated to another byte using self and mixed substitutions. The masking process is presented shortly and it is encapsulated in algorithm Mask shown in Figure 3.

The algorithm is made up of four sections. In section one, (Step 1 of the algorithm) the profile is dissected to extract masking parameters and they, in turn, generate primary and complementary sub-patterns for five patterns:($Pattern_p^0$, $Pattern_c^0$), ($Pattern_p^{B1}$, $Pattern_c^{B1}$), ($Pattern_p^{B2}$, $Pattern_c^{B2}$), ($Pattern_p^{B3}$, $Pattern_c^{B3}$), and ($Pattern_p^{B4}$, $Pattern_c^{B4}$) used for masking the chosen byte of the ciphertext and the four key bytes, respectively. The array of pt with five elements keeps track of those primary and complementary sub-patterns of the five patterns that are in use. The model is also extracted in this step.

```
Algorithm Mask
Input:    A 32–bit key, a pattern's profile of 96-bit, and a
          ciphertext, CT.
Output:   Delivering IC as the masking version of CT.
Method:
 Step1- //Dissection of the profile and initializations
    Dissection delivers primary and secondary sub-patterns of
    five patterns ($Pattern_p^0$, $Pattern_c^0$), ($Pattern_p^{B1}$,
    $Pattern_c^{B1}$), ($Pattern_p^{B2}$, $Pattern_c^{B2}$), ($Pattern_p^{B3}$,
    $Pattern_c^{B3}$), and ($Pattern_p^{B4}$, $Pattern_c^{B4}$).
    κ ←Model obtained by using Preference bits, Flag bits,
          and key;
    IC ← “”; C ← CT;
    pt[5]← 0;//pt gives turn to the primary (pt[•]=0) and
          complementary (pt[•] =1) sub-patterns of the five
          patterns for initializing  the CurrentP [5];
    Step 2-Repeat until C is exhausted
       a- Get the set of decimal numbers from κ in ascending
          order: D ={$d_1$, $d_2$, . . . $d_{y-1}$, $d_y$};
          Get the next random size block,
          $\beta_n$,=Substr(C, 0, $d_y$);
       b- CL = 0; //Current location in C
       c- Repeat for i =1 to y-1
          //Divide $\beta_n$ into y-1 segments;
          $s_i$ = Substr($\beta_n$, CL, $d_i$ - CL);
          CL = CL+ $d_i$;
          CurrentP[m]=$Pattern_{pt}^m$  //for m =0 to 4;
          d- Repeat for each byte, $c_j$, in $s_i$
             $d_1$- If ($c_j$ is a flagged byte) Then continue;
             $d_2$- If (CurrentP[0] is exhausted)
                  Then CurrentP[0] = $Pattern_{pt}^0$;
             $d_3$- $c_j$' = Flip $c_j$ bits using CurrentP[0];
             $d_4$- $c_j$' = Circularly swap proper $c_j$ bits using
                       CurrentP[0];
             $d_5$- σ = Select($c_j$', CurentP[1],
                  CurrentP[2],CurrentP[3],CurrentP[4]);
             $d_6$- a = $c_j$' ⊕ σ;
             $d_7$- IC←IC || a;
          End;
        pt[•]++; pt[•] ← pt[•] mode 2;
       End;
       e- Remove block $\beta_n$ from C;
       f- Apply one-bit-left-rotation on κ;
    End;
End;
```

Figure 3. Algorithm Mask

The second section (Step 2.a of the algorithm) identifies a random size block prescribed by κ—the model. The identification process is done by creating y binary numbers using κ. The i-th binary number starts from the least significant bit of the κ and ends at the bit with the i-th value of "1" in κ. The binary numbers are converted into decimal numbers and sorted in ascending order, $\{d_1, d_2, \ldots d_{y-1}, d_y\}$. The block, β_n=Substr(C, 0, d_y), where C is initially a copy of the cipher text.

The third section (Step 2.c of the algorithm) divides block β_n into a number of random size segments. The size and the number of segments are dictated by κ internal representation. Block β_n has y segments: $\{ s_0 \ldots s_{y-1}\}$.

The segment s_i starts from the first byte after the segment s_{i-1} (the location is preserved in variable CL) and contains $\lambda_i=d_{i+1} - d_i$ bytes. The number of segments and their lengths are not the same for different blocks.

To get the next block of the ciphertext, the block β_n is removed from C (Step 2.e) and κ is changed by having a one-bit-left-rotation (Step 2.f). Using the above process along with new κ, the next block with a different size is identified. This process continues until C is exhausted. It is clear that the lengths of blocks are not necessarily the same. In fact, the lengths of blocks are random. It needs to be mentioned that length of the block β_i and β_{i+8} are the same when κ is one byte long. When κ is two bytes long, the length of the block β_i and β_{i+16} are the same. And a block on average is 32,768 bytes long. As a result, the ciphertext, on average, must be longer than 491,520 bytes before the blocks' lengths are repeated.

```
Algorithm Select
Input: A byte (c), Key, and four patterns for the four key bytes.
Output: key image, k.
Method:
    a. Repeat for (w = 1 to 4)
        | If (CurrentP[w] is exhausted)
        | Then CurrentP[w] = $Pattern_{pt}^{w}$;
       End;
    b. h ← -1;
    c. Repeat for v= 1 to 4;
        | i.  $c_v$ ← An instantiated version of KeyByte$_v$ using
        |     related sub-pattern.
        | ii. If HD(c, $c_v$) >h //HD is Hamming distance function
        |     Then      h = HD(c, $c_v$); k = $c_v$;
      End;
End;
```

Figure 4. Algorithm Select

The fourth section (Step 2.d of the algorithm) delivers the masked version of the ciphertext, byte by byte, for a given segment. Flagged bytes are not masked (Step 2.d_1). If the number of bytes in the segment s_i is greater than the cardinality of the pattern then, the pattern repeats itself (Step 2.d_2). Each byte, c_j, of the segments s_i (for i=1 to y-1) are masked by applying (i) the relevant member of the current sub-pattern on byte c_j (Step 2.d_3 and 2.d_4), (ii) identifying the key image (Step 2.d_5), by invoking the Algorithm Select (Figure 4), (iii) create c_j', the masked version of the c_j, by XORing the outcome of process (i) and process (ii), (Step 2.d_6), and (iv) concatenate the masked version of the c_j, to string of IC which ultimately becomes the inoculated version of the inputted ciphertext (Step 2.d_7).

2) Unmasking of the Ciphertext: For unmasking a masked ciphertext, those steps that were taken during the masking process are applied in reverse order. Therefore, the Algorithm Mask with a minor change in step 2.d can be used for unmasking. We show only the changes to Step d of Figure 3 in Figure 5.

```
d- Repeat for each byte, $c_j$', in $s_i$
    $d_1$- If ($c_j$ is a flagged byte) Then continue;
    $d_2$- If (CurrentP[0] is exhausted)  Then CurrentP[0] = $Pattern_{pt}^{0}$;
    $d_3$- σ = Select($c_j$', CurentP[1], CurrentP[2], CurrentP[3], CurrentP[4];
    $d_4$- α = $c_j$' ⊕ σ;
    $d_5$- α = Circularly swap bits of α using CurrentP[0];
    $d_6$- α = Flip a bits using CurrentP[0];
    $d_7$- UM←UM||α; //UM is the unmasked ciphertext;
```

Figure 5. The modified part of the Algorithm Mask

IV. EMPIRICAL RESULTS

To measure the effectiveness of the proposed Vaccine, we compare its performance with the performance of the well-established masking algorithms of CBC and CFB. The behavior of Vaccine was observed using three separate profiles of simple, moderate, and complex. These observations are named VAC_s, VAC_m, and VAC_c.

Two plaintext templates of *natural* and *synthetic* were chosen and 100 plaintexts were generated for each template. Each plaintext following the first template was selected from a natural document made up of the lower and upper case alphabets and the 10 digits—total of 62 unique symbols. Each plaintext following the second template was randomly synthesized using the10 symbols set of {A, b, C, L, x, y, 0, 4, 6, 9}. The goal was to synthesize plaintexts with high occurrences of a small set of symbols. Each plaintext created under both templates was 1K bytes long.

For each plaintext, two ciphertexts of C_a and C_d were generated using Advanced Encryption System (AES-128) and Data Encryption Standard (DES) algorithms [12][13][14]. The masking approaches of CBC, CFB, VAC_s, VAC_m, and VAC_c were applied separately on C_a and C_d generating the masked ciphertexts of:

$\{C_a^{cbc}, C_a^{cfb}, C_a^{vac_s}, C_a^{vac_m}, C_a^{vac_c}\}$ and
$\{C_d^{cbc}, C_d^{cfb}, C_d^{vac_s}, C_d^{vac_m}, C_d^{vac_c}\}$.

When CFB applied on C_a and C_d the key lengths were 64-bit and 128-bit, respectively, and IV chosen from a natural document. (The least significant 64 bits of the 128-bit key was used as the key when CFB was applied on C_a. The key used by VAC_s, VAC_m, and VAC_c was also borrowed from the least significant 32 bits of the 128-bit key used for CFB.)

Let us consider the first set of masked ciphertexts $\{C_a^{cbc}, C_a^{cfb}, C_a^{vac_s}, C_a^{vac_m}, C_a^{vac_c}\}$ generated from C_a. The following steps are used to compare the effectiveness of the proposed Vaccine with CBC and CFB. (The same steps are also followed to compare the effectiveness of the proposed Vaccine with CBC and CFB using the masked ciphertexts of $\{C_d^{cbc}, C_d^{cfb}, C_d^{vac_s}, C_d^{vac_m}, C_d^{vac_c}\}$.)

a. Get the list of unique symbols that the plaintext is made up of, List=$\{\sigma_1 \ldots \sigma_m\}$.
b. Get the frequency of symbol σ_i, for i = 1 to m, and calculate the average frequency of the symbols.
c. Repeating the next two steps for every symbol, σ_i, in the list.
d. Identify the locations for all the occurrences of the symbol, σ_i, in the plaintext, $(\ell^i_1 \ldots \ell^i_n)$.
e. Identify the bytes in the locations of $(\ell^i_1 \ldots \ell^i_n)$ within the $C_a^\bullet$ and calculate the Hamming distance, h_j, between the two bytes in location ℓ_j, for j=1 to n, in the plaintext and $C_a^\bullet$. The overall average of Hamming distance for the symbol σ_i is $h_{\sigma i}$ =Average$(h_1 \ldots h_n)$,
f. Concluding that the underline masking methodology with the highest average values of the Hamming distances have a superior performance.

The outcome of applying the above steps on the ciphertexts of $\{C_a^{cbc}, C_a^{cfb}, C_a^{vac_s}, C_a^{vac_m}, C_a^{vac_c}\}$ and $\{C_d^{cbc}, C_d^{cfb}, C_d^{vac_s}, C_d^{vac_m}, C_d^{vac_c}\}$ are shown in Table 1.a and Table 1.b. We have also used the system clock to calculate the average throughput (in millisecond) for the masking approaches of CBC, CFB, VAC_s, VAC_m, and VAC_c and reported in Tables 2.a and 2.b.

TABLE I. AVERAGE OF HAMMING DISTANCES BETWEEN THE TWO 100 PLAINTEXTS OF 1K BYTE LONG (GENERATED BY TWO TEMPLATES) AND THEIR RELATED MASKED CIPHERTEXTS: (A) ENCRYPTED BY AES AND (B) ENCRYPTED BY DES

Tem.	Avg. Symb. Freq.	AES-128				
		CBC	***CFB128***	***VAC_s***	***VAC_m***	***VAC_c***
		Dist.	**Dist.**	**Dist.**	**Dist.**	**Dist.**
Syn.	103	3.568	3.570	4.415	4.373	4.411
Natu.	16.5	3.569	3.561	4.423	4.361	4.411
(a)						
Tem.	**Avg. Symb. Freq.**	**DES**				
		CBC	***CFB64***	***VAC_s***	***VAC_m***	***VAC_c***
		Dist.	**Dist.**	**Dist.**	**Dist.**	**Dist.**
Syn.	103	3.527	3.526	4.182	4.153	4.223
Natu.	16.5	3.513	3.515	4.176	4.141	4.221
(b)						

In addition, a *masking strength* of μ (0 < μ < 1), is introduced that is defined as $\mu = N_{inst} / N_{occ}$, where N_{inst} is the number of unique bytes in the masked ciphertext representing the instantiations of the N_{occ} occurrences of symbol σ_i in the underlying plaintext of the masked ciphertext. The masking strength for CBC, CFB, VAC_s, VAC_m, and VAC_c are presented in Tables 3.a and 3.b.

V. CONCLUSIONS AND FUTURE RESEARCH

The performance of the presented new cipher block approach, Vaccine, for masking and unmasking of ciphertexts seems superior to the performance of the well-known masking approaches of CBC and CFB.

TABLE II. THROUGHPUT AVERAGE IN MILISECOND FOR THE TWO 100 PLAINTEXTS OF 1K BYTE LONG (GENERATED BY TWO TEMPLATES): (A) ENCRYPTED BY AES AND (B) ENCRYPTED BY DES

Tem.	Avg Symb. Freq.	AES-128				
		CBC	***CFB128***	***VAC_s***	***VAC_m***	***VAC_c***
		TPut.	***TPut.***	***TPut.***	***TPut.***	***TPut.***
Syn.	103	4545	11111	25000	33334	20000
Natu.	16.5	12500	10000	16667	20000	12500
(a)						
Tem.	**Avg Symb. Freq.**	**DES**				
		CBC	***CFB64***	***VAC_s***	***VAC_m***	***VAC_c***
		TPut.	***TPut.***	***TPut.***	***TPut.***	***TPut.***
Syn.	103	3846	11111	20000	25000	14286
Natu.	16.5	10000	10000	14286	20000	11111
(b)						

TABLE III. AVERAGE MASKING STRENGTH FOR THE TWO 100 PLAINTEXTS OF 1K BYTE LONG (GENERATED BY TWO TEMPLATES): (A) ENCRYPTED BY AES AND (B) ENCRYPTED BY DES

Tem.	Avg. Symb. Freq.	AES-128				
		CBC	***CFB128***	***VAC_s***	***VAC_m***	***VAC_c***
		μ	μ	μ	μ	μ
Syn.	103	0.506	0.486	0.451	0.540	0.571
Natu.	16.5	0.882	0.878	0.845	0.890	0.889
(a)						
Tem.	**Avg. Symb. Freq.**	**DES**				
		CBC	***CFB64***	***VAC_s***	***VAC_m***	***VAC_c***
		μ	μ	μ	μ	μ
Syn.	103	0.501	0.494	0.490	0.564	0.570
Natu.	16.5	0.878	0.894	0.880	0.909	0.893
(b)						

The advantages of Vaccine over CBC and CFB are numerated as follows:

a. The key and patterns' profile may hide in the masked ciphertext.
b. The block size for Vaccine is not fixed and it is selected randomly.
c. Each block is divided into segments of random size.
d. The masking pattern changes from one byte to the next in a given segment.
e. Masking a ciphertext using Vaccine demands mandatory changes in the ciphertext. Therefore, the identity transformation could not be provided through the outcome of Vaccine. The simple proof is that the Hamming weight is modified.
f. The results revealed that on average:
    i. The Hamming distance between masked and unmasked occurrences of a byte using Vaccine is 0.72 bits higher than using CBC and CFB.
    ii. Vaccine throughput is 3.4 times and 1.8 times higher than throughput for CBC and CFB.
    iii. Vaccine masking strength is 1.5% and 1.8% higher than masking strength for CBC and CFB.

iv. VAC_m masking strength is 3.6% and 3.7% higher than masking strength for CBC and CFB. And VAC_c masking strength is 3.9% and 4.2% higher than masking strength for CBC and CFB.

As the future research, building a new version of Vaccine is in progress to make the throughput and the masking strength of the methodology even higher. In addition, the use of Vaccine in a parallel processing environment also will be investigated. In addition, a feasibility study for using Vaccine as an authentication method is in progress.

REFERENCES

[1] A. A. Rasheed, M. Cotter, B. Smith, D. Levan, and S. Phoha, "Dynamically Reconfigurable AES Cryptographic Core for Small, Power Limited Mobile Sensors", The 35th IEEE International Performance Computing and Communication Conference and Workshop, pp. 1-7, 2016.

[2] G. P. Saggese, A. Mazzeo, N. Mazzocca and A. G. M. Strollo, "An FPGA-based performance analysis of the unrolling, tiling, and pipelining of the AES algorithm", LNCS 2778, pp. 292-302, 2003.

[3] N. Pramstaller and J. Wolkerstorfer, "A Universal and Efficient AES Co-processor for Field Programmable Logic Arrays", Lecture Notes in Computer Science, Springer, Vol.3203, pp. 565-574, 2004.

[4] B. Moeller. *Security of CBC Cipher suites in SSL/TLS: Problems and Countermeasures. [Online]*. Available from: https://www.openssl.org/~bodo/tls-cbc.txt

[5] W. Stallings, "Cryptography and Network Security: Principles and Practice", Pearson, 2014.

[6] C. A. Henk and V. Tilborg, "Fundamentals of Cryptology: "A Professional Reference and Interactive Tutorial", Springer Science & Business Media, 2006.

[7] N. Feruson, B. Schneier, and T. Kohno, "Cryptography Engineering: Design Principles and Practical Applications", Indianapolis: Wiley Publishing, Inc., pp. 63-64, 2010.

[8] W. F. Ehrsam, C. H. W. Meyer, J. L. Smith, and L. W. Tuchman, "Message Verification and Transmission Error Detection by Block Chaining", US Patent 4074066, 1976.

[9] C. Kaufman, R. Perlman, and M. Speciner, "Network Security", 2nd ed., Upper Saddle River, NJ: Prentice Hall, p. 319, 2002.

[10] National Institute of Standards and Technology (NIST), Advanced Encryption Standard (AES), Federal Information Processing Standards Publications 197 (FIPS197), Nov. 2001.

[11] S. Vaudenay, "Security Flaws Induced by CBC Padding — Applications to SSL, IPSEC, WTLS....", Lecture Notes in Computer Science, Springer, vol. 2332, pp. 534-546, 2002.

[12] H. Kuo-Tsang, C. Jung-Hui, and S. Sung-Shiou, "A Novel Structure with Dynamic Operation Mode for Symmetric-Key Block Ciphers", International Journal of Network Security & Its Applications, Vol. 5, No. 1, p. 19, 2013.

[13] H. Feistel, "Cryptography and Computer Privacy", Scientific American, Vol. 228, No. 5, pp 15–23, 1973.

[14] F. Charot, and E. Yahya, and C. Wagner, "Efficient Modular-Pipelined AES Implementation in Counter Mode on ALTERA FPGA", (FPL 2003), Lisbon, Portugal, pp. 282-291, 2003.