A Novel Cryptosystem Based on Steganography and Automata Technique for Searchable Encryption

Nguyen Huy Truong*
School of Applied Mathematics and Informatics, Hanoi University of Science and Technology, Vietnam
truong.nguyenhuy@hust.edu.vn
*Corresponding author: Nguyen Huy Truong

Received July 5, 2019; revised September 8, 2019; accepted April 1, 2020; published May 31, 2020

Abstract

In this paper we first propose a new cryptosystem based on our data hiding scheme $(2,9,8)$ introduced in 2019 with high security, where encrypting and hiding are done at once, the ciphertext does not depend on the input image size as existing hybrid techniques of cryptography and steganography. We then exploit our automata approach presented in 2019 to design two algorithms for exact and approximate pattern matching on secret data encrypted by our cryptosystem. Theoretical analyses remark that these algorithms both have $O(n)$ time complexity in the worst case, where for the approximate algorithm, we assume that it uses $\lceil (1-\varepsilon)m \rceil$ processors, where ε, m and n are the error of our string similarity measure and lengths of the pattern and secret data, respectively. In searchable encryption, our cryptosystem is used by users and our pattern matching algorithms are performed by cloud providers.

Keywords: Searchable Encryption, Cryptography, Steganography, Pattern Matching, Automata Technique.

1. Introduction

1.1 Background

Nowadays, with the rapid development of applications based on Internet infrastructure, cloud computing becomes one of the hottest topics in the information technology area. Indeed, it is a computing system based on Internet that provides on-demand services from application and system software, storage to processing data. For example, when cloud users use the storage service, they can upload information to the servers and then access it on the Internet online. Meanwhile, enterprises can not spend big money on maintaining and owning a system consisting of hardware and software. Although cloud computing brings many benefits for individuals and organizations, cloud security is still an open problem when cloud providers can abuse their information and cloud users lose control of it. Thus, guaranteeing privacy of tenants' information without negating the benefits of cloud computing seems necessary [8, 11, 12, 13, 16, 32, 33].

In order to protect cloud users' privacy, sensitive data need to be encoded before outsourcing them to servers. Unfortunately, encryption makes the servers perform search on ciphertext much more difficult than on plaintext. To solve this problem, many searchable encryption (SE) techniques have been presented since 2000. SE does not only store users' encrypted data securely but also allows information search over ciphertext [7, 8, 9, 11, 12, 16, 19, 22, 32].

In cryptography, SE is a cryptosystem such that search can be done on encrypted data directly. SE can be either searchable symmetric encryption (SSE) or searchable asymmetric encryption (SAE). In SSE, only private key holders can create encrypted data and produce trapdoors for search. In SAE, users who have the public key can make ciphertexts but only private key holders can generate trapdoors [7, 8, 12, 16, 32].

Many SE methods pay attention to the problem of searching for pre-chosen keywords in ciphertext. For this problem, suppose that each data (document) contains a set of keywords. Then there are two approaches to SE. First way is to create an index which contains keywords and the corresponding document (forward index) or a keyword and the corresponding documents (inverted index). Second way is to do a sequential search without an index. Recently, to perform search more flexibly and keep away from wrong or no matching results, apart from traditional solutions only providing exact keyword search, the development of new methods supporting approximate (fuzzy) keyword search has been also studied [7, 8, 9, 11, 12, 13, 16, 19, 22, 25, 32].

However, the keyword based SE faces a problem. Keywords must be determined and also encrypted in a form, and then all files encrypted will be uploaded to the cloud. Then searching for new keywords can follow false results, even if the user data contains these keywords but not mentioned in the set of defined keywords. Furthermore, for the index base SE, very large indexes would make the efficiency of keyword search low [7, 12, 13, 19].

To deal with the above problem, there are some SE techniques proposed such as supporting file update functionality [9, 12, 16, 32], creating index file small [19] and providing pattern matching for search pattern is only asked at search time [7, 13, 25].

As we know that pattern matching is applied to search for information and analyze data every day, for example find and replace in text editing systems, in the search engine Google, database queries, searching on genomic data, etc [7, 26, 28].

Here, our work takes an interest in the problem of pattern matching on encrypted data, which is an important research direction in SE.

In spite of the considered problem's importance, it has not been invested properly. To the best of our knowledge, there have only existed a few SE methods for exact pattern matching, but not for approximate pattern matching. Haynberg et al. [13] introduced SSE for exact pattern matching by using directed acyclic word graphs in the encryption algorithm (for more details about this data structure, see [3]). However, their technique needs the partial decryption of the ciphertext, it follows that the plaintext would be leaked to the attackers. Further, the searching is performed on users. Strizhov et al. [25] allowed pattern search on ciphertexts using the position heap tree data structure (see [10] for more details about the position heap). For this method, server does not perform search on encrypted data directly but only on index constructed from secret data. Desmoulins et al. [7] proposed SE for exact pattern matching, where the search phase Test is a pattern matching algorithm whose time complexity is *O(mn)* in the worst case for *m*, *n* are lengths of the pattern and the secret data.

The goal of this paper is to propose a novel symmetric cryptosystem that is used on users side, and algorithms for exact and approximate pattern matching on ciphertexts which are used on cloud servers side. These are essential components in SSE.

As we know that cryptography and data hiding are two branches of information security. Cryptography is used to distort data such that the data is not understood by attackers, it includes symmetric and asymmetric cryptography. While data hiding is used to hide data in digital media such as image, audio, video files, etc. It can be classified into steganography that protects secret data by hiding the existence of them and watermarking that prevents digital media by embedding watermarks in them [6, 34, 27].

Although cryptography and steganography are both capable of protecting secret data separately, different combinations of them are being developed to create systems with better security. The well studied hybrid technique of cryptography and steganography is to encode secret data using cryptography and then embed the ciphertext using steganography [2, 5, 30]. For gray images, Song et al. [23] introduced the first method which encrypts and embeds at once. However, since these methods must all guarantee acceptable imperceptibility of the digital media, the total number of secret data hidden is limited by the size of them. In other digital media formats, image steganography is used the most popularly because digital images are often transmitted on Internet and they have high degree of redundancy. Furthermore, the technique of image steganography is mainly image steganography in spatial domain [4, 14, 31]. So, to address the limitation of the existing hybrid methods, we propose a novel approach to construct a new cryptosystem based on spatial domain image steganography.

In our approach, we use the data hiding scheme (2,9,8) that is block based method in spatial domain, where 9 is the number of pixels in any image block, 8 is the number of secret bits which can be embedded in a block by changing colors of at most 2 pixels in the block. This scheme is near optimal for gray and palette images with high efficiency in embedding capacity, speed, security as well as visual quality, which are main properties of data hiding schemes [27]. Since our cryptosystem is designed to solve the problem of pattern matching on encrypted data and for an assumption that secret data is a string over the alphabet of size 256, the cryptosystem allows to encrypt letters of the secret data one by one.

For a given letter in the alphabet, corresponding to a 8-bit string, based on the embedding function in the data hiding scheme (2, 9, 8), we compute the information (called the flip information) to change the input image block. This flip information consists of positions of pixels in the block and way changing color of these pixels. The code word of the letter is a binary string presenting the flip information. The security analyses show that our cryptosystem provides high security with a key space of $2^{20}3^{9}9!2^{90}2^{8}!$ for gray images and $2^{20}3^{9}9!2^{18+9t}2^{8}!$ for palette images, where *t* is the number of bits representing color indexes.

Return to the remaining main objective in our work which is the problem of pattern matching on encrypted data on cloud providers side. In our results introduced [28, 29], automata technique was applied to the problem of exact pattern matching and the longest common subsequence problem on plaintexts. With using this technique, we have achieved aims which are to design effective algorithms in practice to solve these problems. In this paper, we apply the algorithms to constructing exact and approximate pattern matching algorithms on ciphertexts performed by severs. Our main idea is to encrypt the automaton corresponding to a given pattern, and then we consider this encrypted automaton as a part of the trapdoor. Some theoretical analyses remarked that our algorithms all have *O(n)* time complexity in the worst case, where for approximate algorithm, we need an assumption that it uses $\lceil (1-\varepsilon)m \rceil$ processors, where ε, m and n are error of the string similarity measure and lengths of the pattern and secret data, respectively.

1.2 Contributions

Our contributions in this paper can be summarized as follows.

1. We propose a novel approach to construct a cryptosystem based on steganography. The outstanding advantages of our cryptosystem are to allow encoding and decoding done at once, and ciphertexts that do not depend on the input image size as existing hybrid techniques of cryptography and steganography. In particular, this cryptosystem can be applied in SSE to encrypt and decrypt secret data by users.
2. We propose two sequential and parallel algorithms for exact and fuzzy pattern matching on secret data encrypted by our cryptosystem. These algorithms can be used by servers in SSE to perform pattern search. The outstanding feature of the algorithms is that, they can be applied sufficiently to all cryptosystems that support encrypting letters of the secret data one by one.

1.3 Organization

We organize the rest of this paper as follows. In Section 2, we recall some terminologies, definitions and results in [24, 27, 28, 29]. Section 3 proposes algorithms used by users and servers in SSE, we first construct a novel cryptosystem based on the data hiding scheme in [27], apply this cryptosystem to the process of encrypting and decrypting a secret data sequence and some security analyses are also discussed in Subsection 3.1. We then use the automata approach in our previous researches [28, 29] to design exact and approximate pattern matching algorithms on secrete data encrypted by our cryptosystem in Subsection 3.2. Finally, Section 4 provides our conclusions.

2. Preliminaries

In this section, we will attempt to recall terminologies, definitions and results in [24, 27, 28, 29], which are really needed in order to present our new results clearly and logically, as well as help readers follow our paper's content easily.

Now, we start with our near optimal data hiding scheme (2, 9, 8) proposed in [27], one of our data hiding schemes based on the Galois field $GF(2^2)$, constructed from the polynomial ring $Z_2[x]$ [24]. This scheme is a core material for constructing our new cryptosystem.

The data hiding scheme (2, 9, 8) is a five tuple $(\mathcal{I}, \mathcal{M}, \mathcal{K}, Em, Ex)$, where $\mathcal{I}$ is a set of all image blocks of 9 pixels with the same image format, $\mathcal{M} = GF^4(2^2)$, $\mathcal{K}$ is a finite set of secret keys of 9 elements of $GF(2^2)$ and two functions *Em* and *Ex* are designed as follows [27].

Without loss of generality, we assume that for $I \in \mathcal{I}$, $K \in \mathcal{K}$, they can be given by

$$I = \{I_1, I_2, ..., I_9\},$$

where I_i is a color index in the palette for palette images or color value for gray images of the i^{th} pixel in *I*, $\forall i = \overline{1,9}$.

$$K = \{K_1, K_2, ..., K_9\}$$

with $K_i \in GF(2^2), \forall i = \overline{1,9}$.

Given a secret element $M \in \mathcal{M}$, an image block $I \in \mathcal{I}$, a key $K \in \mathcal{K}$. Let *G* be the flip graph for gray and palette images constructed as in [27] and the automaton $A(I, M, K)$ defined as in [27]. With q_0 is the initial state and δ is the state transition function of $A(I, M, K)$, $Adjacent(I_{i_t}, a_t)$ is an adjacent vertex of the vertex I_{i_t} in *G* and a_t is the weight of the arc $(I_{i_t}, Adjacent(I_{i_t}, a_t))$. Then we have [27]

The embedding function *Em* embedding *M* in *I*:

```
$q = q_0$;                                                    (2.1)
For $i$ = 1 to 9 Do $q = \delta(q, I_i)$;                     (2.2)
$q = \delta(q, M)$;                                           (2.3)
For each $(i_t, a_t)$ in $q$ Do $I_{i_t}$ = Adjacent$(I_{i_t}, a_t)$;   (2.4)
$I' = I$;                                                     (2.5)
```

The extracting function *Ex* extracting *M* from I':

```
$q = q_0$;                                                    (2.6)
For $i$ = 1 to 9 Do $q = \delta(q, I'_i)$;                    (2.7)
$M = q$;                                                      (2.9)
```

Remember that the data hiding scheme (2,9,8) means we can hide a secret string of length 8 bits in an image block of 9 pixels with at most 2 pixels modified.

According to our construction of the data hiding scheme (2,9,8) and assume that we publish parameters *Em*, *Ex*, the vector space $GF^4(2^2)$ and the flip graph *G* in this scheme. Then the security of the data hiding scheme (2,9,8) is given by the following formula [27]

$$c3^9 9! 2^{18} 2^8!, \text{ where } c \approx 2^{20}. \tag{2.10}$$

We then recall the components and properties of a cryptosystem in [24].

Definition 2.1 ([24]). A five tuple $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ is called a cryptosystem if the following properties hold.

1. $\mathcal{P}$ is a finite set of plaintexts.
2. $\mathcal{C}$ is a finite set of ciphertexts.
3. $\mathcal{K}$ is a finite set of secret keys.
4. For $\forall k \in \mathcal{K}$, there exists an encrypting function $e_k \in \mathcal{E}$ and a corresponding decrypting function

$d_k \in \mathcal{D}$, where $e_k : \mathcal{P} \rightarrow \mathcal{C}$ and $d_k : \mathcal{C} \rightarrow \mathcal{P}$ holds $d_k(e_k(x)) = x$ with $\forall x \in \mathcal{P}$.

Next, we present main terminologies and facts in [28] to design the exact pattern matching on encrypted data.

We call a finite set Σ an alphabet, denote the size of Σ by $|\Sigma|$. Any element in Σ is called a letter. A string on Σ is a finite sequence of letters of Σ. Denote the set of all strings over Σ by Σ^*. The empty string is denoted by ε. The length of the string x, denoted by $|x|$, is the number of letters of x. For any string x of length n we can represent by

$$x = x[1]x[2]..x[n],\ x[i] \in \Sigma, 1 \le i \le n,$$

where n is a positive integer.

Denote the concatenation operator of the two strings u_1 and u_2 by u_1u_2.

A string p is called a substring of the string x, if $x = u_1pu_2$ for any strings u_1 and u_2. In case $u_1 = \varepsilon$ (resp. $u_2 = \varepsilon$), the string p is called a prefix (resp. suffix) of x. If $p \neq x$, the prefix (resp. suffix) p is called proper.

We denote the i^{th} element of x by $x[i]$ and i is called a position in x, the substring $x[i]x[i+1]..x[j]$ of x by $x[i..j]$ for $\forall 1 \le i \le j \le n$. Let p be a substring of length m of x, where m is a positive integer, then there exists i for $1 \le i \le n - m + 1$ such that $p = x[i..i + m - 1]$. We say that i is an occurrence of p in x or p occurs in x at position i.

Definition 2.2 ([28]). Given a string p and a letter a of p. Let i be some position in p for $1 \le i \le |p|$. Then call i the last position of appearance of a in p, denoted by $Pos_p(a)$ if $a = p[i]$ and $\forall j > i, j \le |p|, a \neq p[j]$.

Definition 2.3 ([28]). Let p be a pattern of length m over the alphabet Σ. Then $Next_p$ of p is a function such that $Next_p : \{1,...,m\} \rightarrow \{0,...,m - 1\}$ defined by $Next_p(l) = max\{|s| \ |s$ is both a suffix and proper prefix of $p[1..l]\}$ for $l \in \{1,...,m\}$.

Lemma 2.4 ([28]). *Let p be a pattern, x be a text over the alphabet Σ and suppose that the degree of appearance of p in x at the position i is equal to l,* $0 \le l \le |p|$. *Then the degree of appearance l' at the position i+1 in x is given by the formula l' =* $Appearance_p(l,a)$, *where* $a = x_{i+1}$, *and the function* $Appearance_p$ *corresponding to p is determined by*

$$Appearance_p(l,a) = \begin{cases} 0 & l = 0, a \neq p[1]; \text{or } a \notin p, \\ 1 & l = 0, a = p[1], \\ l+1 & 0 < l < |p|, a = p[l+1], \\ Appearance_p(Next_p(l), a) & 0 < l < |p|, a \neq p[l+1]; \text{or } l = |p|. \end{cases}$$

Theorem 2.5 ([28]). *Let p be a pattern of length m and* $A_p = (\Sigma, Q_p, q_0, \delta_p, F_p)$ *corresponding to p be an automaton over the same Σ, where*

- $Q_p = \{0,1,...,m\}$ *is a set of states,*
- $q_0 = 0$ *is the initial state,*
- $F_p = \{m\}$ *is a set of final states,*
- δ_p *is the transition function satisfying* $\delta_p : Q_p \times \Sigma \rightarrow Q_p$ *and* $\delta_p(q,a) = Appearance_p(q,a)$, *where the function* $Appearance_p$ *corresponding to p as given in Lemma 2.4,*
- *To accept an input string, we can extend the transition function* δ_p: $\delta_p : Q_p \times \Sigma^* \rightarrow Q_p$ *such that* $\forall q \in Q_p, \delta_p(q,\varepsilon) = q, \forall s \in \Sigma^*, \forall a \in \Sigma, \delta_p(q,as) = \delta_p(\delta_p(q,a),s)$.

Then the pattern p is accepted by the automaton A_p.

Finally, we recall important definitions in [29] to construct the approximate pattern matching on encrypted data.

Let $LCS(p, x)$ be a longest common subsequence of p and x. Denote $|LCS(p, x)|$ by $lcs(p, x)$. We let the $lcs(p, x)$ equal 0, if there does not exist any longest common subsequences of strings p and x (for more details about the concept "longest common subsequence of two strings," see [29]).

We see that a subsequence u has at least a location in p. Note that $u = p[j_1]p[j_2]..p[j_l]$ is a subsequence of p, then vector $(j_1,j_2,..., j_l)$ is a location of u in p. We sort all the different locations of u into the dictionary order, then call the leftmost location of u the least element, denoted by $LeftID(u)$. The last component in $LeftID(u)$ is denoted by $Rm_p(u)$ [29].

The symbol $Config(p)$ is the set of all configurations of p. If $C \in Config(p)$, then C can be the empty set, denoted by C_0, or C can be an ordered set $\{u_1,u_2,...,u_l\}$ with $1 \leq l \leq |p|$, where u_i is a subsequence of p, $1 \leq i \leq l$ (see more detail in [29]).

Definition 2.6 ([29]). Let p be a string of length m and $C \in Config(p)$. Then the weight of C is a ordered set, denoted by $W_p(C)$, is given by

(a) $W_p(C)$, denoted by W_0, is the empty set if $C = C_0$.

(b) $W_p(C) = \{W_p(u_1),W_p(u_2),...,W_p(u_l)\}$ if $C = \{u_1,u_2,...,u_l\}$ for $1 \leq l \leq m$, where the weight of u_i in p, denoted by $W_p(u_i)$ and $W_p(u_i) = |p| + 1 - Rm_p(u_i)$ for $1 \leq i \leq l$.

Denote the set of all the weights of all the configurations of p by $WConfig(p)$.

Definition 2.7 ([29]). Let p be a string of length m on the alphabet Σ and Σ_p be the set of all the letters of p. Then Ref_p of p is a function such that $Ref_p : \{1,...,m\} \times \Sigma_p \to \{1,...,m-1\}$ determined by

$$Ref_p(i,a) = \begin{cases} 0 & i = 1, \\ max\{W_p^j(a) \mid W_p^j(a) < i \ for \ m+1-i < j \leq m & 2 \leq i \leq m, \end{cases}$$

where $a \in \Sigma_p$, where the weight of the letter a at the location j in p, $W_p^j(a) = m + 1 - j$.

Notice that for an assumption that p contains a. With $1 \leq i \leq |p|$ if $a \neq p[i]$, we let $W_p^i(a) = 0$. At any location, the letter a has a weight. Denote the heaviest weight of a in p by $Wm_p(a)$ [29].

Definition 2.8 ([29]). Let p be a string of length m over the alphabet Σ, W be a weight of a configuration of p and $a \in \Sigma$. Then a function δ_p is given by $\delta_p : WConfig(p) \times \Sigma \to WConfig(p)$ and

1. If $a \notin p$, then $\delta_p(W,a) = W$.
2. If $a \in p$, then $\delta_p(W_0,a) = \{Wm_p(a)\}$.
3. Assume that $a \in p$ and $W = \{w_1,w_2,...,w_l\}$ for $1 \leq l \leq m$. Put $W' = \delta_p(W,a)$. Then W' is computed by the following parallel algorithm:
   (i) Put $W' = W$;
   Perform the block of the following commands in parallel way:
   (ii) $w'_{l+1} = Ref_p(w_l\ a)$ if $Ref_p(w_l,a) \neq 0$;
   (iii) The following commands are executed in parallel: for $\forall i \in \{1, 2,..., l - 1\}$, $w'_{i+1} = Ref_p(w_i,a)$ if $Ref_p(w_i,a) > w_{i+1}$;
   (iv) $w'_1 = Wm_p(a)$ if $Wm_p(a) > w_1$;
4. To accept an input string, we extend the function δ_p: $\delta_p : WConfig(p) \times \Sigma^* \to WConfig(p)$ such that $\forall W \in WConfig(p), \delta_p(W,\varepsilon) = W, \forall u \in \Sigma^*, \forall a \in \Sigma$, $\delta_p(W,au) = \delta_p(\delta_p(W,a),u)$.

3. Main Results

Section 3 consists of Subsections 3.1 and 3.2 in order to construct major components in SSE, which are encrypting and decrypting algorithms, and pattern matching algorithms. In

Subsection 3.1, we propose a novel cryptosystem based on our data hiding scheme (2,9,8) re-presented in Section 2 (Theorem 3.2 and Security analyses (3.3), (3.4)) and apply this cryptosystem to the process of encrypting and decrypting a secret data sequence (Proposition 3.4 and Security analyses (3.9), (3.10)). In Subsection 3.2, we use our automata approach recalled in Section 2 to design two algorithms for exact and approximate pattern matching on secret data encrypted by our cryptosystem proposed in Subsection 3.1 (Theorems 3.12 and 3.17).

3.1 A Novel Cryptosystem

Call Em' to be a function which is derived from the function Em by removing two Statements (2.4) and (2.5). As in [27], the state q in Statement (2.3) is computed by q = $\delta(q, M) = \delta_2(q, M)$, where $q, M \in GF^4(2^2)$ and

$$\delta_2(q,M) = \begin{cases} \varnothing & \text{if v = q,} \\ \{(i_t, a_t) \mid 1 \le i_t \le 9, t = \overline{1,k'}, k' \le 2, v_{i_t} \in S, a_t \in GF(2^2)\setminus\{0\}, M + (-q) = \sum_{t=1}^{k'} a_t v_{i_t} \ (on\, GF^4(2^2))\} & \text{otherwise,} \end{cases}$$

where $S = \{v_1, v_2, \ldots, v_9\}$ is a *2-Generators S* for $GF^4(2^2)$. Note that the number of S is given by [27]

$$c3^9 9!, \text{ where } c \approx 2^{20}. \tag{3.1}$$

Then it is easy to check that the function Em' satisfies $Em' : \mathsf{I} \times \mathsf{M} \times \mathsf{K} \to 2^{\{1,2\ldots,9\}\times GF(2^2)\setminus\{0\}}$. Ex' is a function obtained from Ex by replacing image blocks I_{i_t} with image blocks I'_{i_t} in Statement (2.4) and then inserting two Statements (2.5) and (2.4) before Statement (2.6) in Ex, then the function Ex' holds $Ex' : 2^{\{1,2\ldots,9\}\times GF(2^2)\setminus\{0\}} \times \mathsf{I} \times \mathsf{K} \to \mathsf{M}$. Since we have [27]

$$\forall (I, M, K) \in \mathsf{I} \times \mathsf{M} \times \mathsf{K}, Ex(Em(I, M, K), K) = M$$

and for our construction of two functions Em' and Ex', similary, we also follow

$$\forall (I, M, K) \in \mathsf{I} \times \mathsf{M} \times \mathsf{K}, Ex'(Em'(I, M, K), I, K) = M. \tag{3.2}$$

Remark 3.1. From defining two functions Em' and Ex' as above, all image blocks I used are not changed.

Consider Σ to be an alphabet of size 256. Set $\mathcal{P} = \Sigma$.

In [27], $(GF^4(2^2), +, \cdot)$ is considered a vector space over the field $GF(2^2)$, where $GF^4(2^2) = \{(x_1, x_2, x_3, x_4) \mid x_i \in GF(2^2), \forall i = \overline{1,4}\}$ with the vector addition and scalar multiplication given as follows.

$$x + y = (x_1 + y_1, x_2 + y_2, x_3 + y_3, x_4 + y_4),$$
$$ax = (ax_1, ax_2, ax_3, ax_4), a \in GF(2^2),$$

where $x, y \in GF^4(2^2)$ and $x = (x_1, x_2, x_3, x_4), y = (y_1, y_2, y_3, y_4)$. In addition, by the decimal representation of the vector space $GF^4(2^2)$ over the field $GF(2^2)$, then $|\mathcal{P}| = |GF^4(2^2)| = 256$, hence there exists a bijective function f from $\mathcal{P}$ to $GF^4(2^2)$, denote the inverse function of f by f^{-1}. Put $\mathcal{F}$ to be a set of all f.

From the function δ, the state q of the automaton $A(I,M,K)$ computed by Statement (2.3) is a set. The state q may be one of the following sets: $\emptyset$, $\{(i, a)\}$ for $i \in \{1,2,...,9\}$, $a \in GF(2^2)\backslash\{0\}$ or $\{(i, a),(j, b)\}$ for $i, j \in \{1,2,...,9\}$, $a, b \in GF(2^2)\backslash\{0\}$.

The index $i \in \{1,2,...,9\}$ and the coefficient $a \in GF(2^2)\backslash\{0\}) = \{1,2,3\}$ can be presented by binary strings of lengths 4 and 2, respectively. Hence, we can use 12 binary bits to present a state q. Suppose B is a binary string of length 12 to present an arbitrary state q, $B = B_{12}...B_2B_1$, then the storage structure of q in B is given as follows.

1. If $q = \emptyset$, then the value of any bit in B equals 0.
2. If $q = \{(i, a)\}$, then the values of 6 bits $B_7,B_8,...,B_{12}$ are 0; 6 remaining bits present (i, a), where 2-bit string B_2B_1 presents a, 4-bit string $B_6B_5B_4B_3$ presents i.
3. If $q = \{(i, a),(j, b)\}$, then the 6-bit string $B_{12}B_{11}..B_7$ presents (i, a) and the remaining 6-bit string $B_6B_5..B_1$ presents (j, b) in the above mentioned way.

Put Q to be a set of all possible states q, $\mathcal{C}$ is a set of all 12-bit strings B presenting q, $q \in Q$. Consider a function h, $h : Q \rightarrow \mathcal{C}$, $h(q) = B$, where q is presented by B. Obviously, h is a bijective function. Denote the inverse function of h by h^{-1}.

Let $\mathcal{K} = \{(f, K, I) \mid f \in \mathcal{F}, K \in \mathsf{K}, I \in \mathsf{I}\}$ is a finite set of secret keys. For $k \in \mathcal{K}$, $k = (f, K, I)$, we define e_k and d_k as follows.

1. $e_k : \mathcal{P} \rightarrow \mathcal{C}$, $e_k(x) = h(Em'(I, f(x),K))$ for $x \in \mathcal{P}$.
2. $d_k : \mathcal{C} \rightarrow \mathcal{P}$, $d_k(y) = f^{-1}(Ex'(h^{-1}(y),I,K))$ for $y \in \mathcal{C}$.

Set $\mathcal{E} = \{e_k \mid k \in \mathcal{K}\}$, $\mathcal{D} = \{d_k \mid k \in \mathcal{K}\}$. From Definition 2.1, the correctness of the cryptosystem $(\mathcal{P},\mathcal{C},\mathcal{K},\mathcal{E},\mathcal{D})$ is guaranteed by the following theorem.

Theorem 3.2. *Let* $\forall x \in \mathrm{P}$, $\forall k \in \mathcal{K}$, $e_k \in \mathcal{E}$ *and* $d_k \in \mathcal{D}$. *Then* $d_k(e_k(x)) = x$.

Proof. Set $M = f(x), q = Em'(I,M,K), B = h(q)$, then $e_k(x) = B = y$. We have $h^{-1}(y) = h^{-1}(B) = q$, $Ex'(q,I,K) = M$ by Formula (3.2), $f^{-1}(M) = x$, then $d_k(y) = x$. □

Security analysis of the cryptosystem $(\mathcal{P},\mathcal{C},\mathcal{K},\mathcal{E},\mathcal{D})$: Assume that we publish parameters the flip graph G, Em', Ex', $GF^4(2^2)$ and h in the cryptosystem $(\mathcal{P},\mathcal{C},\mathcal{K},\mathcal{E},\mathcal{D})$. The plaintext x is obtained from y by the Formula

$$x = d_k(y) = f^{-1}(Ex'(h^{-1}(y),I,K)).$$

So, to have accurately x, we need to know S and $k = (f, K, I)$. The number of choices for the image block I is 256^9 with gray images, 2^{9t} with palette images, where t is the number of bits to represent color indexes. Furthermore, by Formula (2.9), the security of the cryptosystem $(\mathcal{P},\mathcal{C},\mathcal{K},\mathcal{E},\mathcal{D})$ is given by the following formula

$$c3^9 9!2^{18}2^8!256^9 = c3^9 9!2^{90}2^8! \text{ for gray images,} \tag{3.3}$$

$$c3^9 9!2^{18}2^{9t}2^8! = c3^9 9!2^{18+9t}2^8! \text{ for palette images.} \tag{3.4}$$

Remark 3.3. By Remark 3.1, all pairs of functions (e_k, d_k) in the cryptosystem $(\mathcal{P},\mathcal{C},\mathcal{K},\mathcal{E},\mathcal{D})$ do not make the image blocks I change for $\forall k \in \mathcal{K}$, $k = (f, K, I)$. In addition, we can see that encrypting and hiding are done at the same time.

Consider an arbitrary subset of image blocks F as an input image, $F \subset \mathsf{I}$, $F = \{F_1,F_2,...,F_{t2}\}$, t_2 is the number of image blocks. Next, we give a way applying the cryptosystem $(\mathcal{P},\mathcal{C},\mathcal{K},\mathcal{E},\mathcal{D})$ to the process of encrypting and decrypting secret data over an insecure channel. By Remark 3.3, we can use a secret key subset H instead of one secret key k,

$$H = \{(f, K, I) \mid K \in \mathsf{K}, I \in F\} \subset \mathcal{K}$$

for $f \in \mathcal{F}$, $\mathsf{K} = \{K^1,K^2,...,K^{t1}\}$.

Suppose that secret data is a string $x = x_1x_2..x_{t3}$ for $x_i \in \mathcal{P}$, $\forall i = \overline{1, t_3}$, $t_3 \geq 1$. The encrypting algorithm e_H used to encrypt x is given as follows.

```
$i_K = 1; i_F = 1;$
For $i$ = 1 to $t_3$ Do
{
    $k_i = (f, K^{iK}, F_{iF});$                                        (3.5)
    $y_i = e_{ki}(x_i);$                                                (3.6)
    $i_K = (i_K - 1)$ mod $t_1 + 1;$
    $i_F = (i_F - 1)$ mod $t_2 + 1;$
}
$y = y_1y_2..y_{t3};$
```

The decrypting algorithm d_H used to decrypt y is given as follows.

```
$i_K = 1; i_F = 1;$
For $i$ = 1 to $t_3$ Do
{
    $k_i = (f, K^{iK}, F_{iF});$                                        (3.7)
    $x'_i = d_{ki}(y_i);$                                               (3.8)
    $i_K = (i_K - 1)$ mod $t_1 + 1;$
    $i_F = (i_F - 1)$ mod $t_2 + 1;$
}
$x' = x'_1x'_2..x'_{t3};$
```

Propostion 3.4. *Let F, x, H and the cryptosystem* ($\mathcal{P},\mathcal{C},\mathcal{K},\mathcal{E},\mathcal{D}$) *based on the data hiding* (2,9,8) *as above. Then* $d_H(e_H(x)) = x$.

Proof. Clearly, $\forall i = \overline{1, t_3}$, k_i determined in Statement (3.5) is the same as in Statement (3.7). In addition, by Theorem 3.2, x_i is encrypted by Statement (3.6) and obtained by Statement (3.8) such that $x'_i = x_i$. Then $x = x'$, hence $d_H(e_H(x)) = x$. □

Security analysis of process of encrypting and decrypting the secret data x using two algorithms e_H and d_H: Assume that we also publish parameters as in the cryptosystem ($\mathcal{P},\mathcal{C},\mathcal{K},\mathcal{E},\mathcal{D}$). Hence, to restore exactly x, we need to know S and H. The number of choices for S is $c3^9 9!$ by Formula (3.1). The number of choices for H is $2^8!2^{18.t1}256^{9.t2}$ (for gray images), $2^8!2^{18.t1}2^{9.t.t2}$ (for palette images, where t is the number of bits to represent color indexes). Then for a brute force attack, the number of all possible combinations of S and H used in two algorithms e_H and d_H is

$$c3^9 9!2^8!2^{18.t1}256^{9.t2} = c3^9 9!2^8!2^{18.t1+72.t2} \text{ for gray images,} \tag{3.9}$$

or

$$c3^9 9!2^8!2^{18t1}2^{9.t.t2} = c3^9 9!2^8!2^{18.t1+9.t.t2} \text{ for palette images.} \tag{3.10}$$

Remark 3.5. For two algorithms e_H and d_H given as above, an arbitrary image block I in the input image F can be used many times in process of encrypting and decrypting the secret data. So, for a give input image F, the secret data encrypted is not limited by the size of the input image F.

3.2 Automata Technique for Pattern Matching on Encrypted Data

Suppose that Alice has a secret data and prefers to outsource this data to a cloud provider Bob. As the server is semi-trusted, Alice needs to encrypted her plaintext and wishes to only store ciphertext in the cloud. Assume that Alice uses the cryptosystem ($\mathcal{P},\mathcal{C},\mathcal{K},\mathcal{E},\mathcal{D}$) proposed in Subsection 3.1 to encrypt data with a pair of two secret parameters (S, k) in the cryptosystem, where S is a *2-Generators* for $GF^n(p^m)$, $|S| = 9$ and $k = (f, K,I) \in \mathcal{K}$.

Because of limited storage space and computing ability, instead of downloading ciphertext, decrypting it and searching locally, Alice may ask Bob to perform pattern matching tasks on the ciphertext directly with a trapdoor of the pattern received from her.

To be able to support pattern matching on the server side without leaking information in plaintext, bellow we will construct pattern matching algorithms which can search for any pattern directly in the ciphertext.

Consider Σ to be an alphabet of size 256. Suppose that the secret data is a string over Σ

$$x = x_1x_2..x_{t3}$$

for $x_i \in \mathcal{P}$, $\forall i = \overline{1,t_3}$, $t_3 \geq 1$ and t_3 is often a large natural number, where $\mathcal{P} = \Sigma$.

Before uploading the secret data x to Bob, Alice use the encrypting function $e_k \in \mathcal{E}$ to encrypt each x_i. Then Alice computes $y_i = e_k(x_i)$, $\forall i = \overline{1,t_3}$, and the encrypted secret data is a string over Σ'

$$y = y_1y_2..y_{t3}$$

which is sent to Bob, where Σ' is an alphabet

$$\Sigma' = \{a' \mid a' = e_k(a), a \in \Sigma\}.$$

In general case, for x is any string over the alphabet Σ and a string y is obtained from x by the above way. Then we can write $y = e_k(x)$ for short and y is a string over the alphabet Σ'.

Remark 3.6. By using only one pair of two secret parameters (S, k), then the security of process of encrypting and decrypting the secret data x is similar to Formulas (3.3) (for gray images) or (3.4) (for palette images).

Suppose that Bob needs to perform exact or approximate pattern matching tasks of an arbitrary pattern p on encrypted data y. Based on our previously introduced results in [28, 29], we continue using automata technique to meet the requirements.

We first introduce some theoretical results to follow the exact pattern matching.

Propostion 3.7. *Let p be a pattern over the alphabet Σ. Then $Pos_{p'}(a') = Pos_p(a)$ for $\forall a' \in \Sigma'$, $a = d_k(a')$, where $p' = e_k(p)$.*

Proof. Set $i = Pos_p(a)$, then $a = p_i$, hence $a' = p_i'$. Without loss of generality, suppose $Pos_{p'}(a') > i$, then $\exists i' > i, p'_{i'} = a'$ by Definition 2.2, then $a = p_{i'} = d_k(p'_{i'})$. Then $i < Pos_p(a)$, a contradiction. So, we complete the proof. □

Propostion 3.8. *Let a pattern p and a text x be two strings over the same alphabet Σ and the function Sign be given by*

$$\forall a' \in \Sigma', Sign(a') = \begin{cases} 1 & If\ a \in p, \\ 0 & Otherwise. \end{cases}$$

Then $\forall a' \in \Sigma'$, $a' \in p'$ if and only if $Sign(a') = 1$.

Proof. Suppose $\forall a' \in \Sigma'$, $a' \in p'$ if and only if $\exists i$, $i = 1..|p'|$, $a' = p'_i$ if and only if $a = p_i$ if and only if $Sign(a') = 1$. □

Propostion 3.9. *Let a pattern p and a text x be two strings over the same alphabet Σ. Then p occurs at any position i in x if and only if p' occurs at the position i in y, where $y = e_k(x)$.*

Proof. Suppose that p occurs at any position i in x if and only if $p = x_ix_{i+1}..x_{i+|p|-1}$ if and only if $y_iy_{i+1}..y_{i+|p|-1} = p'$ if and only if p' occurs at the position i in y. □

Propostion 3.10. *Let p be a pattern over the alphabet Σ. Then $\forall l, 1 \leq l \leq |p|$, $Next_{p'}(l) = Next_p(l)$, where $p' = e_k(p)$.*

Proof. Without loss of generality, suppose that $lm = Next_p(l) < Next_{p'}(l)$ for $\forall l, 1 \leq l \leq |p|$. Since $p'_i = e_k(p_i), \forall i = 1..|p|$, then $p'_1 p'_2 .. p'_{lm+1}$ is both a proper suffix and prefix of $p'[1..l]$ by Definition 2.3. Hence, $p_1 p_2 .. p_{lm+1}$ is also both a proper suffix and prefix of $p[1..l]$ by Definition 2.3. Then $Next_p(l) > lm$. This is a contradiction to our supposition. So, the proof is complete. □

Propostion 3.11. *Let p be a pattern over the alphabet* Σ. *Then for* $\forall l, 0 \leq l \leq |p|$ *and* $\forall a' \in \Sigma'$, $a = d_k(a')$, $Appearance_{p'}(l,a') = Appearance_p(l,a)$, *where* $p' = e_k(p)$.

Proof. Clearly, $|p| = |p'|$ and for $\forall i, 1 \leq i \leq |p'|, \forall a', a' \in \Sigma'$, $a' = p'_i$ if and only if $a = p_i$. By Lemma 2.4 and Proposition 3.10, $Appearance_{p'}(l,a') = Appearance_p(l,a)$. □

Theorem 3.12. *Let p be a pattern over the alphabet* Σ. *Let two automata* $A_p = (\Sigma, Q_p, q_0, \delta_p, F_p)$ *and* $A_{p'} = (\Sigma', Q_{p'}, q_0, \delta_{p'}, F_{p'})$ *be determined as in Theorem 2.5. Then* $Q_{p'} = Q_p$, $F_{p'} = F_p, \forall q \in Q_{p'}, \forall a' \in \Sigma', a = d_k(a'), \delta_{p'}(q,a') = \delta_p(q,a)$, *where* $p' = e_k(p)$.

Proof. It is easy to verify that $|p| = |p'|$. In addition, by Theorem 2.5 and Proposition 3.11, then $Q_{p'} = Q_p$, $F_{p'} = F_p$ and $\delta_{p'}(q,a') = \delta_p(q,a)$. □

Remark 3.13. The meaning of Theorem 3.12 in practice is to compute $\delta_{p'}$ from δ_p.

Let a pattern p and a text (secret data) x be two strings over the same alphabet Σ and assume $|p| = |x|$. For assuming that we have only the encrypted secret data y which is not decrypted to the secret data x, from Propositions 3.7, 3.8 and 3.9, Theorem 3.12, based on the MR$_c$ algorithm for $c = 1$ and using the type a breaking point and the concept of *Pos* in [28], and by using the automaton $A_{p'}$ given as in Theorem 2.5, we have an exact pattern matching algorithm immediately that finds all occurrences of the pattern p in x as follows. Note that the trapdoor according to the search pattern p is computed based on p, which includes the length of p, the functions *Sign*, $Pos_{p'}$ and the automaton $A_{p'}$.

```
jump = |p|;
While ($jump \leq |y|$)
{
        If ($sign(y_{jump})$ == 1)
        {
          $q = 0$;
          $i = jump - Pos_{p'}(y_{jump}) + 1$; Do
          {
                $q = \delta_{p'}(q, y_i)$;
                If ($q == |p|$) Mark an occurrence of $p$ at $i - |p| + 1$ in $x$;
                $i++$;
          } While ($q \neq 0$ and $i \leq |y|$);
          $jump = i - 1$;
        }
        $jump = jump + |p|$;
}
```

Remark 3.14. Obviously, the time complexity of the above algorithm is the same as our MR$_1$ algorithm in the worst case, $O(n)$ [28]. Then in the worst case, our new algorithm's time complexity is also $O(n)$.

Next, theoretical results for approximate pattern matching are shown as follows.

Propostion 3.15. *Let p be a pattern over the alphabet* Σ. *Then* $WConfig(p') = WConfig(p)$, *where* $p' = e_k(p)$.

Proof. Obviously, $W_0 \in WConfig(p')$ and $WConfig(p)$. Consider $\forall W' \in WConfig(p')\backslash\{W_0\}$, then we can set $W'=\{w_1', w_2', ..., w_l'\}$ for $1 \le l \le |p'|$. Then $\exists C' = \{u'_1, u'_2, ..., u'_l\} \in Config(p')$ by Definition 2.6, where $W_{p'}(u'_i) = w_i'$ for $1 \le i \le l$. Then $\exists! C = \{u_1, u_2, ..., u_l\} \in Config(p)$, $u_i = d_k(u'_i)$ for $1 \le i \le l$. Set $W = W_p(C)$, then $W \in WConfig(p)$ by Definition 2.6. It easy to verify that $Rm_{p'}(u'_i) = Rm_p(u_i)$ for $1 \le i \le l$, then $W_{p'}(u_i') = W_p(u_i)$ for $1 \le i \le l$ by Definition 2.6. Hence, $W' = W$, then $WConfig(p') \subset WConfig(p)$. Similarly, we have $WConfig(p) \subset WConfig(p')$. So, the proof is complete. □

Propostion 3.16. *Let p be a pattern over the alphabet Σ. Then* $Ref_{p'}(i,a') = Ref_p(i,a)$ *for* $\forall i$, $0 \le i \le |p'|$ *and* $\forall a' \in \Sigma', a = d_k(a')$, *where* $p' = e_k(p)$.

Proof. Clearly, $W_{p'}^i(a') = W_p^i(a)$ by Definition 2.7. So, $Ref_{p'}(i,a') = Ref_p(i,a)$ by Definition 2.7. Hence, we complete the proof. □

Theorem 3.17. *Given a pattern p on Σ and a positive integer constant c with* $1 \le c \le |p|$. *Let two automata* $A_p^{Pc} = (\Sigma, Q_p, q_0, \delta_p, F_p)$ *and* $A_{p'}^{Pc} = (\Sigma', Q_{p'}, q_0, \delta_{p'}, F_{p'})$ *be determined as in Theorem 39* [29]. *Then* $Q_{p'} = Q_p$, $F_{p'} = F_p$, $\forall q \in Q_{p'}$, $\forall a' \in \Sigma'$, $a = d_k(a')$, $\delta_{p'}(q,a') = \delta_p(q,a)$, *where* $p' = e_k(p)$.

Proof. By Proposition 3.15, $Q_{p'} = Q_p$. Evidently, $\forall a' \in \Sigma'$, $a = d_k(a')$, $a' \in p'$ if and only if $a \in p$. Furthermore, by Definition 2.8 and Proposition 3.16, $\delta_{p'}(W,a') = \delta_p(W,a)$. Then $F_{p'} = F_p$. So, the proof is complete. □

Remark 3.18. The meaning of Theorem 3.17 in practice is to compute $\delta_{p'}$ from δ_p.

Based on the approximate pattern matching problems considered in [17, 18, 20], we introduce a new concept of the appearance of the pattern p in x with a given error. This is a basis for giving requirements for the approximate pattern matching algorithm.

Definition 3.19. Given two strings p and x over Σ, and a string similarity measure d. Let an error $\varepsilon, \varepsilon > 0, \varepsilon \in R$. Then p appears in x with the error if there exists a substring u of x such that $d(p,u) \le \varepsilon$.

To construct the approximate pattern matching algorithm, we need a function to measure the string similarity. The most commonly used similarities are recalled in [19, 20, 21]. Bakkelund [1] proposed a well known string similarity measure which is based on the longest commonly subsequence. Similarly, here we define a new measure of similarity between two strings

$$d(p,u) = 1 - \frac{lcs(p,u)}{min\{|p|,|u|\}}, \tag{3.11}$$

where p is a pattern and u is a substring of x. Clearly, d given above is positive definite and symmetric.

Propostion 3.20. *Given two strings p and x on Σ. Then* $\forall u'$, u' *is an arbitrary substring of* y, $d(p', u') = d(p, u)$, *where* $p' = e_k(p), y = e_k(x), u = d_k(u')$.

Proof. Clearly, $|p'| = |p|$, $|u'| = |u|$ and $lcs(p,u) = lcs(p',u')$. By Formula (3.11), $d(p', u') = d(p, u)$. So, we complete the proof. □

By using the string similarity measure given in Formula (3.11), the automata technique for computing $lcs(p', u')$ [29] will make an approximate pattern matching algorithm fast, and especially efficient for one pattern and a set of a large number of encrypted texts.

Given a pattern p and a text (secret data) x over the same alphabet Σ, and an arbitrary substring u of x. Let $\varepsilon, 0 < \varepsilon < 1$ and $d(p, u)$ be given as in Formula (3.11) such that $d(p,u) \le \varepsilon$. Then by Proposition 3.20, $d(p,u) \le \varepsilon$. By Formula (3.11), we have

$$lcs(p', u') \geq (1-\varepsilon) min\{|p'|, |u'|\}. \quad (3.12)$$

If there is u' which is a substring of y such that $lcs(p', u') \geq (1-\varepsilon)|p|$, then Formula (3.12) holds that means $d(p', u') \leq \varepsilon$. Hence, $\exists u$, u is a substring of x, $d(p,u) \leq \varepsilon$. So, the constant c in Theorem 39 [29] is determined by $c = \lceil (1-\varepsilon)|p| \rceil$.

Without decrypting y, based on Theorem 3.17, Definition 3.19 and Formula (3.11), use the automaton $A_{p'}^{Pc}$ given as in Theorem 39 [29], we immediately have an approximate pattern matching algorithm which determines whether p appears in x with the error ε or not as follows. Here, the trapdoor responding to the pattern p is determined from p and ε, which consists of the constant c and the automaton $A_{p'}^{Pc}$.

```
app = 0;
q = $W_0$; //The initial state of the automaton $A_{p'}^{Pc}$ is started from $W_0$
For $i$ = 1 to |$y$| Do
{

    q = $\delta_{p'}(q, y_i)$;
    If (|q| = c) {app=1; Break;}
}
If (app = 1) Announce the appearance of the pattern $p$ in $x$ with the
error;
Else Announce that $p$ does not appear in $x$ with the error $\varepsilon$.
```

Remark 3.21. Since we can compute $\delta_{p'}$ from δ_p, our proposed algorithm is similar to the Algorithm 2 (the parallel algorithm) in [29]. In addition, according to Theorem 39 [29], δ_p is computed in parallel way and the Algorithm 2 costs the worst case time complexity $O(n)$ with the supposition that the Algorithm 2 uses k processors for k is an upper estimate of the $lcs(p,x)$. As an immediately consequence, in the worst case, we have the $O(n)$ time complexity of the above algorithm when it uses $\lceil (1-\varepsilon)|p| \rceil$ processors.

4. Conclusions

From our results in the steganography and pattern matching areas and some suggestions in the next works in [27, 28, 29], this paper has completed some parts of those works. Based on the data hiding scheme (2, 9, 8) in [27], we construct a novel cryptosystem with high security. This method allows both of encrypting and hiding to be done at once, the ciphertext not to depend on the input image size as existing hybrid techniques of cryptography and steganography. Next, we use this cryptosystem to encrypt secret data on users side. With the ciphertext, we design two pattern matching algorithms to search for any pattern in it directly on cloud servers side. The idea of the design is to apply our automata approach for the exact pattern matching and the longest common subsequence problems in [28, 29]. For the assumption that the approximate algorithm uses $\lceil (1-\varepsilon)m) \rceil$ processors, the time complexities of these algorithms are both $O(n)$ in the worst case, where ε, m and n are the error of our measure of similarity between two strings and lengths of the pattern and secret data, respectively.

With our automata approach to pattern matching algorithms, the automata constructed are only based on search patterns. Then the algorithms will have lots of advantages in case of a given pattern and a very large set of ciphertexts stored in the cloud. So, in the future, we continue studying this technique to apply in SE.

Acknowledgements

The author is truly grateful to Phan Trung Huy, Phan Thi Ha Duong and Vu Thanh Nam for their valuable suggestions and help. This work was partially funded by the Vietnam National Foundation for Science and Technology Development (NAFOSTED) under the grant number 101.99-2016.16.

References

[1] D. Bakkelund, "An LCS-based String Metric," *University of Oslo (Norway)*, September 23, 2009. Article (CrossRef Link).
[2] P. Bharti, R. Soni, "A New Approach of Data Hiding in Images Using Cryptography and Steganography," *International Journal of Computer Applications*, 58(18), pp. 1-5, 2012. Article (CrossRef Link).
[3] A. Blumer, J. Blumer, D. Haussler, A. Ehrenfeucht, M. T. Chen, J. Seiferas, "The Smallest Automation Recognizing The Subwords of A Text," *Theoretical Computer Science*, Volume 40, pp. 31-55, 1985. Article (CrossRef Link).
[4] S. Chakraborty, S. K. Bandyopadhyay, "Steganography Method Based on Data Embedding by Sudoku Solution Matrix," *International Journal of Engineering Science Invention*, 2(7), pp. 36-42, 2013. Article (CrossRef Link).
[5] A. Chatterjee, A.K. Das, "Secret Communication Combining Cryptography and Steganography," *Progress in Advanced Computing and Intelligent Engineering*, Vol. 563, pp. 281-291, 2018. Article (CrossRef Link).
[6] G. Chugh, "Information Hiding - Steganography & Watermarking: A Comparative Study," *International Journal of Advanced Research in Computer Science*, 4(4), pp. 165-171, 2013.
[7] N. Desmoulins, P. A. Fouque, C. Onete, O. Sanders, "Pattern Matching on Encrypted Streams," *Advances in Cryptology – ASIACRYPT 2018*, pp. 121-148, 2018. Article (CrossRef Link).
[8] Q. Dong, Z. Guan, L. Wu, Z. Chen, "Fuzzy Keyword Search over Encrypted Data in The Public Key Setting," *Web-Age Information Management*, pp. 729-740, 2013. Article (CrossRef Link).
[9] R. Dowsley, A. Michalas, M. Nagel, N. Paladi, "A Survey on Design and Implementation of Protected Searchable Data in The Cloud," *Computer Science Review*, Volume 26, pp. 17-30, 2017. Article (CrossRef Link).
[10] A. Ehrenfeucht, R, M. McConnell, N. Osheim, S. W. Woo, "Position Heaps: A Simple and Dynamic Text Indexing Data Structure," *Journal of Discrete Algorithms*, Vol. 9, pp. 100-121, 2011. Article (CrossRef Link).
[11] Y. K. Gedam, J.N. Varshapriya, "Fuzzy Keyword Search over Encrypted Data in Cloud Computing," *Journal of Engineering Research and Applications*, 4(7), pp. 197-202, 2014. Article (CrossRef Link).
[12] F. Han, J. Qin, J. Hu, "Secure Searches in The Cloud: A Survey," *Future Generation Computer Systems*, Vol. 62, pp. 66-75, 2016. Article (CrossRef Link).
[13] R. Haynberg, J. Rill, D. Achenbach, J. Müller-Quade, "Symmetric Searchable Encryption for Exact Pattern Matching Using Directed Acyclic Word Graphs," in *Proc. of 2013 International Conference on Security and Cryptography (SECRYPT)*, pp. 403-410, 2013. Article (CrossRef Link).
[14] M. Jain, S. K. Lenka, "A Review of Digital Image Steganography Using LSB and LSB Array," *International Journal of Applied Engineering Research*, 11(3), pp. 1820-1824, 2016.

Article (CrossRef Link).

[15] N. S. Jho, D. Hong, “Symmetric Searchable Encryption with Efficient Conjunctive Keyword Search,” *KSII Transactions on Internet and Information Systems*, 7(5), pp. 1328-1342, 2013. Article (CrossRef Link).

[16] M. S. John, P. SumaLatha, M. Joshuva, “A Comparative Study of Index-Based Searchable Encryption Techniques,” *International Journal of Advanced Research in Computer Science*, 6(3), pp. 13-15, 2015.

[17] G. M. Landau, U. Vishkin, “Efficient String Matching with k Mismatches,” *Theoretical Computer Science*, Vol. 43, pp. 239-249, 1986. Article (CrossRef Link).

[18] J. V. Leeuwen, “Handbook of Theoretical Computer Science,” *Elsevier MIT Press*, Vol. A, pp. 290-300, 1990.

[19] Z. Mei, B. Wu, S. Tian, Y. Ruan, Z. Cui, “Fuzzy Keyword Search Method over Ciphertexts Supporting Access Control,” *KSII Transactions on Internet and Information Systems*, 11(11), pp. 5671-5693, 2017. Article (CrossRef Link).

[20] G. Navarro, “A Guided Tour to Approximate String Matching,” *ACM Computing Surveys*, 33 (1), pp. 3188, 2001. Article (CrossRef Link).

[21] P. H. Paris, N. Abadie, C. Brando, “Linking Spatial Named Entities to The Web of Data for Geographical Analysis of Historical Texts,” *Journal of Map & Geography Libraries*, 13(1), pp. 82-110, 2017. Article (CrossRef Link).

[22] D. X. Song, D. Wagner, A. Perrig, “Practical Techniques for Searches on Encrypted Data,” in *Proc. of 2000 IEEE Symposium on Security and Privacy*, pp. 44, 2000. Article (CrossRef Link).

[23] S. Song, J. Zhang, X. Liao, J. Du, Q. Wen, “A Novel Secure Communication Protocol Combining Steganography and Cryptography,” *Procedia Engineering*, Vol. 15, pp. 2767-2772, 2011. Article (CrossRef Link).

[24] D. R. Stinson, “Cryptography: Theory and Practice (CRC Press Series on Discrete Mathematics and Its Application),” *CRC Press*, pp. 1-20, 180-184, 1995.

[25] M. Strizhov, Z. Osman, I. Ray, “Substring Position Search over Encrypted Cloud Data Supporting Efficient Multi-User Setup,” *Future Internet*, 8(3), 28, 2016. Article (CrossRef Link).

[26] D. M. Sunday, “A Very Fast Substring Search Algorithm,” *Communications of The ACM*, 33(8), pp. 132-142, 1990. Article (CrossRef Link).

[27] N. H. Truong, “A New Digital Image Steganography Approach Based on The Galois Field *GF*(p^m) Using Graph and Automata,” *KSII Transactions on Internet and Information Systems*, 13(9), pp. 4788-4813, 2019. Article (CrossRef Link).

[28] N. H. Truong, “A New Approach to Exact Pattern Matching,” *Journal of Computer Science and Cybernetics*, 35(3), pp. 197-216, 2019. Article (CrossRef Link).

[29] N. H. Truong, “Automata Technique for The LCS Problem,” *Journal of Computer Science and Cybernetics*, 35(1), pp. 21-37, 2019. Article (CrossRef Link).

[30] Varsha, R. S. Chhillar, “Data Hiding Using Steganography and Cryptography,” *International Journal of Computer Science and Mobile Computing*, 4(4), pp. 802-805, 2015. Article (CrossRef Link).

[31] R.M. Yadav, D. S. Tomar, R. K. Baghel, “A Study on Image Steganography Approaches in Digital Images,” *Engineering Universe for Scientific Research and Management*, 6(5), pp. 1-6, 2014. Article (CrossRef Link).

[32] W. Yunling, W. Jianfeng, C. Xiaofeng, “Secure Searchable Encryption”: A Survey, *Journal of Communications and Information Networks*, 1(4), pp. 52-65, 2016. Article (CrossRef Link).

[33] L. Wei, H. Zhu, Z. Cao, X. Dong, W. Jia, Y. Chen, A. Vasilakos, “Security and Privacy for Storage and Computation in Cloud Computing,” *Information Sciences*, Vol. 258, pp. 371-386, 2014. Article (CrossRef Link).

[34] B.B. Zaidan, A. A. Zaidan, A. K. Al-Frajat, H. A. Jalab, “On The Differences between Hiding Information and Cryptography Techniques: An Overview,” *Journal of Applied Science*, 10(15), pp. 1650-1655, 2010. Article (CrossRef Link).

Nguyen Huy Truong was born in Hanoi, Vietnam. He received his Bachelor of Science in Mathematics and Informatics from VNU University of Science in 1999, Bachelor of Economics in Banking from National Economics University in 2000 and Master of Science in Mathematical Assurances for Computers and Computing Systems from VNU University of Science in 2003. He is currently a final year PhD student of Mathematics and Informatics and a lecturer in School of Applied Mathematics and Informatics, Hanoi University of Science and Technology, Vietnam. His research interests include theory of codes and applications, and information security.