International Journal of Innovative
Computing, Information and Control
Volume **8**, Number **10(A)**, October **2012**

ICIC International ©2012 ISSN 1349-4198
pp. 6585–6598

DATA HIDING IN DNA SEQUENCES BASED ON TABLE LOOKUP SUBSTITUTION

Jin-Shiuh Taur[1], Heng-Yi Lin[1], Hsin-Lun Lee[1] and Chin-Wang Tao[2]

[1]Department of Electrical Engineering
National Chung Hsing University
No. 250, Kuo Kuang Rd., Taichung 402, Taiwan

[2]Department of Electrical Engineering
National I-Lan University
No. 1, Sec. 1, Shen-Lung Rd., I-Lan 260, Taiwan
cwtao@niu.edu.tw

Received June 2011; revised October 2011

Abstract. *Hiding secret messages in DNA sequences has become a popular research in recent years. In this paper, we propose an improved algorithm named the Table Lookup Substitution Method (TLSM) to enhance the performance of an existing data hiding method called the substitution method. Moreover, a general form of the TLSM is discussed, which includes the original method as a special case. After that, a Base-t TLSM and an extended TLSM (ETLSM) are proposed to further enhance the TLSM. The Base-t TLSM encodes the secret message with radix t to fully utilize the substitution table. The ETLSM takes additional letters into account to increase the number of selectable substitution tables, which can improve the security of the TLSM remarkably. Compared with the original substitution method, the proposed Base-t ETLSM can significantly improve the cracking probability and the amount of information hidden in the DNA sequence.*

Keywords: DNA, Data hiding, Table lookup substitution method

1. **Introduction.** In recent years, emerging Internet applications make the transmission security become a very important issue. The exposedness during transmission gives rise to the requirement of encryption when transmitting private or confidential data. Data hiding is one of the popular schemes for this purpose. The basic idea of data hiding is to embed confidential data into existing media. Lots of data hiding schemes are proposed for intellectual property applications. In recent years, the research concerning transmitting secret data by data hiding to avoid malicious intruding and fulfill a safe transmission has become popular. The most common data hiding medium is the image [1] and the issue is focused on hiding messages without affecting or corrupting the original image too much. However, hiding data in the image is usually restricted by the original image structure [2]. Recently, data hiding based on the DNA sequence has been attracting much attention, and several methods have been proposed [2-6]. Briefly speaking, the DNA sequence is a sequence composed of four distinct letters, A, C, G and T; each letter corresponds to a specific nucleotide. The biological property of a DNA sequence has been manipulated in most DNA-based methods. However, using only biological properties may limit the performance of data hiding methods. Besides the biological view, Chang et al. [6] and Shiu et al. [2] have developed methods based on the fact that there are almost no differences between a real DNA sequence and a faked DNA sequence. As an emerging medium, DNA sequences have several advantages compared with images [7]. First, DNA sequences are composed of letters which are meaningless for most people. This means we

do not need to worry about the distortion. Second, the performance of the DNA sequence on embedding capacity and computational efficiency is potentially much better than that of the image. For a gray-level image, each pixel requires 8 bits memory. In contrast, for a DNA sequence, each nucleotide needs only 2 bits. Therefore, considering the memory requirement for hiding a 1-bit message, one *bpn* (bits per nucleotide) is equivalent to about four *bpp* (bits per pixel). In our approach, the equivalent *bpp* is larger than 6.34 (cf. Table 9). In contrast, most image-based methods have *bpp* values less than three in order to have a reasonable PSNR [8] and the values depend on the characteristics of the images. Consequently, the DNA medium is more cost-effective than the image in memory requirement.

In [2], Shiu et al. proposed three interesting data hiding methods: the insertion method, the complementary pair method, and the substitution method. Among these three methods, the substitution method is more compact and effective than the other two methods and many other existing methods [2]. It provides good performance and security properties using a simple framework.

In this paper, we propose a table lookup substitution method (TLSM), which is an improved method based on the original substitution method in [2]. In the TLSM, the basic framework of the original method is adopted and a rule table is proposed to replace the complementary rule. Similar to the original method, the TLSM substitutes a specific letter for its corresponding letter in the reference DNA sequence according to the rule table and the secret bits to be hidden. The rule table is able to hide two secret bits for each letter conversion, while the original complementary rule can store one bit only. According to this property, the TLSM is intrinsically capable of hiding more information in each letter. In addition, the TLSM can be extended to a general approach which hides data in any sequences of letters or symbols. Furthermore, the original substitution method can be considered as a special case of the generalized TLSM. We also propose two approaches to further enhance the performance of the generalized TLSM; they are the Base-t TLSM and the Extended TLSM (ETLSM). With the secret message expressed in base-t representation, the TLSM can utilize the substitution table more efficiently since it can fully utilize all conversion entries with a proper radix. The ETLSM can largely increase the safety level of the basic TLSM by taking additional letters into account.

The rest of this paper is organized as follows. In Section 2, we briefly describe the algorithm of the original substitution method in [2]. In Section 3, the proposed TLSM is described. In Section 4, a comparison between the TLSM and the original method is given. In Section 5, the generalized TLSM, the Base-t TSLM, and the ETLSM are described. At last, Section 6 gives the conclusion.

2. Original Substitution Method.

In this section, we first illustrate how the original method hides messages into DNA sequences [2]. According to the secrete message, the hiding procedure substitutes another letter for an existing letter on a specific location decided by the system. The embedding algorithm of the substitution method is described in Algorithm 1, where the conversion function $\theta(s)$ converts a given letter s to a specific letter defined by the complementary rule. For instance, if we have a complementary rule defined as (AC)(CG)(GT)(TA), then the result of θ(G) will be T, and the result of θ(T) will be A. Additionally, the substitution method will convert the letter s into s (unchanged), $\theta(s)$ and $\theta(\theta(s))$ when the secrete message is 0, 1 and no data, respectively.

As an example, given a reference DNA $\mathbf{S}$ = ACGGAATTGCTTCAG, a secret message $\mathbf{M} = 0111010$, a hiding location set $\mathbf{A} = 2, 3, 5, 10, 12, 13, 15$ and the complementary rule (AT)(TG)(GC)(CA), the faked DNA sequence $\mathbf{S}'$ will be GCCATGCCAACTAGG. The extracting algorithm, a reversion of the embedding one, is described in Algorithm 2.

ALGORITHM 1. The embedding algorithm of the substitution method

1) Input: a reference DNA sequence $\mathbf{S} = \{s_1, \ldots, s_n\}$, a secret binary message $\mathbf{M} = \{m_1, \ldots, m_p\}$, $(p \leq n)$ and a complementary rule $\theta(s)$.
2) Initialize the faked DNA sequence $\mathbf{S}'$ as an empty sequence.
3) Randomly generated a sorted ascending distinct integer set $\mathbf{A} = \{a_1, \ldots, a_p\}$ and $\forall a_j \in \mathbf{A}$, $a_j \leq n$. /* m_j will be hidden at position a_j */
4) **for** integer i from 1 to n,
   if $i = a_j$ and $m_j = 1$, **then** $\mathbf{S}' = \mathbf{S}' + \{\theta(s_i)\}$.
   elseif $i = a_j$ and $m_j = 0$, **then** $\mathbf{S}' = \mathbf{S}' + \{s_i\}$.
   elseif $i \neq a_j$, **then** $\mathbf{S}' = \mathbf{S}' + \{\theta(\theta(s_i))\}$.

ALGORITHM 2. The extracting algorithm of the substitution method

1) Input: a faked DNA sequence $\mathbf{S}^{'} = \{s_1^{'}, \ldots, s_n^{'}\}$, a reference DNA sequence $\mathbf{S} = \{s_1, \ldots, s_n\}$ and the corresponding complementary rule $\theta(s)$.
2) Initialize the secret message $\mathbf{M}$ as an empty sequence.
3) **for** integer i from 1 to n,
   if $s_i^{'} = s_i$, **then** $\mathbf{M} = \mathbf{M} + \{0\}$.
   elseif $s_i^{'} = \theta(s_i)$, **then** $\mathbf{M} = \mathbf{M} + \{1\}$.

Note that in the original hiding method, the theoretical maximum amount of messages that can be hidden in a single nucleotide is one bit. Since the complementary rule has to satisfy the constraint that all s, $\theta(s)$, $\theta(\theta(s))$ and $\theta(\theta(\theta(s)))$ should be different to each other, there are totally six legal complementary rules as follows [2]: (AT)(TC)(CG)(GA), (AT)(TG)(GC)(CA), (AC)(CT)(TG)(GA), (AC)(CG)(GT)(TA), (AG)(GT)(TC)(CA) and (AG)(GC)(CT)(TA).

3. Table Lookup Substitution Method (TLSM).

To improve the effectiveness of the substitution method, we propose a Table Lookup Substitution Method (TLSM), which can double the capacity of message hiding. Based on the same framework of the original method, in the TLSM, we replace the complementary rule with a rule table. The key idea of the TLSM is to extend the 1-bit complementary rule into a 2-bit rule table so that each conversion of letters can represent two bits of the secret message.

3.1. Algorithms of the TLSM.

The DNA sequence is a combination of ACGT letters. In the TLSM, we construct a rule table to replace a letter in the reference DNA sequence with one of these four letters according to the (2-bit) input message. Table 1 shows an example of the rule table. Note that the entries in each *sbs* column have to be assigned with different letters so that the input message can be extracted correctly. Since the entries in the table can be arbitrarily assigned, there are totally $24^4 (= (4!)^4)$ possible situations of this table. Let m_i and m_{i+1} denote two consecutive secret message bits. The conversion function $s^{'} = \phi(s, m_i, m_{i+1})$ and the reversion function $\{m_i, m_{i+1}\} = \phi^{'}(s, s^{'})$ are used to hide and extract secret message bits based on the rule table, respectively, where s and $s^{'}$ are ACGT letters. In Table 1, the first row of the left-most table indicates that a letter A in the reference DNA sequence will be replaced with the letter C when the message bits are 00. That is, $\mathrm{C} = \phi(\mathrm{A}, 0, 0)$ and $\{0, 0\} = \phi'(\mathrm{A}, \mathrm{C})$.

TABLE 1. An example of the rule table of the TLSM, where the first row is the letter in the reference sequence, the msg column shows the 2-bit message and the sbs column shows the substituted symbol with respect to the message

A			C			G			T		
msg		*sbs*	*msg*		*sbs*	*msg*		*sbs*	*msg*		*sbs*
0	0	C	0	0	A	0	0	T	0	0	G
0	1	A	0	1	C	0	1	G	0	1	T
1	0	T	1	0	G	1	0	A	1	0	A
1	1	G	1	1	T	1	1	C	1	1	C

ALGORITHM 3. The embedding algorithm of the TLSM

1) Given a reference DNA sequence $\mathbf{S}$, a binary secret message $\mathbf{M}$, a location set $\mathbf{A}$, and a conversion function ϕ.
   - $\mathbf{S} = \{s_1, \ldots, s_n\}$.
   - $\mathbf{M} = \{m_1, \ldots, m_{2p}\}$, where $p \leq n$.
   - $\mathbf{A} = \{a_1, \ldots, a_p\}$, and $\forall a_j \in \mathbf{A}$, $a_j \leq n$.
   - $\phi(s, m_t, m_{t+1})$ returns a DNA letter.
2) Initialize the faked DNA sequence $\mathbf{S}'$ as an empty sequence.
3) **for** integer i from 1 to n,
   if $i = a_j$ **then** $s'_i = \phi(s_i, m_{2j-1}, m_{2j})$.
   else randomly assign a letter to s'_i.
   $\mathbf{S}' = \mathbf{S}' + \{s'_i\}$.

ALGORITHM 4. The extracting algorithm of the TLSM

1) Given a faked DNA sequence $\mathbf{S}'$, a reference DNA sequence $\mathbf{S}$, a location set $\mathbf{A}$, and a reversion function ϕ':
   - $\mathbf{S}' = \{s'_1, \ldots, s'_n\}$, $\mathbf{S} = \{s_1, \ldots, s_n\}$.
   - $\mathbf{A} = \{a_1, \ldots, a_p\}$, and $a_j \leq n$.
   - $\phi'(s, s')$ returns a 2-bit data.
2) Initialize $\mathbf{M}$ as an empty sequence.
3) **for** integer i from 1 to n,
   if $i = a_j$ **then** $\mathbf{M} = \mathbf{M} + \{\phi'(s_i, s'_i)\}$.

To implement the TLSM, similar to the original method, it is essential to have three prerequisites, including the reference DNA sequence, the rule table, and the location set indicating where the bits of the secret message are hidden. The embedding algorithm of the TLSM is listed in Algorithm 3. Note that the location set is prior information required for both the sender and the receiver because the situation of hiding nothing is not coded. For example, we can select one of the r predefined position set. It may need some extra handling. However, it can increase the transfer efficiency and improve the security since the location information is not exposed in the transmission. This requirement can be relieved in Section 5. As for the message extracting, the process needs four prerequisites including, the reference DNA sequence, the faked DNA sequence, the location set for the message, and the rule table. The extracting algorithm is described in Algorithm 4.

3.2. **An example for the TLSM.** Example 1 illustrates the embedding and extracting algorithms of the TLSM. Compared to the result from the original substitution method, we observe that the length of the secret message carried by the TLSM doubles that by the original method under the same condition.

EXAMPLE 1. An example for the TLSM, where $n = 15$, $p = 14$

Embedding
Given a DNA sequence **S**, a secret message **M** and a location set **A**: • **S** = ACGGAATTGCTTCAG. • **M** = 01110100101110. • **A** = 2, 3, 5, 10, 12, 13, 15. Mapped with Table 1, the faked DNA sequence **S′** is TCCCACATCAAATAA
Extracting
Given. • **S′** = TCCCACATCAAATAA. • **S** = ACGGAATTGCTTCAG. • **A** = 2, 3, 5, 10, 12, 13, 15. Mapped with Table 1, the extracted secret massage **M** is 01110100101110

4. **Performance Comparison.** In this section, some commonly used data hiding performance indices [2], including the capacity, payload, bits-per-nucleotide (*bpn*), and cracking probability, will be discussed. Also, we will compare the TLSM with other DNA-based data-hiding methods using the *bpn*.

4.1. **The capacity, payload and bits-per-nucleotide.** According to [2], capacity, payload and *bpn* are defined as follows. The capacity is defined as the length of the faked sequence. The payload is defined as the length of the extra nucleotides created after the embedding process. The *bpn* is the average number of secret bits hidden in each character of the embedded sequence. Suppose that we have a reference sequence $\mathbf{S} = \{s_1, \ldots, s_{n_R}\}$ and the faked sequence $\mathbf{S}' = \{s_1', \ldots, s_{n_F}'\}$ after embedding a binary secret message $M = \{m_1, \ldots, m_{n_M}\}$ into **S**. The capacity (*cp*) is

$$cp = |\mathbf{S}'| = n_F, \tag{1}$$

the payload (*pl*) is

$$pl = |\mathbf{S}'| - |\mathbf{S}| = n_F - n_R, \tag{2}$$

and the *bpn* is

$$bpn = \frac{|\mathbf{M}|}{cp} = \frac{n_M}{n_F}. \tag{3}$$

To compare the TLSM with the original substitution method, we adopt the experimental results in [2], which are listed in Table 2, and use the same tested DNA sequence with the same number of hiding positions. These eight test DNA sequence are adopted from NCBI database [9]. Under the same condition, the number of bits that can be hidden in each letter for the proposed method is two while the number is only one for the original method. Therefore, the TLSM can hide 40,000 bytes while the original method can hide 20,000 bytes in the same situation.

The optimal *bpn* of the original method is one while the optimal *bpn* of the TLSM is two. The optimum of the *bpn* for both methods occurs when all letters in the sequence are embedded with data. Suppose that the reference DNA sequence is long enough to hide the message, and the unused letters in the reference sequence are removed before

TABLE 2. The results of hiding a 20,000/40,000 bytes message

		Original (20,000 bytes)			TLSM (40,000 bytes)		
test DNA	length	capacity	payload	*bpn*	capacity	payload	*bpn*
AC153526	200,117	200,117	0	0.80	200,117	0	1.60
AC166252	149,884	149,884	0	1.00	149,884	0	2.00
AC167221	204,841	204,841	0	0.78	204,841	0	1.56
AC168874	206,488	206,488	0	0.77	206,488	0	1.55
AC168897	200,203	200,203	0	0.80	200,203	0	1.60
AC168901	191,456	191,456	0	0.84	191,456	0	1.67
AC168907	194,226	194,226	0	0.82	194,226	0	1.65
AC168908	218,028	218,028	0	0.73	218,028	0	1.47
			Avg. *bpn*	0.82		Avg. *bpn*	1.64

transmitting to the receiver end. Then the average *bpn*'s for both methods will reach the maxima.

4.2. **Cracking probability.** In computing the cracking probability, two different situations are considered. In the first situation, the reference DNA sequence must be selected from a known database, and the size of the sample space is equal to the number of all available DNA sequences in public, which is about 163 million (cf. EMBL nucleotide sequences database release 101 [11]) In the original substitution method, the number of all legal complementary rules is 6. Therefore, the cracking probability of the original method is

$$P_1^{Ori} = \frac{1}{1.63 \times 10^6} \times \frac{1}{6}. \tag{4}$$

As for the proposed TLSM, the number of all possible tables is $(4!)^4$. Also, assuming that the system has totally r position sets available, the cracking probability is

$$P_1^{TLSM} = \frac{1}{1.63 \times 10^6} \times \frac{1}{24^4} \times \frac{1}{r}. \tag{5}$$

As we can see, in terms of the cracking probability, the TLSM outperforms the original method due to the larger number of possible rules. Moreover, it is obvious that the cracking probability will be further improved as r grows. Note that the information about which position set is used has to be available at the receiver end.

In the second situation, the reference DNA sequence is not necessarily to be real, i.e., the entire reference DNA sequence can be artificial. Since the artificial DNA sequence is not available in public, the secret message is guessed from the faked DNA sequence directly. For the original method, each letter may be embedded with three kinds of messages, including 0, 1 and nothing. Therefore, the cracking probability is

$$P_2^{Ori} = \frac{1}{3^n}, \tag{6}$$

where n is the length of the faked DNA sequence. As for the proposed method, there are five possible message hiding situations in each letter: 01, 10, 11 and nothing. Therefore, the cracking probability of the TLSM is

$$P_2^{TLSM} = \frac{1}{5^n}. \tag{7}$$

The comparison on cracking probability for both situations is summarized in Table 3. It can be observed that the TLSM is much better than the original one in either situation.

TABLE 3. The cracking probabilities of the original method and the TLSM

	original	TLSM with r location sets
Real DNA sequence	$\frac{1}{1.63\times10^6} \times \frac{1}{6}$	$\frac{1}{1.63\times10^6} \times \frac{1}{24^4} \times \frac{1}{r}$
Artificial DNA sequence of length n	$\frac{1}{3^n}$	$\frac{1}{5^n}$

ALGORITHM 5. The embedding algorithm of the generalized TLSM

1) Given a reference sequence $\mathbf{S}$, a binary secret message $\mathbf{M}$, and a conversion function ϕ:
   - $\mathbf{S} = \{s_1, \ldots, s_n\}$, where s_i is one of the $2^k + 1$ different letters.
   - $\mathbf{M} = \{m_1, \ldots, m_{k\cdot p}\}$, where $p \leq n$.
   - $\phi(s, [m_i, \ldots, m_{i+k-1}] \text{ or } \infty)$ returns a letter, where ∞ denotes the situation that nothing is hidden.
2) Initialize the faked sequence $\mathbf{S}'$ as an empty sequence.
3) Randomly generate a sorted ascending distinct integer set $\mathbf{A} = \{a_1, \ldots, a_p\}$, where $a_j \leq n$.
4) **for** integer i from 1 to n,
   if $i = a_j$, **then** $s_i' = \phi(s_i, [m_{k\cdot(j-1)+1}, \ldots, m_{k\cdot j}])$;
   else $s_i' = \phi(s_i, \infty)$.
   $\mathbf{S}' = \mathbf{S}' + \{s_i'\}$.

ALGORITHM 6. The extracting algorithm of the generalized TLSM

1) Given a faked sequence $\mathbf{S}'$, a reference sequence $\mathbf{S}$, and a reversion function ϕ':
   - $\mathbf{S}' = \{s_1', \ldots, s_n'\}$, $\mathbf{S} = \{s_1, \ldots, s_n\}$.
   - $\phi'(s, s')$ returns either a k-bit data or the signal ∞.
2) Initialize the secret binary message $\mathbf{M}$ as an empty set.
3) **for** integer i from 1 to n,
   if $\phi'(s_i, s_i')$ is a k-bit binary message, **then** $\mathbf{M} = \mathbf{M} + \{\phi'(s_i, s_i')\}$.

5. **General Cases of the TLSM.** If the reference sequence is not necessarily a real DNA sequence, the TLSM can be further extended to more general cases. Assuming that we have a kind of sequences composed of $2^k + 1$ different letters, the proposed rule table is capable of hiding a k'-bits secret message ($k' \leq k$) in a letter and extra location information is not required in the extracting procedure. Moreover, if the number of distinct letters of a sequence, h, falls in $(2^k+1, 2^{k+1}]$, we can legally define at most a k-bit rule table. And there will be $h - (2^k + 1)$ unused letters in the table (cf. Table 5). The embedding and extracting algorithms of the generalized TLSM are given in Algorithm 5 and Algorithm 6, respectively.

5.1. **An example of $k = 2$.** In order to explain the generalized TLSM more clearly, we use an artificial sequence with $k = 2$, i.e., 5 distinct letters are used to construct a sequence. The pseudo sequence we used here is derived by adding an extra letter X to AGCT letters of DNA. A possible rule table is defined in Table 4. Example 2 illustrates

TABLE 4. The rule table for the pseudo sequence, where ∞ denotes that nothing is hidden

A			C			G			T			X		
msg		*sbs*	*msg*		*sbs*	*msg*		*sbs*	*msg*		*sbs*	*msg*		*sbs*
0	0	C	0	0	A	0	0	T	0	0	X	0	0	G
0	1	A	0	1	C	0	1	G	0	1	T	0	1	X
1	0	T	1	0	G	1	0	X	1	0	A	1	0	C
1	1	G	1	1	X	1	1	A	1	1	C	1	1	T
∞		X	∞		T	∞		C	∞		G	∞		A

EXAMPLE 2. The generalized TLSM of $k = 2$, where $n = 15$, $p = 14$

Embedding:

Given an artificial sequence **S**, a secret message **M**, and a location set **A**:

- **S** = ACXTGCXTTGACXAG;
- **M** = 01110100101110;
- **A** = $2, 3, 5, 10, 12, 13, 15$;

Using the rule table in Table 4, the faked sequence $\mathbf{S}'$ is XCTGGTAGGTXGTXX.

Extracting:

Given:

- $\mathbf{S}'$ = XCTGGTAGGTXGTXX
- **S** = ACXTGCXTTGACXAG

Using Table 4, the extracted secret message **M** is 01110100101110.

the embedding and extracting algorithms of the generalized TLSM to hide the 14-bit secret message used in Example 1 for the case of $k = 2$.

5.2. **A special case.** In the following, we will show that the original substitution method is a special case of the generalized TLSM. We focus on the situation when the number of distinct letters of a sequence falls in $(2^k + 1, 2^{k+1}]$. As we mentioned before, we can at most define a k-bit rule table for this sequence. Since there are four distinct letters in the DNA sequence, the DNA sequence belongs to the situation of $k = 1$ and a 1-bit rule table is used. It is obvious that the rule table can actually cover all original substitution combination representations. As an example, the rule table for the complementary rule (AT)(TG)(GC)(CA) used in the example in Section 2 is shown in Table 5. Therefore, we could treat the original substitution method as a special case of the generalized TLSM. Note that there are $(4!)^4$ possible tables, which is larger than the number (6) of complementary rules.

5.3. **Base-*t* TLSM.** To fully utilize the substitution table, we can encode the secret message using a base-t representation, i.e., the radix is t (cf. Table 6). Then, for a generalized base-t TLSM, if the sequence has $h = t^k + 1$ distinct letters, we can embed k secret symbols in the base-t representation. Whatever the number of distinct letters may be, we can always find a proper t and k to make the letter conversion in the rule table be fully utilized. For instance, the DNA sequence can be considered as a sequence of $4(= 3^1 + 1)$ distinct letters, so we can encode the message in radix 3. If the message 01110100101110_2 is encoded with ternary representation as 101020200_3, it becomes shorter and we can potentially hide more data into the sequence. The example of a rule table is shown in Table 6.

TABLE 5. The rule table equivalent to the complementary rule in the original substitution method, where ∞ and $\times$ denotes that nothing hidden and conversion unused, respectively

A		C		G		T	
msg	*sbs*	*msg*	*sbs*	*msg*	*sbs*	*msg*	*sbs*
0	A	0	C	0	G	0	T
1	T	1	A	1	C	1	G
∞	G	∞	T	∞	A	∞	C
$\times$	C	$\times$	G	$\times$	T	$\times$	A

TABLE 6. A rule table for the TLSM with the secret message in base-3 representation, where ∞ denotes that nothing hidden

A		C		G		T	
msg	*sbs*	*msg*	*sbs*	*msg*	*sbs*	*msg*	*sbs*
0	A	0	C	0	G	0	T
1	T	1	A	1	C	1	G
2	G	2	T	2	A	2	C
∞	C	∞	G	∞	T	∞	A

ALGORITHM 7. The embedding algorithm of the generalized Base-t ETLSM

1) Given a reference sequence $\mathbf{S}$, a binary secret message $\mathbf{M}$ and a conversion function ϕ:
   - $\mathbf{S} = \{s_1, \ldots, s_n\}$, where s_i can be one of the $h = t^k + 1$ different letters.
   - $\mathbf{M} = \{m_1, \ldots, m_{k \cdot p}\}$,where m_i is in the base-t representation and $p \leq n$.
   - $\phi\left(s_i, [s_{f(1)}, \ldots, s_{f(c)}], [m_j, \ldots, m_{j+k-1}] \text{ or } \infty\right)$ returns a letter, where $f(x) = (i + x) \bmod n$, $x = \{1, \ldots, c\}$, ∞ denotes that nothing is hidden, c is the number of additional letters, and s_{i+v} is the v-th additional letter of s_i.
2) Initialize the faked sequence $\mathbf{S}'$ to be an empty sequence.
3) Randomly generate a sorted ascending distinct integer set $\mathbf{A} = \{a_1, \ldots, a_p\}$, $a_j \leq n$.
4) **for** integer i from 1 to n,
   if $i = a_j$, **then** $s'_i = \phi\left(s_i, [s_{f(1)}, \ldots, s_{f(c)}], [m_{k \cdot (j-1)+1}, \ldots, m_{k \cdot j}]\right)$;
   else, $s'_i = \phi\left(s_i, [s_{f(1)}, \ldots, s_{f(c)}], \infty\right)$.
   $\mathbf{S}' = \mathbf{S}' + \{s'_i\}$.

5.4. **Extended TLSM.** In order to further increase the safety and capacity levels of the TLSM, here we propose an extended TLSM (ETLSM), which takes one or more neighboring letters into consideration. The ETLSM treats the additional letters as auxiliaries to select rule tables. If additional c letters are used, the number of possible rule tables for the Base-t ETLSM will be $[(h!)^h]^{h^c}$, where $h(= t^k + 1)$ is the number of distinct letters. The advantage of the ETLSM is that the rule table is fully utilized and the cracking probability can be very low. The blind guessing probability is $1/h^n$, where n is the length of the faked sequence. Assuming that c cyclic right neighbors of the target letter are

ALGORITHM 8. The extracting algorithm of the generalized Base-t ETLSM

1) Given a faked sequence $\mathbf{S}'$, a reference sequence $\mathbf{S}$ and a reversion function ϕ':
   - $\mathbf{S}' = \{s'_1, \ldots, s'_n\}$ and $\mathbf{S} = \{s_1, \ldots, s_n\}$.
   - $\phi'(s_i, s'_i, [s_{f(1)}, \ldots, s_{f(c)}])$ returns either k base-t data or the non-hidden signal ∞, where $f(x) = (i+x) \bmod n$, $x = \{1, \ldots, c\}$ and s_{i+v} is the v-th additional letter of s_i.
2) Initialize the secret binary message $\mathbf{M}$ to be an empty set.
3) **for** integer i from 1 to n,
   if $\phi'(s_i, s'_i, [s_{f(1)}, \ldots, s_{f(c)}])$ returns k base-t data,
   then $\mathbf{M} = \mathbf{M} + \{\phi'(s_i, s'_i, [s_{f(1)}, \ldots, s_{f(c)}])\}$.

selected as the additional letters, the embedding and extracting algorithms of the Base-t ETSLM are described in Algorithm 7 and Algorithm 8, respectively.

In the following, we use the DNA sequence as an illustration. Assume that the message is encoded in base 3 and the number of additional letters is one for the Base-t ETLSM. With the number of permutations of choosing c letters in $n-1$ elements, the cracking probability with the real DNA database is

$$\begin{aligned} P_1^{\text{ETLSM}} &= \frac{1}{1.63 \times 10^6} \times \frac{1}{24^{16}} \times \frac{(n-1-c)!}{(n-1)!} \\ &= \frac{1}{1.63 \times 10^6} \times \frac{1}{24^{16}} \times \frac{1}{n-1}, \text{ when } c = 1 \end{aligned} \tag{8}$$

and the blind guessing probability is

$$P_2^{\text{ETLSM}} = \frac{1}{4^n}. \tag{9}$$

Table 7 shows an example of substitution table of the Base-3 ETLSM with $c = 1$. Example 3 is an example based on Table 7 using the DNA sequence in Example 1. Table 8 shows the comparison of several methods using DNA sequences. To evaluate the capacity performance of substitution methods, we use the index bits-per-letter (*bpl*) to indicate how many bits can be embedded into one letter. The *bpl* is defined as:

$$bpl = \log_2 y, \tag{10}$$

where y is the number of symbols can be hidden in a single letter. For instance, y is equal to 4 and t^k for the TLSM and the Base-t TLSM, respectively. Table 9 lists the comparison of data hiding algorithms on the equivalent *bpp*. The scheme proposed by Lee and Chen is an irreversible data-hiding method which provides high capacity performance (*bpp* up to 4) using image-based methods. However, it is obvious that high *bpp* leads to poor PSNR. As shown in Table 9, data hiding algorithms based on DNA sequences have better embedding efficiency.

5.5. **Data-hiding example using Internet.** In the following, we introduce a scenario of a real world data-hiding application based on Internet as shown in Figure 1. This system involves web services that process requests/responses for clients and the DNA database. The clients can be laptops, PCs, mobile phones, etc. The DNA database can be a real database like NCBI and EBI or a private (artificial) database. In this system, the client requests a reference DNA sequence from the server and then embeds the secret messages in it. The embedded DNA sequence will be sent back to the server and stored in the DNA

TABLE 7. A substitution table of the Base-3 ETLSM with $c = 1$ using DNA sequence, where "A/C/G/T+A" means we substitute letters for A, C, G, or T according to the additional letter A. That is, if the additional letter is A, the rule table in (a) is adopted.

(a) The additional letter is A

A+A		C+A		G+A		T+A	
msg	*sbs*	*msg*	*sbs*	*Msg*	*sbs*	*msg*	*sbs*
0	G	0	T	0	T	0	G
1	A	1	G	1	C	1	T
2	T	2	C	2	A	2	C
∞	C	∞	A	∞	G	∞	A

(b) The additional letter is C

A+C		C+C		G+C		T+C	
msg	*sbs*	*msg*	*sbs*	*msg*	*sbs*	*msg*	*sbs*
0	C	0	A	0	A	0	G
1	A	1	C	1	T	1	C
2	T	2	T	2	G	2	A
∞	G	∞	G	∞	C	∞	T

(c) The additional letter is G

A+G		C+G		G+G		T+G	
msg	*sbs*	*msg*	*sbs*	*Msg*	*sbs*	*msg*	*sbs*
0	C	0	T	0	A	0	G
1	T	1	A	1	G	1	T
2	G	2	C	2	C	2	A
∞	A	∞	G	∞	T	∞	C

(d) The additional letter is T

A+T		C+T		G+T		T+T	
msg	*sbs*	*msg*	*sbs*	*msg*	*sbs*	*msg*	*sbs*
0	A	0	C	0	T	0	C
1	C	1	G	1	A	1	G
2	G	2	T	2	G	2	A
∞	T	∞	A	∞	C	∞	T

database for other clients to access. Compared with the original substitution method, the TLSM only requires half of the DNA sequence when hiding the same message according to the analysis of the average *bpn* in Section 4.1. Therefore, the TLSM can make the Internet-based data-hiding system more efficient in transmission.

As an example, we hide a 32-character secret message "The jewels are in the fireplace." using this system. This string can be converted to a 256-bit binary code using the standard ASCII code. In the original substitution method, it needs a reference DNA sequence which has 256 nucleotides at least. If a 256-nucleotide reference sequence is used, the system will generate a 256-nucleotide faked sequence. Assuming that each nucleotide is encoded with a 2-bit binary code, each communication between the server and client needs to transmit 64 bytes. For the TLSM, only 128 nucleotides are needed. Therefore, each communication transmits 32 bytes. As a result, the TLSM saves 50% bandwidth compared with the original substitution method. Furthermore, this system can be applied to other kinds of databases easily with the proposed generalized TLSM and the Base-t ETLSM. For

example, if the database is composed of lowercase Greek alphabets, a base-23 ETLSM can be adopted. Assume that the additional letter is not considered here, the substitution table can be defined as in Table 10. Since the reference sequence is not available to the public, the cracking probability and the *bpl* are $1/24^n$ and 4.524, respectively, where n is the length of the reference sequence.

TABLE 8. Comparisons of methods using DNA sequences (P_1 and P_2 represent the cracking probability based on the real database and the blind guessing probability, respectively)

	Locations required	P_1	P_2	*bpl*
Substitution method [2]	no	$\frac{1}{1.63\times10^6} \times \frac{1}{6}$	$\frac{1}{3^n}$	1
2-bit TLSM with r location sets	yes	$\frac{1}{1.63\times10^6} \times \frac{1}{24^4} \times \frac{1}{r}$	$\frac{1}{5^n}$	2
Base-3 TLSM	no	$\frac{1}{1.63\times10^6} \times \frac{1}{24^4}$	$\frac{1}{4^n}$	1.585
Base-3 ETLSM with $c = 1$	no	$\frac{1}{1.63\times10^6} \times \frac{1}{24^{16}} \times \frac{1}{n-1}$	$\frac{1}{4^n}$	1.585
Base-3 ETLSM with $c = 2$	no	$\frac{1}{1.63\times10^6} \times \frac{1}{24^{64}} \times \frac{(n-3)!}{(n-1)!}$	$\frac{1}{4^n}$	1.585

TABLE 9. Comparisons of the methods with the equivalent *bpp*

	Medium	**Approaches**	**Eq. *bpp***
Lee and Chen [8]	image	Modulus functions scheme (system parameter = 8, PSNR = 37.9)	3
		Modulus functions scheme (system parameter = 16, PSNR = 31.8)	4
Shiu et al. [2]	DNA	Substitution method	4
Our work	DNA	TLSM	8
		Base-3 TLSM	6.34

EXAMPLE 3. An example of the Base-3 ETLSM using the DNA sequence in Example 1 based on Table 7

Embedding

Given a reference DNA sequence **S**, a ternary secret message **M**, and a location set **A**:

- **S** = ACGGAATTGCTTCAG.
- **M** = 101020200_3 encoded from 01110100101110_2.
- **A** = $2, 3, 5, 7, 8, 10, 12, 13, 14$.

Mapped with Table 7, the faked DNA sequence **S′** is GAAGATCACCTATCG.

Extracting

Given:
S′ = GAAGATCACCTATCG.
S = ACGGAATTGCTTCAG.
Mapped with Table 7, the extracted secret massage **M** is 101020200.

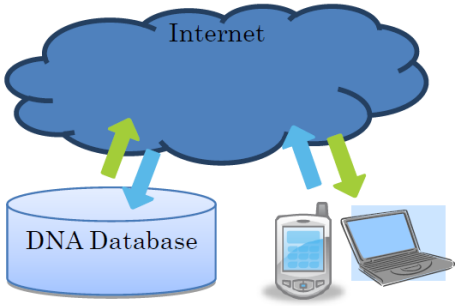

FIGURE 1. Diagram of the Internet data-hiding application

TABLE 10. A substitution table of 24 Greek alphabets for the base-23 TLSM

α		β		...		ω	
msg	*Sbs*	*msg*	*sbs*	*msg*	*sbs*	*Msg*	*sbs*
0_{23}	α	0_{23}	ψ	0_{23}	χ	0_{23}	φ
1_{23}	β	1_{23}	ω	1_{23}	ψ	1_{23}	χ
...	...	...	...	...	...	...	...
22_{23}	ψ	22_{23}	φ	22_{23}	υ	22_{23}	τ
∞	ω	∞	χ	∞	φ	∞	υ

6. **Conclusion.** Compared with other existing methods, the original substitution method provides good performance in either the *bpn* or the cracking probability [2]. Based on their approach, we propose the TLSM to improve the effectiveness of the original substitution method by replacing the 1-bit complementary rule with the 2-bit rule table. Experimental results in Section 4 show that the TLSM provides better performance in both capacity and security. Moreover, we illustrate that the original substitution method is a special case of the general TLSM. In addition, we further improve the TLSM by using a base-t representation of the secret message and extend the TLSM by taking more letters into account. In the Base-t TLSM, the secret message is encoded with a proper radix to fully utilize the substitution table. As to the ETLSM, it can further enhance the cracking probability of the Base-t TLSM. According to Table 8, it can be observed that all the proposed methods provide improvement over the original substitution method in cracking probability and bits-per-letter.

Acknowledgment. This work was supported by the Taiwan National Science Council through Grant NSC 100-2221-E-005-081.

REFERENCES

[1] F. A. P. Petitcolas, R. J. Anderson and M. G. Kuhn, Information hiding – A survey, *Proc. of the IEEE*, vol.87, no.7, pp.1062-1078, 1999.
[2] H. J. Shiu, K. L. Ng, J. F. Fang, R. C. T. Lee and C. H. Huang, Data hiding methods based upon DNA sequences, *Information of Science*, vol.180, no.11, pp.2196-2208, 2010.
[3] C. T. Clelland, V. Risca and C. Bancroft, Hiding messages in DNA microdots, *Nature*, vol.399, no.6736, pp.533-534, 1999.
[4] A. Leier, C. Richter, W. Banzhaf and H. Rauhe, Cryptography with DNA binary strands, *Biosystems*, vol.57, no.1, pp.13-22, 2000.
[5] B. Shimanovsky, J. Feng and M. Potkonjak, Hiding data in DNA, *The 5th Int. Workshop on Information Hiding*, 2003.

[6] C.-C. Chang, T.-C. Lu, Y.-F. Chang and C.-T. Lee, Reversible data hiding schemes for deoxyribonucleic acid (DNA) medium, *International Journal of Innovative Computing, Information and Control*, vol.3, no.5, pp.1145-1160, 2007.
[7] L. Kencl and M. Loebl, DNA-inspired information concealing: A survey, *Computer Science Review*, vol.4, no.4, pp.251-262, 2010.
[8] C.-F. Lee and H.-L. Chen, A novel data hiding scheme based on modulus function, *Journal of Systems and Software*, vol.83, no.5, pp.832-843, 2010.
[9] *Ncbi Database*, http://www.ncbi.nlm.nih.gov/.
[10] European Bioinformatics Institute, *http://www.ebi.ac.uk/*.