# High Capacity Data Hiding Method in DNA with Mutation Handling

Kevin Nathanael Santoso
Pukyong National University
Daeyeon Campus (608-737) 45, Yongso-ro,
Nam-Gu. Busan, Korea
+82-51-629-6257
kev_nathanael@yahoo.com

Ki-Ryong Kwon
Pukyong National University
Daeyeon Campus (608-737) 45, Yongso-ro,
Nam-Gu. Busan, Korea
+82-51-629-6257
krkwon@pknu.ac.kr

Suk-Hwan Lee
Tongmyong University
(608-711) 179, Sinseonno,
Nam-Gu, Busan, Korea
+82-51-629-1285
skylee@tu.ac.kr

Seong-Geun Kwon
Kyungil University
(712-701) 50, Gamasil-Gil,
Hayang Eup, Gyeongsan, Gyeongbuk, Korea
+82-53- 600-5544
sgkwon@kiu.ac.kr

## ABSTRACT
DNA engineering and sequencing technology have great influence to boost DNA computing world. With superior capability to record information to magnetic storage, many researches are done to achieve the maturity of DNA storage and secure communication. The new data embedding method explained in this paper is designed to insert binary data into noncoding regions of DNA sequence in order to preserve its biological function. To reach a high standard of security and capacity, data is encrypted, translated, and then encoded accordingly into DNA sectors using a reference-dependent mechanism. Error correction method using Reed-Solomon encoder and parity check gives a good error handling against substitution and insertion or deletion (*indel*) mutations. Simulations show that proposed method has higher bit-per-nucleotide (*bpn*) value even it is restricted to noncoding regions only.

## Categories and Subject Descriptors
E.m [**Miscellaneous**]

## General Terms
Algorithms, Security.

## Keywords
Capacity, DNA steganography, DNA storage, error correction, noncoding region.


*IWIHC'14*, June 3, 2014, Kyoto, Japan.

http://dx.doi.org/10.1145/2598908.2598911

## 1. INTRODUCTION
The tiniest functional part of living things in fact holds great potentials for rapidly developing digital world. Constructed by chains of monomeric subunits called nucleotides, DNA (Deoxyribonucleic acid) contains the biological information to maintain organism's living [1]. This information is still readable even after hundreds of years. By mutating its molecular sequence, desired foreign information can be encoded to the host DNA. In their publication, Wong, et al. [2] stated that providing a living host, which is able to grow and multiply, could provide adequate protection for encoded DNA to survive even extreme environmental conditions. Thus, DNA is a potential medium for long-term data storage.

There are 4 types of DNA nucleotide attached to double-helix sugar phosphate "backbone": Adenine (A), Thiamine (T), Cytosine (C), and Guanine (G). Unlike magnetic or semi-conductor storage commonly used now, each DNA nucleotide records 1 out of 4 bases instead of binary '0' or '1'. Therefore, a string of $n$ elements holds $4^n$ combinations instead of $2^n$ – which shows DNA's strength in capacity point-of-view.

Steganography is the art of hiding message or data secretly, well enough that unauthorized party do not suspect the presence of hidden message at all [3]. With absorbed definition, DNA steganography aims to secretly hide information using DNA as its medium. Since Clelland, et al. published a DNA steganography technique for sending classified message secretly using microdot, many improvements has been done [4-10]. DNA steganography is still developing until it achieves desired level of standards. Standard steganography requirements are: (1) invisibility, it should be undetectable by a 3rd party, (2) capacity, it must have sufficient embedding capacity to contain the message, (3) integrity, after embedding the information must not change, and (4) robustness against modifications [11],[12]. Therefore, a new DNA steganography method is presented which will satisfy mentioned requirements.

DNA strand is grouped into two parts: coding regions, which contain amino acid templates for protein synthesis, and noncoding regions, which role is still unclear. Noncoding region does not contain organism's functional information for living, yet in some species, it takes great part of whole DNA strand. Assumed that noncoding regions can be mutated without disturbing any protein synthesis process, it will provide a relatively large space to store foreign information which in this case binary data. Drawbacks of using DNA as medium of our data are: (1) DNA is organic compound, which is constructed by molecules. After some time, there is possibility that mutagens alter the DNA sequence, consequently make embedded data unreadable. (2) DNA sequencing technologies are not 100% free of errors. To handle this situation, data must be initially enforced by error correction method.

This paper will explain a method to embed digital data into DNA strand. The rest of this paper is organized as follows: Section 2 gives brief description about previous works. Section 3 explains main features and constraints of proposed method. Section 4 explains data embedding method while Section 5 explains how to retrieve it. Section 6 gives simulation results, and the last is Section 7 with the conclusion.

## 2. RELATED WORKS

The first well-known steganography technique using DNA as its medium is the microdot technology by Clelland, et al [4]. The microdot contains DNA strands with secret characters, numbers, and symbols encrypted inside. By using a translation table, characters are represented by DNA triplets and then flanked by PCR primer sequences. In 2013, Wang, et al. brought back Clelland's idea of using microdot to send half of the encrypted message, while the other half is sent via public channel [5]. First, message is encrypted by Vigenere cryptography into ASCII code and then separate odd and even sequence number. The even number sequence is encoded by substitution table into DNA triplets, and then concealed into human DNA and confined into a microdot. If the microdot is received without interruption or damage, the other half is sent publicly.

Leier, et al. proposed a method to encode binary information into DNA sequence [6]. A short DNA sequence represents bit '1' and another DNA sequence represents bit '0'. To create the full binary information, those short DNA sequences are arranged accordingly, concatenated, and then marked by start and stop DNA sequences as marker for decoder to start and stop reading. Then Heider, et al. proposed DNA Crypt algorithm which had better capacity efficiency for binary information than previous works as well as error handling [7]. He encoded each two binary data into one nucleotide letter: 00='T', 01='G', 10='C', 11='A' and these letters were embedded into coding region of original DNA based on synonymous codon principle. Concerning the probability of mutations, Hamming code and WDH error correction method are implemented under a fuzzy controller to decide which method should be used or no correction method is required.

In 2010, Shiu, et al. introduced three binary data hiding methods into DNA from digital point of view, not biological view: Insertion method, Complementary pairs method, and Substitution method [8]. In Insertion method, DNA strand is translated into binary array according to a translation rule and then message bits are inserted every after a number of bits. The new array is translated back to nucleotide letters creating a new fake DNA strand. In Complementary pairs method, message bits are also translated into nucleotide letters. The algorithm creates a unique sequence of nucleotides that must be the longest sequence in the strand, and then create its complementary pair. Those sequences are padded by a nucleotide, and then message nucleotides are inserted in front of it one at a time and concatenated with the rest of the sequences without overlapping. The third method, Substitution method, encodes binary message in random position of original DNA depends on its binary value. If message bit is '0', the nucleotide in corresponding position does not change. If message bit is '1', change the nucleotide in that position to its complementary pair.

The next year, Haughton, et al. gave more concern about mutations that may occur to DNA as its carrier [9]. They recommended Repetition coding to enforce data from errors, which might be caused by substitution or *indel* (insertion or deletion) mutation. Binary data is simply translated into nucleotide letters and then embedded into noncoding regions of host DNA. Tulpan, et al. proposed a new hybrid DNA encryption (HyDEn) approach which uses DNA Hamming code as error correction method [10]. They created a set of DNA Hamming code words and then randomly assigned one ASCII characters to one/many code words each. ASCII characters message is translated into a string of DNA letters and then a circular permutation is applied. The final DNA string is embedded into host DNA.

In this paper, we propose a new method while keep in mind important features from previous ones. This method encodes binary message instead of ASCII characters with higher capacity efficiency than others. We also propose different method to deal with mutations and errors.

## 3. FEATURES AND CONSTRAINTS

In this section, we give notable features as well as constraints while inserting foreign information into DNA. There are steganography requirements to achieve, DNA properties to preserve, and type of mutation to take into consideration.

1. To secure the data, it is encrypted by cryptographically secure algorithm and then encoded into DNA strand by different rules for each sectors.
2. Encoding algorithm has high rate of space usage efficiency with more than 1 data bit per nucleotide ($bpn \geq 1$).
3. Data can be retrieved only if secret keys are known.
4. Original and encoded DNA strand has the same sequence length.
5. Data is embedded into noncoding regions only to preserve protein-coding regions.
6. Encoded noncoding regions must not create falsely recognized "fake" protein-coding regions.
7. Data retrieval is mutation resistant. Both nucleotide substitution and *indel* mutation are considered.

Point 1 until 4 will be explained further in Section 4. Point 5 and 6 are meant to keep the host organism's biological functions. Appearance of unexpected start codon must be prevented such that cell's protein translation agents read and translate the real

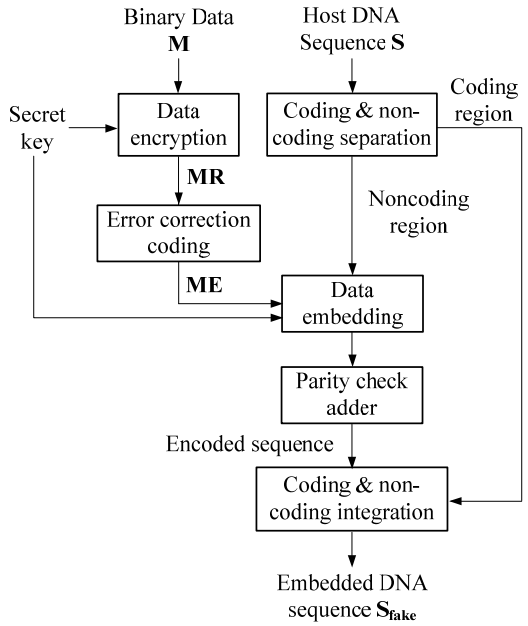


**Figure 1. Block diagram for data embedding.**

protein coding regions only [13]. This is a challenge for encoding algorithm. Nucleotide substitution mutations mentioned at point 7 refers to singular base mutation, multiple base mutation at random position, and burst of neighborhood bases mutations. We also propose an idea to deal with *indel* mutation while keeping the data capacity as high as possible, further explanation at Section 4.

# 4. EMBEDDING METHOD

As steganography technique does not guarantee the security of our data, firstly the digital data in form of binary array **M** is encrypted using secret key. Encrypted data **MR** goes through error correction encoding and mutation handling process before it is ready to be encoded into host DNA's noncoding region, denoted by **ME**. Encoding process also use certain key which is needed for data decoding. Noncoding region with embedded data is recombined with coding regions to be one complete DNA sequence $\mathbf{S}_{\mathbf{fake}}$. Figure 1 describes the flow of embedding process.

## 4.1 Encryption

We can simply secure our binary data using XOR cypher. An array of random bits **R** is generated by cryptographically secure pseudo-random bit generator (PRBG) such as Blum-Blum-Shub that has been analyzed for its security [14-16]. The seed of PRBG is kept secret as key ($R_{seed}$). Determine $L_R$, the length of random bits we desire, then use $L_R$ bits of **R** repeatedly as shown in equation (1) to get **MR** for $k$=0,1,2,… until it covers the length of message **M**.

$$\mathbf{MR}(1+k.\ L_R : (k+1).\ L_R) = \mathbf{R} \oplus \mathbf{M}(1+k.\ L_R : (k+1).\ L_R) \qquad (1)$$

**Table 1. Translation agreement table**

| Letter | Integer | Binary |
|---|---|---|
| A | 0 | 10 |
| T | 1 | 00 |
| G | 2 | 11 |
| C | 3 | 01 |

## 4.2 Translation Agreement

Translation agreement is required for both encoding and decoding process. Nucleotide letter (x), decimal number (y), and binary digits (z) are translated using a secret translation table which is predetermined by authorized party. Table 1 is given as an example, which will also be used as our reference for the rest of this paper. There are 4!×4! different possible combinations of ($x$, $y$, $z$) to use in order to lower the chance of unauthorized party to decode our data. For $x \in \{A,T,G,C\}$, $y \in \{10,00,11,01\}$, and $z \in \{0,1,2,3\}$, the relation of $x$, $y$, and $z$ is

$$z = f(x) \text{ and } x = f^{-1}(z), \qquad (2)$$

$$z = h(y) \text{ and } y = h^{-1}(z). \qquad (3)$$

Function $f(x)$ and $f^{-1}(z)$ translates a nucleotide letter to an integer and vice versa. Function $h(y)$ and $h^{-1}(z)$ translates two consecutive binary numbers to an integer and vice versa.

## 4.3 Data Sectors

Noncoding DNA regions are separated from coding regions. Data is encoded into noncoding regions, sequence by sequence. For every noncoding DNA region, we allocate short data sectors with $p$ data bases each, as shown in Figure 2. The length of data sector ($L$) is $p$+2 bases. For $i$ represents $i$ th data sector and $j$ = 1,2,…,$L$, $b_{i,j}$ represents $j$ th element of nucleotide base in the corresponding sector $i$. First base at each sector represents the reference letter $S_i$ and the last base represents the parity check letter $P_i$.

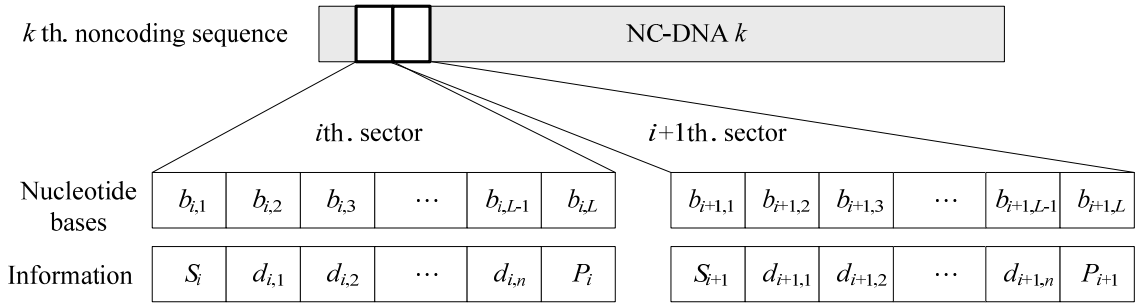


**Figure 2. Structure of data in each sector.**

## 4.4 Substitution Error Correction

There are lots of option of error correction technique such as mentioned at [7], [9], and [10], but we want to focus on Reed-Solomon code because:

1. RS code is non-binary error correction code which works in symbols of GF($2^m$). It can detect and correct multiple errors including burst bit error.
2. Check symbols do not consume big part of overall data so it is more space efficient.
3. Each block has length $n$ symbols consists of $k$ data symbols and ($n$-$k$) check symbols. It is flexible to choose its error correction strength, whether to be compact data oriented or error free oriented.

Deciding RS code parameters for data embedded into DNA may depend on: 1) data size, 2) predicted mutation rate, and 3) processing time. Since there is no exact rule about how to choose $n$ and $k$, considering the trade-off between capacity or error correction, a fuzzy logic system can be used. The fuzzy logic has three input dimensions as mentioned above and two output dimensions: elements of Galois field ($2^m$) and number of check symbols required in each block.

Data size as the first input dimension is separated into 3 sets: "short" for data size less than $10^6$ bits, "medium" for data size between $10^6$ and $10^9$ bits, and "long" for data more than $10^9$ bits. This is based on great variation of organism's DNA length which spans from around $10^5$ to more than $10^{11}$ bases [17] and assumption that 1 base able to store 1 bit of data (1 *bpn*).

Second input dimension is presumed mutation or error, which may happen when DNA is stored, transmitted, or sequenced. There are also 3 sets: "low" for possibility of error less than $10^{-8}$, "middle" between $10^{-8}$ and $10^{-4}$, and "high" for possibility more than $10^{-4}$. This is based on common mutation rate of living organisms and also Next-generation Sequencing (NGS) error rates [7],[9],[18],[19]. Third input dimension is represented by 2 sets: "fast" processing time which will suggest the shortest symbol length and the lowest error correction capability for RS encoder, and "moderate" with more flexible options.

By a set of defuzzification rules, it suggests one m value from 4, 6, 8, or 10, and error check symbol allocated space whether it be "smallest" for less than 0.2%, "small" for less than 0.78%, or "regular" for more than 0.78% up to 2% off the block length. For example if "$m$=6" and "smallest" error check allocation are selected, Reed-Solomon code suggested is RS(63,61) with only 2 symbols used for error check. While if "$m$=10" and "small" are selected, RS(1023,1015) is the best option.

This method able to handle singular base mutation as well as multiple base mutation, which scatter in random position. One mutated nucleotide base is translated into maximum of two binary errors. Because $m$ value is more than 4, only one symbol is affected and RS decoder can handle it very well. In case of burst mutation, $c$ consecutive mutated bases are translated to $2c$ binary errors. As long as $2c \leq m$, mutated bases only affect one symbol. But if $2c > m$, it depends on correction capability of RS encoder since more than one symbol is affected. When *indel* mutation happens, RS decoder is ineffectual. It requires another method to handle the situation so we propose additional method in subsection 4.6.

## 4.5 Encoding Rules

Encrypted data **MR** is enforced by Reed-Solomon encoder as explained above resulting binary data **ME**. **ME** is translated into nucleotide base letter $d_{i.k}$ depending on each sector's reference base $S_i$ as follows:

$$\mathbf{ME'} = h(\mathbf{ME}) \tag{4}$$

$$d'_{i.k} = \{f(S_i) + \mathbf{ME'}(k)\} (\mathrm{mod}\, 4),\ k=1,2,..,p \tag{5}$$

$$d_{i.k} = f^{-1}(d'_{i.k}) \tag{6}$$

For example, we set $p$ = 4 for a given noncoding DNA strand sequence **S**="**T**AGACT**C**GGCTC" and **ME'** = 21301301. First sector has letter 'T' as its reference so according to (5) the first four $d'$ is 3201. Translate these integer numbers to nucleotide letters according to Table 1 to get $d$ = "CGAT". Likewise, do the same for second sector with reference letter 'C' to get the next four $d$ = "AGCA". For now, we get the encoded DNA $\mathbf{S_{fake}}$="**T**CGATT**C**AGCAC".

There are 4 possibilities of reference letter for one sector so it is expected to be different from one sector to its next or previous one. Each message is encoded based on each sector's reference, which resulting to diversely assign message letter to its binary representative. In decoding process, sector with reference letter $S_i$ = 'T' translates nucleotide letter 'A' to '01' while in sector with reference letter $S_i$ = 'G', nucleotide letter 'A' is translated to '11'.

Determining $p$ has to consider trade-off between capacity and error correction efficiency. Using bigger $p$ means fewer nucleotide bases needed to store data as shown in Figure 3. Nevertheless, in case of mutation or error happen to the reference base of a certain sector, it is best to use $p = (m/2)$ so it will only cost 1 symbol error instead of 2 or more. In addition, $p$ is the key to decode the data correctly so both $p$ and translation table has to be kept secret.

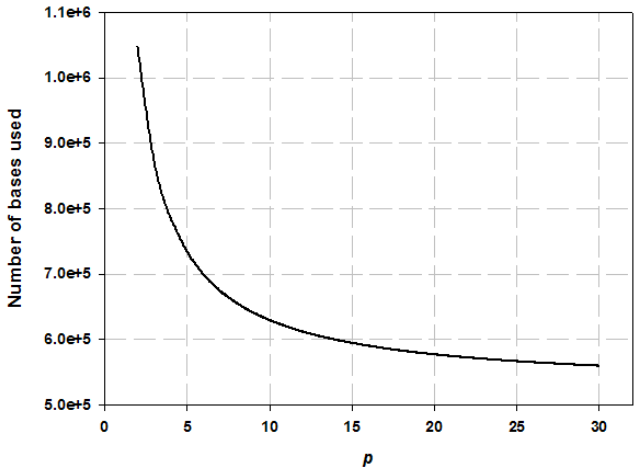


**Figure 3. Graph of nucleotide base used to encode 1 Mbit data using $p$ vary from 2 to 30.**

## 4.6 *Indel* Mutation Preempt

*Indel* mutation is the least possible mutation happen but the damage caused to data is the most obstructive. We put a parity check letter $P_i$ at the end of each sector using equation (7) and (8) as follows:

$$P_i^{'} = \left\{ \sum_{j=1}^{L-1} f(b_{i,j}) \right\} (\text{mod}\, 4) \quad (7)$$

$$P_i = f^{-1}(P_i^{'}) \quad (8)$$

Use example at previous subsection, $\mathbf{S_{fake}}$="**T**CGAT**T**CAGCA**C**" before parity check letter is applied. Adding integer values of first 5 letters in sector 1 including its reference base $S_i$, $\{f(\text{T}) + f(\text{C}) + f(\text{G}) + f(\text{A}) + f(\text{T})\}$ gives {1+3+2+0+1} = 7 ≡ 3 (mod4). According to Table 1, parity check value 3 is translated to 'C'. Do the same for sector 2 and we get parity check letter 'A'. Finally we get the final $\mathbf{S_{fake}}$="**T**CGAT**C**CAGCA**A**". Parity check letter is useful to detect consecutive sectors error, which indicates reading frame misalign caused by an inserted nucleotide base or a missing one.

Haughton and Balado mentioned in their paper that it is not desired that any fake start codon appear at neither sense strand nor antisense strand of DNA [13]. It means the appearance of standard start codons and their complementary codons must be avoided. However, there are many alternative start codons discovered and analyzed which varies by its species [20]. Unless there is sufficient information, it is best to refer to the list of non-ATG (or AUG for mRNA) start codons provided by [21] or other resources for more accurate result.

We would like to apply this constraint without losing the consistency of our encoding rule. After *i* th sector encoding finished, the encoder screens (*i*-1) th and *i* th sector to search for any undesired start codons. If any start codon exists, encoder choose another reference letter from *x* to replace current reference letter $S_i$, repeat encoding rule, replace parity letter $P_i$, and rescreen both sectors until no start codon exists. This will not reduce the performance of parity check letter to detect *indel* mutation.

## 5. RETRIEVING METHOD

To retrieve data from embedded DNA strand $\boldsymbol{S_{fake}}$, we need the agreement table, *p* number, $R_{seed}$, and also RS decoder parameters (*n* and *k*). As shown in Figure 4, firstly we separate noncoding DNA regions from coding DNA regions. The following proces is the reverse of embedding process, comply with steps below.

1. We receive encoded DNA $\mathbf{S_{fake}}$ = "TCGATCCAGCAA" and we know *p* number, which means for each sector, the reference letter at first base is followed by *p* data nucleotides and one parity check. First thing to do is to check the parity letters for any sign of *indel* mutation. Use equation (7) and (8) to compare calculation results with existing parity check letters. For a longer DNA strand, we can determine whether there is an *indel* mutation by observing the error pattern shown by these parity checks. A missing or inserted nucleotide base will cause a frame shift to our decoder such that parity checks at occurring sector until the last sector of mentioned noncoding DNA region show a near consecutive error.

   Two *indel* position candidates are *i* th sector, which shows first error, and (*i*-1) th sector. Even if the position of occurring sector is known, the exact base position of inserted or deleted nucleotide is unknown; what letter is inserted or deleted is also unknown. But this is not a crucial issue because we can rely on RS decoder to handle this situation later. To remove the inserted base or to add the missing base is a trial-and-error process. Decoder tries to insert any base to *i* th sector and reevaluate the parity check. If no more error shown, then the problem is solved. Otherwise, decoder tries to remove any base from *i* th sector and reevaluate. Unless the problem is solved, decoder do the same treatment to (*i*-1) th sector.

2. Decode binary array **ME** from $\mathbf{S_{fake}}$ by equation below, also based on Table 1.

$$\mathbf{ME'}(k) = \{f(d_{i,k}) - f(S_i)\} (\text{mod}\, 4),\ k=1,2,..,p \quad (9)$$

$$\mathbf{ME}(k) = h^{-1}(\mathbf{ME}(k)) \quad (10)$$

3. Apply Reed-Solomon decoder to **ME** in order to correct any errors using known parameters (*n* and *k*). Thus we get **MR**, our data which is still secured with XOR crypt.

4. Regenerate PRBG to create random binary array **R** using $R_{seed}$ then apply XOR function to **MR**. Use equation (1) to calculate $\mathbf{M} = \mathbf{MR} \oplus \mathbf{R}$, then original data is decrypted.

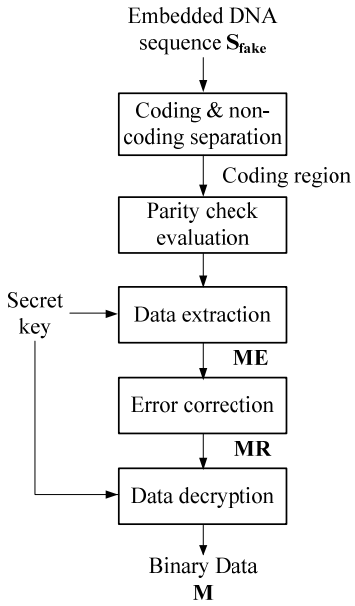


**Figure 4. Block diagram of information retrieval.**

## 6. SIMULATIONS AND COMPARISON

To evaluate space efficiency performance, we refer to Shiu's previous work in [8], where 160,000 bit of data is used to get the capacity, payload, and *bpn* values. DNA strands used in simulation are downloaded from NCBI database [22] and embedding process is done *in silico*. Table 2 shows its performance in compare with Insertion method (Ins), Complementary Pairs method (Com), and Substitution method (Sub).

Input data is analyzed by fuzzy controller to give most optimal RS encoder. Inputs for fuzzy controller are: Data size = 5.2, Mutation rate = -8.3, and Processing time = 0.5. Fuzzy controller suggests that Symbol size = 6 and Check symbols = 0.42%. Therefore, RS encoder selected is RS(63,61) and it gives 165,564 bit data as encoding result.

**Table 2. Capacity, payload, and *bpn* comparison with previous work.**

| Sequence Name | Number of Nucleotides | Category | Proposed Method | Shiu's Method | | |
|---|---|---|---|---|---|---|
| | | | | Ins | Com | Sub |
| GJ060459 *Bos taurus chromosome 1* | 213,382 | capacity | 210,601 | 293,382 | 2,533,382 | 213,382 |
| | | payload | 0 | 80,000 | 2,320,000 | 0 |
| | | *bpn* | 0.786 | 0.55 | 0.06 | 0.75 |
| GJ061116 *Bos taurus chromosome 7* | 187,446 | capacity | 185,550 | 267,446 | 2,347,446 | 187,446 |
| | | payload | 0 | 80,000 | 2,160,000 | 0 |
| | | *bpn* | 0.892 | 0.60 | 0.07 | 0.85 |
| GJ061125 *Bos taurus chromosome 7* | 221,140 | capacity | 215,512 | 301,140 | 2,541,140 | 221,140 |
| | | payload | 0 | 80,000 | 2,320,000 | 0 |
| | | *bpn* | 0.768 | 0.53 | 0.06 | 0.72 |
| NW_001502389 *Bos taurus breed Hereforf chromosome 19* | 171,877 | capacity | 168,235 | 251,877 | 2,331,877 | 171,877 |
| | | payload | 0 | 80,000 | 2,160,000 | 0 |
| | | *bpn* | 0.984 | 0.64 | 0.07 | 0.93 |

**Table 3. Maximum data load and maximum *bpn* for different *p* values.**

| DNA Sequence ID | Number of Nucleotides | NC-DNA Region Length | Maximum Data Load (bits) | | | Maximum *bpn* Value | | |
|---|---|---|---|---|---|---|---|---|
| | | | $p = 3$ | $p = 5$ | $p = 10$ | $p = 3$ | $p = 5$ | $p = 10$ |
| NC_001709 | 19,517 | 8,347 | 9,990 | 11,870 | 13,800 | 1.1968 | 1.4221 | 1.6533 |
| GJ060459 | 213,382 | 210,601 | 252,708 | 300,850 | 350,960 | 1.1999 | 1.4285 | 1.6665 |
| AE017199 | 490,885 | 38,932 | 46,056 | 54,460 | 62,240 | 1.1830 | 1.3988 | 1.5987 |
| NW_003610316 | 885,243 | 865,473 | 1,038,192 | 1,235,710 | 1,441,060 | 1.1996 | 1.4278 | 1.6651 |
| NC_006033 | 1,195,132 | 393,739 | 471,186 | 560,110 | 651,460 | 1.1967 | 1.4225 | 1.6545 |
| NW_001939315 | 1,551,108 | 1,538,850 | 1,846,506 | 2,198,180 | 2,564,260 | 1.1999 | 1.4285 | 1.6663 |
| NC_006047 | 2,007,515 | 516,557 | 617,157 | 733,290 | 851,020 | 1.1947 | 1.4196 | 1.6475 |
| CP000108 | 2,572,079 | 301,761 | 357,870 | 423,510 | 486,940 | 1.1859 | 1.4035 | 1.6137 |
| NW_001939300 | 4,840,945 | 4,308,775 | 5,167,416 | 6,149,790 | 7,169,300 | 1.1993 | 1.4273 | 1.6639 |
| CP000247 | 4,938,920 | 570,214 | 674,850 | 798,390 | 915,300 | 1.1835 | 1.4002 | 1.6052 |

**Capacity** is defined as the length of embedded DNA sequence. Proposed method has limited capacity because it is restricted to noncoding regions only. Table 2 shows that capacity of proposed method is smaller than three other method by Shiu, et al. not only because of noncoding DNA region restriction, but also because of the final length of embedded DNA does not change. The **payload** is defined as the length of extra bases added because of encoding process. Proposed method preserves the length of original DNA strand so the payload is always 0, similar with Substitution method. The ***bpn*** (bits per nucleotide) is the average number of secret bits hidden in each character of the embedded sequence. If there is many unused space remaining, the *bpn* will be lower than if it is fully used. Maximum *bpn* value achieved if there is no more space to encode data. The lowest *bpn* is Complementary pairs method with average of 0.065, next is Insertion method with 0.580, and then Substitution method with 0.813. The new method gives superior result with average value 0.858.

Available space to embed data inside DNA strand is greatly influenced by total length of noncoding regions of its host DNA, which depends on its species. Table 3 shows that DNA with ID GJ060459, which source is from *Bos taurus* or cattle, has noncoding region 98.7% of its total length. In the other hand, DNA with ID AE017199 which source is from *Nanoarchaeum equitans*, a species of marine Archaea, has shorter noncoding region (7.9%) even it has longer total DNA strand. However, maximum data to be embedded inside this limited space also depends on *p* number. The average *bpn* value for $p = 3$, 5, and 10 are 1.1938, 1.4174, and 1.6429 respectively.

Simulation also shows that without the correct *p* number, translation table, and decryption key, decoder fails to retrieve original data. In another simulation, RS decoder successfully corrects up to $(n-k)/2$ symbol errors as expected. Coding DNA regions are preserved and no error found.

Proposed method stores binary data instead of alphanumeric mapping as in [4], [5], and [10]. DNA binary strand algorithm [6] is not space efficient to store digital information as it uses a lot of space (nucleotide bases) just to represent binary '0' or '1', the

same issue we find in Complementary pair method. DNA Crypt algorithm has a good handling for errors but its data capacity is also low, considering coding regions as data location is only small part of DNA strand.

In [7], error correction for embedded message is introduced together with fuzzy controller to determine whether WDH code, Hamming code, or no correction code is required. Proposed method also uses fuzzy logic but only to determine parameters of Reed-Solomon code to use. HyDEn method uses DNA Hamming codes to represent each ASCII characters, with the disadvantage of huge space it consumes. Error correction in [9] is able to handle *indel* mutation using repetition code with the cost of data capacity. Error handling in [5] is to send secret information repeatedly until microdot received is not damaged or altered. This method is exhaustive and very time consuming.

Method to preserve host DNA's life information is only given by [7], [9], and our proposed method. DNA crypt algorithm uses silent mutation principle in coding regions while our method preserves coding regions completely. Substitution method has considerably high *bpn* value but to decode secret data it requires original host DNA strand as reference. Data is retrieved by comparing these two strands and analyze the difference. To mention that this method is implemented in coding and noncoding regions, that means amino acid information is damaged. Our method doesn't require host DNA to decode secret data, even with higher *bpn* value.

## 7. CONCLUSION

DNA is a robust yet fragile medium for data storage. It is potent to record huge amount of digital information but there are threats to its biological attribute which is fatal to the information it contains. A new steganographic-based embedding method is proposed, with higher capability than its previous ones. Data is secured by cryptographic technique and then embedded to noncoding DNA regions following sector-wise encoding rules. Data is also enforced by Reed-Solomon code and parity check letters to handle errors and mutations. A fuzzy controller is used to give the most optimal RS code parameters for high efficiency. This method gives comparably higher *bpn* value without disturbing host organism's protein synthesis process. From information technology point of view, it has fulfilled its requirements as a potential data storage. While in the future, we believe that if combined with sufficient bio-molecular technology, it is not impossible to do *in vivo* implementations.

## 8. ACKNOWLEDGMENTS

This work was supported under the framework of international cooperation program managed by National Research Foundation of Korea (NRF-2012K2A1A2032979).

## 9. REFERENCES

[1] Brown, T.A. 2007. *Genomes 3*. Garland Science Publishing, 3rd edition, New York.

[2] Wong, P.C., Wong, K.K., and Foote, H. 2003. Organic data memory using the DNA approach. Commun. ACM 46, 1 (January 2003), 95-98. DOI=10.1145/602421.602426 http://doi.acm.org/10.1145/602421.602426

[3] Artz, D. 2001. Digital steganography: Hiding data within data. *Internet Computing, IEEE*, vol.5, 75-80. DOI= 10.1109/4236.935180

[4] Clelland, C., Risca, V., and Bancroft, C. 1999. Hiding messages in DNA microdots. *Nature* 399, 533-534. DOI=10.1038/21092

[5] Wang, Z., Zhao, X., Wang, H., and Cui, G. 2013. Information hiding based on DNA steganography. In *4th IEEE International Conference on Software Engineering and Service Science* (Beijing, China, May 23-25, 2013). DOI=10.1109/ICSESS.2013.6615462

[6] Leier, A., Richter, C., Banzhaf, W., and Rauhe, H. 2000. Cryptography with DNA binary strands. *BioSystems*, vol. 57, issue 1, 13-22. DOI= http://dx.doi.org/10.1016/j.bbr.2011.03.031

[7] Heider, D. and Barnekow, A. 2007. DNA-based watermarks using the DNA-Crypt algorithm. In *BMC Bioinformatics* 8:176. DOI= 10.1186/1471-2105-8-176

[8] Shiu, H.J., Ng, K.L., Fang, J.F., Lee, R.C.T., and Huang, C.H. 2010. Data hiding methods based upon DNA sequences. In *Information Sciences*, vol. 180, 2196–2208. DOI= http://dx.doi.org/10.1016/j.bbr.2011.03.031

[9] Haughton, D. and Balado, F. 2011. Repetition coding as an effective error correction code for information encoded in DNA. In *11th IEEE International Conference on Bioinformatics and Bioengineering* (Taichung, Taiwan, October 24-26, 2011). DOI= 10.1109/BIBE.2011.45

[10] Tulpan, D., Regoui, C., Durand, G., Belliveau, L., and Leger, S. 2013. HyDEn: A hybrid steganocryptographic approach for data encryption using randomized error-correcting DNA codes, BioMed Research International, vol. 2013, Article ID 634832, 11 pages, 2013. DOI= 10.1155/2013/634832.

[11] Morkel, T., Eloff, J.H.P., and Oliver, M.S. 2005. An overview of image steganography. In *Proceedings of the Fifth Annual Information Security South Africa Conference* (Sandton, South Africa, 2005).

[12] Cummins, J., Diskin, P., Lau, S. and Parlett, R. 2004. Steganography and digital watermarking. School of Computer Science, The University of Birmingham.

[13] Haughton, D. and Balado, F. 2013. A modified watermark synchronisation code for robust embedding of data in DNA. In *2013 IEEE International Conference on Acoustics, Speech, and Signal Processing* (Vancouver, BC, May 26-31, 2013). DOI= 10.1109/ICASSP.2013.6637830

[14] Menezes, A., Oorschot, P., and Vanstone, S. 1996. Handbook of applied cryptography. CRC Press, pp. 169-190.

[15] Sidorenko, A. and Schoenmakers, B. 2005. Concrete security of the Blum-Blum-Shubpseudorandom generator. In *Cryptography and Coding: 10th IMA International Conference* (Cirencester, UK, December 19-21, 2005). DOI= 10.1007/11586821_24

[16] Junod, P. 2013. Cryptographic secure pseudo-random bits generation : The Blum-Blum-Shub generator.

[17] Wikipedia. Genome. http://en.wikipedia.org/wiki/Genome, accessed on December 2013.

[18] Zagordi, O., Klein, R., Daumer, M., and Beerenwinkel, N. 2010. Error correction of next-generation sequencing data

and reliable estimation of HIV quasispecies. In *Nucleic Acids Research*, Vol. 38:21, 7400-7409. DOI= 10.1093/nar/gkq655

[19] Nachman, M.W. and Crowell, S.L. 2000. Estimate of the mutation rate per nucleotide in humans. *Genetics*, 156(1), 297-304.

[20] Tikole, S. and Sankararamakrishnan, R. 2006. A survey of mRNA sequences with a non-AUG start codon in RefSeq database, Journal of Biomolecular Structure and Dynamics, 24:1, 33-41, DOI= 10.1080/07391102.2006.10507096.

[21] NonAUG @ IIT Kanpur. Database of mRNA sequences with non-AUG start codons. http://bioinfo.iitk.ac.in/bioinfo/organism.html. Accessed on February 2014.

[22] NCBI. National Center for Biotechnology Information. www.ncbi.nlm.nih.gov, accessed on October 2013.