Arabian Journal for Science and Engineering
https://doi.org/10.1007/s13369-018-3520-8

RESEARCH ARTICLE - COMPUTER ENGINEERING AND COMPUTER SCIENCE

DNA-Based AES with Silent Mutations

Hatem M. Bahig[1] · Dieaa I. Nassr[1]

Received: 25 January 2018 / Accepted: 13 August 2018

Abstract
We present a new version of Advanced Encryption Standard (AES), called DNAES, based on deoxyribonucleic acid (DNA) sequences with silent mutations. We present how to encode and decode data in a DNA sequence and how to perform the different steps of AES. The proposed cipher has the following features: (1) it can be applied to any type of data (e.g., text, videos, images); (2) it has the same security level as AES; (3) it can be implemented in a biological environment or on DNA computers; (4) because the ciphertext generated by DNAES does not actually change the amino acid sequence of the protein, side effects are avoided; and (5) besides encryption, the proposed cipher can be used to hide data in a DNA sequence.

Keywords Advanced Encryption Standard (AES) · DNA sequence · Silent mutation · Block ciphers

1 Introduction

Biologists and mathematicians have developed a new data storage technique that depends on deoxyribonucleic acid (DNA) [1,2]. Based on the advance of discovering various properties of DNA sequences, several methods are presented to store data in DNA sequence in which one gram of DNA can be used to store about 10^6 TB of data, sufficient to store all of our world's data. A new type of data processing known as DNA computing [3] has been created. Adelman [4] showed that, by biochemical operations of DNA, molecules could be used to perform computations and to solve the Hamiltonian path problem. The computations are performed in efficient parallel operations. Lipton [5] presented an encoding schema, using operations of DNA molecules, to solve the satisfiability problem with a small number of variables. A generalization of Lipton's schema has also been presented in [6]. Boneh et al. [7] presented a molecular program, using the concept of DNA computations, to break the data encryption standard (DES). The power of DNA computation relies not on DNA molecules but also on writing and hiding data in a private position in a strand of DNA to protect them from unauthorized persons for a long time [8–18]. In the literature, encoding data into a DNA sequence have been categorized in two ways [19,20]:

1. the data are converted into a binary string. Then, the binary string is transformed into a DNA sequence. For example, the partial binary strings "00," "01," "10," and "11" are transformed into the nucleotides A, C, G, and T, respectively. See [21–26] for some details.
2. the alphabets of the data are converted into a m alphabet DNA sequence using a predefined encoding table. See [27–29] for some details.

The transformation of data into a chain of artificial nucleotides to form a new DNA strand of a living organism may involve reproduction and prejudicial deterioration of the behavior of this species. To solve this problem, one may use silent mutations [30–34] in the DNA, which do not damage living cells.

A mutation is a modification of a DNA sequence caused either by an error that occurred when the DNA was copied or by chemical damage. Genomic regions, generally called genes, provide guidelines for the creation of protein molecules, which carry out most of the important work in cells. Some DNA mutations (called silent mutation) do not affect the protein production but some do.

Silent mutations are mutations in which the amino acid sequence is not changed, and therefore, there is no change in protein production. Silent mutations produce an alteration

✉ Hatem M. Bahig
hmbahig@sci.asu.edu.eg; h.m.bahig@gmail.com

Dieaa I. Nassr
diaa.rsa@gmail.com

[1] Computer Science Division, Department of Mathematics, Faculty of Science, Ain Shams University, Cairo, Egypt

Published online: 05 September 2018

Table 1 Notations to be used in this paper

Notation	Meaning
AES	Advanced Encryption Standard
A	The nitrogenous base (Adenine)
C	The nitrogenous base (Cytosine)
G	The nitrogenous base (Guanine)
T	The nitrogenous base (Thymine)
GF(2^8)	Galois Field of order 2^8
$a \oplus b$	Bitwise XOR operation between two binary strings
$\oplus_{\text{DNA}}$	DNAES-XOR operation (see Definition 6)
$(e_{n-1} \dots e_1 e_0)_2$	A binary string
DNAES	AES based on DNA sequences with silent mutations

of one of the nucleotides in the triplet code that represents a codon.

In this paper, we are interested in designing the Advanced Encryption Standard (AES) based on storing binary data in silent mutations of DNA, which is already located in a living cell. We will explain how to perform the different steps of AES on DNA strands based on silent mutations. Furthermore, in order to ensure the same security level as AES, we consider the same parameters used to define AES, such as the 16×16 lookup table in the substitution bytes step and the Galois finite field of order 2^8, denoted by GF(2^8), in which the additions and multiplications in the step of the mixing columns are performed.

The new proposed version of AES (called DNAES) has the following properties: (1) it can be applied to any type of data (e.g., text, video, images); (2) it has the same security level as AES; (3) it can be implemented in a biological environment or on DNA computers; (4) the ciphertext generated by DNAES does not actually change the amino acid sequence of the protein that is made, thereby avoiding the danger of using a chain of artificial nucleotides; and (5) besides encryption, the DNAES can be used to hide data in DNA sequences by simply fixing the key.

DNA is not the only nucleic acid that carries genetic information; there is also another nucleic acid called ribonucleic acid (RNA) that is a chain of nucleotides. But unlike DNA, RNA is often observed, in nature, as a single strand folded onto itself rather than as a paired double strand in DNA. In this paper, we focus on DNA but note that our methodology and result can be presented using RNA. Table 1 lists some notations used in this paper.

The paper is organized as follows. In Sect. 2, we give some basic concepts of DNA needed in the paper. A description of AES is presented in Sect. 3. A new version of AES based on silent mutations in DNA is presented in Sect. 4. In Sect. 5, we give an example of encryption using AES and DNAES, and execution times for a simple implementation of the DNAES-cipher. Section 6 includes the conclusion.

2 DNA

This section provides a short background of DNA [1,2,30,35,36]. DNA is very long molecule made up of a long chain of nucleotides, most of them located in the cell nucleus. Every nucleotide contains deoxyribose, a phosphate group and a nitrogenous base (Adenine A, Cytosine C, Guanine G, or Thymine T). The kind of nitrogenous base located in every nucleotide distinguishes one nucleotide from another. Therefore, a long chain (sequence) of nucleotides is written as a sequence of nitrogenous bases with the corresponding appearance in the nucleotides. The sequence of nitrogenous bases forms the genetic code of cells.

The genetic code consists of three consecutive nucleotides (written as three letters of corresponding nitrogenous bases) called a codon (e.g., ACT, CAG, TTT). Every codon of DNA, except for TAA, TAG and TGA, works on the production of one of 20 amino acids. The TAA, TAG and TGA codons signal the end of a sequence of codons. A group of joined amino acids and the order in which they are joined defines a protein. Therefore, a sequence of codons in a DNA molecule may code for some type of protein and therefore form a gene. One gene may contain a thousand or more codons.

The number of possible codons is equal to $4^3 = 64$, and the number of codons that are used to produce amino acids is equal to 61 (i.e., 64 minus the three codons TAA, TAG and TGA). Because the number of possible amino acids is 20, which is smaller than 61 different codons produce the same amino acid. Table 2 lists all 61 codons with their corresponding productions of amino acids.

A mutation is a change in a codon sequence of a gene. That is, the replacement of at least one codon by another different codon in the codon sequence of a gene causes a mutation. A silent mutation occurs when one or more codons is replaced by different codon producing the same amino acid. For example, CAT is replaced by CAC. Note that the silent mutation does not result in a change in the amino acid sequence of a protein.

Table 2 Amino acids with their corresponding codons

Amino acid	Codons
Isoleucine	ATA, ATC, ATT
Leucine	CTA, CTC, CTG, CTT, TTA, TTG
Valine	GTA, GTC, GTG, GTT
Phenylalanine	TTC, TTT
Methionine	ATG
Cysteine	TGC, TGT
Alanine	GCA, GCC, GCG, GCT
Glycine	GGA, GGC, GGG, GGT
Proline	CCA, CCC, CCG, CCT
Threonine	ACA, ACC, ACG, ACT
Serine	AGC, AGT, TCA, TCC, TCG, TCT
Tyrosine	TAC, TAT
Tryptophan	TGG
Glutamine	CAA, CAG
Asparagine	AAC, AAT
Histidine	CAC, CAT
Glutamic acid	GAA, GAG
Aspartic acid	GAC, GAT
Lysine	AAA, AAG
Arginine	AGA, AGG, CGA, CGC, CGG, CGT

3 AES

This section presents a brief description of AES [37,38]. AES is a symmetric block cipher algorithm that was adopted by the National Institute of Standards and Technology (NIST) of the United States Government in 2001. In AES, the length of the plaintext/ciphertext is 128 bits. The encryption process is performed by applying 10 (12 or 14) sequential rounds, all of which are similar except for the last round. In each round, there is a round key of length 128 bits where the round keys are derived from a master key of length 128 (192 or 256) bits by an algorithm known as the *key expansion algorithm* [37,38]. The general structure of AES is shown in Fig. 1. This figure shows the steps that are performed in each round. Each round consists of four steps (substitution bytes, shift rows, mix columns and add round key). The exception is the last round, which does not perform the mix columns as shown in Fig. 1. Note that all the steps are invertibles for the decryption process. Below, we briefly describe each step.

In the substitution bytes step, the input, called *input state*, is an array of 16 bytes (4 × 4 matrix of bytes). This step consists of using a 16 × 16 lookup table, called **S-box**, to find a substitution byte for each byte in the input state array, i.e., byte-by-byte substitution. The input byte is divided into two parts: the first part is the leftmost 4 bits and the second part is the rightmost 4 bits. Each part produces an integer value between 0 and 15, and the first part is used as a row index, while the second part is used as a column index for reaching into the 16 × 16 lookup table.

In the shift rows step, the input state is considered to be a 4 × 4 matrix. The first row is not shifted, but each of the second, third and fourth rows is circularly shifted to the left by one, two and three bytes, respectively. Let the input state be represented by a 4 × 4 matrix S :

$$S = \begin{pmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{pmatrix}. \tag{1}$$

Then, the output of the shift rows step on S is as follows:

$$\begin{pmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,1} & s_{1,2} & s_{1,3} & s_{1,0} \\ s_{2,2} & s_{2,3} & s_{2,0} & s_{2,1} \\ s_{3,3} & s_{3,0} & s_{3,1} & s_{3,2} \end{pmatrix}. \tag{2}$$

As in the shift rows step, the input state of the mix columns step is considered to be a 4 × 4 matrix.

The mix columns step operates on each column individually. Each byte of a column is mapped to a new value that is a function of the four bytes in that column. The mixing columns on the input state S can be defined by the following matrix multiplication, where additions and multiplications are performed in GF(2^8), see [39].

$$\begin{pmatrix} s'_{0,0} & s'_{0,1} & s'_{0,2} & s'_{0,3} \\ s'_{1,0} & s'_{1,1} & s'_{1,2} & s'_{1,3} \\ s'_{2,0} & s'_{2,1} & s'_{2,2} & s'_{2,3} \\ s'_{3,0} & s'_{3,1} & s'_{3,2} & s'_{3,3} \end{pmatrix} = \begin{pmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{pmatrix} \times \begin{pmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{pmatrix}.$$

Let j be an index of a column. Then the output of the mix columns step can be expressed using the bitwise XOR operation ($\oplus$) as follows:

$$\begin{aligned} s'_{0,j} &= 2s_{0,j} \oplus 3s_{1,j} \oplus s_{2,j} \oplus s_{3,j} \\ s'_{1,j} &= s_{0,j} \oplus 2s_{1,j} \oplus 3s_{2,j} \oplus s_{3,j} \\ s'_{2,j} &= s_{0,j} \oplus s_{1,j} \oplus 2s_{2,j} \oplus 3s_{3,j} \\ s'_{3,j} &= 3s_{0,j} \oplus s_{1,j} \oplus s_{2,j} \oplus 2s_{3,j} \end{aligned} \tag{3}$$

Let $\text{temp} = s_{0,j} \oplus s_{1,j} \oplus s_{2,j} \oplus s_{3,j}$. Then Eq. 3 can be rewritten as follows [40]:

$$\begin{aligned} s'_{0,j} &= s_{0,j} \oplus \text{temp} \oplus 2(s_{0,j} \oplus s_{1,j}) \\ s'_{1,j} &= s_{1,j} \oplus \text{temp} \oplus 2(s_{1,j} \oplus s_{2,j}) \\ s'_{2,j} &= s_{2,j} \oplus \text{temp} \oplus 2(s_{2,j} \oplus s_{3,j}) \\ s'_{3,j} &= s_{3,j} \oplus \text{temp} \oplus 2(s_{3,j} \oplus s_{0,j}) \end{aligned} \tag{4}$$

Note that multiplying by 2 in GF(2^8) can be accomplished with a 1-bit left shift followed by a conditional bitwise XOR

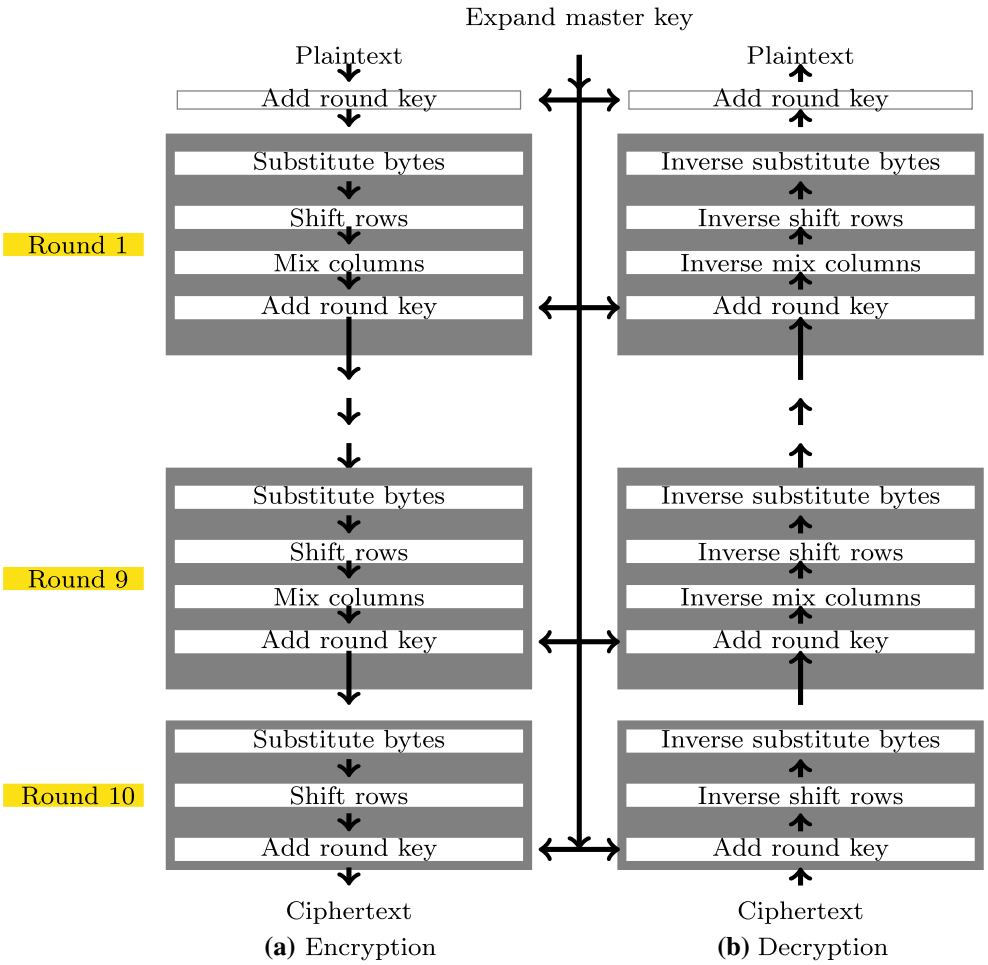

Fig. 1 General structure of AES

with $(00011011)_2$. That is, for $e = (e_7 e_6 \dots e_0)_2 \in \mathrm{GF}(2^8)$, multiplying by 2 is given by

$$2e = \begin{cases} (e_6 e_5 \dots e_0 0)_2, & \text{if } e_7 = 0; \\ (e_6 e_5 \dots e_0 0)_2 \oplus (00011011)_2, & \text{if } e_7 = 1. \end{cases} \quad (5)$$

In [40], the inverse of the mix columns step can be expressed as:

$$\begin{aligned} \text{term} &= 9(s_{0,j} \oplus s_{1,j} \oplus s_{2,j} \oplus s_{3,j}) \\ s'_{0,j} &= s_{0,j} \oplus \text{term} \oplus 2(2(s_{0,j} \oplus s_{2,j})) \oplus 2(s_{0,j} \oplus s_{1,j}) \\ s'_{1,j} &= s_{1,j} \oplus \text{term} \oplus 2(2(s_{1,j} \oplus s_{3,j})) \oplus 2(s_{1,j} \oplus s_{2,j}) \\ s'_{2,j} &= s_{2,j} \oplus \text{term} \oplus 2(2(s_{0,j} \oplus s_{2,j})) \oplus 2(s_{2,j} \oplus s_{3,j}) \\ s'_{3,j} &= s_{3,j} \oplus \text{term} \oplus 2(2(s_{1,j} \oplus s_{3,j})) \oplus 2(s_{3,j} \oplus s_{0,j}) \end{aligned} \quad (6)$$

Note that the multiplication by 9 in $\mathrm{GF}(2^8)$ can be implemented as 3 times multiplication by 2 in $\mathrm{GF}(2^8)$ followed by one bitwise XOR operation. That is, for $a \in \mathrm{GF}(2^8)$,

$$9a = 2(2(2a)) \oplus a.$$

Figure 2 shows two steps: shift rows and mix columns. In the add round key step, the round key with a length of 128 bits is added to the output of the previous step during the forward process. This addition is a simple bitwise XOR operation.

4 DNAES-Cipher

In this section, we propose a new version of AES, called DNAES-cipher, using silent mutations in DNA. Throughout the structure processing of DNAES-cipher, we presume that the data are stored in a DNA strand using silent mutations. The proposed cipher DNAES mimics AES, see Fig. 3. This section presents (1) new definitions necessary for the construction of the DNAES-cipher; (2) two algorithms to encode and decode data from/to DNA strands and binary strings; (3) all of the steps of AES using a DNA model with silent mutations.

In the following, the letters x, y and z (also x', y' or z') stand for an arbitrary choice of a nitrogenous base (A, C, G, or T). Accordingly, we write an arbitrary choice of codon as

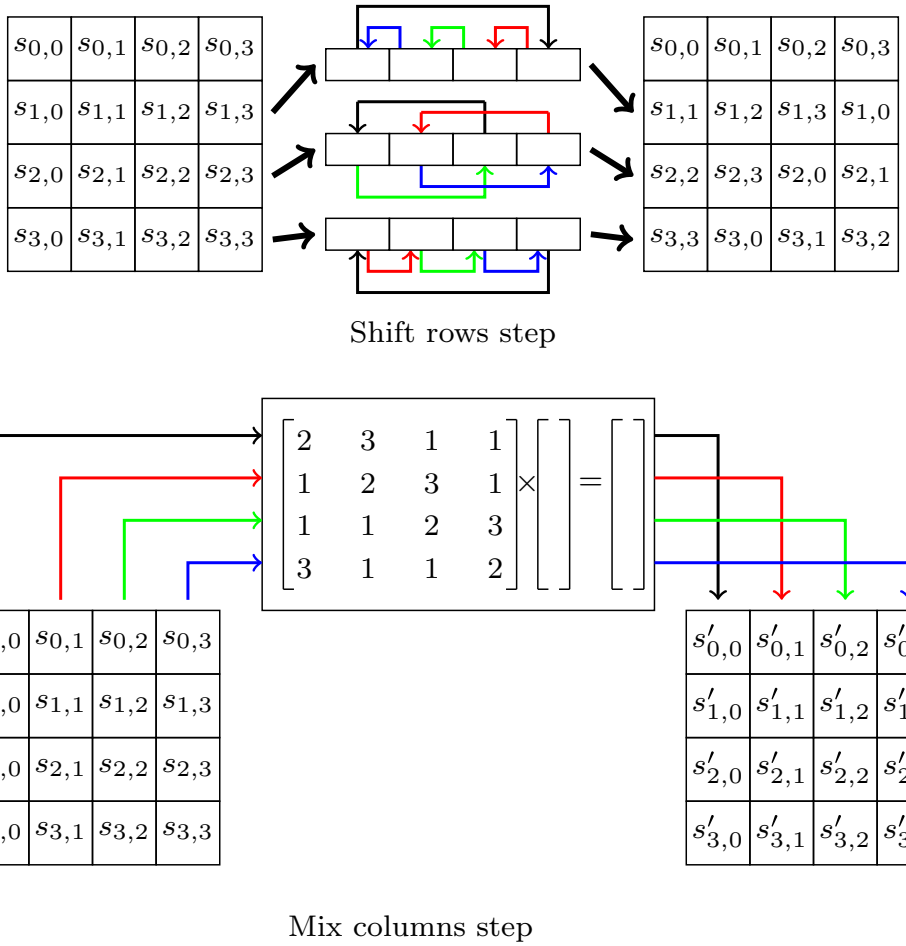

Fig. 2 The steps of shift rows and mix columns

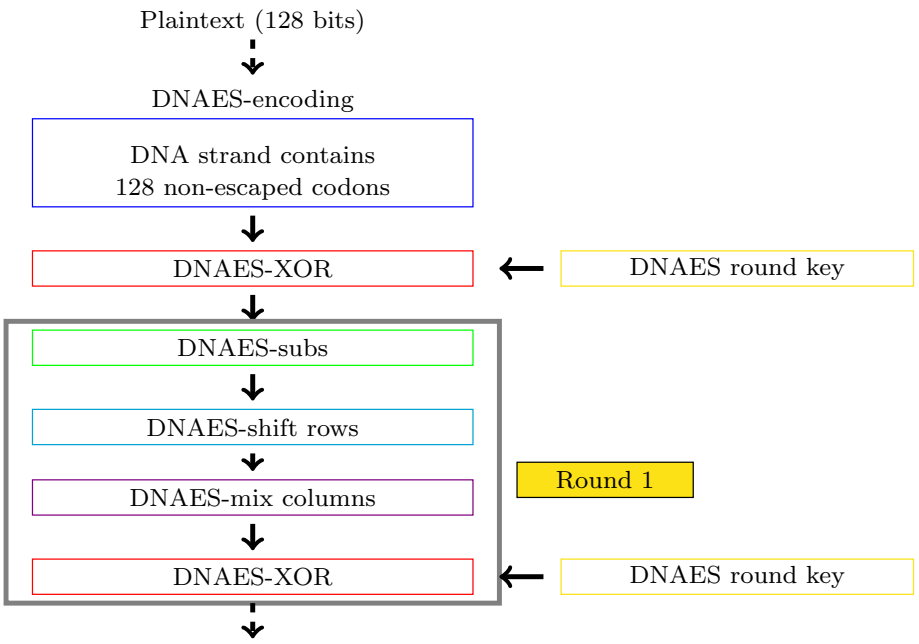

Fig. 3 General structure of DNAES-cipher

xyz (or $x'y'z'$). In addition, we sometimes write two arbitrary choices of codons that have the same two nitrogenous bases at the beginning as xyz and xyz'.

A substitution of a codon with another one may lead to a non-silent mutation. Therefore, it is natural to substitute/encode a codon with another one that produces the same amino acid to preserve on silent mutation. Table 2 lists all of the possible codons and their corresponding amino acids. In Table 3, for each amino acid, we classify the codons according to their first two nitrogenous bases. That is, codons that start with the nitrogenous base x and are followed by the nitrogenous base y and produce the same amino acid l are grouped in an array (list) $S_{xy}(l)$. This array is sorted alphabetically. Therefore, we define the following:

Definition 1 Let l be an amino acid and x, y be two arbitrary nitrogenous bases. Then, $S_{xy}(l)$ is defined to be an alphabetically sorted array (or list) of codons that start with the nitrogenous base x followed by y and produce the amino acid l.

In other words, $S_{xy}(l)$ contains all codons that (1) produce the amino acid l; (2) start with xy; (3) are alphabetically sorted. For example, suppose that the symbol I refers to the amino acid Isoleucine. Therefore, according to Table 2, $S_{\mathrm{AT}}(I) = \{\mathrm{ATA}, \mathrm{ATC}, \mathrm{ATT}\}$.

Since $S_{xy}(l)$ is a sorted array, the codons in $S_{xy}(l)$ are numbered starting with 0. If the codon $xyz \in S_{xy}(l)$, then $|xyz|$ defines the position (or index) of xyz in $S_{xy}(l)$. For examples, $|\mathrm{ATA}| = 0$, and $|\mathrm{ATT}| = 2$ in $S_{\mathrm{AT}}(I)$.

Therefore, throughout this paper, the arithmetic operations performed on codon xyz are performed with respect to its index $|xyz|$.

Since a bit is 0 or 1, a list $S_{xy}(l)$ with at least two codons can be used to make a substitution. In other words, a codon that has no alternative codon to produce the same amino acid (i.e., the number of elements in $S_{xy}(l)$ is one) cannot be used to make a silent mutation. Therefore, we exclude these codons, which we call "escaped codons."

Definition 2 *(Escaped Codon)* The array $S_{xy}(l)$ that contains only one codon is called an *escaped array,* and its codon is called an *escaped codon.*

The codon that is not an escaped codon is called a *non-escaped codon.* For example, ATG and AGG are escaped codons, and ATA is a non-escaped codon.

In AES, the state is an array of length 128 bits (i.e., 16 bytes or 4×4 matrix of bytes). In DNAES, a DNA state is a DNA strand with 128 non-escaped codons (i.e., with $S_{i,j}$, $i, j = 0, 1, 2, 3$, where each $S_{i,j}$ is 8 non-escaped codons.

Table 3 Codons distributed in sorted lists according to Definition 1

Amino acid	Sorted list				
Isoleucine	S_{AT}	ATA	ATC	ATT	
		0	1	2	
Leucine	S_{CT}	CTA	CTC	CTG	CTT
		0	1	2	3
	S_{TT}	TTA	TTG		
		0	1		
Valine	S_{CT}	GTA	GTC	GTG	GTT
		0	1	2	3
Phenylalanine	S_{TT}	TTC	TTT		
		0	1		
Methionine	S_{AT}	ATG			
		0			
Cysteine	S_{TG}	TGC	TGT		
		0	1		
Alanine	S_{GC}	GCA	GCC	GCG	GCT
		0	1	2	3
Glycine	S_{GG}	GGA	GGC	GGG	GGT
		0	1	2	3
Proline	S_{CC}	CCA	CCC	CCG	CCT
		0	1	2	3
Threonine	S_{AC}	ACA	ACC	ACG	ACT
		0	1	2	3
Serine	S_{AG}	AGC	AGT		
		0	1		
	S_{TC}	TCA	TCC	TCG	TCT
		0	1	2	3
Tyrosine	S_{TA}	TAC	TAT		
		0	1		
Tryptophan	S_{TG}	TGG			
		0			
Glutamine	S_{CA}	CAA	CAG		
		0	1		
Asparagine	S_{AA}	AAC	AAT		
		0	1		
Histidine	S_{CA}	CAC	CAT		
		0	1		
Glutamic acid	S_{GA}	GAA	GAG		
		0	1		
Aspartic acid	S_{GA}	GAC	GAT		
		0	1		
Lysine	S_{AA}	AAA	AAG		
		0	1		
Arginine	S_{AG}	AGA	AGG		
		0	1		
	S_{CG}	CGA	CGC	CGG	CGT
		0	1	2	3

4.1 DNAES-coding

In this section, we show how to code binary data into DNA without changing the production of the amino acid sequence of a protein, i.e., preserving silent mutations.

We need to define a map, called a silent map, that transforms a non-escaped codon xyz to another non-escaped codon xyz' such that both codons produce the same amino acid. Because there may be more than one xyz', we define the silent map as a function of xyz and the position of xyz' in $S_{xy}(l)$.

Definition 3 *(Silent Map)* Let $xyz \in S_{xy}(l)$ be a non-escaped codon and i be an integer. The silent map ST is defined by $\mathrm{ST}(xyz, i) = xyz'$, where $xyz' \in S_{xy}(l)$ and $|xyz'| = |xyz| + i \pmod 2$.

Because some $S_{xy}(l)$ contain more than two codons, the encoding of binary data into a DNA strand using only ST may lead to a decoding problem. Therefore, we encode the non-escaped codon xyz to the first codon in $S_{xy}(l)$ before applying ST. The function that encodes the codon to the first codon, index 0, in $S_{xy}(l)$ is called a *zero silent map*.

Definition 4 *(Zero Silent Map)* Let $xyz \in S_{xy}(l)$ be a non-escaped codon. The zero silent map ZST is defined by $\mathrm{ZST}(xyz) = xyz'$, where $xyz' \in S_{xy}(l)$ and $|xyz'| = 0$.

Note that, if $\mathrm{ST}(xyz, i) = xyz'$ (or $\mathrm{ZST}(xyz) = xyz'$), then the two codons xyz and xyz' produce the same amino acid, i.e., the result of applying ST (and ZST) is a silent mutation.

In Algorithms 1 and 2, we show how to use the two maps ST and ZST on a known part in DNA to encode and decode binary data according to the concept of silent mutations.

Algorithm 1 ***DNAES-encoding***
Input: *$e = (e_{m-1}e_{m-2}\dots e_0)_2$ is a binary data and S is a known part in DNA with m non-escaped codons.*
Output: *S after storing e in S as silent mutations.*
Begin
Let the m non-escaped codons in S be $x_i y_i z_i$, $i = 0, 1, \dots, m-1$.
for *i=0 to m-1* ***do***
  $x_i y_i z_i' = ZST(x_i y_i z_i)$, *i.e.,* $|x_i y_i z_i'| = 0$.
  $x_i y_i z_i = ST(x_i y_i z_i', e_i)$.
end for
The m non-escaped codons in S are exposed to silent mutations.
End.

To illustrate the steps of Algorithm 1, we provide the following example of encoding binary data into S, a known part in DNA.

Example 1 Suppose that the binary data $e = (110101)_2$ and that the known part of DNA is S = TCC-TCG-TGG-CAA-GCA-ATC-CAG. Furthermore, suppose that e and S are inputs of Algorithm 1. Note that the six non-escaped codons in S are TCC-TCG-CAA-GCA-ATC-CAG. By applying the map ZST on the six codons, recover TCA-TCA-CAA-GCA-ATA-CAA, respectively. We have the following:

1. since $e_0 = 1$, the codon TCA is transformed to TCC.
2. since $e_1 = 0$, the codon TCA is transformed to TCA.
3. since $e_2 = 1$, the codon CAA is transformed to CAG.
4. since $e_3 = 0$, the codon GCA is transformed to GCA.
5. since $e_4 = 1$, the codon ATA is transformed to ATC.
6. since $e_5 = 1$, the codon CAA is transformed to CAG.

Therefore, encoding $e = (110101)_2$ into S = TCC-TCG-TGG-CAA-GCA-ATC-CAG yields the silent mutations TCC-TCA-TGG-CAG-GCA-ATC-CAG. Note that the codon TGG is not taken into consideration during the encoding because TGG is an escaped codon according to Definition 2.

Algorithm 2 ***DNAES-decoding***
Input: *S is a known part in DNA that stores $e = (e_{m-1}e_{m-2}\dots e_0)_2$ as silent mutations.*
Output: *$e = (e_{m-1}e_{m-2}\dots e_0)_2$.*
Begin
Let the non-escaped codons in S be $x_i y_i z_i$, $i = 0, 1, \dots, m-1$.
for *i=0 to m-1* ***do***
  $e_i = |x_i y_i z_i| \pmod 2$
end for
return *e.*
End

To illustrate the steps of Algorithm 2, we give the following example.

Example 2 Suppose that the output S = TCC-TCA-TGG-CAG-GCA-ATC-CAG of the encoding example is an input of Algorithm 2. The non-escaped codons in S are TCC-TCA-CAG-GCA-ATC-CAG. Using Table 3 and Algorithm 2, the index of

(1) TCC is 1 and $e_0 = 1$. (2) TCA is 0 and $e_1 = 0$.
(3) CAG is 1 and $e_2 = 1$. (4) GCA is 0 and $e_3 = 0$.
(5) ATC is 1 and $e_4 = 1$. (6) CAG is 1 and $e_5 = 1$.

Therefore, the output is $e = (110101)_2$.

4.2 DNAES-XOR

The purpose of defining the DNAES-XOR operation is to imitate the bitwise XOR operation $\oplus$ in AES.

In AES, the bitwise XOR operation is performed between two binary strings of the same length. In DNAES, we perform the XOR operation, termed DNAES-XOR, between two strands of DNA. We treat the first strand of DNA as a sequence of codons, and we treat the second strand of DNA as a sequence of nucleotides. The number of codons in the first strand of DNA is equal to the number of nucleotides in the second strand.

Before performing the DNAES-XOR operation, the second DNA strand is encoded into a sequence of digits ($0 = A, 1 = C, 2 = G, 3 = T$) that encode the nitrogenous bases. Then, we perform the DNAES-XOR between the first DNA strand and the sequence of digits using the silent mutation map ST. We accordingly need to define two terms: the digit sequence of encoded nitrogenous bases and DNAES-XOR.

Definition 5 *(Key Sequence)* Let $S = x_0, x_1, \ldots, x_n$ be a sequence of nucleotides (DNA strand). The corresponding digit sequence of S is given by $k_0, k_1, \ldots, k_n$ where

$$k_i = \begin{cases} 0, & \text{if } x_i = A; \\ 1, & \text{if } x_i = C; \\ 2, & \text{if } x_i = G; \\ 3, & \text{if } x_i = T. \end{cases}$$

Definition 6 *(DNAES-XOR)* Let $S_1 = x_0y_0z_0 - x_1y_1z_1 - \ldots - x_ny_nz_n$ be a sequence of codons and S_2 be a sequence of nucleotides with digit sequence $k_0, k_1, \ldots k_n$. Then the DNAES-XOR operation ($\oplus_{\text{DNA}}$) is defined by

$$S_1 \oplus_{\text{DNA}} S_2 = x_0y_0z_0' - x_1y_1z_1' - \ldots - x_ny_nz_n',$$

where $x_iy_iz_i' = ST(x_iy_iz_i, k_i)$, $i = 0, 1, \ldots, n$.

Note that, given two binary data e and e' of the same length, the bitwise XOR operation $\oplus$ satisfies the following relation:

$$(e \oplus e') \oplus e' = e.$$

For the DNAES-XOR operation $\oplus_{\text{DNA}}$, we have

$$(S \oplus_{\text{DNA}} S') \oplus_{\text{DNA}} S' = S'' \tag{7}$$

where S' is a digit sequence corresponding to a sequence of nucleotides and S and S'' are sequences of codons satisfying

$$\texttt{DNAES-decoding}(S) = \texttt{DNAES-decoding}(S''),$$

i.e., S and S'' encode the same binary data.

To prove Eq. 7, let i be the index of a codon in S and j be the corresponding digit that encodes the nucleotide in S'. We have

$$((i+j) \pmod 2) + j \pmod 2 = i + j + j \pmod 2 = i + 2j \pmod 2 = i.$$

This proves $\texttt{DNAES-decoding}(S) = \texttt{DNAES-decoding}(S'')$, and so the correctness of Eq. 7.

Thus, Eq. 7 is used in the decryption process.

In DNAES-cipher, the DNAES-XOR operation $\oplus_{\text{DNA}}$ is used in two steps: the DNAES-mix columns (Sect. 4.5) and add DNA round key (Sect. 4.6).

4.3 DNAES-Substitution Bytes

We show how to define the step DNAES-subs to imitate the substitution bytes step in AES. We give Algorithm 3 to perform DNAES-subs on a DNA state with 16 $S_{i,j}$ $(i, j = 0, 1, 2, 3)$, where each $S_{i,j}$ is 8 non-escaped codons.

Algorithm 3 ***DNAES-subs***
Input: *a DNA state* $\bar{S}$:

$$S_{0,0}, S_{1,0}, S_{2,0}, S_{3,0}, S_{0,1}, S_{1,1}, S_{2,1}, S_{3,1}, \ldots, S_{3,3},$$

where each $S_{i,j}$ *is* 8 *non-escaped codons.*
Output: *perform the substitution on* $\bar{S}$
Begin
1: ***for*** *i=0 to 3* ***do***
2: ***for*** *j=0 to 3* ***do***
3: $d_{i,j} = DNAES\text{-}decoding(S_{i,j})$
4: $DNAES\text{-}encoding(S\text{-}box(d_{i,j}), S_{i,j})$
5: ***end for***
6: ***end for***
End

The inverse of the DNAES-subs step is similar to Algorithm 3 but we use the inverse of **S-box**, denoted by **IS-box**, instead of **S-box**. Note that the **IS-box** is the **S-box** run in reverse.

4.4 DNAES-Shift Rows

Now, we present the DNAES-shift rows step to imitate the shift rows step in AES. In AES, the shift rows step requires a 4×4 matrix of bytes and returns it after swapping some elements/bytes according to Eq. 2. In DNAES, the DNAES-shift rows operation takes a DNA state $S_{0,0}, S_{0,1}, \cdots, S_{3,3}$, where each $S_{i,j}$ is 8 non-escaped codons, and returns it after making a swap between some $S_{i,j}$ according to Eq. 2. Therefore, we need first to show how to imitate the swap between two $S_{i,j}$. This step can be accomplished using Algorithm 4. Next, we use Algorithm 4 to define the DNA-shift rows step as in Algorithm 5.

Algorithm 4 ***DNAES-swap***
Input: *S_0 and S_1 are two DNA strands and each one is 8 non-escaped codons with $d_0 =$ DNAES-decoding(S_0) and $d_1 =$ DNAES-decoding(S_1).*
Output: *Changing in S_0 and S_1 such that $d_1 =$ DNAES-decoding(S_0) and $d_0 =$ DNAES-decoding(S_1).*
Begin
1: ***for*** *i=0 to 1* ***do***
2: $d_i =$ DNAES-decoding(S_i)
3: ***end for***
4: DNAES-encoding(d_1, S_0)
5: DNAES-encoding(d_0, S_1)
End

Algorithm 5 ***DNAES-shift rows***
Input: *a DNA state $\bar{S}$:*

$$S_{0,0}, S_{1,0}, S_{2,0}, S_{3,0}, S_{0,1}, S_{1,1}, S_{2,1}, S_{3,1}, \ldots, S_{3,3},$$

where each $S_{i,j}$ is 8 non-escaped codons.
Output: *perform DNAES-shift rows on $\bar{S}$.*

1: DNAES-swap($S_{1,0}$, $S_{1,1}$) ▷ *shifting second row:*
2: DNAES-swap($S_{1,1}$, $S_{1,2}$)
3: DNAES-swap($S_{1,2}$, $S_{1,3}$)
4: DNAES-swap($S_{2,0}$, $S_{2,2}$) ▷ *shifting third row:*
5: DNAES-swap($S_{2,1}$, $S_{2,3}$)
6: DNAES-swap($S_{3,0}$, $S_{3,1}$) ▷ *shifting fourth row:*
7: DNAES-swap($S_{3,2}$, $S_{3,3}$)
8: DNAES-swap($S_{3,0}$, $S_{3,2}$)
End

Because all of the steps in Algorithm 5 are invertible the inverse of DNAES-shift rows that is used in DNAES-decryption is similar.

4.5 DNAES-Mix Columns

The DNAES-mix columns step is imitating the mix columns step in AES. In AES, throughout the mix columns step, the value of some bytes is multiplied by 2 in GF(2^8) (see Eq. 5). Algorithm 6 imitates the multiplication by 2 in GF(2^8). Then, we use Algorithm 7 to perform the DNAES-mix columns step.

Algorithm 6 ***(DNAES-MultBy2)*** *DNAES-multiplication by 2 in* GF(2^8)
Input: *S is a chain of 8 nucleotides with the digit sequence $(e_7e_6e_5e_4e_3e_2e_1e_0)_2$.*
Output: *$S' = 2S$.*
Begin
1: ***if*** *$e_7 = 0$* ***then***
2: *create a chain of nucleotides S' with the digit sequence* $(e_6e_5e_4e_3e_2e_1e_00)_2$
3: ***else***
4: *create a chain of nucleotides S' with the digit sequence* $(e_6e_5e_4e_3e_2e_1e_00)_2 \oplus (00011011)_2$
5: ***end if***
End

Algorithm 7 ***DNAES-mix columns***
Input: *a DNA state $\bar{S}$:*

$$S_{0,0}, S_{1,0}, S_{2,0}, S_{3,0}, S_{0,1}, S_{1,1}, S_{2,1}, S_{3,1}, \ldots, S_{3,3},$$

where each $S_{i,j}$ is 8 non-escaped codons and $d_{i,j} =$ DNAES-decoding($S_{i,j}$).
Output: *perform DNAES-mix columns on $\bar{S}$.*
Begin
1: ***for*** *j=0 to 3* ***do***
2: *U_0 is a chain of nucleotides with the digit sequence* $(d_{0,j} \oplus d_{1,j})$
3: *U_1 is a chain of nucleotides with the digit sequence* $(d_{1,j} \oplus d_{2,j})$
4: *U_2 is a chain of nucleotides with the digit sequence* $(d_{2,j} \oplus d_{3,j})$
5: *U_3 is a chain of nucleotides with the digit sequence* $(d_{3,j} \oplus d_{0,j})$
6: $V_0 =$ DNAES-MultBy2(U_0)
7: $V_1 =$ DNAES-MultBy2(U_1)
8: $V_2 =$ DNAES-MultBy2(U_2)
9: $V_3 =$ DNAES-MultBy2(U_3)
10: *$term$ is a chain of nucleotides with the digit sequence* $(d_{0,j} \oplus d_{1,j} \oplus d_{2,j} \oplus d_{3,j})$
11: $(S_{0,j} \oplus_{DNA} term) \oplus_{DNA} V_0$
12: $(S_{1,j} \oplus_{DNA} term) \oplus_{DNA} V_1$
13: $(S_{2,j} \oplus_{DNA} term) \oplus_{DNA} V_2$
14: $(S_{3,j} \oplus_{DNA} term) \oplus_{DNA} V_3$
15: ***end for***
End

Using Eq. 6, the inverse of DNAES-mix column is similar to Algorithm 7.

4.6 DNA Round Keys Construction

In AES, there is a so-called master key of length 128, 192 or 256 bits. Round keys are derived from this master key using the key expansion algorithm [37,38]. The number of round keys is 10, 12 or 14 based on the length of the master key.

In the DNAES-cipher, we suggest using one of the following two methods to construct the DNA round keys:

1. Let f_i and g_i be selected at random from the sets $\{A, G\}$ and $\{C, T\}$, respectively, i.e., $f_i = A$ or $f_i = G$ with equal probabilities (similarly, $g_i = C$ or $g_i = T$ with equal probabilities), $i = 0, 1, \ldots 127$. Let $r =$

$(e_{127} \ldots e_1 e_0)_2$ be a round key in AES. The function $L(r)$ is defined as $L(r) = x_{127} \ldots x_1 x_0$, where

$$x_i = \begin{cases} f_i, & \text{if } e_i = 0\,; \\ g_i, & \text{if } e_i = 1 \end{cases}$$

Therefore, if the round keys $r_0, r_1, \ldots r_{n-1}$, $(n = 10, 12 \text{ or } 14)$ are created using the key expansion algorithm [37,38] for some given master key, then these round keys can be used to create DNA round keys $L(r_0), L(r_1), \ldots, L(r_{n-1})$.

2. If we consider that the sequential appearance of nitrogenous bases in a nucleotide chain of DNA strand is random, the 10, 12 or 14 round keys are selected as being certain known (specified) sequences of nitrogenous bases each with a length of 128 nitrogenous bases. Furthermore, the master key can be suggested to be the positions of these sequences of round keys in a DNA strand.

5 Implementation

This section presents an example of encrypting a message using AES and DNAES and execution times for a simple implementation of DNAES.

5.1 Example

Since all rounds of encryption process are similar except for the last round, we explain the first encryption round.

Suppose that the plaintext is "Video conference" and the master key is the sequence of 128 nucleotides:

A G T C C C T A T T A G A T C T C C A A T C A G G T A G G T A C T G T T C G T A T T T A C T C G T C C A G C A G G A A T G A A C G T T C T C T A T C A G A T G C G C A G T C A G T T A T G T A C G G T A C C T A T C A G A C C T C A A A T A A G A T A G A T A C

Suppose also that we encode the plaintext in a DNA pattern of human insulin gene using the first 128 non-escaped codons of the following DNA strand:

GGT-GTG-GGG-ACA-GGG-GTG-TGG-GGA-CAG-GGG-TCT-GGG-GAC- AGG-GGT-GTG-GGG-ACA-GGG-GTC-CTG-GGG-ACA-GGG-GTG-TGG- GGA-TAG-GGG-TGT-GGG-GAC-AGG-GGT-GTG-GGG-ACA-GGG-GTG- TGG-GGA-CAG-GGG-TCT-GGG-GAC-AGC-AGC-GCA-AAG-AGC-CCC- GCC-CTG-CAG-CCT-CCA-GCT-CTC-CTG-GTC-TAA-TGT-GGA-AAG- TGG-CCC-AGG-TGA-GGG-CTT-TGC-TCT-CCT-GGA-GAC-ATT-TGC- CCC-CAG-CTG-TGA-GCA-GGG-ACA-GGT-CTG-GCC-ACC-GGG-CCC- CTG-GTT-AAG-ACT-CTA-ATG-ACC-CGC-TGG-TCC-TGA-GGA-AGA- GGT-GCT-GAC-GAC-CAA-GGA-GAT-CTT-CCC-ACA-GAC-CCA-GCA- CCA-GGG-AAA-TGG-TCC-GGA-AAT-TGC-AGC-CTC-AGC-GGT-GTG- GGG-ACA-GGG-GTG-TGG-GGA-CAG-GGG-TCT-GGG-GAC-

Note that the binary data of the plaintext are:

v : 01110110, i : 01101001, d : 01100100,
e : 01100101, o : 01101111, $space$: 00100000,
c : 01100011, o : 01101111, n : 01101110, f : 01100110,
e : 01100101, r : 01110010,
e : 01100101, n : 01101110, c : 01100011, e : 01100101

By applying Algorithm 1, we encode the binary data of the plaintext into the DNA pattern of Human insulin gene as follows:

GGA-GTC-GGC-ACA-GGC-GTC-TGG-GGC-CAA-GGC- TCA-GGA-GAT-AGA-GGC-GTC-GGA-ACA-GGA-GTC-CTA-GGA-ACC- GGC-GTA-TGG-GGC-TAG-GGA-TGT-GGA-GAC-AGG-GGC-GTA-GGC- ACC-GGC-GTC-TGG-GGA-CAG-GGC-TCA-GGA-GAC-AGC-AGC-GCA- AAG-AGC-CCA-GCC-CTC-CAA-CCA-CCA-GCC-CTC-CTA-GTC-TAA- TGT-GGC-AAG-TGG-CCA-AGG-TGA-GGC-CTA-TGC-TCC-CCC-GGC- GAC-ATC-TGT-CCA-CAA-CTC-TGA-GCC-GGA-ACA-GGC-CTC-GCA- ACC-GGA-CCC-CTA-GTA-AAG-ACC-CTA-ATG-ACA-CGC-TGG-TCA- TGA-GGA-AGG-GGC-GCC-GAC-GAT-CAA-GGC-GAC-CTA-CCC-ACC- GAC-CCA-GCC-CCC-GGC-AAA-TGG-TCC-GGC-AAC-TGT-AGT-CTA- AGC-GGA-GTC-GGC-ACA-GGC-GTA-TGG-GGC-CAA-GGA-TCC-GGC- GAC-

In Table 4, we give the first two round keys and the last one for both ciphers: AES and DNAES. The data are in hexadecimal for AES.

In Tables 5 and 6, we show the output of running both ciphers (AES, DNAES) step by step for the first round. Then we show the ciphertext (output of the last round).

5.2 Experimental Results

This section gives a simple (but not optimized and parallelized) implementation of the DNAES-cipher. Table 7 shows that time taken, in seconds, by encoding process and encryption algorithm as in Fig. 3. The table presents the average of the CPU time in seconds to run DNAES for 100 random samples for each data type used (text, videos, images). The implementation was written in the C++ language. Our platform was Pentium IV 3.2 GHz with Linux operating system.

6 Conclusion

We have presented a new version of AES, called DNAES, based on a DNA sequence and silent mutations that have not

Table 4 Round keys according to the key expansion algorithm

AES	DNAES
round key 0	
0x7c 0xe3 0x33 0xa2 0x5d 0x77 0x27 0x88	G G C C T C C A C C G G G T T T T C
0x7e 0xa3 0x32 0xab 0x74 0xe3 0x11 0xa2	G A T C A G A C G A A T G C T A T C
	T G T A T T C G T C T G T C T A A T
	G G G G G C A G A C A C C C C C C G
	C T A A A T G C G T A A C C A A T C
	A T A T G C A A T G T T T A C C G A
	G C T T C A A G T G G A A C G A G T
	G C
round key 1	
0x6c 0x61 0x09 0x30 0x31 0x16 0x2e 0xb8	G A C C G C C A T G G G G T T G T G
0x4f 0xb5 0x1c 0x13 0x3b 0x56 0x0d 0xb1	G C G A A A A A G A C T A G C G A G
	C T G G A C T A C A A G A T C C G C
	G G A A A C T T A T C T T T A G T A
	T A T G C T G C A G C C C G G A T T
	G G T G A G C C A C C C A G G C T G
	C A C A C G C T A G G A C G A A C T
	A C
⋮	
round key 10	
0xbe 0x82 0x89 0x3a 0x1e 0x1e 0xf5 0x6c	A T C C C T A C G T G A G A A T C G
0x18 0x4f 0x3e 0xb1 0x58 0x01 0x0f 0xd5	A T A G G C G T A T C T A G G T T C
	C G A G A C T T T G G G C A C A T C
	T T A A C T A T T G G G G T C A A A
	C T C C G A T A G C C C C C A G C G
	G G C T G C A A A T T A T A T G G G
	A A G A C C C C G A A G C G T A C A
	C T

Table 5 Running of DNAES-cipher

AES	DNAES
Add round key:	DNAES-XOR (Definition 6):
0x0a 0x8a 0x57	GGA-GTC-GGA-ACC-GGA-GTA-TGG-GGA-CAA-GGA-
0xc7 0x32 0x57	TCC-GGA-GAT-AGA-GGA-GTA-GGC-ACC-GGC-GTC-
0x44 0xe7 0x10	CTA-GGC-ACA-GGC-GTA-TGG-GGC-TAG-GGC-TGT-
0xc5 0x57 0xd9	GGA-GAC-AGA-GGC-GTC-GGA-ACC-GGA-GTA-TGG-
0x11 0x8d 0x72	GGC-CAG-GGA-TCA-GGC-GAT-AGT-AGC-GCC-AAA-
0xc7	AGT-CCA-GCA-CTA-CAG-CCA-CCA-GCA-CTC-CTA-
	GTC-TAA-TGT-GGC-AAA-TGG-CCA-AGG-TGA-GGC-
	CTC-TGC-TCA-CCA-GGA-GAT-ATA-TGC-CCA-CAG-
	CTA-TGA-GCC-GGA-ACA-GGA-CTC-GCC-ACC-GGC-
	CCC-CTA-GTC-AAA-ACC-CTA-ATG-ACC-CGA-TGG-
	TCA-TGA-GGC-AGG-GGA-GCC-GAT-GAT-CAA-GGA-
	GAC-CTC-CCA-ACA-GAC-CCC-GCA-CCC-GGC-AAA-
	TGG-TCA-GGA-AAT-TGC-AGT-CTA-AGC-GGC-GTC-
	GGC-ACA-GGC-GTC-TGG-GGC-CAA-GGA-TCA-GGC-
	GAT -

Table 5 continued

AES	DNAES
Substitute bytes:	DNAES-subs (Algorithm 3):
0x67 0x7e 0x5b 0xc6 0x23 0x5b 0x1b 0x94 0xca 0xa6 0x5b 0x35 0x82 0x5d 0x40 0xc6	GGC-GTC-GGC-ACA-GGA-GTC-TGG-GGC-CAA-GGA- TCC-GGC-GAT-AGG-GGC-GTC-GGA-ACC-GGC-GTA- CTC-GGC-ACA-GGC-GTA-TGG-GGA-TAG-GGC-TGT- GGA-GAC-AGA-GGC-GTC-GGC-ACC-GGA-GTA-TGG- GGA-CAG-GGA-TCA-GGC-GAT-AGC-AGT-GCC-AAA- AGT-CCA-GCC-CTC-CAA-CCC-CCC-GCA-CTA-CTA- GTA-TAA-TGC-GGC-AAA-TGG-CCC-AGA-TGA-GGA- CTC-TGC-TCC-CCA-GGC-GAC-ATA-TGT-CCC-CAA- CTC-TGA-GCC-GGA-ACA-GGC-CTA-GCC-ACC-GGC- CCA-CTC-GTC-AAA-ACC-CTA-ATG-ACC-CGA-TGG- TCC-TGA-GGA-AGG-GGC-GCA-GAC-GAC-CAG-GGA- GAC-CTA-CCA-ACA-GAT-CCC-GCA-CCC-GGC-AAG- TGG-TCA-GGC-AAC-TGC-AGC-CTA-AGC-GGA-GTA- GGC-ACA-GGA-GTC-TGG-GGC-CAA-GGA-TCA-GGC- GAT-
Shift rows: 0x67 0x5b 0x5b 0xc6 0x23 0xa6 0x40 0xc6 0xca 0x5d 0x5b 0x94 0x82 0x7e 0x1b 0x35	DNAES-shift rows (Algorithm 5): GGC-GTC-GGC-ACA-GGA-GTC-TGG-GGC-CAA-GGC - TCC-GGA-GAT-AGG-GGA-GTC-GGA-ACC-GGC-GTA - CTC-GGC-ACA-GGC-GTA-TGG-GGA-TAG-GGC-TGT- GGA-GAC-AGA-GGC-GTC-GGC-ACC-GGA-GTA-TGG- GGA-CAG-GGA-TCA-GGA-GAT-AGT-AGC-GCA-AAG- AGC-CCC-GCA-CTA-CAA-CCA-CCA-GCA-CTC-CTA- GTA-TAA-TGT-GGC-AAA-TGG-CCA-AGA-TGA-GGC- CTC-TGC-TCC-CCA-GGC-GAC-ATA-TGT-CCC-CAG- CTA-TGA-GCC-GGC-ACC-GGA-CTC-GCA-ACC-GGC- CCA-CTC-GTC-AAA-ACC-CTA-ATG-ACA-CGA-TGG- TCC-TGA-GGA-AGG-GGA-GCA-GAT-GAC-CAG-GGA- GAC-CTA-CCA-ACA-GAT-CCA-GCC-CCC-GGC-AAG- TGG-TCC-GGC-AAC-TGT-AGT-CTA-AGT-GGC-GTA- GGA-ACA-GGC-GTA-TGG-GGC-CAA-GGC-TCC-GGA- GAC -

Table 6 Continuation of the running of AES and DNAES ciphers

AES	DNAES
Mix columns: 0xbe 0xfa 0xdb 0x3e 0x31 0x72 0x54 0x14 0xa7 0x09 0x86 0x70 0xb3 0x66 0x95 0x92	DNAES-mix columns (Algorithm 7): GGA-GTC-GGC-ACC-GGC-GTC-TGG-GGA-CAG-GGA- TCC-GGA-GAT-AGG-GGC-GTC-GGC-ACC-GGC-GTA- CTC-GGC-ACA-GGC-GTC-TGG-GGA-TAG-GGC-TGT- GGC-GAT-AGG-GGA-GTA-GGC-ACA-GGA-GTA-TGG- GGC-CAG-GGA-TCA-GGA-GAT-AGC-AGC-GCC-AAG- AGT-CCA-GCA-CTA-CAG-CCA-CCC-GCA-CTC-CTA- GTA-TAA-TGC-GGC-AAA-TGG-CCC-AGA-TGA-GGA- CTA-TGT-TCC-CCC-GGA-GAC-ATC-TGC-CCC-CAG- CTA-TGA-GCA-GGC-ACA-GGA-CTA-GCA-ACA-GGC- CCC-CTA-GTA-AAA-ACA-CTC-ATG-ACA-CGA-TGG- TCA-TGA-GGA-AGG-GGC-GCC-GAC-GAT-CAG-GGA-

Table 6 continued

AES	DNAES
	GAC-CTC-CCC-ACA-GAT-CCA-GCC-CCC-GGA-AAA-TGG-TCC-GGC-AAC-TGT-AGC-CTC-AGC-GGC-GTA-GGA-ACC-GGA-GTC-TGG-GGA-CAA-GGC-TCA-GGA-GAT-
Add round key: 0xd2 0x9b 0xd2 0x0e 0x00 0x64 0x7a 0xac 0xe8 0xbc 0x9a 0x63 0x88 0x30 0x98 0x23	DNAES-XOR (Definition 6): GGA-GTC-GGA-ACA-GGC-GTA-TGG-GGC-CAG-GGC-TCC-GGA-GAT-AGG-GGA-GTA-GGC-ACA-GGC-GTA-CTA-GGC-ACA-GGC-GTC-TGG-GGA-TAG-GGC-TGT-GGC-GAC-AGA-GGA-GTA-GGA-ACA-GGA-GTA-TGG-GGA-CAA-GGA-TCA-GGA-GAC-AGT-AGC-GCA-AAG-AGT-CCA-GCA-CTC-CAA-CCC-CCC-GCC-CTC-CTA-GTA-TAA-TGC-GGC-AAG-TGG-CCA-AGG-TGA-GGA-CTC-TGC-TCA-CCA-GGC-GAC-ATC-TGT-CCC-CAA-CTA-TGA-GCC-GGC-ACC-GGC-CTA-GCC-ACA-GGC-CCA-CTC-GTC-AAA-ACA-CTC-ATG-ACC-CGC-TGG-TCA-TGA-GGA-AGA-GGC-GCC-GAC-GAC-CAA-GGA-GAT-CTA-CCA-ACA-GAT-CCA-GCA-CCA-GGA-AAG-TGG-TCC-GGA-AAC-TGC-AGC-CTA-AGT-GGC-GTA-GGA-ACC-GGC-GTC-TGG-GGA-CAA-GGA-TCC-GGA-GAC -
⋮	
output of the last round: 0x07 0x76 0x83 0xe5 0xdd 0xf8 0x9d 0x70 0xf0 0xbd 0xe6 0x77 0xb6 0xfc 0xd1 0xbd	 GGC-GTC-GGC-ACA-GGA-GTA-TGG-GGA-CAA-GGA-TCC-GGC-GAC-AGG-GGC-GTC-GGA-ACC-GGC-GTA-CTA-GGA-ACA-GGA-GTC-TGG-GGC-TAG-GGA-TGT-GGA-GAC-AGG-GGC-GTC-GGC-ACA-GGC-GTC-TGG-GGC-CAA-GGC-TCC-GGA-GAC-AGC-AGT-GCC-AAG-AGT-CCC-GCC-CTA-CAG-CCC-CCC-GCA-CTA-CTC-GTA-TAA-TGC-GGA-AAA-TGG-CCC-AGG-TGA-GGC-CTA-TGC-TCA-CCA-GGA-GAT-ATC-TGT-CCC-CAG-CTA-TGA-GCC-GGC-ACC-GGC-CTA-GCC-ACA-GGC-CCC-CTA-GTA-AAG-ACC-CTC-ATG-ACC-CGC-TGG-TCC-TGA-GGA-AGG-GGC-GCC-GAC-GAC-CAG-GGC-GAC-CTC-CCC-ACA-GAT-CCA-GCA-CCC-GGC-AAG-TGG-TCC-GGC-AAT-TGT-AGC-CTA-AGC-GGC-GTA-GGC-ACC-GGC-GTA-TGG-GGC-CAG-GGC-TCC-GGA-GAT-

Table 7 The average of execution times in seconds for DNAES

DNAES			
Type of data	Number of blocks	Compile time	Encryption Time
text	1000	0.05	2.37
image	32819	1.65	98.93
audio	53783	2.5	187.23

altered the composition of the resulting proteins. We have presented how to encode and decode binary data in DNA strands and algorithms for performing the steps of the AES cipher based on the computation of DNA with silent mutations. If someone fixes the key of DNAES, then DNAES can be used to hide data.

The new presented technique that used to design the DNAES-cipher can be generalized to many block ciphers. An interesting question related to this study is how to apply the same (or similar) technique to some special-purpose encryption algorithms such as [41–43].

Acknowledgements We are grateful to the referees for their valuable comments and remarks.

References

1. Calladine, C.; Drew, H.; Luisi, B.; Travers, A.: Understanding DNA : the molecule and how it works, 3rd edn. Academic Press, Cambridge (2004)
2. Watson, J.: Molecular Biology of the Gene Molecular Biology of the Gene, No. 1 in Molecular Biology of the Gene. Benjamin, New York (1987)
3. Kari, L.; Seki, S.; Sosík, P.: DNA computing–foundations and implications. In: Rozenberg, G., Bäck, T., Kok, J. (eds.) Handbook of Natural Computing, pp. 1073–1127. Springer, Berlin (2012)
4. Adleman, L.: Molecular computation of solutions to combinatorial problems. Science **266**(11), 1021–1024 (1994)
5. Lipton, R.: Using DNA to solve NP-complete problems. Science **268**, 542–545 (1995)
6. Boneh, D.; Dunworth, C.; Lipton, R.; Sgall, J.: On the computational power of DNA. Discrete Appl. Math. **71**(1–3), 79–94 (1996)
7. Boneh, D.; Dunworth, C.; Lipton, R.: Breaking DES using a molecular computer. In: DNA Based Computers, Proceedings of a DIMACS Workshop, Princeton, New Jersey, USA, April 4, 1995, pp. 37–66 (1995)
8. Abbasy, M.; Manaf, A.; Shahidan, M.: Data Hiding method based on DNA basic characteristics. In: Ariwa, E., El-Qawasmeh, E. (eds.) Proceedings of the Digital Enterprise and Information Systems: International Conference, DEIS 2011, London, UK, July 20–22, 2011, pp. 53–62. Springer (2011)
9. Abbasy, M.; Nikfard, P.; Ordi, A.; Torkaman, M.: DNA base data hiding algorithm. Int. J. New Comput. Archit. Appl. **2**(1), 183–193 (2012)
10. Gehani, A.; LaBean, T.; Reif, J.: DNA-based cryptography, in aspects of molecular computing. In: Jonoska, N., ăun, G.P., Rozenberg, G. (eds.) Essays Dedicated to Tom Head, on the Occasion of His 70th Birthday, pp. 167–188. Springer, Berlin (2004)
11. Hamed, G.; Marey, M.; El-Sayed, S.; Tolba, F.: DNA based steganography: survey and analysis for parameters optimization. In: Hassanien, A., Grosan, C., Tolba, F. (eds.) Applications of Intelligent Optimization in Biology and Medicine: Current Trends and Open Problems, pp. 47–89. Springer, Cham (2016)
12. Tang, Q.; Ma, G.; Zhang, W.; Yu, N.: Reversible data hiding for DNA sequences and its applications. Int. J. Digit. Crime For. **6**(4), 1–13 (2014)
13. Atito, A.; Khalifa, A.; Rida, S.: DNA-based data encryption and hiding using playfair and insertion techniques. J. Commun. Comput. Eng. **2**, 44–49 (2012)
14. Guo, C.; Chang, C.; Wang, Z.: A new data hiding scheme based on DNA sequence. Int. J. Innov. Comput. Inf. Control **8**, 1–11 (2012)
15. Khalifa, A.: LSBase: a key encapsulation scheme to improve hybrid crypto-systems using DNA steganography. In: 8th International Conference on Computer Engineering and Systems. IEEE (2013)
16. Khalifa, A.; Atito, A.: High-capacity DNA-based steganography. In: 8th International Conference on Informatics and Systems. IEEE (2012)
17. Skariya, M.; Varghese, M.: Enhanced double layer security using RSA over DNA based data encryption system. Int. J. Comput. Sci. Eng. Technol. **4**, 746–750 (2013)
18. Taur, J.; Lin, H.; Lee, H.; Tao, C.: Data hiding in DNA sequences based on table lookup substitution. Int. J. Innov. Comput. Inf. Control **8**, 6585–6598 (2012)
19. UbaidurRahmana, N.H.; Balamuruganb, C.; Mariappanab, R.: A novel dna computing based encryption and decryption algorithm. Proc. Comput. Sci. **46**, 463–475 (2015)
20. UbaidurRahmana, N.H.; Balamuruganb, C.; Mariappanab, R.: A novel string matrix data structure for DNA encoding algorithm. Proc. Comput. Sci. **46**, 820–832 (2015)
21. Cui, G.; Qin, L.; Wang, Y.; Zhang, X.: An encryption scheme using DNA technology. In: Third International Conference on Bio-Inspired Computing: Theories and Applications, pp. 37–42 (2008)
22. Sabry, M.; Hashem, M.; Nazmy, T.; Khalifa, M.: Design of DNA-based advanced encryption standard (AES). In: 2015 IEEE Seventh International Conference on Intelligent Computing and Information Systems (ICICIS), pp. 390–397 (2015)
23. Xin-she, L.; Lei, Z.; Yu-pu, H.: A novel generation key scheme based on DNA. In: International Conference on Computational Intelligence and Security, pp. 264–266 (2008)
24. Amin, S.; Saeb, M.; El-Gindi, S.: A DNA-based implementation of YAEA Encryption Algorithm. In: IASTED International Conference on Computational Intelligence, pp. 120–125 (2006)
25. Sabry, M.; Hashem, M.; Nazmy, T.: Three reversible data encoding algorithms based on DNA and amino acids structure. Int. J. Comput. Appl. **54**(8), 24–30 (2012)
26. Sadeg, S.; Gougache, M.; Mansouri, N.; Drias, H.: An encryption algorithm inspired from DNA. In: IEEE Proceedings of International Conference on Machine and Web Intelligence (ICMWI), pp. 344–349 (2010)
27. Wang, X.; Zhang, Q.: DNA computing-based cryptography. In: 2009 Fourth International on Conference on Bio-Inspired Computing, pp. 1–3 (2009)
28. Padma, B.T.: DNA computing theory with ECC. http://www.scribd.com/doc/55154238/Report (2010)
29. Agrawal, A.; Bhopale, A.; Sharma, J.; Ali, M.; Gautam, D.: Implementation of DNA algorithm for secure voice communication. Int. J. Sci. Eng. Res. **3**, 362 (2012)
30. Alberts, B.; Bray, D.; Lewis, J.; Raff, M.; Roberts, K.; Watson, J.: Molecular Biology of the Cell, 4th edn. Garland, Oxfordshire (2002)
31. Clelland, C.; Risca, V.; Bancroft, C.: Hiding messages in DNA microdots. Nature **399**(6736), 533–534 (1999)
32. Cui, G.; Wang, Y.; Han, D.; Wang, Y.; Wang, Z.; Wu, Y.: An encryption scheme based on DNA microdots technology. In: Pan, L., Păun, G., Pérez-Jiménez, M., Song, T. (eds.) Proceedings of the Bio-Inspired Computing-Theories and Applications: 9th International Conference, BIC-TA 2014, Wuhan, China, pp. 78–82. Springer, Heidelberg (2014)
33. Jiao, S.; Goutte, R.: Hiding data in DNA of living organisms. Nat. Sci. **1**(3), 181–184 (2009)
34. Santoso, K.; Kwon, K.; Lee, S.; Kwon, S.: High capacity data hiding method in DNA with mutation handling. In: Proceedings of the 1st International Workshop on Information Hiding and Its Criteria for Evaluation, IWIHC '14, ACM, New York, NY, USA, pp. 56–63 (2014)
35. Claybourne, A.: Introduction to Genes and DNA. Usborne Publishing Ltd., London (2014)

36. Calladine, C.; Drew, H.; Luisi, B.; Travers, A.: Understanding DNA: The Molecule and How it Works, 3rd edn. Academic Press, Cambridge (2004)
37. Daemen, J.; Rijmen, V.: The Design of Rijndael. Springer, Secaucus (2002)
38. Stallings, W.: Cryptography and Network Security, 4th edn. Prentice-Hall Inc., Upper Saddle River (2005)
39. Silverman, J.: Fast multiplication in finite fields GF(2^n). In: KoçÇ, C., Paar, C. Proceedings of the Cryptographic Hardware and Embedded Systems: First International Workshop, CHES'99 Worcester, MA, USA, August 12–13, 1999, pp. 122–134. Springer (1999)
40. Ahmed, E.; Shaaban, E.; Hashem, M.: Lightweight mix columns implementation for AES. In: Proceedings of the 11th WSEAS International Conference on Mathematical Methods and Computational Techniques in Electrical Engineering MMACTEE'09, pp. 48–53 (2009)
41. Kaur, M.; Kumar, V.: Colour image encryption technique using differential evolution in non-subsampled contourlet transform domain. IET Image Process. **12**(7), 1273–1283 (2018)
42. Kaur, M.; Kumar, V.: An efficient image encryption method based on improved lorenz chaotic system. Electron. Lett. **54**(9), 562–564 (2018)
43. Kaur, M; Kumar, V (2018) Adaptive differential evolution-based lorenz chaotic system for image encryption. Arab. J. Sci. Eng. https://doi.org/10.1007/s13369-018-3355-3