Implementing Modern Cryptographic Protocols Using DNA and RNA Information Processing

Arash Karimi †,

†Iran University of Science and Technology (IUST), Narmak, Tehran, Iran

Summary
DNA computing is promising in providing primitives of classical cryptography since it provides a variety of advantages over conventional silicon-based computing paradigms. These advantages include massive parallelism and data hiding capability which give it the computing flavor of Turing machine as well as the power of parallel processing which makes it suitable in different applications of classical cryptography. Although modern cryptography has expanded ideas of classical cryptography to include more rigorous security proofs and first class protocols based on concrete mathematics for ensuring secrecy in cryptographic protocols, the DNA computers have not followed the rapid pace by which cryptography community is going ahead and no biological computer has been designed to meet the growing demand of modern cryptographic protocols. For this reason, in this paper, we propose four DNA/RNA computers which implement some modern security primitives such as the mental poker game, zero-knowledge proofs, signature schemes and public key cryptography. Security protocols based on wetware for these primitives have been given which are intended for implementation in the wet lab using standard genetic engineering techniques on DNA and RNA molecules. Security proofs have been presented for each one of our proposed methods which complete the theoretical merits of our modern security protocols.

Key words:
DNA computing, Mental poker, Public-key cryptography, Signature schemes, Zero-Knowledge proof systems.

1. Introduction

Classical cryptography went on until the end of the last century with a special focus on designing and breaking codes to ensure secrecy of data mostly in military applications. But spreading those applications which demand higher levels of secrecy, such as online banking transactions and internet services, has enlarged scopes of classical cryptography to a more sophisticated extent for rigorous analysis of cryptosystems and protocols and to guarantee secrecy of any system which aims to ensure security of messages over an insecure communication channel. Therefore, cryptography has evolved from an art to design codes to the scientific discipline of modern cryptography which provides mathematical reasoning and techniques for proving correctness of secrecy protocols and designing mathematically sound cryptosystems and protocols. In this regard, Due to their splendid advantages for carrying out computations which need a large amount of space and time such as cryptanalysis, DNA computers have proved promising substitutions for silicon-based computers in that their building blocks (DNA molecules) are massively parallel and can store tremendous amounts of data which make them suitable for data hiding and storage as well as utilization in cryptosystem-cracker machines for the inherent parallel processing capabilities they provide.

The first theoretical idea of providing computation capabilities of DNA information processing was put forward by T. Head in 1987 [1] in which he showed the computing power equal to that of Turing machines (the most powerful model of computation) for the operation of DNA cleavage and pasting which takes place as an integral part of DNA processing in the cells of organisms. The above idea did not become feasible until the breakthrough discovery of capability of DNA molecules in solving NP-complete problems which was conducted by L. Adleman [2] in 1994. This outstanding paper established an interdisciplinary science called DNA computing. Following this evolutionary step toward a new computing media many other papers showed up that addressed computationally hard problems which can be solved using DNA computers. Many of these proposals were actually implemented in genetic engineering laboratories. On the other hand, RNA molecules which can be gained by transcription of DNA molecules showed promising as other useful media to build computers based on bio-molecules. As an example, in [3] an RNA computer is designed for solving chess problems and in [4] RNA synthetic devices are designed for information processing inside cells of an organism. A large number of papers have addressed usual security primitives such as information hiding [4], breaking cryptosystems [5]-[8] using DNA molecules. We previously proposed a new watermarking and authentication scheme in context of DNA computation in [9] and [10], respectively.

The bulk of research in DNA computing for providing secrecy has mostly focused on provision of security in the classical framework of building or breaking cryptosystems but has not considered emerging modern cryptographic

Manuscript received November 5, 2011
Manuscript revised November 20, 2011

primitives such as zero-knowledge proofs or public key or signature schemes, etc. In this paper we design a number of modern cryptographic primitives based on emergent DNA and RNA based computers and then prove the security provided by each one of our secrecy primitives. Our proposed schemes are implementable solely by DNA or RNA molecules and consist of a wet public key cryptosystem, a wet signature scheme, a zero-knowledge proof system based on synthetic RNA devices and a mental poker protocol based on DNA information processing techniques. The proposed *in-vitro* and *in-vivo* methods for modern cryptography are the first modern secrecy protocols implemented using genetic engineering techniques.

The rest of the paper is organized as follows. In section 2, the preliminary backgrounds are presented. In section 3, the proposed scheme for a public-key cryptosystem has been presented. A DNA-based signature scheme is also introduced in section 4. A protocol for playing mental poker in wet environment is introduced in section 5. We utilize RNA manipulation techniques for a biological method for solving the Sudoku puzzle in section 6. Finally, conclusions are drawn in section 7.

2. Preliminary backgrounds

In this section, we introduce the encoding scheme which is used in the proposed ideas of this paper.

We previously defined an encoding scheme based on the silent mutation property of the genetic code in [9] which can be defined as follows.

Each bit of the message is encoded in a codon which contains three nucleotides of DNA. In order to encode logical zero in that codon, we do not change the nucleotides of the codon therefore, neither the codon nor the encoded amino acid changes. But if we aim to encode logical one, we must mutate the third nucleotide of the corresponding codon according to the table of Fig. 1. highlighted with yellow marker so that the resultant amino acid and so the corresponding phenotype does not change but the sequence of transmitted DNA changes. Therefore, by sequencing the received DNA sequence with the original sequence of nucleotides the receiver side can guess the transmitted digital data sequence. The data can be encoded into a DNA sequence as described above. If the codons are selected from the yellow sections of the table of Fig. 1, they are said to have multiforms. In order to encode data into an artificial gene we must make sure that the codons that encode logical zeros and ones have multiple forms and therefore they must be selected from the highlighted sections of Fig. 1.

First Position	Second Position of Codon: T	Second Position of Codon: C	Second Position of Codon: A	Second Position of Codon: G	Third Position
T	TTT Phe [F] TTC Phe [F] TTA Leu [L] TTG Leu [L]	TCT Ser [S] TCC Ser [S] TCA Ser [S] TCG Ser [S]	TAT Tyr [Y] TAC Tyr [Y] TAA Ter [end] TAG Ter [end]	TGT Cys [C] TGC Cys [C] TGA Ter [end] TGG Trp [W]	T C A G
C	CTT Leu [L] CTC Leu [L] CTA Leu [L] CTG Leu [L]	CCT Pro [P] CCC Pro [P] CCA Pro [P] CCG Pro [P]	CAT His [H] CAC His [H] CAA Gln [Q] CAG Gln [Q]	CGT Arg [R] CGC Arg [R] CGA Arg [R] CGG Arg [R]	T C A G
A	ATT Ile [I] ATC Ile [I] ATA Ile [I] ATG Met [M]	ACT Thr [T] ACC Thr [T] ACA Thr [T] ACG Thr [T]	AAT Asn [N] AAC Asn [N] AAA Lys [K] AAG Lys [K]	AGT Ser [S] AGC Ser [S] AGA Arg [R] AGG Arg [R]	T C A G
G	GTT Val [V] GTC Val [V] GTA Val [V] GTG Val [V]	GCT Ala [A] GCC Ala [A] GCA Ala [A] GCG Ala [A]	GAT Asp [D] GAC Asp [D] GAA Glu [E] GAG Glu [E]	GGT Gly [G] GGC Gly [G] GGA Gly [G] GGG Gly [G]	T C A G

Fig. 1 The genetic code table.

A plasmid is a circular sequence of DNA which is a usual material used in genetic engineering laboratory. A number of genes can be inserted or cloned in any plasmid. Data must be coded according to the above encoding scheme inside artificial sequences of DNA in the plasmid. Different materials and chemicals can be added to the plate which contains synthetic plasmids which serves as a testbed in our proposed experiments.

3. The Proposed Scheme for a Public Key Cryptosystem

In this section we show our proposed scheme for a public-key cryptosystem using the procedures which occur naturally as an integral part of gene expression in all living organisms. We define our initial setup for the public-key cryptosystem as shown below:

We utilize the silent mutation property of amino acids as demonstrated in section 2 for encoding our messages into the blocks of DNA sequences in the synthetic plasmids conveying the information. Furthermore, we define public and private key of the cryptosystem as shown in Eq. (1)-(2).

$$private\ Key \equiv (A\, biochemical\ indirect\ \ activator) \quad (1)$$

$$public\ Key \equiv (A\ biochemical\ inhibitor,\ A\ known\ gene\ with\ a\ certain\ phenotype\ (a\ reporter\ gene),\ A\ known\ sequence\ of\ nucleotide\ s\ for\ padding\ after\ the\ message\ sequence) \quad (2)$$

Eq. (1) and Eq. (2) show the private and public key of our proposed cryptosystem, respectively. The first element of both of which is a biochemical substance which can be naturally found in the bacteria we work with. As can be seen in Eq. (1), the private key is a biochemical indirect activator which indirectly activates expression of the genes which lie downstream of the promoter of the synthetic gene sequence which encodes the message.

Furthermore, as Eq. (2) shows, the first element of the public key is a biochemical inhibitor which effectively blocks expression of the downstream gene(s) of the promoter of the plasmid which encodes the public key of the proposed cryptosystem, the second element of Eq. (2) is a known gene with a specific phenotype and the third element of it demonstrates a known sequence of nucleotides which shows that the message data has ended. Any gene to be expressed needs a promoter which is upstream of it along with that gene which comes after it as shown in Fig. 2.

In order to provide an example to demonstrate our encryption mechanics, we use Eq. (3)-(4) to express the private-public key pairs of the encryption scheme.

$$private\ Key \equiv (IPTG) \quad (3)$$

$$public\ Key \equiv (LacI, GFP\ gene, A\ DNA\ sequence\ for\ padding\ after\ data) \quad (4)$$

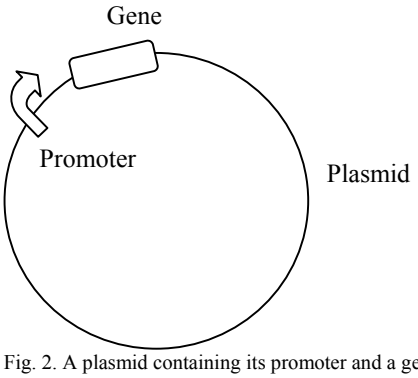

Fig. 2. A plasmid containing its promoter and a gene

In Eq. (3), IPTG or Isopropyl β-D-1-thiogalactopyranoside is a biochemical reagent which induces transcription of the gene that encodes for beta-galactosidase, a hydrolase enzyme which cooperates in catalyzing the hydrolysis of β-galactosides to monosaccharide.

Also, in Eq. (4), the public key contains a biochemical substance (*LacI* protein) which inhibits transcription of the upstream gene(s) of the promoter which belongs to the message-encoding plasmid.

IPTG molecule (with the following chemical formula $C_9H_{18}O_5S$), when connected to *LacI*, detaches it from the promoter and unblocks expression of the gene(s) downstream of the promoter this process is shown in Fig. 3. With this explanation at hand, we are now ready to describe the algorithm in which Alice encrypts a message and send it to Bob.

Algorithm 1. The proposed scenario for secure communication of Alice and Bob

Step 1. Alice encodes her intended message in accordance with the silent mutation property of the genetic code in some gene(s) which have been cloned in the message information-bearing plasmid.

Step 2. Alice inserts the third element of Eq. (4) which is a known sequence of nucleotides to the message she wishes to send to Bob.

Step 3. Alice, using the public key of Bob which is defined in Eq. (2), encrypts the padded message (chosen from the message space and encoded in message-encoding plasmid using the silent mutation property of the genetic code). The public key can be composed by concatenation of *LacI* and the DNA sequence of a known gene (such as green florescent protein gene) as shown in Fig. 4 which serves as the terminator that reveals that there exists a hidden information after ending the sequence of this reporter gene.

The encryption procedure can be accomplished easily by binding the synthetic concatenation of Fig. 4 to the plasmid which conveys the message information and therefore by blocking expression of the gene(s) which lie downstream of promoter of the message-encoding plasmid.

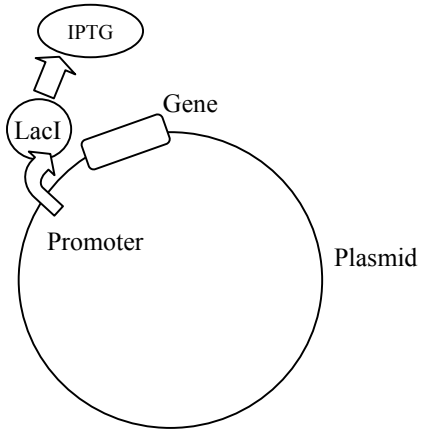

Fig. 3. Detaching LacI from the plasmid by IPTG

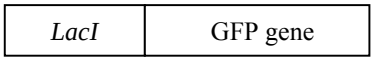

Fig. 4. The first and second elements of the public key of the proposed cryptosystem

In this way, transcription of these genes will be stopped and therefore, we can hide the message encrypted by Alice.
Step 3. Bob receives the solution which contains the encrypted and hidden message sent from Alice. Since using Bob's public key Alice has blocked expression of her intended message, only Bob who possesses his private key has the means to unveil the information which is hidden in the solution received by Alice and then he can extract the message information sent by Alice by unblocking expression of the downstream genes of the message promoter. In our example, Bob by adding IPTG can remove *LacI* and by removing it, the GFP gene is expressed and the solution which contains hidden and coded information turns to green.
Step 4. By analyzing the resulting plasmid, Bob can unveil the message sent by Alice which lies between the GFP gene and the known sequence of nucleotides which was previously defined as a part of Bob's public key.
Step 5. By decoding the sequence of nucleotides which was derived by Bob in step 4, according to the genetic code table shown in Fig. 1, he can find out the message Alice sent for her.

4. A DNA-based signature scheme

In this section, we introduce a new signature scheme based on the hybrid DNA manipulation in the cell.
We consider the following scenario in which Alice wants to prove her identity to Bob by providing her signature on the message she has transmitted to him.
In what follows, we define an algorithmic procedure by which Alice can prove her identity to Bob as the first step of establishing a legitimate contact with him.
Algorithm 2. A wetware signature scheme
Step 1. Alice encodes her intended message using the silent mutation property of the genetic code.
Step 2. Alice pads end of the message by inserting a known predefined sequence of nucleotides.
Step 3. Alice adds her wet signature on her intended message and prepares a solution containing them. Her wet signature is a biochemical transcription inhibitor such as LacI which is bind to the promoter of the plasmid that encodes her intended message.
Step 4. Alice also adds a reporter gene such as florescent genes to the plasmid that encodes her intended message.
Step 5. Alice sends the solution containing the encoded message and her signature to Bob.
Step 6. Bob receives the solution containing the signed message and uses public key of Alice (a biochemical indirect activator (such as IPTG)) to remove signature of Alice and thus to make sure the message has been sent from Alice.
Step 7. After removal of signature of Alice from her signed message, Bob decodes the message which was sent by Alice using the genetic code table.

5. A protocol for playing mental poker using DNA manipulation techniques

In the sequel, we use the commutative property of our encryption scheme and then propose a solution for playing mental poker in wet environment. We call this mental poker game, *Wet Poker*. First of all, we show the commutative property of our encryption method in the following theorem.

Theorem 1. The public key protocol proposed in section 3 with the elements shown in Eq. (1)-(2) has the commutative property, i.e. if data is encrypted more than once, the order in which data is decrypted does not matter. Therefore, assuming that we show the encryption operation of message M using key I by E_I (M) and decryption of ciphertext C with key J with D_J (C), if Alice encrypts some message M with her public key A to produce E_A(M) and Bob encrypts E_A(M) with his public key, B, to achieve E_B (E_A (M)), for our encryption scheme we have: $D_{A'}$ ($D_{B'}$ (E_B (E_A (M)))) = $D_{B'}$ ($D_{A'}$ (E_B (E_A (M)))) = $D_{A'}$ ($D_{B'}$ (E_A (E_B (M)))) = $D_{B'}$ ($D_{A'}$ (E_A (E_B (M))))

In the above notation, pairs of (A, A') and (B, B') are (transmitter, receiver) key pairs for Alice and Bob (i.e. if for example A is used to encrypt a message, A' should be utilized for decrypting that message).

Proof. Since the encryption of a message by key A adds a biochemical compound to the plasmid which is bind to the promoter of the plasmid, if we assume that substance A is added downstream of the promoter and blocks expression of the message, if we provide another encryption to the message, say by key B, then we have added a new blocking biochemical compound downstream of the promoter which inhibits expression of the genes downstream of promoter. Now, in order to decrypt the double encrypted message M which is $E_A(E_B(M))$, we have to add a biochemical activator to the resultant solution to unbind the biochemical inhibitors from the promoter. Since A' has no effect on B and when it is added to the solution just finds its pair (i.e. A) and since the chemical interactions take place in space, it does not matter which one of A' or B' are added first because they are assumed to have no effect on the other. Therefore, it does not matter if we decrypt the resultant solution using

A' or B' at first. The $D_{A'}(D_{B'}(E_A(E_B(M))))$ and $D_{B'}(D_{A'}(E_A(E_B(M))))$ are equal. Their encryption order is not important as well; because encryption of a message in our setting is basically adding a biochemical substance to the solution which contains a plasmid which encodes the message to be encrypted. So, in general, order of encryption and decryption is not the matter of importance in our wetware encryption setup if we use the appropriate key pairs for encryption and decryption. So $D_{A'}(D_{B'}(E_A(E_B(M))))$ and $D_{A'}(D_{B'}(E_B(E_A(M))))$ are also equal. Therefore in general, $D_{A'}(D_{B'}(E_A(E_B(M)))) = D_{B'}(D_{A'}(E_A(E_B(M)))) = D_{A'}(D_{B'}(E_B(E_A(M)))) = D_{A'}(D_{B'}(E_A(E_B(M))))$ and our encryption scheme is commutative.

Assuming that we have a deck of cards containing 52 cards (messages) encoded in *ASCII* format and then coded in the message plasmid as shown in table 1, we utilize the commutative property of our proposed encryption as stated in theorem (1) to devise a protocol for shuffling the input genetic deck between Alice and Bob in Algorithm 3.

We assume that each message of cards in table 1 is encoded in one part of one or more genes. In table 1 three hypothetical genes are demonstrated which encode certain phenotypes (i.e. reporter genes). Each codon of the genetic code is translated into an amino acid which must be selected between those amino acids with multiple forms as shown in genetic code table of Fig. 1. In order to use our previously proposed encoding scheme [9], we must apply silent mutation to those codons that encode logical one according to the genetic code table of Fig. 1. In order to use our previously proposed encoding scheme [9], we must apply silent mutation to those codons that encode logical one according to the genetic code table of Fig. 1 and do not change those codons that encode zero in their ASCII code. As can be seen in table 1, all the amino acids of the fourth column have multiform property, i.e. by changing the third nucleotide of the corresponding codon of column 3 of table 1, we can achieve to an amino acid according to table of Fig. 1 which is in the multiform set of each amino acid.

Algorithm 3. A protocol for shuffling the deck encoded in synthetic genetic blocks

Step 1. Alice and Bob agree on the rules of table 1 and are given the information of this table for all cards specified in the first column of the table.

Step 2. Alice prepares a solution encoding an encryption key and encrypts each card of the deck using this key according to the proposed encryption scheme in a single plasmid in which the promoter sequence comes before the genetic code sequences of all cards in the deck.

Step 3. Alice changes the placement of the encoded messages (genes) in a random fashion or in the other words, *shuffles* the deck of cards.

Step 4. Alice gives the solution containing the encrypted and shuffled deck to Bob.

Step 5. Bob also prepares a solution encoding his encryption key with which he encrypts all the shuffled and encrypted messages received from Alice.

Step 6. Bob, randomly, permutes the deck which is composed of genes containing the card information in the message-encoding plasmid.

Step 7. Bob gives the resultant solution back to Alice.

Step 8. Alice unveils and decrypts each card with her key. She does not know about these cards, since Bob has already encrypted them.

Step 9. Alice encrypts each card using predefined keys each one in each plasmid.

Step 10. Alice gives the resultant plasmid containing the encrypted version of all decks to Bob.

Step 11. Bob uses his key to decrypt all cards which still have Alice's keys on them.

Step 12. Bob also encrypts each card using predefined keys each one in each plasmid.

Step 13. Bob renders the resultant solution to Alice.

Note that the solution derived in step 13 is the shuffled deck which has been produced during a series of solution modification and exchange between Alice and Bob.

Now that the players of the wet poker game shuffled the deck of cards, they are ready to play the game. First, we define the game in a mental manner.

The game starts with three cards chosen at random between decks of shuffled cards. And we further assume that Bob starts the game.

In the following, we demonstrate an algorithm for playing wet poker.

Algorithm 4. A protocol for playing wet poker

Step 1. Three cards are randomly chosen from the shuffled deck of cards produced in algorithm 3.

Step 2. Bob prepares a DNA solution containing the encoded version of his selected key.

Step 3. Bob encrypts these three cards using his chosen key.

Step 4. Bob sends these cards in a random ordering to Alice.

Step 5. Since Alice has no idea about the contents of these cards, she just chooses one card for Bob and one for herself and distinguishes between these cards by applying an encryption with her own key on that card.

Step 6. Alice shuffles the order of received cards by randomly changing the order of received genes which convey the cards information encoded via the silent mutation property of the genetic code table in accordance with the third column of table 1.

Step 7. Alice sends the soup containing the shuffled and encrypted messages to Bob.

Step 8. Bob has no way to know which card is which. So, he just decrypts both cards which are in the DNA soup and keeps the soup which codes for his own card.
Step 9. Bob sends back Alice's DNA soup which contains her own card.
Step 10. Alice decrypts her received DNA soup to find out the contents of it.
Step 11. Bob also decrypts the soup containing his card to know about his hand.
Step 12. Ultimately, Alice and Bob compare their hands to determine whose hand is bigger and therefore who is the winner.

TABLE 1: INFORMATION OF ALL CARDS ENCODED IN DIFFERENT FORMATS

Messages (Cards)	The *ASCII* code of messages	Genetic code of messages	The corresponding Amino acids
"TWO OF CLUBS"	"1010100101011 1 10011110100000 10011111000110 01000001000011 10011001010101 10000101010011 "	# 7&7#&&7#$&&#*77# 7&$#**&#$&7#$&*# *7&#*&$#*&7#$&$# *&*#$&7#7&*#*7$# **$#*&&#&*&#$&$# &*$#&7$#&*&#&&&# &7*#*&&#$&7#7&$# &*$#$&7#*&*#*77# &7*#&&*#$&&#$&*# &*$#&&7#7&7#7&$# **7#*&*#&7$#&7*# **7#*&*#&*&#$&*# *&*#7&7#$&7#&7*# ***#7&&#$&*#$&$# *&7#7&*#&&*#7&$# $&$#**&#*7*#$&&# &*7#*7$#*&7#7&&# *77#&&$#*&$#&*&# $&7#*&&#*&&#$&7# 7&7#$&$#&&7#7&&# *&7#7&*#*&$#$&7#	# 6˘ß# 3ßŁ# 7ªß# 9˚°# 6˘ß# *Ê# 7ªß# 7ªß# 9˚°# $°°# $°°# 7ªß# $°°# 7ªß# 6˘ß# 9˚°# *Ê# $°°# $ßÆ# 7ªß# $ßÆ# /˘°# $ßÆ# 3ßŁ# /˘°# $°°# 7ªß# 6˘ß# $ßÆ# 7ªß# $°°# 9˚°# /˘°# 3ßŁ# 7ªß# 7ªß# $ßÆ# 3ßŁ# 6˘ß# 6˘ß# *Ê# $°°# /˘°# /˘°# *Ê# $°°# $ßÆ# 7ªß# $°°# 6˘ß# 7ªß# /˘°# *Ê# 6˘ß# 7ªß# 7ªß# $°°# 6˘ß# 3ßŁ# 6˘ß# 7ªß# *Ê# 9˚°# 7ªß# $ßÆ# 9˚°# $°°# 6˘ß# 9˚°# 3ßŁ# $°°# $ßÆ# 7ªß# $°°# $°°# 7ªß# 6˘ß# 7ªß# 3ßŁ# 6˘ß# $°°####6˘ß#####$°°#####7ªß#
"THREE OF CLUBS"	"1010100100100 0 10100101000101 10001010100000 10011111000110 01000001000011 10011001010101 10000101010011 "	# $&&#$&*#&&$#$&$# $&7#$&$#*77#7&7# &&$#&7&#$&7#$&$# &7$#$&$#&*&#$&$# &7$#**&#&7$#$&$# &*&#$&$#7&&#7&&# &7$#&7&#$&*#&&&# &&*#7&7#&&*#7&$# *&*#&&&#$&*#*&*# *&$#7&*#$&7#*&&# *&7#&7$#*77#7&*# &7&#&77#&7&#7&$# &**#7&7#**7#7&*# $&*#7&$#*7$#&*$# 7&7#&&7#7&*#&&*# 7&$#$*7#7&*#**&# $&*#$&$#$&*#&&7# 7&*#**$#&77#&7&# 7&7#$&&#$&7#&**# &&&#&*&#7&7#&7$# 7&&#$&$#$&*#*&&#	# 7ªß####7ªß#####/˘°###7ªß# 7ªß#####7ªß####9˚°####6˘ß# 3ßŁ#####/˘°####7ªß####7ªß# /˘°#####7ªß####$ßÆ###7ªß# /˘°#####*Ê#####/˘°###7ªß# $ßÆ#####7ªß####6˘ß####6˘ß# /˘°#####/˘°####7ªß###3ßŁ# 3ßŁ#####6˘ß####3ßŁ#####6˘ß# $°°#####3ßŁ####7ªß####$°°# $°°#####6˘ß#####7ªß####$°°# $°°#####/˘°####9˚°#####6˘ß# /˘°####/˘°####/˘°####6˘ß# $ßÆ####6˘ß#####*Ê#####6˘ß# 7ªß####6˘ß#####9˚°####$ßÆ# 6˘ß####3ßŁ#####6˘ß####3ßŁ# 6˘ß#####7ªß#####6˘ß#####*Ê# 7ªß####7ªß#####7ªß###3ßŁ# 6˘ß#####*Ê#####/˘°###/˘°# 6˘ß#####7ªß#####7ªß###$ßÆ# 3ßŁ####$ßÆ######6˘ß###/˘°# 6˘ß####7ªß#####7ªß###$°°#
⋮	…	…	…
"ACE OF SPADES"	"1000001100001 1 10001010100000 10011111000110 01000001010011 10100001000001 10001001000101 1010011"	# **7#&7$#$&*#$&&# 7&*#$&&#&**#7&&# $&$#$&*#&*$#&77# 7&*#***#**$#$&*# &&*#$&$#*&7#&*$# 7&7#7&*#**$#$&*# *77#**7#*7&#*77# $&7#**7#&&7#&7$# &**#&&*#**7#$&$# &&$#$&*#**7#$&*# 7&7#&*$#7&*#7&7# **&#$&&#**$#$&*# $&$#&&7#&*7#**7# *7*#*&$#**&#&&7# &77#7&7#&7$#$&7# $&7#$&&#**&#*&&# *7*#*&$#**&#7&$# &*&#**&#*&$#*7*# ***#7&7#&*7#$&&# &**#7&&#**&#*7&# &77#&*7#**7#7&7	# *Ê# /˘°# 7ªß# 7ªß# 6˘ß# 7ªß# $ßÆ# 6˘ß# 7ªß# 7ªß# $ßÆ# /˘°# 6˘ß# *Ê# *Ê# 7ªß# 3ßŁ# 7ªß# $°°# $ßÆ# 6˘ß# 6˘ß# *Ê# 7ªß# 9˚°# *Ê# 9˚°# 9˚°# 7ªß# *Ê# 3ßŁ# /˘°# $ßÆ# 3ßŁ# *Ê# 7ªß# 3ßŁ# 7ªß# *Ê# 7ªß# 6˘ß# $ßÆ# 6˘ß# 6˘ß# *Ê# 7ªß# *Ê# 7ªß# 7ªß# 3ßŁ# $ßÆ# *Ê# 9˚°# $°°# *Ê# 3ßŁ# /˘°# 6˘ß# /˘°# 7ªß# 7ªß# 7ªß# *Ê# $°°# 9˚°# $°°# *Ê# 6˘ß# $ßÆ# *Ê# $°°# 9˚°# *Ê# 6˘ß# $ßÆ# 7ªß# $ßÆ# 6˘ß# *Ê# 9˚°# /˘°####$ßÆ#####*Ê#####6˘ß#

6. A biological solution for Sudoku puzzle using hybrid RNA manipulation methods

In this section, we introduce a solution for Sudoku puzzle which uses RNA in-vitro manipulation methods. At first, we define the preliminaries of our problem.

6.1 Definitions

A general instance of Sudoku is defined on an $n \times n$ grid in which $n = k^2$ and subgrids are of size $k \times k$. Assuming that some of the cells of subgrids are already filled with Natural numbers in range {1,…,n}, the goal is to fill the remaining cells of grid with numbers in the above range so that different numbers appear in all rows, columns as well as all subgrids.
Note that Sudoku problem is an NP-complete problem in general and although solving the problem requires non-polynomial time, verification of the solved problem can be accomplished in polynomial time.
In this paper, we assume 2×2 Sudoku problem as shown in Fig. 5 in which subgrids are of size 1×1.
Assuming that only symbols a_1 and a_2 can be placed in each cell of the Sudoku grid, we have six ($\binom{4}{2}$) possible configurations if we further assume that there exist exactly two numbers of each a_i, $i \in \{1,2\}$. These possible configurations are shown in Fig. 6.

a_1	a_2
a_2	a_1

Fig. 5. 2×2 Sudoku table

As can be seen in Fig. 6, only two configurations ((a) and (d)) are answers of the given Sudoku problem and depending on the given symbols inside cells of the grid, one of them can be the answer of the given instance of Sudoku problem.
In order to solve the problem using RNA computations, we assume a hypothetical table as shown in Fig. 7 which demonstrates the locations of the symbols in the grid. Each symbol (a, b, c or d) of the grid defines one of four possible geometric locations of symbols in the grid.
We assume the following coding scheme for symbols and locations as shown in Fig. 8 in which we consider that a_i, $i \in \{1,2\}$ has values in $\{0,1\}$. As can be seen in Fig. 8 we have produced new symbols which have been made out of the position of symbols and the values (0 or 1)

which fit into them. These new symbols which have been shown by capital form of their positions, uniquely define any number in any location of the Sudoku table. The operation of || can be defined as concatenation of its left and right symbols which are RNA segments.

Note that information of the value of just one cell suffices to find a unique answer of the given Sudoku problem, i.e. configurations (a) or (d) in Fig. 6. In order to solve the problem we should mention that all cells that are in the vicinity of each other should be filled with different values and therefore, no two neighboring cells are allowed to have the same symbol.

It is worthwhile that by neighboring we mean that for instance, in Fig. 7, a is in the vicinity of b and c but not d and the set of neighboring locations for b is $\{a, d\}$. So, in the first step, we should beware of the neighboring sets for all positions in the Sudoku table which can be stated as table 2 shows.

We can write algorithm 5 for finding the answer for the defined Sudoku puzzle which is called the neighboring problem.

TABLE 2: THE SET OF NEIGHBORING LOCATIONS

Symbol for cell location	The neighboring set
a	$\{b, c\}$
b	$\{a, d\}$
c	$\{a, d\}$
d	$\{b, c\}$

Algorithm 5. Solving the neighboring problem for the Sudoku puzzle

START

% We are given the initial conditions for the Sudoku problem under consideration as statements in the set: $\{A_i, B_i, C_i, D_i\}$, $i \in \{1,2\}$

IF ($A_1 \mid B_2 \mid C_2 \mid D_1$)

THEN CONFIG = CONFIG 1;

% CONFIG means Sudoku table of the answer

% CONFIG 1 means table (a) in Fig. 6.

ELSEIF ($A_2 \mid B_1 \mid C_1 \mid D_2$)

THEN CONFIG = CONFIG 2;

% CONFIG 2 means table (d) in Fig. 6.

END

IF (CONFIG == CONFIG 1 | CONFIG 2)

PRINT ("TRUE")

END

The above algorithm easily finds the answer for the problem and defines if there is any answer for the given Sudoku problem or not.

Accordingly, we can encode algorithm 5 to a class of SAT (satisfiability) problems as shown in the logical program below.

$$[(\neg A_1 \vee (\neg B_1 \wedge \neg C_1)) \vee (\neg B_2 \vee (\neg A_2 \wedge \neg D_2)) \vee$$
$$(\neg C_2 \vee (\neg A_2 \wedge \neg D_2)) \vee (\neg D_1 \vee (\neg B_1 \wedge \neg C_1)) \vee$$
$$(\neg A_2 \vee (\neg B_2 \wedge \neg C_2)) \vee (\neg B_1 \vee (\neg A_1 \wedge \neg D_1)) \vee$$
$$(\neg C_1 \vee (\neg A_1 \wedge \neg D_1)) \vee (\neg D_2 \vee (\neg B_2 \wedge \neg C_2))]$$

In the above transformation, a true variable means that we are given that specific condition.

Just as utilized in [3], we use RNA library to solve this SAT problem. At first, we prepare the mandatory materials for initialization of our problem.

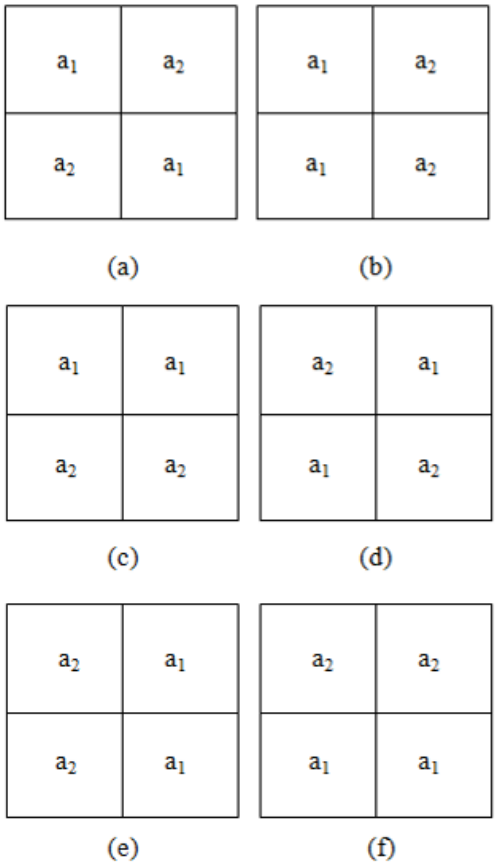

Fig. 6. All possible configurations for 2×2 Sudoku table

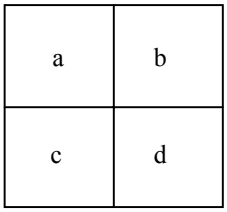

Fig. 7. A hypothetical table showing the possible locations in the Sudoku table

$$\left\{\begin{matrix} a \,||\, a_1 \rightarrow A_1 \\ a \,||\, a_2 \rightarrow A_2 \\ b \,||\, a_1 \rightarrow B_1 \\ b \,||\, a_2 \rightarrow B_2 \end{matrix}\right\}, \left\{\begin{matrix} c \,||\, a_1 \rightarrow C_1 \\ c \,||\, a_2 \rightarrow C_2 \\ d \,||\, a_1 \rightarrow D_1 \\ d \,||\, a_2 \rightarrow D_2 \end{matrix}\right\}$$

Fig. 8. mapping different values in different places to new symbols in the Sudoku table

6.2 Preparing the initial settings for solving Sudoku problem

Our approach for solving the considered Sudoku problem utilizes the RNA model of computation which was proposed in [3] (i.e. a gel-cut-pour computation) and uses RNase H digestions of any inappropriate answers which do not fit the conditions of our considered problem (i.e. the neighboring condition demonstrated in algorithm 5).

We show each strand of RNA which encodes a possible configuration of our considered Sudoku table as shown below:

$5'\,prefix|\ Bit\,1\,(A_1\ or\ A_2)|Spacer\,1|Bit\,2\,(B_1\ or\ B_2)|$ $Spacer\,2|Bit\,3\,(C_1\ or\ C_2)|Spacer\,3|Bit\,4\,(D_1\ or\ D_2)|$ In $Spacer\,4|Suffix\,3'$

the above template of RNA strands, the Spacer sequences are oligonucleotides of a certain length (say 5 bases) and one bit has been devoted to each position (cell) of the Sudoku table which has been assumed to be of 15 bases for A_1, B_1, C_1, D_1 and 16 bases for A_2, B_2, C_2, D_2 and prefix and suffix are also considered to be 24 and 32 bases, respectively [3].

According to the above template of individual RNA strands by putting different bits in the locations of the table and therefore, in their corresponding RNA template, and considering fixed prefix, suffix and spacer sequences, we can produce 16 strands each of which represents a possible configuration of the Sudoku table from which only one configuration is correct given the initial conditions for the problem. The reason of choosing different lengths for a_1 and a_2 is that we should filter the strands synthesized according to the template (16 kinds of strands) to extract 6 strands which represent all table configurations with two numbers of a_1 and two numbers of a_2 as shown in Fig. 6.

In each step of solving the Sudoku problem, we devote an RNA pool to each cell of the Sudoku table with the RNA template as shown above. In the first step of solving the table, we need to make sure that we have filtered all the generated strands to exclude those strands that do not have equal numbers of a_1 and a_2.

This filtering for generation of 6 kinds of strands with two numbers of a_1 and a_2 between 16 kinds of strands can be conducted as follows. We should pour the prepared solution containing all 16 possible configurations of the Sudoku table on a gel (such as agarose gel or polyacrylamide gel) to extract those strands with length of 138 nucleotides and therefore those strands which possess exactly two numbers of a_1 and a_2. In this step 6 kinds of strands are selected.

Then in the next steps we remove the strands which introduce false configurations by digestion and the remaining strands introduce the answer to the considered Sudoku problem. In algorithm 6 we explain this method.

Algorithm 6. Solving the Sudoku problem using RNA library

Step 1. Divide the filtered RNA pool for each cell of the Sudoku table of Fig. 5 into two halves in each tube in which for example for the first cell one tube represents strands that have a_1 in position a and the other tube contains strands that do not have a_1 in position a.

Step 2. In the pool that contains strands which represent a_1 in position a, using RNase H, digest strands which do not have a_1 in position a as well as those strands which represent a_1 in position b and c. Therefore, we have implemented the first logical statement which states ($A_1 \rightarrow \neg B_1 \wedge \neg C_1$) (i.e. if A_1 takes place, B_1 and C_1 events cannot happen).

Step 3. In the pool that contains strands which do not have a_1 in position a digest strands which have a_1 in the same position.

Step 4. Since in the tubes there should be no loss in the consumed mass, we should remove those DNA strands which were used to do the above digestions using spin column purification.

Step 5. Start over from step 1 and repeat this algorithm starting from B_2.

Algorithm 6 implements the above propositional formula and the result of it contains RNA strands which represent the answer for the Sudoku table of Fig. 5.

6.3 Application of the proposed problem in Zero-Knowledge system

A zero-knowledge proof, as defined in [11] is an interactive proof between two parties who wish to communicate. These parties are called *prover* and *verifier*. The prover knows solution to a certain problem. There is a game between the prover and the verifier during which they exchange information and at the end of these sessions, the verifier accepts or rejects the execution.

We show the implications of the above method of solving Sudoku problem for zero-knowledge proof systems and

then prove the completeness of our proposed protocol. For this reason, we consider a scenario in which Alice wants to prove Bob that she has solved the Sudoku problem with the initial conditions selected by Bob. In what follows, we devise a protocol in which Alice convinces Bob that she has solved a problem (the Sudoku problem).

Algorithm 7. Proving identity of Alice to Bob
Step 1. Alice (the prover) and Bob (the verifier) agree on the given Sudoku problem.
Step 2. Bob gives some initial conditions to Alice in terms of some prefilled cells.
Step 3. Alice shows the results of electrophoresis of all initially generated RNA strands to Bob to prove him that she has found the collection of 6 possible types of strands one of which is the answer to the problem.
Step 4. If Bob verified the result of electrophoresis, he can give Alice some other challenges to make sure that she has found the answer to the given Sudoku problem.
Step 5. Alice uses the given initial conditions of Bob in algorithm 6 to solve the problem.
Step 6. Alice utilizes genetic engineering standard procedures to answer the challenge of Bob.
Step 7. Bob checks for what he expects in the answered challenge.
Step 8. Bob verifies the response of the challenge Alice has answered, if it was how he expected, or rejects the response of the challenge otherwise.
Using the 8-step procedure as shown in algorithm 7 Alice can prove Bob that she has solved the Sudoku problem posed by Bob.
The challenge which is mentioned in step 4 of the above algorithm can be a specific pattern that defines the answer uniquely. These specific patterns can unveil the correct answer of the challenge. In the following, we bring some sort of challenges that Bob may give Alice in step 4 through which he can find out if Alice has solved the problem or not.
In Fig. 9 we can see the RNA patterns which have been extracted from the filter (gel) during the electrophoresis operation.

1) $5'prefix|A_1|S|B_2|S|C_2|S|D_1|S|suffix3'$
2) $5'prefix|A_2|S|B_2|S|C_1|S|D_1|S|suffix3'$
3) $5'prefix|A_1|S|B_1|S|C_2|S|D_2|S|suffix3'$
4) $5'prefix|A_1|S|B_2|S|C_1|S|D_2|S|suffix3'$
5) $5'prefix|A_2|S|B_1|S|C_1|S|D_2|S|suffix3'$
6) $5'prefix|A_2|S|B_1|S|C_2|S|D_1|S|suffix3'$

Fig. 9. All extracted RNA patterns from the initial electrophoresis operation (S is the spacer sequence)

As can be seen in Fig. 9, only patterns of number (1) and (5) represent answer to the given Sudoku problem and depending on one initial condition provided by Bob, one of them can be considered as the answer.
The RNA patterns demonstrated in Fig. 9 reveal some information which could guide Bob to propose challenges which uniquely determine the correct answer. For instance, the symmetry of the correct answers can be utilized as follow.
Assume that Bob has provided Alice with a Sudoku table with initial condition as shown in Fig. 10.
At first, he can use restriction enzymes of some kind to cut the strands in two halves as shown in Fig. 11.
After applying restriction enzymes of some kind on the strands of Fig. 11, according to his initial cell information provided to Alice, Bob can observe if there is a certain pattern of oligonucleotides which is a subsequence of B_2SC_2 (and since the first cell of the table contains a_1, Bob knows that the answer to this problem is number (6) of Fig. 9) or not. If he found it, he can verify the answer of the challenge by Alice. No need to say that he does not see the complete answer given by Alice to his challenge and he just checks for some certain points that he knows. Therefore, using our proposed protocol Alice does not need to reveal the whole answer to Bob to prove that she solved the problem.
The completeness of a protocol is defined as the probability that an honest verifier verifies a correct proof. As explained above, in our proposed zero-knowledge protocol there is only one type of strand between all 16 generated types of RNA strands that fits the criteria for acceptance of the correct strands and our wetware zero-knowledge proof system proves to be complete. ■

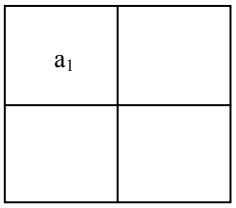

Fig. 10. The proposed Sudoku problem of Bob

1) $5'prefix|A_1|S|B_2|S|C_2|S|D_1|S|suffix3'$
2) $5'prefix|A_2|S|B_2|S|C_1|S|D_1|S|suffix3'$
3) $5'prefix|A_1|S|B_1|S|C_2|S|D_2|S|suffix3'$
4) $5'prefix|A_1|S|B_2|S|C_1|S|D_2|S|suffix3'$
5) $5'prefix|A_2|S|B_1|S|C_1|S|D_2|S|suffix3'$
6) $5'prefix|A_2|S|B_1|S|C_2|S|D_1|S|suffix3'$

Fig. 11. RNA strands of Fig. 9 after applying appropriate restriction enzymes

7. Conclusion

DNA computing is promising in providing secrecy primitives because their substrates (DNA molecules) have the ability to hide information and their power of computation equals that of a parallel computer. In this paper some primitives of modern cryptography are designed in the context of DNA computing. All building blocks of the proposed models are fully constructible solely with biological parts. Our proposed primitives for ensuring secrecy consist of a genetically engineered public key cryptosystem, a DNA-based signature scheme, a protocol for playing mental poker on the wetware, and an RNA-based zero-knowledge proof system based on solving the Sudoku problem. Our proposed genetically engineered computers are designed based on the standard parts and techniques of genetic engineering which, considering the assumed sizes for the problems, can be easily constructed and verified in laboratory. Furthermore, security proofs have been presented for each one of the proposed schemes which show that our proposed genetic machines are mathematically sound.

References

[1] T. Head, Formal Language Theory and DNA: an analysis of the generative capacity of specific recombinant behaviors, *Bull. Math. Biology, vol. 49,* 1987, pp. 737–759.

[2] L.M. Adleman, Molecular computation of solutions to combinatorial problems**,** *Science*, *vol. 266,* 1994, pp. 1021-1024.

[3] A. R. Cukras, D. Faulhammer, R. J. Lipton, L. F. Landweber, Chess games: a model for RNA based computation, *BioSystems vol. 52,* 1999, pp. 35–45.

[4] A. Gehani, T. LaBean, J. Reif, DNA-based Cryptography, Aspects of Molecular Computing, *Springer-Verlag Lecture Notes In Computer Science, vol. 2950,* 2004.

[5] D. Boneh, C. Dunworth, R. Lipton, Breaking DES Using a Molecular Computer, *Princeton CS Tech-Report CS-TR-489-95*, 1995.

[6] L. M. Adleman, P. W. K. Rothemund, S. Roweis and E. Winfree, On Applying Molecular Computation to the Data Encryption Standard, *In Proc. Second Annual Meeting on DNA Based Computers, DIMACS Workshop, Princeton University*, USA, June 1996, pp. 28-48.

[7] S. N. Krishna, R. Rama, Breaking DES Using P System. *Theoretical Computer Science, vol. 299*, 2003, pp. 495-508.

[8] A. Choudhary, K. Krithivasan, Breaking DES Using Networks of Evolutionary Processors with Parallel String Rewriting Rules, *International Journal of Computer Mathematics, Vol. 86,* No. 4, 2009, pp. 567-576.

[9] A. Karimi, R. Dastanian, H. S. Shahhoseini, A New Watermarking Scheme for An *in-vivo* Computer Based on Infection of E. coli, *Proceedings of International Conference on Computer and Electrical Engineering (ICCEE)*, vol. 8, 2010, pp. 484-488. Chengdu, China.

[10] R. Dastanian, A. Karimi, H. S. Shahhoseini, A Novel Multi-Client Authentication Method Using Infection of Bacteria, *Proceedings of International Conference on Communication and Electronics Information (ICCEI)*, vol. 1, 2011, pp. 310-314, Haikou, China.

[11] S. Goldwasser, S. Micali and C. Rackoff, The knowledge complexity of interactive proof systems, *SIAM J. Computing Vol. 18,* no. 1, 1989, pp. 186–208.

Arash Karimi was born in Tehran, Iran in April the 29th, 1985. He received the B.S. and M.S. degrees in the Dept. of Electrical Engineering from Amirkabir University of Science and Technology (Tehran Polytechnic) and Iran University of Science and Technology (IUST), Tehran, Iran, in 2008 and 2011, respectively. His major field of study is telecommunications and cryptography.

He has published a number of papers in international conferences and journals in the field of information assurance. His current field of interest include statistical cryptanalysis, unconventional methods in computation with a focus on cryptanalysis, Biochemical computing, and formal languages and automata. Mr. Karimi is a member of Iran Cryptography Research Association, student branch and a member of Security Study Group Association (SSG), IT organization of Iran.