Commun Nonlinear Sci Numer Simulat 19 (2014) 3136–3148

Contents lists available at ScienceDirect

Commun Nonlinear Sci Numer Simulat

journal homepage: www.elsevier.com/locate/cnsns

Design of fast one-pass authenticated and randomized encryption schema using reversible cellular automata

Kamel Mohamed Faraoun [*]

Computer Science Department, Djilalli Liabbes University, Sidi Bel Abbès, Algeria

ARTICLE INFO

Article history:
Received 30 August 2013
Received in revised form 19 December 2013
Accepted 5 February 2014
Available online 18 February 2014

Keywords:
Authenticated encryption
Second-order cellular automatons
Data security

ABSTRACT

A novel way to provide fast authenticated and randomized encryption is proposed using reversible cellular automata for the first time. A block-based cryptosystem is presented and shown to provide security against both active and passive attackers by the way of a strong authentication mechanism. The proposed system is much faster than existing authenticated encryption standards since only one pass per block is performed to ensure both encryption and integrity. Experimental results show robustness of the cryptosystem against several cryptanalysis techniques, when a formal proof of strict integrity preserving is included. Obtained results demonstrate superiority of the approach in both security and rapidity aspects.

1. Introduction

The most important aspects of cryptographic systems are confidentiality and authenticity handled generally using electronic signatures and hash functions. A cryptosystem that provides both authenticity and confidentiality is considered as an authenticated encryption (AE) schemas. An AE schema allows the receiver of the cipher-text to verify its integrity and authenticity while deciphering to check if a given alteration or modification has occurred during transmission. The corresponding encryption algorithm use a key, a plain-text and an initialization vector (IV)/nonce to produce the cipher-text. When deciphering, the algorithm use the same key and the same IV/nonce to reconstruct the original plain-text if no modification has affected the cipher-text, else its output a special symbol noted $\perp$ to state that the cipher-text is invalid (i.e. has been modified or altered).

The authenticity provided by AE schemas prevents adversaries to create new modified and valid cipher-texts using existing ones. Any new improperly constructed cipher-text is automatically rejected during the decryption phase, so that decryption become only possible for the cipher-texts constructed using the correct key and plain-text. Such mechanism prevents any active attacker that has modification power on cipher-text from gaining any useful information that he hasn't to know. A mode of encryption that offers such security level against active attackers is often called a CCA-secure encryption mode (i.e. secure against chosen cipher-text attack).

Existing authenticated encryption schemas are generally built using symmetric block ciphers. They are constructed by combining a hash functions or a message authentication codes (MAC) that is secure under chosen message attacks, with a semantically CPA-secure (secure against chosen plain-text attack) block cipher. According to [1], a combination that provides the mentioned properties is proofed to be secure against adaptive chosen cipher-text attack and can provide

* Tel.: +213 775323650.
E-mail address: Kamel_mh@yahoo.fr

http://dx.doi.org/10.1016/j.cnsns.2014.02.005
1007-5704/© 2014 Elsevier B.V. All rights reserved.

authenticated encryption. The same author presents and analyzes three possible AE schemas using compositions that meet these required properties. In practice, six different AE modes has been standardized recently by ISO standards [1]: OCB 2.0, Key Wrap, CCM, EAX, Encrypt-then-MAC (EtM) and GCM.

Since standardized constructions of AE schemas use two cryptographic primitives (encryption and hashing) to meet the security requirements, at least two pass per plain-text block are needed to construct the final cipher-text. It is clear that AE schema's performances can be greatly enhanced if only one computation pass is performed, while preserving the same security level of the cryptosystem. Design of such constructions can then be considered as an interesting goal, since highest secure encryption rates can be achieved.

Another important security aspect of encryption schemas is randomization. Randomized encryption produces a cipher-text via a nondeterministic function of the plain-text and the encryption key. For each key, a given message may be encrypted in different ways; the encryption system makes a random choice to decide which way to use. In contrast, encryption schema must be uniquely decodable: for each key each cipher text to at most one plain message. At the cost of increasing the required bandwidth, such procedure may achieve greater cryptographic security than deterministic counterparts by eliminating the threat of chosen plaintext attack (CPA-security).

In the present paper, we propose a simple and robust encryption algorithm that provides high security level against chosen plain-text attacks, and provide secure authenticated encryption without the need of MAC computation and with only one encryption pass for each block of the plain-text. The proposed approach uses the special class of second order reversible elementary cellular automata.

Cellular automata with their intrinsic characteristics of parallelism, homogeneity, and unpredictability are a good tool to build secure cryptosystems. To our knowledge, the proposed approach is the first attempt to use cellular automata for authenticated encryption that can provides a high level of security against active attackers. The paper is organized as follow: Section 2 gives preliminaries of cellular automaton model. The Section 3 gives a brief state-of-the-art about cellular automatons use for cryptography and existing cryptosystems. In Section 4, full description of the proposed approach is detailed with different components of the cryptosystems. Security analysis, performances and experimental results are presented in Sections 5 and 6. Finally, conclusions are drawn in Section 7.

2. Cellular automaton preliminaries

A cellular automata (CA) consist of a number of cells arranged in a regular lattice, each cell has its own state that can change in a discrete time step. States of the whole CA's cells are changed synchronously using a local transition rule that define each new cell's state using its old state, and the states of the corresponding neighbors. The neighbors are a specific selection of cells relatively chosen with respect to a given cell's position that can be defined for each cell i using a radius r on the lattice. This will give $n = 2r + 1$ different neighbor including the cell i itself. The boundaries cells of the lattice are concatenated together in a cyclic form to deal with the finite size automatons.

Formally, if we define the state of a cell i at the time t with q_i^t, its state on time $t + 1$ will depend only on the states of the corresponding neighborhood at the time t, by applying a transition rule f that define the way states are updated. If the neighborhood radius is r, and only two cell states are defined, the length of each transition rule is then 2^{2r+1} bit, and the number of possible rules is $2^{2^{2r+1}}$. For one-dimensional binary CAs, a transition rule is generally coded using the decimal value of the corresponding binary representation. In the present work, we consider one-dimensional binary CAs with radius $r = 3$, so that we have 2^{128} possible rule.

A reversible cellular automaton (RCA) is a class of cellular automata in which every configuration has a unique predecessor. That is, RCAs are constructed in such a way that the state of each cell prior to an update can be determined uniquely from the updated states of all the cells. The time-reversed dynamics of a reversible cellular automaton can always be described by another cellular automaton rule, possibly on a much larger neighborhood. Several methods are known to construct cellular automata rules that are reversible. The second-order cellular automaton method invented by [19], in which the update rule combines states from two previous steps of the automaton permit to turn any one-dimensional binary rule into a reversible one using the fact that the state of a cell at time t depends not only on its neighborhood at time $t - 1$, but also on its state at time $t - 2$. This is achieved by combining the ith cell state at time t with the state of the same cell in time $t - 2$ using the xor operator.

Let define the configuration state of a given CA at each time step t by C^t, we can then characterize the evolution dynamic of a CA by the following equation:

$$C^t = F(C^{t-1}) \tag{1}$$

when the map "F" denote the global evolution map of the used basic CA, and is defined according to the space relation between the cells. We can build a second-order RCA based on the CA defined by (1) using the following equation:

$$C^t = F(C^{t-1}) \oplus C^{t-2} \tag{2}$$

The defined RCA can then be reversed trivially using the following equation:

$$C^{t-2} = F(C^{t-1}) \oplus C^t \tag{3}$$

Second-order RCAs defined using Eq. (3) are always reversible even if the basic used CA defined by the map F is not. We can so construct as mush RCAs as possible existing CAs. Reversibility is performed using the same transition rule in both directions, raising qualitatively the same behavior of one-order CAs as pointed by Wolfram [21]. The use of such defined RCAs is very appropriate for cryptosystems building, and security of such RCAs based cryptosystems is assured by the impossibility to reconstruct initial conditions pair from any given pair of consecutive configurations without the knowledge of the transition rule used initially.

Instead of using one initial configuration like standard one-dimensional CA, two initial configurations are used to evolve a second-order RCA. Starting from two configurations C^{-1} and C^{0} evolving gives two configurations C^{n-1} and C^{n} after n time step. By running the RCA backward starting from C^{n-1} and C^{n} as initial configurations, we can recover the two configurations C^{-1} and C^{0} after exactly n iteration using the same transition rule and the same principle of combining with the $(t-2)$th state at each time step t. Second-order rules constructed in this way are named by Wolfram [21] by appending an “R” to the number or code of the base rule.

3. Cellular automaton for cryptography

Many researchers had explored different cryptosystems based on CAs, since the first Wolfram’s attempt to propose the first secret key process based on them [3]. CAs has been used especially to build stream-based ciphers by exploiting their capability to generate pseudo-random sequences [4,5]. Hortensius et al. [7] used non-uniform CAs with two rules 90 and 150, and they found that the quality of generated pseudo random numbers was better than the quality of the initial Wolfram system. Tomassini and Perrenoud [9] proposed to use non-uniform 1D CAs with $r = 1$ and the four rules 90, 105, 150 and 165, to provide high pseudo-random sequence’s quality with a very large space of possible secret keys that is difficult to cryptanalysis. In [6], a genetic algorithm is used to evolve a population of hybrids one-dimensional CA that was found to produce sequences of high entropy that also passes the FIPS 140-2 statistical tests.

Similarly, CAs has been also used into build block-based ciphers. In [8], the authors implemented a block cryptosystem based on additive CAs with group properties. Kari [11] proposed a cryptosystem with reversible CA, when Zhang presented in [12] a different method of encryption based on reversible CA that has a larger key space. Another reversible CA (RCA) based encryption algorithm is proposed in [13] that satisfies the avalanche criteria, but trades off with additional communication overhead. In [5] a CA based cryptosystem (CAC) is proposed, where non-linearity is achieved by intermixing affine CA with non-affine transformations. However, all these block-based schemas use fixed CA rules with specific properties that constitute only a small fraction of whole CA rule space [14], so could not exploit other potential CA rules that show good qualities for encryption. Many of the proposed approaches have been broken and only some of them has been admirably tested and analyzed. Recent other works on CAs based cryptosystems can be found in [15-18,22].

4. The proposed approach

The principal objective of the proposed symmetric block-based encryption schema is to provide authenticated encryption that is CCA-secure unlike the most existing CA-based cryptosystems that provides only security against eavesdropping (CPA-secure). We aim to provide a more advanced security level against tempering, when the attacker is considered to be active and can submit its own constructed cipher-text to break the semantic security of the system. The proposed cryptosystem use keys of length 128 bit, and operate using 256bit length blocks. This can be easily extended to higher or lower values due to the scalability of the approach.

We consider in the following the class of reversible second order one-dimensional binary cellular automata with radius size 3, which have 2^{128} possible transition rule each one coded using 128 bit. A transition rule is considered as the encryption key, used to evolve plaint-text blocks to obtain cipher-text ones. The initial plain-text is decomposed into equal size blocks of size 256 bit that are ciphered sequentially. The principal details of the proposed approach are presented in the following.

4.1. Single block enciphering schema

As stated above, second-order RCAs start from two initial states C^{-1} and C^{0} (named per-initial and initial configurations, respectively) and give after n iteration two output configurations C^{n-1} and C^{n} (named per-final and final configurations, respectively). Using the same rule, the two configurations C^{-1} and C^{0} can be recovered back by running the same transition rule backward starting from C^{n-1} as per-initial configuration and C^{n} as initial one. This mechanism is used as basic construction in the proposed encryption schema: plain-text blocks are considered to be initial configurations (C^{0}), and cipher-text blocks are defined to be pre-final obtained configurations (C^{n-1}).

The pre-initial configurations (C^{-1}) are defined like the following: the first one (used for the first plain-text block) is a 256 bit value named RPC, derived from the key and a random initialization vector (IV) generated using the key expansion mechanism presented in the next section. The remaining pre-initial configurations of the following plain-text blocks are defined to be the outputted final configurations of the previously ciphered blocks.

Before iterating the transition rule on any (pre-initial/plain-text block) configurations pair, a non-linear transformation is first applied on the plain-text blocks using a Xor and an addition modulo 2^{256} with the randomized configuration RPC used as an expanded key value. This key mixing step is important to make the ciphering resistant against both linear and differential cryptanalysis techniques as shown in [21]. The transition rule is then applied to the resulting configurations during n iterations. A second key mixing step is finally performed in reversed order on the pre-final configuration to produce the ciphered block, while the final configuration is used as per-initial one for the next plain-text block enciphering. Decryption process is performed by running the same steps backward starting from the ciphered block and the obtained final configuration to reproduce the plain-text block/pre-initial configuration back. Fig. 1 illustrates a single block encryption/decryption schema using the mechanism explained above. An algorithmic description of the single bloc enciphering mechanism is detailed like the following:

```
Function EncipherBloc:
Input: Plain-text block $P_i$; pre_initial_conf ; RPC ; secret key K;
Output: Cipher-text bloc $C_i$; final_conf;
    Num_itr: = 32;
    $x$: = $((P_i \oplus \text{RPC}) + \text{RPC}) \bmod 2^{256}$;
    (pre_final_conf, final_conf) = EvolveRCA(pre_initial_conf, $x$, K, num_itr);
    $C_i$: = $((\text{pre\_final\_conf} + \text{RPC}) \bmod 2^{256}) \oplus \text{RPC}$;
End
```

The same code can perform decryption if it is inputted with the final configuration and the ciphered block, and will give the plain-text block with the pre-initial configuration as output.

4.2. Key expansion schema

To provide a randomized encryption schema, the secret key is combined during an initial phase of key expansion with a 128 bit uniformly generated random value (IV) to produce a large randomized configuration on 256 bit.

The key expansion process is described like the following: the secret key is considered as a pre-initial configuration when the random generated value IV is used as initial configuration. This pair is then evolved using the secret key K as a transition rule during m_k iteration to obtain a pair of per-final/final configurations both on 128 bit that are then combined to produce a 256 bit randomized configuration named RPC, and used as a pre-initial configuration for the encryption of the first plain-text block. Fig. 2 illustrates the proposed key expansion schema, and the following pseudo-code give the corresponding algorithmic description:

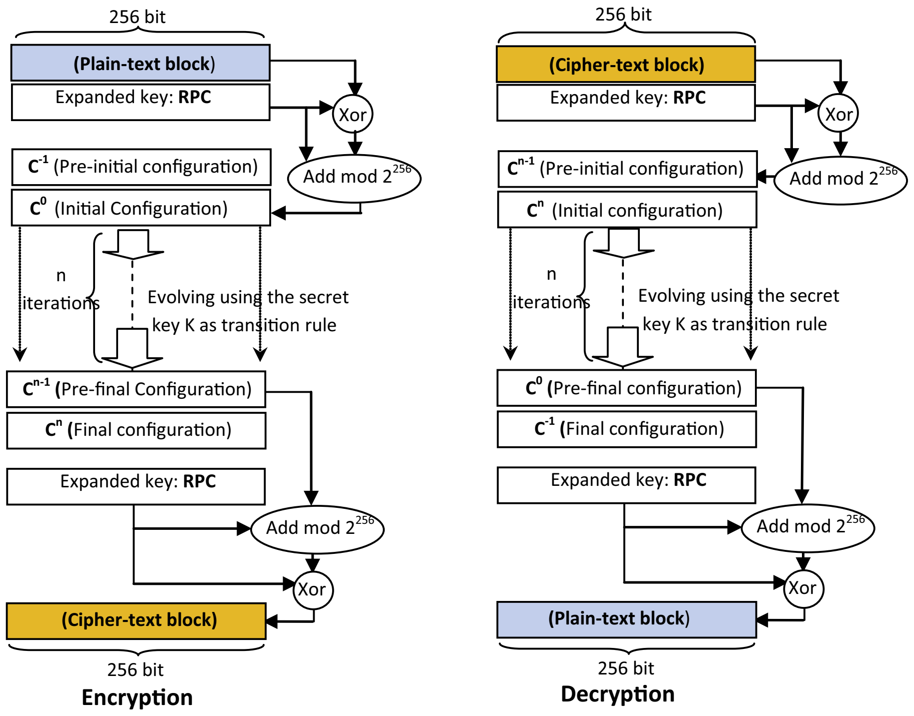
256 bit
(Plain-text block)
Expanded key: RPC
Xor
C^{-1} (Pre-initial configuration)
Add mod 2^{256}
C^0 (Initial Configuration)
n iterations
Evolving using the secret key K as transition rule
C^{n-1} (Pre-final Configuration)
C^n (Final configuration)
Expanded key: RPC
Add mod 2^{256}
Xor
(Cipher-text block)
256 bit
Encryption
256 bit
(Cipher-text block)
Expanded key: RPC
Xor
C^{n-1} (Pre-initial configuration)
Add mod 2^{256}
C^n (Initial configuration)
n iterations
Evolving using the secret key K as transition rule
C^0 (Pre-final configuration)
C^{-1} (Final configuration)
Expanded key: RPC
Add mod 2^{256}
Xor
(Plain-text block)
256 bit
Decryption

Fig. 1. Encryption /decryption schema of a single 256 bit block.

```
Function KeyExpansion:
Input: Secret key K; a random vector IV on 128 bit
Output : The expanded key RPC
    Num_itr: = 32;
    (pre_final_conf, final_conf) = EvolveRCA(K, IV, K, num_itr);
    RPC: = pre_final_conf || final_conf; //the operator || denote bit strings concatenation
End
```

The "*EvolveRCA*" call involve a function that perform second-order RCA's mechanisme on two inputs using a transition rule by the way described in Section 2.

Using a different random IV each time encryption hold will provide a randomized encryption mechanism such that the same plain-text will never be encrypted in the same way when encrypted many times. With respect to existing randomized encryption schemas, the IV is not transmitted with the cipher-text either in plain form or encryption form; it is instead diffused in the whole cipher-text using the blocks chaining schema explained in the next section.

4.3. The blocks chaining schema: details of the proposed block cipher

In order to encipher a plain-text P into a cipher-text CT, the former is first decomposed into contiguous L blocks $P_0, \ldots, P_{L-1}$ of 256 bit size each one. The obtained blocks are ciphered sequentially using the encryption chaining diagram presented in Fig. 3. The produced cipher-text is an array of L 256 bit ciphered blocks $CT_0, \ldots, CT_{L-1}$ plus an extra-data block CT_L.

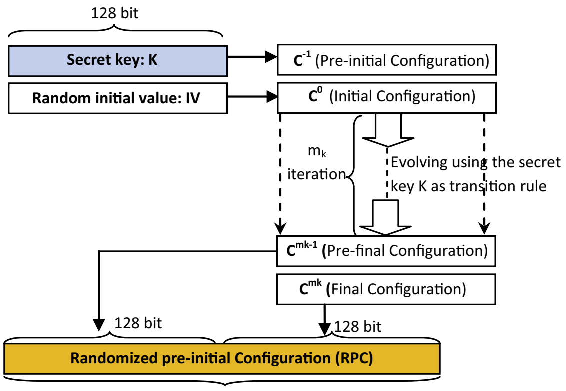
128 bit
Secret key: K
Random initial value: IV
C^-1 (Pre-initial Configuration)
C^0 (Initial Configuration)
mk iteration
Evolving using the secret key K as transition rule
C^mk-1 (Pre-final Configuration)
C^mk (Final Configuration)
128 bit
128 bit
Randomized pre-initial Configuration (RPC)

Fig. 2. Key expansion and randomization schema.

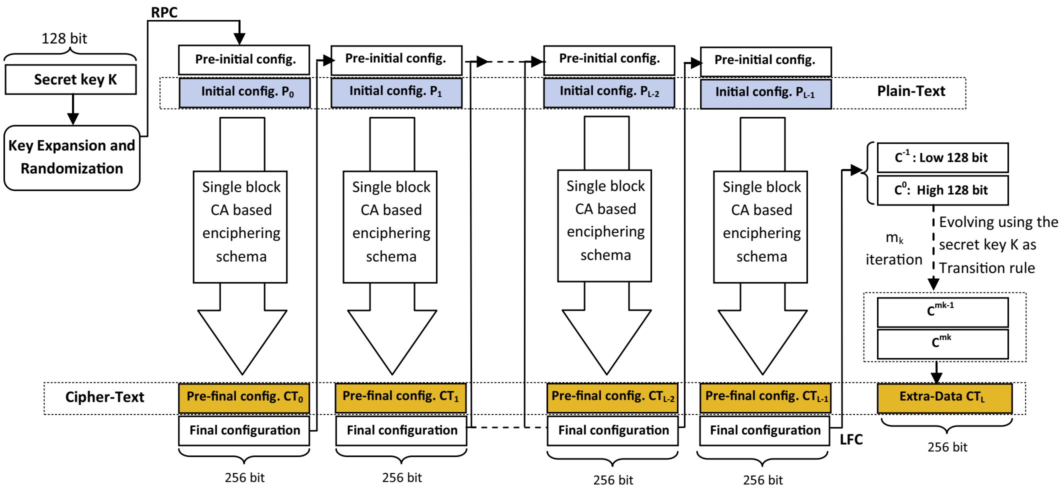

Fig. 3. Block-based encryption chaining of the proposed cryptosystem.

The key expansion process is first performed to obtain a 256 bit initial randomized configuration (RPC) depending on the secret key value and the generated IV. This configuration is used as pre-initial configuration to encipher the first plain-text block using the single block enciphering procedure presented in the Fig. 2, to produce the first cipher-text block (defined to be the pre-final configuration), and a second block (the final configuration) that will serve as pre-initial configuration to encipher the next plain-text block. This process is continued until all plain-text blocks are ciphered. Finally, the enciphering of the last block produces a supplementary configuration (the last final one named *LFC*) that is necessary to perform decryption. This last configuration is encoded and appended to the resulting cipher-text as an extra-data block using the following procedure: it is first split into two equal length 128 bit blocks that are used as a pair of configurations to be evolved using the secret key as a transition rule during m_k iteration. The resulting pre-final/final blocks are then concatenated to form the extra-data block. The resulting 256 bits block is appended to the whole obtained cipher-text. This extra-data block is necessary to ensure both randomized and authenticated aspects of the cryptosystem. No information about the key or the plain-text can be derived from it, unlike other existing randomized schemas when the IV is transmitted in clear with the cipher-text. The above block enciphering steps are illustrated algorithmically like the following:

```
Input: the secret key K; the set of L plain-text blocs $P_0, \ldots, P_{L\text{-}1}$
Output: the set of cipher-text blocs $CT_0, \ldots, CT_{L\text{-}1}$ with the extra-data block $CT_L$
IV: = Random-128bit-block;
RPC: = KeyExpansion (key,IV);
Pre_init_Conf: = RPC;
Num_itr: = 32;
For $i$: = 0 to $L-1$ do {Initial_Conf: = $P_i$;
    (pre_final_conf, Final_conf): = EncipherBlock(key, Pre_init_conf, Init_conf, num_itr);
    $C_i$: = Pre_final_conf;
    Pre_init_conf: = Final_Conf;
    }
LFC: = Final_Conf;
(Low($CT_L$),High($CT_L$)): = EvolveRCA(Low(LFC), High(LFC), K, num_itr);
End
```

4.4. Decryption and authentication schema

The decryption process is trivial and simple. The last block of the cipher-text (the extra-data block) is first decoded by running backward the key as a transition rule on its two low and high parts C^{mk-1} and C^{mk} used as pre-initial/initial configurations to obtain C^0 and C^{-1} that are combined to produce the first initial configuration used to decipher the last block. This configuration should be exactly equal to the last final one (*LFC*) obtained during enciphering for the last block. The decryption process is then executed in reverse order starting from the last ciphered block to the first one. Fig. 4 illustrate the decryption schema with the integrity verification mechanism executed at last.

Since the proposed cryptosystem ensure authenticated encryption, it provide a verification test to be performed after decryption process to tell if the cipher-text has or not been modified. The test output "valid" if the decryption succeeds to proof that integrity of the cipher-text is preserved else it output ⊥. The security of the integrity check in the proposed

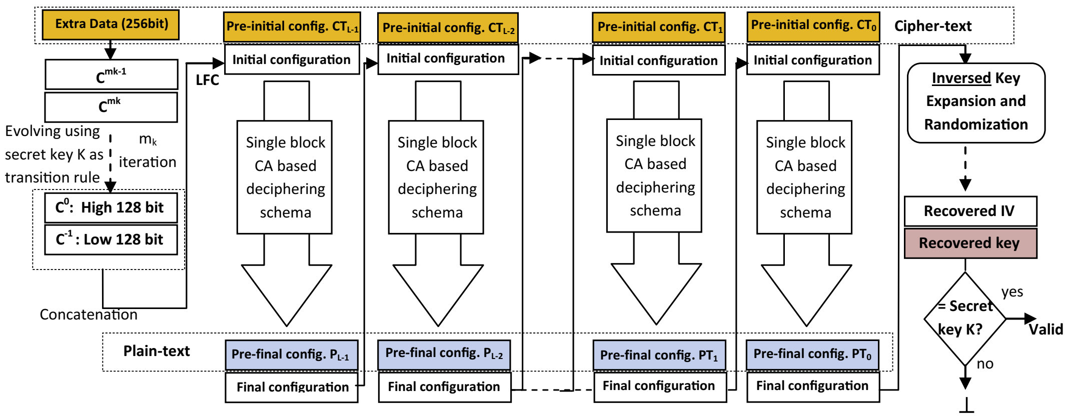

Fig. 4. Block-based decryption chaining of the proposed cryptosystem with the integrity verification mechanism.

approach is based on the bijective nature of the RCAs transformations, and its sensitivity to small initial configurations variations, while it has been noted by several researches that second order CAs can exhibit chaotic behavior especially when the neighborhood size is relatively large [20].

The integrity verification is performed like the following: after decryption of CT_0 the first cipher-text block (decrypted at the last step), we recover the final configuration that must be identical the first pre-initial configuration used during encryption (RPC) generated firstly by the key expansion schema. This recovered 256 bit configuration is split into two 128 bit blocks C^{mk-1} and C^{mk}, and the key expansion schema is run backward to recover both the random state IV (as the pre-final configuration) and the secret key K (as the final configuration). If only one cipher-text block has been modified (so incorrectly decrypted), the recovered pair of configurations will never correspond to the IV/secret key used initially and so the system will detect the modification and output $\perp$, else if the recovered key is exactly equal to the secret key, the versification successes and the cipher-text is considered to be valid. The next section gives the necessary theoretic justification of the modification's propagation from any single modified ciphered block to the final recovered key. The whole described process of decryption and integrity verification can be summarized by the following pseudo-code:

```
Input: the secret key K; the set of $L+1$ cipher-text blocs $CT_0,\ldots,CT_{L-1}$ with the extra-data block $CT_L$
Output : the set of plain-text blocs $P_0,\ldots,P_{L-1}$ with the integrity response (valid/$\perp$)
(Low(LFC), High(LFC)): = EvolveRCA(Low($CT_L$), High($CT_L$), K, num_itr);
init_conf: = LFC;
Num_itr: = 32;
For $i$: = $L-1$ down to 0 do {
pre_init_conf: = $CT_i$;
        (pre_final_conf, Final_conf): = DecipherBlock(key, Pre_init_conf, Init_conf, num_itr);
        $P_i$: = pre_final_conf;
        init_conf: = Final_conf;
        }
(Recovred_key, Recovered_IV): = ReverseKeyExpantion(Final_conf,K);
if Recovered_key = K then Return "valid"
else Return $\perp$;
End
```

The ReverseKeyEncryption subroutine called during the verification step represent the process of obtained the key and the random initialization vector IV back from the recovered RPC value, and can be described by the following algorithm:

```
Input: A configuration C on 256 bit; secret key K
Output: Recovered secret key R_Key; recovered random vector R_IV
Num_itr: = 32;
(pre_final_conf, final_conf) = EvolveRCA(Low(C), High(C), K, num_itr);
R_key: = pre_final_conf;
R_IV: = final_conf;
End
```

4.5. Reversibility, bijection and modifications propagation

Let express the RCA mechanism used in the single block enciphering schema as a transformation T_n (n is the number of iterations) defined like follow:

$$\begin{aligned} &T_n : \{0,1\}^{256} \times \{0,1\}^{256} \to \{0,1\}^{256} \times \{0,1\}^{256} \\ &x = (C^{-1}, C^0) \mapsto y = (C^{n-1}, C^n) \end{aligned} \tag{4}$$

Since RCAs are by definition reversible, the transformation T_n is reversible too. It is clear that the pair $y = (C^{n-1}, C^n)$ is unique for a fixed $x = (C^{-1}, C^0)$. For a modified value x' of x, its image $y' = T_n(x')$ should be strictly different from y since T_n is bijective.

The fact that two pairs of configurations y and y' are different (i.e. $(y = (C^{n-1}, C^n)) \neq (y' = (C^{n-1\prime}, C^{n\prime}))$) doesnot imply that $(C^{n-1} \neq C^{n-1\prime})$ and $(C^n \neq C^{n\prime})$ since the bijection is defined on pairs of configurations not on individual ones. Instead, $y \neq y'$ imply that $(C^{n-1} \neq C^{n-1\prime})$ or $(C^n \neq C^{n\prime})$.

For our proposed integrity schema to be secure, alteration of any ciphered block must propagate into all the subsequent blocks and reach the recovered value of the key in order be detected by the integrity verification mechanism. The following theorem shown in [14] justify the choices performed when designing the proposed cryptosystem, and can explain how alterations and modifications of the cipher-text propagate to alter the recovered key value.

Theorem 1. *Let consider a reversible second-order one-dimensional cellular automaton that start form a pair of pre-initial/initial configurations C^{-1}, C^0 to produce a pair of pre-final/final configurations C^{n-1}, C^n after n iteration.*

(1) *Any modification of the pre-initial configuration C^{-1} provokes always the modification of both pre-final/final configurations C^{n-1} and C^n.*
(2) *Any modification of the initial configuration C^0 provokes always the modification of the final configuration C^n, but the pre-final configuration C^{n-1} may remain unchanged with a small probability value.*

As a consequence of the theorem, the authentication schema can fail to detect some modifications if they affect only initial configurations since they will not propagate to both pre-final/final configurations and so integrity may not be preserved. For this reason, we have chosen that the ciphered blocks be defined by the pre-final configurations instead of the final ones. By this way, they will be is used as a pre-initial configurations during decryption, and this will force any alteration to always propagate according to the first assumption of the theorem. The system will then be able to detect any alteration and so to preserve cipher-text integrity.

The same problem of alteration propagation can occur during key expansion process. According to the theorem, the final configuration is always altered whatever the modification affect either pre-initial/initial configurations or only a single one of them. This justify the fact that the secret key is used as the pre-initial configuration not the initial one during key expansion process to make the recovered key defined by the final reverse expansion configurations absolutely affected by any possible alteration.

Based on Theorem 1, a formal proof of the integrity preserving aspect by the proposed encryption schema is presented in Section 5.1. Security analysis and performances evaluations are presented in the remaining sections.

5. Security analysis of the proposed block cipher

In the following, we show that the proposed block cipher provides authenticated encryption. According to [1], a given cryptosystem provides authenticated encryption if and only if:

(1) The system provide cipher-text integrity: attackers cannot create new cipher-texts that decrypt properly.
(2) The system provide semantic security under CPA attacks (CPA-secure);

In the following, we show first that the proposed approach satisfy the integrity criterion by proofing that any cipher-text alteration will result in an alteration of the recovered key value and so the authentication system will output $\perp$ (alteration detected). Then, the second criterion is demonstrated experimentally by showing that the proposed block cipher satisfy the conditions of confusion/diffusion, avalanche criterion and pseudo-random statistical distribution.

5.1. Proof of the cipher-text integrity preserving

We consider a plain-text P constituted of L blocks of 256 bit each one, and CT the corresponding obtained cipher-text. Let consider first the following block sets:

– $\{P_0,P_1,\ldots,P_{L-1}\}$ the set of all the plain-text blocks.
– $\{CT_0,CT_1,\ldots,CT_L\}$ the set of all the cipher-text blocks.
– $FD = \{FD_0,FD_1,\ldots,FD_{L-1}\}$ the set of all the final configurations obtained during decryption.
– $FE = \{FE_0,FE_1,\ldots,FE_{L-1}\}$ the set of all the final configurations obtained during encryption.

We consider the notation $T_n(x,y)(0)$ and $T_n(x,y)(1)$ to express the first and the second element of the pair obtained when applying the transformation T_n to (x,y) (i.e. $T_n(x,y) = (T_n(x,y)(0), T_n(x,y)(1))$. We define also the non-linear key mixing transformation used during the single blocks enciphering schema by:

$$\begin{aligned} &f : \{0,1\}^{256} \to \{0,1\}^{256} \\ &x \mapsto f(x) = ((x \oplus \mathrm{RPC}) + \mathrm{RPC})\ mod\ 2^{256} \end{aligned} \tag{5}$$

f is a reversible bijection and its inverse f^{-1} can be trivially defined. The encryption function E can then be defined for each plain-text block by:

$$\begin{aligned} &E : \{0,1\}^{256} \to \{0,1\}^{256} \\ &P_i \mapsto CT_i = \begin{cases} f^{-1}(T_n(FE_{i-1},f(P_i))(0)) & \text{if } i > 0 \\ f^{-1}(T_n(\mathrm{RPC},f(P_i))(0)) & \text{if } i = 0 \end{cases} \end{aligned} \tag{6}$$

The decryption function for a given cipher-text is defined by:

$$D : \{0,1\}^{256} \to \{0,1\}^{256}$$
$$CT_i \mapsto P_i = \begin{cases} f^{-1}(T_n(f(CT_i), FD_{i-1})(0)) & \text{if } i > 0 \\ f^{-1}(T_n(f(CT_i), LFC)(0)) & \text{if } i = 0 \end{cases} \tag{7}$$

It is clear from the described encryption/decryption schema that if there is no cipher-text modification (no tampering attack), the following equalities always hold:

$$FD_i = FE_{L-i-2} \quad \forall 0 \leqslant i \leqslant L-2, \quad \text{and} \quad FD_{L-1} = \mathrm{RPC} \tag{8}$$

It can also be noticed from Fig. 5 that:

$$FD_i = T_n(f(CT_i), FD_{i-1})(1) \quad \forall 0 \leqslant i \leqslant L-1, \quad \text{and} \quad FD_{-1} = LFC \tag{9}$$

Let express the recovered value of RPC (named R_RPC) computed during decryption process, and serving to deduce the recovered key used for authentication:

$$\begin{aligned} \mathrm{R\_RPC} &= FD_{L-1} = T_n(f(CT_0), FD_{L-2})(1) \\ &= T_n(f(CT_0), T_n(f(CT_1), FD_{L-3})(1))(1) \\ &\quad \ldots\ldots\ldots\ldots\ldots\ldots \\ &= T_n(f(CT_0), T_n(f(CT_1) \ldots\ldots\ldots T_n(f(CT_{j-1}), T_n(f(CT_j), FD_{L-2-j})(1))(1) \ldots\ldots\ldots\ldots\ldots)(1))(1) \\ &\quad \forall 0 \leqslant j \leqslant L-1 \text{ if we consider } FD_{-1} = LFC \end{aligned} \tag{10}$$

We suppose now that a given attack has modified the value of a cipher-text block CT_j to CT'_j. Let show that the recovered value R_RPC' after the attack will always be different than the expected value of R_RPC equal to RPC.

Since T_n is a bijection, modifying CT_j to CT'_j will always produce a value of $T_n(CT'_j, FD_{L-2-j})(1)$ that is different than $T_n(CT_j, FD_{L-2-j})(1)$ since CT_j is a pre-initial configuration (according to Theorem 1 part 1).

$$\begin{aligned} & T_n(CT_j, FD_{L-2-j})(1) \neq T_n(CT'_j, FD_{L-2-j})(1) \\ & \Rightarrow T_n(CT_{j-1}, T_n(CT_j, FD_{L-2-j})(1))(1) \neq T_n(CT_{j-1}, T_n(CT'_j, FD_{L-2-j})(1))(1) \text{ [Theorem 1, part 2]} \\ & \Rightarrow \ldots \\ & \Rightarrow T_n(CT_0, T_n(CT_1, \ldots, T_n(CT_{j-1}, T_n(\boldsymbol{CT}_j, FD_{L-2-j})(1))(1) \ldots\ldots\ldots)(1))(1) \neq \\ & \qquad T_n(CT_0, T_n(CT_1, \ldots, T_n(CT_{j-1}, T_n(\boldsymbol{CT}'_j, FD_{L-2-j})(1))(1) \ldots\ldots\ldots)(1))(1) \text{ [Eq. 10. Theorem 1, part 2]} \\ & \Rightarrow \mathrm{R\_RPC'} \neq \mathrm{RPC} \end{aligned}$$

We know that the recovered key and IV values (R_key, R_IV) are the results of T_{mk}(Low(R_RPK'), High(R_RPK')) according to the inversed key expansion process. Since the recovered key serving for authentication is recovered as a final configuration, it will be altered by any modification of either low part or high part of R_RPK' (since R_RPK'<>RPC) according to Theorem 1. So recovered r-key will always be different from the secret key K and the attack is then detected.

If we suppose now that instead of modifying a cipher-text block CT_j, the attacker alters the appended extra data CT_L. The same reasoning permit to proof that integrity is also preserved:

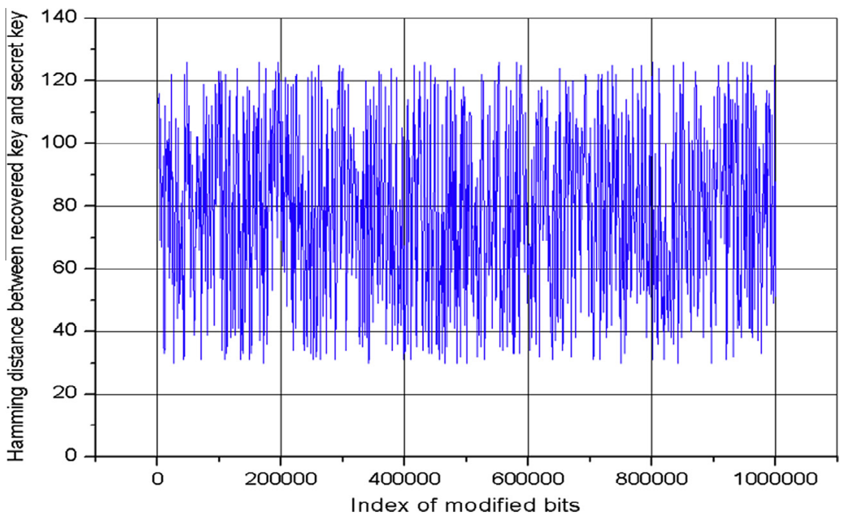

Fig. 5. Distance between recovered keys and the secret key for 10^6 different altered cipher-texts.

$$\begin{aligned} \text{R\_RPC} &= T_n(f(CT_0), T_n(f(CT_1) \ldots\ldots\ldots T_n(f(CT_{j-1}), T_n(f(CT_j), FD_{L-2-j})(1))(1) \ldots\ldots\ldots\ldots)(1))(1) \text{ [Eq. 10]} \\ &= T_n(f(CT_0), T_n(f(CT_1) \ldots\ldots\ldots T_n(f(CT_{L-2}), T_n(f(CT_{L-1}), LFC)(1))(1) \ldots\ldots\ldots)(1))(1) \end{aligned} \tag{11}$$

When the Extra data CT_L is modified, the recomputed *LFC* is always modified since it is obtained by a direct transformation $T_{mk}(\text{low}(CT_L), \text{Hi}(CT_L))$: if R_RPC' is the recovered RPC value when CT_L is modified, we proof in the following that R_RPC' differ from RPC, and the alteration is then detected by the system:

$$\begin{aligned} & CT_L \neq CT'_L \\ & \Rightarrow LFC' \neq LFC \text{ (recovered LFC is altered)} \\ & \Rightarrow T_n(f(CT_{L-1}), LFC')(1) \neq T_n(f(CT_{L-1}), LFC)(1) \text{ (Theorem 1, part 1)} \\ & \Rightarrow T_n(f(CT_{L-2}), T_n(f(CT_{L-1}), LFC')(1))(1) \neq T_n(f(CT_{L-2}), T_n(f(CT_{L-1}), LFC)(1))(1) \\ & \ldots \\ & \Rightarrow T_n(f(CT_0), T_n(f(CT_1) \ldots\ldots\ldots T_n(f(CT_{L-2}), T_n(f(CT_{L-1}), \boldsymbol{LFC'})(1))(1) \ldots\ldots\ldots)(1))(1) \\ & \neq T_n(f(CT_0), T_n(f(CT_1) \ldots\ldots\ldots T_n(f(CT_{L-2}), T_n(f(CT_{L-1}), \boldsymbol{LFC})(1))(1) \ldots\ldots\ldots)(1))(1) \text{ (Eq. 11)} \\ & \Rightarrow \text{R\_RPC}' \neq \text{RPC} \end{aligned}$$

Since recovered R_RPC after CT_L alteration is different than the expected RPC, the system will detects the alteration propagated to the recovered key in the same way as explained above. We have then shown that altering any cipher-text block will alter the recovered value of RPC, and the alteration is always propagated to the recovered key and finally detected by the verification test. The proposed system so preserves the integrity and as a result provide authentication.

In order to illustrate experimentally the capacity of the system to detect alterations, the following experiments were performed: a random file of size 100 Mb (that can be an image, a video or anything else) is ciphered using a given key K. Then, for each one of 10^6 randomly selected bits from the resulting cipher-text, the value is flipped and the resulting modified cipher-text is decrypted to compute the recovered key serving for authentication. Fig. 5 show a plot of the Hamming distances between the recovered keys of the modified cipher-texts and the expected legal value of the key. It is clear that for any elementary small modification of the cipher-text, the hamming distance is not null, and so the authentication process always succeed to detect the alteration.

5.2. Robustness of the block cipher against chosen plain text attacks

There is no one generic theoretic standard way to proof for a given blocks cipher that is CPA-secure. Instead, the proof can be based on some properties that must be verified in order to show that the block cipher is robust against most common attacks such like differential and linear cryptanalysis. These properties are easy to check statistically and are generally sufficient to ensure robustness and security of a proposed cryptosystem. Basic properties to be checked for a block based cryptosystem are avalanche property, confusion/diffusion and pseudo-randomness of the cipher-text.

5.2.1. Confusion/diffusion and avalanche citerions

Randomized property of the cryptosystem implies always that it is CPA-secure since no plain-text information is preserved between two successive encryptions. However, we show in the following that the proposed cryptosystem satisfy confusion/diffusion and avalanches criterions, as a result is remain highly secure even if the random initialization vector is predictable or constant.

The avalanche effect is evident if, when an input is changed slightly (for example, flipping a single bit) the output changes significantly (e.g., half the output bits flip). In the case of quality block ciphers, such a small change in either the key or the plaintext should cause a drastic change in the cipher-text. Confusion refers to making the relationship between the cipher-text and the symmetric key as complex and involved as possible; diffusion refers to making the relationship between the plain-text and the cipher-text as complex and involved as possible. These two criterions are implicitly guaranteed by the satisfaction of avalanche criterion, and if the proposed block-cipher satisfies such conditions, it has been proven that it can resist both linear and differential attacks.

According to the proposed schema description, it is sufficient to verify the avalanche criterion with respect to a single block enciphering in-order to deduce it satisfaction for any arbitrary long cipher-texts. We performed statistical testing using 10,000 generated single plain-text block and 10,000 random generated keys. For each pair of key/plain-block, a one bit change is performed, and then the percentage of changed bits in the resulting ciphered block is measured. To disable the randomized aspect of the approach, and so reflect the real influence of key and plain-text modifications, the value of IV is fixed during experiments. Fig. 6 shows average percentage of bits changed in the cipher-text with respect to the number of iterations n of the T_n transformation when varying the values of the plain-text with key and IV fixed (first experiment), when varying the value of the key with IV and plain-text fixed (second experiment), and when varying the value of the random value IV with key and plain-text fixed (third experiment). It is clear from the results that complementing one bit in the key has provide a changing rate of approximately 50% (the optimal value) in the cipher-text when the number of iterations of the T_n transformation is greater than 25. The minimal number of iterations required is approximately 29 to obtain an equivalent rate with one bit changing in the plain-text or in the IV value. As a result, the number of iterations n is chooses to be

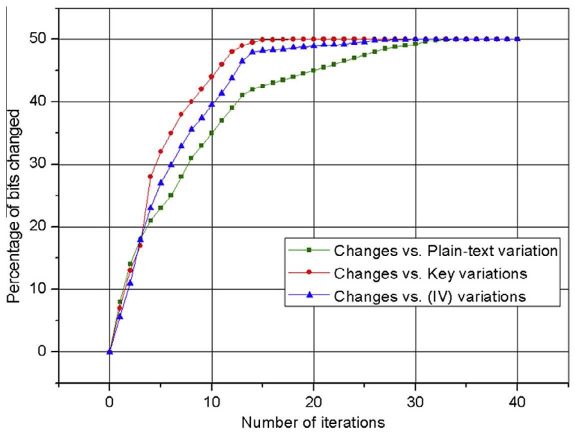

Fig. 6. Average bit changed in cipher-text with respect to changes in the key, plain-text, and the IV value.

Table 1
Averaged results obtained with Diehard test using 1000 different cipher-texts.

Test name	Averaged *P*-value	Interpretation
Birthday spacings	0.739346	Pass
Overlapping permutation	0.973987	Pass
Rank test 31 × 31	0.877621	Pass
Rank test 32 × 32	0.515578	Pass
Monkey DNA	0.584985	Pass
Count-the-1's test for specific bytes	0.444859	Pass
Parking lot	0.129446	Pass
Minimum distance	0.357143	Pass
Rundom sphere	0.321231	Pass
The sqeeze test	0.646705	Pass
Overlapping sums	0.449443	Pass
The runs-up test, down test-	0.938687 0.391889	Pass
Craps-no of wins, thrwos/game-	0.908809 0.570591	Pass
Rank test	0.902134	Pass
Monkey 20 bits per word	0.856931	Pass
Monkey OPSO, OQSO	0.978510	Pass

$n = 32$ iteration, with same value fixed for $m_k = 32$ representing the number of iterations used during key expansion and extra-data computation. Using these parameter values, the cryptosystem can be considered CPA-secure even if the randomized IV is predictable or constant. It is clear also that randomized aspect is guaranteed since the cryptosystem is very sensitive to the random IV variations when n is higher than 32 iteration.

5.2.2. Pseudo-randomness of cipher-text

According to Shannon's theory, a good encryption scheme should have excellent performance in resisting statistical attack. The generated cipher-text from any given plain-text having any statistical distribution must be indistinguishable from a random sequence to insure security against chosen plain-text attacks. The indistinguish ability of the sequences produced by the proposed cryptosystem is shown experimentally by generating a number of different cipher-texts corresponding to a set of plain-texts having different types and sizes (images, text, and video). Each obtained cipher-text is analyzed using the Diehard and ENT statistical Tests batteries [10] to evaluate the corresponding randomness degree. In the case of diehard battery results, the averaged p-value for each test is taken over 1000 different generated cipher-text, and the results are reported in the Table 1. The Table 2 list average values of ENT tests values obtained over same cipher-text generated data set.

It is clear from the obtained experimental results that the cryptosystem's outputs pass all statistical testes provided by Diehard and ENT suits which mean that cipher-texts cannot be polynomially distinguishable from uniform random sequences and security of the cryptosystem is then assured against any statistical chosen plain text attacks.

6. Encryption speed analysis

As stated in the introduction, the main advantage of the proposed authentication encryption schema is the fact that only one-pass is needed on each plain-text block to ensure both ciphering and authenticating. Unlike existing standards that

Table 2
Averaged results obtained with ENT Test using 1000 different cipher-texts.

Test	Value	Norm
Entropy	7.999012	Max = 8.0
Arithmetic mean value of data bytes	127.356	127.5 = random
Monte Carlo	3.1491040 (error = 0.24%)	π value
Serial correlation coefficient	−0.00192	0.0
Optimum compression	0.00	0.0

Table 3
Speed performances comparison with existing AE schemas.

Authenticated encryption schema	Speed (MB/s)
AES/CCM	61
AES/EAX	61
AES/CMAC	109
AES/GCM	108
AES/OCB	129
Proposed approach	119

require at least two passes (one for ciphering and a second pass provide integrity check), the proposed approach incorporate the two processes in a single pass since no hash or checksum are computed and only cellular automaton properties are sufficient to ensure both security aspects.

It is clear from Table 3 that the proposed approach can provide very high encryption/decryption rates with respect to existing schemas. Furthermore, since cellular automatons are natively parallelizable, a hardware implementation of the proposed cryptosystem is very efficient and can achieve higher performances than software implementation does. The Table 3 lists a comparison of different enciphering rates (deciphering speed is equivalent to the enciphering one) between the proposed approach and existing standards using AES encryption [23]. We have implemented our approach using Delphi 6, when the elementary cellular automaton subroutines has been coded using assembly MMX instructions set to provide high computation performances.

7. Conclusion

The present paper develops an authenticated and randomized block cipher through the use of reversible second order cellular automata. Dynamical properties and reversibility of this CA's class permit to construct a one-pass authenticated encryption schema that can ensure security of the cryptosystem against tampering and active chosen cipher-text attacks. We showed that cipher-text integrity is always preserved even for small variations in any ciphered block, when statistical distribution of the generated cipher-text is indistinguishable from random data, which ensure security against statistical, linear and differential cryptanalysis techniques. The implicit randomized aspect of the proposed cryptosystem is another advantage that eliminates the threat of chosen plain-text attack. The proposed schema can easily be adapted to be secure against replay attacks by replacing the random value IV by an incremental nonce to synchronies the sender and the recipient.

Even with a non-optimized code, the implementation of the proposed approach shows very competitive speed performances with respect to most of existing authenticated encryption schemas. We assume a hardware implementation of the cryptosystem can provide mush enhanced performances due to the inherent parallelism nature of the cellular automatons.

References

[1] Bellare M, Namprempre C. Authenticated Encryption: Relations among notions and analysis of the generic composition paradigm. In: Okamoto T, editor. Extended Abstract in Advances in Cryptology: Asiacrypt 2000 Proceedings. Lecture Notes in Computer Science. Springer-Verlag; 2000.

[2] Information technology – security techniques – authenticated encryption, 19772, 2009. ISO/IEC. Retrieved at March 12, 2013.

[3] Wolfram S. Random sequence generation by cellular automata. Adv. Appl. Math. 1986;7(2):123–69.

[4] Seredynski F, Bouvry P, Zomaya AY. Cellular automata computation and secret key cryptography. Parallel Comput. 2004;30:753–66.

[5] Sen S, Shaw C, Chowdhuri DR, Ganguly N, Pal Chaudhuri P. Cellular automata based cryptosystem (CAC). In: ACRI-2002. LNCS, 2513. Springer-Verlag; 2002. p. 303–14.

[6] M. Szaban, F. Seredynski, P. Bouvry, Collective behavior of rules for cellular automata- based stream ciphers, in: IEEE Congress of Evolutionary Computation, CEC, Jul. 16–21, 2006, pp. 179–183.

[7] Hortensius PD, McLeod RD, Card HC. Parallel random number generation for VLSI systems using cellular automata. IEEE Trans. Comput. 1989;38:1466–73.

[8] Nandi S, Kar BK, Chaudhuri PP. Theory and applications of cellular automata in cryptography. IEEE Trans. Comput. December 1994;43:1346–57.

[9] Tomassini M, Perrenoud M. Stream ciphers with one- and two-dimensional cellular automata. In: Schoenauer M et al., editors. Parallel Problem Solving from Nature – PPSN VI. LNCS, 1917. Springer; 2000. p. 722–31.

[10] J. Soto, Statistical testing of random number generators, in: Proceedings of the 22nd National Information Systems Security Conference, Crystal City, Virginia, October 1999.
[11] J. Kari, Cryptosystems based on reversible cellular automata, University of Turku, Finland, Preprint, April 1992, http://users.utu.fi/jkari/CACryptoScanned.pdf.
[12] C. Zhang, L. Hua, Reconfigurable pipelined cellular automata array for cryptography, in: Circuits and Systems and West Sino Expositions, IEEE International Conference, vol. 2, Jun. 29–Jul. 1, 2002, pp. 1213–1217.
[13] Seredynski F, Pienkosz K, Bouvry P. Reversible cellular automata based encryption. In: NPC '04. LNCS, 3222. Springer-Verlag; 2004. p. 411–8.
[14] Toffoli T, Margolus N. Invertible cellular automata: a review. Physica D 1990;45:229–53.
[15] Ray A, Das D. Encryption algorithm for block ciphers based on programmable cellular automata. Inf. Process. Manage. 2010:269–75.
[16] Tripathy S, Nandi S. LCASE: lightweight cellular automata-based symmetric-key encryption. Int. J. Network Secur. 2009;8(2):243–52.
[17] Kumaravel A, Meetei O. An application of non-uniform cellular automata for efficient cryptography. Indian J. Sci. Technol. 2013;6(5S):4560–6.
[18] Anghelescu P. Security of telemedical applications over the internet using programmable cellular automata. Int. J. Intell. Comput. Res. 2012;3(1/2):245–51. ISSN: 2042-4655.
[19] Margolus N. Physics-like models of computation. Physica D 1984;10:81–95. Reprinted in S. Wolfram, Theory and applications of cellular automata, in: Advanced Series on Complex Systems, vol. 1, World Scientific, pp. 232–246, 1986.
[20] Wolfram S. A New Kind of Science, Wolfram Media. World Scientific; 2002. pp. 437–440, ISBN: 1-57955-008-8.
[21] Mukhopadhyay D, Roy Chowdhury D. Key mixing in block cipher through addition modulo 2n. Int. J. Comput. Math. Sci. Appl. 2008;1:211–24.
[22] Abdoa AA et al. A cryptosystem based on elementary cellular automata. Commun. Nonlinear Sci. Numer. Simul. January 2013;18(1):136–47.
[23] W. Dai, Crypto++ 5.6.0 Benchmarks, <http://www.cryptopp.com/benchmarks.html>.