Security of Telemedical Applications over the Internet using Programmable Cellular Automata

Petre Anghelescu
University of Pitesti, Faculty of Electronics, Communications and Computers, Str. Targu din Vale, No. 1, 110040, Pitesti, Arges, Romania

Abstract

As the increasing number of telemedical applications over the Internet using mobile devices (PDAs, Tablet PC, various medical devices, and so on), for access of medical data in terms of telemedicine and remote patient telemonitoring, the necessity of new powerful encryption techniques becomes a crucial issue. In this paper is presented a fully functional software encryption system based on a combination of a one-dimensional hybrid cellular automaton (CA) and a programmable cellular automaton (PCA) used for securing of medical data sent over the internet. The experimental results are based on the powerful of cellular automata based encryption systems, already validated in my previously works. The proposed algorithm belongs to the class of symmetric key systems.

1. Introduction

The increasing number of telemedical applications using the Internet, make the need for data security and privacy to become a strong necessity. Furthermore, as the recently introduced mobile devices (PDAs, Tablet PC, various medical devices, and so on), for access of medical data in terms of telemedicine and remote patient telemonitoring, the necessity of new powerful encryption technique becomes a crucial issue. Also, nowadays everybody knows how to make use of a packet sniffer and probably, if the channel is not already secured, find some sensitive information. By placing such a tool "under the skin" of your network in a couple of sensitive zones, someone could intercept, corrupt and tamper the medical records.

Accordingly, cryptography has become more important in data security. Also, in the recent years, researchers have remarked similarities between bio-inspired systems – as cellular automata (CA), chaos and cryptography. Several of the CA features can be correlated with the cryptographic properties.

The essence of the theoretical and practical efforts which are done in this new field is represented by the idea that CA based cryptosystems are capable to have similar performances regarding the classic methods based on computational techniques. Also, I have to mention that there is not a method of ideal codification at present and this is an additional reason regarding the suitability of the elaboration of new methods and encryption systems. The goal is to develop an alternative cryptogram based on a combination of cellular automata (CA) and programmable cellular automata (PCA), in which several CA technologies such as Wolfram approach [1], transform-based approach [2] and five evolution rules are combined in some way to form a one cryptogram.

In this paper we approach the advanced encryption method based on CA and PCA on the particularly telemedical application. This consists in: a telemedical pilot network including medical databases, different mobile devices located at the patients that communicate a large amount of data over the wireless ad-hoc networks (as depicted in Figure 1).

![Figure 1. The architecture and relationships of the telemedical application.](image)

This paper is organized as follows. The next section provides a brief introduction to CA and PCA concepts. Section 3 is about the proposed CA and PCA encryption algorithm. In this section is presented the structure of the entire cellular automata based encryption system. Testing and experimental results are presented in section 4. Here, the proposed encryption method was tested on a particular application of medical data sent over the internet. Section 5 points out the conclusions and future research directions of the paper.
2. Basics on CA and PCA

CA is a bio-inspired paradigm which has been attracting the attention of researchers in theoretical computer science as well as of scientists in other disciplines including information security. PCA is a modified CA structure including switches in order to allow the self-organizing of the cellular structure.

The CA, first introduced by J. v. Neumann [3] and further popularized by S. Wolfram [1], are computational models that can perform complex computation with only local information. The simple structure of CA has attracted researchers from different fields of interests and has undergone rigorous theoretical and experimental analysis.

CA are the mathematical idealizations of physical systems in which space and time are discrete, and each cell can assume the value either 0 or 1. CA are models for physical systems where space and time are discrete and interactions are only local. In the CA theory, there are two classic types of neighbourhoods: the Moore neighbourhood that comprises 3 cells for one-dimensional CA and 9 cells for two-dimensional CA (Fig. 2a); the von Neumann neighbourhood with 3 cells for one-dimensional CA and 5 cells for two-dimensional CA (Fig. 2b).

The CA cells evolve in discrete time steps in accordance with some deterministic rule(s). The state of the cell depends only on local neighbours (including the cell itself as neighbour). In effect, every cell essentially comprises of a memory element (D flip-flop) and a combinational logic (CL) that generates the next-state of the cell.

Combination logic is called the “rule” of the CA. When all cells evolve according to the same logic function, the CA is called uniform, otherwise it is called hybrid. If the rule of a CA involves only XOR logic, then it is called a linear rule. Rules involving XNOR logic are referred to as complement rules. A CA with all its cells having linear rules is called a linear CA. A CA having a combination of linear (XOR) and complement (XNOR) rules is called additive CA. The next-state function describing a rule for a three neighborhood CA cell can be expressed as follows:

\[ a_i(t+1) = f[a_i(t), a_{i+1}(t), a_{i-1}(t)] \quad (1) \]

where \( i \) is the position of an individual cell in one-dimensional array of cells, \( t \) is the time step, and \( f \) is the rule of CA.

In case of 1-D, 3-neighborhood, 2-states – 0 and 1– the number of all possible uniform CA rules is \( 2^8 = 256 \). These rules are enumerated using Wolfram’s naming convention [4] from rule number 0 to rule number 255 and can be represented by a 3-variable Boolean function. Under this perspective an example of five rules, which will be used in latter sections, are considered fundamental and are presented in Table 1.

![Figure 2. CA classical neighborhoods](image)

(a) - Moore Neighborhood, (b) - von Neumann Neighborhood

Table 1. Rules used for the next state of the CA and PCA cells.

<table>
<thead>
<tr>
<th>Rules</th>
<th>000</th>
<th>001</th>
<th>010</th>
<th>011</th>
<th>100</th>
<th>101</th>
<th>110</th>
<th>111</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>90</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>100</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>150</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>51</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>60</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>102</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Based on the statistical properties of the three neighborhoods, CA can be classified into four categories [4]:

**Class 1:** CAs that evolve to a homogeneous final global state.

**Class 2:** CAs in which each state lies in some cycle (periodic behavior).

**Class 3:** CAs that exhibit chaotic or pseudo-random behavior. This model is suitable for pseudo-random pattern generation.

**Class 4:** CAs having complicated localized and propagating structures. CA of this class are capable of universal computation.

The four CA classes are more or less self-explanatory. However, the temporally periodic evolution of a spatially disordered configuration (“spatial chaos”) falls into class 2. In class 3, “chaotic” is taken to mean “spatio-temporal disorder”, in the sense that the number of distinct space-time patches growth exponentially with the linear size of the patch along both the spatial and temporal axes. It seems that CA from class 4 could be capable of universal computation, so could implement a universal Turing machine. Class 4 was defined by the presence of spatially isolated transients of many shapes evolving for arbitrarily long times.

The PCA was firstly introduced in [5], where the CL of each cell is not fixed but controlled by a number of control signals such that different functions (i.e. rules) can be implemented on the same CA structure. As the matter of fact, PCA are essentially a modified CA structure. It employs some control signals on a CA structure. By specifying certain values of control signals at run time, a PCA can implement various functions dynamically in terms of different rules.
For example, using such a cell structure as in Figure 3, all possible non-complemented additive rules can be achieved through the combinations of the control signals of C1, C2, C3 switches.

Figure 3. Three neighbors PCA: a) non-complemented additive rules, b) complemented additive rules CA cell.

In practice, a control program, stored in a memory, can be employed to activate the switches. For example, the state 1 or 0 of the bit i-th of a memory word either opens or closes the switch that controls that cell. Basically, such a structure is referred as a programmable CA (PCA). A huge flexibility into this programmable structure can be introduced via control signals in CL. For an n-cell CA structure can be used for implementing 2^n CA configurations.

The inventors of the cellular automata, S. Ulam and John von Neumann, wanted to show that the most elementary rules could lead to too complex consequences for someone to guess them. Although the local behaviour of the automatons is perfectly transparent, the overall behaviour it seems that will never be completely understood. The very large phenomenology of the cellular automata model, its apparently big complexity and massive parallelism (however, this parallelism, when emulated in software or in sequential hardware, disappears) offer a good basis for applications in cryptography.

3. CA and PCA based encryption algorithm

In this section is presented the proposed cryptosystem in order to encrypt medical data sent over the Internet.

In the block cipher scheme, one 8-bit message block is enciphered by one enciphering function. The PCA control signals are activated with the help of the signals that are generated with the memory rules file or CA PRNG. The enciphering function has five fundamental transformations FTs (PCA = 5) to
operate on 8-bit data. It is obvious that for high security applications, more fundamental transformations are to be used.

The block cipher (decipher) procedure can be defined as follows:

1. Load the PCA1 with one byte plaintext (ciphertext) from I/O. The initial block of the message is the initial state of the PCA. The global configuration of the PCA5 represents the encrypted message.
2. Load a rule configuration control word from CA-PRNG or FPGA memory file into the PCA.
3. Run the PCA (1, 2, 3, 4 and 5) for 1 … 7 cycles (in the next paragraph I will explain why must have 1…7 cycles).
4. Repeat steps 2 and 3 for five times.
5. Send one byte ciphertext (plaintext) to I/O (from the PCA5). If not end of the plaintext (ciphertext) go to step 1. Otherwise, stop the process.

3.1. PCA array

The PCA is a null boundary CA configured with the rules 51, 60 and 102 (the rules are presented in Table 1 and in equations 2, 3 and 4 and its state-transition diagram consists of equal circles of even length (as is presented in Figure 5).

![Figure 5. The state transitions diagram of an 8-cell null boundary non-maximum length PCA configured with rules: 51, 102, 102, 102, 51, 102, 51 and 102.](image)

In the state-transition graph, each cycle has a length 8. This phenomenon is a basic requirement of the enciphering scheme. For example, if we employ this CA as our enciphering function and define a plaintext as its original state, it goes to its intermediate state after one to seven cycles. This is the process of enciphering. After running other seven to one cycles, the intermediate state returns back to its original state, so the ciphertext is deciphered into plaintext. This is the process of deciphering. The rules with 8-cycle length are presented in detail in my previous paper [8].

In concordance with the (P)CA theory, a single basic programmable CA cell was designed as is depicted in Figure 6.

![Figure 6. The structure of a single PCA cell.](image)

3.2. Rules generator: CA-PRNG or memory files

The rules generator is designed to provide encryption rules for the PCA encryption system. The rule generator consists of a modulus 156 counter and memory file with 156 lines with rules or a CA-PRNG used as a key stream generator. If we use a CA as a key stream generator, CA pseudo-random number generator (PRNG) that combines in some way two rules 90 and 150 (the rules are presented in Table 1 and in equations 2 and 3), to provide the key sequence [7]. The operation of CA-PRNG can be represented by a state-transition graph. Each node of the transition graph represents one of the possible states of the CA. The direct edges of the graph correspond to a single time step transition of the automata. Figure 7 shows the state transitions graph of a 4-bit hybrid null boundary condition CA with rules <90, 150, 90, and 150>. It generates a 4-bit pseudo-random pattern.

![Figure 7. The state transition diagram of an 4-cell null boundary maximum-length CA configured with rules: 90, 150, 90 and 150.](image)

It consists of a D flip-flop and a logic combinational circuit (LCC). The LCC includes multiplexers and XNOR logic gates to implement the rules of CA and to control the loading of data and operation of the CA. When the load control signal (LoadData) is “logic 1", data is loaded into D flip-flop. When LoadData is “logic 0", data is run into the cell according to the rules applied to the rule control signals (S1, S0) and the states of neighborhoods. After an established number of cycles (1 to 7), the data on the Q output of the flip-flop is sent out and new data is loaded in. In this research we decide to connect together eight cells in order to build an 8-cell PCA.
five PCA (as is presented in Figure 4). When the encryption process begins, rules are read out in sequence and sent to the PCA arrays. An example of applying the evolution CA rules stored in memory files or generated with the help of CA-PRNG is presented in Table 2.

Table 2. An example of evolution rules used for encryption/decryption.

<table>
<thead>
<tr>
<th>Rule number</th>
<th>Rules applied</th>
<th>Binary code</th>
<th>Hexadecimal code</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>51 60 60 60 51 51 51 51</td>
<td>000001110</td>
<td>0e</td>
</tr>
<tr>
<td>1</td>
<td>51 60 60 60 51 51 51 51</td>
<td>000111001</td>
<td>1c</td>
</tr>
<tr>
<td>2</td>
<td>60 51 60 60 60 51 51 51</td>
<td>000111011</td>
<td>1d</td>
</tr>
<tr>
<td>......</td>
<td>......</td>
<td>......</td>
<td>......</td>
</tr>
<tr>
<td>155</td>
<td>102 102 102 102 102 102 102 102</td>
<td>111111111</td>
<td>1ff</td>
</tr>
</tbody>
</table>

In Table 2, column 3 (Binary code) first bit (bold bit) is for $S_1$ and the other eight are for $S_0$. In the 8-cell PCA, the data path is 8 bits and there are 9 rules configuration signals and one load data control signal. Because all the cells share the same common rule signal $S_1$, only rule 51 or one of the rules 60 or 102 can be applied at a given time. The left and right input terminals of every cell are connected to left and right neighbors terminals, thus it is configured as a 3-neighbourhood CA. The left and right terminals of the leftmost and rightmost cells are connected to “logic 0” providing a null boundary condition.

A C# program has been developed to generate rules, change rule sequence and output binary codes for the CA automatically.

4. Testing, results and security analyses

Many experiments were carried out throughout the development of the cryptosystem, first for investigation of the (P)CA time evolution (PCA comportment), second for assuring that the generated sequences respect the CA theory and third to be sure that the PCA based cryptosystem has a good security and high speed. In our previous paper [8], we tested the response of the PCA to the all possible combinations of the control signals according to the rules 51, 60 or 102 using an application program. We find that in the state-transition graph, each cycle has a length 8 (a basic requirement for enciphering).

The complete encryption application including the five pipelines PCA was fully implemented in software using C# programming language.

In Figure 8 is presented a relevant instance of a demonstrative encryption and decryption task applied to a common text message.

The proposed CA and PCA based encryption method was tested and validated in practice using the content of medical databases and XML (EXtensible Markup Language) files stored on a PC or PDA that refers the drug prescription including dozes and administration of the treatment. A special application program runs on the personal patient’s terminal (PDA) for 24 hours under the control of his or her personalized therapy which is hosted on the medical DB server [9]. The program that runs on the mobile device performing the following tasks: the client-server communication that means the patient terminal is capable to initiate a connection to upload or download interested data at the prescribed time. During the periodic data exchanging the particular XML files are transmitted. These are the subject of encryption with the proposed PCA system. We exemplify the functionality of the CA and PCA.
encryption method using the content of medical databases and XML (EXtensible Markup Language) files stored on a PDA or PC, that refers the drug prescription (Figure 9 is for plaintext, Figure 10 is for ciphertext), and multi-parametric monitoring, too.

**Figure 9. Plaintext information displayed for users.**

Also, in the PCA encryption algorithm, the same ciphertext may be generated from different plaintext, and any ciphertext may give rise as well to different plaintext depending on the different PCA’s rule configuration.

In order to be sure that the encryption system is valid, the PCA encrypted sequences was tested using a set of 16 statistical tests proposed by the National Institute of Standards and Technology (NIST) [10].

**Table 3. Results obtained for NIST statistical tests.**

<table>
<thead>
<tr>
<th>No.</th>
<th>Test applied</th>
<th>Results (p_value)</th>
<th>Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Frequency Test</td>
<td>0.835346</td>
<td>SUCCESS</td>
</tr>
<tr>
<td>2.</td>
<td>Frequency Test within a Block</td>
<td>0.469023</td>
<td>SUCCESS</td>
</tr>
<tr>
<td>3.</td>
<td>Runs Test</td>
<td>0.712944</td>
<td>SUCCESS</td>
</tr>
<tr>
<td>4.</td>
<td>Test for the Longest Run of Ones in a Block</td>
<td>0.210251</td>
<td>SUCCESS</td>
</tr>
<tr>
<td>5.</td>
<td>Binary matrix Rank Test</td>
<td>0.992326</td>
<td>SUCCESS</td>
</tr>
<tr>
<td>6.</td>
<td>Discrete Transform</td>
<td>0.866944</td>
<td>SUCCESS</td>
</tr>
<tr>
<td>7.</td>
<td>Non-Overlapping Template Matching Test</td>
<td>0.131868</td>
<td>SUCCESS</td>
</tr>
<tr>
<td>8.</td>
<td>Overlapping Template Matching Test</td>
<td>1.000000</td>
<td>SUCCESS</td>
</tr>
<tr>
<td>9.</td>
<td>Maurer’s Universal Statistical Test</td>
<td>0.016163</td>
<td>SUCCESS</td>
</tr>
<tr>
<td>10.</td>
<td>Linear Complexity Test</td>
<td>0.775136</td>
<td>SUCCESS</td>
</tr>
<tr>
<td>11.</td>
<td>Serial Test p1 = 0.29 p2 = 0.67</td>
<td>0.775136</td>
<td>SUCCESS</td>
</tr>
<tr>
<td>12.</td>
<td>Approximate Entropy Test</td>
<td>0.279436</td>
<td>SUCCESS</td>
</tr>
<tr>
<td>13.</td>
<td>Cumulative Sums Forward</td>
<td>0.495808</td>
<td>SUCCESS</td>
</tr>
<tr>
<td>14.</td>
<td>Cumulative Sums Reverse</td>
<td>0.678943</td>
<td>SUCCESS</td>
</tr>
<tr>
<td>15.</td>
<td>Random Excursions Test</td>
<td>0.537136</td>
<td>SUCCESS</td>
</tr>
<tr>
<td>16.</td>
<td>Random Variants Excursions Test</td>
<td>0.852371</td>
<td>SUCCESS</td>
</tr>
</tbody>
</table>

The generated sequences pass all the NIST tests (as is presented in Table 3) and the system is accepted as possible random.

### 5. Conclusions and Future research directions

This study provides an alternative CA and PCA based cryptogram. The cryptogram was tested and verified using an illustrative example in area of e-health applications.

General accepted expectations were confirmed regarding the following aspects: the properties of the pseudo-random binary sequences generated depend of the initial state of the cellular automaton; the sequence generated is much better when the dimension of the cellular automata is big enough (has a lot of cells); computer effort depends on the size of the cellular space and the length of the
Development of new encryption techniques based on PCA implies a huge simulation effort in order to choose a number of local rules, combined with appropriate initial states and topology, which can be effectively applied in cryptography.

The encryption/decryption modules share the same structure, so it is easy for implementation (also in hardware). Because of the nature of the CA, the encryption algorithm is most efficient when implemented in massively parallel integrated circuits (FPGA).

In the immediate future, the PCA encryption project will be implemented in hardware, in FPGA circuits, to achieve high speed (specialized hardware has a significant performance advantage) and good security (there is no physical protection for an encryption algorithm written in software).

6. Acknowledgements

This work was supported by CNCSIS/CNCS UEFISCSU/UEFISCDI, project number PN II-RU PD 369/2010, contract number 10/02.08.2010.

7. References