
Security analysis of a key agreement protocol based on Spiking Neural P Systems

Mihail-Iulian Pleșa*¹, Marian Gheoghe², Florentin Ipate¹, and Gexiang Zhang³

¹ Department of Computer Science, University of Bucharest, Bucharest, Romania
mihail-iulian.plesa@s.unibuc.ro

² School of Electrical Engineering and Computer Science, University of Bradford, Bradford, UK

³ School of Automation, Chengdu University of Information Technology, Chengdu 610225, China

Summary. Key agreement protocols are a central part of cryptography research. The idea of such a protocol is to allow two or more parties to reach a shared secret by communicating over a public channel. There are three main types of key agreement protocols: based on hard mathematical problems, based on quantum effects and based on neural synchronization. Although they have not been much studied until now, key agreement protocols based on neural synchronization have several advantages. First, their security does not involve any number theory problem which may be solved on a quantum computer. Second, unlike quantum key agreement protocols, they do not require special hardware which is expensive and hard to manage. Recently, a new neural key agreement protocol based on the Anti-Spiking Neural Tree Parity Machine P system, ASNTPM P system for short, has been proposed. Although the protocol is more efficient than the rest of the neural key agreement protocols, no security analysis was performed.

In this paper, we study the security of the protocol based on ASNTPM P systems from a cryptographic perspective. We analyze the running time of the protocol with respect to the parameters of the system. We adopt multiple attacks from the neural cryptography literature and show that the ASNTPM P system-based protocol is secure. Through a series of experiments, we show that the running time of the protocol grows polynomially in the system parameters while the probability that an attack will succeed decreases exponentially.

Key words: Spiking Neural P system, Anti-spikes, ASNTPM P systems, Tree Parity Machine, Cryptography

1 Introduction

Key agreement protocols are the basis of all modern cryptographic protocols. From simple web traffic which is secured using the TLS protocol to the more complex

end-to-end encrypted messaging communication platforms which are based on Signal protocol, most cryptographic systems use some sort of key agreement protocols [10, 11, 42]. The main role of such protocols is to establish a shared secret among two or more parties using public communication channels.

Currently, most key agreement protocols are based on hard number theory problems e.g., the discrete logarithm problem (DLP), the Diffie-Hellman problem (DHP), the decisional Diffie-Hellman problem (D-DHP), the factorization problem, etc [15]. The disadvantage of these protocols is that they are vulnerable if a large quantum computing is built. In [47], Peter Shor proposed a quantum algorithm that can solve the DLP and the factorization problem in polynomial time. Until this moment, there is no large enough quantum computer to endanger the cryptographic primitives and protocols deployed in the industry. However, it is expected that such a computer will be built shortly [38].

There are three alternatives to the current key agreement protocols:

1. Post-quantum key agreement protocols
2. Quantum key agreement protocols
3. Neural key agreement protocols

The post-quantum key agreement protocols are based on hard mathematical problems for which no efficient solution is known on classical or quantum computers [4]. The disadvantage is that there is no mathematical proof that there is indeed no solution to those problems. The quantum key agreement protocols are based on quantum effects e.g., the collapse of the probability wave, entanglement, no-cloning theorem, etc. [23]. Although these protocols are secure even if a large quantum computer is built, they require specialized hardware which is hard to manage and expensive.

There are many important applications of neural networks [1, 2, 16, 25, 27, 29–32, 36, 39, 41]. A less known application is the construction of key agreement protocols. The neural key agreement protocols represent an alternative to post-quantum or quantum protocols. The idea behind these protocols is to synchronize over a public channel two special neural networks called Tree Parity Machines, TPMs for short [21]. Unlike post-quantum key agreement protocols, they are not based on any hard mathematical problem so they are quantum secure. Their principal advantage over the quantum key agreement protocols is that they do not require special hardware and can be implemented on any classical computer.

Until recently, most TPMs were constructed using neurons modeled after the perceptrons. In [40], the authors proposed for the first time a neural key agreement protocol based on Spiking Neural P systems. They constructed a special type of TPM called the Anti-Spiking Neural Tree Parity P system (ASNTPM P system) and showed experimentally that their protocol is more efficient than the classical neural key agreement protocols based on TPMs. In this paper, we study the security of this protocol from a cryptographic perspective. We use the most simple security model in which the attacker only eavesdrops on the communication channel. The attacker is not allowed to alter the messages exchanged by the

legitimate parties nor to insert or delete messages. We adopt multiple attacks on neural key agreement protocols from the literature and test whether the ASNTPM P system-based protocol is secure against them. The paper is organized as follows: in Section 2 we present related work and our contribution. In Section 3 we introduce the definition of the model proposed by [40]. In Section 4 we analyze the running time of the protocol with respect to the parameters of the system. In Section 5 we discuss four different types of attacks against neural cryptography and show experimentally that the protocol proposed in [40] is secure. Section 5 is left for conclusions.

2 Related work

The idea of using neural synchronization to build a key agreement protocol was first proposed in [21]. The TPM proposed by the authors was a three-layer neural network with binary inputs. Shortly after the idea was launched, three types of attacks were proposed in [22]. The authors showed through multiple experiments that they can recover more the 90% of the shared key using the geometric attack. In [28] the authors experimentally proved that increasing the range of the weight can improve the security of the protocol. The paper provides evidence that increasing the range increases the synchronization time polynomially while decreasing the probability that the geometric attack will succeed exponentially. In [46] is presented a more powerful attack than the geometric one. This attack called the majority attack cannot be mitigated by increasing the range of the weights.

There are also other strategies for improving the security of neural key agreement protocols. In [43] the authors proposed a mechanism by which the inputs of the TPM are generated based on the current internal state of the network. In [7] and [8] the authors presented two algorithms for perturbing the output of the TPM in such a way that the attacker cannot recover the original information but the two legitimate parties can synchronize. This improves the security of the protocol because every attack uses the fact that the eavesdropper can intercept the outputs exchanged by legitimate parties over a public channel.

In [51] and [20] the authors proposed other architectures for constructing a TPM. In [51] the idea of using non-binary input values is presented improving the running time of the protocol. In [20] it is shown that using vector values as inputs can further improve the efficiency and the security of the protocol. Similarly, [12] proposed a TPM with complex input values. In [52] the authors analyzed the impact of non-binary input values on the security of a TPM-based key agreement protocol. Regarding the practical aspects of neural cryptography, in [45] several sets of parameters for TPMs were analyzed. For each set, the authors presented the synchronization time and the security impact.

In this work, we are dealing with TPM instantiated with Spiking Neural P systems. These systems are a special type of the membrane computing model introduced in [37]. Membrane computing models have been used to solve hard problems like Hamiltonian Path in polynomial time [58]. In [6] the authors proposed a

new sorting algorithm based on P systems. There are also P systems inspired by various physical phenomena. In [5] the authors presented a P system in which the membranes have a limited capacity and [17] presents a new model inspired by the controlled circulation of water. Also, there are attempts to make these models in the laboratory [26].

Spiking neural P systems were first introduced in [18]. Over time, new functionalities inspired by various biological phenomena were added to the original model. The most known Spiking Neural P systems are the following [9, 33–35, 49, 56, 57]:

1. SN P systems with astrocytes
2. SN P systems with communication on request
3. SN P systems with polarizations
4. SN P systems with colored spikes
5. SN P systems with asynchronous systems
6. SN P systems with anti-spikes
7. SN P systems with a flat maximally parallel use of rules

The protocol analyzed in this paper is instantiated with a TPM based on Spiking Neural P system with anti-spikes. Several variations of this model include Spiking Neural P systems without the annihilating priority [55] and Spiking Neural P systems with multiple channels [50]. Spiking Neural P systems were studied from both a practical and a theoretical perspective. The computational power of SN P systems with multiple channels was investigated in [24] while a formal verification of SN P systems by mapping them to kernel P systems was made in [14, 19]. SN P systems were also used to simulate uniform sequential computing models [3].

Apart from key agreement protocol, SN P systems have other applications in cryptography [48, 59]. In [13] the authors implemented the famous RSA algorithm using SN P systems and in [54] the authors used a variant of SN P systems to break the same cryptosystem by proposing a new and efficient factorization procedure [53].

2.1 Our contribution

In this paper, we make a security analysis of the key agreement protocol proposed in [40]. In the paper, the authors proposed a new TPM called Anti-Spiking Neural Tree Parity Machine which is based on SN P systems with anti-spikes. We propose a new algorithm for computing the synchronization percentage between two AS-NTPM P systems and also study the efficiency of the protocol with respect to the parameters of the SN P system. Our main contribution is a series of experiments in which we show that increasing the number of input neurons or the number of hidden neurons can exponentially decrease the percentage of the key recovered by the attacker. This growth in the number of neurons increases the synchronization time only polynomially. Our experiments adopt known attacks on TPM-based key agreement protocols in a simple security model in which the attacker can only eavesdrop on the messages exchanged by the legitimate parties.

3 ASNTPM P Systems

The definition of an ASNTPM P system as stated in [40] is the following:

Definition 1. *An Anti-Spiking Neural Tree Parity Machine P system is defined as the following construct:*

$$\Pi = (O, \{\sigma_{in_{11}}, \sigma_{in_{12}}, \dots, \sigma_{in_{KN}}\}, \{\sigma_{h_1}, \sigma_{h_2}, \dots, \sigma_{h_K}\}, \sigma_{out}, N, K, L, syn_0, f)$$

where:

1. $O = \{a, \bar{a}\}$ is an alphabet formed by two symbols:
 - a) The symbol a denotes a spike
 - b) The symbol \bar{a} denotes an anti-spike
2. $\sigma_{in_{ij}}$ is an input neuron formed by the following tuple $(n_{ij}, \bar{n}_{ij}, R_{ij})$:
 - a) n_{ij} denotes the number of spikes from the neuron
 - b) \bar{n}_{ij} denotes the number of anti-spikes from the neuron
 - c) R_{ij} is a finite set of rules of the following forms:
 - i. Firing rules: $b^c \rightarrow b^{c*w_{ij}}$ where $b \in \{a, \bar{a}\}$, $1 \leq c \leq L$ and w_{ij} is a positive integer that will be defined below.
If at the moment t a neuron has c spikes or anti-spikes it will fire, consuming either c spikes or c anti-spikes and sending $c * w_{ij}$ spikes or $c * w_{ij}$ anti-spikes to the hidden neuron σ_{h_i} with which it is connected. At moment $t = 0$, there are no spikes or anti-spikes in the neuron i.e., $n_{ij} = 0$, $\bar{n}_{ij} = 0$.
 - ii. Annihilation rule: $a\bar{a} \rightarrow \lambda$
This rule indicates that at any moment t , an input neuron cannot contain spikes and anti-spikes simultaneously. If a spike and an anti-spike are present in an input neuron, they will annihilate each other instantaneously.

Here, $1 \leq i \leq K$ and $1 \leq j \leq N$.

3. σ_{h_i} is a hidden neuron formed by the following tuple (n_i, \bar{n}_i, R_i) :
 - a) n_i denotes the number of spikes from the neuron
 - b) \bar{n}_i denotes the number of anti-spikes from the neuron
 - c) R_i is a finite set of rules of the following form:
 - i. Firing rules: rule of the form $b^c \rightarrow b$ where $b \in \{a, \bar{a}\}$, $1 \leq c \leq NL$.
If at the moment t the neuron has c spikes it will fire consuming c spikes and sending 1 spike to the output neuron. If at the moment t the neuron has c anti-spikes it will fire consuming c anti-spikes and sending 1 anti-spike to the output neuron. At moment $t = 0$, there are no spikes or anti-spikes in the neuron i.e., $n_i = 0$, $\bar{n}_i = 0$.
 - ii. Annihilation rule: $a\bar{a} \rightarrow \lambda$
This rule indicates that at any moment t , a hidden neuron cannot contain spikes and anti-spikes simultaneously. If a spike and an anti-spike are present in a hidden neuron, they will annihilate each other instantaneously.

Here, $1 \leq i \leq K$.

4. σ_{out} is the output neuron formed by the following tuple $(n_{out}, \bar{n}_{out}, r_{out})$:
 - a) n_{out} denotes the number of spikes from the neuron
 - b) \bar{n}_{out} denotes the number of anti-spikes from the neuron
 - c) r_{out} is an annihilation rule of the form $a\bar{a} \rightarrow \lambda$. The rule indicates that the output neuron cannot hold spikes and anti-spikes simultaneously. At moment $t = 0$, there are no spikes or anti-spikes in the neuron i.e., $n_{out} = 0$, $\bar{n}_{out} = 0$.

The output of the system is the number of spikes or anti-spikes from this neuron.

5. syn_t is the set of synapses at the computational step t . A synapse is defined by the triplet $(\sigma_i, \sigma_j, w_{ij})$ meaning the existence of a synapse between the neuron σ_i and the neuron σ_j . Here, σ_i and σ_j can be input, hidden, or output neurons. The weight on the synapse, $w_{ij} \in \mathbb{Z}^+$ has the role to amplify the spikes or the anti-spikes passing through the synapse e.g., if the neuron σ_i fires sending c spikes or anti-spikes to the neuron σ_j and the weight on the synapse is w_{ij} then the neuron σ_j will receive $c * w_{ij}$ spikes or anti-spikes. syn_0 represents the set of synapses at moment $t = 0$. Initially, the weights between the input and the hidden neurons are randomly chosen from the set $\{1, 2, \dots, L\}$. The weights between the hidden and the output neurons are always 1.
6. N is the number of input neurons connected to a single hidden neuron.
7. K is the number of neurons hidden neurons.
8. L represents the maximum value of a weight i.e., $0 < w_{ij} \leq L$.
9. The learning function f has the role of updating the weights on the synapses according to (1):

$$syn_{t+1} = f(syn_t) \quad (1)$$

The exact form of the learning function f is described by the procedure from Algorithm 3.

The system is initialized using the initialization procedure described in Algorithm 1. The input consists of the ASNTPM P System and a vector of $N * K$ elements $X = (x_{11}, x_{12}, \dots, x_{KN})$, $-L \leq x_{ij} \leq L$, $x_{ij} \neq 0$, $\forall 1 \leq i \leq K, 1 \leq j \leq N$. The input of the system is defined by the vector. If $x_{ij} < 0$ then the input neuron σ_{ij} will receive from the environment $|x_{ij}|$ anti-spikes. If on the other hand, $x_{ij} > 0$ then the input neuron σ_{ij} will receive from the environment x_{ij} spikes. The initialization procedure is described by Algorithm 1.

The input neurons fire sending all the spikes or all the anti-spikes to the hidden neurons. The number of spikes or anti-spikes is amplified by the corresponding weight of the synapse. After the annihilation rule is applied the maximum number of times in each hidden neuron, they send one spike or one anti-spike to the output neuron. After the annihilation rule is applied the maximum number of times in the output neuron, it can be in one of the following states:

1. The output neuron is empty if the number of spikes received from the hidden neurons is equal to the number of anti-spikes.

2. The output neuron contains one spike if the number of spikes received from the hidden neurons is greater than the number of anti-spikes.
3. The output neuron contains one anti-spike if the number of spikes received from the hidden neurons is smaller than the number of anti-spikes.

The output of the system is the state of the output neuron. The system evolves by the application of the learning function f which modifies the weights of the synapses between the input and the hidden neurons. The running procedure of an ANSTPM P System is described in Algorithm 2. A generic ASNTPM P System is presented in Figure 1.

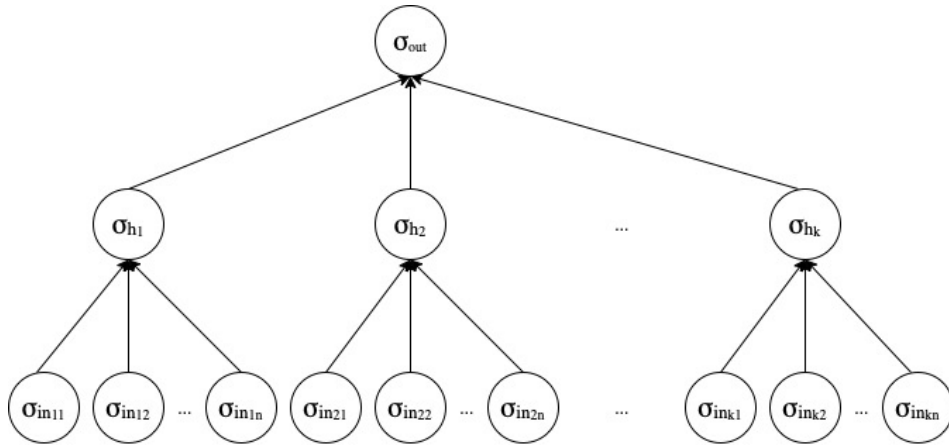


Fig. 1. A generic ASNTPM P System

Let $N(\Pi, \sigma)$ be the number of spikes from neuron σ of the ASNTPM Π . Similarly, let $\bar{N}(\Pi, \sigma)$ be the number of anti-spikes from neuron σ of the ASNTPM Π . Here σ can be an input, a hidden, or an output neuron.

Let W_Π be the weights on the synapses of the ASNTPM Π . More exactly, W_Π is a $K \times N$ matrix in which the element on line i and column j denoted as $W_\Pi[i][j]$ represent the weight between the hidden neuron σ_{h_i} and the input neuron $\sigma_{in_{ij}}$.

Algorithm 1 ASNTPM P System initialization

```

1: function INITIALIZE( $\Pi, X$ )
2:   for  $i = 1; i \leq K; i = i + 1$  do
3:     for  $j = 1; j \leq N; j = j + 1$  do
4:       if  $X[i * N + j] \leq 0$  then
5:          $\bar{N}(\Pi, \sigma_{in_{ij}}) = |X[i * N + j]|$ 
6:       else
7:          $N(\Pi, \sigma_{in_{ij}}) = X[i * N + j]$ 
8:       end if
9:        $W_{\Pi}[i][j] \stackrel{\$}{\leftarrow} [0, 2L]$ 
10:    end for
11:  end for
12:  for  $i = 1; i \leq K; i = i + 1$  do
13:     $\bar{N}(\Pi, \sigma_{h_i}) = 0$ 
14:     $N(\Pi, \sigma_{h_i}) = 0$ 
15:  end for
16:   $\bar{N}(\Pi, \sigma_{out}) = 0$ 
17:   $N(\Pi, \sigma_{out}) = 0$ 
18: end function

```

Algorithm 2 ASNTPM P System running

```

1: function RUN( $\Pi$ )
2:   for  $i = 1; i \leq K; i = i + 1$  do
3:     for  $j = 1; j \leq N; j = j + 1$  do
4:        $\bar{N}(\Pi, \sigma_{h_i}) = \bar{N}(\Pi, \sigma_{h_i}) + W_{\Pi}[i][j] * \bar{N}(\Pi, \sigma_{in_{ij}})$ 
5:        $N(\Pi, \sigma_{h_i}) = N(\Pi, \sigma_{h_i}) + W_{\Pi}[i][j] * N(\Pi, \sigma_{in_{ij}})$ 
6:     end for
7:   end for
8:   for  $i = 1; i \leq K; i = i + 1$  do
9:      $\bar{N}(\Pi, \sigma_{out}) = \bar{N}(\Pi, \sigma_{out}) + \bar{N}(\Pi, \sigma_{h_i})$ 
10:     $N(\Pi, \sigma_{out}) = N(\Pi, \sigma_{out}) + N(\Pi, \sigma_{h_i})$ 
11:   end for
12: end function

```

Algorithm 3 The learning function

```

1: function UPDATEWEIGHTS( $\Pi$ )
2:   for  $i = 1; i \leq K; i = i + 1$  do
3:     if  $[[N(\Pi, \sigma_{h_i}) = N(\Pi, \sigma_{out})] \vee [\bar{N}(\Pi, \sigma_{h_i}) = \bar{N}(\Pi, \sigma_{out})]]$  then
4:       for  $j = 1; j \leq N; j = j + 1$  do
5:         if  $N(\Pi, \sigma_{out}) > 0$  then
6:            $W_{\Pi}[i][j] = |W_{\Pi}[i][j] + N(\Pi, \sigma_{in_{ij}})|$ 
7:         else if  $\bar{N}(\Pi, \sigma_{out}) > 0$  then
8:            $W_{\Pi}[i][j] = |W_{\Pi}[i][j] - \bar{N}(\Pi, \sigma_{in_{ij}})|$ 
9:         end if
10:       end for
11:     if  $W_{\Pi}[i][j] > L$  then
12:        $W_{\Pi}[i][j] = L$ 
13:     end if
14:   end if
15: end for
16: end function

```

4 The synchronization of two ASNTPM P Systems

Two ASNTPM P Systems are synchronized if their weights are identical. To formalize this idea, we introduce a new quantitative indicator called the synchronization percentage which describes how much two ASNTPM P Systems are synchronized. This indicator is computed by the function *SynchronizationPercentage* presented in Algorithm 4.

Algorithm 4 The synchronization percentage

```

1: function SYNCHRONIZATIONPERCENTAGE( $\Pi_1, \Pi_2$ )
2:    $total \leftarrow 0$ 
3:    $counter \leftarrow 0$ 
4:   for  $i = 1; i \leq K; i = i + 1$  do
5:     for  $j = 1; j \leq N; j = j + 1$  do
6:       if  $W_{\Pi_1}[i][j] \neq 2L \wedge W_{\Pi_2}[i][j] \neq 2L$  then
7:          $total \leftarrow total + 1$ 
8:       if  $W_{\Pi_1}[i][j] = W_{\Pi_2}[i][j]$  then
9:          $counter \leftarrow counter + 1$ 
10:      end if
11:    end if
12:  end for
13: end for
14: return  $counter/total$ 
15: end function

```

A simple function for synchronizing two ASNTPM P Systems defined by the same parameters K , N and L is presented in Algorithm 5. Since both systems update their weights based on their mutual output, we say that the two ASNTPM P systems are synchronizing with mutual learning.

Algorithm 5 Synchronization of two ASNTPM P Systems

```

1: function SYNCHRONIZE( $\Pi_1, \Pi_2$ )
2:   while SYNCHRONIZATIONPERCENTAGE( $\Pi_1, \Pi_2$ )  $\neq$  1 do
3:      $x \stackrel{R}{\leftarrow} \mathbb{Z}^{KN}$ 
4:     INITIALIZE( $\Pi_1, x$ )
5:     INITIALIZE( $\Pi_2, x$ )
6:     RUN( $\Pi_1$ )
7:     RUN( $\Pi_2$ )
8:     if  $N(\Pi_1, \sigma_{out}) = N(\Pi_2, \sigma_{out}) \vee \bar{N}(\Pi_1, \sigma_{out}) = \bar{N}(\Pi_2, \sigma_{out})$  then
9:       UPDATEWEIGHTS( $\Pi_1$ )
10:      UPDATEWEIGHTS( $\Pi_2$ )
11:     end if
12:   end while
13: end function

```

We study the efficiency of the function *Synchronize* with respect to the parameters K and N of the two ASNTPM P Systems. We note that both systems have the same parameters K , N and L . We denote by $T(\textit{Synchronize})$ the number of steps taken by the function *Synchronize*.

Hypothesis 1. The number of steps taken by the function *Synchronize* to synchronize two ASNTPM P Systems is quadratic in the parameter K of the inputs:

$$T(\textit{Synchronize}) = 0.8K^2 - 30K + 1184 \quad (2)$$

Experiment. We run the algorithm for 50 times and compute the mean of the results for each $K \in \{4, 8, 16, 32, 64, 128, 256\}$ with $L = 256$ and $N = 128$. The results are presented in Table 1 and Figure 2.

Table 1. The efficiency of Algorithm 5 with respect to K

K	4	8	16	32	64	128	256
T(Synchronize)	158.08	286.66	623.34	1443.48	3532.3	10253.7	47492.06

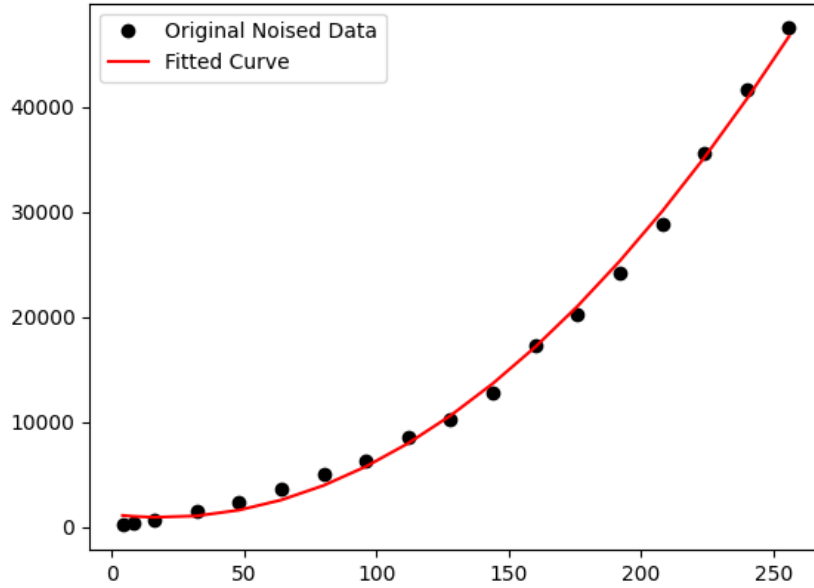


Fig. 2. The number of iterations of Algorithm 5 with respect to K

Using the least square method with the Levenberg-Marquardt algorithm on the values recorded during the experiment we computed the best polynomial that fits the data and obtained (2).

Hypothesis 2. The number of steps taken by the function *Synchronize* to synchronize two ASNTPM P Systems is linear in the parameter N of the inputs:

$$T(\textit{Synchronize}) = 27N + 191 \tag{3}$$

Experiment. We run the algorithm for 50 times and compute the mean of the results for each $N \in \{4, 8, 16, 32, 64, 128, 256\}$ with $L = 256$ and $K = 64$. The results are presented in Table 2 and Figure 3.

Table 2. The efficiency of Algorithm 5 with respect to N

N	4	8	16	32	64	128	256
T(Synchronize)	277.42	483.64	744.28	1173.58	1960.0	3594.54	7441.92

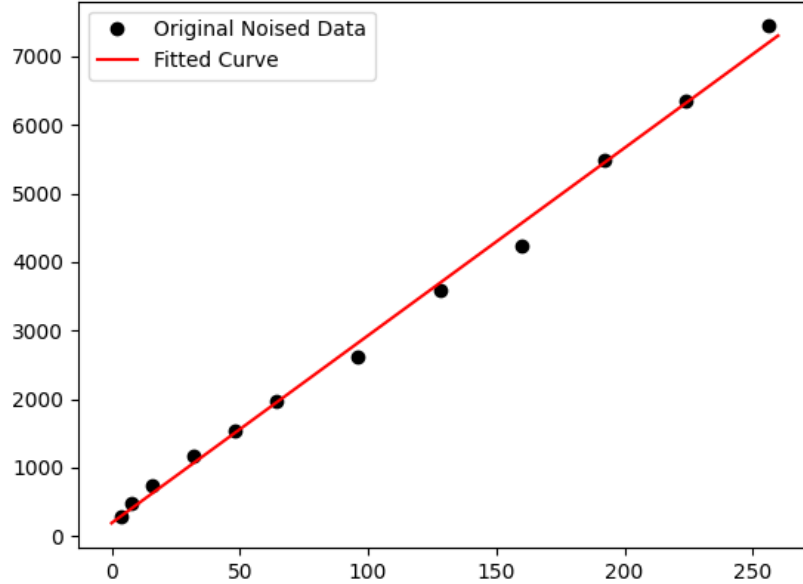


Fig. 3. The number of iterations of Algorithm 5 with respect to N

Using the least square method with the Levenberg-Marquardt algorithm on the values recorded during the experiment we computed the best polynomial that fits the data and obtained (3).

5 Attacks on the synchronization process

In this section, we will examine three algorithms through a series of experiments that aim to synchronize two ASNTPMs without mutual learning. These algorithms are inspired by various attacks on key agreement protocols based on TPMs [22, 44, 46]. All algorithms received as inputs three ASNTPMs: Π_1, Π_2 and Π_3 . The purpose of each algorithm is to synchronize Π_3 with Π_1 without mutual learning in the time frame in which Π_1 and Π_2 synchronize with mutual learning using Algorithm 5.

Algorithm 6 presents the naive solution inspired by [22] while Algorithm 7 presents the geometric solution inspired by [46]. The genetic attack presented in [44] is exponential in K so we do not include it here. We denote by $\rho(\Pi_x, \Pi_y)$ the synchronization percentage between Π_x and Π_y after the execution of the synchronization algorithm.

5.1 The naive attack

The function *NaiveAttack* stops execution when Π_1 and Π_2 are fully synchronized. The algorithm returns the synchronization percentage between Π_1 and Π_3 . Hypotheses 3 and 4 capture the relation between the parameters K and N of the ASNTPM P systems and $\rho(\Pi_1, \Pi_3)$.

Algorithm 6 The naive attack on the synchronization process

```

1: function NAIVEATTACK( $\Pi_1, \Pi_2, \Pi_3$ )
2:   while SYNCHRONIZATIONPERCENTAGE( $\Pi_1, \Pi_2$ )  $\neq$  1 do
3:      $x \xleftarrow{R} \mathbb{Z}^{KN}$ 
4:     INITIALIZE( $\Pi_1, x$ )
5:     INITIALIZE( $\Pi_2, x$ )
6:     INITIALIZE( $\Pi_3, x$ )
7:     RUN( $\Pi_1$ )
8:     RUN( $\Pi_2$ )
9:     RUN( $\Pi_3$ )
10:    if  $N(\Pi_1, \sigma_{out}) = N(\Pi_2, \sigma_{out}) \vee \bar{N}(\Pi_1, \sigma_{out}) = \bar{N}(\Pi_2, \sigma_{out})$  then
11:      UPDATEWEIGHTS( $\Pi_1$ )
12:      UPDATEWEIGHTS( $\Pi_2$ )
13:    if  $N(\Pi_1, \sigma_{out}) = N(\Pi_3, \sigma_{out}) \vee \bar{N}(\Pi_1, \sigma_{out}) = \bar{N}(\Pi_3, \sigma_{out})$  then
14:      UPDATEWEIGHTS( $\Pi_3$ )
15:    end if
16:  end if
17: end while
18:   return SYNCHRONIZATIONPERCENTAGE( $\Pi_1, \Pi_3$ )
19: end function

```

Hypothesis 3. The synchronization percentage between Π_1 and Π_3 after running the function *NaiveAttack* drops exponentially in K according to (4).

$$\rho(\Pi_1, \Pi_3) = 1.27e^{-0.11K} + 0.04 \tag{4}$$

Experiment. We run the algorithm for 50 times and compute the mean of the results for each $K \in \{4, 8, 16, 32, 64, 128, 256\}$ with $L = 256$ and $N = 128$. The results are presented in Table 3 and Figure 4.

Table 3. $\rho(\Pi_1, \Pi_3)$ using Algorithm 6 with respect to K

K	4	8	16	96	128	192	256
$\rho(\Pi_1, \Pi_3)$	0.83	0.53	0.23	0.06	0.05	0.04	0.02

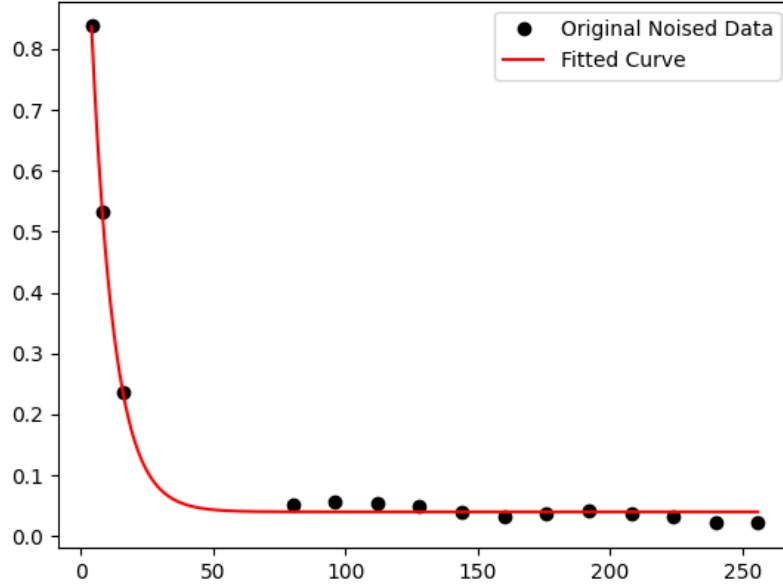


Fig. 4. $\rho(\Pi_1, \Pi_3)$ using Algorithm 6 with respect to K

Using the least square method with the Levenberg-Marquardt algorithm on the values recorded during the experiment we computed the best power function that fits the data and obtained (4).

Hypothesis 4. The synchronization percentage between Π_1 and Π_3 after running the function *NaiveAttack* drops in N according to (5).

$$\rho(\Pi_1, \Pi_3) = 0.02e^{-0.03N} + 0.06 \quad (5)$$

Experiment. We run the algorithm for 50 times and compute the mean of the results for each $N \in \{4, 8, 16, 32, 64, 128, 256\}$ with $L = 256$ and $K = 64$. The results are presented in Table 4 and Figure 5.

Table 4. $\rho(\Pi_1, \Pi_3)$ using Algorithm 6 with respect to N

\mathbf{K}	4	8	16	96	128	192	256
$\rho(\Pi_1, \Pi_3)$	0.28	0.25	0.18	0.09	0.08	0.07	0.06

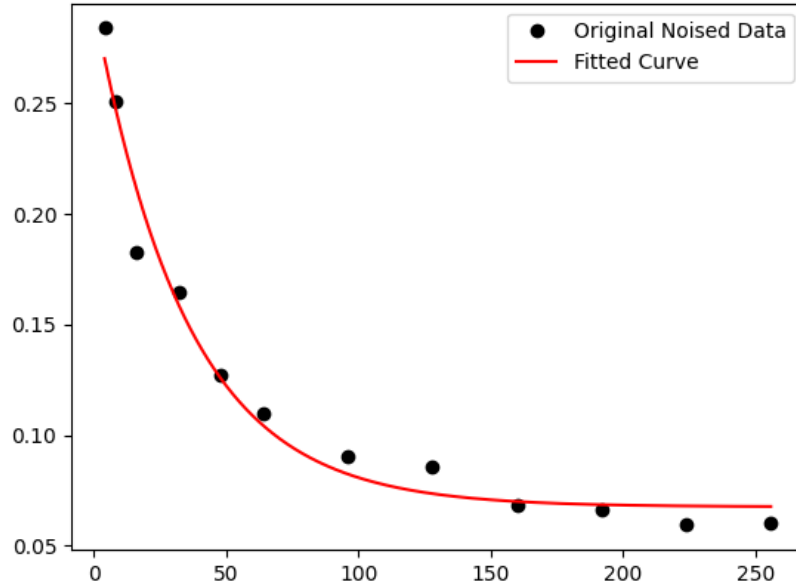


Fig. 5. $\rho(\Pi_1, \Pi_3)$ using Algorithm 6 with respect to N

Using the least square method with the Levenberg-Marquardt algorithm on the values recorded during the experiment we computed the best power that fits the data and obtained (5).

5.2 The geometric attack

Another solution for the synchronization of three ASNTPM P Systems is presented in Algorithm 7. This solution is inspired by the geometric attack on neural cryptography presented in [22].

The idea of this solution is to interpret the weights and the input associated with each hidden neuron as points in a $N - dimensional$ discrete space. When the outputs of Π_1 and Π_2 are the same but the output of Π_3 is different then at least one hidden neuron of Π_3 has the wrong number of spikes or anti-spikes.

To correct this error, we compute the distance between the input and the weights associated with each hidden neuron of Π_3 . The hidden neuron which presents the minimum distance will have the number of spikes and anti-spikes inverted. The output of Π_3 will be set to the output of Π_1 and the weights of Π_3 will be updated using the new configuration.

Although this solution is more efficient than the one presented in Algorithm 6, the synchronization percentage still decreases as K or N increases.

Algorithm 7 The geometric solution for the synchronization of three ASNTPM P Systems

```

function GEOMETRICATTACK( $\Pi_1, \Pi_2, \Pi_3$ )
  while SYNCHRONIZATIONPERCENTAGE( $\Pi_1, \Pi_2$ )  $\neq$  1 do
     $x \stackrel{R}{\leftarrow} \mathbb{Z}^{KN}$ 
    INITIALIZE( $\Pi_1, x$ )
    INITIALIZE( $\Pi_2, x$ )
    INITIALIZE( $\Pi_3, x$ )
    RUN( $\Pi_1$ )
    RUN( $\Pi_2$ )
    RUN( $\Pi_3$ )
    if  $N(\Pi_1, \sigma_{out}) = N(\Pi_2, \sigma_{out}) \vee \bar{N}(\Pi_1, \sigma_{out}) = \bar{N}(\Pi_2, \sigma_{out})$  then
      UPDATEWEIGHTS( $\Pi_1$ )
      UPDATEWEIGHTS( $\Pi_2$ )
    if  $N(\Pi_1, \sigma_{out}) = N(\Pi_3, \sigma_{out}) \vee \bar{N}(\Pi_1, \sigma_{out}) = \bar{N}(\Pi_3, \sigma_{out})$  then
      UPDATEWEIGHTS( $\Pi_3$ )
    end if
    if  $N(\Pi_1, \sigma_{out}) \neq N(\Pi_3, \sigma_{out}) \wedge \bar{N}(\Pi_1, \sigma_{out}) \neq \bar{N}(\Pi_3, \sigma_{out})$  then
       $distance = \|W_{\Pi_3}[1] - x\|$ 
       $minimum = distance$ 
       $index = 1$ 
      for  $i = 2; i \leq K; i = i + 1$  do
         $distance = \|W_{\Pi_3}[i] - x\|$ 
        if  $distance < minimum$  then
           $minimum = distance$ 
           $index = i$ 
        end if
      end for
       $aux = \bar{N}(\Pi_3, \sigma_{h_{index}})$ 
       $\bar{N}(\Pi_3, \sigma_{h_{index}}) = N(\Pi_3, \sigma_{h_{index}})$ 
       $N(\Pi_3, \sigma_{h_{index}}) = aux$ 
       $\bar{N}(\Pi_3, \sigma_{output}) = \bar{N}(\Pi_1, \sigma_{output})$ 
       $N(\Pi_3, \sigma_{output}) = N(\Pi_1, \sigma_{output})$ 
      UPDATEWEIGHTS( $\Pi_3$ )
    end if
  end while
  return SYNCHRONIZATIONPERCENTAGE( $\Pi_1, \Pi_3$ )
end function

```

Hypothesis 5. The synchronization percentage between Π_1 and Π_3 after running the function *GeometricAttack* drops in K according to (6).

$$\rho(\Pi_1, \Pi_3) = 0.91e^{-0.004K} \tag{6}$$

Experiment. We run the algorithm for 50 times and compute the mean of the results for each $K \in \{4, 8, 16, 32, 64, 128, 256, 512\}$ with $L = 256$ and $N = 128$. The results are presented in Table 5 and Figure 6.

Table 5. $\rho(\Pi_1, \Pi_3)$ using Algorithm 7 with respect to K

K	4	8	16	32	64	128	256	512
$\rho(\Pi_1, \Pi_3)$	0.95	0.88	0.82	0.73	0.65	0.55	0.27	0.05

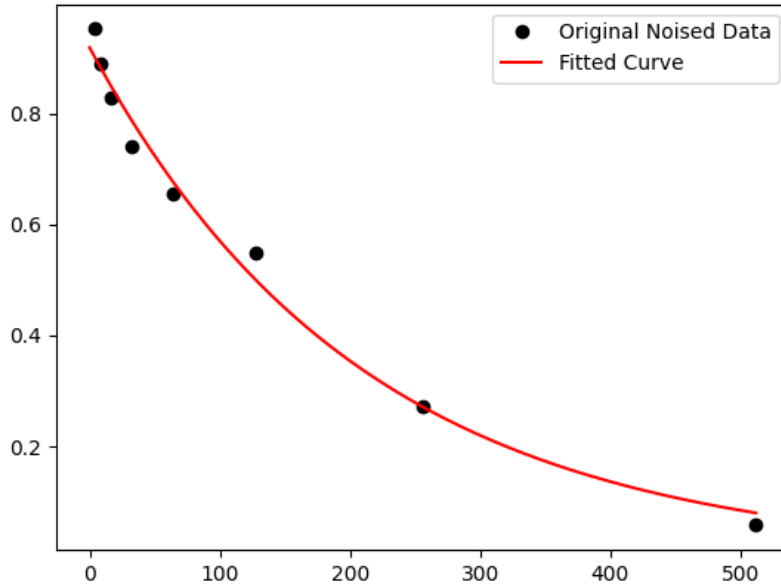


Fig. 6. $\rho(\Pi_1, \Pi_3)$ using Algorithm 7 with respect to K

Using the least square method with the Levenberg-Marquardt algorithm on the values recorded during the experiment we computed the best exponential function that fits the data and obtained (6).

Hypothesis 6. The synchronization percentage between Π_1 and Π_3 after running the function *GeometricAttack* drops in N according to (7).

$$\rho(\Pi_1, \Pi_3) = 0.99e^{-0.003N} \quad (7)$$

Experiment. We run the algorithm for 50 times and compute the mean of the results for each $N \in \{4, 8, 16, 32, 64, 128, 256, 512\}$ with $L = 256$ and $K = 64$. The results are presented in Table 6 and Figure 7.

Table 6. $\rho(\Pi_1, \Pi_3)$ using Algorithm 7 with respect to N

N	4	8	16	32	64	128	256	512
$\rho(\Pi_1, \Pi_3)$	0.98	0.97	0.95	0.90	0.78	0.65	0.46	0.23

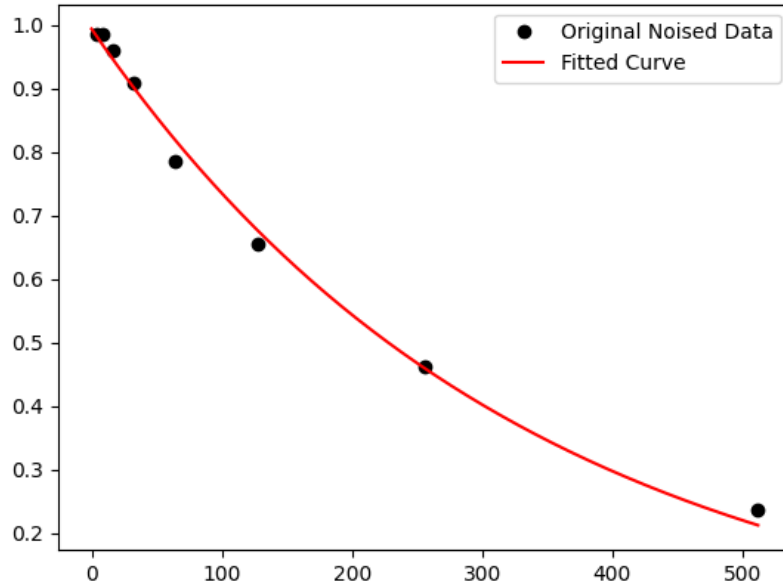


Fig. 7. $\rho(\Pi_1, \Pi_3)$ using Algorithm 7 with respect to N

Using the least square method with the Levenberg-Marquardt algorithm on the values recorded during the experiment we computed the best exponential function that fits the data and obtained (6).

5.3 The majority attack

Let the ordered set $H_{\Pi} = (N(\Pi, \sigma_{h_1}), \bar{N}(\Pi, \sigma_{h_1}), N(\Pi, \sigma_{h_2}), \bar{N}(\Pi, \sigma_{h_2}), \dots, N(\Pi, \sigma_{h_K}), \bar{N}(\Pi, \sigma_{h_K}))$ be the hidden state of the ASNTPM P System Π . Let $S = \{H_{\Pi_1}, H_{\Pi_2}, \dots, H_{\Pi_M}\}$ be the set of hidden states of M ASNTPM P Systems. We denote by $F_S(H_{\Pi_i})$ the frequency of the element H_{Π_i} in the set S . Given two ASNTPM P Systems Π_A and Π_B we can synchronize them in parallel with each of the M ASNTPM P Systems Π_i , for each $1 \leq i \leq M$ using the geometric solution. Similar to [46] we could try to design a solution that uses the frequency $F_S(H_{\Pi_i})$ to synchronize Π_A and Π_B with at least one of the M ASNTPM P Systems. However, this is not possible given the fact that for ASNTPM P Systems there exist certain values of K , s.a. $F_S(H_{\Pi_i}) = \frac{1}{M}$, $\forall 1 \leq i \leq M$. This is illustrated by hypotheses 7 and 8.

Hypothesis 7. Given two ASNTPM P Systems Π_A and Π_B and a set of M ASNTPM P Systems $\{\Pi_1, \Pi_2, \dots, \Pi_M\}$, the mean of the frequencies $F_S(H_{\Pi_i})$ after each iteration of *GeometricAttack* (Π_A, Π_B, Π_i), $1 \leq i \leq M$ converges to $\frac{1}{M}$ as K increases according to (8).

$$\frac{1}{M} \sum_{i=1}^M F_S(H_{\Pi_i}) = 1.13e^{-0.06K} \quad (8)$$

Experiment. We run *GeometricAttack* (Π_A, Π_B, Π_i) in parallel for each $1 \leq i \leq M$ and averaged the result. The procedure was repeated 50 times and the mean of the results were computed for each $K \in \{4, 8, 16, 32, 64, 128, 256, 512\}$ with $L = 256$, $N = 128$ and $M = 128$. The results are presented in Table 7 and Figure 8. Let $\mu_{F_S} = \frac{1}{M} \sum_{i=1}^M F_S(H_{\Pi_i})$.

Table 7. The average μ_{F_S} with respect to K

K	4	8	16	32	64	128	256	512
μ_{F_S}	0.85	0.66	0.36	0.13	0.018	0.00786	0.0078125	0.0078125

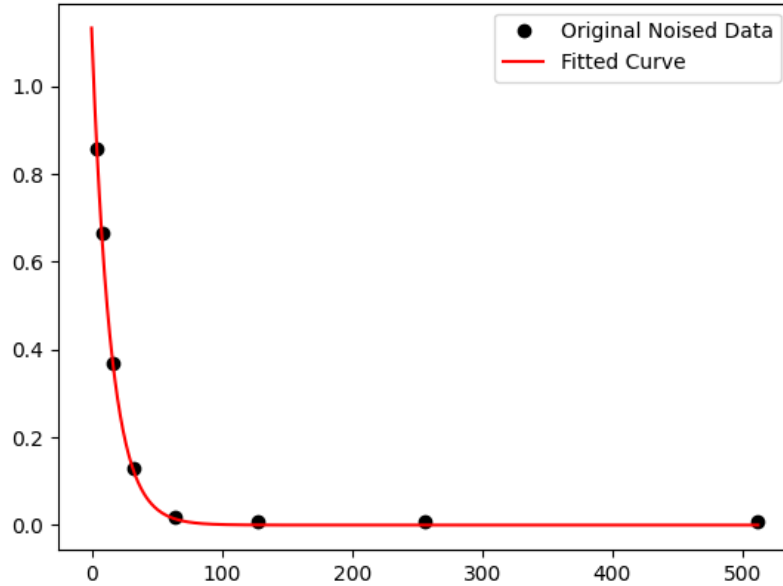


Fig. 8. μ_{FS} with respect to K

Using the least square method with the Levenberg-Marquardt algorithm on the values recorded during the experiment we computed the best exponential function that fits the data and obtained (8).

6 Conclusions and further directions of research

In this paper, we analyzed the security of the neural key agreement protocol proposed in [40]. The protocol proposed by the authors is based on a new type of TPM called the Anti-Spiking Neural Tree Parity Machine. Unlike classical TPM, the model of the neuron used by ASNTPM is inspired by Spiking Neural P systems with anti-spikes. We adopt four different types of attacks on TPM-based protocols: the naive attack, the geometric attack, the majority attack and the genetic attack. We showed through a series of experiments that increasing the number of neurons decreases exponentially the percentage of the key recovered by the attacker. This growth in the number of neurons implies only a polynomial increase in the running time. From a cryptographic perspective, this behavior is similar to trapdoor problems on which the currently used cryptosystems are based.

A further direction of research is to formalize the hard problem on which the security of the TPM-based key agreement protocol is based. Analyzing the security of a specific protocol is dependent on the security model. Constructing a hard problem enables the creation of many cryptographic primitives whose security can be proved by reduction to the underlining hard problem.

Another direction of research implies analyzing the security of the protocol using another security model in which the attacker can alter the messages exchanged by the legitimate parties. The current protocol is insecure in such a security model given the fact that any third party can mount a Man-in-the-Middle (MitM) attack.

The third direction of research is to construct neural key agreement protocols for groups. This is particularly important because most applications for secure communications are designed for group messaging.

References

1. Abu-Zidan, Y., Nguyen, K., Mendis, P., Setunge, S., Adeli, H.: Design of a smart prefabricated sanitising chamber for covid-19 using computational fluid dynamics. *Journal of Civil Engineering and Management* **27**(2), 139–148 (2021)
2. Adeli, H., Kim, H.: *Wavelet-based vibration control of smart buildings and bridges*. CRC Press (2022)
3. Adorna, H.N.: Computing with SN P systems with I/O mode. *Journal of Membrane Computing* **2**(4), 230–245 (2020)
4. Alagic, G., Alagic, G., Alperin-Sheriff, J., Apon, D., Cooper, D., Dang, Q., Liu, Y.K., Miller, C., Moody, D., Peralta, R., et al.: Status report on the first round of the nist post-quantum cryptography standardization process (2019)
5. Alhazov, A., Freund, R., Ivanov, S.: P systems with limited number of objects. *Journal of Membrane Computing* **3**(1), 1–9 (2021)
6. Alhazov, A., Sburlan, D.: Static sorting P systems. In: *Applications of Membrane Computing*, pp. 215–252. Springer (2006)
7. Allam, A.M., Abbas, H.M.: Improved security of neural cryptography using don't-trust-my-partner and error prediction. In: *2009 International Joint Conference on Neural Networks*. pp. 121–127. IEEE (2009)
8. Allam, A.M., Abbas, H.M.: On the improvement of neural cryptography using erroneous transmitted information with error prediction. *IEEE transactions on neural networks* **21**(12), 1915–1924 (2010)
9. Cavaliere, M., Ibarra, O.H., Păun, G., Egecioglu, O., Ionescu, M., Woodworth, S.: Asynchronous spiking neural P systems. *Theoretical Computer Science* **410**(24-25), 2352–2364 (2009)
10. Cohn-Gordon, K., Cremers, C., Dowling, B., Garratt, L., Stebila, D.: A formal security analysis of the signal messaging protocol. *Journal of Cryptology* **33**, 1914–1983 (2020)
11. Dierks, T., Allen, C.: *The TLS protocol version 1.0*. Tech. rep. (1999)
12. Dong, T., Huang, T.: Neural cryptography based on complex-valued neural network. *IEEE Transactions on Neural Networks and Learning Systems* **31**(11), 4999–5004 (2019)

13. Ganbaatar, G., Nyamdorj, D., Cichon, G., Ishdorj, T.O.: Implementation of RSA cryptographic algorithm using SN P systems based on HP/LP neurons. *Journal of Membrane Computing* **3**(1), 22–34 (2021)
14. Gheorghe, M., Lefticaru, R., Konur, S., Niculescu, I.M., Adorna, H.N.: Spiking neural P systems: matrix representation and formal verification. *Journal of Membrane Computing* **3**(2), 133–148 (2021)
15. Goldreich, O.: *Foundations of cryptography: volume 2, basic applications*. Cambridge university press (2009)
16. Hassanpour, A., Moradikia, M., Adeli, H., Khayami, S.R., Shamsinejadbabaki, P.: A novel end-to-end deep learning scheme for classifying multi-class motor imagery electroencephalography signals. *Expert Systems* **36**(6), e12494 (2019)
17. Hinze, T., Happe, H., Henderson, A., Nicolescu, R.: Membrane computing with water. *Journal of Membrane Computing* **2**(2), 121–136 (2020)
18. Ionescu, M., Păun, G., Yokomori, T.: Spiking neural P systems. *Fundamenta Informaticae* **71**(2-3), 279–308 (2006)
19. Ipate, F., Lefticaru, R., Mierlă, L., Cabrera, L.V., Han, H., Zhang, G., Dragomir, C., Jiménez, M.J.P., Gheorghe, M.: Kernel P systems: Applications and implementations. In: *Proceedings of The Eighth International Conference on Bio-Inspired Computing: Theories and Applications (BIC-TA)*, 2013. pp. 1081–1089. Springer (2013)
20. Jeong, S., Park, C., Hong, D., Seo, C., Jho, N.: Neural cryptography based on generalized tree parity machine for real-life systems. *Security and Communication Networks* **2021** (2021)
21. Kanter, I., Kinzel, W., Kanter, E.: Secure exchange of information by synchronization of neural networks. *EPL (Europhysics Letters)* **57**(1), 141 (2002)
22. Klimov, A., Mityagin, A., Shamir, A.: Analysis of neural cryptography. In: *International Conference on the Theory and Application of Cryptology and Information Security*. pp. 288–298. Springer (2002)
23. Kumar, A., Garhwal, S.: State-of-the-art survey of quantum cryptography. *Archives of Computational Methods in Engineering* **28**, 3831–3868 (2021)
24. Lv, Z., Yang, Q., Peng, H., Song, X., Wang, J.: Computational power of sequential spiking neural P systems with multiple channels. *Journal of Membrane Computing* **3**(4), 270–283 (2021)
25. Mammone, N., Ieracitano, C., Adeli, H., Morabito, F.C.: Autoencoder filter bank common spatial patterns to decode motor imagery from eeg. *IEEE Journal of Biomedical and Health Informatics* (2023)
26. Mayne, R., Phillips, N., Adamatzky, A.: Towards experimental P-systems using multivesicular liposomes. *Journal of Membrane Computing* **1**(1), 20–28 (2019)
27. Mirzaei, G., Adeli, H.: Machine learning techniques for diagnosis of alzheimer disease, mild cognitive disorder, and other types of dementia. *Biomedical Signal Processing and Control* **72**, 103293 (2022)
28. Mislovaty, R., Perchenok, Y., Kanter, I., Kinzel, W.: Secure key-exchange protocol with an absence of injective functions. *Physical Review E* **66**(6), 066102 (2002)
29. Mosavi, A.H., Mohammadzadeh, A., Rathinasamy, S., Zhang, C., Reuter, U., Levente, K., Adeli, H.: Deep learning fuzzy immersion and invariance control for type-i diabetes. *Computers in Biology and Medicine* **149**, 105975 (2022)
30. Murugappan, M., Murugesan, L., Jerritta, S., Adeli, H.: Sudden cardiac arrest (sca) prediction using ecg morphological features. *Arabian Journal for Science and Engineering* **46**, 947–961 (2021)

31. Nogay, H.S., Adeli, H.: Machine learning (ml) for the diagnosis of autism spectrum disorder (asd) using brain imaging. *Reviews in the Neurosciences* **31**(8), 825–841 (2020)
32. Nogay, H.S., Adeli, H.: Diagnostic of autism spectrum disorder based on structural brain mri images using, grid search optimization, and convolutional neural networks. *Biomedical Signal Processing and Control* **79**, 104234 (2023)
33. Pan, L., Păun, G.: Spiking neural P systems with anti-spikes. *International Journal of Computers Communications & Control* **4**(3), 273–282 (2009)
34. Pan, L., Păun, G., Zhang, G., Neri, F.: Spiking neural P systems with communication on request. *International Journal of Neural Systems* **27**(08), 1750042 (2017)
35. Pan, L., Wang, J., Hoogboom, H.J.: Spiking neural P systems with astrocytes. *Neural Computation* **24**(3), 805–825 (2012)
36. Pan, X., Yang, T., Xiao, Y., Yao, H., Adeli, H.: Vision-based real-time structural vibration measurement through deep-learning-based detection and tracking methods. *Engineering Structures* **281**, 115676 (2023)
37. Păun, G.: Computing with membranes. *Journal of Computer and System Sciences* **61**(1), 108–143 (2000)
38. Peng, W., Wang, B., Hu, F., Wang, Y., Fang, X., Chen, X., Wang, C.: Factoring larger integers with fewer qubits via quantum annealing with optimized parameters. *SCIENCE CHINA Physics, Mechanics & Astronomy* **62**, 1–8 (2019)
39. Pereira, D.R., Piteri, M.A., Souza, A.N., Papa, J.P., Adeli, H.: Fema: A finite element machine for fast learning. *Neural Computing and Applications* **32**, 6393–6404 (2020)
40. Plesa, M.I., Gheoghe, M., Ipate, F., Zhang, G.: A key agreement protocol based on spiking neural p systems with anti-spikes. *Journal of Membrane Computing* pp. 1–11 (2022)
41. Rafiei, M.H., Gauthier, L.V., Adeli, H., Takabi, D.: Self-supervised learning for electroencephalography. *IEEE Transactions on Neural Networks and Learning Systems* (2022)
42. Rescorla, E.: The transport layer security (tls) protocol version 1.3. Tech. rep. (2018)
43. Ruttor, A., Kinzel, W., Kanter, I.: Neural cryptography with queries. *Journal of Statistical Mechanics: Theory and Experiment* **2005**(01), P01009 (2005)
44. Ruttor, A., Kinzel, W., Naeh, R., Kanter, I.: Genetic attack on neural cryptography. *Physical Review E* **73**(3), 036121 (2006)
45. Salguero Dorokhin, É., Fuertes, W., Lascano, E.: On the development of an optimal structure of tree parity machine for the establishment of a cryptographic key. *Security and Communication Networks* **2019** (2019)
46. Shacham, L.N., Klein, E., Mislovaty, R., Kanter, I., Kinzel, W.: Cooperating attackers in neural cryptography. *Physical Review E* **69**(6), 066137 (2004)
47. Shor, P.W.: Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Review* **41**(2), 303–332 (1999)
48. Song, T., Pan, L., Wu, T., Zheng, P., Wong, M.D., Rodríguez-Patón, A.: Spiking neural P systems with learning functions. *IEEE Transactions on Nanobioscience* **18**(2), 176–190 (2019)
49. Song, T., Rodríguez-Patón, A., Zheng, P., Zeng, X.: Spiking neural P systems with colored spikes. *IEEE Transactions on Cognitive and Developmental Systems* **10**(4), 1106–1115 (2017)
50. Song, X., Wang, J., Peng, H., Ning, G., Sun, Z., Wang, T., Yang, F.: Spiking neural P systems with multiple channels and anti-spikes. *Biosystems* **169**, 13–19 (2018)

51. Stypiński, M., Niemiec, M.: Synchronization of Tree Parity Machines using non-binary input vectors. arXiv preprint arXiv:2104.11105 (2021)
52. Stypiński, M., Niemiec, M.: Impact of nonbinary input vectors on security of tree parity machine. In: *Multimedia Communications, Services and Security: 11th International Conference, MCSS 2022, Kraków, Poland, November 3–4, 2022, Proceedings*. pp. 94–103. Springer (2022)
53. Vasile, R., Gheorghe, M., Niculescu, I.M.: *Breaking RSA Encryption Protocol with Kernel P Systems* (2023)
54. Wang, H., Zhou, K., Zhang, G., Paul, P., Duan, Y., Qi, H., Rong, H.: Application of Weighted Spiking Neural P Systems with Rules on Synapses for Breaking RSA Encryption. *International Journal of Unconventional Computing* **15**(1-2), 37–58 (2020)
55. Wang, X., Song, T., Zheng, P., Hao, S., Ma, T.: Spiking neural P systems with anti-spikes and without annihilating priority. *Romanian Journal of Science and Technology* **20**(1), 32–41 (2017)
56. Wu, T., Jiang, S.: Spiking neural P systems with a flat maximally parallel use of rules. *Journal of Membrane Computing* **3**(3), 221–231 (2021)
57. Wu, T., Păun, A., Zhang, Z., Pan, L.: Spiking neural P systems with polarizations. *IEEE Transactions on Neural Networks and Learning Systems* **29**(8), 3349–3360 (2017)
58. Zandron, C., Ferretti, C., Mauri, G.: Solving NP-complete problems using P systems with active membranes. In: *Unconventional Models of Computation, UMC'2K*, pp. 289–301. Springer (2001)
59. Zhang, G., Zhang, X., Rong, H., Paul, P., Zhu, M., Neri, F., Ong, Y.S.: A layered spiking neural system for classification problems. *International Journal of Neural Systems* p. 2250023 (2022)