Private Inference on Layered Spiking Neural P Systems

Mihail-Iulian Pleșa*[1], Marian Gheoghe[2], and Florentin Ipate[1]

[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

Abstract. Layered Spiking Neural P systems (LSN P systems) are a special class of Spiking Neural Networks (SNNs) used to solve classification problems. These types of networks are inspired directly by the biological brains, and their main advantage is low energy consumption when run on neuromorphic hardware. Since this special type of hardware is not yet available on conventional computers, the most convenient method for using an LSN P system to perform data classification is through a cloud platform. This raises privacy concerns for the users since they expose their data to the cloud provider. This paper presents a new privacy-preserving inference protocol for LSN P systems. The protocol allows one party, called the client, to use a pre-trained LSN P system hosted by another party, called the server, without compromising the privacy of the input or the result. The paper also discusses two brute-force attacks on the protocol and shows that the probability that the server compromises the client's confidentiality is negligible.

Keywords: Privacy-preserving · Spiking Neural P systems · Machine learning as a service.

1 Introduction

Spiking Neural Networks (SNNs) are a new type of neural network inspired directly by the biological brain [10]. Unlike current deep learning networks that are trained through gradient descent, SNNs are more efficient in terms of energy consumption when run on neuromorphic hardware [8, 13, 12, 32]. There are several mathematical models of the spiking neuron, the most well-known being leaky integrate-and-fire, Hodgkin–Huxley and Spiking Neural P systems [31, 18, 14]. Layered Spiking Neural P systems (LSN P systems) are a subclass of SN P systems designed to solve classification problems [33]. Although the accuracy of this type of system outperforms other SNN approaches, the main advantage in terms of energy consumption is obtained only if they are run on neuromorphic hardware. A feasible way to use such a platform is through a cloud platform. Among the advantages of this strategy, the most important are reduced costs, flexibility, and scalability [30]. Nevertheless, the major drawback is the privacy compromise on behalf of the user [6]. Consider the following scenario: a pre-trained LSN P system is deployed on a remote server to solve classification problems. A client wants to classify some confidential data using the remote LSN P system. Since privacy is important for the client, it cannot simply upload the data to the server.

We solve this problem by proposing a new privacy-preserving inference protocol for LSN P systems. Our construction allows the client to use a pre-trained LSN P system hosted on a remote server without compromising the privacy of the data it wants to classify. With the proposed construction, the server cannot learn any information about the client's data or the classification result with a non-negligible probability. The security of the protocol is based on additive secret sharing, thus being efficient from a computational point of view [4]. The paper is structured as follows: in Section 2, we present related work on SNN, SN P systems and various privacy-preserving machine learning algorithms; in Section 3, we give a brief overview of the LSN P system, in Section 4 we show our approach on private inference for LSN P system and discuss the complexity of two brute-force attacks, Section 5 is left for the conclusions and further directions of research.

2 Related work

SNNs and SN P systems have many practical applications, which justifies the effort to build methods by which they can be used safely and privately. In [9], the authors used SNNs for EEG classification. In [28], SNN applications for image classifications are discussed. A new approach of series forecasting using SN P systems is presented in [19] while [34] shows how to construct a convolutional neural network using such systems.

Regarding privacy-preserving machine learning, most approaches are based on homomorphic encryption (HE) [1]. The idea of these cryptographic schemes is to allow data processing in encrypted format. In [20], the authors used HE for secure aggregation in swarm learning. In [24] is presented a new privacy-preserving image classification algorithm based on HE. In [11], the authors propose a general method of using an pre-trained neural network to make predictions over encrypted data and [2] presented a protocol that allows multiple parties to train a neural network hosted on a remote server using their local data without revealing it to the server.

In addition to the works stated above, in [7], the authors proposed a guide on how to use homomorphic encryption for bioinformatics applications, and [3] presents one of the first extensive solutions for privacy-preserving machine learning. Also, in [27], the authors did an extensive survey on privacy-preserving methods for machine learning as a service, e.g., machine learning models deployed on a remote server.

There are also papers discussing the privacy aspects of SNNs. In [16], the authors discuss the privacy-preserving weights transfer from a trained artificial neural network to an SNN. In [29], the authors proposed a decentralized learning method for SNNs using federated learning.

We note that classical HE methods are not suitable for constructing privacy-preserving inference for LSN P systems for two reasons [5]. First of all, the HE schemes are defined over a finite field. This is in contradiction to how the potential values of spiking neurons are modeled over a dense set. Secondly, HE schemes do not preserve the operations from the plaintext space in the ciphertext space. Consider, for example, the Paillier cryptosystem [22]. Given a ciphertext and a constant, to compute a ciphertext that encrypts the product between the initial plaintext and the constant, the ciphertext must be raised to the power of the constant. Simply multiplying the ciphertext with the constant will not result in a valid ciphertext, i.e., a ciphertext that encrypts the product between the corresponding plaintext and the constant. The energy efficiency of spiking neural networks relies on the neuromorphic hardware. Changing the configuration of such hardware from performing the operations over the clear data to operations over the ciphertext space implied increased costs [25].

3 Layered Spiking Neural P Systems

LSN P systems were proposed in [33] as a new model for spiking neural networks designed to solve classification problems. There are two types of neurons in the system: proposition neurons and rule neurons. The system is composed of four layers:

1. The input layer has k proposition neurons denoted as σ^1_{pi}, $1 \leq i \leq k$, where k represents the number of features of the input. The spiking rules of these neurons are in the form $r^1_i : E^1/a^{\alpha_i} \to a^{\alpha^i}$ where α_i is the potential value of neuron σ^1_{pi} and $E^1 = \{\alpha_i \geq 0\}$.
2. The hidden layer has n rule neurons denoted as σ^2_{rj}, $1 \leq j \leq n$, where n represents the number of possible classes in which the input can be classified. The spiking rules for these neurons are in the form $r^2_j : E^2/a^{\theta_j} \to a^{\theta^i}$ where θ_j is the potential value of the neuron σ^2_{rj} and $E^2 = \{\theta \geq 0\}$.
3. The comparison layer has one proposition neuron denoted as σ^3_{p1}. This neuron spikes according to the rule $r^3_1 : E^3/a^o \to a^o$ where o is the potential value of the neuron and $E^3 = \{o \geq 0\}$.
4. The output layer has n rule neurons denoted as σ^4_{rj}. The spiking rules for these neurons are in the form $r^4_1 : E^4_j/a^{\theta_j} \to a;\ d_j$ where $E^4_j = \{\theta_j \geq o\}$ and d_j represents a time delay equal to j.

The input of the LSN P system is a vector of real numbers from the interval $[0, 1]$. The potential value of each input neuron σ^1_{pi} is initialized with the corresponding value from the input vector. The output of the system, i.e., the classification results, is the index of the neurons that fire in the output layer. Each layer from the system is fully connected with the next layer through synapses. The weight of the synapse that connects the proposition neuron σ^1_{pi} with the rule neuron σ^2_{rj} is denoted by w^1_{ij}. It is initialized with a value chosen uniformly at random from the interval $[0, 1]$. Apart from the synapses that connect the input layer with the hidden layer, all the other weights are 1. The learning process of an LSN P system involves updating the weights for each sample in the training dataset according to the Widrow–Hoff rule:

$$\boldsymbol{W} \leftarrow \boldsymbol{W} + \eta \left(t - \tilde{t}\right) \boldsymbol{\alpha} \tag{1}$$

where $\boldsymbol{W}$ is the weight matrix between the input and the hidden layer, t represents the output of the system, $\tilde{t}$ denotes the true classification label, and $\boldsymbol{\alpha}$ is the vector of potential values of the input neurons. An overview of the LSN P system is depicted in Figure 1.

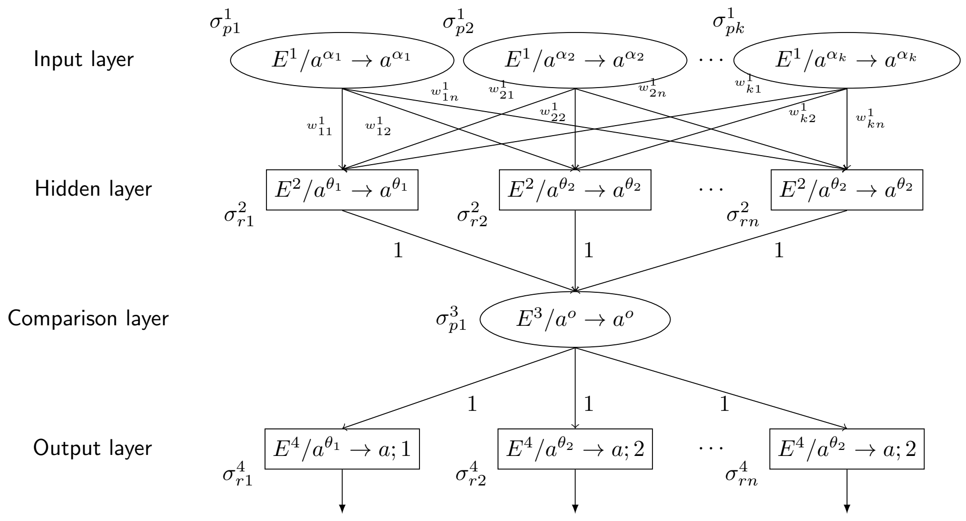

Fig. 1: LSN P system

4 Private inference

In this section, we first present our protocol that allows a client to classify data using a pre-trained LSN P system hosted on a server without compromising the privacy of the data. We then introduce a security analysis from the perspective of a brute-force attack.

4.1 The protocol

The protocol involves two parties, denoted as the client and the server. The server exposes a pre-trained LSN P system thorugh a service. The client wants to use the service to classify data without revealing it to the server. Formally, the main security requirement of the protocol is that the probability of the server learning information about the client's data is negligible [15]. Let $\boldsymbol{\alpha}$ be the client's input. $\boldsymbol{\alpha}$ represents a vector of k potential values modeled as real numbers from the interval $[0, 1]$. We denote the security parameter by r. The protocol goes like follows:

The client side - Phase 1

1. The client use Algorithm 1 to split the input vector $\boldsymbol{\alpha}$ into r shares such as $\boldsymbol{\alpha} \leftarrow \sum_{i=1}^{r} \boldsymbol{\alpha}^{(i)}$.
2. The client randomly generates r vectors of k real numbers from the set $[0, 1]$ denoted as $\boldsymbol{\alpha}^{(r+1)}, \ldots, \boldsymbol{\alpha}^{(2r)}$.
3. The client randomly generates a secret permutation π over the set $\{1, 2, \ldots, 2r\}$.
4. The client sends to server the ordered set $S = \left(\boldsymbol{\alpha}^{(\pi(1))}, \boldsymbol{\alpha}^{(\pi(2))}, \ldots, \boldsymbol{\alpha}^{(\pi(2r))}\right)$.

The server side

1. For each input vector, $\boldsymbol{\alpha}^{(i)}$, $1 \leq i \leq 2r$, the server runs the LSN P system and records the potential values of the neurons $\sigma^4_{r1}, \sigma^4_{r2}, \ldots, \sigma^4_{rn}$ into the vector $\boldsymbol{\theta}^{(i)}$.
2. The server sends the client the ordered set $\left(\boldsymbol{\theta}^{(1)}, \boldsymbol{\theta}^{(2)} \ldots, \boldsymbol{\theta}^{(2r)}\right)$.

The client side - Phase 2

1. Using the secret permutation π the client computes $\boldsymbol{\theta} \leftarrow \sum_{i=1}^{r} \boldsymbol{\theta}^{\left(\pi^{-1}(i)\right)}$.
2. The client determines the classification result, i.e., the label, for the input vector $\boldsymbol{\alpha}$ as the index of the maximum element from the vector $\boldsymbol{\theta}$.

Algorithm 1 Shares generation

```
    Input: $\boldsymbol{\alpha}$
    Output: $\boldsymbol{\alpha}^{(1)}, \boldsymbol{\alpha}^{(2)}, \ldots, \boldsymbol{\alpha}^{(r)}$
1: for i = 1 to r-1 do
2:     $\boldsymbol{\alpha}^{(i)} \overset{\$}{\leftarrow} [0,1]^k$        ▷ Choose $r-1$ random vectors from the set $[0,1]$
3: end for
4: $\boldsymbol{\alpha}^{(r)} \leftarrow \boldsymbol{\alpha} - \sum_{i=1}^{r-1} \boldsymbol{\alpha}^{(i)}$        ▷ Set the last vector such as the sum of the $r$ shares is $\boldsymbol{\alpha}$
```

The protocol is depicted in Figure 2. The following theorem captures the correctness of the protocol:

Theorem 1. *The result of the classification returned by the protocol is the same as the result of the classification returned by the LSN P system on the input* $\boldsymbol{\alpha}$.

Proof. We analyze the running of an LSN P system at each step of its execution on the input $\boldsymbol{\alpha}$.

1. On a input vector $\boldsymbol{\alpha} \in S$, the potential value of the proposition neuron σ^1_{pi} is initialed with $\boldsymbol{\alpha}_i$, $1 \leq i \leq k$.
2. The firing rule $r^1_i : E^1/a^{\boldsymbol{\alpha}_i} \to a^{\boldsymbol{\alpha}_i}$, $E^1 = \{\boldsymbol{\alpha}_i \geq 0\}$ is applied and the proposition neuron σ^1_{pi} sends its potential value to each rule neuron σ^2_{rj}, $1 \leq j \leq n$.
3. Given the set of weights between the input layer and the hidden layer, the potential value of the rule neuron σ^2_{rj} becomes $\boldsymbol{\theta}_j = \sum\limits_{i=1}^{k} w^1_{ij}\boldsymbol{\alpha}_i$.
4. The firing rule $r^j_2 : E^2/a^{\boldsymbol{\theta}_j} \to a^{\boldsymbol{\theta}_j}$, $E^2 = \{\boldsymbol{\theta}_j \geq 0\}$ is applied and the rule neuron σ^2_{rj} seds its potential value to the proposition neuron σ^3_{p1}.
5. The firing rule $r^3_1 : E^1/a^o \to a^o$, $E^3 = \{o \geq 0\}$ where $o = \max(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \ldots, \boldsymbol{\theta}_n)$ is applied and the neuron sends its potential value to each rule neuron σ^4_{rj}.
6. The firing rule $r^4_1 : E^4_j/a^{\boldsymbol{\theta}_j} \to a;\ j$, $E^4_j = \{\boldsymbol{\theta}_j \geq o\}$ is applied and the rule neuron σ^4_{rj} sends a spike at the moment j to the environment if its action potential $\boldsymbol{\theta}_j$ is greater or equal o, i.e., only the rule neuron with the maximal potential value will fire.
7. The result of the classification is the time moment at which one of the output neurons fires, i.e., the index of the rule neuron with the maximal potential value.

Similarly, when the LSN P system is run with the input $\boldsymbol{\alpha}^{(t)}$, the potential value of the rule neuron σ^4_{rj} will be $\boldsymbol{\theta}^{(t)}_j = \sum\limits_{i=1}^{k} w^1_{ij}\boldsymbol{\alpha}^{(t)}_i$.

After the LSN P system is run over the all the inputs $\left(\boldsymbol{\alpha}^{(1)}, \boldsymbol{\alpha}^{(2)}, \ldots, \boldsymbol{\alpha}^{(2r)}\right)$ the client will receive the order set of potential values $\left(\boldsymbol{\theta}^{(1)}, \boldsymbol{\theta}^{(2)}, \ldots, \boldsymbol{\theta}^{(2r)}\right)$ and compute $\boldsymbol{\theta}$ as:

$$\boldsymbol{\theta}_j = \sum_{t=1}^{r} \boldsymbol{\theta}^{(t)}_j = \sum_{t=1}^{r}\sum_{i=1}^{k} w^1_{ij}\boldsymbol{\alpha}^{(t)}_i = \sum_{i=1}^{k} w^1_{ij} \sum_{t=1}^{r} \boldsymbol{\alpha}^{(t)}_i \tag{2}$$

Since $\sum\limits_{t=1}^{r} \boldsymbol{\alpha}^{(t)} = \boldsymbol{\alpha}$, we get:

$$\boldsymbol{\theta}_j = \sum_{i=1}^{k} w^1_{ij}\boldsymbol{\alpha}_i \tag{3}$$

The value $\boldsymbol{\theta}_j$ obtained in (3) by the client represents the potential value of the rule neuron σ^4_{rj} when the LSN P system is run with the input $\boldsymbol{\alpha}$ so the label computed by the client as the index of the maximum value between $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \ldots, \boldsymbol{\theta}_n$ corresponds to the time moment at which the LSN P system spikes with the input $\boldsymbol{\alpha}$. Therefore, the results computed using the protocol correspond to the results obtained directly over the input. □

4.2 Security discussion

The main idea behind the security of the protocol is based on the fact that the server does not know which of the vectors received from the client are shares of the inputs. This is due to the random shuffling based on the secret permutation π in step 4 of the first phase of the client side.

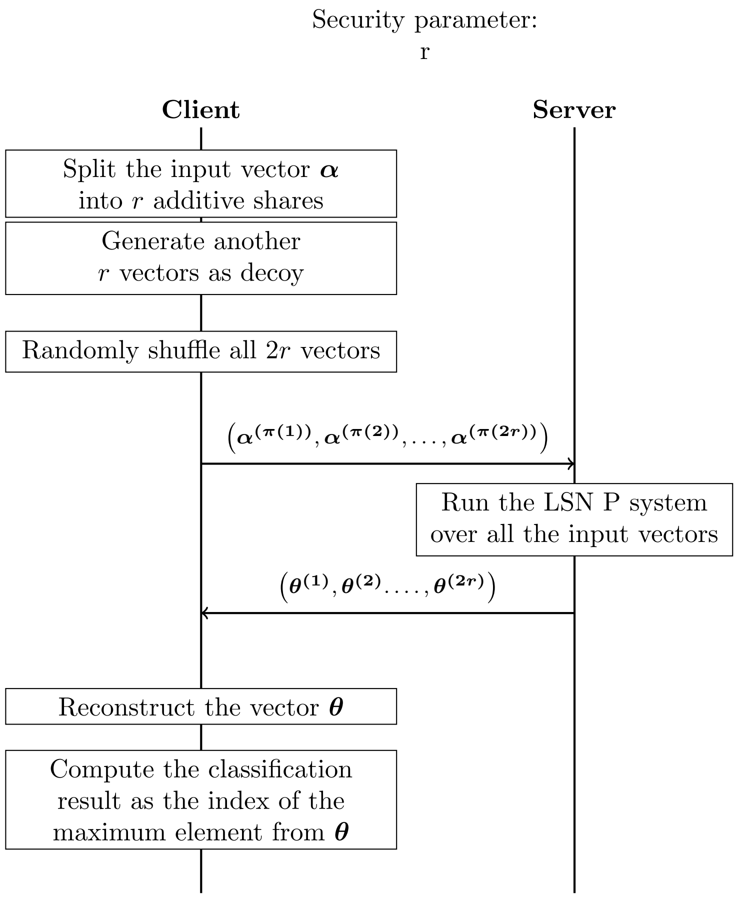
Security parameter: r
Client
Server
Split the input vector $\boldsymbol{\alpha}$ into r additive shares
Generate another r vectors as decoy
Randomly shuffle all $2r$ vectors
$\left(\boldsymbol{\alpha}^{(\pi(1))}, \boldsymbol{\alpha}^{(\pi(2))}, \ldots, \boldsymbol{\alpha}^{(\pi(2r))}\right)$
Run the LSN P system over all the input vectors
$\left(\boldsymbol{\theta}^{(1)}, \boldsymbol{\theta}^{(2)} \ldots, \boldsymbol{\theta}^{(2r)}\right)$
Reconstruct the vector $\boldsymbol{\theta}$
Compute the classification result as the index of the maximum element from $\boldsymbol{\theta}$

Fig. 2: Privacy-preserving inference protocol

There are two ways to attack the protocol from a brute-force perspective. We denote by P_{attack} the probability that the server determines the input of the client. The first way is for the server to try to find out the permutation π, thus determining the positions of the input shares. Since the permutation is considered over a set of $2r$ elements, the probability that the server finds the one generated by the client is:

$$P_{attack} = \frac{1}{(2r!)} \tag{4}$$

The second way is for the server to try every possible combination of r elements from the ordered set received from the client. The number of such combinations is given by $\binom{2r}{r}$; thus, the probability that the server finds the combination that sums the input is:

$$P_{attack} = \frac{1}{\binom{2r}{r}} \tag{5}$$

In both cases, the probability that the server finds the input of the client has a negligible upper bound for $r \geq 1$:

$$P_{attack} \leq \frac{1}{2^r} \tag{6}$$

5 Conclusions and further directions of research

In this paper, we introduced a new privacy-preserving inference protocol for LSN P systems [33]. The protocol allows a client to classify data using a pre-trained LSN P system hosted on a remote server without compromising the confidentiality of the input or the result. The idea of the protocol is to split the input into r additive shares, which are then mixed with other r decoy shares chosen uniformly at random. All the $2r$ shares are then sent to the server to be classified. Since the client is the only one who can distinguish between a real share and a decoy one, he is also the only one who can reconstruct the result. We also discussed two brute-force attacks and showed that the probability that the server determines the input of the client decreases exponentially in r.

The first direction of research is to construct a formal proof of security for the protocol following a cryptographic approach [17]. One approach is to consider a proof by reduction to the problem of the subset sum [26, 21]

The second direction of research is to perform experiments to analyze the running time of the protocol. This is an important consideration when deploying a privacy-preserving protocol for machine learning tasks since most such systems have a large number of users. Therefore, the aspect of scalability is essential. A first version of an LSN P simulator for experiments is provided in [23] as a work in progress.

References

1. Acar, A., Aksu, H., Uluagac, A.S., Conti, M.: A survey on homomorphic encryption schemes: Theory and implementation. ACM Computing Surveys (Csur) **51**(4), 1–35 (2018)
2. Aono, Yoshinori and Hayashi, Takuya and Wang, Lihua and Moriai, Shiho and others: Privacy-preserving deep learning via additively homomorphic encryption. IEEE transactions on information forensics and security **13**(5), 1333–1345 (2017)
3. Bost, Raphael and Popa, Raluca Ada and Tu, Stephen and Goldwasser, Shafi: Machine learning classification over encrypted data. Cryptology ePrint Archive (2014)
4. Chattopadhyay, A.K., Saha, S., Nag, A., Nandi, S.: Secret sharing: A comprehensive survey, taxonomy and applications. Computer Science Review **51**, 100608 (2024)
5. Doan, T.V.T., Messai, M.L., Gavin, G., Darmont, J.: A survey on implementations of homomorphic encryption schemes. The Journal of Supercomputing pp. 1–42 (2023)
6. Domingo-Ferrer, J., Farras, O., Ribes-González, J., Sánchez, D.: Privacy-preserving cloud computing on sensitive data: A survey of methods, products and challenges. Computer Communications **140**, 38–60 (2019)
7. Dowlin, Nathan and Gilad-Bachrach, Ran and Laine, Kim and Lauter, Kristin and Naehrig, Michael and Wernsing, John: Manual for using homomorphic encryption for bioinformatics. Proceedings of the IEEE **105**(3), 552–567 (2017)
8. Galán-Prado, F., Morán, A., Font, J., Roca, M., Rosselló, J.L.: Compact hardware synthesis of stochastic spiking neural networks. International journal of neural systems **29**(08), 1950004 (2019)
9. Ghosh-Dastidar, S., Adeli, H.: Improved spiking neural networks for eeg classification and epilepsy and seizure detection. Integrated Computer-Aided Engineering **14**(3), 187–212 (2007)
10. Ghosh-Dastidar, S., Adeli, H.: Spiking neural networks. International journal of neural systems **19**(04), 295–308 (2009)
11. Gilad-Bachrach, Ran and Dowlin, Nathan and Laine, Kim and Lauter, Kristin and Naehrig, Michael and Wernsing, John: Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In: International conference on machine learning. pp. 201–210. PMLR (2016)
12. Grassmann, C., Anlauf, J.K.: Fast digital simulation of spiking neural networks and neuromorphic integration with spikelab. International Journal of Neural Systems **9**(05), 473–478 (1999)
13. Han, Bing and Roy, Kaushik: Deep spiking neural network: Energy efficiency through time based coding. In: European Conference on Computer Vision. pp. 388–404. Springer (2020)
14. Ionescu, Mihai and Păun, Gheorghe and Yokomori, Takashi: Spiking neural P systems. Fundamenta informaticae **71**(2-3), 279–308 (2006)
15. Katz, J., Lindell, Y.: Introduction to modern cryptography: principles and protocols. Chapman and hall/CRC (2007)
16. Kim, Youngeun and Venkatesha, Yeshwanth and Panda, Priyadarshini: Privatesnn: privacy-preserving spiking neural networks. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 36, pp. 1192–1200 (2022)
17. Kobeissi, N., Nicolas, G., Tiwari, M.: Verifpal: Cryptographic protocol analysis for the real world. In: Progress in Cryptology–INDOCRYPT 2020: 21st International Conference on Cryptology in India, Bangalore, India, December 13–16, 2020, Proceedings 21. pp. 151–202. Springer (2020)
18. Lobo, Jesus L and Del Ser, Javier and Bifet, Albert and Kasabov, Nikola: Spiking neural networks and online learning: An overview and perspectives. Neural Networks **121**, 88–100 (2020)

19. Long, L., Liu, Q., Peng, H., Yang, Q., Luo, X., Wang, J., Song, X.: A time series forecasting approach based on nonlinear spiking neural systems. International Journal of Neural Systems **32**(08), 2250020 (2022)
20. Madni, H.A., Umer, R.M., Foresti, G.L.: Swarm-fhe: Fully homomorphic encryption-based swarm learning for malicious clients. International journal of neural systems p. 2350033 (2023)
21. Nguyen, P., Stern, J.: The hardness of the hidden subset sum problem and its cryptographic implications. In: Advances in Cryptology—CRYPTO'99: 19th Annual International Cryptology Conference Santa Barbara, California, USA, August 15–19, 1999 Proceedings 19. pp. 31–46. Springer (1999)
22. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: International conference on the theory and applications of cryptographic techniques. pp. 223–238. Springer (1999)
23. Plesa, M.I.: LSNP-Simulator. https://github.com/miiip/LSNP-Simulator (2024), accessed: February 28, 2024
24. Rovida, L., Leporati, A.: Encrypted image classification with low memory footprint using fully homomorphic encryption. International Journal of Neural Systems (2024)
25. Schuman, C.D., Potok, T.E., Patton, R.M., Birdwell, J.D., Dean, M.E., Rose, G.S., Plank, J.S.: A survey of neuromorphic computing and neural networks in hardware. arXiv preprint arXiv:1705.06963 (2017)
26. Siqueira, H., Santana, C., Macedo, M., Figueiredo, E., Gokhale, A., Bastos-Filho, C.: Simplified binary cat swarm optimization. Integrated Computer-Aided Engineering **28**(1), 35–50 (2021)
27. Tanuwidjaja, Harry Chandra and Choi, Rakyong and Baek, Seunggeun and Kim, Kwangjo: Privacy-preserving deep learning on machine learning as a service—a comprehensive survey. IEEE Access **8**, 167425–167447 (2020)
28. Vaila, R., Chiasson, J., Saxena, V.: Deep convolutional spiking neural networks for image classification. arXiv preprint arXiv:1903.12272 (2019)
29. Venkatesha, Yeshwanth and Kim, Youngeun and Tassiulas, Leandros and Panda, Priyadarshini: Federated learning with spiking neural networks. IEEE Transactions on Signal Processing **69**, 6183–6194 (2021)
30. Wang, X., Wang, Y., Cui, Y.: Energy and locality aware load balancing in cloud computing. Integrated Computer-Aided Engineering **20**(4), 361–374 (2013)
31. Wang, Z., Guo, L., Adjouadi, M.: A generalized leaky integrate-and-fire neuron model with fast implementation method. International journal of neural systems **24**(05), 1440004 (2014)
32. Young, A.R., Dean, M.E., Plank, J.S., Rose, G.S.: A review of spiking neuromorphic hardware communication systems. IEEE Access **7**, 135606–135620 (2019)
33. Zhang, Gexiang and Zhang, Xihai and Rong, Haina and Paul, Prithwineel and Zhu, Ming and Neri, Ferrante and Ong, Yew-Soon: A layered spiking neural system for classification problems. International journal of neural systems **32**(08), 2250023 (2022)
34. Zhou, C., Ye, L., Peng, H., Liu, Z., Wang, J., Ramirez-de Arellano, A.: A parallel convolutional network based on spiking neural systems. International Journal of Neural Systems (2024)