Received June 28, 2021, accepted August 25, 2021, date of publication September 1, 2021, date of current version September 15, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3109635

Neural Networks-Based Cryptography: A Survey

ISHAK MERAOUCHE[1], (Member, IEEE), SABYASACHI DUTTA[2], HAOWEN TAN[3], (Member, IEEE), AND KOUICHI SAKURAI[4], (Member, IEEE)

[1]Graduate School of Information Science and Electrical Engineering, Kyushu University, Fukuoka 819-0395, Japan
[2]Department of Computer Science, University of Calgary, Calgary, AB T2N 1N4, Canada
[3]Cyber Security Center, Kyushu University, Fukuoka 819-0395, Japan
[4]Department of Information Science and Electrical Engineering, Kyushu University, Fukuoka 819-0395, Japan

Corresponding author: Ishak Meraouche (meraouche.ishak.768@s.kyushu-u.ac.jp)

The work of Ishak Meraouche was supported by the Ministry of Education, Culture, Sports, Science and Technology (MEXT) through Kyushu University. The work of Sabyasachi Dutta was supported by the National Institute of Information and Communications Technology, Japan, through Kyushu University, during 2018–2019. This work was supported in part by the Collaboration Hubs for International Program (CHIRP) of Strategic International Collaborative Research Program (SICORP) through Japan Science and Technology Agency (JST).

ABSTRACT A current trend of research focuses on artificial intelligence based cryptography which although proposed almost thirty years ago could not attract much attention. Abadi and Anderson's work on adversarial cryptography in 2016 rejuvenated the research area which now focuses in building neural networks that are able to learn cryptography using the idea from Generative Adversarial Networks (GANs). In this paper, we survey the most prominent research works that cover neural networks based cryptography from two main periods. The first period covers the oldest models that have been proposed shortly after 2000 and the second period covers the more recent models that have been proposed since 2016. We first discuss the implementation of the systems from the earlier era and the attacks mounted on them. After that, we focus on post 2016 era where more advanced techniques are utilized that rely on GANs in which neural networks compete with each other in order to achieve a goal e.g. learning to encrypt a communication. Finally, we discuss security analysis performed on adversarial cryptography models.

INDEX TERMS Cryptography, deep learning, neural networks, generative adversarial networks.

I. INTRODUCTION

With the rapid expansion of communication through networks among multiple terminals (computers, smartphones etc.), it is stringent to develop technologies to protect the information exchanged in those networks. Often, when one device communicates with one or more devices, a cryptographic protocol is applied to encrypt all the transmitted data in order to protect the communication(s). Two kinds of cryptographic protocols are typically considered in the literature: one to establish a common secret key, and the other to encrypt the messages exchanged. In the practical applications, the lightweight and secure protocols are highly desired especially for some terminals with low performance, such as devices with limited battery-lives. To meet the application requirements, cryptography is consistently evolving through time according to the extensive development and improvement on the security of cryptographic protocols. Among others, the RSA cryptosystem [1] is widely used as a standard for public-key encryption and digital signature; and the Rijndael algorithm (also known as AES) [2] for symmetric encryption.

Machine learning plays a major role in cryptanalysis, a subdomain of cryptology [3]–[5]. Roughly speaking, cryptanalysis aims to test and analyze the security of cryptographic protocols by feeding different inputs to the cryptographic algorithm and analyzing the outputs in order to find a common or repetitive pattern in the outputs that might help find the secret key or even decrypt the ciphertext without access to the key. Machine learning can help learn from the data generated by the cryptographic algorithm and detect significant patterns [5]–[7]

In late 90's and early 2000's, several cryptographic protocols using machine learning and deep learning models were

The associate editor coordinating the review of this manuscript and approving it for publication was Muhammad Afzal.

proposed such as [8]–[10], but were deemed insecure and even some concrete attacks [11] were shown subsequently. The interest in neural network based cryptography took a dip because of the fact that simple computations, even as basic as exclusive-or (XOR) operation could not be computed by simple neural networks.

However recently, Abadi and Andersen [12] initiated a research direction on learning to protect communications with adversarial neural cryptography. Specifically, it aims to create neural networks that can learn to encrypt a communication without being taught any specific encryption algorithm. This technique is based on generative adversarial networks (GANs), in which neural networks try to achieve a goal in the presence of an adversary (i.e. another neural network) by pitting against each other [13]. The main idea behind GANs is to have two neural networks competing in order to generate a new set of data that can be taken as the real data. GANs are powerful in their ability of mimicking various types of data, and hence broadly used especially in image and voice generation [14]–[16] to generate synthetic data which are indistinguishable from the true data distribution. Following Abadi and Andersen's work [12], a flow of research appeared in order to study the security of their model (e.g. [17]), as well as extend it to an assumed perfectly secure protocol [18], and many more [19]–[22].

In this paper, we aim to survey the recent progress on neural networks based cryptography, how it evolved since the late 90s, explain the model proposed by Abadi and Andersen [12] and how it learns to encrypt a communication. We will also see how other researchers [19], [20] ported this model to steganography. Finally, we will evaluate the security of the model proposed by Abadi and Andersen [12] based on the security analysis done by Zhou *et al.* [17]. We will also see how it was improved by Coutinho *et al.* [18] and Li *et al.* [23].

The remainder of this paper is organized as follows. In Section II, we review the technical terms and techniques that are used in this survey by giving the essential terminology on neural networks, deep learning (DL), generative adversarial neural networks (GANs) and how neural networks work in general.

In section III, we discuss the Tree Parity Machine which is deemed to be the first work in neural networks based cryptography [9]. We will see how it was broken in [11] and some of its improvements especially in [24]. Other works will also be discussed. We will then discuss the GANs based encryption technique [12] and how the neural networks learn to encrypt the communication as well as some follow up works [19], [20], [22].

In Section IV, we will analyse the security of the GANs based encryption model proposed by Abadi and Andersen [12]. The security analysis is focused on the randomness of the ciphertext in order to see if it reveals any information about the key. We will finally see how it was improved by Coutinhou *et al.* [18].

Finally, in Section V, we give a conclusion on the survey as well as possible future research paths.

II. BACKGROUND

In this section we introduce the background material that will be used in the later sections. We begin with some basic terminologies.

A. CRYPTOGRAPHY

Cryptography's main aim is towards data protection and communication security. Cryptographic protocols are designed in a way that only the authorized parties are able to join read the communication.

In the early days, the main focus of cryptology was to design systems related to secure encryption schemes and their analysis. However with the massive growth in communications, the field has acquired new sets of techniques and protocols to make encryption tasks more reliable and less dependent on physically meetings to exchange an encryption/decryption key or to change it.

In terms of security, cryptography can be broadly divided into two main models – information theoretic security and computational security. In the former model the adversary, against whom a cryptographic protocol is supposed to ensure security, is taken to be computationally unbounded and in the latter one the adversary is assumed to be bounded with respect to its computational power. We make a note of the fact that any cryptographic primitive providing information theoretic security does not depend on any kind of hardness assumption and hence cannot be broken (in a provable manner) even with unlimited computing power. On the other hand, computationally secure primitives are based on *hardness* assumptions e.g. *integer factorization, discrete log computation* where the security is based on the infeasiblility of obtaining any "practical" algorithm to break the hardness problem(s).

Among several important existing cryptographic primitives our main area of focus in this paper will be on key exchange, symmetric key encryption and steganography.

1) ONE TIME PAD

One time pad (OTP) is a symmetric key encryption technique which requires an n bit message to be xor-ed with a uniform n bit key to compute the ciphertext. The recipient who is already in possession of the n bit key can recover the message. It can be observed that this primitive is an information theoretically secure encryption scheme. However, OTP suffers from some serious drawbacks – size of the secret key has to be same as the message as well as the key has to be uniformly distributed over the key space and that the key cannot be reused. For every message an independently chosen key has to be used and this makes the scheme impractical to implement.

B. ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

Artificial Intelligence is the domain that aims to build robots and computer software that are able to mimic the human behavior. An AI software or robot can be either explicitly programmed with a big stack of conditions and actions or it can be a self learning program that learns how to do or mimic specific tasks.

Machine learning is a subset of AI that creates computer programs that learn to do a specific task by building a model from many observable examples. For example, in the case of machine learning, an algorithm builds a model based on the features given and based on those features, the model can perform predictions, recognition or actions depending on the task. Machine learning also contributes in the domain of cryptanalysis – researchers can, for example, use machine learning algorithms to detect patterns in ciphertexts that can help break the encryption technique or find flaws inside it.

1) CLASSIFICATION TASKS

Classification tasks are considered to be one of the most widely used techniques in machine learning. The machine learning algorithm aims to predict the class of a given input based on its training on already observed example-inputs. For example, a machine learning algorithm that can classify an input picture of a person into a class of facial expressions – the class of facial expressions can be $\{smile, laugh, cry, \ldots\}$ and the goal of the algorithm would be to assign each picture to its correct class.

Another important example of classifications tasks is spam detection, the machine learning algorithm learns to classify an e-mail to be either spam or not. Machine learning classification tasks are countless – besides spam detection and image recognition some applications are used in our daily life such as credit approval, advertising, etc.

2) NEURAL NETWORKS

Neural Networks are one of the building blocks of machine learning algorithm and are inspired from the structure human brain which is composed of a large number of neurons connected to each other and messages (signals) transit through each of them.

The basic structure of neural networks are organized into layers as shown in Figure 1. The first layer (Input Layer) is composed of the neurons that read the data without changing it; The second layer (Contains one or more hidden Layer) is composed of the neurons processes the data. Lastly, the final layer is the output layer and generally the activated neuron in the output layer is the decision, action or recognition made by the neural network. It can either have a single neuron which contains a value or have multiple neurons where each neuron represents a class or a possibility. In the case of an output layer with multiple neurons, each neuron represents will have a value and usually the neuron with the higher value is the neuron activated.

When dealing with machine learning, we only use one hidden layer. But when dealing with deep learning, there is more than one hidden layer. Each layer will have a specific task such as features extraction, data processing, etc.

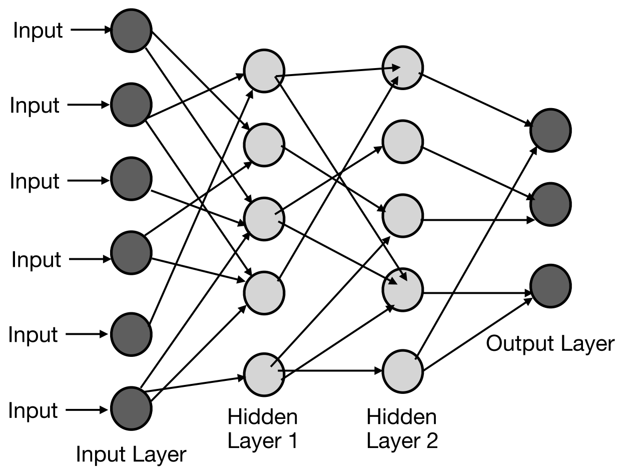


FIGURE 1. General Structure of Neural Networks.

3) SUPERVISED LEARNING AND UNSUPERVISED LEARNING

To train a machine learning algorithm, there are two main methods: Supervised Learning and Unsupervised Learning.

In supervised learning, the model is trained with data that is labeled. This means that the training data is already tagged with the correct answer and the neural networks compare their prediction with the correct answer. Supervised learning is mainly useful when trying to predict, expect or foresee a behavior or event based on previous data. For example training a neural network to test if a person can get a credit or loan based on his/her credit payment history.

However in unsupervised learning the training is done without any labeled data: extraction is done on deterministic features from the data before processing it. Nearest neighborhood algorithm is one such example of unsupervised learning.

C. TYPES OF NEURAL NETWORKS

There are several different types of neural networks and each are designed for a specific target. We discuss the most used ones in the following.

1) FEEDFORWARD NEURAL NETWORKS

This is one of the simplest types of neural networks. In a feedforward neural network, the data passes through the different input nodes till it reaches the output node.

This means that data moves in only one direction from the first tier until it reaches the output node. This is also known as a front propagated wave which is usually achieved by using a classifying activation function.

2) CONVOLUTIONAL NEURAL NETWORKS (CNN)

These neural networks are mostly used in image and video editing, natural language processing (NLP) and recommendation systems as CNNs produce very efficient results.

A CNN contains one or more convolutional layers that can either be completely connected or pooled. Before passing the data to the next layer, the convolutional layer uses a convolutional operation on it.

TABLE 1. Summary of the contributions in GANs based cryptography.

YEAR	WORK	SUMMARY OF CONTRIBUTION	TECHNIQUE	SUMMARY OF VULNERABILITIES
2016	Learning to Protect Communications With Adversarial Neural Cryptography [12].	Use the idea of GANs for 2-party secure communication.	Two neural networks synchronize and build a secure communication in the presence of an eavesdropper.	Vulnerable against probabilistic attacks. The ciphertexts generated have many repetitive patterns.
2017	Generating Steganographic Images via Adversarial Training [20].	Uses the same model in [12] to learn steganography instead of encryption.	Two neural networks learn steganography in the presence of an eavesdropper that learns to distinguish between cover and steganographic images.	As it is based on the model by Abadi *et al.* [12], it is also weak against probabilistic attacks however the authors show good results against known steganalysers.
2018	Security Analysis and New Models on the Intelligent Symmetric Key Encryption [17].	Prove that the model learned in [12] is weak against probabilistic attacks.	The authors generate multiple ciphertexts and test their randomness using different techniques.	Not Applicable.
2018	Learning Perfectly Secure Cryptography to Protect Communications with Adversarial Neural Cryptography [18].	The authors change the structure of the neural networks so that they learn the One-Time Pad which is secure against probabilistic attacks.	The two neural networks learn are given a set of keys and plaintexts.They learn to XOR each plaintext with a unique key from the set.	The one time pad is based on a pseudo-random generator which still makes it vulnerable when not using quantum computers.
2018	Neural Cryptography Based on the Topology Evolving Neural Networks [22].	The authors investigate whether neural networks in Spectrum-Diverse Neuroevolution with Unified Neural Models [22] can learn encryption with the same training process by Abadi *et al.* [12].	Similarly to [12], the authors implement the same model but by using Spectrum-Diverse Neuroevolution with Unified Neural Models [22] and train the neural networks.	The authors do not provide any security analysis as their work focused on establishing a successful communication.

D. GENERATIVE ADVERSARIAL NEURAL NETWORKS (GANs)

Generative Adversarial Networks [13], or GANs for short, are an approach to generative modeling using deep learning methods, such as convolutional neural networks.

Generative modeling is an unsupervised learning task that involves automatically discovering and learning the regularities or patterns in input data in such a way that the model can be used to generate or output new examples that plausibly could have been drawn from the original data.

GAN has proven to be a useful approach to build cryptographic tools in presence of a neural network considered as an adversary.

One should note however that while adversarial learning using GANs is generally used for image processing, it is not limited to that. GANs-Based cryptography is an example of that. Another example is by Dash *et al.* [25] where the authors investigate whether it is possible to apply adversarial neural networks for playing the popular hide-and-search board game called Scotland Yard. The authors show that neural networks can indeed learn to assess the game like humans and find the hider.

III. NEURAL NETWORKS BASED CRYPTOGRAPHY

Deep Learning based cryptography is a fairly new way of doing cryptography. While first attempts to design cryptographic protocols based on machine learning were implemented in the late 90s, the security was not satisfying. The main idea was to use make neural networks learn a specific cryptographic task. For example, use two neural networks and train them to learn how to exchange a key or encrypt and decrypt sequences of data. This is different from common methodologies where algorithms are explicitly implemented to perform the specified task.

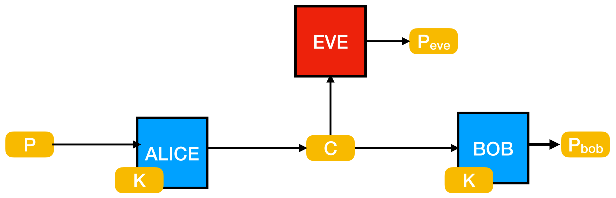


FIGURE 2. 2-party symmetric communication general scheme.

As mentioned above, deep learning based cryptography is fairly new, with very few research works especially before the development of GANs and the advances in deep learning. One of the first papers [9] related to this research direction was back in 2000, showcasing a secure key exchange through the synchronization of two neural networks. However the model was proven to be vulnerable by Klimov *et al.* [11]. After the development of GANs, a research spurring paper appeared in late 2016, featuring two neural networks learning a symmetric key encryption system in the presence of an adversary [12]. Several follow up works performed a study on the security [17], ported it to steganography [19], [20] or improved its security [18]. More details and a security analysis are summarized in Table 1 and explained in Section IV.

Figure 2 shows the default setup of a secure symmetric encryption between two parties. Two parties Alice and Bob

TABLE 2. Summary of contributions in Neural Networks based Cryptography.

YEAR	WORK	SUMMARY OF CONTRIBUTION	TECHNIQUE	SUMMARY OF VULNERABILITIES
2001	Secure Exchange of Information by Synchronization of Neural Networks [9]	Fast synchronization of neural networks to agree on a secret key.	Two neural networks mutually synchronize and end up with the same weights vector that is used as a key.	Broken in [11] using three technique: Passive, Geometric and Probabilistic attack. (See the row below)
2002	Analysis of Neural Cryptography [11].	Shows that the work done in [9] is vulnerable against attacks.	There are three different attacks: passive, geometric and probabilistic. The attacks are detailed in Section III-A	Not Applicable.
2007	Dynamics of neural cryptography [27]	Show that the geometric attack shown in [11] can be overcome.	Shows that increasing the number of neurones increases the complexity of the geometric attack exponentially.	Not Applicable.
2010	Permutation parity machines for neural cryptography [28].	Improves the TPM by building the Permutation Parity Machine (PPM).	The PPM has the same structure as the TPM but its weights take different values.The authors claimed it robust against probabilistic attacks.	Broken in [29]. (See the row below)
2012	Successful attack on permutation-parity-machine based neural cryptography [29].	Proves that the PPM [28] can be vulnerable against probabilistic attacks.	An attacker that simulates the space of possible weight vectors using the Monte-Carlo probabilistic method can predict the final weight vector before the two parties.	Not Applicable
2019	On the Development of an Optimal Structure of Tree Parity Machine for the Establishment of a Cryptographic Key [24].	Improves the security of the TPM.	Looks for the optimal values for the strcture of the TPM. Results a 0.00004% probability of successful passive attacks and 0% on geometric attacks.	No insights regarding probabilistic attacks provided.
2020	3D CUBE Algorithm for the Key Generation Method: Applying Deep Neural Network Learning-Based [30].	Generates a mutual secret key between two parties.	Both parties shuffle a 3D cube using the same algorithm and if the patterns match, a key is generated using XOR and Hash operations.	No insights regarding attacks provided.

share a secret key K, the Encryption/Decryption algorithm is known to all including the Eavesdropper Eve that is listening to the communication but cannot replay messages. When Alice wants to send a message P, she inputs it to the algorithm along the secret key K in order to encrypt it. The cipher text C is the output of the algorithm and will be sent publicly to Bob who will use the decryption algorithm in order to decrypt the ciphertext C using the same key K that have been used during encryption by Alice.

A common problem in this kind of communications is how to share the secret key K without having to meet physically.

There are many classical cryptography methods to share a secret key between two parties e.g. the Diffie-Hellman Key Agreement Protocol [26]. One can also use public key encryption protocols such as RSA [1] to encrypt a secret key and send it to the recepient. However our focus will be on neural networks based protocols.

One very popular protocol by Kanter *et al.* [9] was proposed during the year 2001 and showed how two neural networks can learn to exchange a secret key without using any sort of known cryptography methods. The mechanism will be discussed in Section III. Table 2 highlights the most prominent works.

A. SECURE EXCHANGE OF INFORMATION BY SYNCHRONIZATION OF NEURAL NETWORKS

Kanter *et al.* [9], were among the first researchers to make use of machine learning to learn cryptography. In their case, it was to perform a key agreement between two neural networks.

The idea consists of having two neural networks called Tree Parity Machines (TPMs) and synchronize them to convey on a key securely in the presence of passive eavesdroppers that have access to the communication but cannot change or replay messages.

The structure of the two neural networks considered in [9] consists of three layers. A single-neuron output layer, K hidden neurons and $K \cdot N$ input neurons as shown in Fig. 3.

The leading party (Alice) starts with generating a random input of size N and shares it publicly with the other party (Bob). They both pass them through their neural network and get the output O. They compare their outputs and if they are equal then the two neural networks are said to be synchronized (have the same weights) and can use their weights vector W as a secret key [9].

However shortly after this proposal, Klimov *et al.* [11] three working methods that can break the protocol. We describe the three attacks in the following.

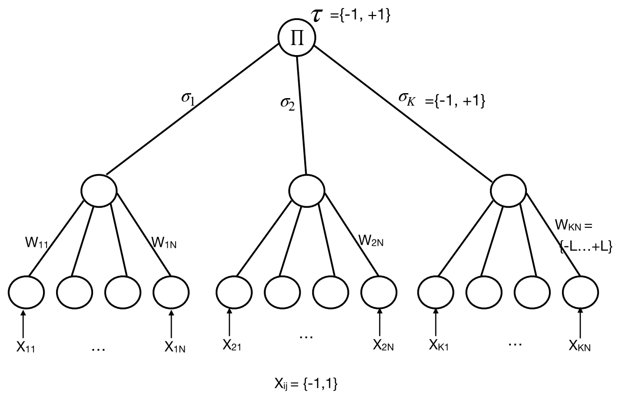


FIGURE 3. Neural Network structure of the parity tree machine.

B. ANALYSIS OF NEURAL CRYPTOGRAPHY

Klimov *et al.* [11] analysed and showed that the work done by Kanter *et al.* [9] is insecure and impractical as it is vulnerable against different attacks that are explained below.

1) THE GENETIC ATTACK

This attack looked at the two neural networks from the biological point of view to make an attack using genetic algorithms. The general idea is to simulate a big population of neural networks that have the same structure as Alice and Bob and train them with same public inputs. The neural networks from the population whose outputs are similar to the two targeted neural networks, are now synchronised with the targets and can read the communication between them.

2) THE GEOMETRIC ATTACK

In this attack, the authors simulate each input of the two target neural networks as a K random hyperplanes $X_1, \ldots, X_K$ corresponding to K perceptrons and the weights of each neural network as K points $W_1 \ldots W_K$ in the N-discrete space $U = \{-L, \ldots, L\}^N$ where $W_i = (w_{i1}, \ldots, w_{iK})$. Concretely, an attacker constructs a neural network with random weights but with the same neural network structure as the target and at each step of training, the weights are updated according to these rules:

- If the two target neural networks have different outputs, the attacker does not update his weights.
- If the two target neural networks have the same outputs and the attacker also has the same output, the neural network's weights will be updated in the normal way.
- If the two target neural networks have the same outputs but not the attacker, then the attacker should find an i_0 that minimises this formula: $\left|\sum_{j=0}^{N} w_{ij}^C \cdot x_{ij}\right|$ and updates the weights assuming the hidden bits and the target's outputs.

The authors of [9] however conducted a study on the geometric attack in [27] to prevent it. In their study, they deducted that neural networks with a larger value N of the hidden units will increase the complexity of the geometric attack exponentially and therefore render it difficult to conduct. Brute force attacks and similar attacks are also affected by the size of N.

3) THE PROBABILISTIC ATTACK

In the probabilistic attack, the attacking Tree Parity Machine is actually a probabilistic Tree Parity Machine this means that the weights are actually probabilistic weights $p_{i,j}(l) = l \in [-L, +L]$ where each probabilistic weight is a probabilistic distribution that represents the probability of the Tree Parity Machine A taking l as a parameter. Then, by passively eavesdropping the inputs $x_{i,j}$ the attacker can use either the Hebbian learning rule or the Monte-Carlo method to update $p_{i,j}(l)$ and end up with identical weights to the parties communicating. This is mainly due to the limited possible values in $[-L, +L]$ which makes them easy to simulate.

The work done in [9] was improved by Prabakaran *et al.* [31]. The authors worked on a solution for the probabilistic attacks that were used before to break [9]. In order to improve the security and remove the possibility of an attacker passively synchronizing, they introduced queries: instead of generating random inputs, Alice and Bob generate (in turn) at every iteration a set of inputs that is correlated to their respective weights, by doing this, the probability of an attacker passively synchronizing is low because the input is either linked to Alice's weights or Bob's weights. The inputs are generated using a specific algorithm and do not reveal much information about the weights of the neural network and allow to have a mutual influence between A and B which highly reduces the probability of a successful passive attack.

C. IMPROVEMENTS TO THE TREE PARITY MACHINE

The tree parity machine [9] has seen several improvements and attacks since it was first introduced.

One of those improvements is the work done by Reyes *et al.* [28] where the authors transformed the Tree Parity Machine into a Permutation Parity Machine (PPM) to improve the security.

A Permutation Parity Machine has the same overall neural network structure as a Tree Parity Machine; however the number of parameters and their values are different from the TPM.

A Permutation Parity Machine is defined as a neural network with K hidden units just like the Tree Parity Machine. These units are simple perceptrons (neurons) each having its own input. There are N units with N inputs that take binary values (either 0 or 1).

As for the weights W, they are drawn from a state vector $S \in \{0, 1\}^G$ where G must be greater than $K \cdot N$.

The i^{th} hidden units are calculated using an exclusive or between the weight w and the input x. The final output is either 1 or 0.

Reyes *et al.* [28] conduct comparative attacks on the TPM and the PPM. The results show that the Permutation Parity Machine performs better against the attacks proposed by Klimov *et al.* [11] compared to the Tree Parity Machine. The authors then demonstrate that the probability order of a successful attack on the Permutation Parity Machine can be as low as 10^{-20} when the value of N is equal to 16 and the value of G is equal to 128.

The probability of a successful attack is demonstrated to be dependent on the value of G by the following formula: $P_E = \frac{1}{2^{G-1}}$.

We can see that with $G = 128$ we have $P_E = 10^{-20}$.

The result is therefore lighter than the method proposed in [27] as they use a value of 1000 for N which will significantly increase the synchronization time and resources usage compared to this method.

However Seoane *et al.* [29] demonstrate a successful probabilistic attack on the Permutation Parity Machine which therefore renders the PPM discussed by Reyes *et al.* [28] non-secure.

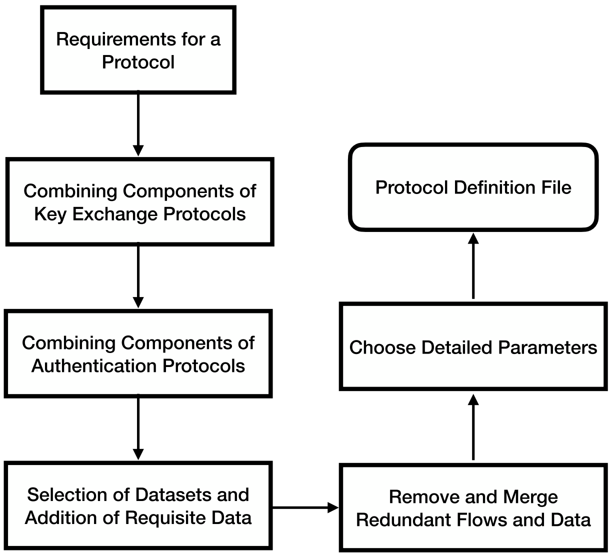
Another improvement to the Tree Parity Machine has been done by Salguero *et al.* [24]. They studied the original TPM and proposed an optimal structure that generated a 512 bits key. This was done by doing over 10 million simulations with different parameters and neural network sizes. All of these simulations were accompanied by a passive adversary trying to synchronize in a passive way along Alice and Bob. In their simulations, the authors showed a case where the neural networks synchronize in a maximum of 6 seconds with a 0% success rate for the attacker. This was done by using the values $K = 8$, $N = 16$ and $L = 2^3$ for the structure of the TPM. The authors finally validate their results with the heuristic rule and the results show that a small change in the parameters would lead to a polynomial increase of the synchronization time and therefore the authors presume that their method is secure enough.

D. AUTOMATIC SECURITY PROTOCOL GENERATION

Another variant of "self-learned" cryptography is automatic generation of secure protocols or automatic security verification for protocols. Basically, these protocols are algorithm-based and do not rely on deep learning or machine learning. The reason they are stated here is because they mimic human behavior by generating or evaluating security protocols which is considered as an Artificial Intelligence Behavior. The algorithm generally has as input the security requirements for the entity and then the algorithm generates a protocol that is conform to the security requirements. A good example of protocol generation is done by Kiyomoto *et al.* [32]. Another work by Ota *et al.* [33] showcases automatically verifying the security in exchange schemes in a 3-party scenario. Figure 4 shows the workflow for generating a security protocol in the model proposed by Kiyomoto *et al.* [32].

E. 3D CUBE ALGORITHM FOR THE KEY GENERATION METHOD

Jin and Kim [30] proposed a novel method where two parties mutually generate the same secret key. The two parties synchronize by shuffling and solving a 3D cube using a neural network model. The patterns obtained during the shuffles are combined with some XOR operations to obtain the key.



Requirements for a Protocol
Combining Components of Key Exchange Protocols
Combining Components of Authentication Protocols
Selection of Datasets and Addition of Requisite Data
Remove and Merge Redundant Flows and Data
Choose Detailed Parameters
Protocol Definition File

FIGURE 4. Workflow for generating a security protocol Kiyomoto *et al.* [32].

F. LEARNING TO PROTECT COMMUNICATIONS WITH ADVERSARIAL NEURAL CRYPTOGRAPHY

Abadi-Anderson [12] were the researchers that spurred the research in the adversarial cryptography topic. Their model shows how to train two neural networks in a GAN setup to learn a symmetric encryption protocol without being taught any algorithm.

The model consists of two neural networks (Alice and Bob) sharing a secret key k and their goal is to establish a secure communication in the presence of an adversary, the third neural network (Eve).

Alice and Bob's goal is to communicate securely by minimizing the error between the original plaintext and Bob's deciphered output text. Eve's goal is to reconstruct the plaintext using the cipher text only i.e. without knowing the secret key.

While in the setup of a GAN, Eve's goal would be to distinguish between the cipher text C and a random value from a certain distribution; Her goal here is the reconstruction of the plaintext from the ciphertext only. It does not matter if the cipher text contains some meta data that proves that it comes from a certain plaintext.

The setup of the neural networks is the same as in Figure 2 and the training process is separated in two phases that are explained below:

1) TRAINING PHASE

In this phase, a random key K and a random plaintext P are generated at every iteration. The key is known to Alice and Bob but not Eve. P and K are fed into Alice's neural network which is a series of convolutions and activations in order to transform the plaintext as shown in figure 5. The

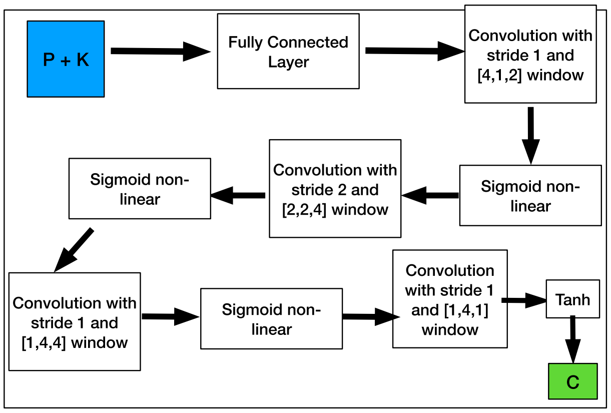

P + K
Fully Connected Layer
Convolution with stride 1 and [4,1,2] window
Sigmoid non-linear
Convolution with stride 2 and [2,2,4] window
Sigmoid non-linear
Convolution with stride 1 and [1,4,4] window
Sigmoid non-linear
Convolution with stride 1 and [1,4,1] window
Tanh
C

FIGURE 5. Encryption flow through Alice's neural network [12].

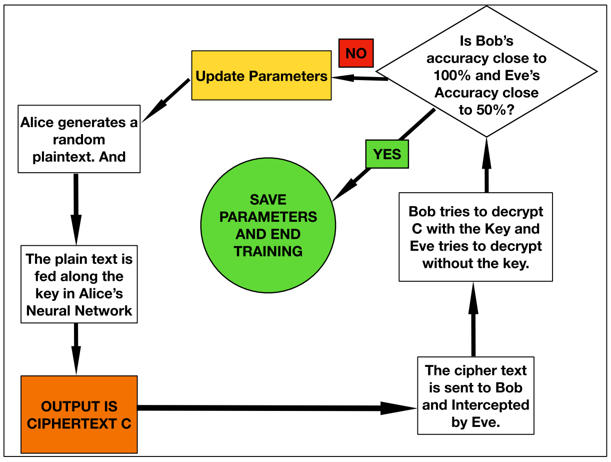

Alice generates a random plaintext. And
The plain text is fed along the key in Alice's Neural Network
OUTPUT IS CIPHERTEXT C
The cipher text is sent to Bob and Intercepted by Eve.
Bob tries to decrypt C with the Key and Eve tries to decrypt without the key.
Is Bob's accuracy close to 100% and Eve's Accuracy close to 50%?
NO
Update Parameters
YES
SAVE PARAMETERS AND END TRAINING

FIGURE 6. Diagram showing the training process of the neural networks [12].

output of Alice's neural network (the ciphertext C) will be fed into Bob's neural network along the key K. Bob will have the same neural network structure as Alice and will use the same series of convolutions and activations to decrypt C and output P_{bob}. An additional malicious neural network called Eve with the same structure as Alice and Bob will eavesdrop the communication all the time and use every C that she intercepts as a input to her neural network to output P_{eve}. The neural networks are trained until Bob's accuracy is as close as possible to the original plaintext and Eve's output is around 50%. The reason why Eve's accuracy needs to be around 50% is because in probabilities, when an entity is making random guesses; the worst case scenario is to be 50% wrong and 50% correct. In that case, you cannot tell which guesses are correct and which are not. In the case of Eve, she cannot know which bits are correct and which are not. Assuming if she was trained to be 100% wrong, she can just flip the bits and become 100% correct. The model used is the same as in Figure 2. Figure 6 summarizes the training process. The loss function used to train the neural networks is as follow:

$$L_B(\theta_A, \theta_B, P, K) = d(P, D_B(\theta_B, E_A(\theta_A, P, K), K))$$

where $\theta_A, \theta_B, \theta_E$ represent the parameters of Alice, Bob and Eve respectively. D_B represents the decryption process of Bob and E_A represents the encryption process of Alice. Lastly, P represents the plaintext and K the secret key which means that the loss for Bob is the distance between the original plaintext and his tentative decrypted ciphertext P_{Bob}.

2) COMMUNICATION PHASE

After training is done, the parameters/weights that define the state of the neural networks can be used for current and future secure communications.

a: COMMENTS

As stated before, this model is very interesting but one might ask "how does it differ from classical cryptography protocols?" or "what advantages does it provide?". The answer is that there is no need to build a specific algorithm with detailed steps which is a big difference and an advantage at the same time. The neural networks will work on learning a method on their own without being taught or shown any specific encryption method such as AES. The only drawback is that it takes a considerable amount of time to synchronize two parties for the first time as they do not have pre-saved parameters.

Purswani *et al.* [34] propose the same model but use chaos theory to generate a more random key. Their results show that the accuracy of the model can increase up to 21% when replacing the python built-in random function with other techniques such as the logistic chaotic key or the Henon key.

G. STEGANOGRAPHY

Besides encryption, different researchers in two recent papers [19], [20] pushed the idea of the model proposed by Abadi and Andersen [12] in order to build a steganography model based on neural networks.

In their models and similarly to the work by Abadi and Andersen [12], Alice will use an image and a secret text as input to her neural network. Alice's output will be the steganographic image that Bob is going to try to extract the secret texts from. A different image/secret-text combination is used at every training iteration in order to prevent Alice and Bob from learning a model specific to one particular image or secret text.

H. ADVERSARIAL CRYPTOGRAPHY BASED ON THE TOPOLOGY EVOLVING NEURAL NETWORKS

The authors of this work [35] wanted to build a model based on a new topology called Spectrum-Diverse Neuroevolution with Unified Neural Models [22] which is basically a type of neural network structure that can evolve by adding or removing neurons to/from its structure. So concretely, in [35] they do not use a fixed neural network structure but a structure that can evolve on the go. however the training process and the concept is the same as the original adversarial cryptography model proposed by Abadi and Andersen [12]. The results from [35] show that it is possible to implement such neural networks and they can evolve and learn a symmetric encryption protocol.

I. GAN-BASED KEY SECRET-SHARING SCHEME IN BLOCKCHAIN

The authors of this work [36] implement a secure key sharing scheme based on GANs. The idea consists of transforming the text of a private key into an image which will be the original image for the GAN. The original image is then divided into several sub-images and each of them is encoded using DNA coding. Finally, the proposed scheme is trained to extract the secret key using the encoded sub-images. This scheme helps lower the hardness of recovering a lost private key in block chain.

J. MULTI PARTY ADVERSARIAL CRYPTOGRAPHY

Talking among multiple parties using Adversarial Cryptography can be a useful feature to be implemented however training multiple parties on learning the scheme might be challenging and time consuming. The authors in [21] implemented a 3-party scheme that showed how to train three parties so that they learn the same encryption and decryption scheme in different scenarios, and also gave a workaround for communicating in larger groups.

K. GENERATIVE ADVERSARIAL PRIVACY

Training neural network models requires having on hand a lot of data. This data is generally is difficult to acquire due to privacy problems. A solution that is often used is to anonymize the data by removing any identifying details like names, unique identifying numbers, etc. However recent attacks such as in [37], [38] show that it is possible to deanonymize the data and link it to its original holders.

This is where the work Chong *et al.* [39] comes into play, through what they called Generative Adversarial Privacy (GAP) the authors built a model that can protect the data and anonymize it properly while preserving its utility.

The model is composed of two learning blocks: A privatizer that learns to process the public data in order to output a sanitized version of it and an adversary that tries to learn private data from the public data. This is done through competing in a constrained minimax zero-sum game. The privatizer trains on minimizing the adversary's performance and the adversary tries to find the best strategy to maximize its performance. A loss function is used to measure the efficiency of the adversary.

L. AN APPROACH TO CRYPTOGRAPHY BASED ON CONTINUOUS-VARIABLE QUANTUM NEURAL NETWORK

While Abadi and Andersen [12] used classic neural networks for their setup and training, Shi *et al.* [40] did a similar work but using another approach based on Quantum Neural Networks. The neural networks learn to encrypt plaintexts in an adversarial setup. The training starts with creating a classical neural network that can theoretically do the specified task (Encryption, Classification, etc). The model is optimized with the Adam algorithm [41] and the authors perform their experiments using the Strawberry Fields32 tool. There are two neural networks with the same structure, and the authors adopted a 3-layer (Input Layer, Hidden Layer, Output layer) structure.

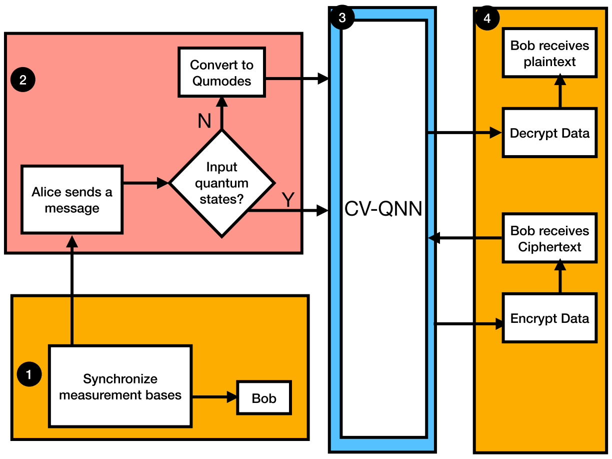


FIGURE 7. 4 Stages of communication [40].

The communication is between Alice and Bob and consists of four stages as illustrated in figure 7:

- The first stage is to obtain Legitimate Measurement bases for Alice and Bob.
- The second stage preprocesses and transforms the data into quomodes.
- The third stages handles the key preparations.
- The last stages is the communication stage where data is encrypted and decrypted.

IV. SECURITY ANALYSIS OF GANS-BASED CRYTOGRAPHY SCHEMES

In this section, we will see a security analysis conducted by Zhou *et al.* [17] to see how the secure are the ciphertexts generated by the neural networks in the model proposed by Abadi and Andersen [12]. We will then see how the model has been improved by Coutinho *et al.* [18] and Li *et al.* [23].

A. SECURITY ANALYSIS AND NEW MODELS ON THE INTELLIGENT SYMMETRIC KEY ENCRYPTION

Zhou *et al.* [17] proposed a security analysis as well as a follow up work on the new way to do encryption based on GANs as proposed by Abadi and Andersen [12]. The authors start by investigating the security of the ciphertexts generated by Alice by testing the randomness of the output to see if it can be distinguished from a randomly picked one. Next, they perform different experiments that will push Alice to generate more complicated and therefore more secure ciphertexts.

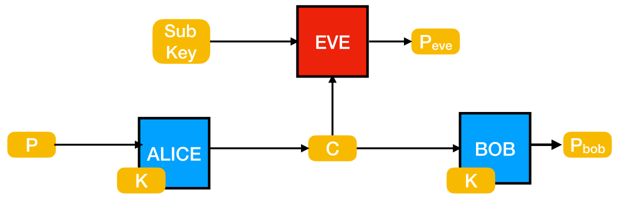


FIGURE 8. Setup of model 1: Eve has a part of the key [17].

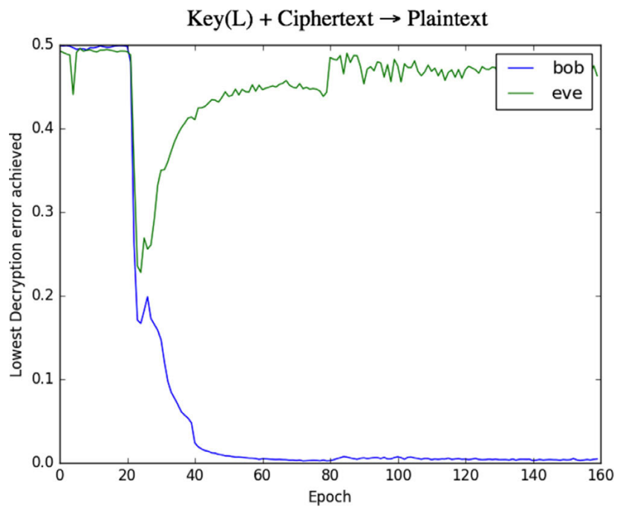


FIGURE 9. Results of experiment 1:Eve gets 4 bits of the key and Bob converges to 100% accuracy [17].

1) TEST OF THE RANDOMNESS OF THE CIPHERTEXTS

In this phase, the authors want to evaluate the randomness of the ciphertexts generated by Alice. To do this, the authors trained Alice in a normal setup as the one proposed by Abadi and Andersen [12]. After the training is done, Alice is fed a large number of keys and plaintexts that she will encrypt and output. The output is saved in a local file for analysis.

The authors used three different methods for analysing the randomness of the output: the χ^2 approach, the Kolmogorov–Smirnov (KS) approach and finally the NIST statistical test. All the results show that the majority of ciphertexts are not secure which makes this model insecure in terms of distinguishability.

2) MODEL IMPROVEMENTS

To get more secure ciphertexts, [17] proposed other setups for the model proposed by Abadi and Andersen [12]. Instead of giving Eve the ciphertext only, they train Alice in different scenarios where in each scenario has a part of the key and/or a part of the original plaintext.

When Eve gets more information, the authors noticed that the behavior of Alice changes as Eve starts getting more accuracy. Figure 8 shows the setup of the model where Eve gets a part of the key.

We can see in figure 9 and 10 that when Eve has only 4 bits of the key, Alice and Bob can beat her and the synchronization is successful with 100% accuracy for Bob and a little less than 50% accuracy for Eve. However with 8 bits of key,

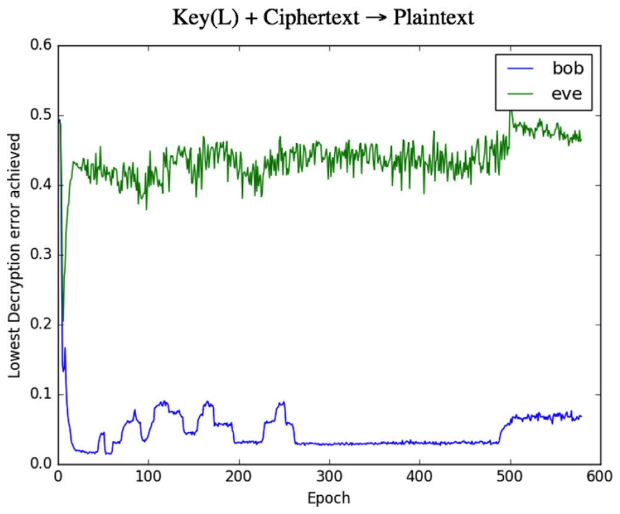


FIGURE 10. Results of experiment 2: Eve gets 8 bits key and Bob cannot converge [17].

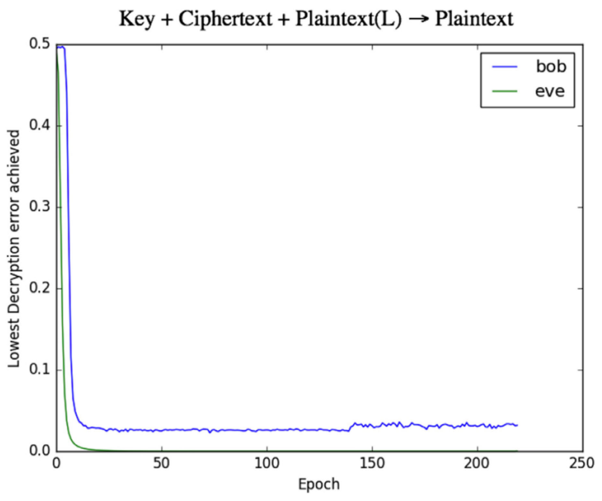


FIGURE 11. Results of experiment 3: Eve gets 16 bits plaintext and the key [17].

Eve is much more accurate in her decryption which pushes Alice to make the encryption method so complicated that Bob even cannot converge and get a perfect decryption of the ciphertexts sent by Alice. Eve's accuracy is also a little higher than the goal of 0.5 after some iterations.

The authors wanted to push Alice and Bob further and made another experiment where Eve gets rich information about the key and the plaintext. In such a case and as shown in Figure 11, Alice and Bob will give up the security and only focus on ensuring the communication. We can see that Bob has a 100% accuracy while Eve has a little less than that.

3) OVERALL RESULTS

The overall experiments and results in this work show that the security can be improved by training against stronger adversaries by giving them for example a part of the key and/or a part of the plaintext. However the randomness of

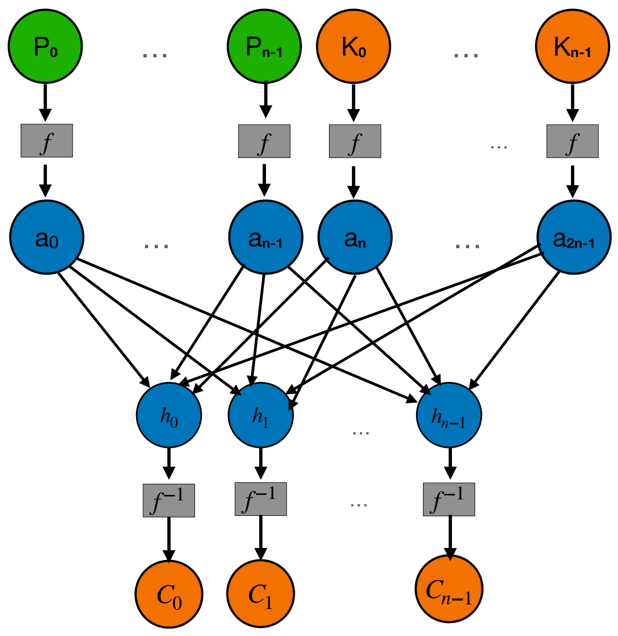


FIGURE 12. Structure used for Alice and Bob in order to Generate a onetime pad [18].

the output needs to be improved in a way that does not leak information on the plaintext or key.

B. LEARNING PERFECTLY SECURE CRYPTOGRAPHY TO PROTECT COMMUNICATIONS WITH ADVERSARIAL NEURAL CRYPTOGRAPHY

Coutinho *et al.* [18] propose a solution to solve the problem of randomness of the ciphertexts generated by Alice. The authors change the structure and the function of the model in order for the neural networks to learn the one time pad with the use of a stronger Eve. The neural networks will do this by learning the One Time Pad (OTP). Concretely, learning the XOR operation and encrypting each given plaintext with a unique key.

1) STRUCTURE FOR ALICE AND BOB

In order to learn the one time pad, the neural network structure of Alice and Bob has to be changed. The new structure is shown in Figure 12 and was used by Coutinho *et al.* [18].

As we can see in Figure 12, the neural network takes as input the plaintext (represented by $p_0 \dots p_{n-1}$) and the secret key (Represented by $k_0 \dots k_{n-1}$) where n is the size of the plaintext and the key k. the function f transforms the input into an angle and is represented as follow:

$$f(b) = arccos(1 - 2b)$$

A fully connected layer combines the angles to form the variables $h_0 \dots h_{n-1}$. Finally, the function f^{-1} reverts back the angles to continuous bits (real numbers in the interval [0, 1]). f^{-1} is defined as follow:

$$f^{-1}(a) = \frac{1 - cos(a))}{2}$$

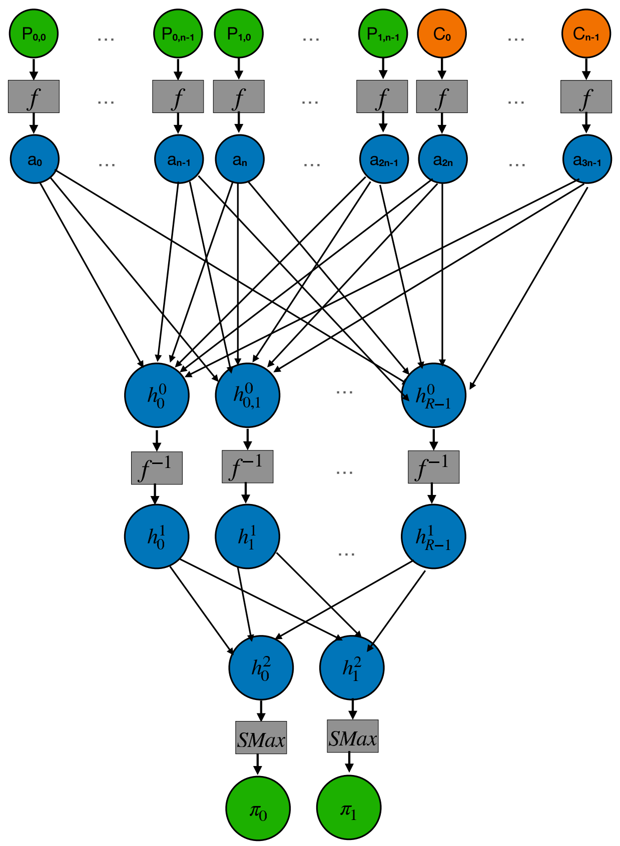


FIGURE 13. Structure used for Eve [18].

TABLE 3. Number of successful communications and the number of OTPs learned [18].

Size of Key	Number of Trial	Successful Communication	OTPs Learned
4-bit	20	19	19
8-bit	20	20	20
16-bit	20	20	19

TABLE 4. Number of successful trials when using three Eves [23].

Order	Number of Trial	Successful Communication	OTPs learned
1	2048	100%	98.9%
2	2048	100%	99.4%
3	2048	100%	100%
4	2048	100%	99.8%
5	2048	100%	99.1%

The output will be $c_0 \dots c_{n-1}$ and will represent the cipher text.

By processing the inputs this way, the authors show that the neural networks were able to communicate but were not able to learn the one time pad. A new structure for Eve was needed in order to get a fully working one time pad for Alice and Bob. The new structure used for Eve is shown in Fig. 13

Eve receives as input two plaintexts P_0 and P_1 as well as a cipher text C_1. The same equation f used by Alice will be used by Eve to transform the input into angles which is

TABLE 5. Table that summarizes the contributions in security analysis done on the model by Abadi and Andersen [12].

Date	Title	Contribution	Positive Points	Negative Points
2016	Learning to Protect Communications With Adversarial Neural Cryptography [12].	Proposed a research spurring model that allows two neural networks to learn an encryption model in the presence of an eavesdropper.	Opened the path to a new way to use GANs which might prove useful against quantum computers as the model learned does not depend on a difficult mathematical problem.	The proposed model learns a weak encryption model. The encryption model is weak against probabilistic attacks.
2018	Security Analysis and New Models on the Intelligent Symmetric Key Encryption [17].	Prove that the model proposed in [12] is weak against probabilistic attacks. Trained the neural networks against stronger versions of Eve which shows that the security can be improved by doing so.	Trains the neural networks against stronger versions of Eve which highly reduces the repetitive patterns in the ciphertexts generated. The model learned can do well when the attacker does not know the structure of the neural networks or information about the plaintext and/or key.	The authors do not test how effective the ciphertext is after Alice and Bob train against stronger versions of Eve.
2018	Learning Perfectly Secure Cryptography to Protect Communications with Adversarial Neural Cryptography [18]	Also shows that the mode learned in [12] is weak against probabilistic attacks. The authors then propose another model that learns the One Time Pad which improves the security and the weakness against probabilistic attacks.	The neural networks learned a perfectly secure model: The One Time Pad. The neural networks encrypt each plaintext using XOR with a unique key per plaintext.	As classical computers cannot generate a fully random pattern, the model will not prove useful unless used with a quantum computer.
2020	Information Encryption Communication System Based on the Adversarial Networks Foundation [23].	Trains alice and Bob against multiple Eves where each Eve has access to different information. Shows that Alice and Bob can learn the OTP without altering their structure.	The neural networks learned the OTP without the need for major changes their structure. Alice and Bob produce more complex ciphertexts.	As Alice and Bob train against three Eves, the model needs a lot of resources and time to be trained.

followed by a fully connected layer to obtain the hidden layer containing the values h_0 through h^1_{R-1}. Then, the function f^{-1} used by Alice previously will also be used by Eve in order to revert the angles into real numbers. The real numbers will go through a Softmax layer outputting two probabilities: π_0 is the probability that P_1 is a plaintext for the cipher text and π_0 is the probability that P_1 is a plaintext for the cipher text.

By using this method, the authors were able to get up to 100% successful communications during their trials and between 95% and 100% learned OTPs. Table 3 illustrates the number of trials and successful communications.

C. INFORMATION ENCRYPTION COMMUNICATION SYSTEM BASED ON THE ADVERSARIAL NETWORKS FOUNDATION

Zhengze *et al.* [23] proposed training the model in another way. Instead of using just one Eve, they used three Eves. The three Eves differ from each other in the following manner:

- The first Eve has access not only to the ciphertext but also the full key. The authors assumed the attacker got the key in advance.
- The second Eve is the same as in [12] and has only access to the ciphertext.
- The third Eve is a little similar to the one used by Coutinho [18]. She receives a plaintext and two ciphertexts one being the real ciphertext and the second randomly generated. She has to decide which ciphertext corresponds to the plaintext.

The rest of the model is the same as by Abadi and Andersen [12] but the data goes through 6 convolutions instead of just 4.

The authors conducted trials with two Eves and three Eves separately and the results are pretty impressive. In their trials with two Eves (Eve 1 and Eve 2), the maximum number of successful trials was around 77%. But with the 3 Eves, they got between 97 and 100% successful communications and OTPs learned.

Table 4 illustrates the number of trials and the successful rate of communications. The results reconfirms our intuition that by training against stronger opponents, Alice and Bob can perform better in terms of encryption in order to outperform the attackers.

D. SUMMARY

We presented a variety of cryptography techniques that are based on neural networks; Most of them are based on deep neural networks, GANs and the work done by Abadi and Andersen [12].

The most important and prominent techniques were:

- The GANs based encryption model [12] where two neural networks can synchronize and learn to encrypt a communication in the presence of eavesdroppers.
- The GANs based steganography models [19], [20], [42] which are based on [12] and enable neural networks to learn steganography by hiding a text inside an image and send it through an open network subject to eavesdropping.
- The introduction of the OTP by Coutinho *et al.* [18] and Zhengze *et al.* [23] that solves the problem of randomness and improving the model by Abadi and Andersen [12].
- The Tree Parity Machine and the various improvements it has seen especially by Salguero *et al.* [24].

E. USAGE CASES OF NEURAL NETWORKS BASED CRYPTOGRAPHY

As Neural Network models are usually lightweight, fast and efficient, the first possible usage case is to apply the learned protocols between IoT devices. As IoT devices work on time-limited batteries, using neural networks based models for encryption is expected to provide substantial security with reduced CPU usage of the devices.
On the other hand, as the security is not based on any computational hardness assumption, neural networks based cryptographic models (with suitable enhancements and modifications) may be expected to provide a lightweight post-quantum secure primitive.

Table 5 summarizes the most important works in the security of GANs based encryption.

V. DISCUSSION AND TREND

We presented the advances of AI-based cryptography during the last two decades especially with GANs. We also presented the importance of GANs in developing cryptosystems and observed that GANs-based neural networks can learn symmetric encryption as well as privacy preserving. We have also discussed several attack models which scrutinize the security of the developed systems.

Neural network based cryptography has drawn significant attention with the hope that it could provide post-quantum cryptographic primitives. However, the research in this field is still in a nascent stage and several developments in different fronts are observed on a regular basis. One important direction that researchers are trying to explore include public key cryptography and secret sharing – both of these act as fundamental building blocks for several cryptographic tasks. Moreover, for realizing real world secure NN based distributed systems friendliness with low-capacity devices is also a current trend of research.

Another interesting research area focuses on quantum-enhancing the neural networks which requires writing a quantum circuit of the neural network and running it on a quantum machine simulator or on an online available quantum machine [43], [44].

ACKNOWLEDGMENT

The authors would like to thank Dr. Yujie Gu for her comments to improve the presentation of the paper. This project was originally conceived during that period and a preliminary draft was discussed with the first author.

REFERENCES

[1] R. L. Rivest, A. Shamir, and L. Adleman, ''A method for obtaining digital signatures and public-key cryptosystems,'' *Commun. ACM*, vol. 21, no. 2, pp. 120–126, Feb. 1978, doi: 10.1145/359340.359342.

[2] T. Jamil, ''The Rijndael algorithm,'' *IEEE Potentials*, vol. 23, no. 2, pp. 36–38, Apr. 2004.

[3] S. Prajapat and R. Singh, ''Various approaches towards cryptanalysis,'' *Int. J. Comput. Appl.*, vol. 127, no. 14, pp. 15–24, Oct. 2015.

[4] R. Spillman, M. Janssen, B. Nelson, and M. Kepner, ''Use of a genetic algorithm in the cryptanalysis of simple substitution ciphers,'' *Cryptologia*, vol. 17, no. 1, pp. 31–44, Jan. 1993.

[5] N. D. Truong, J. Y. Haw, S. M. Assad, P. K. Lam, and O. Kavehei, ''Machine learning cryptanalysis of a quantum random number generator,'' *IEEE Trans. Inf. Forensics Security*, vol. 14, no. 2, pp. 403–414, Feb. 2019.

[6] J. So, ''Deep learning-based cryptanalysis of lightweight block ciphers,'' *Secur. Commun. Netw.*, vol. 2020, Jul. 2020, Art. no. 3701067.

[7] F. Wang, R. Ni, J. Wang, Z. Zhu, X. Chen, and Y. Hu, ''Novel fully convolutional network for cryptanalysis of cryptosystem by equal modulus decomposition,'' *Laser Phys. Lett.*, vol. 17, no. 9, Jul. 2020, Art. no. 095201, doi: 10.1088/1612-202x/aba1f1.

[8] T.-W. Yue and S. Chiang, ''A neural network approach for visual cryptography,'' in *Proc. Int. Joint Conf. Neural Netw.*, vol. 5, Jul. 2000, pp. 494–499.

[9] I. Kanter, W. Kinzel, and E. Kanter, ''Secure exchange of information by synchronization of neural networks,'' *Europhys. Lett.*, vol. 57, no. 1, pp. 141–147, Jan. 2002.

[10] W. Yu and J. Cao, ''Cryptography based on delayed chaotic neural networks,'' *Phys. Lett. A*, vol. 356, nos. 4–5, pp. 333–338, Aug. 2006.

[11] A. Klimov, A. Mityagin, and A. Shamir, ''Analysis of neural cryptography,'' in *Advances in Cryptology* (Lecture Notes in Computer Science), vol. 2501, Y. Zheng, Ed. Queenstown, New Zealand: Springer, Dec. 2002, pp. 288–298, doi: 10.1007/3-540-36178-2_18.

[12] M. Abadi and D. G. Andersen, ''Learning to protect communications with adversarial neural cryptography,'' *CoRR*, vol. abs/1610.06918, pp. 1–15, Oct. 2016.

[13] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, ''Generative adversarial nets,'' in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 27, 2014.

[14] Y. Gao, R. Singh, and B. Raj, ''Voice impersonation using generative adversarial networks,'' 2018, *arXiv:1802.06840*. [Online]. Available: http://arxiv.org/abs/1802.06840

[15] Y. Kataoka, T. Matsubara, and K. Uehara, ''Image generation using generative adversarial networks and attention mechanism,'' in *Proc. IEEE/ACIS 15th Int. Conf. Comput. Inf. Sci. (ICIS)*, Jun. 2016, pp. 1–6.

[16] A. Siarohin, S. Lathuiliere, E. Sangineto, and N. Sebe, ''Appearance and pose-conditioned human image generation using deformable GANs,'' *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 4, pp. 1156–1171, Apr. 2021.

[17] L. Zhou, J. Chen, Y. Zhang, C. Su, and M. A. James, ''Security analysis and new models on the intelligent symmetric key encryption,'' *Comput. Secur.*, vol. 80, pp. 14–24, Jan. 2019.

[18] M. Coutinho, R. de Oliveira Albuquerque, F. Borges, L. G. Villalba, and T.-H. Kim, ''Learning perfectly secure cryptography to protect communications with adversarial neural cryptography,'' *Sensors*, vol. 18, no. 5, p. 1306, Apr. 2018.

[19] M. Yedroudj, F. Comby, and M. Chaumont, ''Steganography using a 3-player game,'' *J. Vis. Commun. Image Represent.*, vol. 72, Oct. 2020, Art. no. 102910.

[20] J. Hayes and G. Danezis, ''Generating steganographic images via adversarial training,'' in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 1954–1963.

[21] I. Meraouche, S. Dutta, and K. Sakurai, ''3-Party adversarial cryptography,'' in *Advances in Internet, Data and Web Technologies*, L. Barolli, Y. Okada, and F. Amato, Eds. Cham, Switzerland: Springer, 2020, pp. 247–258.

[22] D. V. Vargas and J. Murata, "Spectrum-diverse neuroevolution with unified neural models," 2019, *arXiv:1902.06703*. [Online]. Available: http://arxiv.org/abs/1902.06703
[23] Z. Li, X. Yang, K. Shen, R. Zhu, and J. Jiang, "Information encryption communication system based on the adversarial networks foundation," *Neurocomputing*, vol. 415, pp. 347–357, Nov. 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231220313175
[24] É. S. Dorokhin, W. Fuertes, and E. Lascano, "On the development of an optimal structure of tree parity machine for the establishment of a cryptographic key," *Secur. Commun. Netw.*, vol. 2019, pp. 1–10, Mar. 2019.
[25] T. Dash, S. N. Dambekodi, P. N. Reddy, and A. Abraham, "Adversarial neural networks for playing hide-and-search board game Scotland yard," *Neural Comput. Appl.*, vol. 32, no. 8, pp. 3149–3164, Apr. 2020.
[26] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE Trans. Inf. Theory*, vol. 22, no. 6, pp. 644–654, 1976.
[27] A. Ruttor, W. Kinzel, and I. Kanter, "Dynamics of neural cryptography," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 75, no. 5, May 2007, Art. no. 056104.
[28] O. M. Reyes and K.-H. Zimmermann, "Permutation parity machines for neural cryptography," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 81, no. 6, Jun. 2010, Art. no. 066117.
[29] L. F. Seoane and A. Ruttor, "Successful attack on permutation-parity-machine-based neural cryptography," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 85, no. 2, Feb. 2012, Art. no. 025101.
[30] J. Jin and K. Kim, "3D CUBE algorithm for the key generation method: Applying deep neural network learning-based," *IEEE Access*, vol. 8, pp. 33689–33702, 2020.
[31] N. Prabakaran and P. Vivekanandan, "A new security on neural cryptography with queries," *Int. J. Adv. Netw. Appl.*, vol. 2, pp. 437–444, 2008.
[32] S. Kiyomoto, H. Ota, and T. Tanaka, "On-the-fly automatic generation of security protocols," in *Proc. ICEIS*, vol. 2, Jan. 2008, pp. 97–104.
[33] H. Ota, S. Kiyomoto, and Y. Miyake, "Automatic security verification for 3-party authentication and key exchange protocols," in *Proc. 5th Int. Conf. Netw. Syst. Secur.*, Sep. 2011, pp. 254–258.
[34] J. Purswani, R. Rajagopal, R. Khandelwal, and A. Singh, "Chaos theoryon generative adversarial networks for encryption and decryption of data," in *Proc. Adv. Bioinf., Multimedia, Electron. Circuits Signals*. Springer, 2020, pp. 251–260.
[35] Y. Zhu, D. V. Vargas, and K. Sakurai, "Neural cryptography based on the topology evolving neural networks," in *Proc. 6th Int. Symp. Comput. Netw. Workshops (CANDARW)*, Nov. 2018, pp. 472–478.
[36] W. Zheng, K. Wang, and F. Wang, "GAN-based key secret-sharing scheme in blockchain," *IEEE Trans. Cybern.*, vol. 51, no. 1, pp. 393–404, Jan. 2020.
[37] A. Narayanan and V. Shmatikov, "Robust de-anonymization of large sparse datasets," in *Proc. IEEE Symp. Secur. Privacy*, May 2008, pp. 111–125.
[38] E. S. Finn, X. Shen, D. Scheinost, M. D. Rosenberg, J. Huang, M. M. Chun, X. Papademetris, and R. T. Constable, "Functional connectome fingerprinting: Identifying individuals using patterns of brain connectivity," *Nature Neurosci.*, vol. 18, no. 11, pp. 1664–1671, Nov. 2015.
[39] C. Huang, P. Kairouz, X. Chen, L. Sankar, and R. Rajagopal, "Context-aware generative adversarial privacy," 2017, *arXiv:1710.09549*. [Online]. Available: http://arxiv.org/abs/1710.09549
[40] J. Shi, S. Chen, Y. Lu, Y. Feng, R. Shi, Y. Yang, and J. Li, "An approach to cryptography based on continuous-variable quantum neural network," *Sci. Rep.*, vol. 10, no. 1, Dec. 2020, Art. no. 2107, doi: 10.1038/s41598-020-58928-1.
[41] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*. [Online]. Available: http://arxiv.org/abs/1412.6980
[42] Y. Ke, M. Zhang, J. Liu, and T. Su, "Generative steganography with Kerckhoffs' principle based on generative adversarial networks," *CoRR*, vol. abs/1711.04916, pp. 1–8, Nov. 2017.
[43] V. Dunjko, J. M. Taylor, and H. J. Briegel, "Quantum-enhanced machine learning," *Phys. Rev. Lett.*, vol. 117, no. 13, Sep. 2016, Art. no. 130501, doi: 10.1103/PhysRevLett.117.130501.
[44] V. Havlíček, A. D. Córcoles, K. Temme, A. W. Harrow, A. Kandala, J. M. Chow, and J. M. Gambetta, "Supervised learning with quantum-enhanced feature spaces," *Nature*, vol. 567, no. 7747, pp. 209–212, 2019, doi: 10.1038/s41586-019-0980-2.

ISHAK MERAOUCHE (Member, IEEE) received the B.S. degree in computer science and the M.S. degree in cryptography and security from the University of Batna 2, Batna, Algeria, in 2016 and 2018, respectively. He is currently pursuing the Ph.D. degree in informatics with Kyushu University, Fukuoka, Japan.

From 2019 to 2020, he was a Research Student with Sakurai Laboratory, Kyushu University. His research interest includes building cryptography protocol especially the ones based on deep learning.

Mr. Meraouche received a Scholarship after his master's degree to pursue his Ph.D. studies in Japan.

SABYASACHI DUTTA received the B.Sc., M.Sc., and Ph.D. degrees in mathematics from the University of Calcutta, Kolkata, India.

From 2017 to 2018, he was a Visiting Scientist with the Indian Statistical Institute, Kolkata. In 2018, he received a Prestigious Fellowship from the National Institute of Information and Communications Technology (NICT), Japan, and subsequently spent one year, from 2018 to 2019, at Kyushu University, Fukuoka, Japan, as a Guest Researcher. He is currently a Postdoctoral Fellow with the University of Calgary, Canada. His research interests include information theoretic cryptography and in building quantum-safe cryptographic protocols.

HAOWEN TAN (Member, IEEE) received the B.S. and M.S. degrees in computer science from Nanjing University of Information Science and Technology, Nanjing, China, in 2013 and 2016, respectively, and the Ph.D. degree in computer science from Chosun University, Gwangju, South Korea, in 2020.

From 2020 to 2021, he was a Research Fellow with the IT Research Institute, Chosun University. He is currently a Postdoctoral Fellow with the Cyber Security Center, Kyushu University, Fukuoka, Japan. His current research interests include network security, ubiquitous sensor networks, blockchain, and vehicular ad hoc networks.

KOUICHI SAKURAI (Member, IEEE) received the B.S. degree in mathematics from the Faculty of Science, Kyushu University, in 1986, and the M.S. degree in applied science and the Ph.D. degree in engineering from the Faculty of Engineering, Kyushu University, in 1988 and 1993, respectively.

He was engaged in research and development on cryptography and information security at the Computer and Information Systems Laboratory, Mitsubishi Electric Corporation, from 1988 to 1994. Since 1994, he has been working as an Associate Professor with the Department of Computer Science, Kyushu University, where he became a Full Professor, in 2002. He is currently working with the Institute of Systems and Information Technologies and Nanotechnologies, as the Chief of the Information Security Laboratory, for promoting research co-operations among the industry, university and government under the theme "Enhancing IT-Security in Social Systems." He has been successful in generating such co-operation between Japan, China, and South Korea for security technologies as the Leader of a Cooperative International Research Project supported by the National Institute of Information and Communications Technology (NICT) during 2005–2006. Moreover, in March 2006, he established research co-operations under a Memorandum of Understanding in the field of information security with Prof. Bimal Kumar Roy, the first time Japan has partnered with The Cryptology Research Society of India (CRSI). He has published more than 400 academic articles around cryptography and information security.