Tel-Hai College

[Neural Network Cryptologic Key Exchange]

Present: Or David Zinger

Department of Computer Science

Instructors: Prof. Dani Kotler

Dr. Ofer M. Shir

Fall semester, 2016-2017

Thanks

Many thanks to my both instructors for help and guidance along my project, for all the consultations and the intent during writing and programing my project.

To Prof. Dani Kotler, who has opened the door for me to the cryptology world since I took a course with him, for all the consultations on the analysis of the theoretical part - especially in the field of cryptologic research, for understanding the essence of the research and applying the algorithm in all its cryptologic aspects.

To Dr. Ofer M. Shir, who gave me all the tools and guidance to implement the algorithm with comments and professional instructions, which exposed me to the "infinite" libraries and functionality of C ++ programming, and taught me all aspects of neural networks and how it could be a groundbreaking domain through the world of computer science.

Table of contents

Abstract

The idea behind "mixing" the world of cryptology and the world of artificial neural networks was due to the great interest shown by researchers in the fields of engineering and computer science in the power of neural network.

It's common to think that ANN (Artificial Neural Networks) are benefits of computer vision and part of self-learning and recognition of discrete objects, but it this case, we take the power, modularity and the simplicity of ANN to the fields of cryptologic to try build encryption and decryption system, to solving complex problems in cryptography theory, and to give it a new approach.

Google and Microsoft, two of the world's largest companies, showed interest in this new "baby born" field (*we call it "Neural Cryptography"*) and even began to develop their own neural crypto systems and explore the field of it. They set up professional teams to explore the field of *Neural Cryptography* and made breakthrough systems to get strong security in the world of cyber.

The proposed project defines a symmetric key exchange protocol across a public channel, based on Diffie-Hellman exchange key protocol, using the neural networks defined by each of the participants- the sender and the receiver. We will discuss the effectiveness of the algorithm both in terms of protection and offensive, while a third party eavesdropper to them.

Cryptography theory

1. Abstract

   The field of learning algorithms of encrypting and decrypting data between some objects, is called “cryptography theory”, or just “cryptography”.

   The theory deals with research and development of secure systems while maintaining maximum security through formal proofs, preventing third parties from eavesdropping and receiving information that needs to be secured and not public.

   In cryptography, other study areas are involved, like the information theory, group theory, modern algebraic, quantum theory, number theory etc.

   In 19 century, *Auguste Kerckhoffs* formulated a principle in cryptography, known as *Kerckhoffs* principle- Any encryption system should be secure even all his components are known, except the private key. Based on this principle, *Claude Shannon* defined his own principle, mostly the most acceptable principle in cryptography world- perfect secret, which describe an encryption method that is perfectly secured if a ciphertext provides no information about the plaintext without knowledge of the key. It’s important that the code will be as confusing as possible, that no one can collect information from it.

2. Terminology

   2.1- Basic terms

   Alice- Sender.

   Bob- Recipient.

   Eve- Eavesdropper.

Plaintext- The original data to be encrypted.

Ciphertext- The encrypted data the Bob recipient.

Key- The key to encrypt/ decipher the data, can be symmetric or asymmetric.

Algorithm- The algorithm behind the encryption/decryption system.

Cryptography- Field of study the algorithms and proof their security.

Cryptanalysis- Field of study the insecurity and weakness of algorithms.

Brute force- A force attacking on cryptographic key, using all possible keys.

Symmetric key- Synchronously key used for both encryption and decryption.

Asymmetric key- Asynchronously key used only for encryption or decryption.

Key length- Range from 128 bit to 4096 bit, commonly is 256 bit.

2.2-Crtyptographic scheme

In cryptography, we see that there are always three participants, two of them the sender and the recipient, while the third is the eavesdropper.

The sender is called *Alice*, and she wants to send a private message to *Bob*, the recipient of the original message, while the entire send / receive transition occurs via public communication. Meanwhile, there is "*Eve*," and she wants to eavesdrop on *Alice* and *Bob's* private conversation.

Alice want to send private message to *Bob*, named "*Plaintext*", so she encrypt the plaintext with *secret encryption key*, that only she (or maybe even *Bob- see 4. for asymmetric key*) knows it. She encrypt the plaintext with *encryption algorithm* she choose and acceptable by *Bob* while all this procedure occurring in public (*Kerckhoffs and Shannon principle*) and with the key she choose, and then send the encrypt plaintext, named "*Ciphertext*" to *Bob*. *Bob*, the recipient, receive the cyphertext and

decipher it with his own *secret decryption key*, and get the original plaintext from *Alice* (*again, see 4. for asymmetric key*).

3. Classic Cryptography

   Classic cryptography is a term for describing the original preoccupation of cryptography, when people used pencil and paper to encode their messages and sent it through physical public channels, mostly by humans.

   There are several basic encryption methods used in the previous centuries, two of which were "transposition cipher" and "substitution code". The Vigenere cipher is an example of a substitution code, where almost 400 years after it was generated, the algorithm was decoded. Transposition cipher is a cipher in which one symbol is moved and changed his location according to the encryption method.

   Substitution cipher is a cipher that every symbol replacement by other symbol, when the symbols changed in monoalphabetic order (fixed offset) or in polyalphabetic order (the offset changed due to multiset of keys). For example, Caesar cipher is a monoalphabetic order encryption, and Vigenere cipher is a polyalphabetic order encryption.

4. Modern Cryptography

   Modern cryptography is a term for describing the modern usage of cryptography methods through modern applications and advanced methods, most of them based of the number theory. It relies on publicly known mathematical algorithms for coding and decoding the information. Secrecy is obtained through a secrete key which is

used as the seed for the algorithms. The computational difficulty of algorithms, absence of secret key, etc., make it impossible for an attacker to obtain the original information even if he knows the algorithm used for coding. Information theory and number theory are the most useful theories in this field of cryptography. Some of the most known algorithms taken from the modern cryptography, are unbreakable until this days, due to their power and benefits.

In modern cryptography there are two kinds of private keys:

Symmetric key- A key used for both encrypt and decipher the data, usually classified in to "block ciphers" and "stream ciphers".

Asymmetric key - known as a public key and private key, where the public key is used to encrypt data and doesn't care if anyone knows about it, and the private key is used to decipher the message when it is only on the receiving end.

5. Cryptanalysis

Cryptanalysis is a branch of cryptography field, some say that a different field, that refers to study of ciphers and cryptosystems and their weaknesses to achieve the plaintext that was encoded, without the need to know the private key that used to encrypt or decrypt (depends on symmetric or asymmetric keys). In cryptanalysis, researchers and experts are trying to find out how insecure the encryption system in front of them. This refers to finding a property (fault) in the design or implementation of the cipher that reduces the number of keys required in a brute force attack (that is, simply trying every possible key until the correct one is found). [2] There are numerous techniques for performing cryptanalysis, depending on what access the cryptanalyst has to the plaintext, ciphertext, or other aspects of the cryptosystem.

Below are some of the most common types of attacks:

1) Known-plaintext analysis: With this procedure, the cryptanalyst has knowledge of a portion of the plaintext from the ciphertext. Using this information, the cryptanalyst attempts to deduce the key used to produce the ciphertext.

2) Chosen-plaintext analysis (also known as differential cryptanalysis): The cryptanalyst is able to have any plaintext encrypted with a key and obtain the resulting ciphertext, but the key itself cannot be analyzed. The cryptanalyst attempts to deduce the key by comparing the entire ciphertext with the original plaintext. The Rivest-Shamir-Adleman encryption technique has been shown to be somewhat vulnerable to this type of analysis.

3) Ciphertext-only analysis: The cryptanalyst has no knowledge of the plaintext and must work only from the ciphertext. This requires accurate guesswork as to how a message could be worded. It helps to have some knowledge of the literary style of the ciphertext writer and/or the general subject matter.

ANN- Artificial Neural Networks

1. Abstract

Artificial Neural Networks is a branch of artificial intelligence. This study field, try to mimic the human mind to solve problems in space of solutions, try to optimized hard problems and building a model for future using. It is inspired by the human brain, simulates a simple graph that is connected with units called "neurons" and trains itself by examples that the user give him. Neural networks resembles the human brain in the following two ways: a neural network acquires knowledge through learning and it stored it within the interconnection strengths known as synaptic weight. Typically, neurons are organized in layers. Different layers may perform different kinds of transformations on their inputs. Signals travel from the first (input), to the last (output) layer, possibly after traversing the layers multiple times. Artificial Neural Networks (ANN) is the system of elements interacting by adaptive couplings which are trained from a set of examples. After training they function as content addressable associative memory, as classifiers or as prediction algorithms. It consists of three layers. One or more hidden layers might be used in the structure. Neurons in the input layer can be treated as buffer and distribute *xi* input signal to neurons in hidden layer. Output of each neuron *j* in the hidden layer is obtained from sum of multiplication of all input signals *xkj* and weights *wkj* that follows all these input signals. The sum can be calculated as a function of *yj* and can be expressed as: $Y_k = f(\sum W_{ji} * X_j)$. where f can be a simple threshold function, sigmoid function or any another function suitable to the problem.

2. Terminology

Neuron- Node or unit, represent part of the architecture.

Model- The algorithm describes the network, to training and testing it.

Training- Process of training the model on group of examples, some data.

Testing- Process of testing the model after build it, on group of examples.

Layer- Group of neurons that aren't connected with each other called layer.

Weight vector- The synapses, connected the neurons on one layer to another layer.

Learning rule- A function that update the weights, mostly give "penalty" to weights.

Backpropagation- An algorithm to train the model.

Feed forward neural network- Nodes in i layer are connector to i+1 layer.

Classification- Benefit of ANN, networks that classify examples by their tags.

Supervised- Examples given with their tags to score the results.

Unsupervised- Examples without their tags to score the results.

3. Deep Learning

Deep learning is a branch of artificial intelligence, began somewhere at the 90's as part of researches in computer science and electric engineering fields.

There is conventions that deep learning is a part of machine learning field, that it itself part of artificial intelligence. This branch born from the needs of experts and researchers to simulate the computing power of the machine as the human mind, since humans are always in the process of continuous learning.

Deep learning, known as "artificial neural network", are programs designed to solve any problems by trying to mimic the structure and the function of human nervous system.

As it turned out, one of the very best application areas for machine learning for many years was computer vision, though it still required a great deal of hand-coding to get the job done. People would go in and write hand-coded classifiers like edge detection filters so the program could identify where an object started and stopped; shape detection to determine if it had eight sides; a classifier to recognize the letters "S-T-O-P." From all those hand-coded classifiers they would develop algorithms to make sense of the image and "learn" to determine whether it was a stop sign [2]. Neural Networks are inspired by our understanding of the biology of our brains – all those interconnections between the neurons. But, unlike a biological brain where any neuron can connect to any other neuron within a certain physical distance, these artificial neural networks have discrete layers, connections, and directions of data propagation. Each neuron assigns a weighting to its input — how correct or incorrect it is relative to the task being performed. The final output is then determined by the total of those weightings. So think of our stop sign example. Attributes of a stop sign image are chopped up and "examined" by the neurons — its octagonal shape, its fire-engine red color, its distinctive letters, its traffic-sign size, and its motion or lack thereof. The neural network's task is to conclude whether this is a stop sign or not. It comes up with a "probability vector," really a highly educated guess, based on the weighting. In our example the system might be 86% confident the image is a stop sign, 7% confident it's a speed limit sign, and 5% it's a kite stuck in a tree, and so on — and the network architecture then tells the neural network whether it is right or not. Today, image recognition by machines trained via deep learning in some scenarios is better than humans [3].

Diffie-Hellman key exchange protocol

In the past, when the encryption was used mainly by armies, diplomats and spies, the secret key was transferred in a personal meeting or by a courier who was loyal to the parties, however, in online commerce, which includes countless transactions or actions, this method is impractical. It is necessary to pass the encryption key from the sender to the recipient so that the latter can decode the messages he received. If the encryption key falls into the hands of a third party during the transfer, it immediately affects the security of the protocol: the opponent can use this key to decrypt all network traffic encrypted with it. **The key distribution problem** is a problem that deal with cryptographic aspects of transfer, management, and maintenance of secret encryption keys between two or more entities through public channels. In asymmetric encryption, there is no worry about it because both sides have one public key and one private key that should never need to be transferred through public channels. In symmetric encryption, the problem exists as described earlier. The participants in the call use an open channel, so they are easily eavesdropped. The problem stems from the fact that the participants don't have a secure channel to share the encryption key between them. They may never have met or even know each other. Transferring the key by messenger is also not entirely safe and is not practical if they are far apart. They can pass the encryption key in other ways, such as by mail or phone, but they can also be monitored in simple ways. In addition, if there is a secure way to pass the encryption key, participants may be able to use it to transfer the information itself, and no encryption is required at all. A protocol that solves the key distribution problem is called a key construction protocol. It is an algorithm that includes a series of moves or exchanges of messages between two or more participants in an open communications network, at the

end of which the legitimate participants share a secret key so that they are certain that no one knows what it is. Typically, there are two types of protocols:

Key agreement- Two or more participants agree on a secret key so that each contribute their share to its establishment equally.

Key transfer- One party generates and passed a secret key to the other participants, while they do not necessarily contribute to its establishment and have no control over it.

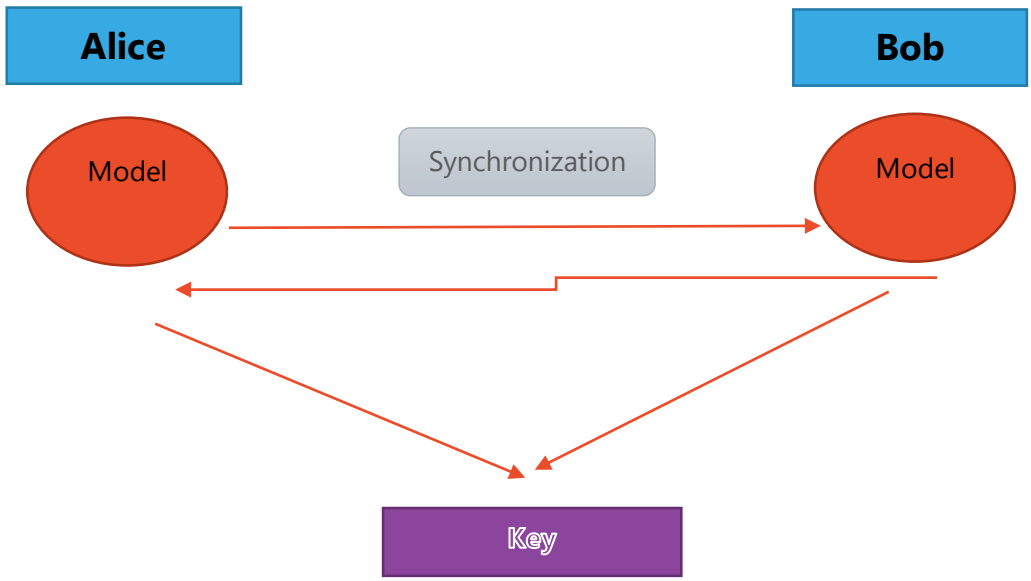
There are two main ways to share a key:

trusted third party -Is a dedicated server whose function is to produce and distribute secret encryption keys for all network users. First, users share a secret encryption key with the KDC server individually, and in turn manages a long list of secret keys for all registered users. It then generates, sends, and sends a temporary one-time call key encrypted with the appropriate encryption keys for users who want to call and use the key when the key is destroyed.

Diffie-Hellman Protocol- The protocol allows two participants who have never met and do not share a common secret in advance, to pass between them on an open channel (which is not secure) any secret so that no one else knows. The Diffie -Hellmann protocol deals with this problem with an asymmetrical method. The protocol exempts them from the need to keep secret cryptographic keys over time; Instead, the encoder can prepare a temporary encryption key, pass it through the protocol to the other side, and then the communication between them can be encrypted. The confidence of the Diffie -Hellmann protocol is based on the hypothesis that it is difficult to solve the Diffie -Hellman problem or the problem of discrete logarithms in certain groups more than in force [4].

The ANN Key Exchange Protocol

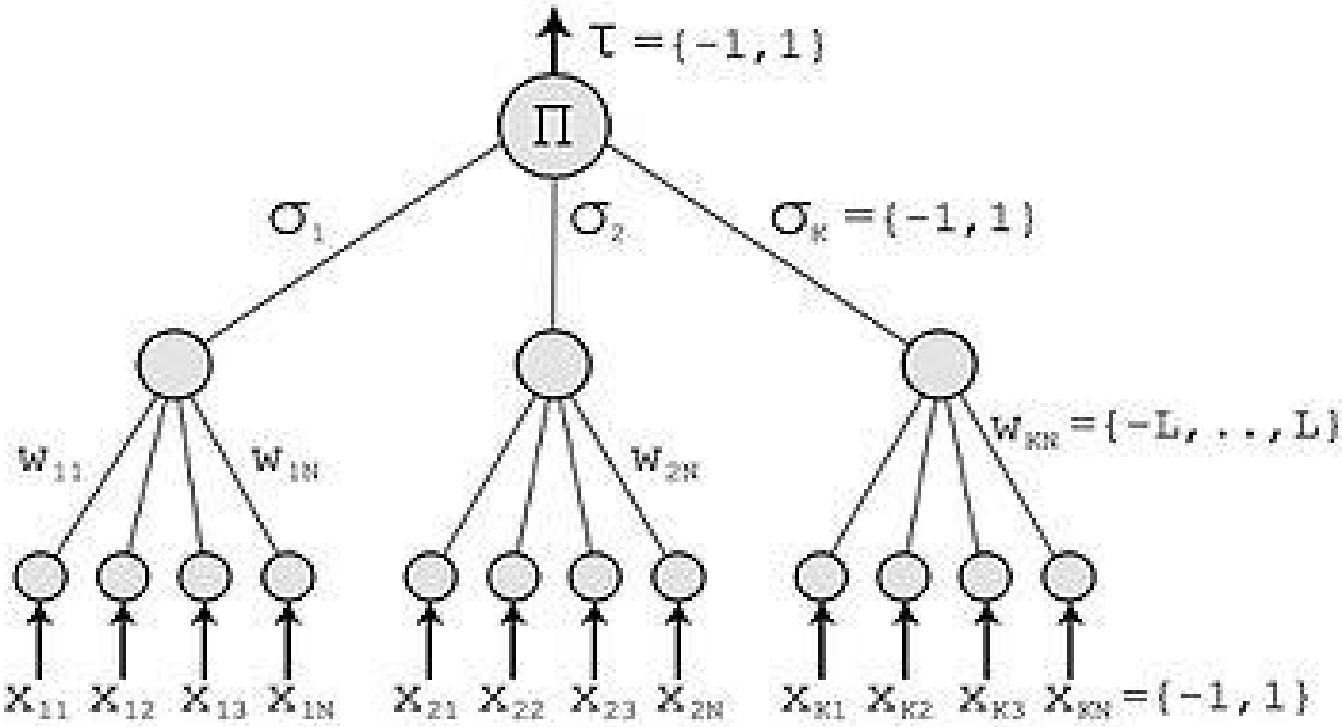
Alice and Bob want to share a secret data. Each of the participants are located in faraway places. They decide on encryption system, probably a system with high security, on public channel. Alice and Bob need to share a private key, without exposed it to any third party. They choose to create a key, and send it via public channel, using Diffie-Hellman exchange key protocol. But instead using the algebraic method, they choose to use neural networks using a model named "Tree Parity Machine" (TPM). To use TPM, each of the participants defined a common TPM in their own locations, when they only share the following parameters for building TPM: K- number of hidden layers, L- limit of the weight vector, and N- number of input layers. After they've shared it, they run the TPM models, and they get a synchronized key, based on their weights vector, on their local computer, which is the same in both of them even though they're in remote locations! Now they have a synchronized key, and they run on it a hash function that will give them the same hash key (hexa-decimal) because they have the same key. Now they can use it as a secret symmetric key to encrypt and decipher data using the encryption system they have agreed upon before.



Tree Parity Machine

The model that the protocol based on is called “tree parity machine”. This model was take from the world of neural networks, and it describe a simple neural network.

The TPM consists of 3 layers, when two of them are known to everyone who eavesdropper, while the third is hidden. It consists of one input layer built from N neurons, selected in such a way that each subset of neurons represents a complete double of the weights vector. It consists of one hidden layer built from K neurons, selected in such a way that each subset of neurons represents a complete double of the weights vector. In total, there are $N * K$ numbers represent the weights vector of the model. It consists of one output layer built from 1 neuron. In total, there are $N * K + 1$ neurons all over the model.



5.2 illustration- Tree Parity Machine with N input, K hidden, 1 output

K and N are known to everyone, and also L, who represent the limit of the numbers on the weights vector. By mutual agreement, the weights range between –L and L.

The input neurons, *N*K* in total, are range between *{-1, 1}* as binary operators, the weights synapses are range between *{ -L, -L+1, …, L-1, L }*, where *[-L, -L+1, … , L-1, L]* are all integers. and the only output neuron range between *{-1,1}*.

It can be described like this: each subset of input neurons is notion by two indexes- *ij* when $1 \leq i \leq N$ and $1 \leq j \leq K$, so $W_{ij} \in [-L, -L+1, \dots, L-1, L]$ (W_{ij} is the weight synapses of each hidden layer *i* and one of the inputs from the input layer as *j*) and so we have even $X_{ij} \in [-1,1]$ input values (for example, hidden neuron which we denote as σ_1 is connected to *N* input neurons so we defined them as- $X_{11}, X_{12}, X_{13}, X_{14}, \dots, X_{1N}$).

Now we run on each hidden neuron and calculate the multiplications of all the synapses connected to it and their input neurons. On each value, we calculate his sign, using the *sign* function, where sign value is *1* for $Y > 0$ and is *-1* for $Y \leq 0$.

$$sign(y) = \begin{cases} 1 & y > 0 \\ -1 & y \leq 0 \end{cases}$$

$$\sigma_i = sign\left(\sum_{j=1}^{j=N} (X_{ij} * W_{ij})\right) (1 \leq i \leq K)$$

The output layer "fired" 1 or -1, due to his calculation: we denote it as τ ,where the output $\tau \in [-1,1]$ and it is multiplication of all the hidden neurons.

$$\tau = \prod_{i=1}^{i=K} \sigma_i$$

Both sides have the identical machines, identical architecture, so they are ready to run their models.

Each participant begin a with random weights vector, range between *[-L, L]*. They begin with common input vector, who's size is *N*K* neurons. This input can be generated from a standard random numeric generator who act due the benefits of the Chaos theory. The models employ their own activation functions, which calculate the product product's synapse with their neurons. The model gets *1* or *-1* for each of the hidden neurons and then multiplies them to each other. Finally, the output neuron hold bit value and fired it to the other machine.

Now each machine get an output result of the other machine, and compare it to their own output. If the outputs are different, nothing occurs, the random generator create new random input (*see [1])*. If the outputs equal, means $O^A = O^B$, each one of the models update their weights vector in the next way: only the hidden neurons who equal with their outputs to the final output are update (e.g., if the final output was 1 ,then only the synapses of hidden neurons whose output was 1 will be updating), by a learning rule, one of the three above, where $\theta(x, y)$ is a function that emits 0 if $a \neq b$ else emits 1 if $a = b$:

Hebbian learning rule - $W_i^+ = W_i + \sigma_i X_i \theta(\sigma_i \tau)\theta(\tau^A \tau^B)$

Anti-Hebbian learning rule - $W_i^+ = W_i - \sigma_i X_i \theta(\sigma_i \tau)\theta(\tau^A \tau^B)$

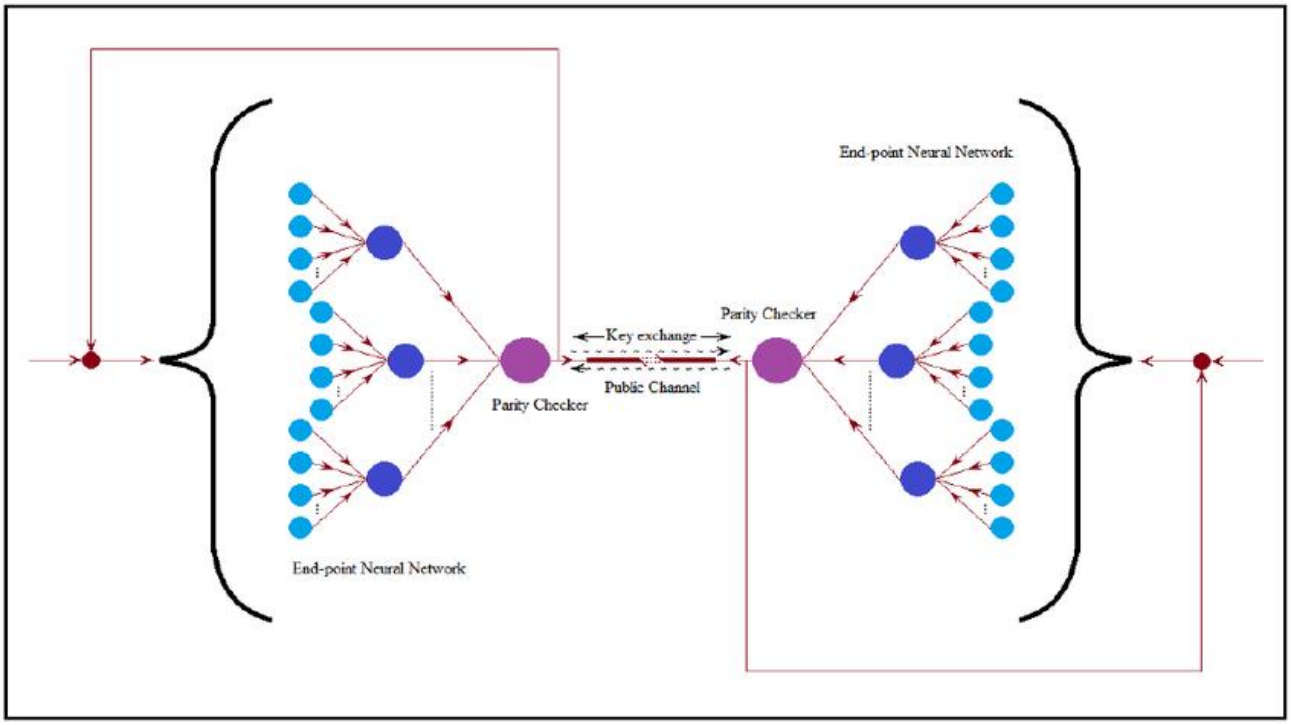
Random-walk learning rule - $W_i^+ = W_i + X_i \theta(\sigma_i \tau)\theta(\tau^A \tau^B)$

If after the synapses had been updated, some weights were out of range, meaning outside the boundary of -L and L, the "problematic" weight is defined to -L or L in accordance.

All the actions described above occur in a linear number of iterations, where in each iteration there is update to the corresponding weights. After all this routine, both A and B

hold the same weights vector in their models, which represent the private key. Now they can be sure that they can safely encrypt and decipher data with their keys. The keys transformed to hexadecimal values by hash function, to which they have previously agreed publicly.

The feedback creates correlations between the weights and the input, therefore the system become sensitive to the learning rule. The entropy is much smaller, because the values of the weights are pushed to the boundary values $\pm L$.



5.3 illustration- Two TPM's exchanging their key between them [8]

Security And Insecurity Of The Protocol

The neural network key exchange protocol doe's neither use number theory nor a public key. It based on learning process of neural network. For that, decipher the key is hard problem to any eavesdropper. Generally, if we look on brute force for example, the eavesdropper need to learn the synchronization of the weights vector. If we limit the weight in boundary of [-L, L], and our total input is in size of N*K, we got that a person who chooses to eavesdrop has $(2L+1)^{N*K}$ possibilities of combinations to get the correct key.

Assume that the eavesdropper, Eve, have the same network as Alice and Bob. It's mean that she has identical architecture as Alice and Bob machines, but she doesn't know the initial weights of them. Eve just want to eavesdrop, so she doesn't make some "noise" and mess all the data transportation, change it. She wants to synchronized her weights vector with Alice and Bob synchronized weights vector. One of the famous attacks is **Geometric attack**:

1. Output(A) ≠ Output(B): None of the parties updates its weights.
2. Output(A) = Output(B) = Output(E): All the three parties update weights in their tree parity machines.
3. Output(A) = Output(B) ≠ Output(E): Parties A and B update their tree parity machines, but the attacker can't do that. Because of this situation his learning is slower than the synchronization of parties A and B [6].

It has been proven, that the synchronization of two parties is faster than learning of an attacker. It can be improved by increasing of the synaptic depth L of the neural network. That gives this protocol enough security and an attacker can find out the key only with small probability. Breaking the security of neural key exchange belongs to the complexity class NP. [7][10][11]

A quantum computer is a device that uses quantum mechanisms for computation. In this device, the data are stored as qubits (quantum binary digits). That gives a quantum computer in comparison with a conventional computer the opportunity to solve complicated problems in a short time, e.g. discrete logarithm problem or factorization. Algorithms that are not based on any of these number theory problems are being searched because of this property. Neural key exchange protocol is not based on any number theory. It is based on the difference between unidirectional and bidirectional synchronization of neural networks. Therefore, something like the neural key exchange protocol could give rise to potentially faster key exchange schemes [*Neural Cryptography- Wikipedia*].
There are several types of attacks, that can be harmful to neural network in public channels exchanging:

Probabilistic Tree Parity Machine- a model of tree parity machine, "same" as the machines used by Alice and Bob, but the difference is that the weights of pTPM (probabilistic tree parity machine) are made of probabilities of $l \in [-L, L]$ to be the same l of A (or B). The eavesdropper build a pTPM and running the same time with A and B running. [6]

Genetic attack -In the genetic attack, the attacker starts with only one TPM but is permitted to use M TPMs. Because the most challenging issue in the mutual learning process is to predict the internal representation of either A or B , the genetic attack directly deals with this difficulty. For a specific value of $O^{A/B}$, there are 2^{K-1} different internal representations to reproduce this value. The genetic attack handles all these possibilities in a parallel fashion. The genetic attack proceeds as follows:

- If $O^A = O^B$ and E has at most $\frac{M}{2^{K-1}}$ TPMs, 2^{K-1} TPMs are generated and each updates its weights based on one of the possible internal representations. This step is known in genetic algorithms as the mutation step.

- If E has more than $\frac{M}{2k-1}$ TPMs, the mutation step will be essentially an overhead due to the exponential storage needed. Therefore, the attacker must discard some of the TPMs to avoid exponential storage increase. As a genetic algorithm, the discarding procedure is based on removing the TPMs with the least fittest function. The algorithm uses two variables *U* and *V* as the fitting functions. The variable *U* represents the number of correct prediction of O^A in the last *V* training steps. [11]

With the majority attack E can improve her ability to predict the internal representation of A's neural network. For that purpose, the attacker uses an ensemble of M Tree Parity Machines instead of a single neural network. At the beginning of the synchronization process the weight vectors of all attacking networks are chosen randomly, so that their average overlap is zero. Similar to other attacks, E does not change the weights in time steps with $\tau^A \neq \tau^B$, because the partners skip these input vectors, too. But for $\tau^A = \tau^B$ an update is necessary and the attacker calculates the output bits $\tau^{E,m}$ of her Tree Parity Machines. If the output bit $\tau^{E,m}$ of the m-th attacking network disagrees with τ^A, E searches the hidden unit i with minimal absolute local field $|{h_i}^{E,m}|$. Then the output bits ${\sigma_i}^{E,m}$ and $\tau^{E,m}$ are inverted similarly to the geometric attack. Afterwards the attacker counts the internal representations $({\sigma_1}^{E,m}, \dots, {\sigma_K}^{E,m})$ of her Tree Parity Machines and selects the most common one. This majority vote is then adopted by all attacking networks for the application of the learning rule. But these identical updates create and amplify correlations between E's Tree Parity Machines, which reduce the efficiency of the majority attack. Especially if the attacking neural networks become fully synchronized, this method is reduced to a geometric attack. In order to keep the Tree Parity Machines as uncorrelated as possible, majority attack and geometric attack are used alternately. the majority attack and the geometric attack are applied alternately in even and odd steps, respectively. [10]

Summary & Conclusions

The world of AI gives some new approach to the cryptography field. In this report, the algorithm described was one part from a huge advanced field, who takes a major place in our life. The report even didn't talk about the how we can encrypt data using neural networks, just present an algorithm to exchange keys via unsafe public channel. Neural networks give a new approach beside mathematics methods from the number theory. We saw that to an attacker, it could take exponential time to hack the neural network and get the same weights vector of Alice and Bob. One of the primary challenges in this field of neural cryptography appears to be the discovery of neural architectures with very high synchronization speed, and designing the encoding and entropy of the information exchanged during mutual learning, to prevent the synchronization of an attacker during the mutual learning process. The future reminds us that nothing is secure. The cyber world is entering the consciousness of many countries and societies, and is bringing in the best researchers and experts to pressure that the world of encryption will break. Mixing the AI world and the world of cryptography will bring new and far-reaching security technologies. Imagine that neural networks not only can create keys without both sides meeting, not only they will encrypt messages and simulate mathematical algorithms from the world of encryption, but also create their own encryption algorithm, an algorithm that will not be known to anyone and the key size will not be known to anyone, which will bring powerful security to both parties who want to send private messages, in contrast to *Kerckhoffs* principle. AI machines will gain the opportunity to create powerful crypto-system, unbreakable systems, and also… maybe they can hack crypto systems and crypto algorithms (e.g. RSA, SHA1, ElGamal encryption, etc.) that until today are considered to be unsolvable at a reasonable time?

List Of Appendices

Application Instructions

The project named "Ann" implement the neural network key exchange protocol. It's designed for two participants which run their models on the same simulation with the simulation function. The code was encoded by C++ language, using C++11 compiler on Visual Studio 2013 environment. Most of the project uses three libraries to implement the advanced part of the protocol- the first one was the open source library *Shark Machine Learning Library* that helped implement all the functions and design of the modules, by its advanced tools which know how to deal with everything related to machine learning and deep learning problems. The second one is the Crypto++, which is an open source library include hundreds of files help to handle with cryptography issues. The last one is the boost libraries, an advanced C++ opens source libraries that give a lot of advanced functionality to the application.

The project consists two files- main.cpp is the main entry of the application, and Ann.hpp is the header file of the application.

The main.cpp include two functions- "*int main*", which is the main entry and a function named *"void simulation"*, which handle simulation of two TPM machines: A and B. The simulation runs each machine separately and returns the identical final weight vector obtained after models synchronization.

Model_Ann
Model_Ann() Model_Ann(std::vector<int> parms, int lim) virtual ~Model_Ann() FFNet<H, O>* getModel() void HebbianFunction(vector<int> elements) int evalLayerModel() void UpdateWeightVector() void diagonal_hidden_layer(RealMatrix& matrix, RealMatrix& hidden_mat) RealMatrix getParameterWeight()
unsigned int numInput unsigned int numHidden unsigned int numOutput unsigned int limit FFNet<H, O>* network vector<int> diag_hidden_layer

Ann_Input
Ann_Input() void PsaudoRandomGenerate() Data<RealVector> getBatch() std::vector<RealVector> getInput() void renderInput() friend ostream& operator <<
Data<RealVector> data independent_bits_engine<mt19937,128,cpp_int> gen128; independent_bits_engine<mt19937, 256, cpp_int> gen256 vector<RealVector> local_input;

./Ann.exe -h

Presents a help menu for the user

./Ann.exe <-debug>

Running the application, with his default model architecture (for both A and B): number of hidden- 10, limit of weights- 3, number of input- 3 (total 30 "chars" on weight vector). The ***-debug*** option is to print the long synchronization that occurred during the process.

./Ann.exe <#hidden> <#limit> <#input> <-debug>

Running the application with custom parameters by the user (for both A and B): ***<#hidden>*** -number of hidden, ***<#limit>*** -limit of weights, ***<#input>*** -number of input. The ***-debug*** option is to print the long synchronization that occurred during the process.

For debugging the source, Shark Machine Learning library and Crypto++ needs to be install. To install SMLL, you need to go to SMLL website and download the package called "shark". Then, need to install CMake program and boost library.

To install Crypto++, need to download it from the website and then build it (or one of his projects) and copy it under same folder of Shark.

Bibliography

[1] Private Inputs to Tree Parity Machine, Pravin Revankar, W. Z. Gandhare and Dilip Rathod

[2] Cryptanalysis, 236500 Prof. Eli Biham, Computer Science department Technion, Haifa, 2016

[3] What's the difference between AI, ML and DL? Nvidia.com blogs, by Michael Copeland

[4] Neural cryptography with queries, Andreas Ruttor, Wolfgang Kinzel and Ido Kanter

[5] Neural Cryptography with Multiple Transfers Functions and Multiple Learning Rule, N.Prabakaran , P.Loganathan, P.Vivekanadan, Department of Mathematics, Anne University, Chennai. India

[6] Probabilistic Attack on Neural Cryptography, Master thesis- Lu´ıs Francisco Seoane Iglesias

[7] Neural Synchronization based Secret Key Exchange over Public Channels: A survey, Sandip Chakraborty, Jiban Dalal, Bikramjit Sarkar, Debaprasad Mukherjee, Dept. of Computer Science and Engineering and Dept. of Information Technology Dr. B. C. Roy Engineering College (West Bengal University of Technology), India

[8] Cryptography based on Neural Network, Michal Janosek, Eva Volna, Martin Kotyrba, Vaclav Kocian, "Cryptography based on Neural Network" Proceedings 26th European Conference on Modeling and Simulation, Year 2012.

[9] Genetic attack on neural cryptography Andreas Ruttor and Wolfgang Kinzel Institut fur¨ Theoretische Physik, Universit¨at Wurzbur ¨ g, Am Hubland, 97074 Wurzbur ¨ g, Germany Rivka Naeh and Ido Kanter Minerva Center and Department of Physics, Bar Ilan University, Ramat Gan 52900, Israel

[10] Neural Synchronization and Cryptography, Andreas Ruttor, Wurzburg 2006

[11] Use of Artificial Neural Networks in Cryptography, Martin Javurek, Michal Turčaník, Marcel Haraka