Neural cryptography and tree parity machines

Jan Pelicon

Affiliation: Fakulteta za računalništvo in informatiko
E-mail: jp1467@student.uni-lj.si

Abstract. Combining fields of cryptanalysis and artificial neural networks, a neural key exchange protocol has been defined using tree parity machines. Parties that want to securely communicate, share the same network architecture and can synchronize network weights over time by mutual learning, exchanging information over a public channel in the process. Once synchronized, both networks have identical weight vectors which can be used as a one-time pad (OTP), as a seed for random bit generators or as a key in other encryption algorithms. Although tree parity machines have low time and memory complexity, their key exchange protocol can be broken in three different ways, making it insecure.

Keywords: Neural cryptography, artificial neural networks, tree parity machine, key exchange

Nevronska kriptografija in paritetna drevesa

Preko področij kriptoanalize in umetnih nevronskih mrež je definiran nevronski protokol za izmenjavo kriptografskih ključev, ki za delovanje uporablja paritetna drevesa. Stranki, ki si želita varen prenos informacij, morata imeti enako arhitekturo nevronske mreže, ki si s postopkom obojestranskega učenja, pri katerem si mreži izmenjujeta informacije preko javnega kanala, postopoma sinhronizirata vektorje uteži. Po sinhronizaciji imata obe mreži identičen vektor uteži, ki se lahko uporabi kot OTP, seme za naključni generator ali kot ključ za druge kriptografske algoritme. Čeprav imajo te mreže majhno časovno in prostorsko zahtevnost je bil protokol za izmenjavo ključev razbit in zaradi tega ni varen.

1 History

Before the Diffie-Hellman key exchange method was conceived in 1976, cryptographic keys had to be transferred over a secret channel to ensure secure communication. The Diffie-Helman method ensures that a shared secret can be safely exchanged over a public channel, assuming that an attacker has a limited amount of time and processing power.

Since artificial neural networks were formalized in the late 1940s, they have been widely used for many tasks in different areas of science. Researchers have also been trying to use neural networks for cryptography. They were first used in cryptanalysis of the DES algorithm.

In January 2002, the Physicists Kanter, Kinzel and Kanter proposed a new key exchange protocol between two parties A and B using a tree parity machine, which is a relatively simple fully connected feedforward neural network.

2 Tree parity machines

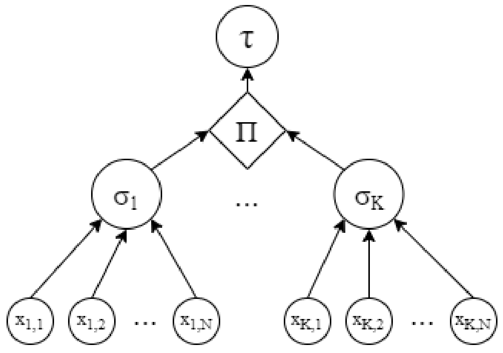

Figure 1. Tree parity machine architecture.

Figure 1 shows a tree parity machine which is a multilayer network that consists of an input layer x_i, a single hidden layer with perceptrons σ_i and an output neuron τ.

2.1 Network parameters

Network properties are defined by parameters K, N and L, where K represents a number of perceptrons σ_i in hidden layer, N is dimensionality of input vector x_i and L is synaptic depth that defines the interval of weight values with the size of $2L+1$:

$$w_{i,j} \in \{-L, -L+1, \ldots, L-1, L\}.$$

There are also other requirements for network in order to behave in a desired way:

$$x_{i,j} \in \{-1, 1\},$$

$$\sigma_i = sign\left(\sum_{j=1}^{N} w_{i,j} x_{i,j}\right),$$

where $sign(x)$ function is defined as:

$$sign(x) = \begin{cases} -1 \; if \; x < 0, \\ +1 \; if \; x > 0. \end{cases}$$

The output τ is calculated as a product of all perceptrons σ and thus it can be represented as a boolean as values can only be -1 or +1:

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

2.2 *Training algorithm*

Both A and B follow a training algorithm which ensures that after its completion, A and B will be fully synchronized, meaning their weights vectors are identical. Before training A and B initialize random weights values. At each training step A and B receive identical input vectors $x_1, x_2, \dots, x_N$ and calculate outputs τ_A and τ_B. Only if outputs are identical $\tau_A = \tau_B$, the weights can be changed using the Hebbian rule:

$$w_i(t+1) = w_i(t) + x_i.$$

In case weight value is not in the interval defined by L, the value is replaced by $\pm L$. Only weights w_i under the perceptron σ_i with positive value $+1$ are changed. The two networks A and B perform a synchronized random walk in the discrete space of $(2L+1)^{KN}$ points. Synchronization of neural networks can immediately be translated to key generation in cryptography. The distribution of synchronization time for a network with parameters $K = 3, N = 100, L = 3$ has a peak at $\cong 400$ training steps.

3 CRYPTANALYTIC ATTACKS

If an attacker E, who owns same tree parity machine as the parties A and B, wants to synchronize the machine with parties A and B, just by observing the inputs x_i and outputs τ_A and τ_B, there are three possible situations for each training step:

- $\tau_A \neq \tau_B \rightarrow$ None of the parties update their weights
- $\tau_A = \tau_B = \tau_E \rightarrow$ All parties update their weights
- $\tau_A = \tau_B \neq \tau_E \rightarrow$ Parties A and B update their weights

This ensures that parties A and B learn faster than attacker E. Larger synaptic depth L scales the problem exponentially for an attacker, thus lowering the probability of successful brute force attack. For a network with $K = 3, N = 100, L = 3$ there are $3.38 * 10^{253}$ key possibilities, making the attack impossible with today's computer power. There are however at least three successful types of attacks against tree parity machines.

3.1 *The genetic attack*

An inspiration for a genetic attack came from genetic algorithms. Attacker E creates a large population of tree parity machines and trains his networks with the same inputs x_i as parties A and B. On average, half of the population will have the same output τ after the first training step and will create copies of themselves, continuing the process. Researchers found that 50% of the time at least one network shared the same weights as parties A and B using the population of 2500 networks with parameters $K = 3, N = 101, L = 3$.

3.2 *The geometric attack*

Researchers have found that, if an attacker network E is close to the network A, but its output τ differs from output of A, it is possible to guess which weights need updating, using the following rule for third situation: $\tau_A = \tau_B \neq \tau_E \rightarrow$ Parties A and B update their weights, attacker E finds $i_0 \in \{1, \dots, K\}$ that minimizes $\left|\sum_{j=1}^{N} w_{i,j} x_{i,j}\right|$. An attack has been tried with 100 networks with random initial states and 90% of the time at least one of them synchronized faster than B.

3.3 *The probabilistic attack*

Even though attacker E does not know the initial state of parties A and B, after a number of training steps it becomes easier to predict the state. This attack considers each weight independently while training and tracks distribution of probabilities for each possible weight ($2L+1$ states). After parties A an B converge to a common set of weights $w_{A,B}$ and stop the protocol:

$$Pr[w_C = w_{A,B}] \approx 1,$$

this means that weight states can be easily predicted most of the time.

4 CONCLUSION

Though neural key exchange protocol with tree parity machines is not an alternative to already established Diffie-Hellman key exchange, it shows that there are other ways of deriving a shared secret using a public channel. There is a possibility that future reserach will be able to overcome the shortcomings of this method and hopefully we could have a fast and secure alternative to the Diffie-Hellman key exchange protocol.

REFERENCES

[1] Kinzel, W. & Kanter, Ido. (2002). Neural cryptography. 3. 1351 - 1354 vol.3. 10.1109/ICONIP.2002.1202841.

[2] Klimov, Alexander & Mityagin, Anton & Shamir, Adi. (2002). Analysis of Neural Cryptography. 2501. 288-298. 10.1007/3-540-36178-2_18.

[3] Seoane, Luís & Ruttor, Andreas. (2012). Successful attack on permutation-parity-machine-based neural cryptography. Physical review. E, Statistical, nonlinear, and soft matter physics. 85. 025101. 10.1103/PhysRevE.85.025101.

[4] Dourlens, Sébastien. (1995). Neuro-Cryptographie Appliquée et Neuro-Cryptanalyse du DES. 10.13140/RG.2.2.35476.24960.