An Encryption cum Hash Function Using Permutation Parity Machine

Shiffin N,[1]* Y Sree Chaitanya,[1] Rajat Pandey,[1] Ravi Jha[1]

Abstract: In the modern world computing has become pervasive and has led to expansive usage of resource constraint devices like RFID devices, IoT devices, embedded systems and other smart devices. Though these devices allow us to run a variety of services with ease, there is a lot of sensitive information that is accessed through these resource constraint devices raising security concerns.The traditional cryptographic techniques are too big, too slow or too energy-consuming for efficient use on such devices. In this paper we present a novel technique to address this issue. We propose use of Permutation Parity Machine (Neural cryptographic technique) for key exchange, encryption and hash functions.Subsequently we demonstrate its performance against common types of attacks and the results of testing it against standard NIST test suites.

Asian Journal of Engineering and Technology Innovation
Volume 4, Issue 7
Published on: 7/05/2016

Cite this article as: Shiffin N *et al.* Intrusion Detection using Passive Infrared Sensor(PIR). Asian Journal of Enginnering Technology and Innovation, Vol 4(7):132-133, 2016.

INTRODUCTION

As the technology became pervasive, the necessity and prevalence of the resource constraint devices like RFID devices[3], IoT devices, embedded systems and other smart objects also grew exponentially. The services run on or through these systems mostly involve use or access to sensitive information. Therefore, there is a need to study the security aspects of these systems. Traditional cryptographic techniques have been applied to these systems, but were rendered inefficient due to low memory, low power and other resource constraints. Most of the algorithms were too slow, too big or highly energy consuming raising the need for a lightweight cryptographic algorithm.

Existing light weight techniques use a set of internal registers and initialization vectors to generate pseudo random stream ciphers. Stream Ciphers like trivium, Grain128a fall into this category.

Recent studies have shown techniques based on neural cryptography perform better for situations involving resource constraint end devices.Neural cryptographic techniques have also been proved to be robust against geometric attacks and other statistics and mathematics based attacks.

In the recent times there have been studies on use of neural network based protocols, particularly tree parity machines for exchanging key or initial secret information that could be used to generate the key. These techniques are based on mutual synchronization capability of artificial neural networks.

This paper proposes another such technique, based on the use of permutation parity machines, to develop a full scale encryption and authentication system for light-weight devices. The basis of this branch of cryptography is briefly pictured in Sec. 2. Section 3 describes the permutation parity machine and its learning rule. In Sec. 4, the encryption protocol is discussed. Sec. 5 describes the hash generation based on the PPM concept. We conclude with the results of testing the key-exchange protocol, the stream cipher generation technique for encryption and hashing method against standard test suites.

NERUAL NETWORKS AND CRYPTOGRAPHY

Traditional Symmetric key encryption and related ideas are based on all the parties involved in the communication (say Alice and Bob), share a common secret generated by one participant or commonly decided or agreed upon by all the parties by using a private channel or by contact[4]. Establishment of such a channel is too costly and is also prone to eavesdropping (Attacks from Darth).

Neural network based key-agreement or key-exchange protocols are based on the ability of neural networks to synchronize. The mechanisms involve use of neural networks(also called parity machines) initialized with random weights at each end. These parity machines are fed with random publicly known input. Machines mutually synchronize by adjusting their structure depending on the output of the other machines (publicly transmitted) for the same input.

[1]Reva Institute of Technology and Management, Rukmini Knowledge Park, Kattigenahalli, Yelahanka, Near Border Security Bustop, Bengaluru, Karnataka-560064, India.
E-mail: nshiffin94@gmail.com
*Corresponding author

The neural key-agreement protocols allow passive attacks as the messages exchanged for synchronizing are public. Since the attacker does not know about the initial random weights of any parties involved, he cannot synchronize his machine with those involved in conversation [2].

PERMUTATION PARITY MACHINE

Permutation parity machines are multilayer neural networks with binary weights consisting of K hidden units each with N receptive fields and an output[1].

Each of these are fed binary input, weights are drawn from pool of binary numbers called state vector of length G. The synaptic weights are drawn based on a matrix π with K*N randomly generated values that select bits of state vector to be used as weights. K, N, G are positive integers[1].

Each individual unit computes exclusive or between weights and inputs.

$$hi=XOR\ (w, input)$$

Output value of each hidden neuron is computed as result of threshold function on sum of all these XORs.

$$\sigma_i = \theta_N(h_i)$$

The activation function of PPM is results in a high(1) if input is greater than N/2 low otherwise.

The output of the machine is XOR of all hidden neuron outputs.

$$\tau = xor_{i=1}^{k}\ \sigma_i$$

SYNCHRONISATION BY MUTUAL LEARNING AND ENCRYPTION

Permutation parity machines involved in the conversation mutually synchronize by exchanging their outputs and adjusting the state vectors accordingly, leading eventually to common state vectors for the machines [2]. The synchronization process consists of several iterations with each iteration consisting of sub iterations (also called outer and inner rounds).

Inner iterations fill empty buffers with output of hidden neuron which are copied into the state vector to complete an outer iteration.

Each of these rounds involve choosing a random K*N matrix and random inputs to feed the neurons and exchanging the output of the PPMs.(^{A}and τ ^{B}denote output of the machines A and B, respectively). During each of these rounds either the overlap between the buffer and the state vector either increases or decreases. Refer to papers on PPM for clarity and further information[1].

The secret in this protocol is given by the initial values ofthe state vectors. The technique does not involve any use of huge numbers or any number theory methods.

After the machines are synchronized both the parties involved share a secret which is the structure of the PPM (along with its weights drawn from state vector). The required stream cipher can be generated by continuing the random input to the synchronized network that gives the same output on both ends (because they share the same structure) which form the bits of stream cipher. This continuous feed of input to the machine lasts until the cipher length matches the message length.

Once the cipher of required length is obtained it is XORed with the message and hashed using the algorithm presented in the next section. The message then along with the appended hash is transmitted to the receiver.

HASH FUNCTION USING PPM

Algorithm

```
1. Divide the cipher text into "N" blocks of 32-bits
2. For i=0 to N:
   • Pass ciphertext_block[i] as input to ppm
   • Calculate $h_i=x_i xor w_i$
   • Calculate $\sigma_i$and $\sigma_{i+1}$
   • Append $\sigma_i$and $\sigma_{i+1}$ to the 2 extreme ends of state vector.
   • Calculate pie matrix from the values of $h_i$.
   • If i==N:
      ▪ Take the pie matrix and select those values from the
        state vector and these values will be the hash of the
        cipher text.
```

CONCLUSION

In this paper, at first we have described the shortcomings of security protocols for resource limited devices.Then, we proposed usage of neural cryptographic techniques for such systems.Finally discussed an implementation of one such algorithm based on PPM for stream cipher generation, hashing and key exchange that could be used to secure the use of these light-weight devices.

The discussed protocols and techniques passed the NIST STS for stream cipher generation and have proven to be robust against collision attacks and geometric attacks.

REFERENCES

1. Oscar Mauricio Reyes1,2,*and Karl-Heinz Zimmermann1 "Permutation parity machines for neural cryptography" , 2010
2. O. M. Reyes, I. Kopitzke, and K.-H. Zimmermann, J. Phys. **42**, 195002 _2009_.
3. A. Juels, "RFID security and privacy: A research survey," IEEE Journal on Selected Areas in Communications, vol. 24, no. 2, pp. 381–394, 2006.
4. Menezes, P. van Oorschot, and S. Vanstone, *Handbook of Applied Cryptography* _CRC Press, Boca Raton, FL.