Diffie-Hellman Protocol with a Combination of Hyperelliptic Curves and Neural Synchronization

Protocolo Diffie-Hellman con una Combinación de Curvas Hiperelípticas y Sincronización Neuronal

Angelo Araya Villanueva[1] Ivan Jiron Araya[2] Ismael Soto Gomez[3]

Recibido 27 de junio de 2018, aceptado 06 de agosto de 2018
Received: June 27, 2018 Accepted: August 06, 2018

ABSTRACT

This work proposes a new cryptosystem, combining a Diffie-Hellman protocol in which hyperelliptic curves over GF(2^n) are implemented, with a Tree Parity Machine (TPM) synchronization. Security proposed for this cryptosystem is focused on overcoming a weakness of neuronal synchronization. Specifically, the stimulus vector that is public, which allows an attacker to try to synchronize with one of the participants of the synchronization. Focusing on this weakness, there are the following attacks: genetic attack, geometric attack and probabilistic attack. In the proposed cryptosystem, the initial stimulus vector will be hidden, because this vector is obtained as the common secret key in the Diffie-Hellman protocol. Then in each iteration, the stimulus vectors will be kept secret. This condition causes the learning time t_{lear} to increase by a term of approximately 115% regarding the synchronization time t_{sync} on average when the proposed cryptosystem is compared to the classic TPM synchronization.

Keywords: Cryptography, Hyperelliptic Curves, Neural Synchronization.

RESUMEN

Este trabajo propone un nuevo criptosistema, que combina el protocolo Diffie-Hellman en el cual se implementan curvas hiperelípticas sobre GF(2^n), con la sincronización de Tree Parity Machines (TPM). La seguridad propuesta para este criptosistema se centra en superar una debilidad de la sincronización neuronal. Específicamente, que el vector de estímulos es público, lo cual permite a un atacante intentar sincronizar con uno de los participantes de la sincronización. Enfocándose en esta debilidad, existen los siguientes ataques: simple, genético, geométrico y probabilístico. En el criptosistema propuesto, el vector de estímulo inicial se encuentra oculto, porque este vector se obtiene como la clave común secreta en el protocolo Diffie-Hellman. Luego, en cada iteración, los vectores de estímulo se mantendrán en secreto. Esta condición hace que el tiempo de aprendizaje t_{lear} aumente en aproximadamente 115% con respecto al tiempo de sincronización t_{sync} en promedio, cuando el criptosistema propuesto se compara con la sincronización de TPM clásica.

Palabras clave: Criptografía, Curvas Hiperelípticas, Sincronización Neuronal.

[1] Departamento de Ingeniería de Sistemas y Computación. Universidad Católica del Norte. Avda. Angamos 0610. Antofagasta, Chile. E-mail: angelo.araya@ucn.cl
[2] Departamento de Matemáticas. Universidad Católica del Norte. Avda. Angamos 0610. Antofagasta, Chile. E-mail: ijiron@ucn.cl
[3] Departamento de Ingeniería Eléctrica. Universidad de Santiago de Chile. Avda. Ecuador 3519, Santiago, Chile. E-mail: ismael.soto@usach.cl

INTRODUCTION

Cryptography is the practice and study of techniques for secure communications, it has been approached by many researchers in several applications as the well-known cryptographic protocol of public-key Diffie-Hellman and the ElGamal encryption [1], another of these public-key applications is the one based on Hyperelliptic Curves [2], despite its computational complexity, this application offers security within smaller keys. Another public-key application is the use of Neural Networks Synchronization [3], which is based on the exchange of information between two neural networks ending in the synchronization of their hidden weights acting as the secret key on a communication. This paper is organized as follows. Section 2 introduces an overview of TPM neural networks and their synchronization. Section 3 presents the essential definitions about hyperelliptic curves. Section 4 provide a detailed explanation and the highlights of the proposed cryptosystem. Section 5 presents the analysis of results. Section 6 presents the conclusions and future work.

NEURAL NETWORKS

Generally, a neural network is a machine that is designed to model the way in which the brain performs a single task or function of interest; the network is usually implemented by using electronic components or is simulated in software on a digital computer [4].

One special kind of neural network called Tree Parity Machine (TPM) are used for a secure key exchange, it is based on the synchronization of two of them [3]. Each TPM has the following elements: it has only one output τ, K hidden neurons and $K * N$ input units. The input units have values $x_{ij} \in \{-11\}$. The synaptic weights are $w_{ij} \in \{-L, 0, .., L\}$ where $L \in \mathbb{Z}$ they are selected previously and each part selects initially their own weight vector $\vec{W} = \left(w_{ij}\right)$ randomly. The output σ_i of $i-th$ neuron is given by (1)

$$\sigma_i = \operatorname{sgn}\left(\sum_{j=1}^{N} w_{ij} w_{ij}\right) \tag{1}$$

where $sgn\,(\cdot)$ is defined by (2)

$$sgn\left(x\right) = \begin{cases} -1, x \le 0 \\ 1, x > 0 \end{cases} \tag{2}$$

And the output τ of the TPM is given by (3)

$$\tau = \prod_{i=1}^{K} \sigma_i \in \{-1,1\} \tag{3}$$

Also, it is necessary to choose a learning rule that adjust the weights because the initial weights in every TPM are different (selected randomly), and is necessary to make them identical to complete the synchronization process. Note that the sender output τ^A goes to the receiver, and the receiver feeds back his output τ^B to the sender. Then, both networks are trained with the output of its partner with the learning rule (4)

$$w_{i,j}^{+} = g\left(w_{i,j} + x_{i,j}\theta\left(\sigma_i \tau\right)\theta\left(\tau^A \tau^B\right)\right) \tag{4}$$

where $\theta(\cdot)$ is the Heaviside function and, $g(w) = sgn(w) \cdot L$ for $|w| > L$, and, $g\,(w) = w$ for $|w| \le L$.

Only weights belonging to the one hidden units which are in the same state as that of their output unit are updated, in each one of the networks. Note that, using this dynamical rule, the sender is trying to imitate the response of the receiver and the receiver is trying to imitate the one of the sender. This rule (Random Walk) has been selected over others because all other suitable learning rules (Hebbian and Anti-Hebbian) converge to it in the limit $N \to \infty$ [5].

HYPERELLIPTIC CURVES

A Galois field $GF(p^n)$ is a finite set with two operations, addition '+' and multiplication '*', such that $(GF(p^n),+)$ is a commutative group. The nonzero elements together with the multiplication $(GF(p^n)-\{0\}, *)$ form a commutative group. Furthermore, the product '*' is distributive over the addition [7, 8].

Galois fields $GF(p^n)$ can be built from $GF(p)$. And these fields are called extensions of $GF(p)$. Primitive polynomials $P(x)$ of degree n defined over $GF(p)$, are used for this construction [8]. Thus, the field

$GF(p^n)$ has p^n-1 distinct elements. Each non-zero element of $GF(p^n)$ can be represented as (5)

$$\alpha^m = \sum_{i=0}^{n-1} \alpha_i \alpha^i ; \alpha_i \in GF(p); m = 0,1,\ldots p^n - 2 \quad (5)$$

where α is a root of the primitive polynomial $P(x)$ chosen for the construction.

Example 1.Galois field $GF(2^5)$ is an extension of $GF(2)$, which has been constructed using the primitive polynomial $P(x)= x^5 + x^2 + 1$ of degree $n = 5$ defined over $GF(2)$. The Table 1 shows some elements of this field [6].

Table 1. Some elements of field $GF(2^5)$.

$\underline{k}$	$\underline{\alpha^k}$	$\underline{k}$	$\underline{\alpha^k}$
0	1	3	α^3
1	α	4	α^4
2	α^2	1	$\alpha^2+\alpha+1$

Hyperelliptic curves are a special class of algebraic curves and can be viewed as generalizations of elliptic curves [2, 6]. They hold a series of definitions and properties that can be viewed next.

Definition 1: A hyperelliptic curve C of genus $g \geq 1$ over a Galois field $GF(p^n)$ is an equation of the form (6)

$$C : v^2 + h(u)v = f(u) \quad (6)$$

where $h(u), f(u)$ are polynomials with coefficients in $GF(p)$. Further, $h(u)$ has degree at most g, and $f(u)$ is a monic polynomial of degree $2g+1$. And, there are no solutions $(u,v) \in GF(p^n) \times GF(p^n)$ which simultaneously satisfy the hyperelliptic curve equation and their partial derivatives $2v + h(u) = 0$ and $h'(u)\, v - f'(u) = 0$.

Example 2: The hyperelliptic curve C: $v^2 + (u^2 + u)\, v = u^5 + u^3 + 1$ over $GF(2^5)$ has the polynomials $h(u) = u^2 + u$ and $f(u) = u^5 + u^3 + 1$. Some of the points that satisfy the equation of C are listed in Table 2.

A divisor is a formal sum of points on C given by (7)

Table 2. Some points of C: $v^2 + (u^2 + u)\, v = u^5 + u^3 + 1$ over $GF(2^5)$.

(0,1)	**(1,1)**	**(α^5, α^{15})**	**(α^7, α^4)**	**(α^7, α^{25})**
(α^9, α^{27})	(α^9, α^{30})	$(\alpha^{10}, \alpha^{23})$	(α^{14}, α^8)	$(\alpha^{14}, \alpha^{19})$
$(\alpha^{20}, \alpha^{15})$	$(\alpha^{20}, \alpha^{29})$	$(\alpha^{23}, 0)$	(α^{25}, α)	$(\alpha^{25}, \alpha^{14})$
$(\alpha^{27}, 0)$	(α^{27}, α^2)	(α^{28}, α^7)	$(\alpha^{29}, 0)$	(α^{29}, α)

$$D = \sum m_i P_i - \left(\sum m_i\right)\infty \quad m_p \in \mathbb{Z} \quad (7)$$

where only a finite number of the integers m_p are nonzero, and ∞ is the point at infinity in the projective plane, for more details about these topics the reader can review the appendix in [2].

A divisor can be represented as two polynomials as stated in the next theorem:

Theorem: Let $D = \Sigma m_i P_i$ - $(\Sigma m_i)\, \infty$ be a divisor, where $P_i = (x_i, y_i)$. Let $a(u) = \prod(u - x_i)^{mi}$. There exists a unique polynomial $b(u)$ satisfying: (1) $deg_u b < deg_u a$; (2) $b(x_i) = y_i$ for which $m_i \neq 0$; and (3) $a(u)$ divides $(b(u)^2 + b(u)\, h(u) - f(u))$. Then $D = (a(u), b(u))$.

The divisors addition is developed with the two following algorithms:

Algorithm 1 [2]
INPUT: Divisors $D_1 = (a_1(u), b_1(u))$ and $D_2 = (a_2(u), b_2(u))$ both defined over $GF(p^n)$.
OUTPUT: A divisor $D = (a(u), b(u))$ defined over $GF(p^n)$ such that $D = D_1 + D_2$.

1. Use the Euclidean algorithm to find polynomials $d_1, e_1, e_2 \in GF(p^n)[u]$ where $d_1 = g.c.d(a_1, a_2)$ and $d_1 = e_1 a_1 + e_2 a_2$.
2. Use the Euclidean algorithm to find polynomials $d, c_1, c_2 \in GF(p^n)[u]$ where $g.c.d\ (d_1, b_1 + b_2 + h)$.
3. Let $s_1 = c_1 e_1$, $s_2 = c_1 e_2$, and $s_3 =$ and $s_3 = c_2$, so that $d = s_1 a_1 + s_2 a_2 + s_3(b_1 + b_2 + h)$
3. Set $a = a_1\, a_2 / d^2$

And

$$b = \frac{s_1 a_1 b_2 + s_2 a_2 b_1 + s_3\left(b_1 b_2 + f\right)}{d}$$

Algorithm 2 [2]
INPUT: A divisor $D = (a(u), b(u))$ defined over $GF(p^n)$.

OUTPUT: The (unique) reduced divisor $D' = (a'(u), b'(u))$ such that $D' = D$.

1) Set
$$a' = (f - bh - b^2)/a$$
And
$$b' = (-h - b) \bmod a',$$
2) If $deg_u a' > g$ then set $a \leftarrow a'$, $b \leftarrow b'$, and go to step 1.
3) Let c be the leading coefficient of a' and set $a' \leftarrow c^{-1}a'$.
4) Output $(a'(u), b'(u))$.

PROPOSED CRYPTOSYSTEM

The proposed cryptosystem consists in a Diffie-Hellman cascade implementation with two layers. In the first layer Diffie-Hellman key exchange is implemented with divisors on a hyperelliptic curve (D-H HC). And in the second layer Diffie-Hellman key exchange is implemented with TPM synchronization (D-H TPM S). Figures 1.a and 1.b show the system diagram of the proposed cryptosystem.

For a better understanding, first it is necessary to clarify which elements are public. In the first level D-H HC the public elements are: the hyperelliptic curve that is used C, the initial reduced divisor D and the public divisors D_A, D_B, which are the public keys of Alice and Bob, respectively. In the second layer the public elements are: the TPM network, the output of both neural networks ρ_A, ρ_B used on the synchronization process, and the shift numbers (S_A, S_B) that will change the input value. Next the key exchange is explained in detail.

Starting at the top of the diagram there are the private keys k_a, k_b of Alice and Bob respectively. Each one of them use his private key to multiply the public divisor D and reduce the result to obtain their own public key divisors ($D_A = k_a D$, $D_B = k_b D$) using algorithms 1 and 2. After this, they exchange

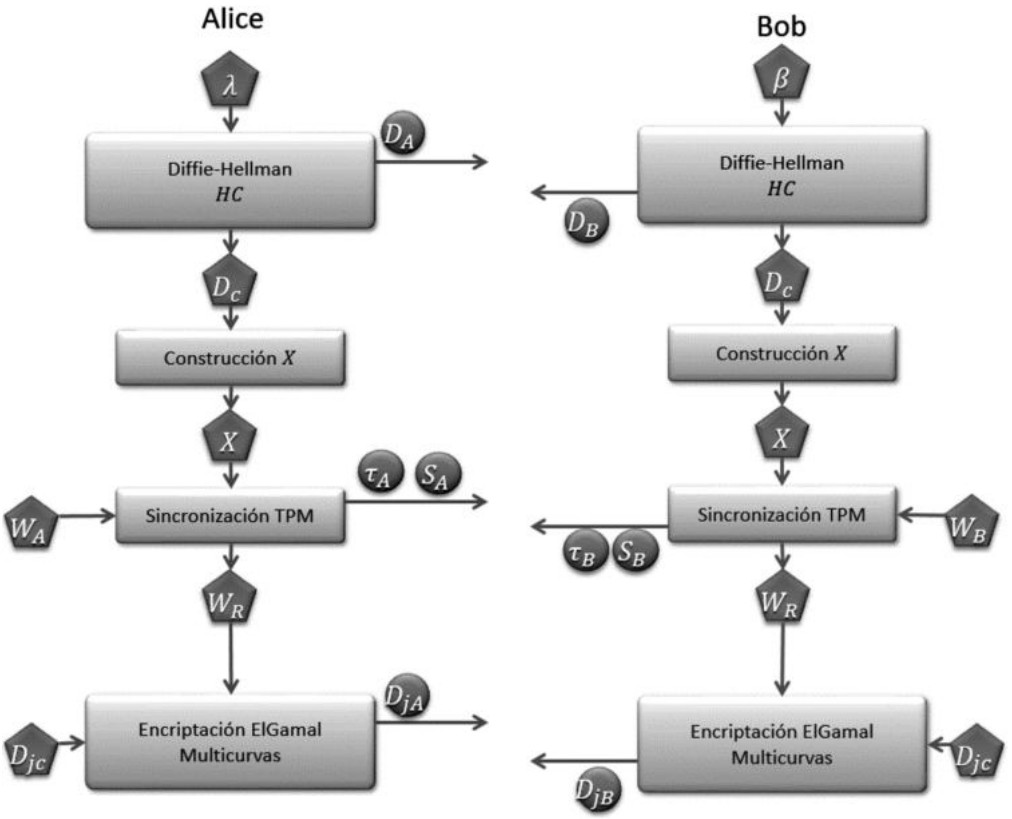

Figure 1.a System Diagram. b) System Diagram.

the divisors D_A, D_B, and apply their secret key over the received divisor and reduce the result again to obtain a common divisor (D_c, $k_a = k_b D_A$) that belong to the curve C.

Now Alice and Bob have a secret common divisor, each one takes this and send it to their Binary Box which will take the divisor and convert it to its binary form, for example let say that the common divisor is $D_c = (u^2 + a^{15}u + a^{26}, a^{23}u + a^{21})$ which belong to the curve C in example 2 will be transformed into {0,0,0,0,1,1,1,1,1,1,1,0,0,1,1,1,0,0,1,1,1,1,1,1,0,0,0}, and this array will be transformed into stimulus vector $\vec{X}_0$ = {-1,-1,-1,-1,1,1,1,1,1,1,1,1,-1,1,1,1,-1,1,1,1,1,1,1,1,-1,-1,-1}.

This conversion is necessary because it is needed in the next level where the TPM will be feed. Then, $\vec{X}_0$ is secret, unlike in the original TPM synchronization where the input values are public. Here it will remain secret in all the synchronization iterations.

Once Alice and Bob has reached this level both sides will select their own weights vector $\vec{W}_A, \vec{W}_B$ randomly and secret, in every step of the synchronization they will exchange their output ρ_A, ρ_B to adjust their weights, and a shift number S_A, S_B that will move the components in vector $\vec{X}_0$ to the right $S_A + S_B$. This way the input values will be kept secret and after the process both sides will have a secret common set of weights $\vec{W}_c$.

RESULTS DISCUSSION

Focusing in the weakness of the neural synchronization, is necessary to clarify that:

In an attack of genetic type, the intention is to define the evolution of the population formed by the stimulus vectors of each TPM used in each iteration to predict the weights [9,10].

In an attack of geometric type, the intention is to define the surface formed by the stimulus vectors of each TPM used in each iteration to predict the weights [9,10].

In a probabilistic attack, the intention is to define the probability distribution for the stimulus vectors of each TPM used in each iteration and try to predict the weights vector [9,10].

By cascading the Diffie-Hellman protocol implemented with hyperelliptic curves with the neuronal synchronization of TPMs, the stimulus vector is hidden in each iteration, keeping the first vector of stimuli secret $\vec{X}_0$, and making slides on it at each iteration.

Thus, the learning time t_{lear}, which corresponds to the time necessary for the attack, increases, making very difficult for the intruder to synchronize his TPM with one used by Alice or Bob.

Using the experimental results presented in [11, 12] it is possible to estimate the growth of learning time t_{lear}, as a time-dependent synchronization variable t_{sync}, In particular, the authors of [11, 12] measure t_{sync} and t_{lear} for L = 1,2,3,4. Where t_{sync} and t_{lear} are the number of steps to synchronize. These measurements are shown in Table 3.

Table 3. Measurement of t_{sync} and t_{lear} times for L = 1,2,3,4 [11, 12].

L	t_{sync}	t_{lear}
1	61 ± 10	$1.1\times10^2 \pm 0.2\times10^2$
2	188 ± 26	$1.5\times10^3 \pm 0.5\times10^3$
3	376 ± 51	$4.5\times10^4 \pm 1.3\times10^4$
4	673 ± 95	$6.9\times10^7 \pm 5.7\times10^7$

Then, from the analysis of these measurements it can be established that t_{lear} grows exponentially in respect of t_{sync}. Figure 2 shows the growth of t_{lear}

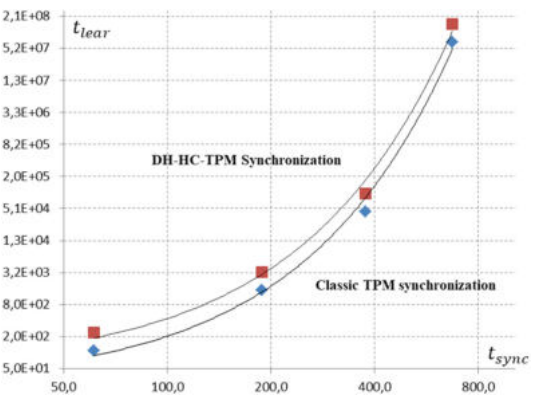

Figure 2. Comparison between Classic TPM synchronization and DH-HC-TPM Synchronization.

as a function of t_{sync}. On average the learning time increases in a term of approximately 115% for 100 ≤ 400, when using the combination of the Diffie-Hellman protocol implemented with hyperelliptic curves with neuronal synchronization of TPMs (DH-HC-TPM Synchronization).

CONCLUSIONS

In this paper has been presented a new cryptosystem that combines Diffie-Hellman protocol using hyperelliptic curves and a public-key exchange based on neural synchronization. Then, in this cryptosystem the initial stimulus vector will be hidden, because this vector is obtained as the secret common key in Diffie-Hellman protocol. Then, in each iteration, the stimulus vectors will be kept secret. This condition makes that the learning time t_{lear} increases in a term of approximately 115% in respect of synchronization time t_{sync} on average, when the proposed cryptosystem is compared to the classic TPM synchronization.

As future works the synchronization for other neural network topologies will be studied. Furthermore, the proposed algorithm will be evaluated on a Visible Light Communication (VLC), Fading, and Wired channels.

REFERENCES

[1] C. Paar, J. Pelzl. "Understanding Cryptography". Springer-Verlag, Berlin Heidelberg, 2010. ISBN 978-3-642-04100-6.

[2] N. Koblitz. "Algebraic Aspects of Cryptography". Springer-Verlag, 1999. ISBN 3-540-63446-0.

[3] Kanter, W. Kinzel, and E. Kanter. "Secure exchange of information by synchronization of neural networks", Europhys. Lett., vol. 1, Nº 1, p. 11, 2002.

[4] S. Haykin. "Neural Networks and Learning Machines". ISBN 978-81-203-4000-8, 2011.

[5] Ruttor, W. Kinzel, I. Kanter. "Dynamics of neural cryptography," Phys. Rev. E - Stat. Nonlinear, Soft Matter Phys., vol. 75, Nº 5, pp. 1-9, 2007.

[6] H. Cohen, G. Frey. "Handbook of Elliptic and Hyperelliptic Curve Cryptography". Chapman & Hall/CRC, 2006. ISBN 1-58488-518-1.

[7] T.W. Hungerford. "Algebra". Springer; 8th ed., 2003. ISBN-10: 0387905189.

[8] S. Lin, D.J. Costello. "Error Control Coding: Fundamentals and Applications". Pearson Prentice Hall, 2004. ISBN 0-13-017973-6.

[9] S. Santhanalakshmi, T.S.B. Sudarshan, and Gopal K. "Patra Neural Synchronization by Mutual Learning Using Genetic Approach for Secure Key Generation". S.M. Thampi. SNDS 2012, CCIS 335, pp. 422-431, Springer-Verlag, 2012.

[10] Alexander Klimov, Anton Mityagin, and Adi Shamir. "Analysis of Neural Cryptography". Y. Zheng (Ed.): ASIACRYPT 2002, LNCS 2501, pp. 288-298, Springer-Verlag, 2002.

[11] Kanter, W. Kinzel "The Theory of Neural Networks and Cryptography". Quantum Computers and Computing. Vol. 5 Nº 1. 2005.

[12] M. Rosen-Zvi, I. Kanter, W Kinzel. "Cryptography based on neural networks-analytical results". J. Phys. A: Math. Gen., vol. 35, pp. L707-L713, 2002.