UNIVERSIDAD CATÓLICA DEL NORTE

FACULTAD DE INGENIERÍA Y CIENCIAS GEOLÓGICAS

Departamento de Ingeniería de Sistemas y Computación

ANÁLISIS DE LA SINCRONIZACIÓN NEURONAL ENTRE REDES TREE PARITY MACHINE MULTICAPAS PARA SU USO EN ESQUEMAS DE INTERCAMBIO DE LLAVES

Anteproyecto de Tesis.

RODRIGO ALONSO JIMÉNEZ GÓMEZ

Profesor tutor: Dr. Iván Jirón Araya

Profesor co-tutor: Dr. Rafael Martínez Peláez

Antofagasta, Chile.
Agosto, 2025

Capítulo 1

Introducción

Agregar introducción.

Capítulo 2

Marco Teórico

2.1. Ciberseguridad

2.1.1. Confidencialidad, Integridad y Disponibilidad

La protección de la información privada es un desafío que antecede a la era digital, pero ha alcanzado nuevas dimensiones y complejidades gracias a diversos avances tecnológicos y la masiva interconexión global de sistemas informáticos [1, 2, 3]. La facilidad del acceso a la información a través del ciberespacio ha exigido la creación de la disciplina de la ciberseguridad. La ciberseguridad es el conjunto de prácticas y herramientas que permiten resguardar la seguridad de un sistema informático y su objetivo es garantizar la confidencialidad, integridad y disponibilidad de la información [4], que se describen a continuación:

- Confidencialidad se refiere a los esfuerzos realizados para asegurar que los datos permanezcan secretos o privados de manera que la información solamente pueda ser accedida por usuarios autorizados.
- Integridad implica asegurar que los datos sean auténticos, precisos y confiables, y que no hayan sido alterados de manera indebida.
- Disponibilidad significa que las personas autorizadas deben poder acceder a la información cuando la necesiten, de manera oportuna y sin retrasos excesivos.

2.1.2. Protocolos para una Comunicación Segura

Un protocolo de internet contiene los lineamientos que establecen el formato y la secuencia de acciones necesarias para establecer una conexión exitosa entre dos o más dispositivos. Estos lineamientos garantizan la coherencia de la comunicación digital y han dado forma a la transmisión a través del internet [5]. Existen diversos protocolos diseñados para distintos tipos de aplicaciones, que incluyen la transferencia de archivos, túneles seguros para la administración remota y protocolos de autenticación. Para lograr una comunicación segura, diversos protocolos han implementado algoritmos de encriptación compuestos, es decir, protocolos de encriptación que utilizan tanto criptografía simétrica como asimétrica. Por ejemplo, el protocolo de acceso remoto Secure Shell (SSH) utiliza DHKE para el intercambio de llaves y Advanced Encryption Standard (AES) para el cifrado de la sesión remota [6].

2.1.3. Control de Acceso y Autenticación

Los sistemas informáticos mantienen sistemas de control de acceso que asocian identidades con niveles de autorización que regulan el privilegio de revisión y modificación del contenido de los repositorios presentes en el sistema. Sin embargo, también necesitan comprobar la legitimidad de la identidad que presenta un usuario que intenta acceder a la información. Un proceso de autenticación verifica la identidad que un usuario utiliza en el proceso de identificación. La deficiente o nula implementación de sistemas de control de acceso ha sido identificado como uno de los problemas cruciales a la hora de analizar la ciberseguridad de dispositivos en la Internet de las Cosas (Internet of Things, IoT), en donde muchos dispositivos son desplegados con protocolos de autenticación vulnerables, débiles o inexistentes [3].

2.2. Criptografía

La criptografía es la ciencia que busca ocultar el significado de los mensajes, de manera que solo pueda ser entendido por el destinatario. Su contraparte es el criptoanálisis, la ciencia que intenta romper un criptosistema y descubrir el significado de los mensajes encriptados, lo que resulta crucial para determinar la seguridad de la comunicación. Ambos conforman a la criptología, el estudio de los mensajes secretos. En esta sección, se resumen los conceptos básicos de la criptografía presentados en el libro [6].

2.2.1. Aritmética Modular

La aritmética modular es un sistema de operaciones matemáticas en el que los números se reducen al resto de su división por un número entero, denominado módulo. Este enfoque permite trabajar con una colección de números finita y es fundamental en el área de la criptografía. La aritmética modular ha sido utilizada desde la antigüedad para la creación de sistemas de encriptación, como por ejemplo el cifrado César, el cual se presume haber sido el método utilizado por Julio César para comunicarse con su ejército. Hoy en día es utilizado por diversos algoritmos criptográficos, como Rivest-Shamir-Adleman (RSA), Diffie-Hellman Key Exchange (DHKE) y Elliptic Curve Cryptography (ECC).

Operación Modulo

La operación módulo permite expresar formalmente la relación entre un número entero y su residuo al ser dividido por un módulo positivo. Para $a, r, m \in \mathbb{Z}$ (en donde $\mathbb{Z}$ es el conjunto de todos los números enteros) y $m > 0$, la operación modulo se define como:

$$a \equiv r \bmod m$$

si m divide $a - r$, m es llamado el módulo y r es llamado el resto o residuo.

Clases de Equivalencia

La operación módulo, al aplicarse con un módulo $m > 0$, genera un número finito de posibles resultados, específicamente los enteros entre 0 y $m - 1$. Esta propiedad permite representar cualquier número entero como un elemento dentro de este conjunto finito, mediante la asignación de su residuo módulo m. De esta manera, los enteros pueden agruparse en clases de equivalencia: conjuntos que comparten el mismo residuo al ser divididos por m. Esto significa que para cada módulo m y un número a, exite un número infinto de residuos válidos. Por ejemplo:

- $12 \bmod 9 = 3$
- $21 \bmod 9 = 3$
- $-6 \bmod 9 = 3$

De esta manera, se forma la clase de equivalencia:

$$\{\dots, -6, 3, 12, 21, \dots\}$$

Además, existen otras ocho clases de equivalencia:

$$\{\dots, -9, 0, 9, 18, \dots\}$$
$$\{\dots, -10, 1, 10, 19, \dots\}$$
$$\vdots$$
$$\{\dots, -1, 8, 17, 26, \dots\}$$

Esta relación de equivalencia puede expresarse mediante la notación

$$12 \equiv 21 \bmod 9$$

utilizando los dos primeros ejemplos mencionados anteriormente.

Anillos de Enteros

El conjunto de clases de equivalencia módulo m permite definir operaciones de suma y multiplicación entre sus elementos. Estas operaciones se realizan tomando representantes de cada clase, operando normalmente y luego aplicando el módulo m al resultado. Con estas reglas, el conjunto forma lo que se conoce como un anillo de enteros módulo m, representado por $\mathbb{Z}_m$, una estructura cerrada bajo la suma y la multiplicación, que cumple propiedades similares a las del conjunto de los enteros, pero dentro de un conjunto finito. Para el módulo m, el anillo de enteros $\mathbb{Z}_m$ consiste de:

- La colección $\mathbb{Z}_m = \{0, 1, 2, \dots, m-1\}$
- Las operaciones de suma y multiplicación para cualquier $a, b \in \mathbb{Z}_m$, de manera que:
  - $a + b \equiv c \bmod m$, $c \in \mathbb{Z}_m$
  - $a \times b \equiv d \bmod m$, $c \in \mathbb{Z}_m$

Por ejemplo, utilizando el anillo de enteros $\mathbb{Z}_9$:

- $6 + 8 = 14 \equiv 5 \bmod 9$
- $6 \times 8 = 48 \equiv 3 \bmod 9$

El método de encriptación César, o por desplazamiento, utiliza la operación de suma en un anillo de enteros de tamaño igual al número de caracteres utilizados por el idioma del mensaje, de modo que cada letra se transforma sumando un valor fijo y luego reduciendo el resultado módulo el tamaño del alfabeto.

2.2.2. Criptografía Simétrica

La criptografía simétrica consiste en dos participantes que intercambian una misma llave para la encriptación y desencriptación de un sistema. Esto implica que:

- La llave es secreta, los participantes deben asegurarse de un atacante no adquiera esta llave, ya que le permitiría revelar el contenido del mensaje.
- Los participantes deben encontrar una forma de acordar una clave secreta antes de lograr una comunicación segura. Una vez establecida una clave, esta puede ser utilizada para mantener una comunicación segura.

La criptografía simétrica constituye el esquema más antiguo para el resguardo de la confidencialidad de la información, así, todos los sistemas criptográficos utilizados desde la antigüedad se basaban exclusivamente en este enfoque. No obstante, la distribución segura de llaves representa un desafío considerable, especialmente al intentar implementar un criptosistema simétrico a través de Internet. Ejemplos de criptografía simétrica incluyen:

- Método de encriptación por Sustitución
- Método de encriptación César o por desplazamiento
- Método de encriptación Vigenère
- Método de encriptación Data Encryption Standard (DES), su variación 3DES y su sucesor AES

La criptografía simétrica puede dividirse en dos categorías principales: Encriptado por flujo y Encriptación por bloques.

Encriptado por Flujo

El encriptado por flujo encripta cada bit de manera individual. Para lograr esto, se agrega un bit de un *flujo de llaves* a un bit de texto plano. La encriptación por flujo puede ser síncrono, es decir, el flujo de llaves depende solamente de la clave secreta, como también puede ser asíncrono, en donde el flujo de llaves depende tanto de la clave como del texto encriptado. La seguridad de la encriptación por flujo depende completamente del flujo de llaves, lo que lo convierte en el problema central a la hora de diseñar un criptosistema de encriptado por flujo [6].

Los encriptacións de flujo son empleados principalmente en sistemas embebidos y equipos con capacidades computacionales limitadas. Un ejemplo de esto es el encriptación A5/1 utilizado en las redes de comunicación celular Global System for Mobile Communications (GSM) [6].

Encriptación por Bloques

La encriptación por bloques encripta un bloque completo de bits de texto plano con la misma llave. Esto implica que la encriptación de cualquier bit de texto plano en un bloque determinado depende de cada bit presente en el mismo bloque. El tamaño del bloque impacta proporcionalmente en la seguridad del criptosistema, es por esto que estándares modernos requieren del uso de una clave secreta al menos 128 bits de longitud, como es el caso de AES. Las encriptaciones por bloques son los esquemas principalmente utilizados en la comunicación a través de Internet [6].

Algoritmos DES y AES

El algoritmo Data Encryption Standard (DES) fué un algoritmo popular desarrollado por IBM con ayuda de la Agencia de Seguridad Nacional de los Estados Unidos (National Security Agency, NSA) y publicado por el Instituo Nacional de Estándares y Tecnología (NIST por sus siglas en inglés, en ese tiempo llamada NBS) en 1977. El algoritmo hoy es considerado inseguro debido a su reducida longitud de llaves, dando paso al uso de su variacion 3DES y el desarrollo de su sucesor AES.
El algoritmo DES es un encriptación por bloques de 64 bits de longitud, con una clave de 56 bits. Por cada bloque, el encriptación se divide en 16 rondas que ejecutan la misma operación. Utiliza dos principios fundamentales para la construcion del algorimtos de encriptación seguros:

- Confusión: Una operación de encriptación en donde la relación entre la clave secreta y el texto encriptación no es directa.
- Difusión: Una operación de encriptación en donde la influencia de un simbolo en texto plano es distribuida sobre muchos símbolos de texto encriptación, con el objetivo de ocultar las propiedades estadísticas del texto plano.

A pesar de haber sido el estándar recomendado por el gobierno de los Estados Unidos, el algoritmo DES fue sujeto a diversas críticas debido a el uso de una clave de 56 bits, volviendo el encriptación vulnerable a ataques de fuerza bruta. Sumado a esto, los parámetros de diseño de las cajas S —las funciones no-lineales son cruciales

para la seguridad del algoritmo— se mantuvieron en secreto por años. Esto creó la especulación de un posible ataque analítico solo conocido por los diseñadores del algoritmo, capaz de explotar las propiedades matemáticas de las cajas S.
Eventualmente, en el año 1998 la *Electronic Frontier Foundation* (EFF) creó una máquina capaz de ejecutar un ataque de fuerza bruta basado en hardware sobre el algoritmo DES. La máquina fué apodada *DeepCrack*, fué la prueba de las vulnerabilidades del algoritmo que obligaron a elevar el estándar de seguridad en el año 1999, cambiando el algoritmo DES por la variante 3DES.
Sin embargo, la variante 3DES, como su nombre indica, es la repetición del proceso DES en tres iteraciones. Esto incrementa la seguridad del encriptación a 112-bits, pero el proceso recibe un impacto en su rendimiento, especialmente en implementaciones de software. Esto, combinado con su reducido tamaño de bloques, motivó al Instituo Nacional de Estándares y Tecnología de los Estados unidos (NIST) a buscar un nuevo algoritmo capaz de reemplazar a DES.
En el año 2001, el NIST publicó el estándar AES, que reemplazaría a DES y 3DES. Es un algoritmo de encriptación por bloques que admite llaves de longitud de 128, 192 y 256 bits, utiliza un sistema de rondas tal como lo hace DES, sin embargo, debido a las diferencias en la estructura interna de cada ronda, AES utiliza comparativamente menos rondas para lograr un mejor estándar de seguridad [6]. Cada ronda del algoritmo AES utiliza capas de operaciones, cada una opera sobre todos los bits presentes en la ruta de datos del algoritmo, o también referido como el estado del algoritmo. Existen tres tipos de capas, las cuales están presentes en todas las rondas a excepción de la primera, tal como muestra la figura 2.1.

1. Capa de adición de llaves: una clave de 128 bits es derivada de la clave original, y es sumada (mediante la operación XOR) al estado del algoritmo.

2. Capa de sustitución de bits (caja S): Cada elemento del estado es transformado no-linealmente utilizando tablas de búsqueda con propiedades matemáticas especiales. Esto introduce *confusión* en la ruta de datos del algoritmo.

3. Capa de difusión: Otorga difusión sobre todos los bits del estado del algoritmo. Consiste en dos subcapas que ejecutan operaciones lineales, una permuta los datos a nivel de bytes (Subcapa *ShiftRow*) y la segunda mezcla bloques de cuatro bytes mediante una operación matricial (Subcapa *MixColumn*).

En el año 2016, una vulnerabilidad de alto riesgo indexada con el código *CVE-2016-2183* demostró que tanto el algoritmo DES y su variante 3DES son vulnerables a ataques que permiten la obtención del texto plano [**?**], lo que eventualmente provo-

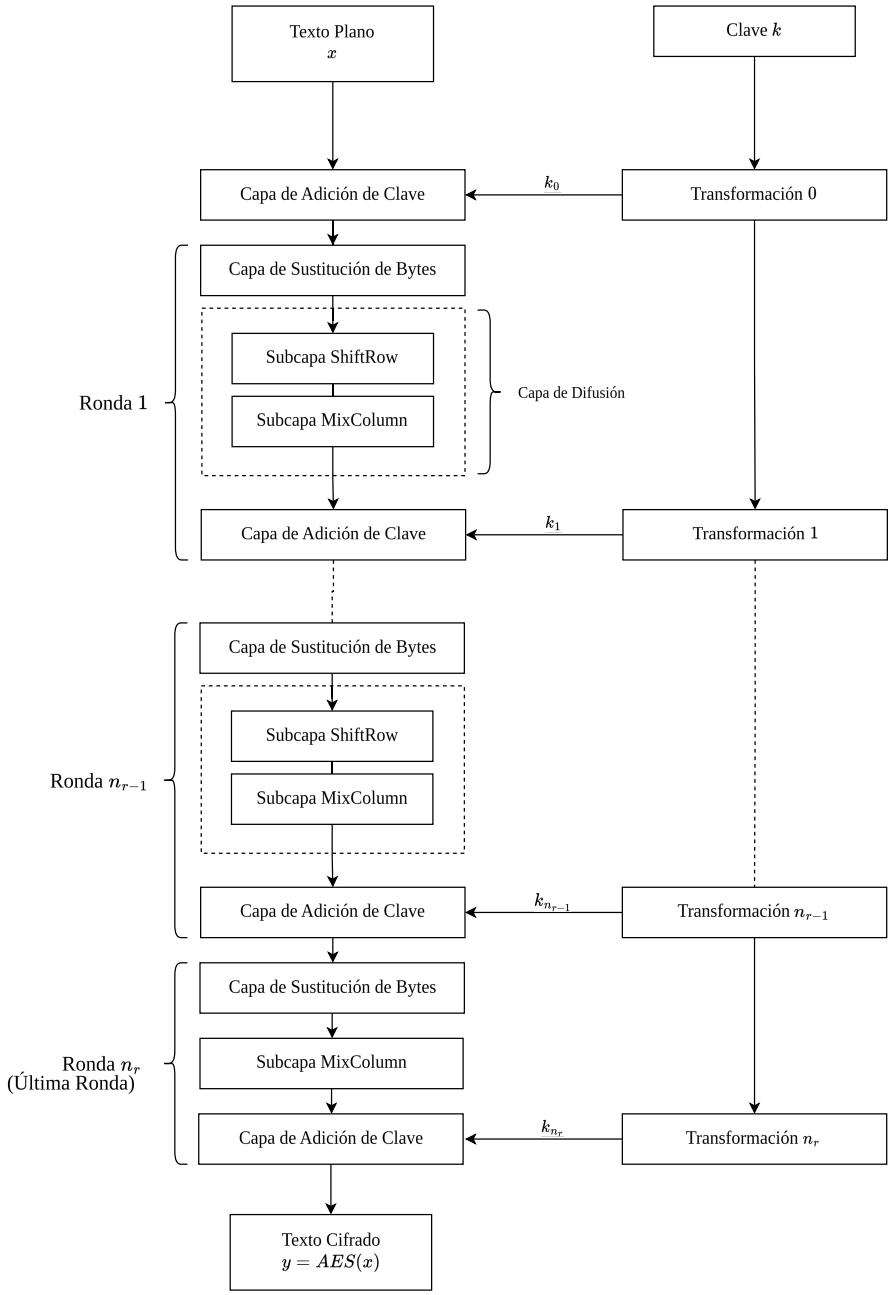

Figura 2.1: Proceso de encriptación mediante AES. Figura original de [6].

caría que ambos algoritmos fuesen declarados como obsoletos para uso moderno por el NIST en el año 2019, favoreciendo el uso de AES para aplicaciones modernas [7].

2.2.3. Criptografía Asimétrica

El algoritmo Diffie-Hellman, propuesto en 1976, fue la primera solución propuesta a los problemas presentes en la criptografía simétrica y dió paso a un nuevo esquema, hoy conocido como criptografía asimétrica. En este esquema, no es necesario que la clave que posee la persona que encripta el mensaje sea secreta. Lo crucial es que el receptor, solo pueda desencriptar utilizando una clave secreta. Para implementar un sistema de este tipo, el receptor publica una clave de encriptación pública que es conocida por todos. El receptor también posee una clave secreta correspondiente, la cual es utlilizada para el desencriptación. Por ende, el receptor tiene una clave k que consiste de dos partes, una clave pública k_{pub} y una clave privada k_{pr}. Los algoritmos de encriptación asimétrica o algoritmos clave pública pueden ser utilizados para:

- Establecimiento de llaves a través de un canal inseguro.
- Proveer No-Repudiación e integridad del mensaje transmitido usando firmas digitales.
- Identificación de los participantes a través de firmas digitales en conjunto con protocolos de desafío-respuesta.
- Encriptación de mensajes.

A pesar de sus ventajas de seguridad, la criptografía asimétrica requiere de uso intenso de los recursos computacionales, provocando que en la práctica el uso de la criptografía asimétrica para encriptación de datos sea mucho menos conveniente que el uso de esquemas simétricos. Sin embargo, al proveer mecanismos de seguridad como la No-Repudiación y la capacidad de establecer llaves en canales inseguros, se suelen implementar ambos esquemas a través de protocolos híbridos.
Estos algoritmos pueden clasificarse según el problema computacional subyacente en el que se basa su seguridad. Es decir, cada familia de algoritmos criptográficos asimétricos depende de un tipo específico de problema matemático difícil de resolver. Solo existen tres familias de algoritmos relevantes para su uso práctico:

- Factorización de Números Enteros de gran tamaño
- Logaritmo Discreto en Campos Finitos y
- Logaritmo Discreto sobre Curvas Elípticas

Algoritmo Diffie-Hellman

El algoritmo Diffie-Hellman (DF o DFKE) fué el primer algoritmo de criptografía asimétrica, publicado en 1976 por Whitfield Diffie y Martin Hellman. Este algoritmo se basa en la intractabilidad computacional del problema del logaritmo discreto, cuya resolución resulta inviable para valores suficientemente grandes. En términos matemáticos, dado $g \in \mathbb{Z}_p^*$ y p un número primo, calcular $g^x \bmod p$ es una operación trivial, pero debido a que se trata de una función de una sola vía, resulta computacionalmente difícil invertirla para determinar x a partir de $g^x \bmod p$.
El algoritmo de intercambio de llaves Diffie-Hellman utiliza un proceso de dos partes, siendo la primera un proceso de configuración en donde se definen los parámetros que serán intercambiados mediante un canal inseguro, para su posterior uso en la segunda parte del proceso, en donde se establece el intercambio de llaves. El proceso de configuración es el siguiente:

1. Escoger un número primo p de gran tamaño. El tamaño de p define el tamaño de la clave pública e impacta directamente en la seguridad del intercambio. El Instituto Nacional de Estándares y Tecnología de los Estados Unidos (NIST) recomienda una clave pública de al menos 2048 bits de longitud.

2. Escoger un número entero $a \in \{2, 3, \ldots, p-2\}$.

3. Publicar p y a.

La segunda parte del proceso de intercambio utiliza la propiedad de conmutatividad de la exponenciación:

$$k = (a^x)^y \equiv (a^y)^x \bmod p$$

Es decir, utilizando los parámetros publicados en la primera fase del intercambio, Alice calcula

$$B^a \equiv (a^b)^a \equiv a^{ab} \bmod p$$

y Bob calcula

$$A^b \equiv (a^a)^b \equiv a^{ab} \bmod p$$

La figura 2.2 es un ejemplo que muestra la segunda parte del intercambio llaves.

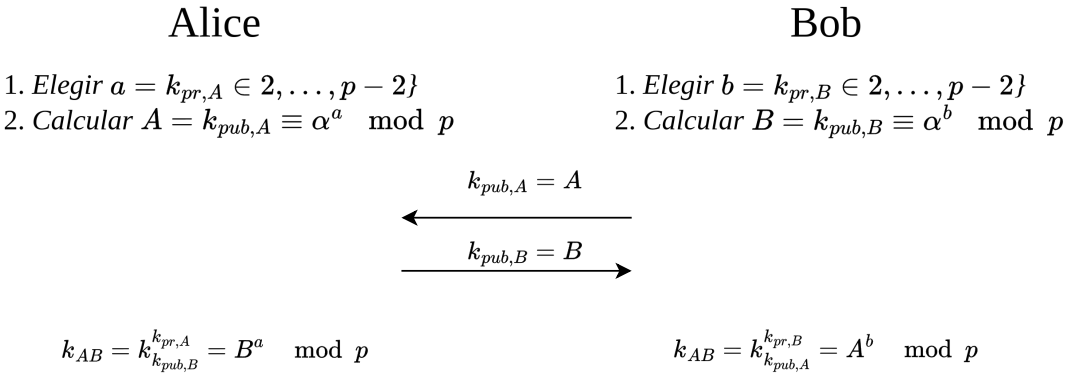

Figura 2.2: Ejemplo de intercambio de llaves DHKE entre Alice y Bob. Ejemplo original de [6].

Algoritmo RSA

El algoritmo RSA (Por sus autores Ron Rivest, Adi Shamir y Leonard Adleman) es un algoritmo basado en la factorización de números enteros de gran tamaño en sus factores primos. En esencia, RSA se basa en la multiplicación de dos números primos grandes, una operación computacionalmente sencilla; sin embargo, factorizar el producto resultante es un problema considerablemente más complejo.
La función de encriptación del algoritmo RSA, dada una clave pública $(n, e) = k_{pub}$ y un texto plano x, puede expresarse de la siguiente forma:

$$y = e_{k_{pub}}(x) \equiv x^e \bmod n$$

donde $x, y \in \mathbb{Z}_n$.
La función de desencriptación del algoritmo RSA, dada una clave privada $d = k_{pr}$ y el texto encriptación y, puede expresarse de la siguiente forma:

$$x = d_{k_{pr}}(y) \equiv y^d \bmod n$$

donde $x, y \in \mathbb{Z}_n$.
Para generar la clave pública $k_{pub} = (n, e)$ y privada $k_{pr} = (d)$, se utiliza el siguiente algoritmo:

1. Elegir dos números primos de gran tamaño, p y q.
2. Calcular $n = p \cdot q$.

3. Calcular $\phi(n) = (p-1)(q-1)$.

4. Seleccionar el exponente público $e \in \{1, 2, \ldots, \phi(n) - 1\}$ de manera que

$$gcd(e, \phi(n)) = 1$$

5. Calcular la clave privada d, de manera que

$$d \cdot e \equiv \mod \phi(n)$$

La figura 2.3 es un ejemplo de comunicación segura utilizando el algoritmo RSA para el encriptación y desencriptación del mensaje. Nótese que en aplicaciones reales los parámetros suelen ser números de gran tamaño, por ejemplo, para los parámetros p y q suelen usarse una longitud de 512 bits, de manera que el módulo n sea de al menos 1024 bits.

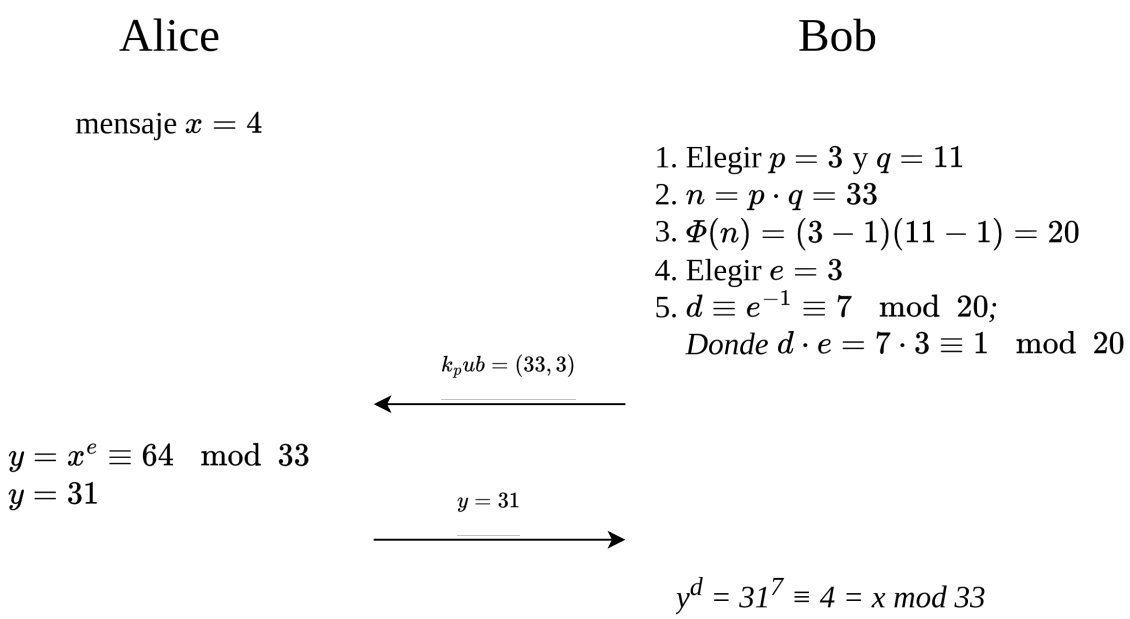

Figura 2.3: Ejemplo de intercambio de mensaje encriptación mediante RSA entre Alice (emisor) y Bob (receptor). Ejemplo original de [6].

El algoritmo RSA es ampliamente utilizado en firmas digitales y encriptación de datos de tamaño pequeño, pero no está diseñado para reemplazar algoritmos de encripción simétrica, debido a su alto costo computacional.

2.2.4. Criptografía Neuronal

A inicios del nuevo milenio una serie de publicaciones sobre las dinámicas de la sincronización neuronal y sus posibles aplicaciones para el problema de intercambio de llaves han dado lugar a la creación de la criptografía neuronal [8]. La criptografía neuronal utiliza propiedades observadas en la sincronización de redes neuronales del tipo Tree Parity Machine mediante el aprendizaje mutuo. El aprendizaje mutuo permite la sincronización de los pesos sinápticos en las redes participantes sin la necesidad de compartirlos de manera explícita por el canal, permitiendo utilizar los pesos sinápticos como la llave secreta.
A diferencia de otros esquemas de intercambio de llaves, las redes neuronales utilizan operaciones matemáticas simples y de bajo costo computacional, permitiendo su uso en equipos de baja potencia y en donde los estándares de seguridad requieren de una rotación de llaves continua.

2.3. Redes Neuronales

Esta subsección resume los áspectos básicos sobre Redes Neuronales Artificiales (RNA) presentados por [9], y los principios de la sincronización neuronal por [8].

2.3.1. Definición y Principios Básicos de las Redes Neuronales Artificiales

Las RNA son modelos matemáticos que se inspiran en la estructura del cerebro humano, reconociendo a éste como un computador de naturaleza diferente a la de un computador digital convencional. El cerebro es una estructura altamente compleja, no lineal y paralela, constituida de múltiples unidades de procesamiento simples, llamadas neuronas. En [9] se definen las redes neuronales de la siguiente manera:
"Una red neuronal es un procesador distribuido de manera altamente paralela, compuesto por unidades de procesamiento simples, que posee una inclinación natural hacia el almacenamiento del conocimiento experiencial y su posterior utilización. Se asemeja al cerebro principalmente de dos maneras:

1. El conocimiento es adquirido por la red a partir de su entorno mediante un proceso de aprendizaje.

2. La fuerza de las conexiones entre neuronas, conocidas como pesos sinápticos, se utiliza para almacenar el conocimiento adquirido."

Las RNA son diseñadas para modelar la manera en que el cerebro ejecuta una tarea específica de interés. Generalmente, se implementa mediante componentes electrónicos, simulaciones a través de software, o una mezcla de software con aceleración por hardware especializado.
A través de un algoritmo de aprendizaje, se modifican los pesos sinápticos de la red para alcanzar un comportamiento deseado. Al modificar los pesos sinápticos, efectivamente se modifica el efecto que ejerce un estímulo de entrada sobre una neurona. Esto permite aumentar, disminuir e incluso anular o revertir la respuesta de una neurona hacia un estímulo determinado.
Las RNA ofrecen una serie de ventajas y capacidades frente a otros sistemas de control inteligente, las cuales han sido ampliamente reconocidas en la literatura. En el contexto de este trabajo, es pertinente destacar las siguientes propiedades clave:

1. No linealidad: Cada neurona puede ser lineal o no-lineal. La no-linealidad distribuida presente en su estructura le permite a una RNA modelar fenómenos de naturaleza no-lineal. Esta propiedad resulta fundamental para la seguridad de una red Tree Parity Machine debido a que actúa como una barrera de complejidad matemática, impidiendo que la relación entre las entradas y las salidas sea directamente invertible.

2. Mapeo de Entrada-Salida: Las RNA adquieren conociemiento generando una relación entre los estímulos de entrada y la señal de salida de la red. Esta propiedad resulta fundamental al momento de lograr una sincronización de dos redes Tree Parity Machine, debido a que para una misma señal de entrada dos redes idénticas deberían producir exactamente la misma salida. Debido a esto, cuando dos RNA no han sido sincronizadas, si para una misma señal tienen el mismo comportamiento (la misma señal de salida) pueden modificar sus pesos sinápticos utilizando una regla de aprendizaje especial y repetir el proceso hasta lograr la sincronización.

2.3.2. Modelo de Neurona Artificial y Funciones de Activación

La neurona es la unidad de procesamiento fundamental de las RNA. En la figura 2.4. se pueden identificar los tres elementos básicos del modelo neuronal:

1. Una colección de sinápsis o enlaces, cada uno caracterizado por un *peso* o *fuerza* propia. Cada señal x_j para la entrada j de la neurona k, es multiplicado por el peso sináptico correspondiente $w_{k,j}$.

2. Un *combinador lineal* que suma las señales de entrada multiplicados por su correspondiente peso sináptico.

3. Una función de activación, que limita la amplitud de la salida de una neurona.

Como se aprecia en la figura 2.4, el modelo contiene un término llamado *bias*, denotado por b_k. El bias b_k tiene el efecto de aumentar o disminuir la entrada total de la función de activación.

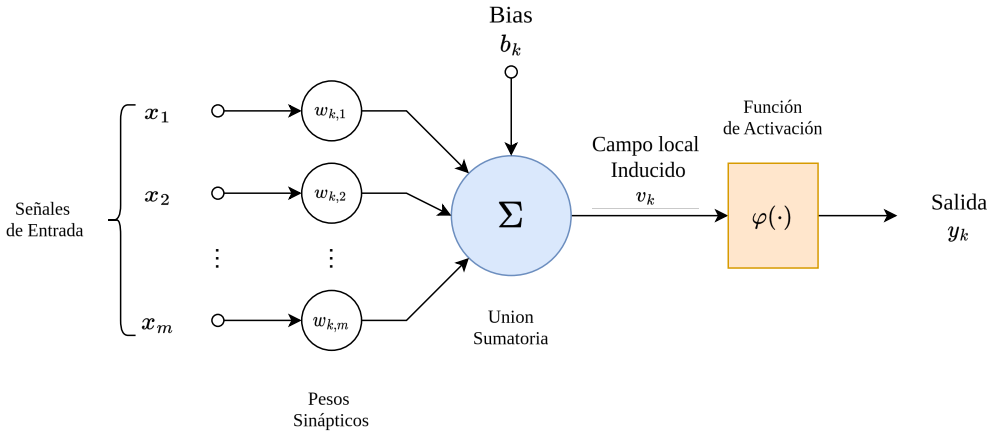

Figura 2.4: Modelo de neurona. Figura original adaptada de [9].

En términos matemáticos, la neurona k presentada en la figura 2.4 puede ser descrita de la siguiente forma:

$$u_k = \sum_{j=1}^{m} w_{k,j} x_j \quad v_k = u_k + b_k \quad y_k = \varphi(v_k)$$

en donde $x_1, x_2, ..., x_m$ son las señales de entrada; $w_{k,1}, w_{k,2}, ..., w_{k,m}$ son los pesos sinápticos de la neurona k; u_k (que no se muestra en la figura 2.4) es la salida del combinador lineal; b_k es el bias; $\varphi(\cdot)$ es la función de activación; e y_k es la señal de salida de la neurona.
La función de activación de una neurona define su reacción frente a un estímulo de entrada, esta función puede ser de carácter lineal o no-lineal. La elección de la función de activación depende del problema que la red intenta resolver. Algunas de las funciones de activación más utilizadas incluyen

- Función Límite o Heaviside:

$$\varphi(v) = \begin{cases} 1 & \text{si} \quad v \geq 0 \\ 0 & \text{si} \quad v < 0 \end{cases}$$

- Función Sigmoidal:

$$\varphi(v) = \frac{1}{1 + exp(-av)}$$

donde a es el parámetro de inclinación

2.3.3. Procesos de Aprendizaje y Sincronización en Redes Neuronales

Los procesos de aprendizaje que utilizan las redes neuronales para ajustar su comportamiento se pueden categorizar en dos: **aprendizaje con un profesor** y **aprendizaje sin un profesor**.
El aprendizaje con un profesor, también conocido como aprendizaje supervisado, emplea datos del entorno en el que opera la red neuronal —generalmente en forma de un conjunto de entrenamiento— para guiar su proceso de aprendizaje. En este esquema, se genera una señal de error que representa la diferencia entre la salida producida por la red y la salida esperada proporcionada por el profesor, correspondiente a una entrada del conjunto de entrenamiento.
El aprendizaje sin un profesor se divide en dos subcategorías: **aprendizaje por refuerzo** y **aprendizaje no supervisado**.
El aprendizaje por refuerzo se basa en la presencia de un módulo evaluador que transforma una señal de refuerzo primaria, proveniente del entorno, en una señal de refuerzo heurística. Esta señal guía el ajuste de la red neuronal, incentivando el proceso de aprendizaje. De esta forma, el sistema puede interactuar con su entorno y desarrollar la capacidad de aprender a partir de las consecuencias de sus acciones.
El aprendizaje no supervisado, también conocido como aprendizaje autoorganizado, no utiliza un crítico ni un profesor para guiar el proceso de aprendizaje. En su lugar, se emplea una medida independiente de la tarea que evalúa la calidad de la representación que la red debe aprender. A partir de esta medida, los parámetros libres de la red, tales como los pesos y bias, se ajustan para capturar las regularidades estadísticas presentes en los datos de entrada, mediante una regla de aprendizaje. Como resultado, la red desarrolla la capacidad de generar representaciones internas que codifican características relevantes del entorno, permitiéndole formar automáticamente nuevas clases o categorías.
La *Teoría de Aprendizaje de Hebb* es la regla de aprendizaje más antigua y conocida de todas. Fue propuesto en un contexto neurobiológico, que más tarde sería reformulado como una regla de dos partes:

1. Si dos neuronas en ambos lados de una conexión son activadas de manera simultánea, entonces la fuerza de esa conexión es selectivamente aumentada.

2. Si dos neuronas en ambos lados de una conexión son activadas de manera asíncrona, entonces la fuerza de esa conexión es selectivamente disminuida (o eliminada).

En términos matemáticos se puede formular la regla de aprendizaje Hebbian considerando los pesos sinápticos $w_{k,j}$ de la neurona k con señales presinápticas y postsinápticas x_j y y_k, respectivamente. De esta manera, el ajuste aplicado al peso sináptico $w_{k,j}$ en el momento n puede ser expresado de la siguiente forma general:

$$\Delta w_{k,j}(n) = f(y_k(n), x_j(n))$$

donde $f(,)$ es una función que depende de las señales presinápticas y postsinápticas. Esta ecuación admite muchas variaciones, las cuales todas califican como regla de aprendizaje Hebbian. Se podría considerar la forma más simple de regla de aprendizaje Hebbian como:

$$\Delta w_{k,j}(n) = \eta y_k(n) x_j(n))$$

en donde η es una constante positiva que determina la *tasa de aprendizaje*.

2.3.4. Redes Tree Parity Machine

Las redes Tree Parity Machine (TPM) son una arquitectura de redes neuronales artificiales diseñadas para explorar procesos de aprendizaje mutuo mediante un proceso de sincronización que permite igualar el estado interno (los pesos sinápticos) entre dos redes a través de un proceso iterativo.
Las redes Tree Parity Machines utilizan la función signo como función de activación, definida de la siguiente forma:

$$\varphi(v) = \begin{cases} 1 & \text{si} \quad v > 0 \\ -1 & \text{si} \quad v \leq 0 \end{cases}$$

Al ser una función no-lineal, esto permite que la relación entre los estímulos de entrada y la señal de salida de la red no sea directamente invertible, generando una barrera de seguridad necesaria para el proceso de sincronización.
En las redes Tree Parity Machine se utiliza un aprendizaje autoorganizado utilizando una regla Hebbian diseñada para tomar en cuenta la influencia de ambas redes presentes en el proceso de sincronización, la cual [8] escribe con una notación ligeramente diferente a la presentada anteriormente, de la siguiente forma:

$$w_{i,j}^{+} = g(w_{i,j} + x_{i,j} \cdot \tau \cdot \Theta(\sigma_i \cdot \tau) \cdot \Theta(\tau^A \cdot \tau^B))$$

En donde $w_{i,j}$ corresponde al peso sináptico de la neurona i en la entrada j, $x_{i,j}$ es el estímulo de entrada j para la neurona i, τ corresponde a la salida de la red en ajuste, τ^A es la salida de la red A y τ^B es la salida de la red B. σ_i es la salida de la neurona i. La ecuación anterior también emplea dos funciones:

1. $g(\cdot)$ - Función de limitación de pesos sinápticos:

$$g(x) = \begin{cases} -L & x \leq -L \\ x & -L < x < L \\ L & x \geq L \end{cases}$$

2. $\Theta(\cdot)$ - Función Heaviside:

$$\Theta(x) \begin{cases} 0, & \text{si} \quad x \leq 0 \\ 1, & \text{si} \quad x > 0 \end{cases}$$

La función $g(\cdot)$ se encarga de limitar el valor de los pesos sinápticos en el rango $[-L, L]$ para evitar un crecimiento explosivo. La función Heaviside se encarga de determinar el ajuste en base a la relación que existe entre el estímulo de entrada de una neurona y el valor producido por ambas redes. El término $\Theta(\tau^A \cdot \tau^B)$ provoca que sólo se realicen ajustes si ambas redes en el proceso de sincronización tienen la misma salida, es decir, reaccionan de la misma manera a un mismo estímulo de entrada.

Capítulo 3

Planteamiento del Problema

3.1. Descripción de la Problemática

Actualmente la generación de claves es realizada mediante algoritmos que utilizan aritmética modular y factorización de números primos de gran tamaño, lo que puede significar un uso elevado de recursos computacionales. Para la regeneración de claves, los participantes deben realizar el proceso de intercambio completo, creando una alta carga computacional de la misma manera que el proceso de intercambio de claves inicial. En un servidor web, las operaciones utilizadas por algoritmos como RSA en TLS corresponden entre un 20-58 % del tiempo de ejecución [10].
El uso de Tree Parity Machines (TPM) en la criptografía neuronal emerge como una alternativa viable, debido a que utiliza operaciones matemáticas simples y no requiere del uso de grandes números enteros [11].
Este trabajo propone un estudio sistemático y experimental sobre variaciones multicapa de las arquitecturas TPM, denominadas Multilayer Tree Parity Machuine (MTPM), las cuales permiten explorar nuevas configuraciones topológicas que podrían ofrecer mejoras en eficiencia en el proceso de sincronización. Para ello, se ha desarrollado un motor de simulación que automatiza el proceso de sincronización entre redes MTPM, permitiendo el análisis comparativo de múltiples arquitecturas y la influencia de los parámetros H, K, N, L y M. Además, se ha construido una interfaz gráfica que facilita la visualización de los resultados a través de gráficos interactivos y representaciones tridimensionales.

3.2. Originalidad de la Investigación

La originalidad de esta investigación radica en la combinación de estímulos no binarios y la presencia de múltiples capas ocultas en la arquitectura de una red Tree Parity Machine.
Finalmente, basado en la revisión bibliográfica, se elegirá el ataque que ha probado ser efectivo contra el proceso de sincronización neuronal, y a continuación se aplicará sobre el protocolo propuesto en esta investigación.

3.3. Pregunta de Investigación

¿Cuáles son los efectos del traslape en los estímulos de entrada de cada neurona en una capa oculta, en el proceso de sincronización neuronal de dos redes Multilayer Tree Parity Machine?

3.4. Hipótesis

El uso estímulos sin traslape en la entrada en cada neurona de una capa oculta, permite que en el proceso de sincronización neuronal entre dos redes Multilayer Tree Parity Machine termine correctamente.

Capítulo 4

Descripción de la Investigación

4.1. Objetivos

4.1.1. Objetivo General

Analizar el uso estímulos sin traslape en la entrada de cada neurona de una capa oculta, en el término del proceso de sincronización neuronal entre dos redes Multilayer Tree Parity Machine.

4.1.2. Objetivo Específico

1. Analizar la literatura relacionada con la sincronización de dos redes Multilayer Tree Parity Machine.

2. Diseñar un algoritmo para la sincronización de dos redes Multilayer Tree Parity Machine con estímulos binarios y no binarios.

3. Implementar algoritmo para la sincronización de dos redes Multilayer Tree Parity Machine con estímulos binarios y no binarios.

4. Evaluar experimentalmente los efectos del traslape en los estímulos de entrada en cada neurona de una capa oculta, en el proceso de sincronización neuronal de dos redes Multilayer Tree Parity Machine.

4.1.3. Alcances y Limitaciones

El proyecto contempla los siguientes alcances y limitaciones:

- La sincronización de las redes MTPM se hara en forma local, para lograr establecer la duración del proceso de sincronización de distintas arquitecturas y escenarios.
- El proyecto se limita a la implementación y la ejecución de pruebas para validar la efectividad de cada uno de los escenarios propuestos.

4.2. Resultados Esperados

En este trabajo de investigación se pretende:

1. Código fuente, manual de usuario y diseño de topología para realizar pruebas de sincronización en lotes.
2. Escribir un artículo científico, para ser sometido en una revista científica de corriente principal con índice WoS o Scopus.
3. Un nuevo algoritmo para el intercambio de claves utilizando Multilayer Tree Parity Machine con estímulos no binarios.

4.3. Justificación de la Investigación

El protocolo de autenticación es la primera barrera de defensa frente a un atacante. Validar la identidad de un usuario permite una restricción del control del dispositivo y sus capacidades para afectar otros equipos a su alcance, sin embargo, los protocolos de autenticación requieren de un elemento secreto conocido solo por el usuario y el sistema que valida su identidad. Para poder intercambiar el elemento secreto en entornos inseguros, se han desarrollado diversas técnicas de intercambio de llaves que han permitido el desarrollo de protocolos que logran mantener la confidencialidad de las llaves compartidas.
En la literatura se pueden encontrar protocolos de autenticación que utilizan criptografía asimétrica y criptografía de bajo costo computacional. Sin embargo, los protocolos de autenticación basados en criptografía asimétrica utilizan matemáticas complejas (por ejemplo, factorización de números gigantes y cálculos de logaritmos discretos) y supone una barrera de entrada para dispositivos con pocas capacidades de procesamiento [12]. Es por esto que se han propuesto una serie de alternativas que combinan técnicas establecidas y nuevos avances orientados a dispositivos con bajo poder computacional [13, 14, 15]. La criptografía neuronal utilizando Tree Parity

Machines (TPM) nace como una alternativa viable, debido a que utiliza operaciones matemáticas simples y no requiere del uso de grandes números enteros. Sin embargo, las redes TPM tienen vulnerabilidades inherentes a la estructura elegida por sus parámetros, es decir, el número de neuronas, el número de entradas para cada neurona y el rango de valores que pueden tomar los pesos sinápticos [8]. Existen una serie de avances que demuestran cómo la seguridad de una red TPM puede ser mejorada al introducir cambios a la estructura, que incluyen el aumento de capas ocultas, la implementación de nuevas reglas de aprendizaje y variaciones particulares a la hora de implementar un protocolo basado en redes TPM [16, 17, 18, 19]. Durante los últimos años, varias propuestas han sugerido distintos usos para las redes TPM, sin embargo la falta de detalles con respecto a su implementación y aplicabilidad a distintos entornos pone en duda su viabilidad en dispositivos con restricciones en su capacidad de procesamiento. A continuación, la tabla 4.1 muestra algunas de las patentes relevantes.

Tabla 4.1: Patentes sobre criptografía neuronal con Tree Parity Machines (TPM)

Código	**Nombre**	**Descripción**	**Enlace**
US 20210250812	*Decentralized Infrastructure Methods and Systems*	Propone el uso de redes TPM para el intercambio de una clave secreta para ser utilizada en transacciones dentro de un sistema descentralizado.	patentscope - wipo.int
CN 112751671	*Novel Key Exchange Method Based on Tree Parity Machine*	Propone un método de intercambio de claves basado en redes TPM, utilizando técnicas de ventana deslizante y un registro de desplazamiento con retroalimentación lineal para acelerar el proceso de sincronización neuronal.	patentscope - wipo.int
IN 201741024592	*Secure System for Debit/Credit Card Swiping Using OTP for Banking Systems Applications*	Propone un método de autenticación basado en redes TPM para la generación de contraseñas de un solo uso (OTP).	patentscope - wipo.int

Tabla 4.1 – continuación

Código	Nombre	Descripción	Enlace
US 20210336779	*Method and Apparatus for Generating Secret Key Based on Neural Network Synchronization*	Método de generación de clave secreta mediante sincronización de redes TPM, intercambiando codewords entre dispositivos y compartiendo información de restauración.	patentscope - wipo.int
AU 2021104099	*A Neural Cryptography Based on Fast Learning Rule*	Propone usar criptografía neuronal con redes TPM para compartir de forma segura una clave secreta, ajustando pesos sinápticos según la sincronización de resultados y reduciendo el tiempo de negociación.	patentscope - wipo.int
US 20230269079	*Device and Method for Sharing Secret Key Based on Neural Network*	Método para compartir una clave secreta entre dos redes TPM mediante sincronización neuronal, utilizando imágenes del servidor y cliente para generar semillas y lograr la clave compartida.	patentscope - wipo.int

En este trabajo se propone integrar diversas modificaciones previamente sugeridas que han demostrado ser efectivas para mejorar la seguridad y el rendimiento en la sincronización de redes neuronales, enfocándose en el uso de redes MTPM con estímulos no binarios. El objetivo es ofrecer una alternativa viable a los algoritmos tradicionales de intercambio de claves, aprovechando que las redes MTPM requieren menor capacidad de cómputo y memoria para restablecer una clave secreta.
Finalmente, se busca validar experimentalmente el algoritmo desarrollado mediante un caso de estudio, utilizando como métrica principal el número de estimulaciones necesarias para lograr la sincronización entre dos redes MTPM. Esta métrica permitirá cuantificar la eficiencia computacional del algoritmo, ofreciendo una evaluación concreta de su rendimiento en términos de carga de trabajo y uso de recursos.

Capítulo 5

Marco de Referencia

5.1. Cuadro Sinóptico

5.2. Estado del Arte

5.2.1. Criptografia Neuronal

La criptografía neuronal nace en el año 2002 con la introducción del trabajo por Wolfgang Kinzel e Ido Kanter [20]. Este trabajo explora la sincronización entre dos redes neuronales mediante el aprendizaje mutuo, es decir, utilizando una regla de aprendizaje en donde los ajustes son condicionados por las salidas de ambas redes. El artículo propone el uso de redes neuronales del tipo Tree Parity Machine (TPM) debido a la dificultad que implica a un atacante establecer una relación directa entre las salidas de las redes y el vector de estímulos de entrada (datos que serán transmitidos a través de un canal público). Para lograr la sincronización en un estado paralelo (es decir $\underline{w}^A = \underline{w}^B$) utiliza la siguiente regla de aprendizaje para la red A:

$$w_{i,j}^{A+} = g(w_{i,j}^A + x_{i,j}^A \cdot \tau^A \cdot \Theta(\sigma_i^A \cdot \tau^A) \cdot \Theta(\tau^A \cdot \tau^B))$$

y de igual manera para la red B:

$$w_{i,j}^{B+} = g(w_{i,j}^B + x_{i,j}^B \cdot \tau^B \cdot \Theta(\sigma_i^B \cdot \tau^B) \cdot \Theta(\tau^A \cdot \tau^B))$$

En donde $w_{i,j}^A$ es el pesos sináptico asociado a la entrada j de la neurona i de la red A, τ^A es la salida de la red A y $x_{i,j}^A$ es el estímulo que ingresa a la red A, en el estímulo j de la neurona i, el cual tiene valores idénticos para ambas redes A y B. El trabajo reconoce inmediatamente el potencial que la sincronización neuronal tiene

en el campo de la criptografía como una alternativa al uso de teoría de números para el intercambio de llaves, acuñando el término criptografía de redes neuronales.
Durante ese mismo año [21] establece un método genérico para el análisis del aprendizaje mutuo en máquinas de paridad con estructura de árbol (o también *Tree Parity Machines* o TPM). Este trabajo prueba que la sincronización de dos redes TPM necesita de un número finito de pasos intercambiando las entradas y salidas de ambas redes. También establece que un atacante que intenta sincronizar con la red sin participar del proceso de aprendizaje es capaz de sincronizar con las redes, sin embargo, suele hacerlo utilizando el doble tiempo que aquellas redes que participan del intercambio de llaves.
Más tarde [22] sería el primer trabajo en introducir los ataques genéticos, geométricos y probabilísticos.

- El ataque genético utiliza algoritmos genéticos, en el cual se simula una gran población de redes neuronales. Este algoritmo elimina a las redes cuyas salidas no coinciden con las redes presentes en el intercambio.

- El ataque geométrico utiliza la interpretación geométrica de la acción del perceptrón para extrapolar información sobre sus pesos sinápticos asociados.

- El ataque probabilístico calcula un nivel de confianza asociada a cada salida de las neuronas ocultas de la red a partir de una red TPM atacante.

Para mejorar la seguridad del proceso de intercambio, el trabajo de [23] presenta un protocolo de intercambio de llaves y describe las siguientes generalizaciones para la optimización del proceso de sincronización:

- Paquetes de bits: Los participantes generan una colección de entradas sobre la cual se calculan las salidas, reduciendo el tiempo de sincronización.

- Varios participantes: El sistema puede extenderse más alla de dos participantes.

- Generación de entradas: Los vectores de estímulos de entrada pueden ser combinados con otros datos presentes en el proceso de sincronización (como por ejemplo, la señal de salida de la red) para maximizar la aleatoriedad de los datos compartidos a través del canal público.

- Permutación de pesos sinápticos: Se puede mejorar la seguridad del canal aplicando permutaciones sobre los pesos sinápticos, sin embargo, el tiempo de sincronización aumenta.

Años más tarde, [24] define un esquema de emisión de llaves basado en redes neuronales utilizando redes TPM. El trabajo detalla un protocolo que permite el establecimiento de una llave secreta utilizando un generador de números pseudo-aleatorio. Este generador se encarga de la creación de estímulos de manera que ambas redes pueden sincronizar sin comunicar los estímulos que son utilizados para realizar ajustes de pesos sinápticos.
El trabajo de [25] introduce consultas para el cálculo de estímulos de entrada, a diferencia del uso de entradas aleatorias. Los autores proponen el uso de consultas, a partir de las cuales se pueden generar los estímulos de entrada que alimentan a ambas redes. Las consultas buscan inducir un campo local h_k en una neurona oculta k previamente definido, para aumentar la seguridad del sistema frente a ataques que utilizan el campo inducido para corregir a una red atacante (como por ejemplo, el ataque geométrico). El trabajo propone dos ecuaciones para la generación de una consulta, las cuales utilizan el estado interno actual de la red.

$$c_{k,l} = \left\lfloor \frac{n_{k,l}+1}{2} + \frac{1}{2l}\left(\sigma_k H\sqrt{N} - \sum_{j=l+1}^{L} j(2c_{k,j} - n_{k,j})\right)\right\rfloor \tag{5.1}$$

$$c_{k,l} = \left\lceil \frac{n_{k,l}-1}{2} + \frac{1}{2l}\left(\sigma_k H\sqrt{N} - \sum_{j=l+1}^{L} j(2c_{k,j} - n_{k,j})\right)\right\rceil \tag{5.2}$$

En donde L es el límite del rango de valores de los pesos sinápticos, H es el valor del campo local que se desea inducir, N es la cantidad de entradas de la neurona k, σ_k es la salida de la neurona k; $c_{k,l}$ es la cantidad de veces que $w_{k,j} \cdot x_{k,j} = l$, con $l \in [-L, L]$; y $n_{k,l} = c_{k,l} + c_{k,-l}$. La ecuación utilizada para generar la consulta es elegida aleatoriamente durante el proceso de sincronización.
Un año más tarde, los autores en [8] determinaron experimentalmente que el rango de valores para los pesos sinápticos L tenía un impacto en la seguridad del proceso de sincronización, aumentando la dificultad para un atacante de lograr una sincronización de manera exponencial, sin embargo también aumentando el tiempo necesario para lograr la sincronización de manera polinomial. El trabajo prueba el impacto del parámetro L frente a ataques simples, geométricos, genéticos y de mayoría. Mientras que en [26] se demuestra que el tiempo de sincronización aumenta exponencialmente al aumentar el parámetro L, cuando la cantidad de neuronas ocultas es mayor a tres.

5.2.2. Variaciones de Arquitecturas MTPM

Si bien las redes Tree Parity Machine han demostrado ser una solución eficiente para el intercambio de llaves, su estructura de una capa oculta, impone ciertas limitaciones en términos de capacidad de aprendizaje, robustez y seguridad. La resistencia de una red TPM frente a ataques de sincronización no autorizada depende principalmente de la profundidad de sus pesos sinápticos y de su arquitectura, es decir, la cantidad de neuronas y entradas en su capa oculta [26]. Alterar cualquiera de los parámetros mencionados anteriormente puede mejorar la seguridad del intercambio, pero impactando negativamente en el tiempo de sincronización, lo que ha impulsado la búsqueda de alternativas para mejorar la seguridad o la eficiencia del proceso de sincronización.

En [16], se explora el uso de números complejos en las entradas, pesos sinápticos y salidas de la red neuronal. Esta nueva estructura es denominada "Complex-Valued Tree Parity Machine" (CVTPM). Sus resultados muestran una gran mejora en la seguridad con respecto a las TPM, aunque requieren de un tiempo de sincronización más extenso.

El trabajo de [27] define una red "Multi-layer feedforward con elementos TPM", utilizando entradas sin traslape para la primera capa y traslape completo entre las entradas de las capas ocultas. El trabajo propone el uso de tres capas ocultas para dificultar la sincronización no autorizada de un atacante.

Otros trabajos han explorado la adición de nuevas capas ocultas y modificaciones al algoritmo de aprendizaje. En [17] se utilizan dos capas ocultas y una regla de aprendizaje basada en el algoritmo "Whale Optimization Algorithm" (WOA). Sus resultados muestran tiempos de sincronización más cortos que una red CVTPM, con mejoras de seguridad con respecto a una red TPM simple que utiliza los mismos parámetros.

En [28] se explora una versión generalizada de las redes TPM, llamada "Vector-Valued Tree Parity Machine" (VVTPM), que es capaz de generalizar otras variaciones propuestas con anterioridad. El trabajo compara los tiempos de sincronización entre las TPM, CVTPM y VVTPM. A través de los resultados, demuestran la capacidad de las VVTPM de generalizar y replicar los resultados obtenidos en otras arquitecturas.

En [18] se comparan los tiempos de sincronización entre redes VVTPM y una variación que incluye tres capas ocultas y la regla de aprendizaje basada en el algoritmo WOA. Esta última variación, llamada "Triple-Layer Tree Parity Machine" (TLTPM), muestra tiempos de sincronización más acotados que otras arquitecturas, tanto para el tiempo mínimo como máximo de sincronización.

En [19] se ha demostrado que la utilización de estímulos no binarios puede reducir los tiempos de sincronización y aumentar la seguridad del sistema.

5.2.3. Implementaciones de Criptografía Neuronal

Los trabajos presentados en la subsección anterior sentaron las bases teóricas sobre las dinámicas de sincronización entre dos redes TPM, lo cual permitió abrir el camino hacia el diseño de implementaciones prácticas de criptografía neuronal. Estas implementaciones buscan aprovechar las propiedades de sincronización para establecer llaves compartidas de manera segura en distintos entornos, incluyendo dispositivos embebidos, sistemas distribuidos y redes IoT. La Tabla 5.1 resume distintos trabajos que implementan redes TPM en esquemas criptográficos, abarcando propuestas desde el año 2005 en adelante.

Tabla 5.1: Resumen de trabajos que implementan redes TPM en esquemas criptográficos.

Año	**Referencia**	**Descripción**
2023	[29]	Un esquema de intercambio de llaves que utiliza un método novedoso de verificación para el proceso de sincronización, basado en la transmisión de llaves parciales.
2020	[30]	Propone un esquema de encriptación para agentes móviles utilizando un intercambio de llaves basado en redes TPM. El esquema de encriptación utiliza la sincronización neuronal para la generación de una llave secreta, la cual es utilizada en el algoritmo de cifrado New Tiny Encryption Algorithm (NTEA).
2020	[31]	Propone el uso de criptografía neural para generar la llave utilizada en la encriptación y desencriptación de imágenes secretas.
2020	[32]	Propone dos estructuras para agrupar distintos usuarios (cada uno con una red TPM) para esquemas para el establecimiento y manejo de llaves grupales.
		Continúa en la siguiente página

(Continuación de la Tabla 5.1)		
Año	**Referencia**	**Descripción**
2017	[33]	Una implementación en hardware que permite el establecimiento de llaves y su restablecimiento a través de circuitos CMOS de 65 nanómetros.
2008	[34]	Propone dos protocolos de autenticación mediante contraseñas de un solo uso (One-Time Password, OTP) basados en TPM para la generación y el restablecimiento de la contraseña.
2007	[35]	Una solución de intercambio de llaves autenticado y sin latencia, resistente a ataques físicos sobre hardware. Permite la autenticación de los participantes del bus de datos, así como la encriptación de la comunicación chip a chip.
2007	[36]	Un esquema de acuerdo de llaves grupales sin necesidad de un servidor central, mediante la sincronización distribuida en pares de dispositivos.
2005	[24]	Un esquema de intercambio de llaves que utiliza funciones hash para verificar la sincronización entre dos TPMs.

5.3. Metodología

5.3.1. Tipo de Metodología

La metodología utilizada en esta investigación es del tipo experimental: Se diseñarán y realizarán simulaciones desarrollando un software de pruebas en un entorno controlado.
La investigación realizará un estudio cuantitativo que comparará las ventajas que presentan las redes Tree Parity Machine Multilayer con estímulos no binarios frente a las redes Tree Parity Machine tradicionales, en términos de seguridad. Las métricas que permiten medir la seguridad de un protocolo que utiliza sincronización neuronal son las siguientes:

1. La cantidad de iteraciones necesarias para lograr la sincronización entre los

participantes del intercambio.

2. El porcentaje de sincronizaciones exitosas para un atacante.

5.3.2. Caso de Estudio

La investigación realizará un estudio cuantitativo que comparará las ventajas que presentan las redes Tree Parity Machine Multilayer con estímulos no binarios frente a las redes Tree Parity Machine tradicionales, en términos de eficiencia. Las métricas que permiten medir la eficiencia de un proceso de sincronización neuronal son las siguientes:

- La cantidad de iteraciones necesarias para lograr la sincronización entre dos redes MTPM.
- El porcentaje de sincronizaciones que finalizan antes de un número determinado de iteraciones.

5.4. Caso de Estudio

En el marco de esta investigación, resulta necesario explorar mediante una implementación práctica el rendimiento de los tres escenarios propuestos para las redes MTPM. En este contexto, se requiere de un caso de estudio que permita generar resultados relevantes que logren ilustrar las capacidades de la propuesta de abordar los desafíos relevantes a equipos IoT de uso general.
Para entender la naturaleza de la sincronización de las redes MTPM, se opta por la creación de una suite de software compuesta por un motor de simulación y una interfaz gráfica basada en navegadores web diseñada para aprovechar al máximo las capacidades integradas al motor de simulación. El objetivo de esta suite de software es generar una base de datos con estadísticas sobre distintas sesiones finalizadas, que permitan analizar la eficiencia de distintas arquitecturas y el impacto de la adición de nuevas capas para tres escenarios propuestos: Redes con conexiones entre capas sin traslape, con traslape parcial y traslape completo (completamente conectadas).
Para lograr los objetivos del proyecto, el motor de simulación debe cumplir los siguientes requisitos:

- Lograr una sincronización entre dos redes MTPM para los tres escenarios y reglas de aprendizaje.

- Almacenar los datos de cada intento de sincronización, sea fallido o exitoso, para el cálculo de estadísticas relevantes.
- Permitir el monitoreo en tiempo real del estado de las sincronizaciones en curso.
- Permitir la implementación de nuevos escenarios y reglas de aprendizaje sin la necesidad de introducir cambios que afecten a las implementaciones previas (también conocidos como *breaking changes*).

Finalmente, se validará la propuesta mediante el desarrollo de una librería para dispositivos IoT que permita utilizar las redes MTPM en dispositivos IoT. La propuesta debe permitir la elección de cualquiera de las tres reglas de aprendizaje (Hebbian, Anti-Hebbian y Random-Walk) y los tres escenarios propuestos.
Para cumplir los requisitos se optan por lenguajes y marcos de trabajo presentados en la tabla 5.2.

Software	**Lenguaje**	**Marco de Trabajo**
Motor de Simulación	*Go*	Concurrencia utilizando librería *conc*
Interfaz Gráfica	*Typescript*	*SvelteKit*
Librería IoT	*C*	Módulo para *MicroPython*

Tabla 5.2: Descripción de herramientas de dessarrollo utilizadas.

La figura 5.1 muestra las interacciones diseñadas entre la interfaz gráfica y el motor de simulación.

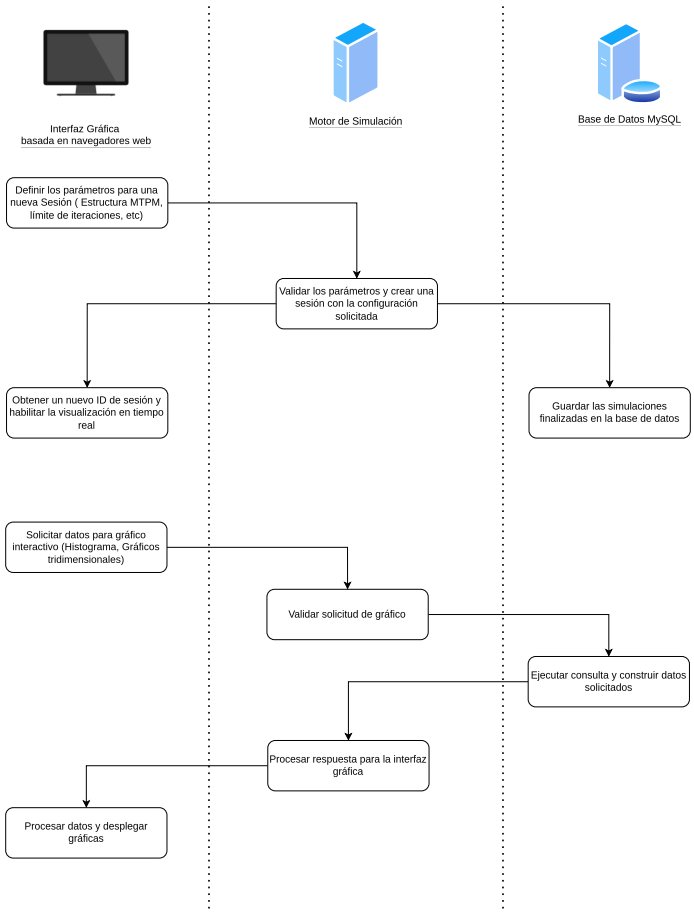

Figura 5.1: Interacciones entre la interfaz gráfica y el motor de simulación.

Para el motor de simulación será necesario un dispositivo multi-núcleo, con capacidades de memoria superiores o iguales a los 8GB DDR4 y con conectividad cableada. La librería IoT será diseñada para la plataforma **ESP32**, un dispositivo Soc utilizado ampliamente tanto por el sector aficionado como profesional y académico.
El software desarrollado para el proyecto será construido sobre lenguajes y librerías open source, con licencias libres que permitan el desarrollo del programa sin la necesidad de adquirir permisos adicionales.

5.5. Tareas

Tabla 5.3: Objetivos y tareas del proyecto

Obj.	**Tarea**	**Descripción**
1	1.1	Búsqueda de artículos, usando el Metabuscador UCN, Springer, WoS, Scopus, IEEE, APS y ScienceDirect.
	1.2	Clasificación de artículos según su relevancia y año.
	1.3	Discusión de los artículos según su relevancia, en particular focalizándose en los parámetros que definen la arquitectura de una red TPM.
2	2.1	Definir la secuencia de pasos para llevar a cabo el proceso de sincronización entre dos MTPM.
	2.2	Comparar el diseño frente a algoritmos de sincronización de una sola capa.
3	3.1	Seleccionar las herramientas, lenguajes de programación y librerías necesarias para desarrollar el algoritmo propuesto.
	3.2	Desarrollar un software de simulación para determinar un rango de parámetros óptimos para el caso de estudio, que definen la estructura de la Multilayer Tree Parity Machine.
	3.3	Desarrollar software para el análisis de los resultados de las simulaciones de sincronización.
4	4.1	Definir las métricas de evaluación del rendimiento del algoritmo, en términos de las iteraciones necesarias para lograr la sincronización y el porcentaje de éxito para un atacante.
	4.2	Evaluar el rendimiento para las arquitecturas de redes MTPM, mediante el uso del software de simulación desarrollado.
	4.4	Análisis de los resultados experimentales y comparación entre los escenarios propuestos.

A continuación, se detallan las tareas indicadas en la tabla 5.3 de acuerdo a su descripción en relación a los objetivos.

O.E. 1 Analizar el estado del arte de las redes Tree Parity Machine Multicapa y sus posibles aplicaciones criptográficas.

1.1 Búsqueda de artículos: Se realizará una búsqueda exhaustiva de artículos científicos utilizando el metabuscador de la UCN y las plataformas Springer, Web of Science, Scopus, IEEE Xplore, APS y ScienceDirect. La búsqueda se enfocará en temas relacionados con redes neuronales, sincronización, criptografía y arquitecturas de Tree Parity Machine. Se establecerán combinaciones de palabras clave tanto en inglés como en español para optimizar los resultados, asegurando una cobertura amplia y relevante.
1.2 Clasificación de artículos según su relevancia y año: Los artículos recopilados serán clasificados según criterios de relevancia y actualidad. Se priorizarán artículos recientes (últimos 10 años) y aquellos con alto impacto, según el número de citaciones y calidad de la revista. Además, se verificará si abordan directamente aspectos de sincronización y arquitectura de redes TPM o sus variantes multicapa.
1.3 Discusión de los artículos según su relevancia, en particular focalizándose en los parámetros que definen la arquitectura de una red TPM: Se desarrollará una discusión crítica entre el estudiante y el tutor, enfocándose en los aspectos técnicos de cada propuesta relevante. Se analizarán los parámetros estructurales como el número de capas, la función de activación, el rango de pesos y el tipo de estímulo utilizado, lo cual permitirá delimitar el marco conceptual de la investigación.

O.E. 2 Diseñar un protocolo de sincronización criptográfica usando redes TPM Multicapa.

2.1 Definir la secuencia de pasos para llevar a cabo el proceso de sincronización entre dos MTPM: Se especificará detalladamente la secuencia lógica del proceso de sincronización, estableciendo roles, flujos de datos, condiciones de inicio y término, así como protocolos de comunicación. Este proceso servirá como base para futuras fases de implementación y prueba.
2.2 Comparar el diseño frente a algoritmos de sincronización de una sola capa: Se realizará una comparación estructurada entre el protocolo propuesto y algoritmos clásicos de sincronización basados en una única capa TPM. El análisis incluirá ventajas, desventajas, complejidad computacional y capacidad de resistencia frente a ataques.

O.E. 3 Implementar una herramienta de simulación para el protocolo propuesto.

3.1 Seleccionar las herramientas, lenguajes de programación y librerías necesarias para desarrollar el algoritmo propuesto: Se realizará un estudio de herramientas y entornos de desarrollo adecuados (como Python, NumPy, Matplotlib, etc.), evaluando su compatibilidad con la simulación de redes neuronales, su capacidad de visualización y su facilidad de integración con bases de datos.
3.2 Desarrollar un software de simulación, para determinar un rango de parámetros óptimos para el caso de estudio, que definen la estructura de la Multilayer Tree Parity Machine: Se desarrollará un motor de simulación capaz de ejecutar múltiples configuraciones de MTPM, permitiendo medir su desempeño y determinar valores óptimos para parámetros como número de capas, longitud de peso, rango sináptico y tipo de estímulo. Este motor debe registrar los resultados de forma estructurada para su posterior análisis.
3.3 Desarrollar software para el análisis de los resultados de las simulaciones de sincronización: Se desarrollarán herramientas de visualización y análisis estadístico, como generación de histogramas, gráficos de convergencia y análisis de éxito de sincronización. Esto facilitará la interpretación de los datos generados por el simulador y permitirá comparar distintas configuraciones de manera objetiva.

O.E. 4 Evaluar el rendimiento del algoritmo propuesto mediante simulación.

4.1 Definir las métricas de evaluación del rendimiento del algoritmo, en términos de las iteraciones necesarias para lograr la sincronización y el porcentaje de éxito para un atacante: Se establecerán métricas cuantitativas clave como el número de iteraciones de aprendizaje requeridas para lograr sincronización, la tasa de éxito de sincronización y la vulnerabilidad frente a un atacante pasivo o activo. Estas métricas permitirán evaluar la eficiencia y seguridad del modelo propuesto.
4.2 Evaluar el rendimiento para las arquitecturas de redes MTPM, mediante el uso del software de simulación desarrollado: Se utilizará el simulador implementado para analizar múltiples configuraciones de MTPM, identificando aquellas que logran sincronización en menor tiempo, con mayor estabilidad y menor susceptibilidad a ataques.
4.4 Análisis de los resultados experimentales y comparar resultados entre

escenarios propuestos: Se elaborará un análisis comparativo entre distintos escenarios experimentales, considerando cambios en la topología, estímulo, profundidad o función de activación. Los resultados serán comparados estadísticamente, destacando las arquitecturas más eficientes y robustas.

Bibliografía

[1] A. Greenberg, "'Crash Override': The Malware That Took Down a Power Grid," June 2017.

[2] IBM, "Cost of a Data Breach Report 2023," tech. rep., 2023.

[3] X. Jin, S. Manandhar, K. Kafle, Z. Lin, and A. Nadkarni, "Understanding IoT Security from a Market-Scale Perspective," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, CCS '22, (New York, NY, USA), pp. 1615–1629, Association for Computing Machinery, 2022.

[4] I. Jirón, "APLICACIONES DE REDES NEURONALES ARTIFICIALES EN CIBERSEGURIDAD," in *XXXI COMCA*, (Antofagasta, Chile), Universidad Católica del Norte, Aug. 2023.

[5] D. Schweitzer, *Internet Security Made Easy*. Amacom Books, Jan. 2002.

[6] C. Paar and J. Pelzl, *Understanding Cryptography. A Textbook for Students and Practitioners*. Springer, Oct. 2011.

[7] E. Barker and A. Roginsky, "Transitioning the use of cryptographic algorithms and key lengths," tech. rep., Mar. 2019.

[8] A. Ruttor, "Neural synchronization and cryptography," Jan. 2006.

[9] S. S. Haykin, *Neural Networks and Learning Machines*. Prentice Hall, Jan. 2009.

[10] C. Coarfa, P. Druschel, and D. S. Wallach, "Performance analysis of TLS web servers," *ACM Transactions on Computer Systems*, vol. 24, pp. 39–69, Feb. 2006.

[11] A. A. M. Teodoro, O. S. M. Gomes, M. Saadi, B. A. Silva, R. L. Rosa, and D. Z. Rodríguez, "An FPGA-Based Performance Evaluation of Artificial Neural

Network Architecture Algorithm for IoT," *Wireless Personal Communications*, vol. 127, pp. 1085–1116, May 2021.

[12] M. Suárez-Albela, T. M. Fernández-Caramés, P. Fraga-Lamas, and L. Castedo, "A Practical Performance Comparison of ECC and RSA for Resource-Constrained IoT Devices," in *2018 Global Internet of Things Summit (GIoTS)*, pp. 1–6, 2018.

[13] S. F. Aghili, H. Mala, P. Kaliyar, and M. Conti, "SecLAP: Secure and lightweight RFID authentication protocol for Medical IoT," *Future Generation Computer Systems*, vol. 101, pp. 621–634, 2019.

[14] R. Krishnasrija, A. K. Mandal, and A. Cortesi, "A lightweight mutual and transitive authentication mechanism for IoT network," *Ad Hoc Networks*, vol. 138, p. 103003, 2023.

[15] S. Yang, X. Zheng, G. Liu, and X. Wang, "IBA: A secure and efficient device-to-device interaction-based authentication scheme for Internet of Things," *Computer Communications*, vol. 200, pp. 171–181, 2023.

[16] T. Dong and T. Huang, "Neural cryptography based on Complex-Valued Neural Network," *IEEE transactions on neural networks and learning systems*, vol. 31, pp. 4999–5004, Nov. 2020.

[17] A. Sarkar, M. Z. Khan, M. M. Singh, A. Noorwali, C. Chakraborty, and S. K. Pani, "Artificial Neural Synchronization Using Nature Inspired Whale Optimization," *IEEE ACCESS*, vol. 9, pp. 16435–16447, 2021.

[18] A. Sarkar, "Chaos-Based mutual synchronization of Three-Layer Tree parity Machine: A session key exchange protocol over public channel," *Arabian Journal for Science and Engineering*, vol. 46, pp. 8565–8584, Apr. 2021.

[19] M. Stypiński and M. Niemiec, "Synchronization of tree parity machines using nonbinary input vectors," *IEEE transactions on neural networks and learning systems*, pp. 1–7, Jan. 2022.

[20] W. Kinzel and I. Kanter, "Interacting neural networks and cryptography," 2002.

[21] M. Rosen-Zvi, E. Klein, I. Kanter, and W. Kinzel, "Mutual learning in a tree parity machine and its application to cryptography," *Physical Review E: Statistical Physics, Plasmas, Fluids, and Related Interdisciplinary Topics*, vol. 66, p. 066135, Dec. 2002.

[22] A. Klimov, A. Mityagin, and A. Shamir, "Analysis of neural cryptography," in *Advances in Cryptology — ASIACRYPT 2002* (Y. Zheng, ed.), (Berlin, Heidelberg), pp. 288–298, Springer Berlin Heidelberg, 2002.

[23] IDO. KANTER and WOLFGANG. KINZEL, "The theory of neural networks and cryptography," in *The Physics of Communication*, pp. 631–642.

[24] T. Chen, B. Chen, and C. Jiamei, "A novel Identity-Based key issuing scheme based on interacting neural network," in *Lecture Notes in Computer Science*, pp. 637–642, Jan. 2005.

[25] A. Ruttor, W. Kinzel, and I. Kanter, "Neural cryptography with queries," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2005, p. P01009, Jan. 2005.

[26] A. Ruttor, W. Kinzel, and I. Kanter, "Dynamics of neural cryptography," *Physical Review E*, vol. 75, May 2007.

[27] E. Shishniashvili, L. Mamisashvili, and L. Mirtskhulava, "Enhancing IoT security using multi-layer feedforward neural network with tree parity machine elements," *International Journal of Simulation Systems Science & Technology*, Mar. 2020.

[28] S. Jeong, C. Park, D. Hong, C. Seo, and N. Jho, "Neural Cryptography Based on Generalized Tree Parity Machine for Real-Life Systems," *Security and Communication Networks*, vol. 2021, pp. 1–12, Feb. 2021.

[29] J. Dey and A. Bhowmik, "Concurring of neural machines for robust session key generation and validation in telecare health system during COVID-19 pandemic," *Wireless Personal Communications*, vol. 130, pp. 1885–1904, Apr. 2023.

[30] P. Kumar, D. N. Singhal, and K. Chaitra, "Securing mobile agents migration using tree parity machine with new tiny encryption algorithm," pp. 129–138, July 2020.

[31] M. Gupta, M. Gupta, and M. Deshmukh, "Single secret image sharing scheme using neural cryptography," *Multimedia Tools and Applications*, vol. 79, May 2020.

[32] K. S. S. Santhanalakshmi and G. K. Patra, "Design of group key agreement protocol using neural key synchronization," *Journal of Interdisciplinary Mathematics*, vol. 23, no. 2, pp. 435–451, 2020.

[33] H. Gómez, Ó. Reyes, and E. Roa, "A fully synthesized key establishment core based on tree parity machines in 65nm CMOS," in *2016 12th Conference on Ph.D. Research in Microelectronics and Electronics (PRIME)*, pp. 1–4, 2016.

[34] T. Chen and S. H. Huang, "Tree parity machine-based One-Time Password authentication schemes," in *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, pp. 257–261, 2008.

[35] S. Muhlbach and S. Wallner, "Secure and Authenticated Communication in Chip-Level Microcomputer Bus Systems with Tree Parity Machines," in *2007 International Conference on Embedded Computer Systems: Architectures, Modeling and Simulation*, pp. 201–208, 2007.

[36] B. Saballus, M. Volkmer, and S. Wallner, "Secure group communication in ad-hoc networks using tree parity machines," Jan. 2007.