Redes Neurais Artificiais Aplicadas à Geração de Chaves Criptográficas Binárias

Gustavo Pasqua de Oliveira Celani Marcelo Vinícius Cysneiros Aragão
Instituto Nacional de Telecomunicações - Inatel
{gustavopasqua,marcelovca90}@gec.inatel.br

***Abstract*— Intellectual patrimony has become an asset that is increasingly valuable and, consequently, more targeted by wrongdoers. Information security comes with the objective of protecting this data. With encryption, is possible to make data unintelligible for those who do not have access to the planned rules. Cryptographic keys guarantee the security of the encrypted data regardless of the algorithm used. These keys are usually generated from pseudo-random values, requiring entropy, limiting their sizes by necessity performance, among other negative consequences. In addition, they have yet to be exchanged or shared. The proposal consists of generating cryptographic keys using different configurations of artificial neural networks in order to improve the performance and reliability of the process. The results obtained were discussed in section V-B.**

***Index Terms*— Information Security, Cryptography, Cryptographic Keys, Artificial Neural Networks, Neural Cryptography.**

***Resumo*— O patrimônio intelectual vem se tornando um bem cada vez mais valioso e, consequentemente, mais visado por malfeitores. A segurança da informação surge com o objetivo de proteger estes dados. Com a criptografia, é possível fazer com que dados se tornem ininteligíveis para os que não tenham acesso às regras planejadas. As chaves criptográficas garantem a segurança dos dados encriptados independentemente do algoritmo utilizado. Estas chaves são, normalmente, geradas a partir de valores pseudo-aleatórios, exigindo entropia, limitando seus tamanhos por necessidade de desempenho, dentre outras consequências negativas. Além disso, as chaves ainda têm que ser trocadas ou compartilhadas. A proposta se consiste em gerar chaves criptográficas utilizando diferentes configurações de redes neurais artificiais com o intuito de melhorar o desempenho e a confiabilidade do processo. Os resultados obtidos foram discutidos na seção V-B.**

***Palavras chave*— Segurança da Informação, Criptografia, Chaves Criptográficas, Redes Neurais Artificiais, Criptografia Neural.**

I. Introdução

Após a revolução industrial, que padronizou as atividades industriais por meio do método criado por Taylor, os produtos passaram a apresentar grande semelhança por conta da maior otimização oferecida na produção de itens idênticos. Com isto, clientes passaram a buscar diferenciais nos produtos que consumiam e, portanto, a empresa que possuísse mais diferenciais em suas mercadorias acabaria se destacando no mercado. Para suprir tal necessidade, as instituições passaram a buscar cada vez mais funcionários e contribuidores com conhecimentos nas mais variadas áreas para que novas ideias surgissem, resultando em produtos inovadores [1].

A soma do conhecimento individual dos funcionários, suas experiências e *know-how*, dados gerados por serviços, informações, procedimentos, rotinas e até mesmo documentações da instituição, formam o conhecido patrimônio intelectual, atualmente considerado o mais valioso no contexto empresarial.

Este foi o divisor de águas da transição da sociedade entre a Era Industrial e a Era da Informação, ou Era Digital, na qual o conhecimento acumulado é tido como uma riqueza maior do que bens materiais. Neste novo contexto, a maior parte dos esforços passaram a ser direcionados ao tratamento, armazenamento e preservação destes dados de forma segura.

Para a maioria das instituições ativas, perder seus dados seria algo catastrófico, levando a empresa a perder registros de funcionários, clientes, fornecedores, contabilidade, produtos, dentre outras inúmeras avarias. Caso uma empresa apresente este cenário, seu prejuízo pode ser tão alto que a única saída seria declarar falência. Como foi o caso de uma *exchange* de bitcoin sul-coreana que declarou falência após perder 17% de suas reservas de ativos [2].

Em 2014, a EY Brasil divulgou uma pesquisa sobre segurança de dados nas empresas brasileiras. Os dados mostravam que 54,2% dos empresários brasileiros afirmaram que os riscos de ataques cibernéticos aumentaram no ano de 2013. Ao mesmo tempo, mais de 62% afirmaram que iriam ampliar seus investimentos em segurança da informação com o objetivo de se protegerem destes tipos de ataques [3]. Entretanto, tais esforços aparentemente não vêm sendo suficientes, uma vez que, de acordo com a empresa de segurança digital Gemalto, mais de 2,6 bilhões de dados foram roubados, perdidos ou expostos mundialmente em 2017, constituindo um aumento de 88% em relação a 2016 [4].

É importante ressaltar que não são apenas as empresas que estão sujeitas a este tipo de ataque. Os usuários comuns também são alvos dos crimes cibernéticos, direta ou indiretamente. De forma indireta, dados pessoais — como fotos, arquivos salvos em nuvem, *e-mails*, senhas, números de cartões, dentre outras informações — podem ser vazados da base de dados de um sistema cujo o usuário possui um cadastro. Por exemplo, em 2016, a empresa Uber foi alvo de um ataque *hacker black*

Trabalho de Conclusão de Curso apresentado ao Instituto Nacional de Telecomunicações como parte dos requisitos para obtenção do Grau de Bacharel em Engenharia da Computação. Orientador: Prof. Me. Marcelo Vinícius Cysneiros Aragão. Trabalho aprovado em 10/11/2018.

hat[1] (também conhecidos como *full-fledged* ou *crackers* [5]) que expôs dados pessoais de 57 milhões de usuários cadastros, tanto de clientes quanto de motoristas [6].

Com o objetivo de proteger e preservar os dados de malfeitores, a área de Segurança da Informação (SI) se tornou indispensável na Era da Informação e é normatizada pela ISO[2]/IEC[3] 27002:2013 [7].

A criptografia é uma das ferramentas mais importantes da SI, uma vez que a técnica apoia-se nos seis atributos básicos ilustrados na Figura 1. A confidencialidade limita o acesso a informação tão somente para aqueles que possuem a devida autorização do proprietário da informação; a integridade garante que a informação permaneça isenta de qualquer alteração; a disponibilidade certifica a acessibilidade da informação para quem seja autorizado à utilizá-la; a autenticidade assegura a fonte da informação; a irretratabilidade impossibilita que a autoria da transação seja negada; a conformidade garante que as convenções estabelecidas serão seguidas de forma correta.

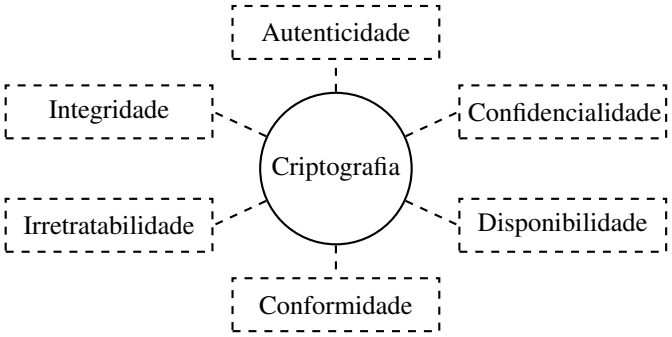

Fig. 1. Criptografia apoiada nos pilares da SI.

O uso da criptografia possibilita serviços ou pessoas trocar informações entre si por meio de um canal inseguro, muitas vezes a Internet, sem que terceiros entendam as mensagens trocadas, mesmo se interceptadas [8].

Para que os algoritmos criptográficos possam proteger uma mensagem legível, é necessária uma chave, cujo a função é controlar a operação do algoritmo de forma que somente esta chave seja capaz de descriptografar a mensagem. A Figura 2 apresenta um cenário típico de uma criptografia simétrica, onde a chave é compartilhada.

[1]*Hackers* que utilizam das vulnerabilidades dos sistemas que encontram para obter dados sigilosos, muitas vezes suplantando os limites da lei.

[2]Do inglês, *International Organization for Standardization.*

[3]Do inglês, *International Electrotechnical Commission*.

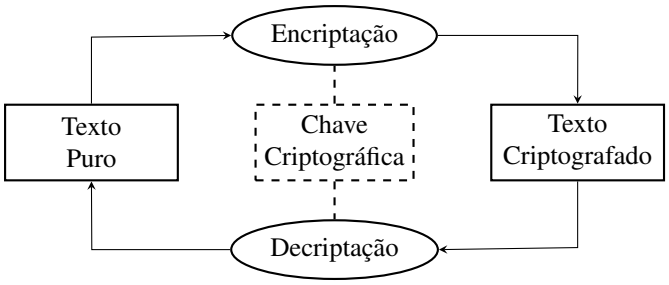

Fig. 2. Compartilhamento de chave em criptografia simétrica.

Se esta chave for descoberta ou até mesmo roubada, as consequências podem ser desastrosas, uma vez que mensagens trocadas pelo serviço que faz o uso desta chave poderão ser descriptografadas e seu conteúdo — muitas vezes privado — exposto. Logo, uma parte essencial para garantir a segurança da informação é a realização da troca das chaves de uma forma segura quando a chave utilizada é compartilhada, como por exemplo na criptografia simétrica.

Isto se evidencia no caso onde o certificado digital do banco Inter foi revogado após o mesmo ter sido publicado, juntamente com sua chave privada. A revogação foi caracterizada por *key compromise*, que designa o vazamento da chave [9]. Em um caso como este, a chave poderia ter sido usada para obtenção dos dados pessoais dos clientes do banco virtual, ou até mesmo para efetuar transações monetárias indiscriminadamente.

Tendo em vista a importância da chave criptográfica, é notório que a principal dificuldade no desenvolvimento de sistemas seguros baseados em trocas de informações não é a escolha de um algoritmo criptográfico ideal, mas sim, ao gerenciamento da estrutura necessária para gerar, autenticar e transmitir as chaves criptográficas [10].

A maneira mais comum de geração destas chaves baseia-se na pseudoaleatoriedade, de forma que nenhuma estrutura matemática ou lógica seja utilizada para recriá-la. Para que esta geração seja o mais aleatória possível, a RFC4086[4] é responsável por padronizar esta geração pseudo-aleatória [11].

Para garantir a pseudoaleatoriedade, é necessário coletar entropia[5] do sistema [12]. Normalmente, ela é capturada por meio de eventos tratados como aleatórios como, por exemplo, as coordenadas de cliques do *mouse*, movimentos do dispositivo de disco, consulta a uma posição de memória qualquer, dentre outros eventos imprevisíveis [13]. Esta situação faz com que, para gerar uma chave, seja necessária um quantidade de entropia do sistema diretamente proporcional ao tamanho da chave. Além disso, em sistemas onde não há muitos periféricos conectados ou fontes de entropia de um modo geral, esta geração acontece de forma ineficiente.

Baseando-se no conteúdo até então apresentado, a proposta deste trabalho consiste na geração de chaves criptográficas

[4]Do inglês, *Request for Comments*: documentos técnicos desenvolvidos e mantidos pelo grupo IETF (do inglês, *Internet Engineering Task Force*, responsável, por exmeplo, pela padronização de protocolos para a Internet.

[5]Informações coletadas do sistema ditas probabilisticamente aleatórias.

binárias de tamanhos arbitrários utilizando redes neurais artificiais, bem como na análise de cada parâmetro utilizado no processo a fim de otimizá-lo.

A. Objetivos

Este trabalho tem como objetivo gerar chaves criptográficas binárias de tamanho arbitrário utilizando criptografia neural. Este procedimento será metrificado com base nos parâmetros necessários para sua realização com a intenção de otimizar o método a partir do tamanho configurado da chave.

B. Contribuições do trabalho

Os estudos deste trabalho visam apontar potenciais melhorias nos modelos atualmente propostos de geração de chaves criptográficas binárias.

II. Revisão da teoria

A. Segurança da Informação

Informação é tudo aquilo considerado relevante o suficiente para ser processado ou armazenado, tanto por humanos, quanto por máquinas. Uma mesma informação possui relevância diferente dependendo de quem a interpreta ou de como é representada.

Exemplo: um conjuntos de dados (informações) referente a uma lista de contatos não teria significado para um humano se fosse representado em forma binária, mas teria caso a representação se desse por meio de caracteres alfanuméricos. Em contrapartida, para que esta mesma lista de contatos seja interpretada por um sistema digital, ela deve estar representada em linguagem de máquina para que estes dados possam ser utilizados em algum serviço. Estas diferenças de representação são ilustradas na Figura 3.

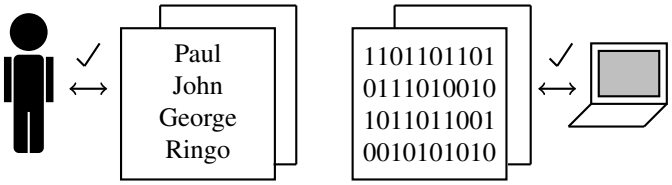

Fig. 3. Diferentes interpretações da informação.

Informação pode ser algo público, como a lista de contatos exemplificada anteriormente, ou dados privados, tais como dados bancários, históricos de transações e localizações, endereços Web visitados, pesquisas realizadas, produtos comprados, número de documentos, relação de bens, endereços, fotos, vídeos, dentre outros. Tendo em vista que o volume de dados vêm crescendo a taxas cada vez maiores [14], surge a necessidade de manter estes dados protegidos contra o uso ou acesso não autorizado; é neste contexto que faz-se relevante o conceito de segurança da informação (SI).

A segurança da informação é padronizada pela norma ISO/IEC 27002:2013 [7]. Nela são definidas as propriedades básicas da SI, sendo elas:

- Confidenciabilidade: resguardo e proteção da informação apenas para quem é autorizado a acessá-la.
- Integridade: garantia da consistência física e/ou lógica dos dados durante toda sua existência.
- Disponibilidade: manutenção da disponibilidade das informações para o maior número de serviços/pessoas possíveis durante o máximo de tempo concebível.
- Autenticidade: garantia que o dado a ser interpretado é íntegro.
- Irretratabilidade ou Não Repúdio: garantia de que o gerador da informação não é capaz de negar sua autoria.
- Conformidade: certeza de que as partes envolvidas seguirão as convenções e regulamentos estabelecidos.

Caso alguma propriedade básica da SI seja desconsiderada ou transgredida, a informação resguardada deixará de ser considerada segura, o que pode acarretar em consequências financeiras, éticas ou morais. Uma série de recursos podem ser utilizados com a finalidade de evitar a violação destas propriedades. Existem mecanismos físicos, como infraestrutura, blindagem, restrição de acesso, dentre outros. Há também os controles lógicos que, dentro de uma gama de métodos existentes, incluem protocolos de comunicação, mecanismos de certificação digital, garantia de integridade por meio de técnicas de *hashing*[6], *firewall*[7] e *proxy*[8] de rede, anti-vírus, assinatura digital e criptografia [15].

B. Criptografia

A criptografia é a ciência da escrita secreta [16]. A necessidade do homem de poder se comunicar sem que as mensagens trocadas fossem interpretadas é antiga. Pode-se observar essa imprescindibilidade da criptografia em cenários como guerras, onde havia necessidade de comunicação com os combatentes sem que os opositores entendessem o que foi passado a eles, ou em situações nas quais é preciso limitar a informação somente para quem é autorizado a obtê-la.

Um dos métodos criptográficos mais tradicionais é o da cifragem simples, que se tornou icônico quando utilizado pelo ex-ditador romano Júlio César para se comunicar com seus subordinados. Neste método de substituição monoalfabético[9], os caracteres das mensagens são substituídos por outros seguindo um número de deslocamento padrão pré-determinado [17]. O método é exemplificado na Figura 4 utilizando a chave "2", ou seja, com deslocamento de duas posições. Foi considerado um alfabeto de 26 letras. $E_n(x)$ representa a função de encriptação e $D_n(x)$ representa a função de decriptação, onde x é o caractere alfabético.

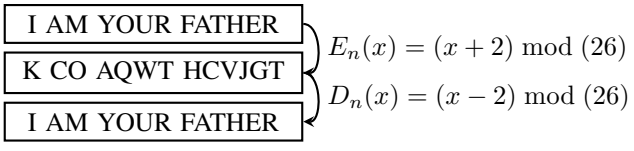

Fig. 4. Cifra de César utilizando a chave "2".

[6]Algoritmos que mapeiam dados de diversos comprimentos para dados de comprimento fixo.

[7]Dispositivo de rede que visa aplicar uma política de segurança na mesma.

[8]Agente intermediário em uma rede que age nas requisições dos clientes.

[9]Cifra de substituição onde cada letra do texto em claro é substituída por uma outra letra no texto cifrado, de forma constante.

Com o recorrente uso de métodos criptográficos nos mais diversos ecossistemas, a criptografia foi adquirindo importância e atraindo o interesse de acadêmicos e entusiastas, implicando no surgimento de novas técnicas cada vez mais aprimoradas.

Como vários pesquisadores passaram a contribuir com o desenvolvimento da criptografia, termos específicos passaram a ser utilizados com fins de padronização. Dentre eles, destacam-se [16][18]:

- Texto puro (*plain text*): Nome dado à mensagem original, sem nenhum tipo de codificação.
- Texto criptografado (*encrypted text*): Se refere à mensagem codificada.
- Encriptação (*encryption*): Ato de codificar o texto puro para o texto criptografado.
- Decriptação (*decryption*): Transformação do texto criptografado de volta ao texto puro.
- Chave criptográfica (*cryptographic key*): Parâmetro utilizado pelos algoritmos de encriptação/decriptação para realizarem suas respectivas funções (explicadas com mais detalhes na seção II-C).

Após o esforço realizado para melhorar cada vez mais as técnicas criptográficas devido a eminente demanda pela segurança da informação, algoritmos relevantes passaram a surgir. Dentre esses algoritmos, podem ser citados:

1) Data Encryption Standard (DES): Seu contexto é dado pela necessidade que o *National Bureau of Standards* (NBS), atual *National Institute of Standards and Technology* (NIST), instituto norte-americano padronizador de tecnologias, sentiu de proteger seus dados virtuais. Para tal, foi criado um algoritmo padrão capaz de proteger dados tanto nas transmissões quanto no armazenamento. A empresa norte-americana IBM (International Business Machines) propôs o Lucifer, um algoritmo de cifragem em blocos de 128 *bits* utilizando uma chave, também, de 128 *bits* [19]. O algoritmo de criptografia simétrica passou por uma série de modificações de terceiros, incluindo diminuição da chave para 64 *bits* (sendo 56 *bits* efetivos e 8 *bits* de paridade com objetivo de controle), e fora publicado em detalhes pela *National Security Agency* (NSA) em 1975. Entretanto, somente no ano seguinte ele foi aceito como padrão e passou a ser chamado de DES [20].

A encriptação utilizando DES (Figura 5) funciona da seguinte maneira: o texto puro é submetido a uma permutação inicial (*Initial Permutation - IP*), o resultado da *IP* passa por uma sequência de 16 estágios idênticos de processamento fazendo uso da Função Feistel (*Feistel-F*) (estrutura simétrica usada na construção de cifras de bloco). A chave criptográfica é então escalonada de forma a gerar 16 sub-chaves de 48 *bits*, cada uma usada em um estágio de processamento. Com o resultado de cada *F* é realizada uma operação lógica OU-EXCLUSIVO (*XOR*) com o anterior. Por fim, o fruto gerado pela sequência de *XOR* das *F* é submetido a uma permutação final (*Final Permutation - FP*), produzindo assim assim o texto criptografado. Para decriptar a mensagem, basta utilizar a mesma estrutura, porém com as sub-chaves geradas espelhadas, ou seja, a *F1* receberá a sub-chave 16 e assim sucessivamente [21].

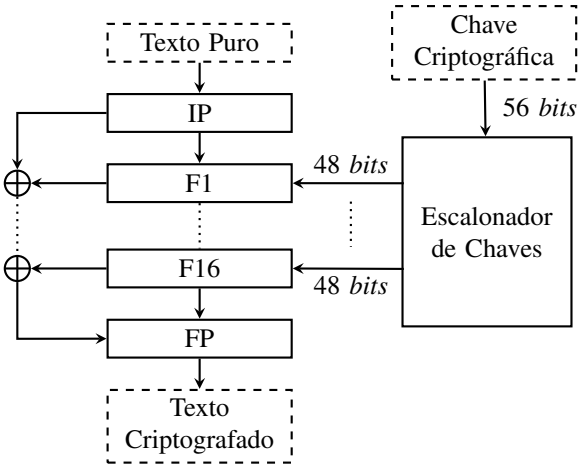

Fig. 5. Procedimento de encriptação utilizando DES.

2) Blowfish e Twofish: Como o DES trabalha com chaves criptográficas de 56 *bits* efetivos, surgiu-se a necessidade da criação de novos algoritmos com chaves maiores para serem seus sucessores. Uma destas propostas foi o *Blowfish*. O funcionamento do algoritmo é semelhante ao DES, porém, comporta chaves de 32 *bits* até 448 *bits*, tornando-o maleável para ser aplicado em diferentes finalidades [22].

Já o algoritmo *Twofish* foi uma proposta baseada no *Blowfish*, porém, utiliza chaves de até 256 *bits*, sendo que metade da chave é utilizada como chave simétrica e a outra metade é usada para remodelar o algoritmo em cada iteração das funções Feistel [23].

3) 3DES: O 3DES, ou *Triple DES*, consistem em três iterações do algoritmo DES, onde a saída de um serve de entrada para o outro. O texto puro é entrada para o primeiro DES, cuja saída é entrada para o segundo DES; analogamente, o terceiro DES utiliza a saída do segundo como entrada e, finalmente, a saída do terceiro DES será o texto criptografado.

4) Cast-128 e Cast-256: Baseado no algoritmo DES, o algoritmo *Cast-128* – também conhecido como *Cast5* – foi publicado em 1997. Este também faz o uso dos 16 *rounds* das funções Feistel, porém, opera com blocos de 64 *bits* e suporta chaves de 40, 64, 80 ou 128 *bits* [24].

Visando aumentar a segurança do algoritmo *Cast-128*, no ano de 1999, foi formalizado o algoritmo *Cast-256* (ou *Cast6*). Este assume blocos de 128 *bits* e aceita chaves de 128, 160, 192, 224, ou 256 *bits* [25]. Para suportar tais parâmetros, ele realiza 48 *rounds*, ou 12 "*quad-rounds*" da função Feistel, onde as sub-chaves escalonadas são de 32 *bits* cada [26].

5) Advanced Encryption Standard (AES): O algoritmo *Rijndael*, eleito como *Advanced Encryption Standard* (AES), opera com blocos de 128 *bits* e chaves de 128, 192 ou 256 *bits*. Ao contrário do algoritmo DES e seus derivados, o AES não utiliza iterações em rede de funções Feistel; em vez disso, faz uso de uma rede de permutação-substituição. Ele foi projetado para ser rápido e exigir poucos recursos de *hardware*, sendo mais eficiente em aplicações com baixo poder de processamento. Ele não possui uma quantidade de *rounds* fixa; para chaves de 128 *bits*, são realizados 10 *rounds*, para

192 *bits*, 12 *rounds* e para chaves de 256 *bits*, 14 iterações são realizadas [27].

No início do processo, o texto puro é disposto em matrizes de 4x4 *bytes* chamadas de estados. Em seguida, a chave é escalonada em sub-chaves de 128 *bits* cada, com a quantidade variando de acordo com o número de *rounds* e sendo uma chave por *round* mais uma adicional para a inicialização do processo. Em cada iteração, uma sub-chave é adicionada, depois é realizada uma etapa de substituição não linear dos *bytes*, de acordo com um estado tido como referência. Na sequência, acontece um deslocamento (*shift*) das linhas dos estados, e os *bytes* de cada coluna dos estados são transformados linearmente. O *round* final se consiste na realização da substituição não linear, transposição das linhas do estado e adição da última sub-chave [27].

6) Rivest-Shamir-Adleman (RSA): Diferentemente dos algoritmos citados previamente, o RSA é dito de criptografia assimétrica, na qual as chaves privadas não são compartilhadas. Tal abordagem é bem difundida em aplicações que realizam transmissão de dados utilizando canais tidos como inseguros como, por exemplo, a Internet. O algoritmo é comumente utilizado para criptografar *e-mails*, autenticação de usuários, transações em sites de comércio eletrônico, aplicações de bate-papo, dentre outros serviços *online*.

A encriptação de mensagens c por meio do RSA é realizada calculando a exponenciação modular da mensagem m, que deve ser obrigatoriamente menor que $n-1$, com o componente e da chave pública: $m^e \equiv c \times mod(n)$. Já a decriptação da mensagem criptografada é realizada com o mesmo processo, porém, utilizando o componente d da chave privada: $c^d \equiv m \times mod(n)$.

Para o processo de geração do par de chaves (pública e privada) são escolhidos pseudo-aleatoriamente dois números probabilisticamente primos (p e q) com quantidade de dígitos na ordem de 10^{100}. Estes números são multiplicados um pelo outro, produzindo n. Em seguida, é calculada a função Carmichael $\phi_{(n)}$ (cálculo do mínimo múltiplo comum entre os antecessores de p e q). Feito isso, um inteiro pseudo-aleatório e maior que 1 e menor que $\phi_{(n)}$ é escolhido de tal forma que sejam primos entre si. Este inteiro é usado no cálculo de d, de modo que d seja o inverso multiplicativo de e (ou seja, $d \times e \equiv 1$). Ao fim deste processo, dois conjuntos serão gerados: o que representa a chave privada (*PVK*) – par $[n, e]$ – e o que caracteriza a chave pública (*PBK*) – $[p, q, d]$. Este processo é ilustrado no diagrama da Figura 6.

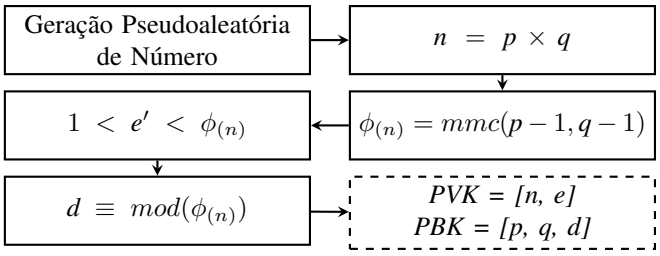

Fig. 6. Diagrama de geração de chaves RSA.

C. Chaves Criptográficas

Chave criptográfica é uma informação que controla os algoritmos de criptografia. Por exemplo, para um mesmo algoritmo criptográfico e um mesmo texto puro, obtêm-se textos criptografados distintos ao utilizar chaves diferentes.

Existem basicamente dois tipos de chaves: as públicas e as privadas. As do primeiro tipo são comumente usadas em algoritmos de criptografia assimétricos, sendo responsáveis pela autenticação e encriptação da mensagem. Já as do segundo, além de serem usadas na criptografia assimétrica para decriptação da mensagem, também se fazem úteis na criptografia simétrica, onde são compartilhadas pelo transmissor e receptor de tal forma que somente ela seja capaz de encriptar e decriptar uma mensagem.

Para que as chaves possam ser trocadas entre serviços, existem algoritmos que possibilitam estas transferências em meio a canais inseguros de comunicação. Um destes algoritmos de intercâmbio de chaves criptográficas é o Diffie-Hellman padronizado pela RFC2631 [28].

O protocolo baseia-se em operações logarítmicas discretas para realizar manipulações locais com uma chave pública comum a ambas partes, de forma que uma nova informação comum secreta possa ser obtida ao final do processo. Entretanto, para realizar a troca pelo algoritmo Diffie-Hellman, é necessária a geração prévia de uma chave pública de uma chave privada. O protocolo está representado na Figura 7.

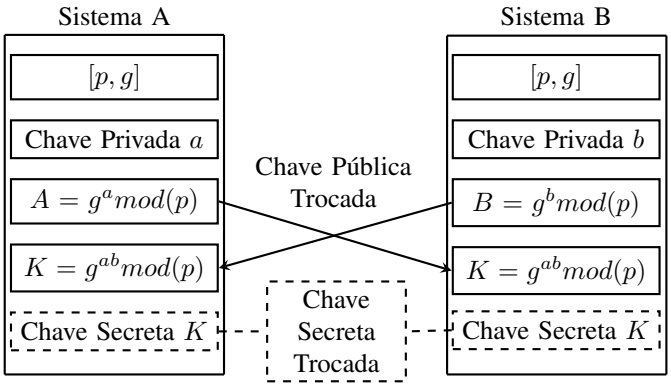

Fig. 7. Diffie-Hellman para troca de chaves em meio a canais inseguros.

A segurança das chaves criptográficas são de grande importância para a segurança da informação, uma vez que se a chave for extraviada, os dados protegidos poderão ser descifrados por qualquer sistema que tenha o conhecimento das chaves envolvidas na criptografia. Como a grande maioria dos algoritmos criptográficos são de código aberto, a única forma de garantir a segurança das informações é mantendo o sigilo das chaves.

Um dos ataques mais simples para encontrar uma chave criptográfica é o chamado "força bruta" (*brute force*). Este ataque se consiste em realizar uma busca exaustiva em todas as possibilidades de combinações de caracteres até que a correta (chave) seja descoberta. O método possui a vantagem de ser 100% eficaz, ou seja, há garantia de que, cedo ou tarde, a

chave secreta será descoberta, uma vez que ela pertence ao conjunto de todas as combinações possíveis. Porém, este tipo de ataque passa a ser computacionalmente inviável quando a chave em questão é longa, pois quanto maior a chave, maior o número de combinações possíveis [8]. Tal grandeza C cresce exponencialmente na razão $C = 2^n$ para chaves binárias, como é mostrado no gráfico da Figura 8.

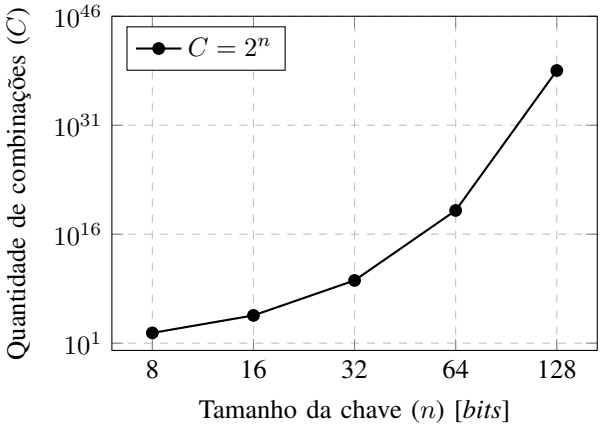

Fig. 8. Quantidade de combinações possíveis para chaves binárias.

Um processo de geração de chaves criptográficas aleatório de fato (e não pseudoaleatório) garantiria que nenhum sistema consiga reproduzir esta chave por meio da repetição de instruções, procedimentos e/ou regras predeterminados(as). Se a chave utilizar somente diretrizes matemáticas, passa a ser possível que sistemas externos utilizem estas mesmas diretrizes e, consequentemente, reproduzam as chaves localmente. Como a aleatoriedade perfeita é computacionalmente utópica, a pseudoaleatoriedade descrita pela RFC4086 é adotada neste tipo de sistema [11].

A pseudoaleatoriedade faz uso de parâmetros definidos externamente, que recebem o nome de entropia [12]. Consistem em informações que chegam a ser tão incertas que podem ser consideradas "aleatórias", ou seja, informações com alto grau de entropia são consideradas pseudoaleatórias. Esta entropia é normalmente capturada em ações realizadas pela sistema como um todo como, por exemplo as coordenadas de cliques do *mouse*, movimentos do dispositivo de disco, consultas a posições arbitrárias de memória, dentre outros eventos probabilísticos considerados imprevisíveis [13].

A entropia coletada do sistema é a base para a geração da maioria das chaves criptográficas. Tal fato implica em uma relação diretamente proporcional com o tamanho da chave e, consequentemente, com sua segurança, uma vez que para gerar chaves longas é necessário um grande volume de entropia. Tal necessidade pode ser um problema para sistemas reduzidos que não possuem variadas fontes de entropia como, por exemplo, sistemas embarcados, que normalmente não dispõem de uma diversidade de periféricos ou fontes de entropia em geral, tornando assim a geração de números pseudoaleatórios ineficiente.

D. Criptografia Neural

Criptografia neural é uma área da criptografia dedicada à utilizar redes neurais artificiais em procedimentos que envolvem geração e troca de chaves criptográficas. Em suas aplicações, são comumente utilizadas uma configuração específica de redes neurais artificiais chamada *Tree Parity Machine* (explicadas adiante na seção II-E).

A técnica de geração de chaves utiliza da sincronização mutua entre duas *Tree Parity Machines* (representada pela Figura 9) inicializadas com os mesmos parâmetros. No processo de treinamento das redes neurais artificiais, o vetor de entrada ($x_{(k,j)}$) utilizado em cada iteração é compartilhado e as saídas ($\tau^{A/B}$) são comparadas para decidir se a regra de aprendizado será aplicada ou não.

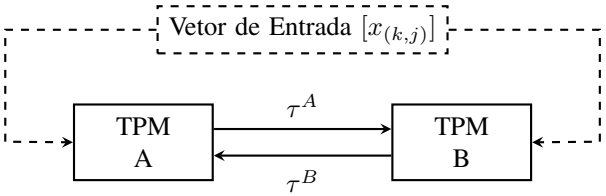

Fig. 9. Sincronização entre *Tree Parity Machines*.

As redes somente atingem a sincronização quando seus vetores de pesos sinápticos (explicadas adiante na seção II-E) se tornem idênticos. Isto é o critério de parada para o treinamento e indica que a chave foi gerada. A chave resultante é o próprio vetor de pesos sinápticos. Como ele está presente em ambas as redes, é possível instanciar cada rede em uma aplicação distinta, assim ao final do processo, ambas as redes possuirão a chave, caracterizando a troca.

E. Redes Neurais Artificiais

Redes neurais artificiais (RNA) são estruturas que têm como objetivo reproduzir computacionalmente o processamento realizado pelo cérebro humano. Uma RNA é uma agregação de unidades discretas de processamento baseadas em neurônios biológicos, intitulados neurônios artificiais [29].

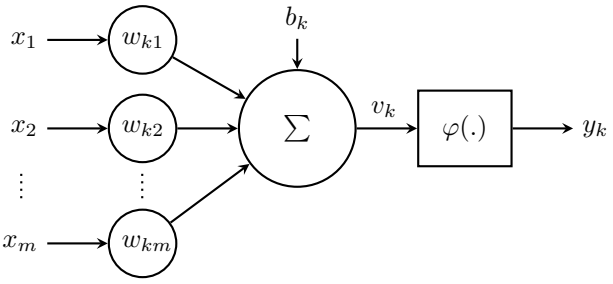

Fig. 10. Representação de um neurônio artificial.

Os neurônios artificiais, representados pela Figura 10, recebem um ou mais estímulos de entrada (x_m), sendo que cada um possui uma influência sobre o processamento conhecida como "peso sináptico" (w_{km}). Estas entradas ponderadas,

juntamente com um fator de correção fixo conhecido como viés ou limiar de ativação ($bias$) (b_k), que possui seu próprio peso sináptico, são combinados linearmente ($\sum$) a fim de gerar um valor denominado potencial de ativação (v_k). Este valor é utilizado como entrada da função de ativação (φ), que serve para limitar a saída (y_k) do neurônio associado.

A rede é submetida a conjuntos de entrada cujas saídas podem ser conhecidas ou não, este procedimento é conhecido como época de treinamento. Em caso afirmativo, o treinamento é dito supervisionado; caso contrário, é dito não-supervisionado. Após calcular as saídas, uma regra de aprendizado é aplicada para atualizar os pesos sinápticos até que um critério de parada seja satisfeito. A representação do conhecimento adquirido é dada, portanto, pelos valores finais dos pesos sinápticos da rede neural.

Uma RNA pode possuir múltiplas camadas; neste caso, ela é denominadas *multilayer*. Quando a saída dos neurônios artificias presentes nas camadas intermediárias das redes não realimentam neurônios das camadas anteriores, as redes são classificadas como "de alimentação adiante" (*feedforward*).

Inúmeras combinações com neurônios artificiais podem ser feitas, logo existem incontáveis configurações de redes neurais. A *Tree Parity Machine* (TPM) é um tipo característico de RNA *multilayer feedforward* frequentemente utilizado em aplicações que envolvem criptografia neural (vide seção II-D).

As TPMs possuem uma única saída ($\tau \in \{-1, +1\}$) e 3 camadas (*layers*) fixas: camada de entrada (*input layer*); camada escondida ou oculta (*hidden layer*) e camada de saída (*output layer*), que contém uma função multiplicadora. Na configuração da TPM são selecionados três parâmetros:

- K: número de neurônios na camada oculta ($1 \leq k \leq K$)
- N: número de entradas para cada neurônio ($1 \leq j \leq N$)
- L: configura a faixa de valores dos pesos sinápticos, na qual $w_{kj}(t) \in [-L, L]$

Em uma TPM tem-se $K \times N$ elementos gerados para compor o vetor de entrada, sendo que as entradas obedecem $x_{kj}(t) \in \{-1, +1\}$ [30]. A Figura 11 apresenta uma *Tree Parity Machine* parametrizada com $K = 3$ e $N = 4$.

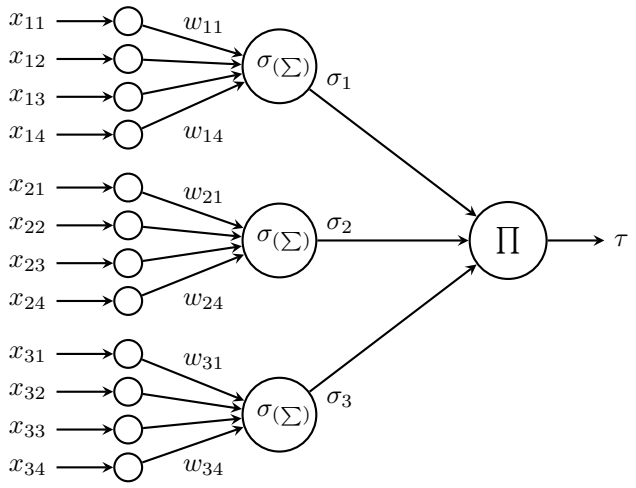

Fig. 11. *Tree Parity Machine* (TPM) parametrizada com *K* = 3 e *N* = 4.

III. Revisão da bibliografia

Estudos direcionados à área de geração de chaves criptográficas por meio de criptografia neural se tornaram recorrentes em publicações a partir dos anos 2000. Desde então, novas propostas utilizando os mais diversos recursos para melhorar o processo de geração de chaves por meio desta técnica vêm sendo propostas. Os trabalhos buscaram não só melhorar o desempenho do método, como também prover mais segurança em aplicações que envolvam troca de chaves em meios de comunicação não confiáveis.

Um ponto crucial ao processo de geração de chaves criptográficas é a fonte de entropia. Monrose et al. [31] e Chang et al. [32] propuseram o uso de características biométricas – captura de voz e captura facial – como fonte de entropia da pseudoaleatoriedade necessária, por serem consideradas únicas em seres vivos. Também foram consideradas o uso de impressões digitais, identificação de iris e geometria da mão. Tal abordagem pode tornar o procedimento de geração de chaves mais aleatório mas, ao mesmo tempo, estreitamente dependente de entradas externas.

Em 2007, Ruttor [33] propôs uma das mais conhecidas e utilizadas técnicas de criptografia neural: a sincronização entre duas redes neurais artificiais do tipo *Tree Parity Machines* baseando-se em seus aprendizados mútuos. Na proposta, as TPM eram inicializadas com pesos sinápticos aleatórios e compartilhavam suas saídas com o intuito de se sincronizarem. Vetores de entrada congêneres alimentavam ambas as redes e, caso suas saídas coincidissem, a regra de aprendizado era aplicada e seus pesos eram atualizados. As RNAs estavam sincronizadas quando seus vetores de pesos sinápticos tornaram-se análogos; tais vetores, após a sincronização, consistiram na chave criptográfica gerada pelo processo. O autor ainda fez um estudo profundo sobre a segurança entre troca de chaves utilizando a sincronização das redes, comparando os quatro principais tipos ataques à esta abordagem, sendo eles: ataque simples, geométrico, de maioria e genético.

Piazentin [34] realizou um estudo fortemente baseado na proposta de Ruttor [33] sobre parâmetros de sincronização entre duas *Tree Parity Machines*. Em seu trabalho foram analisadas as influências dos parâmetros L, K, N e da regra de aprendizado no processo de sincronização com relação à segurança do proposta. Constatou-se que a regra de aprendizado que se mostrou mais eficiente em sua aplicação foi a *Hebbian* [35]. Como o próprio autor afirma, o desempenho não pôde ser aferido de fato, uma vez que o trabalho fora realizado em Python, linguagem de programação com alto nível de abstração. Também não foi possível definir uma configuração ideal dos parâmetros da rede.

No trabalho de Revankar et al. [36] foi proposto um mecanismo de troca de chaves criptográficas utilizando criptografia neural, onde os vetores pseudoaleatórios de entrada são substituídos por consultas que dependem do estado atual da rede neural. Para isso, os autores utilizaram a regra descrita pela Equação (1), na qual $c_{k,\pm l}$ representa o número de produtos, sendo que o vetor de psos sinápticos ($w_{k,j}$) multiplicado pelo

vetor de entrada ($x_{k,j}$) resulta em um parâmetro l ($w_{k,j} \times x_{k,j} = l$). Os autores concluíram que a sincronização realizada com as consultas para formação dos vetores de entrada foi mais rápida, porém, a quantidade de informação trocada entre as *Tree Parity Machines* foi maior. Portanto, embora o tempo de sincronização seja menor, o tempo total da geração de chave pode ser maior do que a geração por meio de sincronização com entradas pseudoaleatórias.

$$h_k = \frac{1}{\sqrt{N}} \times \sum_{l=1}^{L} (c_{k,+l} - c_{k,-l}) \tag{1}$$

Com o intuito de aumentar a segurança no procedimento de troca de chaves utilizando criptografia neural, Allam e Abbas [37] propuseram uma técnica na qual transmissões com conteúdo falso são realizadas durante o processo de sincronização entre as *Tree Parity Machines* (aqui chamadas de A e B). Durante a sincronização, são enviadas mensagens errôneas – baseadas nas distâncias estimadas entre os pesos sinápticos das redes neurais – descritas pela Equação (2). Isto faz com que uma terceira TPM (C), que possa estar interceptando pacotes com trocas de mensagens entre A e B (como em um ataque *Man-in-the-Middle*[10], por exemplo), não consiga se sincronizar e, consequentemente, obter a chave que será gerada ao final do processo.

$$\left| h_k^{A/B} \right| = \left| \sum_{n=1}^{N} (w_{k,n}^{A/B} \times x_{k,n}^{A/B}) \right| \tag{2}$$

Baseado nas propostas revisadas anteriormente, este trabalho objetivou a otimização dos parâmetros das TPMs utilizadas pela criptografia neural no procedimento de geração de chaves criptográficas. Também, propôs uma implementação da técnica na linguagem de programação C[11] visando maior desempenho e tornando possível a comparação dos resultados obtidos pelas métricas do processo descrito na seção IV.

IV. Proposta

A proposta deste trabalho, descrita pelo diagrama da Figura 12, é implementar a estrutura da rede neural artificial *Tree Parity Machine* baseada na proposta de Ruttor [33] utilizando a linguagem de programação C de tal forma que duas TPMs distintas, porém com as mesmas configurações, se convirjam com o objetivo de gerar uma chave criptográfica a partir de suas sincronizações mutuas.

Este procedimento de criptografia neural será metrificado com a intenção de otimizar o processo a partir dos parâmetros requeridos.

Os blocos utilizados no diagrama da Figura 12 para modularizar a implementação estão descritos detalhadamente a seguir:

- Parametrização: Aquisição dos parâmetros necessários para instanciação das duas *Tree Parity Machines* (*K*, *N*, *L*, função de ativação e regra de aprendizado).
- Inicialização: Procedimento no qual ocorre a instanciação de duas TPMs com os mesmos parâmetros *K*, *N*, *L*, função de ativação e regra de aprendizado, porém com pesos sinápticos inicialmente distintos gerados pseudo-aleatoriamente.
- *Tree Parity Machine A* / *Tree Parity Machine B*: *Tree Parity Machines* devidamente instanciadas e inicializadas.
- Saída da TPM A (τ^A) / Saída da TPM B (τ^B): Saída das TPMs ($\tau^{A/B}$) representada pela Equação (3), onde $\sigma^{A/B}$ representa a saída dos *K* neurônios da camada escondida dada pelas suas funções de ativação, $w_{k,n}^{A/B}$ é o vetor de pesos sinápticos e $x_{k,n}^{A/B}$ representa o vetor de entrada.

$$\tau^{A/B} = \prod_{k=1}^{K} \sigma_k^{A/B} = \prod_{k=1}^{K} \sigma \left(\sum_{n=1}^{N} w_{k,n}^{A/B} \times x_{k,n}^{A/B} \right) \tag{3}$$

- Geração do vetor de entrada: Geração pseudo-aleatória de um vetor de $(K \times N) + K$ posições que será utilizado como entrada para ambas as redes neurais artificiais.
- Regra de Aprendizado: Aplicação da regra de aprendizado parametrizada para atualizar os pesos sinápticos para a próxima época de treinamento. As regras de aprendizado utilizadas estão descritas nas equações a seguir, sendo o aprendizado *Hebbian* representado pela Equação (4), *Anti-Hebbian* pela Equação (5) e *Random Walk* descrito pela Equação (6), nos quais a função $\Theta(x)$ é uma função do tipo sinal representada na Equação 7, $W_{k,n}^{A/B}$ é o novo peso sináptico atualizado com a regra de aprendizado, $w_{k,n}^{A/B}$ é o peso sináptico atual, $x_{k,n}$ é um elemento do vetor de entrada referente ao peso sináptico que será atualizado e $\tau^{A/B}$ representa a saída da rede.

$$W_{k,n}^{A/B} = w_{k,n}^{A/B} + x_{k,n} \tau^{A/B} \Theta(\tau^{A/B} \sigma_k^{A/B}) \Theta(\tau^A \tau^B) \tag{4}$$

$$W_{k,n}^{A/B} = w_{k,n}^{A/B} - x_{k,n} \sigma_k^{A/B} \Theta(\tau^{A/B} \sigma_k^{A/B}) \Theta(\tau^A \tau^B) \tag{5}$$

$$W_{k,n}^{A/B} = w_{k,n}^{A/B} + x_{k,n} \Theta(\tau^{A/B} \sigma_k^{A/B}) \Theta(\tau^A \tau^B) \tag{6}$$

$$\Theta(x) := \begin{cases} -1 & \text{if } x \leq 0 \\ +1 & \text{if } x > 0 \end{cases} \tag{7}$$

- Chave Gerada: Como os vetores de pesos sinápticos de ambas as redes se mostraram idênticos, significa que as redes convergiram. Logo a chave criptográfica gerada corresponde ao vetor de pesos sinápticos das *Tree Parity Machines* (excluindo o BIAS de cada neurônio) aplicados à função sinal ($\Theta(x)$).

[10] Ataque de interceptação de dados trocados entre duas partes.

[11] Linguagem de programação compilada de uso geral criada em 1972 por Dennis Ritchie.

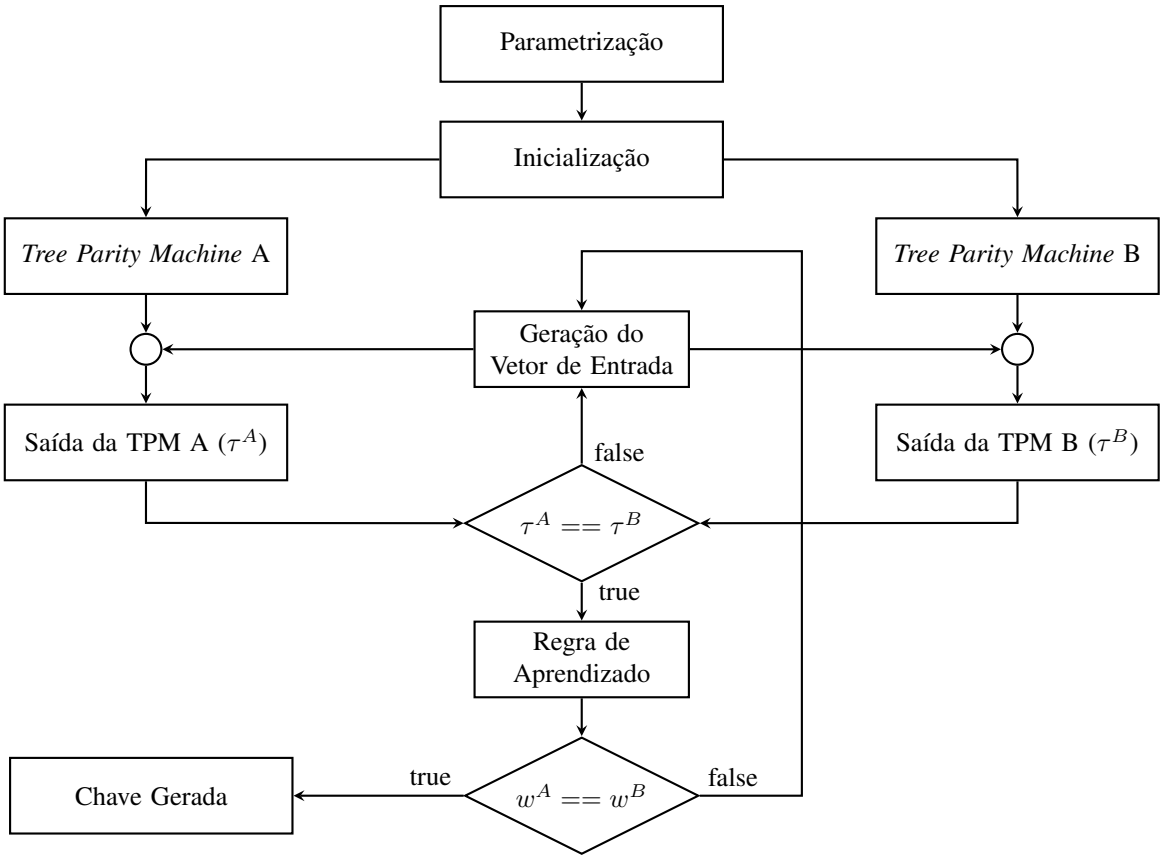

Fig. 12. Diagrama em blocos da proposta.

V. Experimentos

A. Metodologia

Com a finalidade de otimizar o processo de geração de chaves criptográficas por meio da criptografia neural, experimentos foram realizados com diferentes configurações de *Tree Parity Machines* a fim de analisar as influências individuais dos parâmetros de inicialização das redes neurais artificiais utilizadas no procedimento.

Os parâmetros analisados foram:

1) Variação de *K*: Número de neurônios artificiais na camada escondida.
2) Variação de *N*: Número de entradas para cada neurônio artificial da camada escondida.
3) Variação de *L*: Faixa de valores discretos possíveis para os pesos sinápticos.
4) Regra de Aprendizado:
   a) *Hebbian**
   b) *Anti-Hebbian**
   c) *Random-Walk**

   * Descrito detalhadamente na seção IV.
5) Função de Ativação: As funções de ativação utilizadas estão descritas nas equações a seguir, sendo elas:

a) Sinal:

$$\varphi(x) := \begin{cases} -1 & \text{if } x \leq 0 \\ +1 & \text{if } x > 0 \end{cases} \tag{8}$$

b) Sigmóide:

$$\varphi(x) = \frac{1}{1 + e^{-x}} \tag{9}$$

c) Linear:

$$\varphi(x) = x \tag{10}$$

d) Tangente Hiperbólica (tanh):

$$\varphi(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{11}$$

Para garantir consistência nos resultados obtidos e compatibilidade entre todos os experimentos realizados, o gerador de números pseudo-aleatórios teve sua semente de geração fixada em *962611861*. Além disso, cada configuração de *Tree Parity Machine* utilizada nos experimentos gerou 100 chaves binárias consecutivas de tamanho notavelmente superior quando comparadas às chaves utilizadas nos algoritmos citados na seção II-B com o objetivo de salientar os resultados e tornar suas visualizações mais inteligíveis.

As métricas finais coletadas nas gerações sucessivas de chaves foram obtidas por meio de uma média aritmética dos valores obtidos no processo de geração de cada uma das 100 chaves.

As métricas designadas para análise foram:

1) Chaves Repetidas: Número de chaves repetidas no conjunto gerado juntamente com a probabilidade de ocorrência de cada chave (P_k), descrita pela Equação (12).

$$P_k[\%] = \frac{1}{2^{(K \times N)}} \times 100 \tag{12}$$

2) Épocas de Treinamento: Números médio, máximo e mínimo de épocas de treinamento necessárias para que ocorra a sincronização entre as duas *Tree Parity Machines* envolvidas no processo de geração das chaves.
3) Volume de Dados: O volume de dados (V) trocados entre as duas RNAs (em *Bytes*) em função da média de épocas de treinamento (*epochs_avg*) e dos parâmetros da TPM, descrito pela Equação (13). Bastante relevante para analisar situações onde as duas *Tree Parity Machines* não se encontram no mesmo sistema, ou até mesmo na mesma aplicação.

$$V[Bytes] = \frac{epochs\_avg \times [(K \times N) + K + 2]}{8} \tag{13}$$

4) Comprimento do vetor de entrada: O número de elementos no vetor de entrada ($I_{[K,N}$) é o resultado da soma de todas as entradas de cada neurônio da camada escondida ($K \times N$) com o acréscimo de uma entrada para o BIAS de cada um dos neurônios (*K*).

$$I_{[K,N]} = (K \times N) + K \tag{14}$$

5) Tempo de Geração: Tempos médio, máximo e mínimo tal como o desvio padrão do processo de geração de cada uma das 100 chaves separadamente, desde a inicialização das *Tree Parity Machines* até o alcance da sincronização.
Para que haja coesão nas métricas temporais, todos os experimentos foram executados na mesma máquina cujo a configuração é:
   - Sistema Operacional: Xubuntu
   - Processador: Intel(R) Core(TM) i7-5500U
   - Clock Rate: 2.40 GHz
   - Arquitetura: x86_64
   - Núcleos: 4
   - Memória RAM: 16 GB

Para aferir a influência das 3 diferentes regras de aprendizado e das 4 diferentes funções de ativação, os parâmetros *K*, *N* e *L* foram fixados em, respectivamente, 32, 32 e 5. Logo cada uma das 100 chaves criptográficas binárias geradas possuem um tamanho imutável de 1Kib (1Kib $\equiv$ 1024 *bits*).

A rotina de ensaio das 12 possíveis combinações entre regras de aprendizado e funções de ativação que foram utilizados no experimento foi intitulada "Experimento 1" e é representada na Figura 13.

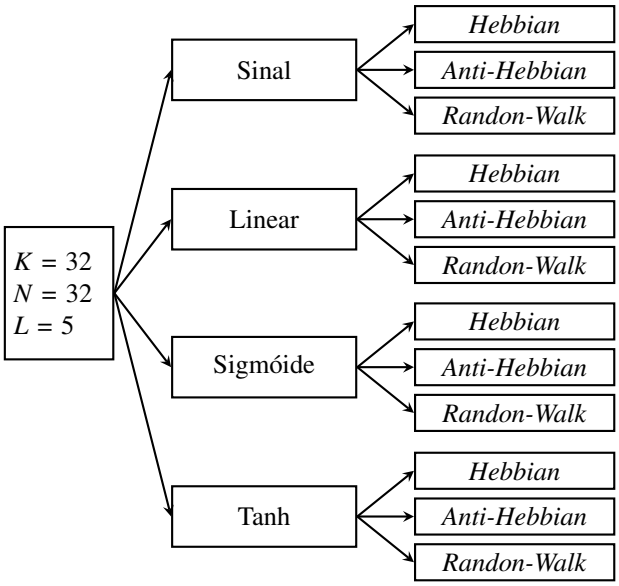

Fig. 13. Experimento 1.

Após a realização do Experimento 1, foram analisados os resultados e foram identificados, para esta implementação, os 3 conjuntos de regra de aprendizado e função de ativação que obtiveram maior desempenho relativo na geração de chaves (resultado discutido detalhadamente na seção V-B.1). Logo, nas rotinas implementadas para análise dos parâmetros *K*, *N* e *L*, as combinações tidas como mais eficientes foram utilizadas nas estruturas de análise dos mesmos.

Para as análises de *K* e *N*, o experimento realizado foi denominado "Experimento 2" e está representado na Figura 14. Nesta rotina, *L* foi fixado em 5 e um conjunto finito de pares de valores inteiros ($C_i[k_i, n_i]$) onde seus produtos resultam em 1Kib foi atribuído aos valores dos dois parâmetros.

$C_i[k_i, n_i]$	
k_i	n_i
1	1024
2	512
4	256
8	128
16	64
32	32
64	16
128	8
256	4
512	2
1024	1

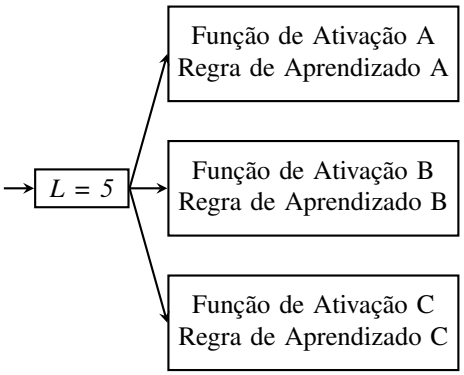

Fig. 14. Experimento 2.

O objetivo de rodar a criptografia neural utilizando o conjunto *C* é analisar a sensibilidade da implementação ao gerar chaves com exatamente o mesmo tamanho (1Kib) porém em configurações diversas. Com isso, espera-se concluir a relação

entre número de entradas e número de neurônios artificiais para gerar chaves equivalentes entre si.

Já na análise de L, o parâmetro foi variado entre 2 e 50 enquanto os demais foram fixados em seus valores ótimos. Esta sequencia de execuções, representada na Figura 15, foi denominada "Experimento 3".

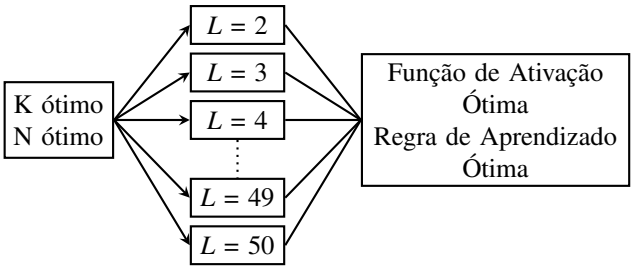

Fig. 15. Experimento 3.

Com os parâmetros individualmente analisados, é possível otimizar os parâmetros de geração das chaves a partir do tamanho de chave requerido.

Para gerar os parâmetros necessários na criptografia neural a partir do tamanho da chave desejada, foram utilizados os resultados dos 4 experimentos realizados que foram descritos detalhadamente nesta seção.

Para asseverar a eficiência do método, foram gerados dois grupos de chaves criptográficas binárias de 10Kib (tamanho 10 vezes maior do que o tamanho utilizado nos demais experimentos). Um grupo com 1.000 (mil) chaves e outro com 10.000 (dez mil) chaves. Este procedimento foi alcunhado "Experimento 4" e é discutido na seção V-B.4.

B. Resultados

Com o objetivo de facilitar o entendimento e a compreensão dos resultados dos experimentos executados, as discussões foram divididas como sendo uma subseção referente a cada um dos 4 experimentos realizados conforme a seguinte distribuição:

- Experimento 1 (Seção V-B.1)
- Experimento 2 (Seção V-B.2)
- Experimento 3 (Seção V-B.3)
- Experimento 4 (Seção V-B.4)

1) Experimento 1: Na análise dos resultados obtidos ao fim do Experimento 1 (Figura 13), 5 das 12 configurações possíveis entre regras de aprendizado e funções de ativação foram descartadas das avaliações. Dois motivos foram responsáveis pela rejeição, o primeiro foi o fato das *Tree Parity Machines* em duas das cinco configurações descartadas divergirem, logo não atingindo o sincronismo e consequentemente não gerando as chaves. O segundo foi que, em três configurações, embora as redes alcançassem o sincronismo e gerassem as chaves, todas as 100 chaves geradas eram iguais com todos os 1024 *bits* em 1 ou em 0, assim inviabilizando sua utilização. As configurações descartadas juntamente com o motivo de suas rejeições estão descritos na tabela da Figura 16.

Configurações descartadas nas análises		
Função de Ativação	Regra de Aprendizado	Motivo do Descarte
Sigmóide	*Random Walk*	Não convergiu
Tangente Hiperbólica	*Random Walk*	Não convergiu
Sigmóide	*Hebbian*	Chaves repetidas
Linear	*Hebbian*	Chaves repetidas
Tangente Hiperbólica	*Hebbian*	Chaves repetidas

Fig. 16. Configurações descartadas de análises pelo Experimento 1.

Levando em consideração as 7 configurações válidas, não houveram ocorrências de chaves repetidas em nenhum dos conjuntos de 100 chaves criptográficas binárias geradas por cada uma destas 7 configurações. A partir dos dados gerados nos conjuntos em questão foram analisadas as métricas propostas na seção V-A.

Para ilustrar os resultados, as métricas coletadas são dispostas na tabela da Figura 17 e visualmente representadas nos gráficos das Figuras 18, 19 e 20. Logo em seguida os resultados são comentados.

A abordagem utilizada para definir a combinação ótima entre regra de aprendizado e função de ativação foi atingir a maior eficiência possível no processo de geração. Não foram levados em consideração fatores como precauções de segurança em rede como por exemplo medidas de segurança contra ataques ao processo da criptografia neural onde um terceiro intercepta os dados trocados entre as duas redes neurais artificiais com o objetivo de sincronizar sua própria *Tree Parity Machine*. Estes temas são tratados na seção VI-A.

Analisando os resultados obtidos neste experimento e visando o melhor desempenho na geração da chave, pode-se afirmar que, para esta aplicação, as 3 combinações de função de ativação e regra de aprendizado que se mostraram mais eficiente foram: [sinal, *Anti-Hebbian*], [*linear*, *Anti-Hebbian*] e [sigmóide, *Anti-Hebbian*]. Isto se dá ao fato dos conjuntos utilizarem os menores números de épocas de treinamento para sincronizar. Isto faz com que menos dados sejam trocados entre as duas *Tree Parity Machines* e, consequentemente, seus tempos de execução foram menores do que as demais configurações, atingindo assim uma maior eficiência no processo.

Uma vez que todas as três combinações que obtiveram maior desempenho no processo de geração das chaves utilizam a regra de aprendizado *Anti-Hebbian*, pode-se concluir que esta é a regra considerada ótima para esta implementação.

2) Experimento 2: Após executadas as rotinas propostas, duas das três combinações de funções de ativação com regra de aprendizado foram descartadas das próximas análises. Os dois motivos responsáveis por esta ação foram os mesmos encontrados no Experimento 1, sendo eles, a divergência entre as redes neurais artificiais em determinadas configurações de K e N ou a geração das 100 chaves idênticas com os 1024 *bits* em 1 ou em 0. As configurações descartadas juntamente com o motivo de suas rejeições estão descritas na tabela da Figura 21.

Resultados do Experimento 1										
Função de Ativação	Regra de Aprendizado	Número de Épocas				Tempo [microsegundos]				Volume de Dados [*Bytes*]
		min	max	médio	σ	min	max	médio	σ	
Sinal	*Hebbian*	45	118	79	16.6	4007	12284	6352	1404.7	10448
	Anti-Hebbian	10	32	21	5	1085	4027	1765	423.9	2777
	Random Walk	45	178	86	22.2	3743	13932	6787	1745.2	11373
Linear	*Anti-Hebbian*	4	12	8	1.9	449	2969	876	501.6	1058
	Random Walk	30	95	55	14	3149	11533	5845	1600.6	7274
Sigmóide	*Anti-Hebbian*	9	26	16	3.9	1003	4453	1748	551.9	2116
Tangente Hiperbólica	*Anti-Hebbian*	32	141	67	21.5	3611	15981	7503	2434.7	8860

Fig. 17. Tabela de resultados do Experimento 1.

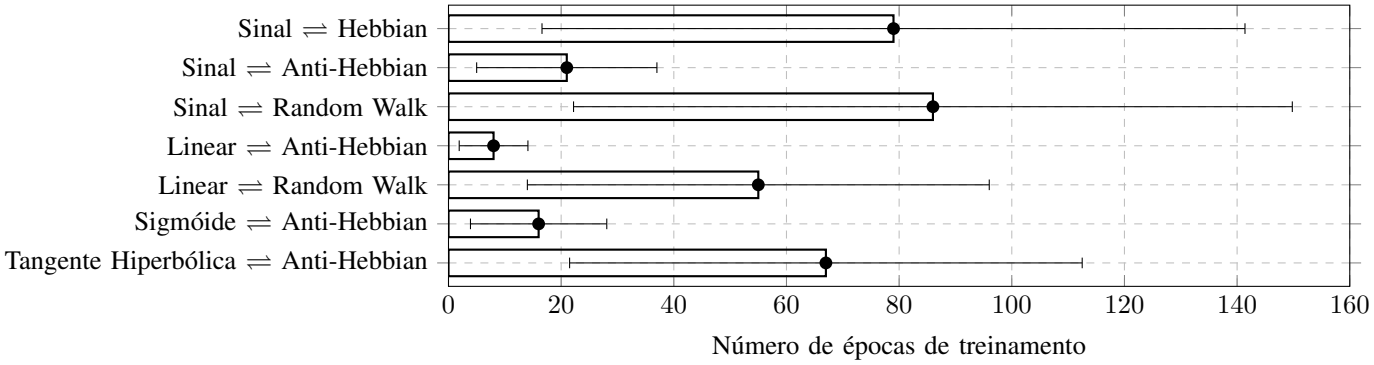

Fig. 18. Análise do número médio de épocas com desvio padrão nos resultados do Experimento 1.

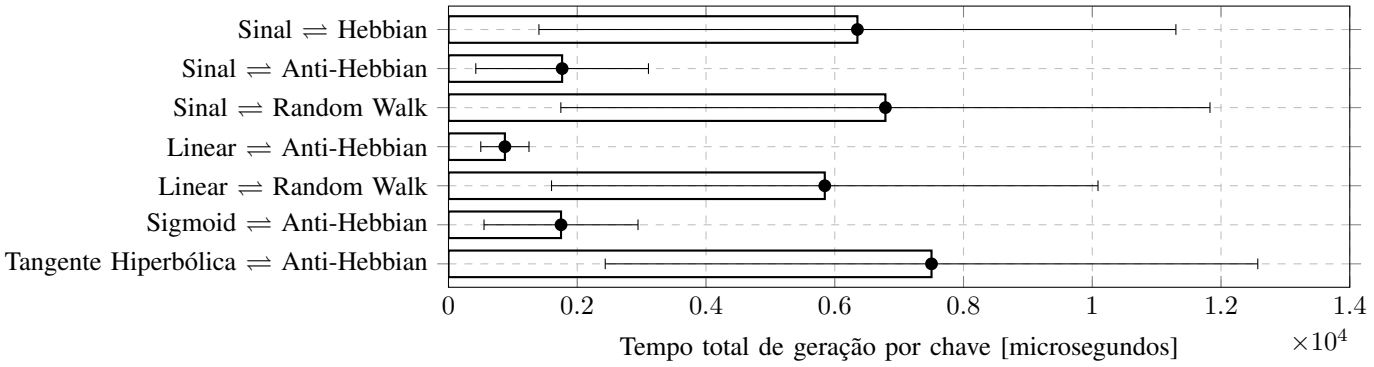

Fig. 19. Análise do tempo médio total de geração por chave com desvio padrão nos resultados do Experimento 1.

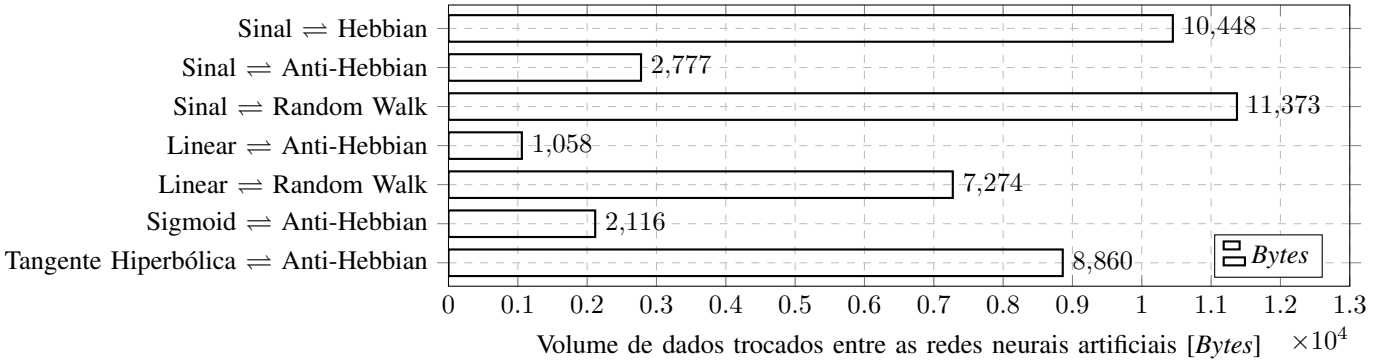

Fig. 20. Análise do volume médio de dados trocados na geração de cada chave com desvio padrão nos resultados do Experimento 1.

Configurações descartadas nas análises		
Função de Ativação	Regra de Aprendizado	Motivo do Descarte
Sigmóide	*Anti-Hebbian*	Não convergiu
Linear	*Anti-Hebbian*	Chaves repetidas

Fig. 21. Configurações descartadas de análises pelo Experimento 2.

A partir desta análise, foi constatado que o conjunto [sinal, *Anti-Hebbian*] se mostrou ótimo para esta implementação. Isto se deu ao fato do conjunto ser o mais rápido no que se refere ao tempo total de geração por chave binária, e ao mesmo tempo ser resiliente às 11 diferentes combinações de *K* e *N* presentes no conjunto *C* que resultam em chaves de 1Kib propostas pelo Experimento 2.

Para obter a proporção ótima entre os parâmetros *K* e *N*, a rotina do Experimento 2 foi aplicada utilizando a regra de aprendizado *Anti-Hebbian* e a função de ativação sinal. Os valores das métricas estão dispostos na tabela da Figura 22 e nos gráficos das Figuras 23, 24 e 25.

Resultados do Experimento 2						
$C_i[k_i, n_i]$		Épocas		Tempo [micro]		Volume de Dados [B]
k_i	n_i	médio	σ	médio	σ	
1	1024	15	3.8	1188	308.8	1925
2	512	16	3.1	1210	209.9	2056
4	256	18	3.9	1263	305.2	2317
8	128	18	3.6	1325	306.8	2455
16	64	19	4.5	1570	427.9	2605
32	32	21	5	1453	309	2645
64	16	21	5.2	1575	350.4	2861
128	8	23	5.4	1751	411	3317
256	4	22	5.9	2010	413	3685
512	2	23	5.4	2485	506.7	4421
1024	1	15	3	2431	440.9	3843

Fig. 22. Tabela de resultados do Experimento 2.

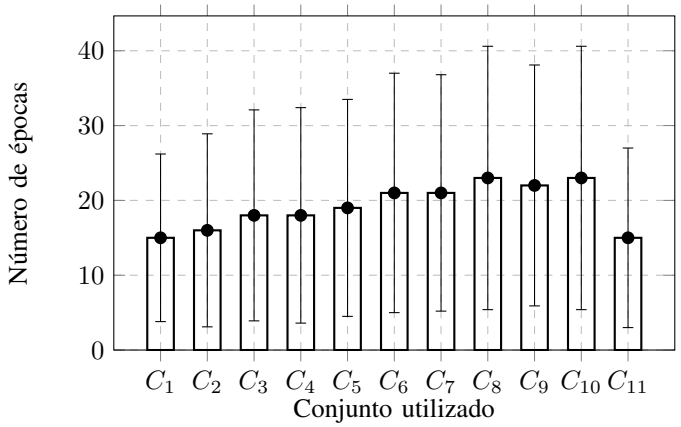

Fig. 23. Análise do número de épocas com desvio padrão.

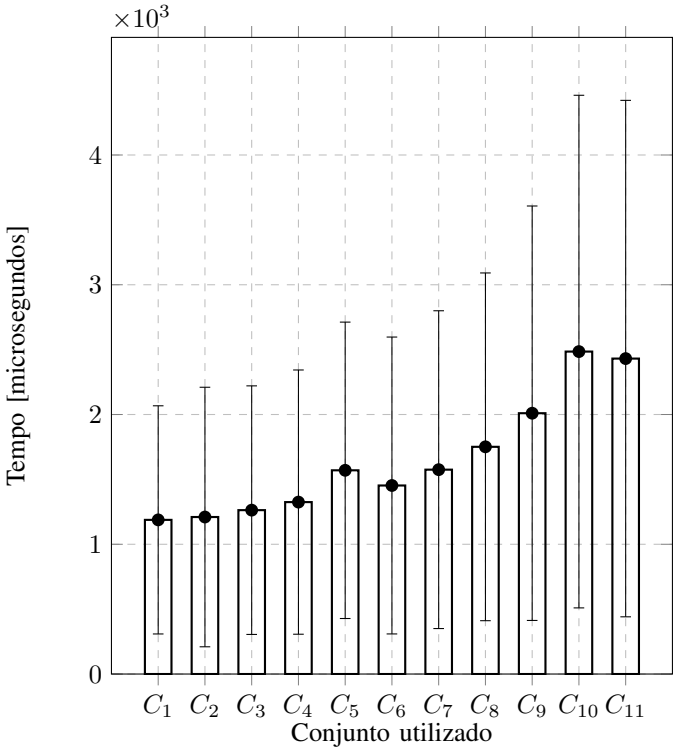

Fig. 24. Análise do tempo de geração com desvio padrão.

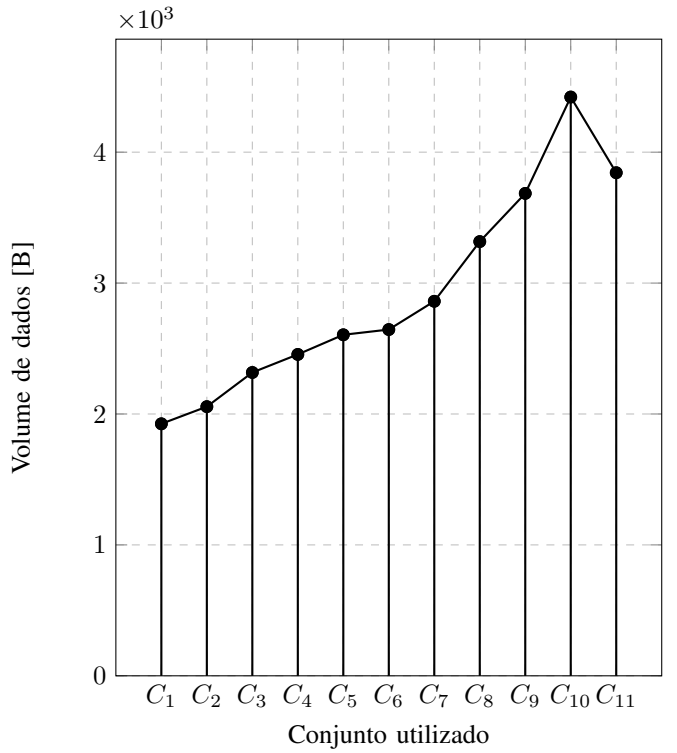

Fig. 25. Análise de volume de dados.

Analisando os resultados do Experimento 2, pode-se concluir que os valores de *K* e *N* que se mostraram mais eficientes foram $K = 1$ e $N = 1024$.

Levando em consideração o número de épocas para sincronizar, ambos os pares [1, 1024] e [1024, 1] obtiveram a mesma quantidade de iterações das redes.

Quando o tempo de geração individual das chaves e o volume de dados trocados entre as duas redes neurais artificiais são analisados, é possível constatar que quanto maior o número de neurônios na camada escondida, maior o tempo de geração e também do volume de dados trocados.

Este comportamento é explicado pela adição do BIAS em cada neurônio da camada escondida. Comparando os dois pares [*K*, *N*] é possível perceber a influência que o acréscimo do BIAS exerce nas gerações. Embora as chaves geradas possuam os mesmos tamanhos (1Kib), os vetores de entrada que são utilizados no treinamento das *Tree Parity Machines* têm seus tamanhos variados dependendo do número de neurônios na camada escondida uma vez que é necessário gerar uma entrada para cada BIAS além do número de entradas. A comparação dos cálculos é demonstrada pelas Equações (15) e (16).

$$I_{[1,1024]} = (1024 \times 1) + 1 = 1025 \tag{15}$$

$$I_{[1024,1]} = (1 \times 1024) + 1024 = 2048 \tag{16}$$

A mesma conduta é identificada no cálculo do volume de dados. Esta está representada nas Equações (17) e (18).

$$V_{[1,1024]} = \frac{15 \times [(1 \times 1024) + 1 + 2]}{8} = 1925[B] \tag{17}$$

$$V_{[1024,1]} = \frac{15 \times [(1024 \times 1) + 1024 + 2]}{8} = 3843[B] \tag{18}$$

Neste ponto pode-se concluir que uma *Tree Parity Machine* com um neurônio na camada escondida se torna mais eficiente no que se refere ao tempo de geração de chaves. Por conseguinte, o parâmetro *K* foi fixado em 1.

Como *K* foi fixado em 1 e o tamanho da chave gerada pela criptografia neural é obtido por meio do produto entre o número de neurônios na camada escondida com o número de entradas de cada neurônio (excluindo o BIAS). Tem-se que o valor de *N* será correspondente ao tamanho da chave a ser gerada.

3) Experimento 3: Na execução das iterações propostas no experimento, os valores de *L* foram variados de 2 à 50, enquanto os demais parâmetros foram fixados nos resultados ótimos aferidos pelos experimentos anteriores. São estes parâmetros: *K* = 1, *N* = 1024, função de ativação sinal e regra de aprendizado *Anti-Hebbian*.

Após a realização do ensaio, a influência do parâmetro *L* no processo de criptografia neural nas condições citadas anteriormente foi representado de forma discreta no gráfico da Figura 26.

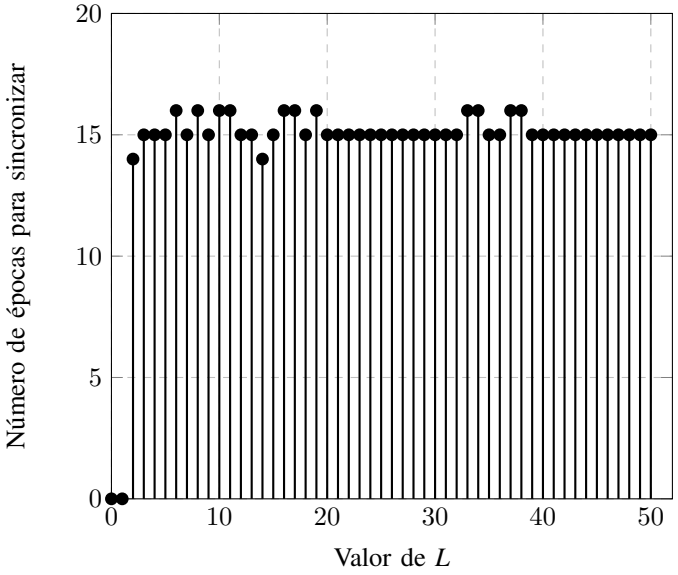

Fig. 26. Análise da influência de *L* nas condições alcançadas.

Com os resultados do Experimento 3 é possível concluir que o parâmetro *L* não interfere no desempenho do processo de geração de chaves criptográficas binárias para esta configuração de *Tree Parity Machine* alcançada. Tendo em vista esta situação, o parâmetro *L* foi fixado em 2, que é o menor valor tangível para o mesmo.

4) Experimento 4: Baseando-se nos experimentos anteriores, a estrutura que foi constituída ótima está representada na Figura 27. Onde a mesma possui a seguinte parametrização:

- *K*: 1
- *N*: Tamanho da chave desejada
- *L*: 2
- Função de Ativação: Sinal
- Regra de Aprendizado: *Anti-Hebbian*

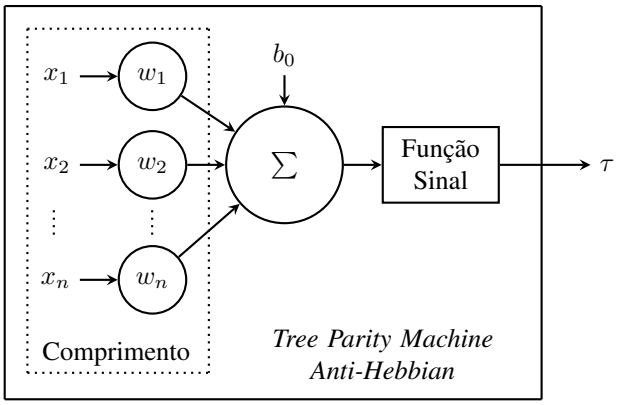

Fig. 27. Estrutura ótima.

Utilizando esta configuração ótima, os dois grupos de chaves propostos pelo Experimento 4 foram geradas, tal qual a coleta das métricas estudadas. Em seguida seus resultados foram gerados e estão representados pela tabela da Figura 28 e pelos gráficos das Figuras 29 e 30.

Resultados do Experimento 4											
Configuração Ótima	Número de Chaves	Número de Épocas				Tempo [microsegundos]				Volume de Dados [B]	Chaves Repetidas
		min	max	médio	σ	min	max	médio	σ		
K = 1 N = 10240 L = 2 Sinal *Anti-Hebbian*	1000	12	30	18	3	9991	23778	14080	1928.6	23046	0
	10000	11	42	18	3	9673	24824	14016	1865	23046	0

Fig. 28. Tabela de resultados do Experimento 4.

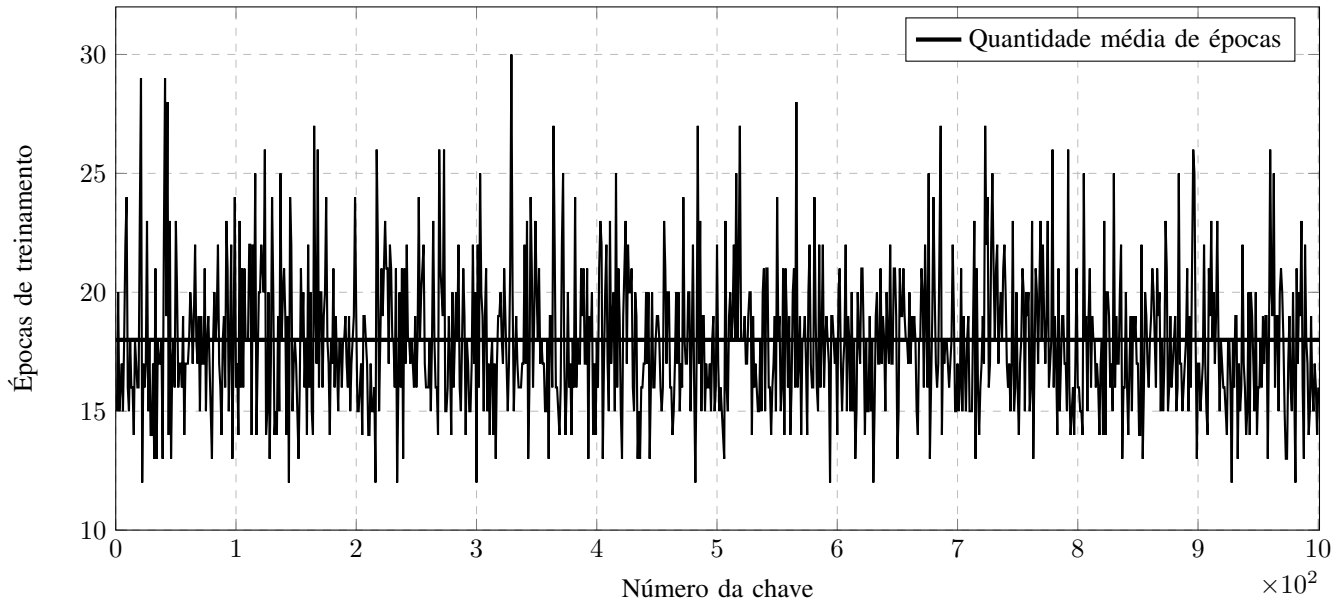

Fig. 29. Épocas de treinamento na geração das 1.000 chaves binárias.

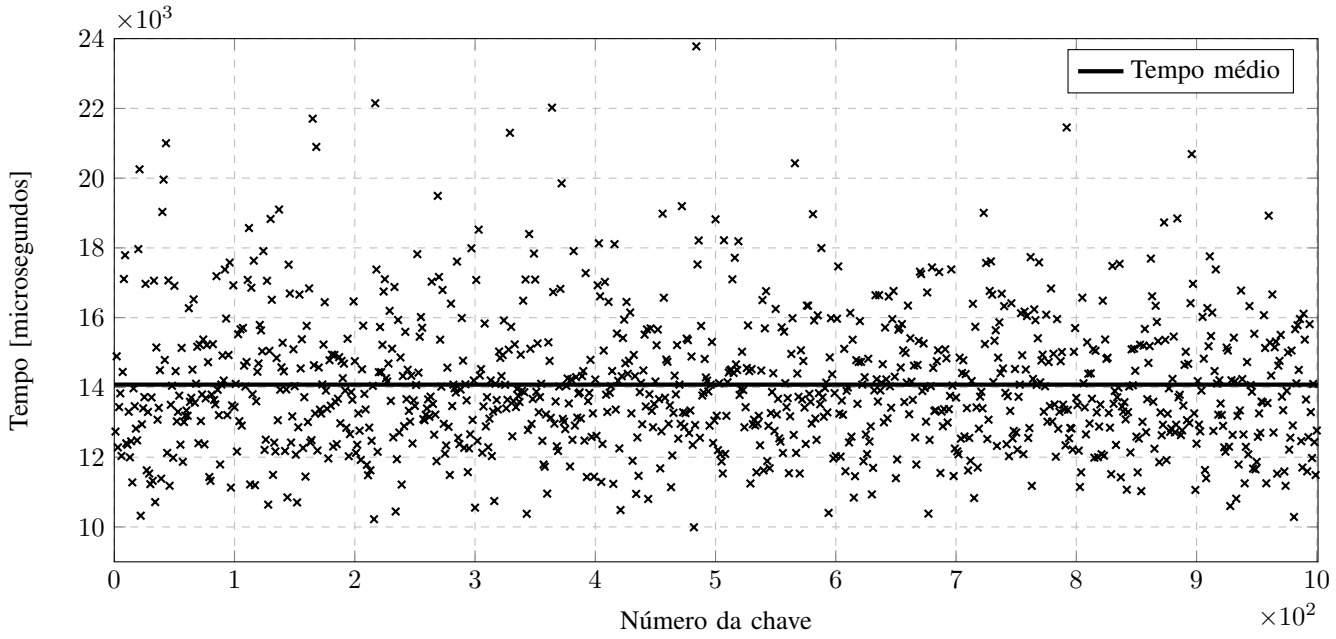

Fig. 30. Tempo por chave de geração das 1.000 chaves binárias.

VI. Conclusão

A partir da revolução industrial, o patrimônio intelectual vem sendo, cada vez mais, a propriedade mais valiosa no contexto empresarial. A ponto de que para a esmagadora maioria das instituições ativas, perder suas informações acarretaria em consequências calamitosas. Isto faz com que este bem tão precioso seja alvo dos afamados crimes cibernéticos, ataques que visam roubar ou inviabilizar estes dados.

Neste contexto surge a segurança da informação com a finalidade de proteger estes dados. A SI têm como uma das principais ferramentas, a criptografia, que possibilita a troca de informações entre partes por meio de canais inseguros de comunicação. Inúmeros algoritmos de criptografia foram e são propostos visando proteger as informações de formas cada vez mais eficientes. Como os algoritmos mais populares e utilizados são *open source*, é possível concluir que a segurança está na chave que é utilizada para encriptar os dados em suas utilizações.

Estas chaves são normalmente geradas com base na entropia (eventos probabilísticos considerados pseudo-aleatórios) coletada do sistema. A necessidade de uma fonte abundante de entropia pode ser um problema para sistemas reduzidos, como por exemplo em sistemas embarcados. Além da geração da chave, a troca da mesma entre serviços pode levar tempo e ser um ponto de ataque para que um terceiro obtenha uma cópia.

Baseando-se na proposta de Ruttor [33] que descreveu a troca e geração de chaves criptográficas binárias por meio de redes neurais artificiais do tipo *Tree Parity Machine*, foi realizada uma implementação do mecanismo na linguagem C com o intuito de aferir o seu real desempenho. A partir desta implementação, 4 experimentos foram elaborados e realizados para que uma configuração de *Tree Parity Machine* ideal seja alcançada.

Como resultado geral, uma configuração ótima foi adotada. O aferimento da validade desta configuração foi que na geração sequencial de dois grupos de chaves criptográficas binárias de 10Kib (tamanho hiperbólico quando comparado ao tamanho das chaves comumente utilizadas), obteve-se um tempo médio de geração por chave de 14080 microsegundos no grupo de mil chaves e de 14016 microsegundos para o grupo de dez mil chaves. Além disso não houve ocorrência de chaves repetidas em nenhum dos dois conjuntos.

A. Trabalhos Futuros

No decorrer do estudo, foram identificados pontos que condescendem possíveis extensões na pesquisa. Os pontos mais significativos no que se refere à trabalhos futuros estão descritos a seguir.

Um dos principais pontos a serem estudados seria a vulnerabilidade da estrutura alcançada aos ataques clássicos à este tipo de implementação. Em seguida, com os resultados obtidos, realizar uma criptoanálise e identificar falhas de segurança e, consequentemente, realizar possíveis melhorias na implementação atual para torná-la viável em ambientes inseguros. Os principais ataques são:

- Ataque simples [38]
- Ataque geométrico [39]
- Ataque de maioria [33]
- Ataque genético [38][39]

Outro ponto relevante a ser estudado seria implementar e analisar o desempenho de um mecanismo de realimentação (*feedback mechanism*) nas redes neurais artificiais, como proposto por Ruttor [33]. Isto acarretaria em uma entrada secreta para cada neurônio na camada escondida que não seria transmitido entre as *Tree Parity Machines*, aumentando a segurança do método proposto, diminuindo o tamanho dos vetores de entrada necessários e, consequentemente, diminuindo o volume de dados trocados entre as redes neurais artificiais.

Referências

[1] M. T. P. Antunes and E. Martins, "Capital intelectual: verdades e mitos," *Revista Contabilidade & Finanças*, vol. 13, no. 29, pp. 41–54, 2002.

[2] R. Tolotti Umpieres, "Corretora de bitcoin sul-coreana declara falência após ataque hacker," *Infomoney*, 2017. Disponível em <https://bit.ly/2PWLRG4>. Acessado em nov 2018.

[3] Redação, "52% Das Empresas Brasileiras Tiveram Seus Dados Roubados Por Malware," *Canal Tech*, 2014. Disponível em <https://bit.ly/2Pl7v7o>. Acessado em nov 2018.

[4] Redação, "Volume de roubo de dados cresce 88% em 2017," *Computer World*, 2017. Disponível em <https://bit.ly/2Dvv8Uc>. Acessado em nov 2018.

[5] E. T. Nakamura and P. L. Geus, *Segurança de redes em ambientes cooperativos*. Novatec Editora, 2007.

[6] Redação, "Uber admite que omitiu ataque hacker que roubou dados de 57 milhões de usuários em 2016," *G1 Globo*, 2017. Disponível em <https://glo.bo/2zrOBna>. Acessado em nov 2018.

[7] ISO/IEC, "Tecnologia da informação — Técnicas de segurança — Código de prática para controles de segurança da informação," *ABNT - Associação Brasileira de Normas Técnicas*, p. 99, 2013. Disponível em <https://bit.ly/2tXuIPd>. Acessado em nov 2018.

[8] D. R. Stinson, *Cryptography: Theory and Practice, Third Edition*. Chapman and Hall/CRC, 2002.

[9] Redação, "Certificado digital do banco Inter é revogado após chave vazar na web," *G1 Globo*, 2018. Disponível em <https://glo.bo/2K0saI1>. Acessado em nov 2018.

[10] S. S. Al-Riyami and K. G. Paterson, "Certificateless Public Key Cryptography," in *International Conference on the Theory and Application of Cryptology and Information Security*, pp. 452–473, Springer, 2003.

[11] D. Eastlake, J. Schiller, and S. Crocker, "Randomness Requirements for Security," *NTWG - Network Working Group*, 2005. Disponível em <https://bit.ly/2DvvLNy>. Acessado em nov 2018.

[12] T. Ristenpart and S. Yilek, "When Good Randomness Goes Bad: Virtual Machine Reset Vulnerabilities and Hedging Deployed Cryptography.," in *Ndss*, The Internet Society, 2010.

[13] A. L. Young and M. Yung, *Cryptography: Malicious Cryptography – Exposing Cryptovirology*, vol. 20. John Wiley & Sons, 2004.

[14] Adshead, Antony, "Analytics, internet of things to drive data volumes to 163ZB by 2025," *Computer Weekly*, 2017. Disponível em <https://bit.ly/2qE9lS1>. Acessado em nov 2018.

[15] R. Saint-Germain, "Information Security Management Best Practice Based on ISO/IEC 17799," *Information Management*, vol. 39, no. 4, pp. 60–66, 2005.

[16] D. E. Rob, *Cryptography and Data Security*. Addison-Wesley Longman Publishing Co., Inc., 1982.

[17] F. Cohen, "A short history of cryptography, 1995," *URL: http://web. itu. edu. tr/˜ orssi/dersler/cryptography/Chap2-1. pdf (2017-09-17)*, 1995. Disponível em <http://all.net/edu/curr/ip/Chap2-1.html>. Acessado em nov 2018.

[18] W. Stallings, *Cryptography and Network Security*, vol. 139. Pearson Education India, 2011.

[19] R. Terada, *Segurança de dados criptografia em rede de computador*. Edgard Blucher, 2008.

[20] J. Grabbe, "The DES algorithm illustrated," *Laissez Faire City Times*, vol. 2, no. 28, pp. 1–15, 1992.

[21] B. Masucci, "Data encryption standard," *Federal Information Processing Standards Publication*, 2003.

[22] J. Thakur and N. Kumar, "DES, AES and Blowfish: Symmetric key cryptography algorithms simulation based performance analysis," *International journal of emerging technology and advanced engineering*, vol. 1, no. 2, pp. 6–12, 2011.

[23] S. Lucks, "The saturation attack—a bait for Twofish," in *International Workshop on Fast Software Encryption*, pp. 1–15, Springer, 2001.

[24] C. Adams, "The cast-128 encryption algorithm," tech. rep., Network Working Group, 1997.

[25] C. Adams and J. Gilchrist, "The cast-256 encryption algorithm," tech. rep., Network Working Group, 1999.

[26] M. Riaz and H. Heys, "The FPGA implementation of the RC6 and CAST-256 encryption algorithms," in *Engineering Solutions for the Next Millennium. 1999 IEEE Canadian Conference on Electrical and Computer Engineering (Cat. No.99TH8411)*, vol. 1, pp. 367–372, IEEE, 1999.

[27] J. Daemen and V. Rijmen, *The design of AES- the Advanced Encryption Standard*. Springer Science & Business Media, 2002.

[28] E. Rescorla, "Diffie-Hellman Key Agreement Method," tech. rep., RFC2631, 1999.

[29] M. V. C. Aragão, "Estudo e avaliação de classificadores para um sistema anti-spam.," *UNIFEI*, 2018.

[30] M. Volkmer and S. Wallner, "Tree parity machine rekeying architectures," *IEEE Transactions on Computers*, vol. 54, no. 4, pp. 421–427, 2005.

[31] F. Monrose, M. K. Reiter, Q. Li, and S. Wetzel, "Cryptographic Key Generation from Voice," in *Proceedings of the {IEEE} Symposium on Research in Security and Privacy*, pp. 202–212, IEEE, 2001.

[32] Y.-j. Chang, W. Zhang, and T. Chen, "Biometrics-Based Cryptographic Key Generation," in *International Conference on Multimedia and Expo (ICME)*, vol. 3, pp. 2203–2206, IEEE, 2004.

[33] A. Ruttor, "Neural synchronization and cryptography," *arXiv preprint arXiv:0711.2411*, 2006.

[34] D. R. d. M. Piazentin, "Troca de chaves criptográficas utilizando criptografia neural," *UNIVEM*, 2011.

[35] D. O. Hebb, *The Organization of Behavior; A Neuropsychological Theory*. New York, New York, USA: Wiley & Sons, 1949.

[36] P. Revankar, W. Gandhare, and D. Rathod, "Private inputs to tree parity machine," *International Journal of Computer Theory and Engineering*, vol. 2, no. 4, p. 665, 2010.

[37] A. M. Allam and H. M. Abbas, "On the improvement of neural cryptography using erroneous transmitted information with error prediction," *IEEE Transactions on Neural Networks*, vol. 21, no. 12, pp. 1915–1924, 2010.

[38] A. Ruttor, W. Kinzel, R. Naeh, and I. Kanter, "Genetic attack on neural cryptography," *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, vol. 73, no. 3, p. 36121, 2006.

[39] A. Klimov, A. Mityagin, and A. Shamir, "Analysis of Neural Cryptography," in *International Conference on the Theory and Application of Cryptology and Information Security*, pp. 288–298, Springer, 2002.