Redes Neurais Artificiais Aplicadas à Geração e Troca de Chaves Criptográficas Binárias

Autor[1]

[1] Instituição
Endereço

email@domínio.com

Abstract. *With cryptography, 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.*

Resumo. *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.*

1. Introdução

O uso da criptografia, uma das principais ferramentas da Segurança da Informação, possibilita serviços a trocarem informações entre si por meio de um canal inseguro sem que terceiros entendam as mensagens trocadas, mesmo se interceptadas [Stinson 2002].

Para que os algoritmos criptográficos possam proteger uma mensagem legível, é necessária uma chave. Seu gerenciamento, estrutura de geração, autenticação e transmissão são considerados pontos críticos no desenvolvimento de sistemas seguros que envolvem criptografia [Al-Riyami and Paterson 2003].

2. Revisão da teoria

2.1. 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 2.2).

A técnica de geração de chaves utiliza da sincronização mutua entre duas *Tree Parity Machines* inicializadas com os mesmos parâmetros. No processo de treinamento das redes neurais artificiais, o vetor de entrada utilizado em cada iteração é compartilhado e as saídas são comparadas para decidir se a regra de aprendizado será aplicada ou não.

As redes somente atingem a sincronização quando seus vetores de pesos sinápticos se tornam 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.

2.2. 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 [Aragão 2018].

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* (mais de uma camada de neurônios) *feedforward* (saídas sempre alimentam a próxima camada) frequentemente utilizado em aplicações que envolvem criptografia neural (vide seção 2.1).

As TPMs possuem uma única saída ($\tau \in \{-1, +1\}$) e 3 camadas (*layers*) fixas: camada de entrada (*input layer*); camada oculta ou escondida (*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 escondida ($1 \leq k \leq K$)
- 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 $x_{kj}(t) \in \{-1, +1\}$ [Volkmer and Wallner 2005].

3. Revisão da bibliografia

Em 2006, Ruttor [Ruttor 2006] 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 congêneres aleatórios 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.

4. Proposta

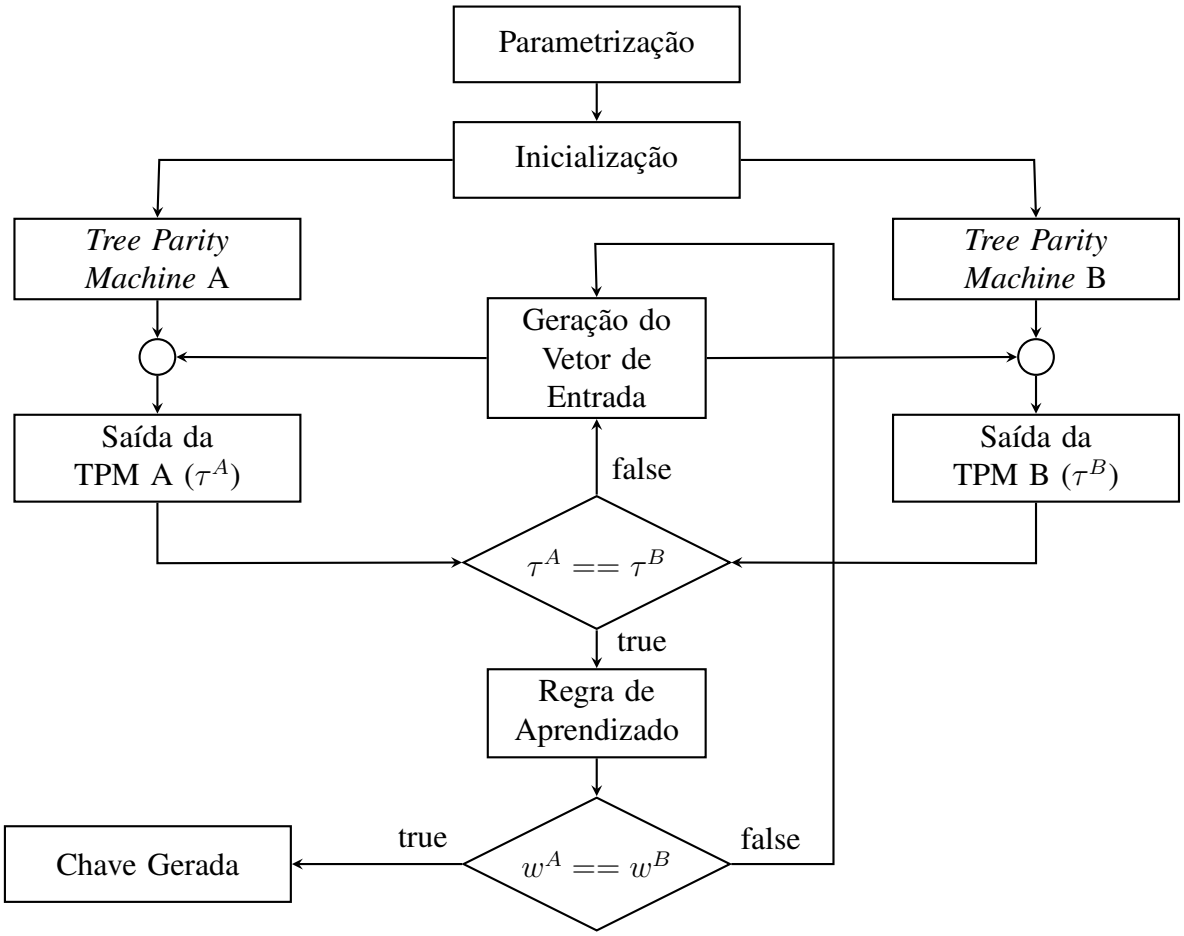

Figura 1. Diagrama em blocos da proposta.

A proposta deste trabalho, descrita pelo diagrama da Figura 1, é implementar a criptografia neural baseada na proposta de Ruttor [Ruttor 2006] 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.

5. Metodologia

Para garantir consistência nos resultados obtidos e compatibilidade entre todos os quatro 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 criptográficos comumente utilizados com o objetivo de salientar os resultados e tornar suas visualizações mais inteligíveis.

No primeiro experimento os valores de K, N e L foram fixados em 32, 32 e 5, respectivamente. As 12 possíveis combinações entre regras de aprendizado (*Hebbian*,

Anti-Hebbian e *Random-Walk*) e funções de ativação (Sinal, Linear, Sigmóide e Tanh) foram comparadas.

Para as análises de K e N, foi realizado o segundo experimento. Nesta rotina, L foi fixado em 5 e um conjunto finito contendo 12 pares de valores inteiros onde seus produtos resultam em 1Kib foi atribuído aos valores dos dois parâmetros. Para a função de ativação e regra de aprendizado, foram utilizados os três conjuntos que se mostraram mais eficientes no experimento 1.

No terceiro experimento foi analisado o parâmetro L. O mesmo foi variado entre 2 e 50 enquanto os demais fparâmetros oram fixados em seus valores ótimos alcançados.

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 5.1.4.

5.1. Resultados

5.1.1. Experimento 1

Na análise dos resultados obtidos ao fim do Experimento 1, 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 1.024 *bits* em 1 ou em 0, assim inviabilizando sua utilização.

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 com relação a prevenções de ataques ao processo.

Analisando os resultados obtidos neste experimento, pôde-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*]. 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.

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.

5.1.2. Experimento 2

Após a execução, 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.

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.

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. 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.

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.

5.1.3. Experimento 3

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.

5.1.4. Experimento 4

Baseando-se nos experimentos anteriores, a estrutura que foi constituída ótima possui a seguinte parametrização:

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

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 1 e pelo gráfico da Figura 2.

Tabela 1. Tabela de resultados do Experimento 4.

Configuração Ótima	Número de Chaves	Número de Épocas		Tempo [microsec]		Volume de Dados [B]
		médio	σ	médio	σ	
$K = 1$ $N = 10.240$ $L = 2$ Sinal *Anti-Hebbian*	1.000 10.000	18 18	3 3	14.080 14.016	1.928,6 1.865	23.046 23.046

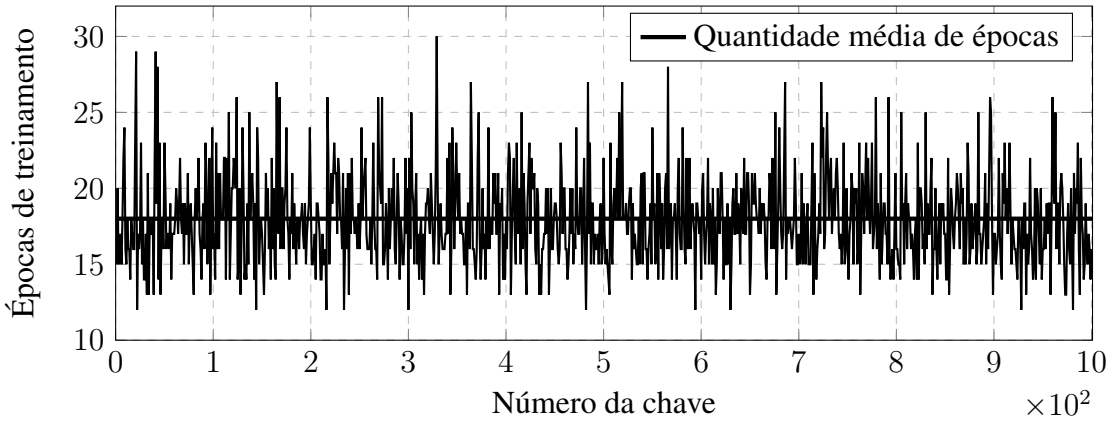

Figura 2. Épocas de treinamento na geração das 1.000 chaves binárias.

6. Conclusão

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 distintas de 10Kib (tamanho hiperbólico quando comparado ao tamanho das chaves comumente utilizadas), obteve-se um tempo médio de geração por chave de 14 milisegundos.

Referências

Al-Riyami, S. S. and Paterson, K. G. (2003). Certificateless Public Key Cryptography. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 452–473. Springer.

Aragão, M. V. C. (2018). Estudo e avaliação de classificadores para um sistema anti-spam. *UNIFEI*.

Ruttor, A. (2006). Neural synchronization and cryptography. *arXiv preprint arXiv:0711.2411*.

Stinson, D. R. (2002). *Cryptography: Theory and Practice, Third Edition*. Chapman and Hall/CRC.

Volkmer, M. and Wallner, S. (2005). Tree parity machine rekeying architectures. *IEEE Transactions on Computers*, 54(4):421–427.