Anais do IX Congresso Brasileiro de Redes Neurais / Inteligência Computacional (IX CBRN)
Ouro Preto 25-28 de Outubro de 2009 

# REDE NEURAL ARTIFICIAL APLICADO A GERENCIAMENTO DE CHAVES COM AUTENTICAÇÃO EM SISTEMAS RFID

Macêdo Firmino*, Gláucio B. Brandão*, Ana Maria G. Guerreiro*

*Departamento de Engenharia de Computação e Automação
Universidade Federal do Rio Grande do Norte
Natal - RN, Brasil

Emails: macedofirmino@dca.ufrn.br, glaucio@dca.ufrn.br, anamaria@dca.ufrn.br

**Abstract**— RFID (Radio Frequency Identification) identifies object by using the radio frequency which is a non-contact automatic identification technique. This technology has shown its powerful practical value and potential in the field of manufacturing, retailing, logistics and hospital automation. However, the main problem that impacts the application of RFID system is the key management. Recently, Kinzel and Kanter (Kinzel and Kanter, 2002) showed that two neural networks with the common input vector can finally synchronize their weights vector by output-based mutual learning. This paper presents an approach to solve the key management problems in RFID systems by designing a new key management protocol with mutual authentication. This protocol is an extension of the protocol proposed by Kinzel and Kanter, being added to the original protocol: mutual authentication and the establishment of frames.

**Keywords**— RFID Security and Privacy, Authentication and Key Management Protocols.

**Resumo**— Identificação por rádio freqüência, também chamado de RFID, representa uma tecnologia de identificação que utiliza ondas eletromagnéticas para a transmissão dos dados. Esta tecnologia está sendo cada vez mais utilizada nas indústrias, hospitais e logística. No entanto, o principal obstáculo para a utilização de sistemas RFID é a segurança da informação, especialmente o gerenciamento de chaves criptográficas. Recentemente, Kinzel e Kanter (Kinzel and Kanter, 2002) demonstraram que duas redes neurais podem aprender mutualmente, gerando ao final do processo de aprendizado pesos sinápticos iguais. Este artigo propõe um novo protocolo de gerenciamento de chaves com autenticação mútua. Este protocolo é uma extensão do protocolo proposto por Kinzel e Kanter, sendo adicionada ao protocolo original: a autenticação mútua e a definição de quadros.

**Palavras-chave**— Sistemas RFID, Autenticação e Gerenciamento de Chaves.

## 1 Introdução

RFID representa uma tecnologia de armazenamento, leitura, gravação e manipulação de dados de forma remota, através da comunicação de dados por ondas eletromagnéticas. Os sistemas RFID são compostos por dispositivos eletrônicos formados, basicamente, por um circuito integrado (para demodulação e modulação do sinal eletromagnético, armazenamento e processamento das informações) e uma antena para recepção e transmissão do sinal. Os dispositivos eletrônicos são chamados de *tag* e leitor. A *tag* é o componente responsável pelo armazenamento dos dados de identificação. O leitor tem a finalidade de obter os dados das *tags* e disponibilizá-los em uma interface gráfica ou sistemas de processamento de dados.

Duas organizações internacionais estão propondo padrões e estudos para a tecnologia RFID, são elas: ISO (*International Organization for Standardization*) e EPCGlobal. Entretanto, os padrões internacionais só trazem especificações de códigos de integridade. Além disso, a capacidade computacional, o espaço de armazenamento e o fornecimento de energia em alguns componentes RFID são muito limitados, resultando em restrições na concepção de mecanismos de segurança. Estes fatos acarretaram no surgimento de sistemas com vulnerabilidades de segurança (Juels, 2006).

Recentemente, têm surgido várias pesquisas relacionadas à questão de segurança da informação em sistemas RFID. Algumas destas pesquisas apresentaram algoritmos de criptografia com chave simétrica para a tecnologia (Juels, 2005) (Feldhofer, 2004). Entretanto, o protocolo de gerenciamento de chaves ainda continua sendo um problema. Em cifradores com chaves simétricas cada par leitor/*tag* deverá possuir uma chave secreta diferente. Então, o leitor necessitará armazenar todas as chaves para se comunicar com todas as *tags*.

Além disso, o fato da chave ser específica da *tag* leva a um paradoxo. A *tag* deverá se identificar para o leitor, para o mesmo obter a respectiva chave de criptografia e poder autenticar a *tag*. A privacidade neste caso não é alcançada, porque um atacante pode obter este identificador e usá-lo em algum tipo de ataque. Por outro lado, se o leitor não identificar a *tag*, como o leitor irá descobrir a chave de criptografia? Outro problema seria o gerenciamento das chaves quando fossem utilizadas chaves dinâmicas. Por exemplo, se fosse modificada a chave de criptografia de uma *tag*, a chave deverá ser divulgada a todos os leitores responsáveis pela obtenção dos dados da *tag*.

Várias pesquisas têm apresentado soluções para o problema de gerenciamento de chaves em sistemas RFID. Castelluccia e Avoine (Avoine and Castel-

luccia, 2006) propuseram a utilização da *noisy tag* em um protocolo de gerenciamento de chaves. A desvantagem desta solução é que ataque de *replay* e rastreamento clandestino não serão evitados se os pseudo-ruídos forem constantes. Por outro lado, na utilização de pseudo-ruídos dinâmicos o gerenciamento do sistema se torna complexo.

Jeng *et al.* (Jeng et al., 2008) propuseram um protocolo de distribuição de chaves baseado em *generic binary tree* para o padrão EPC (*Electronic Product Code*). A *tag* e o leitor trocam índices para uma estrutura de dados que contém chaves secretas. O problema desta abordagem é a dificuldade de gerenciamento quando utilizada com chaves dinâmicas. Lei *et al.* (Lei et al., 2007) propuseram um protocolo de autenticação com distribuição de chaves que utiliza: identificadores de autenticação para cada *tag*, chaves compartilhadas, função XOR e *Hash*. A desvantagem desta técnica é que não apresenta chaves dinâmicas e se faz necessário que os leitores possuam uma grande capacidade de armazenamento.

Kinzel e Kanter (Kinzel and Kanter, 2002) mostraram que duas redes neurais artificiais poderiam ser treinadas mutuamente até que os seus pesos se tornem idênticos. Devido a esta capacidade de sincronização das redes neurais, as mesmas podem ser utilizadas para a construção de protocolos de troca de chaves criptográficas. Uma rede neural artificial é uma estrutura de processamento constituído de neurônios, que têm como função armazenar conhecimentos adquiridos através de um processo de aprendizado (Haykin, 1998).

## 2 *Tree Parity Machine*

A *tree parity machine* corresponde a uma arquitetura para rede neural composta por duas camadas, conforme mostrada na Figura 1. A camada escondida é formada por $k$ neurônios e a camada de saída por apenas um neurônio. Cada neurônio da camada escondida possui $n$ valores de entrada, sendo cada entrada associada a um peso.

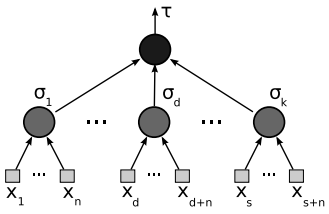


Figura 1: A arquitetura *tree parity machine*.

Os possíveis valores da entrada da rede são: $x_i \in \{-1, 1\}$. Os pesos deverão apresentar valores discretos entre o intervalo de $-l$ a $l$, ou seja, $w_{i,j} \in \{-l, -l+1, ..., l\}$ onde o índice $i = 1, ..., k$ representa o neurônio da camada escondida e o $j = 1, ..., n$ indica o elemento do vetor de entrada.

A saída do neurônio da camada escondida é dada pela equação:

$$\sigma_i = sgn\left(\sum_{j=1}^{n} w_{i,j} x_j\right). \quad (1)$$

A saída da rede neural é obtida pela multiplicação das saídas dos neurônios da camada escondida:

$$\tau = \prod_{i=1}^{k} \sigma_i. \quad (2)$$

O algoritmo de aprendizado é baseado na competição entre forças aleatórias atrativas e repulsivas. Por exemplo, considere duas redes neurais A e B recebendo a mesma entrada ($X$) e obtendo a mesma saída ($\tau$). Os pesos dos neurônios ($W_i$) que apresentarem a saída ($\sigma_i$) igual à saída da rede ($\tau$) serão atualizados. Portanto, dois passos são possíveis (Ruttor, 2007):

- Se $\tau^A = \sigma_i^A = \sigma_i^B = \tau^B$, chamado de passo atrativo, os pesos dos neurônios da camada escondida ($W^{A/B}$) serão atualizados. Como os pesos ($W^A$ e $W^B$) estão no intervalo entre $-l$ e $l$, a distância $d_{i,j} = |W^A - W^B|$ não será modificada. Porém se um dos pesos ultrapassarem o valor limite, será atribuido ao mesmo o valor $\pm l$, fazendo com que a distância diminua em uma unidade, até que $d_{i,j} = 0$;
- Se $\tau^A = \tau^B$, mas $\sigma_i^A \neq \sigma_i^B$, chamado de passo repulsivo, somente um dos pesos da camada escondida será atualizado. Até mesmo pesos já sincronizados poderão perder a sincronização. O passo repulsivo aumenta a distância relativa dos pesos, impedindo o processo de sincronização.

O processo de sincronização só é possível se a quantidade de passos atrativos for maior do que os passos repulsivos. Para um atacante, que escuta as informações entre os participantes, a probabilidade de passos repulsivos será sempre maior (Ruttor, 2007).

Duas redes iniciam o processo de sincronização escolhendo aleatoriamente os seus pesos. Em cada instante de tempo as redes recebem um vetor de entrada $X$ comum, calcula as suas saídas ($\tau_A$ e $\tau_B$) e informa a outra rede o valor obtido.

Se as saídas forem diferentes ($\tau_A \neq \tau_B$) os pesos não deverão ser ajustados. Caso contrário deverá ser utilizado à regra de aprendizado:

$$w_{i,j} = g(w_{i,j} + x_j), \tag{3}$$

onde:

$$g(\zeta) = \begin{cases} sgn(\zeta)l & \text{se } |\zeta| > l \\ \zeta & \text{caso contrário.} \end{cases} \tag{4}$$

Os pesos de um determinado neurônio da camada escondida só serão ajustados se sua saída ($\sigma_i$) for igual à saída da rede ($\tau$). Esta restrição se faz necessária para impedir que um atacante descubra quais pesos foram atualizados, visto que o mesmo não tem conhecimento da estrutura interna da rede.

A função $g(\zeta)$ possui o objetivo de garantir que os valores dos pesos se mantenham na faixa $[-l, l]$, ou seja, se o valor de um determinado peso em módulo for maior que o módulo de $l$, deverá ser atribuído ao mesmo o valor limite ($\pm l$).

A arquitetura *tree parity machine* apresenta algumas peculiaridades, por exemplo, para $k \leq 3$, o tempo de sincronização é diretamente proporcional a $l^2$, enquanto que o tempo de sincronização de um atacante é proporcional a $e^l$. Outra característica é que para $k \leq 2$ o sistema é vulnerável a um ataque chamado de *geometric attack* (Ruttor, 2007). Conseqüentemente poderá atribuir o nível de segurança desejável através da escolha do parâmetro $l$, com $k = 3$. O sistema é seguro se $l \to \infty$. Na prática é atribuído um valor suficientemente grande para garantir o nível de segurança desejado.

## 3 Protocolo de Gerenciamento de Chaves com Autenticação Mútua

Este artigo propõe um protocolo de gerenciamento de chaves com autenticação mútua. Este protocolo visa garantir a segurança da informação em sistemas RFID que apresenta os seguintes requerimentos: acesso as *tags* se mantém restrito a leitores autorizados, leitor e *tag* com mesmo gerador de números aleatórios baseados em sementes, *tags* com memória de leitura e escrita e a utilização de algum algoritmo de criptografia de chave simétrica. O algoritmo de criptografia pode ser, por exemplo, o RC4, 3-DES (*Triple-Data Encryption Standard*) ou AES (*Advanced Encryption Standard*).

O protocolo proposto poderá ser utilizado em conjunto com outros protocolos (tais como, acesso ao meio, interfaces físicas, código de integridade dos dados e formato dos dados) descritos nos padrões da EPCGlobal ou ISO.

A *tree parity machine* utilizada apresenta: $k = 3$, $n = 32$ e $l = 127$. O que resulta na geração de pesos com 8 *bits*, onde o *bit* mais significativo (MSB - *Most Significant Bit*) representa o sinal:

$$MSB = \begin{cases} 1 & \text{se } w_{i,j} < 1 \\ 0 & \text{caso contrário,} \end{cases} \tag{5}$$

e os outros 7 *bits* restantes representando o valor do peso em módulo. A rede neural cria ao final do processo de sincronização um conjunto de pesos de 768 *bits*, ou seja, 6 grupos de 128 *bits*.

O protocolo proposto é formado por duas etapas: geração da chave de criptografia e autenticação das entidades. Na etapa de geração das chaves, o leitor e a *tag* deverão utilizar uma *tree parity machine* com a mesma estrutura. Os parâmetros $k$, $l$ e $n$ são públicos. O processo de geração de chaves começa com a atribuição de valores aleatórios aos pesos das redes neurais. Nesta abordagem a *tag* e o leitor deverão trocar informações relacionadas a entrada e saída das redes neurais para ajustar os seus pesos até obterem a sincronização, observe a Figura 2.

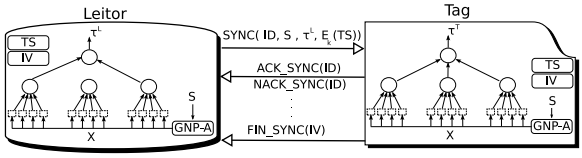


Figura 2: Mensagens de controle utilizadas na etapa de geração das chaves.

O leitor é responsável pela criação do vetor de entrada ($X$) em cada instante de tempo, através de uma semente ($S$) de 128 *bits*. O leitor tem ainda a atribuição de informar a *tag* (através do quadro de sincronização, chamado de SYNC) o valor da semente, sua saída obtida ($\tau^L$) e uma seqüência cifrada de *bits*, conforme mostrado na Figura 2. Esta seqüência é obtida cifrando uma variável do sistema chamada TS, utilizando como chave os 128 primeiros *bits* dos pesos. O envio desta variável cifrada se faz necessário para o teste de sincronização. O quadro SYNC tem ainda um identificador (ID). Este campo tem a finalidade de informar ao leitor e a *tag* qual é o identificador da mensagem. A variável ID começa com zero e é incrementada toda vez que o leitor enviar um quadro de sincronização.

Quando a *tag* recebe a mensagem de sincronização, ela deverá realizar o teste de integridade. Casos os dados não tenham sofrido nenhuma alteração, a *tag* irá realizar o teste de sincronização. O teste de sincronização é realizado da seguinte maneira: a *tag* utiliza os 128 primeiros *bits* dos pesos da rede neural como chave na decriptografia da variável cifrada recebida do leitor ($E_k(TS)$). Caso o resultado obtido seja igual à variável TS, armazenada previamente em sua memória, as redes estão sincronizadas. Na seqüência, a *tag* deverá aleatoria-

mente escolher quais dos seus pesos serão utilizados para a criação da chave de criptografia. Posteriormente, a *tag* deverá informar ao leitor que já obteve a sincronização e o índice dos pesos que serão utilizados para geração da chave. Para isso a *tag* utiliza o campo índice do vetor (IV) da mensagem FIN_SYNC.

Se a decriptografia de $E_k(TS)$ não gerar o resultado esperado, a *tag* deverá utiliza a semente ($S$) no seu gerador de número pseudo-aleatório para obter a respectiva entrada da rede ($X$). Em seguida, a *tag* irá obter a sua saída da rede neural ($\tau^T$). Caso a saída da sua rede seja igual a saída da rede do leitor ($\tau^T = \tau^L$), a *tag* deverá ajustar os seus pesos. Ao término da atualização dos pesos a *tag* deverá informar ao leitor que as saídas foram iguais, para que o mesmo possa também ajustar os pesos dele. Para informar ao leitor a *tag* utiliza a mensagem ACK_SYNC composta apensa do identificador, sendo este igual ao identificador da mensagem SYNC recebida. Se a saída da *tag* for diferente do leitor, a *tag* não deverá ajustar os seus pesos e enviará para o leitor a mensagem NACK_SYNC, com o ID igual ao da mensagem recebida. A mensagem NACK_SYNC irá informar que o leitor não deverá ajustar os seus pesos.

O leitor que estava aguardando a resposta da *tag*, ao receber a mensagem deverá também realizar o teste de integridade. Caso os dados não tenham sofrido nenhuma alteração, o leitor irá verificar se o identificador do quadro recebido corresponde ao identificador do quadro esperado. Se a mensagem for uma mensagem antiga ou se os dados tiverem corrompidos o leitor retornará a aguardar novos quadros da *tag*.

Se a mensagem for válida, o leitor irá verificar o tipo do quadro. Se o quadro for um quadro de término de sincronização (FIN_SYNC), ele deverá extrair as chaves da rede neural de acordo com o índice do vetor informado pela *tag*. Por outro lado, se o quadro for um quadro de confirmação (ACK_SYNC), ou seja, a saída de ambas as rede são iguais, o leitor deverá ajustar os seus pesos. Para qualquer outro tipo de quadro o leitor reiniciará o processo de sincronização.

Na Figura 3 são apresentados resultados de simulações que mostram a relação entre o tempo médio de sincronização em função de $l$ em 1.000 amostras. É possível observar que o aumento no nível de segurança (aumento de $l$) implica no aumento da quantidade de iterações necessárias para a sincronização.

Ao término do processo de treinamento, ambas as redes apresentarão as mesmas chaves de criptografia. Porém, somente o processo de geração de chaves não garante a segurança da informação. Pois,

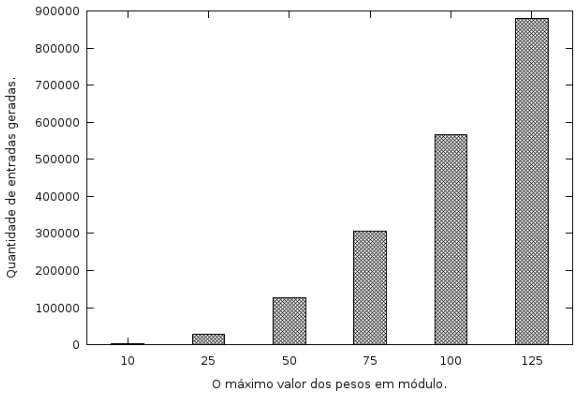


Figura 3: Tempo médio de sincronização em função de $l$, obtido em 1.000 simulações.

qualquer atacante poderá também sincronizar com um dispositivo RFID autorizado, uma vez que o protocolo é de conhecimento público. Desta forma, para garantir que somente entidades autoridades tenham acesso às informações se faz necessário o processo de autenticação.

Autenticação é a garantia da identidade de uma entidade para com outra em um canal de comunicação. Existem vários métodos de autenticação, diferenciados principalmente pela utilização de chaves secretas ou chave pública. Em autenticação com chaves secretas, utilizada pela camada proposta, ambas as entidades deverão ter um código secreto comum, neste trabalho chamadas de CAT (Chave de Autenticação da *Tag*) e CAL (Chave de Autenticação do Leitor). As chaves deverão ser inseridas pelo administrador do sistema.

A autenticação inicia com ambas as redes sincronizadas. As mensagem trocadas entre o leitor e a *tag* durante a etapa de autenticação são mostradas na Figura 4. Primeiramente, o leitor irá utilizar os pesos da sua rede neural como chave para criptografar a variável CAL. Esta variável se faz necessária para informar a *tag* que o leitor é uma entidade autorizada. O leitor, após criptografar a CAL, envia uma mensagem de autenticação (AUTH) para a *tag* e fica aguardando a resposta. Se a *tag* não responder até um determinado tempo limite, o leitor incrementa o número de tentativas. Se esta quantidade não ultrapassar um limiar, o leitor enviará novamente a mensagem de autenticação. Caso contrário, o leitor finalizará o processo de autenticação.

A *tag*, ao receber o quadro de AUTH, deverá verificar a integridade da mensagem. Se a mensagem não tiver corrompida, a *tag* deverá decriptografar os dados obtidos. Se o resultado da decriptografia for igual à chave de autenticação do leitor, a *tag* tomará conhecimento que o dispositivo que está sincronizado com ela é um dispositivo autorizado.

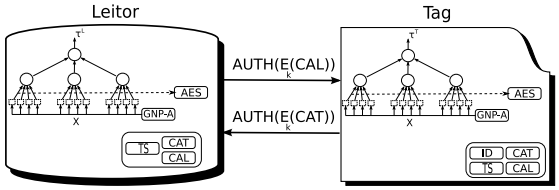


Figura 4: Mensagens de controle utilizadas na etapa de autenticação.

Na seqüência a *tag* deverá se autenticar, para isso ela deverá utilizar os pesos da sua rede neural para criptografar a variável CAT e enviar o texto cifrado para o leitor, através da mensagem AUTH.

O leitor ao receber a mensagem deverá realizar o teste de integridade. Caso os dados não tenham sofrido nenhuma alteração, o leitor deverá decriptografar os dados e verificar a chave de autenticação da *tag*. Se a decriptografia gerar uma chave válida a *tag* está autenticada terminando o processo de autenticação.

Os quadros de controle utilizados pelo protocolo proposto são mostrados na Figura 5.

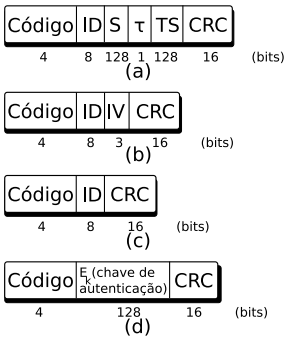


Figura 5: Quadros usados pelo protocolo proposto: (a) SYNC, (b) FIN_SYNC, (c) ACK_SYNC e NACK_SYNC e (d)AUTH.

## 4 Conclusões e Trabalhos Futuros

Para garantir uma comunicação segura entre componentes RFID se faz necessário a utilização de criptografia com chave simétrica e um protocolo de distribuição de chaves. Algoritmos de criptografia já foram apresentados na literatura e vêm se mostrando adequados, porém os protocolos de gerenciamento de chaves propostos são inseguros ou impraticáveis na maioria das aplicações.

Desta forma, foi proposto no presente artigo um novo protocolo de gerenciamento de chaves. Este novo protocolo faz uso de redes neurais artificiais com aprendizado mútuo para o processo de geração da chave de criptografia.

Nos testes realizados, o novo protocolo apresentou um desempenho satisfatório, sendo o nível de segurança função apenas dos recursos computacionais e temporais das entidades do sistema RFID. Após os testes foi possível definir alguns parâmetros inerentes ao protocolo. Sendo esta definição uma sugestão para a implementação do protocolo, ou seja, os valores destes parâmetros poderão ser alterados dependendo dos requisitos das aplicações.

Têm-se como trabalhos futuros: realizar análise temporal para avaliar o tempo necessário para se obter a sincronização e construir uma prova de conceito utilizando FPGA (*Field Programmable Gate Array*).

## Referências

Avoine, G. and Castelluccia, C. (2006). Noisy Tags: A Pretty Good Key Exchange Protocol for RFID Tags, Vol. 3928, Springer-Verlag, pp. 289–299.

Feldhofer, M. (2004). A Proposal for an Authentication Protocol in a Security Layer for RFID Smart Tags, Vol. 2, pp. 759–762.

Haykin, S. (1998). *Neural Networks: A Comprehensive Foundation*, Prentice Hall PTR, Upper Saddle River, NJ, USA.

Jeng, A., Chang, L.-C. and Chen, S.-H. (2008). A Low Cost Key Agreement Protocol Based on Binary Tree for EPCGlobal Class 1 Generation 2 RFID Protocol, *IEICE Transactions* **91-D**(5): 1408–1415.

Juels, A. (2005). Minimalist Cryptography for Low-Cost RFID Tags, pp. 149–164.

Juels, A. (2006). RFID security and privacy: a research survey, *Selected Areas in Communications, IEEE Journal on* **24**(2): 381–394.

Kinzel, W. and Kanter, I. (2002). Neural cryptography, *in Proc. of the 9th International Conference on Neural Information Processing*, pp. 18–22.

Lei, H., Yong, G., Na-Na, L. and Zeng-Yu, C. (2007). A Security-Provable Authentication and Key Agreement Protocol in RFID System, *IEEE - WiCom* pp. 2078 – 2080.

Ruttor, A. (2007). Neural Synchronization and Cryptography.