UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
CENTRO DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA E DE COMPUTAÇÃO

Implementação e Análise de Desempenho dos Protocolos de Criptografia Neural e Diffie-Hellman em Sistemas RFID Utilizando uma Plataforma Embarcada

José Macêdo Firmino Filho

Orientadora: Prof[a]. Dr[a]. Ana Maria Guimarães Guerreiro

Co-orientador: Prof. Dr. Gláucio Bezerra Brandão

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Engenharia Elétrica da UFRN (área de concentração: Engenharia de Computação) como parte dos requisitos para obtenção do título de Mestre em Ciências.

Natal, RN, dezembro de 2009

Divisão de Serviços Técnicos

Catalogação da publicação na fonte. UFRN / Biblioteca Central Zila Mamede

Firmino Filho, José Macêdo.

Implementação e análise de desempenho dos protocolos de criptografia neural e Diffie-Hellman em sistemas RFID utilizando uma plataforma embarcada / José Macedo Firmino Filho. - Natal, RN, 2009.

49 f.

Orientadora: Ana Maria Guimarães Guerreiro.

Co-orientador: Gláucio Bezerra Brandão.

Dissertação (Mestrado) - Universidade Federal do Rio Grande do Norte. Centro de Tecnologia. Programa de Pós-Graduação em Engenharia Elétrica e de Computação.

1. Segurança e privacidade em sistemas RFID - Dissertação. 2. Criptografia neural - Dissertação. 3. Diffie-Hellman - Dissertação. 4. Gerenciamentos de chaves - Dissertação. I. Guerreiro, Ana Maria Guimarães. II. Brandão, Gláucio Bezerra. III. Universidade Federal do Rio Grande do Norte. IV. Título.

RN/UF/BCZM CDU 004.056(043.3)

Implementação e Análise de Desempenho dos Protocolos de Criptografia Neural e Diffie-Hellman em Sistemas RFID Utilizando uma Plataforma Embarcada

José Macêdo Firmino Filho

Dissertação de Mestrado aprovada em 16 de dezembro de 2009 pela banca examinadora composta pelos seguintes membros:

Prof[a]. Dr[a]. Ana Maria Guimarães Guerreiro (orientadora) ..... DCA/UFRN

Prof. Dr. Gláucio Bezerra Brandão (co-orientador) ..... DCA/UFRN

Prof[a]. Dr[a]. Heliana Bezerra Soares ..... UFERSA

Prof. Dr. Ricardo Alexsandro de Medeiros Valentim ..... IFRN

Agradecimentos

À CAPES, pelo apoio financeiro com a manutenção da bolsa de auxílio.

Ao minha orientadora e ao meu co-orientador, professora Ana Maria e Gláucio Brandão, sou grato pela orientação, apoio e amizade.

Ao professor Ricardo Valentim e professora Heliana Bezerra pelas sugestões e incentivos.

Aos colegas do LAHB (Laboratório de Automação Hospitalar e Bioengenharia), pelas reflexões, críticas e sugestões recebidas.

À minha família e minha querida Janaide pelo apoio durante esta jornada.

Ao meu querido amigo João Paulo e a todos, que de alguma forma contribuíram para a realização deste trabalho.

E principalmente, quero agradecer a Deus por mais esta etapa vencida e por me dar o privilégio de conviver com todas estas pessoas.

Resumo

Identificação por rádio freqüência, também chamada de RFID (*Radio Frequency Identification*), representa uma tecnologia de transmissão de dados sem fio. Estes dados são relacionados principalmente a códigos de identificação. A tecnologia RFID vem apresentando um grande potencial de utilização em setores da automação industrial, residencial e hospitalar. No entanto, estas aplicações podem resultar em riscos a segurança e privacidade dos usuários. Recentemente, pesquisadores vêm apresentando possíveis soluções as ameaças de segurança da tecnologia. Entre estas soluções estão os protocolos de distribuição de chaves criptográficas. O presente trabalho tem como objetivo realizar uma avaliação de desempenho dos protocolos de Criptografia Neural e Diffie-Hellman na geração de chaves em sistemas RFID. Para isso, iremos mensurar o tempo de processamento destes protocolos. Para os testes foi desenvolvido uma plataforma em FPGA (*Field-Programmable Gate Array*) com o processador embarcado Nios II®. Sobre esta plataforma foram utilizados os protocolos de Criptografia Neural e Diffie-Hellman no processo de geração de chaves criptográficas. A metodologia de pesquisa baseia-se na agregação de conhecimento ao desenvolvimento de novos sistemas RFID através de uma análise comparativa entre esses dois protocolos de segurança da informação. As principais contribuições deste trabalho são: avaliação de desempenho dos protocolos (Diffie-Hellman e Criptografia Neural) em uma plataforma embarcada e um levantamento bibliográfico de pesquisas relacionadas à segurança da informação em sistemas RFID. Nos resultados obtidos foi possível observar que o protocolo de Diffie-Hellman é mais apropriado para sistemas RFID.

Palavras-chave: Segurança e Privacidade em Sistemas RFID, Criptografia Neural, Diffie-Hellman, Gerenciamento de Chaves.

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. Unfortunately, the key problem that impacts the application of RFID system is the security of the information. Recently, researchers have demonstrated solutions to security threats in RFID technology. Among these solutions are several key management protocols. This master dissertations presents a performance evaluation of Neural Cryptography and Diffie-Hellman protocols in RFID systems. For this, we measure the processing time inherent in these protocols. The tests was developed on FPGA (Field-Programmable Gate Array) platform with Nios II® embedded processor. The research methodology is based on the aggregation of knowledge to development of new RFID systems through a comparative analysis between these two protocols. The main contributions of this work are: performance evaluation of protocols (Diffie-Hellman encryption and Neural) on embedded platform and a survey on RFID security threats. According to the results the Diffie-Hellman key agreement protocol is more suitable for RFID systems.

Keywords: Security and Privacy for RFID Systems, Neural Cryptography, Diffie-Hellman, Key Management Protocol.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

Lista de Símbolos e Abreviaturas

X	Vetor de entrada da rede TPM
σ_i	Saída do neurônio escondido i da rede TPM
τ	Saída da rede TPM
k	Número de neurônios escondidos da rede TPM
l	Maior valor possível dos pesos, em módulo, da rede TPM
n	Dimensão do vetor de entrada da rede TPM
$w_{i,j}$	Peso do neurônio escondido i com relação a entrada com índice j da rede TPM
$(\sigma_1, \sigma_2, ..., \sigma_k)$	Representação interna da rede TPM
ACK_SYNC:	Mensagem de confirmação de saídas iguais
AES:	*Advanced Encryption Standard*
CRC:	Verificação de redundância cíclica
EPC:	*Electronic Product Code*
FIN_SYNC:	Mensagem de término de sincronização
ISO:	*International Organization for Standardization*
NACK_SYNC:	Mensagem de confirmação de saídas diferentes
RFID:	Identificação por rádio freqüência
SYNC:	Mensagem de sincronização
TPM:	*Tree parity machine*

Capítulo 1

Introdução

Sistemas de identificação automáticos estão se tornando cada vez mais comuns em diversos setores da sociedade moderna. Estes sistemas podem ser aplicados, por exemplo, em cobrança de pedágio, hospitais [Florentino et al. n.d.], laboratórios de análise clínica [Valentim et al. 2008], supermercados, controle de acesso a ambientes restritos, localização e identificação de pessoas, inventário de equipamentos, sistemas antifurto de veículos, rastreamento de animais, transporte público, identificação de *container* em portos e bagagens em aeroportos [Finkenzeller 2003].

Código de barras é a tecnologia de identificação automática mais utilizada atualmente. Esta tecnologia corresponde a uma representação gráfica de dados numéricos ou alfanuméricos [Finkenzeller 2003]. Entretanto, outro sistema de identificação automática, conhecido como identificação por rádio freqüência (RFID), vem chamando à atenção da comunidade acadêmica, industrial e comercial.

Sistemas RFID são constituídos de *tags*, leitores e um sistema de processamento de dados. As *tags* e os leitores são compostos, principalmente, por circuitos integrados conectados a uma antena [Finkenzeller 2003]. As *tags* contêm os dados de identificação e possivelmente outros dados necessários para a utilização do sistema (por exemplo, variáveis de sensores). As *tags* são inseridas nos objetos ao qual se deseja identificar. O leitor tem a função de realizar a leitura dos dados das *tags*, quando ela estiver sobre o alcance de leitura. Posteriormente, o leitor deverá disponibilizar estes dados em uma interface gráfica ou a um sistema de processamento de dados.

Em 2002, em média eram lidos diariamente cinco bilhões de código de barras em todo o mundo [Sarma et al. 2002]. No entanto, a leitura dos códigos de barras necessita de um acoplamento óptico, ou seja, se faz necessário que o leitor esteja a certo ângulo e distância para permitir que as ondas luminosas transmitam as informações. Este fato torna a intervenção humana quase sempre necessária nesses sistemas. Outra desvantagem é a impossibilidade de adicionar informações a um código de barras após ele ter sido impresso. Por outro lado, os sistemas RFID não exigem linha de visada e é possível gravar e regravar os dados várias vezes em uma mesma *tag*.

1.1 Estado da Arte

Desde a primeira transmissão de sinais de rádio através do Atlântico (por Guglielmo Marconi) em 1901, ondas de rádio passaram a ser um importante meio de transmissão de dados. Além disso, Alexander Watson-Watt demonstrou em 1935 (através de sua invenção chamada radar) que ondas de rádio também poderiam ser utilizadas para localizar objetos físicos. O radar começou a ser amplamente utilizado, durante a Segunda Guerra Mundial, na localização de aeronaves. No entanto, os operadores não podiam distinguir as suas aeronaves dos aviões inimigos, o que deixava o sistema vulnerável a erros. Os ingleses então criaram uns dos primeiros sistemas RFID ativos, chamado de *Friend or Foe*. Desta forma, os aliados podiam distinguir as suas aeronaves das aeronaves inimigas [Rieback, Crispo & Tanenbaum 2006*b*].

Foi publicado em 1948 um dos primeiros trabalhos acadêmicos que tratam da tecnologia RFID. O artigo de Harry Stockman [Stockman 1948] apresentou as pesquisas relacionadas à tecnologia RFID realizadas durante a Segunda Guerra Mundial. A partir deste artigo, aumentou o interesse da comunidade acadêmica pela tecnologia. Porém, somente em 1983, Charles Walton patenteou um sistema de identificação automática composto por um identificador portátil. Este identificador era formado por um oscilador e um codificador. O identificador de Walton tinha a função de gerar um sinal modulado de acordo com a posição para identificação do usuário [Rieback, Crispo & Tanenbaum 2006*b*]. Este sistema de identificação automático foi o embrião da tecnologia RFID moderna.

Muitas empresas se interessaram pela tecnologia e começaram a produzir equipamentos RFID, o que resultou na produção de dispositivos incompatíveis. Conseqüentemente, surgiu a necessidade da criação de padrões para a tecnologia. Na década de 90, organizações passaram a trabalhar em conjunto com vários fabricantes e pesquisadores a fim de padronizar protocolos e regras de utilização para a tecnologia.

Em 1999, o UCC (*Uniform Code Council*), *EAN Internacional*, Procter & Gamble, universidades e a Gilette se uniram para a criação de um centro de pesquisa, chamado de *Auto-ID Center*, no Instituto de Tecnologia de Massachusetts. Este centro de pesquisa tinha o objetivo de desenvolver padrões globais e abertos para a tecnologia RFID.

O *Auto-ID Center* também foi encarregado de desenvolver uma arquitetura de rede e um banco de dados de identificadores RFID. Estas ferramentas tinham o objetivo de disponibilizar informações associadas a números de identificação armazenados nas *tags*. Desta forma, os fabricantes e fornecedores permitiriam aos seus clientes obterem informações dos produtos em tempo real. Por exemplo, seria possível saber quando os produtos comprados seriam despachados e a data da entrega. O *Auto-ID Center* terminou os seus trabalhos em 2003, surgindo o seu sucessor comercial chamado de EPCGlobal.

As tradicionais técnicas de segurança da informação (utilizadas em redes de computadores) não podem ser aplicadas diretamente a sistemas RFID devida as características específicas da tecnologia. São elas:

- Para tornar o sistema economicamente viável as *tags* RFID deverão ser limitadas com relação ao fornecimento de energias, computação e capacidades de comunicação;

- As *tags* RFID são geralmente colocadas em áreas acessíveis aos possíveis atacantes, resultando num risco adicional de ataque físico;
- Não deve ser possível associar uma *tag* ao seu proprietário por um longo período de tempo.

Conseqüentemente, alguns mecanismos de segurança existentes são inadequados. Desta forma, as pesquisas relacionadas à segurança da informação em sistemas RFID podem ser divididas em duas classes: as que propõem novos paradigmas e as que adaptam as técnicas existentes na literatura à realidade da tecnologia RFID.

Alguns autores mostraram que é possível utilizar algoritmos tradicionais de criptografia com chave simétrica para sistemas RFID [Feldhofer et al. 2004] [Juels 2005]. Entretanto, uma chave simétrica entre duas entidades é útil se, e somente se, for utilizada uma única vez. A chave deverá ser criada no início da sessão e destruída quando a sessão for finalizada [Forouzan 2006].

Piramuthu [Piramuthu 2007] avaliou novos protocolos de autenticação para a tecnologia RFID. Ele identificou vulnerabilidades e sugeriu modificações em alguns deles. Entre estes protocolos está o protocolo de Molnar e Wagner [Molnar & Wagner 2004]. Eles apresentaram um protocolo de autenticação que utilizava uma chave secreta compartilhada. Neste protocolo a *tag* e o leitor geram os números aleatórios e compartilha-os. O identificador da *tag* é obtido através de operações XOR entre estes números aleatórios e a chave secreta. Como os valores aleatórios são atualizados a cada instanciação do protocolo, ataques de *replay* e rastreamento clandestino são evitados.

Para a criação, distribuição e administração das chaves de criptografia se fazem necessário um protocolo de gerenciamento de chaves. 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 as *tags*. Além disso, o fato da chave ser específica da *tag* leva a um paradoxo. A *tag* deverá se identificar ao 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*, ele não tem como 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*, esta chave deverá ser divulgada a todos os leitores responsáveis pela obtenção dos dados.

Várias pesquisas têm apresentado novos paradigmas para o problema de gerenciamento de chaves em sistemas RFID. Castelluccia e Avoine [Avoine & Castelluccia 2006] propuseram a utilização da *noisy tag*. Estas *tags* geram ruídos pseudo-aleatório no canal de comunicação entre o leitor e a *tag*, de modo que um atacante não possa diferenciar a mensagem enviada pela *tag* do ruído enviado pela *noisy tag*. 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 ruídos dinâmicos o gerenciamento do sistema se torna complexo.

Jeng *et al.* [Jeng et al. 2008] propuseram um protocolo de distribuição de chaves baseado numa estrutura de dados, chamada de *generic binary tree*. Nesta abordagem, as entidades trocam índices para chaves secretas contidas nesta estrutura de dados. O

problema desta abordagem é a dificuldade de gerenciamento quando utilizada com chaves dinâmicas.

Outra solução foi apresentada por Lei *et al.* [Lei et al. 2007]. Eles 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 ela não apresenta chaves dinâmicas e a necessidade de uma grande capacidade de armazenamento para os leitores.

Juels [Juels 2006] apresentou uma solução onde se utiliza um conjunto de chaves de criptografia. Cada par leitor/*tag* tem um conjunto de chaves simétricas. A *tag* escolhe aleatoriamente uma chave entre um conjunto de chaves possíveis. Na seqüência, ela envia o seu identificador criptografado para o leitor. O leitor ao receber a mensagem, realiza uma pesquisa em sua base de dados com o objetivo de encontrar uma chave que na decriptografia, dos dados recebidos, ele consiga obter um identificador válido. Caso encontre, o leitor obtém a chave de criptografia da *tag*. Esta solução se torna complexa em sistemas com um grande número de *tags* e se utilizarmos chaves dinâmicas.

Entre os protocolos de geração de chaves presentes em redes de computadores se destaca o protocolo de Diffie-Helman (DH) [Diffie & Hellman 1976]. A finalidade do protocolo é permitir que dois usuários negociem uma chave com segurança, a qual pode então ser usada para a subseqüente criptografia das mensagens, em um canal inseguro. O protocolo tem dois parâmetros (p e g). Ambos são públicos e devem ser utilizados por todos os usuários do sistema. O parâmetro p é um número primo e g é um inteiro menor que p, com a seguinte propriedade: g é raiz primitiva de p. As chaves são obtidas através de operações de exponenciação e operação *mod*. Embora este protocolo seja bastante utilizado em redes de computadores, não existe na literatura implementações e análise de desempenho do mesmo em sistemas RFID.

Outro protocolo para a geração de chaves foi apresentado por Kinzel e Kanter [Kinzel & Kanter 2002], chamado de Criptografia Neural. Eles mostraram que duas redes neurais poderiam ser treinadas mutuamente até que os seus pesos se tornem idênticos. Devido a esta capacidade de sincronização das redes neurais, as mesmas foram utilizadas na construção de um protocolo de troca de chaves criptográficas. Volkmer e Wallner [Volkmer & Wallner 2005] avaliaram a utilização do protocolo de Criptografia Neural em sistemas RFID. Segundo os autores o protocolo se mostrou eficiente, porém limitado pelo canal de comunicação.

1.2 Motivação

Pode-se observar que a tecnologia RFID, embora seja relativamente nova, esta sendo incorporada nos diversos segmentos produtivos e estratégicos da sociedade industrializada. Entretanto, pesquisas vêm mostrando vulnerabilidades da tecnologia com relação à segurança e privacidade das informações. Estas vulnerabilidades estão se tornando uma barreira para a utilização da tecnologia em algumas aplicações, o que justifica a pesquisa e o desenvolvimento de projetos de segurança da informação relacionados ao tema.

1.3 Objetivos e Contribuições

O presente trabalho tem como objetivo realizar uma avaliação comparativa do desempenho dos protocolos de Criptografia Neural e Diffie-Hellman na geração de chaves em sistemas RFID. Para isso, foi mensurado o tempo de processamento destes protocolos. O tempo de processamento corresponde ao tempo necessário para a geração das chaves criptográficas desprezando o tempo de comunicação. O tempo de comunicação é função da quantidade e do tamanho das mensagens. Para mensuramos o tempo de comunicação se fazia necessário a utilização de transmissores e receptores de rádio freqüência. A análise do tempo de comunicação está fora do escopo deste trabalho pois o mesmo não depende do protocolo que esta realizando a comunicação.

Para os testes foi desenvolvido uma plataforma em FPGA (*Field-Programmable Gate Array*) com o processador embarcado Nios II. Sobre esta plataforma foram utilizados os protocolos de Criptografia Neural e Diffie-Hellman no processo de geração de chaves criptográficas.

A metodologia de pesquisa baseia-se na agregação de conhecimento ao desenvolvimento de novos sistemas RFID através de uma análise comparativa entre esses dois protocolos de segurança da informação.

As principais contribuições deste trabalho são:

- Levantamento bibliográfico de pesquisas relacionadas à segurança da informação em sistemas RFID;
- Implementação dos protocolos Diffie-Hellman e Criptografia Neural em uma plataforma embarcada em *hardware* reconfigurável;
- Avaliação de desempenho destes protocolos na plataforma em FPGA;

1.4 Estrutura da Dissertação

Nesta dissertação, o Capítulo 2 apresenta os fundamentos básicos de segurança da informação que acreditamos serem importantes para o entendimento dos diversos aspectos envolvidos nesta dissertação. Estes fundamentos incluem: criptografia, autenticação, distribuição de chaves e códigos de verificação de integridade.

No Capítulo 3, são apresentados os principais componentes que formam um sistema RFID, como eles são classificados e quais são os principais padrões relacionados com a tecnologia. Ainda no Capítulo 3, são apresentadas as principais vulnerabilidades de segurança da tecnologia RFID e as possíveis maneiras de evitá-las.

O Capítulo 4 traz a especificação dos protocolos de Criptografia Neural e Diffie-Hellman, os ataques relacionados a estes protocolos e suas possíveis contra medidas.

No Capítulo 5 é apresentado a plataforma de teste, as características da tecnologia FPGA e do processador Nios II, como os protocolos de Criptografia Neural e Diffie-Hellman foram implementados na plataforma e análises dos resultados dos testes sobre os protocolos.

Finalmente, no Capítulo 6 são mostradas as conclusões, comentários e perspectivas.

Capítulo 2

Segurança da Informação

Sistemas de informação combinam recursos humanos e computacionais para a coleta, armazenamento, recuperação e distribuição de dados. Estes sistemas têm o objetivo de proporcionar a eficiência gerencial (isto é, planejamento, controle, comunicação e tomada de decisão) nas organizações. Eles são formados por um conjunto de medidas que visam garantir quatro propriedades [Tanenbaum 2002]:

- Confidencialidade: é a propriedade de que as informações não estejam disponíveis a indivíduos, entidades ou processos não autorizados;
- Irretrabilidade: é a propriedade que oferece proteção contra negação de autoria por parte de uma das entidades envolvidas na comunicação;
- Integridade: é a propriedade que garante a preservação da exatidão e completeza da informação, ou seja, que os dados devem chegar ao destinatário exatamente como eles foram enviados;
- Disponibilidade: é a propriedade que garante que as informações estarão acessíveis e utilizáveis quando uma entidade autorizada solicitar.

Existem inúmeras situações de insegurança que podem afetar os sistemas de informação, tais como: incêndios, alagamentos, problemas elétricos, poeira, fraudes, uso inadequado dos sistemas, engenharia social, guerras, seqüestros, etc. Portanto podemos dizer que não existe segurança absoluta. Deste modo, torna-se necessário agirmos no sentido de descobrirmos quais são os pontos vulneráveis e a partir daí avaliarmos os riscos e impactos. Após esta avaliação deveremos providenciar para que a segurança da informação seja eficaz.

Na seqüência serão apresentados, resumidamente, quais são os principais serviços que visam garantir as quatro propriedades (confidencialidade, irretrabilidade, integridade e disponibilidade). Para exemplificarmos estes serviços, introduzimos três personagens: Alice, Bob e Eve. Alice é a personagem que necessita transmitir uma mensagem secreta. Bob é o receptor da mensagem. Eve é a personagem que, de algum modo, perturba a comunicação entre Alice e Bob, interceptando mensagens ou enviando mensagens falsas.

2.1 Criptografia

Criptografia pode ser entendida como um conjunto de métodos e técnicas que visam garantir a confidencialidade da informação. Para isso, elas convertem um texto original em texto ilegível, sendo possível mediante o processo inverso, recuperar as informações originais [Singh 1999].

A mensagem original é denominada texto limpo. Após a transformação por meio de um algoritmo, a mensagem passa a ser conhecida como texto cifrado ou texto criptografado. Um algoritmo de cifragem transforma o texto limpo em texto cifrado. Enquanto que, um algoritmo de decifragem reverte o processo, transformando o texto cifrado em texto limpo.

Atualmente, os algoritmos criptográficos são divulgados à comunidade e o sigilo das informações é garantido apenas pela chave. O que significa que se alguém descobrir a chave para decifrar uma determinada informação, todas as outras informações cifradas com esse algoritmo ainda estarão protegidas, por terem chaves diferentes.

Já um algoritmo criptográfico que não utiliza o recurso de chaves para cifrar as informações pode resultar em perigosas conseqüências. Por exemplo, se este algoritmo for quebrado, todas as informações cifradas por ele estarão desprotegidas, pois o que garante o sigilo das informações é o próprio algoritmo.

Por esses motivos toda criptografia moderna opera com chaves. Portanto, para criptografar uma mensagem, precisa-se de um algoritmo de cifragem, uma chave e um texto limpo. Deste conjunto origina-se o texto cifrado. Para decifrar uma mensagem, precisamos de um algoritmo de decifragem, uma chave e um texto cifrado. Deste conjunto revela o texto limpo original.

Suponha que Alice e Bob querem trocar mensagens secretas, ao longo de um canal público. A fim de proteger a confidencialidade da informação, Alice deverá utilizar algum algoritmo de criptografia de modo que a mensagem criptografada pareça incompreensível para Eve. Existem dois principais tipos de algoritmos de cifragem que Alice e Bob poderão utilizar, são eles: criptografia com chave simétrica e criptografia com chave pública (também chamada de criptografia com chaves assimétricas). Na seqüência será apresentado cada um deles.

2.1.1 Criptografia com Chave Simétrica

Na criptografia com chave simétrica Alice usa uma determinada chave e um algoritmo de cifragem para criptografar a mensagem, enquanto que Bob usa a mesma chave e um algoritmo de decifragem recíproco para decifrar a mensagem. Por recíproco deve ficar subtendido que o algoritmo de decifragem realiza operações inversas com relação ao algoritmo de cifragem. A Figura 2.1 ilustra o processo de criptografia com chave simétrica.

Os algoritmos de chave simétrica são eficientes, porém a cada par de usuários deve-se associar uma única chave [Forouzan 2006]. Isto significa que se n pessoas quiserem se comunicar usando este método, serão necessários $n(n-1)/2$ chaves simétricas. O que gera o problema de armazenamento e distribuição das chaves.

Os algoritmos criptográficos com chaves simétricas mais utilizadas são: DES (*Data*

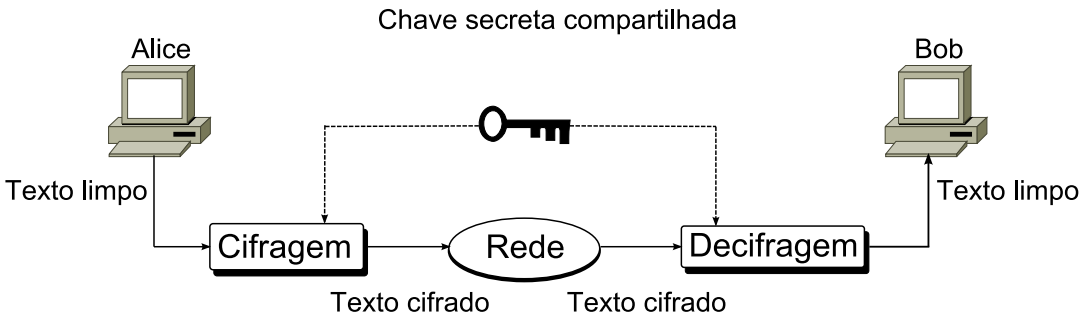

Figura 2.1: Privacidade usando criptografia com chave simétrica.

Encryption Standard), AES (*Advanced Encryption Standard*), 3-DES e RC4 (Rivest Cipher 4). Dentre eles se destaca o AES, pelos seguintes motivos: apresenta uma margem de segurança adequada, apresenta um baixo custo computacional, é utilizável em ambientes de recursos restritos [Stallings 2008] e pode ser implementado em sistemas RFID [Feldhofer et al. 2004].

2.1.2 Criptografia com Chave Pública

Na criptografia com chave pública, há duas chaves: uma chave privada e uma chave pública. A chave privada é mantida em segredo pelo receptor. Enquanto que a chave pública é distribuída publicamente. Uma restrição, com relação a estas chaves, é que a chave privada não pode ser obtida a partir da chave pública.

Se Alice desejar enviar uma mensagem secreta para Bob, ela deverá usar a chave pública de Bob para cifrar a mensagem. Quando a mensagem for recebida por Bob, a chave privada dele será usada para decifrar a mensagem, conforme mostrado na Figura 2.2.

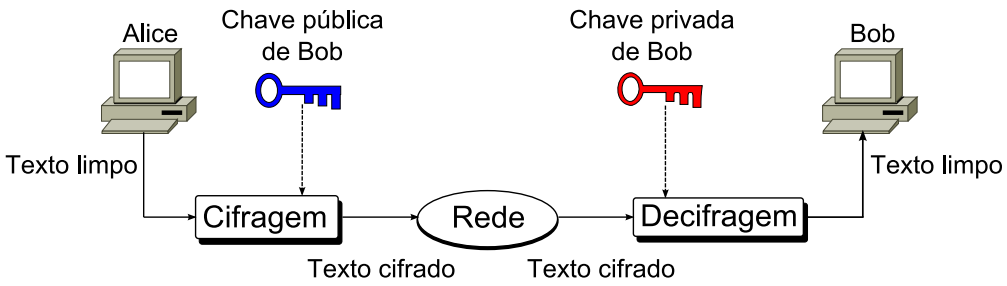

Figura 2.2: Privacidade usando criptografia com chave pública.

A cifragem com chave pública remove a necessidade de uma chave compartilhada entre as duas entidades. Desta forma, para uma comunicação segura com n entidades, serão necessários apenas $2n$ chaves. Por exemplo, se utilizarmos a criptografia com chave pública em uma rede com 100 entidades precisaríamos de 200 chaves. Enquanto que, com criptografia com chave simétrica se fazem necessárias 4.950 chaves.

A maior desvantagem da criptografia com chaves públicas e a complexidade dos algoritmos. Se desejarmos que o método seja relativamente efetivo, precisamos de chaves muito extensas. Porém, o cálculo do texto cifrado a partir do texto limpo usando chaves muito extensas leva a uma quantidade de tempo relativamente grande [Forouzan 2006].

A distribuição de chaves também é um problema nestes tipos de algoritmos. Para evitar problemas de segurança, a associação entre uma entidade e sua respectiva chave pública deverá ser verificada. Por exemplo, se Alice enviar a sua chave pública por e-mail para Bob, como Bob terá certeza que foi realmente Alice que enviou e não Eve. Uma solução faz uso de uma nova entidade na rede chamada de autoridade certificadora. Esta nova entidade se responsabilizará por entregar as chaves públicas de forma confiável.

2.2 Autenticação

Autenticação é um procedimento que verifica a identidade de uma entidade, ou processo, para permitir a propriedade de irretrabilidade. Por exemplo, Alice deseja retirar certa quantia em dinheiro da sua conta corrente. Para isso, ela envia uma mensagem de solicitação de saque para Bob (seu gerente). Através do serviço de autenticação Bob poderá ter certeza que quem enviou a mensagem foi realmente Alice e não Eve.

Existem duas questões centrais relativa ao problema de troca de chave de autenticação: confidencialidade e adequação no tempo. Para evitar que um atacante se passe por uma entidade autorizada, as informações de identificação precisam ser comunicadas de forma criptografada. A segunda questão, adequação no tempo, é importante devido às ameaças de repetições de mensagem.

Uma técnica para lidar com ataques por repetição (também chamado de ataque de *replay*) é chamada desafio/resposta. Nesta técnica Alice primeiro envia a Bob um desafio e exige que a mensagem subseqüente (resposta) recebida de Bob contenha o valor correto.

Existem duas principais maneiras de proporcionar autenticação. A primeira utiliza criptografia com chave simétrica, enquanto que a segunda usa criptografia com chave pública.

2.2.1 Autenticação Através de Criptografia com Chave Simétrica

Os métodos de autenticação que usam criptografia com chave simétrica são divididos em duas abordagens. Na seqüência apresentaremos cada uma separadamente.

Primeira Abordagem

Na primeira abordagem, Alice transmite a sua identidade e uma senha em uma mensagem cifrada para Bob, usando uma chave simétrica K_{AB}. A Figura 2.3 ilustra este procedimento.

Esta abordagem só é considerada segura se a chave K_{AB} for diferente para cada autenticação (ou seja, se as chaves forem dinâmicas), pois Eve pode interceptar a mensagem de autenticação, armazená-la e, mais tarde, reenviá-la para Bob. Neste caso, Bob não tem

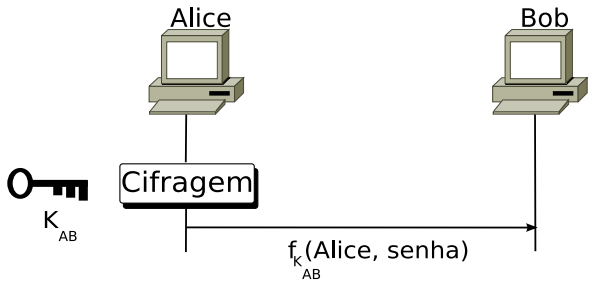

Figura 2.3: Autenticação usando chave simétrica.

como saber que essa mensagem é uma réplica da mensagem original ou uma mensagem válida. Este ataque é denominado ataque de *replay*.

Segunda Abordagem

Se desejarmos utilizar uma chave de criptografia constante e ainda evitar um ataque de *replay*, podemos adicionar informações ao procedimento para ajudar a Bob a distinguir entre uma mensagem de autenticação nova e uma repetida. Isto é feito usando o conceito de desafio/resposta, conforme mostrado na Figura 2.4.

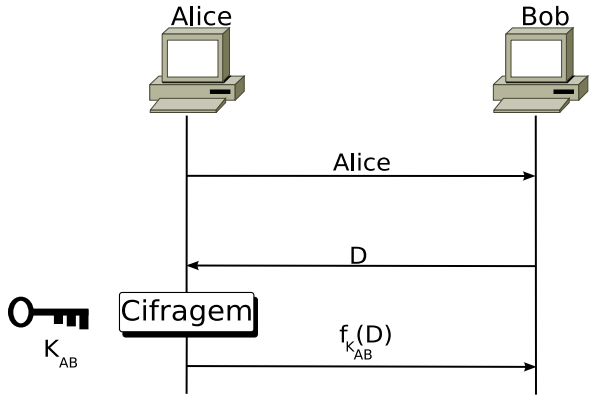

Figura 2.4: Autenticação usando a abordagem desafio/resposta.

Nesta abordagem, a autenticação acontece em três etapas. Primeiramente, Alice envia a sua identidade a Bob. Bob desafia Alice a confirmar a identidade dela, enviando um desafio (*D*). Alice responde ao desafio enviando de volta o desafio cifrado. Para cifrar o desafio, Alice utiliza a chave simétrica de criptografia dela. Eve não poderá reenviar uma antiga mensagem porque o desafio é válido uma única vez.

2.2.2 Autenticação Através de Criptografia com Chave Pública

A criptografia com chave pública pode também ser usada para autenticar um usuário. Por exemplo, Alice pode usar a sua chave privada para cifrar uma senha (ou um desafio)

e deixar que Bob use a chave pública dela para decifrar a mensagem. Se Bob conseguir decifrar a mensagem, ele tem a confirmação de que a mensagem foi enviada por Alice. Pois, somente ela tem acesso à chave privada dela.

Entretanto, pode acontecer um ataque denominado *man-in-the-middle*. Neste ataque, Eve divulga a Bob a sua chave pública se passando por Alice. Assim, Eve pode cifrar, usando a sua chave privada, a mensagem de autenticação, e enviá-la para Bob. Bob irá decifrar a mensagem utilizando a chave pública de Eve, acreditando que o atacante é Alice. Este ataque é solucionado através de mecanismos de certificação de chaves.

2.3 Distribuição de Chaves Simétricas

Foi apresentado como a criptografia com chave simétrica pode ser utilizada para agregar segurança à informação. Contudo, não explicamos como estas chaves são distribuídas. Antes de elucidarmos o processo de distribuição de chaves, iremos apresentar um problema básico em um ambiente de comunicação seguro que utiliza a criptografia com chave simétrica.

Considere o cenário em que n pessoas desejam se comunicar entre si, sendo que a comunicação entre duas pessoas não deve ser inteligível para as outras. Suponha que algum algoritmo de criptografia tenha sido utilizado para proteger a comunicação entre as duas partes. Cada pessoa deverá manter uma lista de $(n-1)$ chaves secretas, que são usados para a comunicação com as outras $(n-1)$ pessoas.

Neste cenário, para cada par se faz necessário uma chave secreta. Portanto, se há 1.000 entidades em uma rede, cada um deles terá uma lista de 999 chaves secretas e o sistema terá $([N(N-1)]/2 = 499.500)$ chaves secretas no total.

Mudanças freqüentes nas chaves normalmente são desejáveis, para limitar a quantidade de dados comprometidos se um atacante descobrir a chave. Desta forma, o processo de gerenciamento das chaves secretas se torna complexo em sistemas com grande número de entidades. Além disso, quando uma nova entidade entrar na rede, esta entidade deverá estabelecer uma nova chave secreta com todas as outras entidades existentes.

A Figura 2.5 ilustra a magnitude da tarefa de distribuição de chaves. De acordo com o gráfico, uma rede com 1.000 entidades precisaria distribuir por volta de 0,5 milhão de chaves. Se essa mesma rede admitisse 10.000 entidades, então aproximadamente 50 milhões de chaves poderiam ser necessárias para a criptografia.

Existem três soluções possíveis para o problema de distribuição de chaves [Ruttor 2007]:

1. Canal seguro: Alice e Bob poderão usar um canal seguro para transmitir as chaves, por exemplo, uma terceira pessoa de confiança. Porém, geralmente a implementação desta solução é difícil ou mesmo impraticável.
2. Criptografia de chaves pública: nesta solução, algoritmos de criptografia com chave pública são utilizados para enviar, de forma segura, chaves simétricas. Porém, fazem-se necessários mecanismos de certificação de chaves. Além disso, criptografia com chave pública não é indicado para alguns sistemas, por exemplo, sistemas embarcados [Stallings 2008].

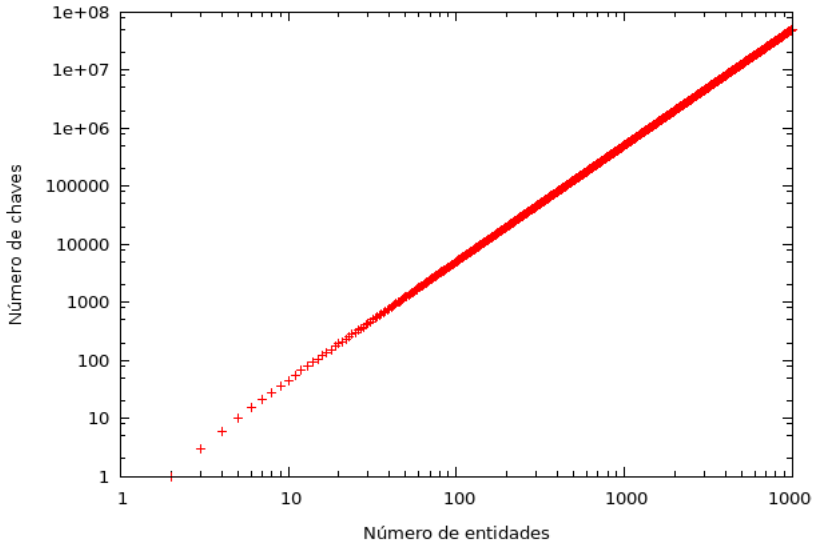

Figura 2.5: Relação entre o número de entidades e o número de chaves.

3. Protocolo de gerenciamento de chaves: neste caso, mensagens são transmitidas ao longo de um canal público. Alice e Bob geram uma chave secreta baseada nessas informações. Entre os protocolos de gerenciamento de chaves se destacam: Criptografia Neural e Diffie-Hellman.

2.4 Código de Verificação de Integridade

Uma premissa fundamental em transmissão de dados é que os sistemas de telecomunicações devem ser capazes de transferir dados de um dispositivo a outro com total precisão, ou seja, garantir a entrega dos dados íntegros. Para isso se fazem necessários mecanismos de detecção de erros. Estes mecanismos utilizam técnicas de redundância. Estas técnicas adicionam *bits* extras no final das informações para facilitar a detecção de erros pelo destinatário. Estes *bits* serão descartáveis tão logo a verificação da transmissão tenha sido realizada. Podemos citar como exemplo de códigos de verificação de integridade: o teste da paridade e a verificação de redundância cíclica (CRC - *Cyclic Redundancy Check*).

No teste da paridade, um *bit* redundante (denominado *bit* de paridade) é adicionado a cada seqüência de dados de tal modo que o número total de uns na seqüência (incluindo o *bit* de paridade) torne-se par ou ímpar. Porém esta abordagem detecta somente erros isolados. Eles podem detectar rajadas de erros se, e somente se, o número total de erros em cada seqüência for ímpar [Forouzan 2006].

O CRC baseia-se numa divisão binária. Um seqüência de *bits*, denominados *bits* de CRC, são acrescentados ao final de cada bloco de dados de maneira a tornar todo o bloco divisível por um número binário predeterminado.

Para o cálculo do CRC utiliza-se o seguinte procedimento:

1. Uma *string* composta por *n* zeros é acrescentada ao bloco de dados;

2. O novo bloco de dados é dividido pelo divisor, usando divisão binária. O divisor deverá ter $n+1$ *bits*. O resto da divisão é o CRC;
3. O CRC de *n bits* calculados substitui os zeros no final do bloco de dados e é transmitido para o receptor;
4. No receptor, o bloco de dados é dividido pelo mesmo número binário.
5. O bloco de dados será assumido intacto se não houver resto da divisão do bloco pelo número binário. Um resto indica que os dados foram corrompidos durante a transmissão, portanto deve ser rejeitado.

O método CRC pode detectar todas as rajadas de erros que afetarem uma quantidade ímpar de *bits* e todas as rajadas de erros cujo comprimento for menor que ou igual ao grau do polinômio gerador (n). Além disso, este método pode detectar, com uma probabilidade muito alta, rajadas de erros cujos comprimentos são maiores do que o grau do polinômio gerador [Forouzan 2006].

Capítulo 3

Sistemas de Identificação por Rádio Freqüência

Identificação por rádio freqüência (RFID) é uma tecnologia de armazenamento e recuperação de dados remotos, através de comunicação sem fio, utilizada em sistemas de identificação automático. Um dispositivo RFID, chamada de *tag*, é normalmente colocado em um objeto, animal ou pessoa que se que deseje identificar. Quando solicitado por um leitor a *tag* retornará informações relacionadas ao objeto.

A necessidade de captura de informações em movimento e a utilização de sistemas de identificação automático em ambientes insalubres estão impulsionando a utilização de sistemas RFID em diversos processos produtivos.

Neste capítulo serão apresentados os principais componentes que formam um sistema RFID, como eles são classificados, quais são os principais padrões relacionados com a tecnologia e suas respectivas especificações. Serão apresentadas ainda as principais vulnerabilidades de segurança da tecnologia e maneiras de evitá-las, quando possível.

3.1 Componentes

Os sistemas RFID são formados, principalmente, por três componentes: *tag*, leitor e um sistema de processamento de dados, conforme mostrado na Figura 3.1.

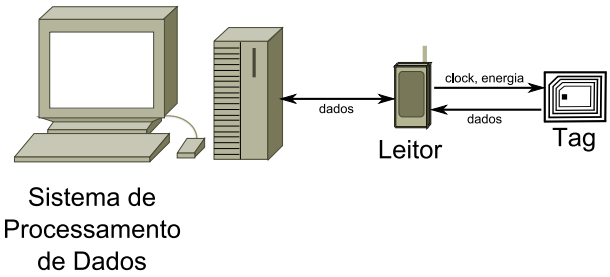

Figura 3.1: Principais componentes de um sistema RFID.

Existem *tags* de diferentes formatos e tamanhos. Elas são compostas principalmente por uma antena e um *microchip*. O *microchip* armazena o código de identificação e infor-

mações adicionais dos objetos que se deseja identificar. Estes dados serão transmitidos para os leitores quando solicitados.

Com relação à utilização de baterias, as *tags* podem ser classificadas em ativas e passivas. As *tags* ativas são alimentadas por uma bateria interna. Elas têm a característica de transmitir o sinal em altas freqüências, obtendo maiores alcances. As *tags* passivas operam sem bateria, sendo sua alimentação fornecida pelo próprio leitor através de energia eletromagnética [Sarma et al. 2002].

O leitor, dependendo da tecnologia utilizada, pode ler ou escrever/ler dados nas *tags*. Ele tipicamente contém módulos de rádio freqüência, uma unidade de controle e uma antena. Além disso, os leitores apresentam interfaces de comunicação (por exemplo, RS 232 ou RS 485) para permitir transmitir os dados lidos para o sistema de processamento de dados. A maioria dos leitores simplesmente capta todos os dados que estão na sua área de interrogação, cabendo ao sistema de processamento de dados organizarem esses dados e transformá-los em informações.

Os componentes básicos de um sistema RFID são combinados geralmente da mesma maneira para todas as aplicações. Todos os objetos a serem identificados são fisicamente marcados com *tags*. O tipo de *tag* utilizada e os dados armazenados variam de aplicação para aplicação. Os leitores continuamente emitem um sinal de interrogatório, criando uma zona de influência. As *tags* que estiverem ao alcance desta zona podem ser lidas, criando um mecanismo para a obtenção de dados associados a objetos físicos. O leitor irá disponibilizar os dados obtidos ao usuário ou a um sistema de processamento de dados.

3.2 Classificação

Os principais critérios para a classificação de sistemas RFID são: freqüência de operação e alcance de leitura [Finkenzeller 2003]. A tecnologia de identificação por rádio freqüência não restringe nenhuma faixa de freqüência para a sua operação. Porém, as faixas de freqüências são geridas por diversas organizações nacionais e internacionais. Deste modo, a freqüência que deverá ser utilizada em uma determinada aplicação, deverá estar de acordo com as permissões dessas organizações. Porém, existem algumas freqüências que podem ser utilizadas para diferentes serviços sem prévia autorização. Estas faixas de freqüências são chamadas de ISM (*Industrial, Scientific and Medical*). As faixas ISM, mostradas na Tabela 3.1, são utilizadas pela maioria das aplicações RFID.

Faixas de freqüências	
LF	125 KHz a 134 KHz
HF	13,56 MHz
UHF	868 MHz a 915 MHz
Microondas	2.5 GHz e 5.8 GHz

Tabela 3.1: Principais faixas de freqüências usadas pelas aplicações da tecnologia RFID.

Com relação ao alcance os sistemas RFID podem ser classificados em: *close coupling*, *remote coupling* e *long-range* [Finkenzeller 2003]. Sistemas RFID com alcance muito

pequeno, tipicamente na região de até 1 cm, são conhecidos como *close coupling*. Para a operação, neste sistema, a *tag* deve ser inserida dentro do leitor ou colocada sobre a sua a superfície. Estes sistemas operam nas faixas de freqüência inferior a 30 MHz. Devido à proximidade entre a *tag* e o leitor ocorre um maior fornecimento de energia. Por isso, microprocessadores com maiores recursos computacionais podem ser utilizados. Estes sistemas são utilizados principalmente em aplicações que estão sujeitas as rigorosas exigências de segurança, mas não exigem um grande alcance, por exemplo, em sistemas de controle de acesso.

Sistemas com leitura e escrita com alcances de até 1 m são conhecidos como *remote coupling*. Estes sistemas correspondem a cerca de 90% de todos os sistemas RFID vendidos atualmente [Finkenzeller 2003]. Operam em freqüências inferiores a 13,56 MHz. Existem diversas aplicações para estes sistemas, tais como identificação de animais e automação industrial.

Sistemas RFID com alcance superior a 1 m são conhecidos como *long-range*. Estes sistemas operam utilizando ondas eletromagnéticas na faixa UHF e microondas. Geralmente, apresentam alcances de 3 m (utilizando *tags* passivas) e 15 m (ou até mesmo maiores utilizando *tags* ativas).

3.3 Padronização

Atualmente, duas organizações internacionais estão propondo padrões e estudos para a tecnologia RFID. São elas: ISO (*International Organization for Standardization*) e EPC-Global. Na seqüência serão apresentados os principais padrões, relacionados à tecnologia RFID, de cada uma destas organizações.

3.3.1 ISO

A ISO vem desenvolvendo padrões que abrangem a estrutura do código de identificação, características físicas (por exemplo, potência do sinal, modulação, tamanho da antena e interferências), protocolos de anticolisão e transmissão, quadros, comandos, interfaces de comunicação, aplicações e métodos de testes. A ISO divide os seus padrões para a tecnologia em três principais aplicações: identificação de animais, gestão de itens em cadeia de abastecimento e cartões de uso geral. Na seqüência serão apresentados os padrões relacionados a cada uma destas aplicações.

Identificação de Animais

- ISO 11784: especifica a estrutura do código de identificação para animais. Este código corresponde a um total de 64 *bits* divididos em código do país e um identificador único para cada país;
- ISO 11785: apresentam os conceitos técnicos para a utilização da tecnologia RFID na identificação de animais. Este padrão define o método de transmissão e as especificações para os leitores.
- ISO 14223: complementa os padrões anteriores, sendo composta por três partes:

1. Interface aérea;
2. Códigos e estrutura de comandos;
3. Aplicações.

Cartões de Uso Geral

Cartões RFID é um caso especial de cartões de identificação definido em ISO 7810. Existem três tipos de cartões RFID que podem ser distinguidos pelo alcance de comunicação: *Close-coupled*, *Proximity* e *Vicinity*. Na seqüência serão apresentados os padrões que especificam estes cartões.

- ISO 10536 (*Close-coupled*): são cartões que operam a uma distância inferior a 10 cm. Este padrão consiste de quatro partes:
  1. Características físicas dos cartões;
  2. Dimensões e localização da antena;
  3. Sinais elétricos e processos de *reset*;
  4. Protocolos de transmissão dos dados.
- ISO 14443 (*Proximity*): são cartões que operam a aproximadamente 10 cm de distância do leitor. A especificação é dividida em quatro partes:
  1. Características físicas dos cartões;
  2. Interferência entre dispositivos;
  3. Protocolos de inicialização e anticolisão;
  4. Protocolos de transmissão dos dados.
- ISO 15693 (*Vicinity*): são cartões com alcance de até 1 m. Este padrão é composto por quatro partes:
  1. Características físicas dos cartões;
  2. Freqüência de operação, interfaces de comunicação e quadros de dados;
  3. Protocolos;
  4. Registro de aplicações.

Gestão de Itens

Os conjuntos de normas da ISO que trata da utilização da tecnologia RFID na gestão de itens são:

- ISO 15961: define os comandos e a sintaxe das mensagens;
- ISO 15962: especifica o formato dos dados;
- ISO 15963: apresenta o identificador único e cria uma entidade administrativa para disponibilizar estes identificadores. Este padrão é dividido em três partes:
  1. Sistema de numeração;
  2. Procedimentos;
  3. Uso de identificadores únicos.

- ISO 18000: especifica as interfaces de comunicação. Este padrão é composto por seis partes:
  1. Parâmetros genéricos;
  2. Parâmetros para interfaces que operam com freqüências abaixo de 135 kHz;
  3. Parâmetros para interfaces que operam com freqüências de 13,56 MHz;
  4. Parâmetros para interfaces que operam com freqüências de 2,45 GHz;
  5. Parâmetros para interfaces que operam com freqüências de 5,8 GHz;
  6. Parâmetros para interfaces que operam na faixa UHF.
- ISO 18001: apresentam as possíveis aplicações da tecnologia na gestão de itens.

Outros protocolos são: ISO 10373 (apresenta métodos para a realização de testes para *smart cards* incluindo a tecnologia RFID) e ISO 10374 (descreve um sistema de identificação automática para *containers* com *tags* ativas operando na faixa de microondas).

Como pode ser observado, os padrões da ISO para a tecnologia não apresentam especificações relacionadas à segurança da informação.

3.3.2 EPCGlobal

A EPCGlobal vem desenvolvendo padrões, para a *Electronic Product Code* (EPC) e EPCGlobal *Network*, com o objetivo de dar suporte ao uso da tecnologia RFID em redes de rastreamento de produtos. Estes padrões visam proporcionar uma maior eficiência em toda a cadeia de abastecimento de produtos, através de troca de informação entre as empresas e os seus parceiros comerciais.

Com objetivo de produzir *tags* de baixo custo, as *tags* EPC apresentam pouca memória e baixo poder computacional. O identificar único de uma *tag* EPC, conhecido como código EPC, pode possuir até 96 *bits* de comprimento. Ele é utilizado para identificar um item específico na cadeia de abastecimento. Para diferenciar as *tags*, a EPCGlobal divide as mesmas em quatro classes. São elas:

- Classe 1: estas *tags*, também chamadas de *tags* de identificação, apresentam:
  - Um identificador EPC;
  - Um identificador de *tag*, chamada de *Tag* ID;
  - Uma função de bloqueio da *tag*;
  - Opcionalmente, um controle de acesso protegido por senha;
  - Opcionalmente, a utilização de memória adicional para dados do usuário.
- Classe 2: conhecida como *tags* com funcionalidades superiores. Elas apresentam as características das *tags* de Classe 1, porém apresentam funcionalidade adicionais. Estas funcionalidades são:
  - Uma extensão no tamanho do identificador da *tag*;
  - Utilização de memória para o usuário;
  - Controle de acesso;
- Classe 3: são as *tags* passivas com baterias, também chamadas de semi-passivas. Estas *tags* apresentam as funcionalidades das *tags* das classes anteriores e apresenta as seguintes características adicionais:

  - Uma fonte de energia que poderá ser utilizada pela *tag* e/ou sensores;
  - Opcionalmente, possuem registro das atividades.

- Classe 4: são *tags* ativas com as seguintes características:
  - Identificador EPC;
  - Uma *tag* ID extendido;
  - Controle de acesso;
  - Fonte de energia;
  - Podem inicializar a comunicação com ou sem um leitor;
  - Pode se comunicar com outras *tags* Classe 4;
  - Opcionalmente, apresenta memória para usuário;
  - Opcionalmente, possuem registro das atividades.

As *tags* EPC são desenvolvidas para aplicações em cadeias de abastecimento e logística. Desta forma, as especificações da EPCGlobal abrangem somente interfaces, protocolos de acesso ao meio físico, protocolos de aplicação nos leitores, redes de informações, formato dos dados de identificação e mensagens de comunicação.

Tags EPC são criadas para as empresas gerirem produtos e não pessoas. Acordos de licenciamento da EPCGlobal proíbe a utilização dos seus padrões para monitoramento e identificação de pessoas, exceto em casos muito específicos e com total transparência, por exemplo, relativas a identificação de pacientes [EPCGlobal 2009]. Desta forma, os padrões da EPCGlobal também não abrangem especificações de segurança da informação.

3.4 Privacidade e Segurança

As *tags* respondem automaticamente as solicitações dos leitores sem nenhuma intervenção humana e, geralmente, não utilizam controle de acesso e não armazenam um histórico das leituras realizadas. Como resultado, as *tags* podem se comunicar com leitores não autorizados sem o conhecimento do usuário.

Devido à falta de especificações de segurança por organizações de padronização, as indústrias vêm apresentando várias técnicas para garantir a segurança da informação na tecnologia. De acordo com os padrões industriais, as *tags* podem ser classificadas (com relação aos seus recursos computacionais) em básicas e com criptografia [Juels 2006].

As *tags* básicas não possuem recurso de criptografia nem bons geradores de números pseudo-aleatórios. Por outro lado, *tags* com criptografia possuem algoritmos criptográficos, geralmente criptografia com chaves simétricas, o que permite a implementação de algoritmos de criptografia e mecanismos de autenticação. Tanto as *tags* básicas quanto as com criptografia visam utilizar a menor quantidade possível de recursos de processamento, bateria e memória. O objetivo destas restrições é permitir a produção de *tags* de baixo custo.

A falta de primitivas de segurança nas *tags* básicas e as restrições de recursos nas *tags* com criptografia criam dificuldades para o desenvolvimento de técnicas que garantam a segurança dos dados. Por outro lado, a falta destas primitivas justifica a análise e desenvolvimento de novos paradigmas para segurança da informação.

Na seqüência serão apresentadas algumas vulnerabilidades de segurança conhecidas, exemplos de ataques bem sucedidos e medidas que estão sendo tomadas para evitar estes ataques.

3.4.1 Ameaças

Alguns ataques a dispositivos RFID se tornaram possíveis devido às características da tecnologia [Rieback, Crispo & Tanenbaum 2006*a*]. Os principais ataques relacionados à tecnologia são: *Sniffing*, Rastreamento Clandestino, *Cloning*, Ataque de *Replay*, Negação de Serviço, RFID *exploits* e Vírus. Na seqüência serão apresentadas de forma resumida as principais características de cada ameaça.

Sniffing

O atacante utiliza leitores não autorizados para capturar informações das *tags*. Estas informações são obtidas através da interceptação de mensagens entre uma *tag* e um leitor autorizado. Este ataque poderá acarretar em graves problemas de privacidade se as *tags* armazenarem informações pessoais dos usuários. Por exemplo, um atacante pode determinar quais os objetos que uma pessoa está carregando (tais como, medicamentos e livros) e informações pessoais.

Rastreamento Clandestino

Neste ataque, o atacante coloca leitores em locais estratégicos para gravar identificadores das *tags*. Estes identificadores são associados com as pessoas que as possuem. Desta forma, o atacante poderá localizar uma determinada pessoa através destes identificadores. Por exemplo, considere um hospital que fornece braceletes RFID aos seus pacientes. Se um atacante colocar leitores espalhados pelo hospital, ele poderá rastrear e localizar os pacientes.

Cloning

As *tags* armazenam códigos, que pode ser copiado como qualquer outro código. Além disso, as *tags* básicas não oferecem nenhum mecanismo de controle de acesso. Desta forma, quando o atacante tiver acesso a uma destas *tags*, ele poderá capturar os dados e copiá-los em uma *tag* clonada. Westhues [Westhues 2005] apresentou esta vulnerabilidade construindo um dispositivo que pode clonar *tags* básicas. Na seqüência veremos um exemplo deste tipo de ataque.

A *Texas Instruments* (TI) fabrica um dispositivo RFID com criptografia embarcada chamado de *Digital Signature Transponder* (DST). O DST é utilizado em sistema de alarme em diversos automóveis (por exemplo, em veículos da Ford e Toyota). O sistema funciona através da utilização de uma *tag* na chave do automóvel. Se o leitor, no interior do automóvel, conseguir realizar a autenticação da *tag*, o sistema de ignição é destravado e o usuário poderá dar a partida no motor. O DST também é utilizado em cartões de pagamento de postos de combustíveis na América do Norte [Juels 2006].

Para a autenticação, o DST utiliza um protocolo de desafio/resposta. Todos os DSTs contêm uma chave secreta k_i. Em resposta a um desafio aleatória (D) de um leitor, o DST executa uma função de encriptação (e) e obtém uma saída ($C = e_k[D]$). O desafio, resposta e a chave secreta possuem 40 *bits*, 24 *bits* e 40 *bits*, respectivamente. A TI não divulgou o algoritmo de encriptação utilizado pelo DST, preferindo a abordagem da "segurança através da obscuridade".

Porém em 2004, uma equipe de pesquisadores da Universidade Johns Hopkins e RSA Laboratório [Bono et al. 2005] apresentaram as vulnerabilidades de segurança dos dispositivos DST. Eles desvendaram o algoritmo de criptografia através de engenharia reversa. Desta forma, eles clonaram *tags* de sistemas de alarme de automóveis, conseguindo "roubar"o seu próprio carro, e comprar gasolina através de um clone de seu próprio cartão de pagamento.

Ataque de *Replay*

Atacantes podem interceptar e retransmitir mensagens RFID. Estas retransmissões têm o objetivo de enganar o sistema que utiliza a tecnologia. Benjamim *et al.* [Heydt-Benjamin et al. 2008] apresentaram um ataque de *replay* em cartões de crédito que utilizam a tecnologia RFID. Para exemplificar um possível ataque, considere que um cliente deseja realizar uma compra em uma loja com um cartão de crédito RFID. Um atacante poderá colocar um leitor posicionado de forma a obter a identificação de uma *tag*. Desta forma, o atacante poderia capturar as informações da *tag* e posteriormente reenviá-las de modo a comprar produtos com o identificador do cartão da vítima.

Negação de Serviço

Este ataque é quando sistemas são impedidos de funcionar corretamente. Este tipo de ataque pode ser desastroso em algumas situações, por exemplo, quando for necessária a leitura de dados médicos em pacientes críticos. Além disso, o combate a este tipo de ataque em sistemas RFID é extremamente difícil, pois o próprio dono da *tag* poderá ser o atacante. Por exemplo, considere um sistema de pagamento de pedágio que utilizam *tags* ativas na identificação dos automóveis. O motorista para não pagar o pedágio poderá remover a *tag*, ou impedir de alguma maneira a sua leitura. A leitura da *tag* pode ser impedida usando a chamada *Faraday Case* ou um ataque de interferência eletromagnética.

RFID *exploits*

Atacantes podem utilizar a tecnologia RFID para explorar vulnerabilidades de segurança de componentes do sistema de processamento dos dados. Quando um leitor realiza a leitura de uma *tag*, ele espera receber informações em um determinado formato. No entanto, um atacante pode escrever dados inesperados em uma *tag*, de modo que durante o seu processamento, estes dados irão comprometer o *software* de processamento. RFID *exploits* tem como alvos componentes específico do sistema (por exemplo banco de dados e servidor *web*). Este ataque faz uso de uma série de ferramentas que incluem *buffer overflows* e inserção de código (por exemplo, inserção de SQL). Exemplos de RFID *exploits*

são apresentados por Suliman *et al.* [Suliman et al. 2008] e Rieback *et al.* [Rieback, Simpson, Crispo & Tanenbaum 2006].

Vírus

Os vírus são programas que criam cópias de si mesmo e se propagam infectando outros dispositivos. Estes vírus podem causar perda de desempenho, alteração dos dados das *tags*, acessar informações confidenciais e monitorar a utilização dos componentes.

Para exemplificar um ataque, considere um aeroporto que realiza o controle de bagagem utilizando *tags* RFID nas malas dos passageiros. Se um viajante malicioso colocar uma *tag* infectada em uma mala, quando a sua bagagem passar pelo sistema de *check-in*, o leitor irá realizar a leitura da *tag* para determinar a rota. A *tag* responderá com um vírus que infecta o banco de dados do aeroporto. Como conseqüência, todas as *tags* produzidas no *check-in* também podem ser infectadas. Quando estas bagagens infectadas forem examinadas em outro aeroporto, o ataque irá se propagar para este novo aeroporto. Dentro de um dia, centenas de banco de dados em vários aeroportos poderão ser infectados. Outros exemplos de ataques são apresentados por Rieback *et al.* [Rieback, Simpson, Crispo & Tanenbaum 2006].

3.4.2 Contra Medidas

Conhecido os principais ataques iremos apresentar as principais contra medidas que podem ser utilizadas para combater estas vulnerabilidades. Dividimos as contra medidas em três classes: as que podem ser utilizadas em *tags* básicas, as que podem ser adotadas em *tags* com criptografia e através do uso do RFID *Firewall*.

Tags Básicas

A maioria das medidas utilizadas nas *tags* básicas incidem sobre o problema de privacidade. As medidas são:

- *Killing*: as *tags* básicas, geralmente, permitem a privacidade das informações através do comando *kill*. Quando uma *tags* perde a sua funcionalidade (por exemplo, após a compra de uma roupa por um cliente), ela deverá receber o comando *kill* de um leitor. Desta forma, a *tag* passa a ficar permanentemente inoperante. Para evitar a destruição por leitores não autorizados, este comando é protegido por senha (chamada de PIN) composta por 32 *bits*. O leitor deverá informar esta senha para deixar a *tag* inoperante. Uma vez a *tag* inoperante a privacidade do usuário estará mantida, pois a *tag* não irá responder a nenhuma leitura.
- Remoção: corresponde a retirada da *tag* do objeto que se desejava identificar. Esta medida tem o mesmo resultado do comando *kill*, porém na maioria das vezes se faz necessário a intervenção humana. Esta intervenção poderá resultar em maiores gastos com funcionários e a perda da automatização do processo produtivo.
- *Sleep*: o comando *kill* e a remoção da *tag* garantem a privacidade do consumidor, mas eliminam todos os benefícios do uso da tecnologia. Por exemplo, se o consumidor desejar descobrir a autenticidade de um produto após a compra, ele poderá

utilizar a *tag* e uma base de dados para localizar os dados referentes a este produto. Além disso, em algumas aplicações o comando *kill* e a remoção não poderão ser utilizadas. Por exemplo, em uma biblioteca que identifica cada livro através de uma *tag*. Se após a realização de um empréstimo a *tag* for destruída ou removida, quando o livro retornar não seria possível a sua identificação. Para solucionar estes problemas, Juels [Juels 2006] propôs a utilização de comandos de ativação e inativação (chamado de *sleep*). Quando a *tag* deixar o ambiente seguro, ela é colocada no estado desabilitado. Quando a *tag* retornar a um ambiente seguro, ela é reativada. Desta forma, a privacidade é garantida. Para evitar a desativação por leitores não autorizados, este comando também deverá ser protegido por senha.

- Conjunto de Identificadores: as soluções apresentadas até agora só visam garantir a privacidade da informação. Uma maneira de combater o ataque de rastreamento clandestino é através da utilização de um conjunto de identificadores para cada *tag* [Juels 2005]. Assim, para cada leitura a *tag* selecionará aleatoriamente um dos seus identificadores. Neste caso, se o atacante desejar realizar o rastreamento, ele terá que associar um conjunto de identificadores a um determinado usuário. Quanto maior for este conjunto de identificadores maior será a dificuldade da realização do ataque.

Tags com Criptografia

As *tags* com criptografia, geralmente, visam combater os ataques de *sniffing*, rastreamento clandestino, *clonning* e ataque de *replay*. Para combater estes ataques dois serviços de segurança são implementados. São eles:

- Criptografia: como foi apresentado no Capítulo 2, criptografia é um conjunto de métodos e técnicas que convertem um texto legível em um texto ilegível, sendo possível mediante o processo inverso recuperar as informações originais. O objetivo da criptografia é garantir a privacidade das informações. Entretanto, se utilizarmos chaves dinâmicas podemos também combater o rastreamento clandestino e o ataque de *replay*.
- Autenticação: verificação da identidade do leitor ou da *tag*. Para isso, são utilizadas chaves secretas compartilhadas entre eles. A autenticação é aplicada na solução do ataque de *clonning*. A autenticação neste tipo de *tag* ocorre geralmente baseada no paradigma desafio/resposta. Neste paradigma, para que o leitor autentique a *tag*, primeiro, ele irá enviar um desafio para a *tag*. Esta usará o desafio e sua chave secreta como entradas em um determinado algoritmo criptográfico, resultando na resposta ao desafio. A resposta será então enviada para o leitor. Se a resposta estiver correta a *tag* é autenticada. Este tipo de autenticação também combatem o ataque de *replay*. Pois, ao verificar a resposta, o leitor sabe que a *tag* possui a chave secreta e conseguiu responder a um desafio recente.

RFID *Firewall*

A utilização de primitivas criptográficas e autenticação resultam em custos adicionais na produção das *tags*. Devido a este fato, as empresas atualmente estão produzindo prin-

cipalmente *tags* básicas. Como foi visto, estas *tags* não garantem a privacidade e nem a segurança na comunicação.

Para garantir a privacidade e a proteção dos dados, o usuário poderá utilizar um dispositivo de proteção, chamado de RFID *firewall*. Com este dispositivo, o usuário poderá definir um conjunto de regras que determinarão quais, onde e por quem as *tags* poderão serem lidas. Um exemplo deste dispositivo é o RFID *Guardian* [Rieback et al. 2005].

A *firewall* inspeciona e restringi a comunicação entre o leitor e as *tags*, por exemplo, simulando uma *tag* sob o seu controle quando necessário. Estes dispositivos possuem recursos computacionais (por exemplo, GPS e acesso a internet) e baterias, que resultam em uma maior capacidade de processamento, criação de políticas de privacidade sofisticadas e registros de leituras.

Capítulo 4

Protocolos de Criptografia Neural e Diffie-Hellman

Conforme apresentado no Capítulo 2, existem dois tipos de algoritmos criptográficos, são eles: chave simétrica e chave pública. Os algoritmos com chave simétrica requerem menos recursos computacionais [Stallings 2008]. Por este motivo, os algoritmos de criptografia com chave simétrica são mais indicados para sistemas RFID. Entretanto, se faz necessário a criação, distribuição e administração das chaves criptográficas.

Foi apresentado ainda no Capítulo 2 as três possíveis soluções para o problema de distribuição de chaves, são elas: canal seguro, criptografia com chave pública e protocolo de gerenciamento de chaves. A abordagem do canal seguro não se aplica a sistemas RFID, pois anularia a automatização proporcionada pelo sistema. A utilização de criptografia com chave pública não se torna interessante devido a necessidade de elevados recursos computacionais. Este fato resultaria num encarecimento do sistema RFID.

Portanto, a principal solução para o problema de distribuição de chaves em sistemas RFID é a utilização de protocolos de gerenciamento de chaves. Nesta abordagem, um algoritmo de troca de chaves é utilizado por duas entidades. Este algoritmo basicamente troca mensagens ao longo de um canal público. As entidades que participam da troca de mensagens são capazes de converter as mensagens trocadas em uma chave secreta. Esta chave é então usada em algum algoritmo de criptografia com chave simétrica.

Na seqüência serão apresentados, resumidamente, dois protocolos de gerenciamento de chaves, são eles: Criptografia Neural e Diffie-Hellman.

4.1 Criptografia Neural

Alguns pesquisadores vêm analisando a utilização de algoritmos estocásticos, principalmente redes neurais artificiais, em criptografia e criptoánalise. Aplicações típicas abrangem criptografia com chave pública [Yee & de Silva 2002], distribuição de chaves simétricas [Guerreiro & de Araujo 2006] [Kinzel & Kanter 2002], criptoánalise [Albassal & Wahdan 2004] e geração de números pseudo-aleatórios [Chan & Cheng 1998] [Karras & Zorkadis 2003].

Na seqüência serão apresentados alguns conceitos de redes neurais artificiais, uma arquitetura neural utilizada no protocolo de Criptografia Neural, ataques relacionados a

este protocolo e suas possíveis contra medidas.

4.1.1 Redes Neurais Artificiais

Uma rede neural artificial consiste de unidades de processamento simples e altamente interconectadas, chamadas de neurônios. Estes neurônios são formados por uma função de ativação e vetores (entrada, pesos e saída), conforme mostrado na Figura 4.1. Cada neurônio recebe sinais de entrada e produz um sinal de saída. Os pesos têm a função de determinar a influência de cada entrada na saída da rede. Cada neurônio processa os dados através da operação de um produto interno entre os vetores entrada e pesos. O resultado é então calculado através de uma função de ativação.

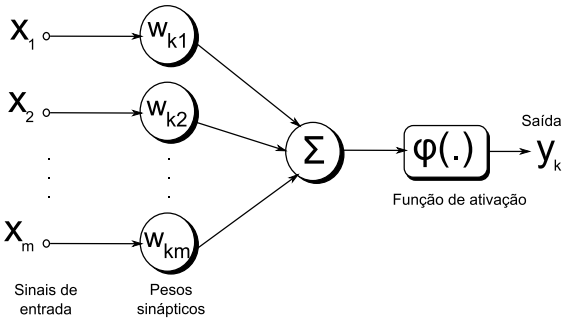

Figura 4.1: Modelo de um neurônio artificial.

Para construirmos uma rede neural teremos que decidir, inicialmente, quantos neurônios deverão ser utilizados e como eles serão conectados. Em outras palavras, teremos que definir a arquitetura da rede. Posteriormente, decidimos qual o algoritmo de aprendizado deverá ser usado. Após estas definições, treinamos a rede. O processo utilizado para treinar as redes neurais é chamado de algoritmo de aprendizado. Este algoritmo está intimamente ligado à maneira pela qual os neurônios de uma rede neural estão estruturados. A função do algoritmo de aprendizado é modificar os pesos sinápticos da rede, de acordo com a resposta da rede para um conjunto de exemplos de treinamento.

Um exemplo de arquitetura é a rede alimentada diretamente com múltiplas camadas. Nesta arquitetura, os neurônios estão organizados na forma de camadas. Os neurônios da camada de entrada fornecem o vetor de entrada, que constituem dos sinais de entrada aplicados aos neurônios da segunda camada. Os sinais de saída da segunda camada são utilizados como entrada para a terceira camada, e assim por diante para o resto da rede. A função dos neurônios das camadas ocultas é intervir entre a entrada externa e a saída da rede, tornando a rede capaz de extrair estatísticas de ordem elevada [Haykin 1998]. O conjunto de sinais de saída dos neurônios da camada de saída constitui a resposta global da rede para o padrão de ativação fornecido pelos neurônios da camada de entrada.

A rede neural é dita totalmente conectada se cada um dos neurônios de uma camada está conectado a todos os neurônios da camada adjacente. Entretanto, se alguns das conexões sinápticas estiverem faltando, dizemos que a rede é parcialmente conectada. Um

exemplo de uma rede neural de múltiplas camadas alimentada adiante e parcialmente conectada é ilustrado na Figura 4.2. Esta arquitetura é chamada de *tree parity machine* e será apresentada na seqüência.

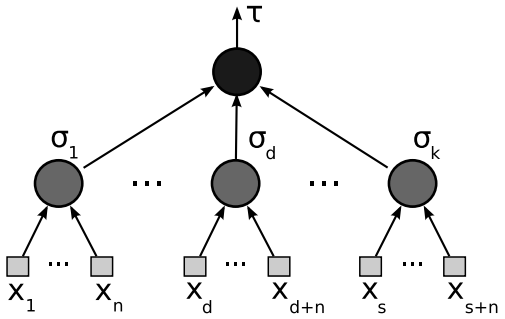

Figura 4.2: Arquitetura *tree parity machine*.

Arquitetura *Tree Parity Machine*

Esta arquitetura consiste de três camadas (entrada, escondida e saída). A camada de entrada, escondida e saída possuem $k*n$, k, e 1 neurônio, respectivamente. Cada neurônio da camada escondida possui n entradas, uma saída e apresenta a função sinal como função de ativação.

Os valores das entradas da rede neural são binários,

$$x_j \in \{-1, +1\},$$

e os pesos apresentam valores discretos entre $-l$ e $+l$, ou seja,

$$w_{i,j} \in \{-l, -l+1, ..., +l\},$$

onde o índice $i = 1, ..., k$ representa o i-ésimo neurônio escondido e $j = 1, ..., n$ o j-ésimo elemento do vetor de entrada.

A saída do i-ésimo neurônio escondido é definido pela fórmula:

$$\sigma_i = sgn\left(\sum_{j=1}^{n} w_{i,j} x_j\right), \tag{4.1}$$

onde,

$$sgn(\zeta) = \begin{cases} 1 & \text{se } \zeta \geq 0 \\ -1 & \text{caso contrário.} \end{cases} \tag{4.2}$$

Então a saída total da rede (τ) é dada pelo produto das saídas obtidas pelos neurônios escondidos, isto é,

$$\tau = \prod_{i=1}^{k} \sigma_i. \tag{4.3}$$

O aprendizado mútuo de duas ou mais *tree parity machine* tem o objetivo de obter a sincronização. Para isso se faz necessário que eles recebam entradas comuns e informem as suas saídas. Ajustando os pesos discretos, de acordo com uma determinada regra de aprendizado, as redes irão obter a sincronização em um número finito de passos [Kinzel & Kanter 2002].

Para exemplificarmos o processo de aprendizado considere duas redes neurais A e B que desejam obter a sincronização. A sincronização é geralmente entendida como a capacidade de objetos diferentes formarem um regime de operação comum, devido à interação ou imposição [Osipov et al. 2007]. Neste caso, a sincronização corresponde a tornar os pesos das redes neurais iguais.

O processo de aprendizado se inicia com a atribuição de valores aleatórios para os pesos das duas redes neurais. Em cada instante de tempo, um vetor de entrada aleatório (X) é gerado e as saídas (τ^Ae τ^B) são calculas.

Se $\tau^A \neq \tau^B$, os pesos não serão modificados. Caso contrário ($\tau^A = \tau^B$), a seguinte regra de aprendizado deverá ser aplicada:

$$w_{i,j} = g(w_{i,j} + x_j). \tag{4.4}$$

A função g(ζ) é para garantir que os pesos permaneçam no intervalo de $-l$ e l.

$$g(\zeta) = \begin{cases} sgn(\zeta)l & \text{se } |\zeta| > l \\ \zeta & \text{caso contrário.} \end{cases} \tag{4.5}$$

Somente os pesos da camada escondida que apresentarem a sua saída igual à saída da rede ($\sigma_i = \tau$) serão atualizados. Devido a esta restrição, é impossível dizer quais os pesos foram atualizados sem conhecer a representação interna da rede, isto é, os valores da saída dos neurônios da camada escondida ($\sigma_1, \sigma_2, ..., \sigma_k$). Esse recurso é especialmente necessário para impedir que outra rede, que não participe do processo de sincronização, possa também obter a sincronização.

Após a atualização dos pesos, concluída uma época de treinamento, todo o processo deverá ser repetido até que os pesos das redes A e B sejam iguais ($W_A = W_B$). Se continuar o processo de treinamento mesmo após as redes estiverem sincronizadas, as mesmas permanecerão sincronizadas, pois, as entradas e os pesos de ambas as redes serão idênticos.

O algoritmo de aprendizado é baseado na competição entre forças aleatórias atrativas e repulsivas. Por exemplo, considere duas redes A e B recebendo a mesma entrada (X) e obtendo a mesma saída (τ). Os pesos do neurônio (i) que apresentarem a saída (σ_i) igual à saída da rede (τ) 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 e W^B) serão atualizados. Como os pesos 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 l será atribuído ao mesmo o valor limitante. Esta atribuição faz 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, de uma das redes, serão atualizados. Até mesmo pesos já sin-

cronizados 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 e deseje obter a sincronização, a probabilidade de passos repulsivos será sempre maior [Ruttor 2007].

A principal aplicação para esta estrutura neural é a geração de chaves criptográficas. Na seqüência será apresentado um protocolo proposto por Kinzel e Kanter [Kinzel & Kanter 2002] que utiliza a arquitetura *tree parity machine*.

4.1.2 Protocolo de Geração de Chaves

Neste protocolo as entidades, que desejam se comunicar de forma segura, utilizam *tree parity machines* com os mesmos atributos. Estas estruturas, que correspondem aos parâmetros k, l e n, são de conhecimento público. A sincronização das redes neurais é alcançada através das etapas apresentadas no Algoritmo 1.

Algoritmo 1 Protocolo de distribuição de chaves neurais

```
1: Determinar os parâmetros das redes neurais;
2: Os vetores de pesos das redes neurais são inicializadas aleatoriamente;
3: repeat
4:     Entradas são geradas;
5:     As saídas das redes são calculadas;
6:     Os valores das saídas das redes são tornados públicos;
7:     if os valores das saída das redes forem iguais then
8:         Os pesos serão atualizados;
9:     end if
10: until que os pesos sejam iguais
```

Após as redes obterem a sincronização, as mesmas deverão utilizar os seus pesos como chave secreta em um algoritmo de criptografia de chave simétrica.

Um atacante deverá obter a sincronização com as *tree parity machines* se ele desejar descobrir as chaves de criptografia e ter acesso as informações. Porém, a falta de conhecimento da representação interna das redes é o principal problema para o atacante. Como o movimento dos pesos depende da representação interna, se faz necessário que o atacante descubra os estados dos neurônios escondidos corretamente. Entretanto, existem 2^{k-1} diferentes representações internas que levam ao mesmo valor de saída τ.

A maioria dos ataques conhecidos utiliza esta abordagem. Porém, poderão surgir outros ataques que consigam quebrar a segurança deste protocolo. No entanto, este risco existe para todos os algoritmos criptográficos.

Ataques

Como os parâmetros da rede, os valores de entrada e as saídas são de conhecimento público, um atacante poderá utilizar estas informações para tentar também obter a sincro-

nização. Existem três tipos de ataques de sincronização, são eles:

- Ataque Simples: neste tipo de ataque [Kanter et al. 2002], um atacante (E) treina uma terceira *tree parity machine* com as entradas e saídas obtidas das duas outras redes neurais (A e B). O atacante possui uma rede neural com a mesma estrutura (parâmetros k, l e n) e utiliza a mesma regra de aprendizado. Como o atacante não sabe quais foram os valores iniciais dos pesos de A e B, ele também irá inicializar os seus pesos com valores aleatórios. Os exemplos de treinamento podem ser obtidos através da interceptação de mensagens transmitidas entre A e B. Para cada passo de treinamento o atacante calcula a saída de sua rede neural e atualiza os seus pesos através da equação:

$$w_{i,j}^{E} = g(w_{i,j}^{E} + x_{j}^{A/B}). \tag{4.6}$$

  Como podemos observar, na equação 4.6, o atacante atualiza os seus pesos baseados na entrada e na saída das outras redes, sem levar em consideração a saída da sua rede e nem a sua representação interna.
- Ataque Geométrico: o atacante leva em consideração a saída da sua rede e a sua representação interna. Este ataque é o ataque mais bem sucedido, entre os ataques que utilizam apenas uma *tree parity machine* [Ruttor 2007]. Este ataque é similar ao ataque simples, porém neste caso só é aplicada a regra de aprendizado quando a saída de sua rede for igual à saída das redes que estão sincronizando ($\tau^{E} = \tau^{A/B}$). Porém, no caso de $\tau^{E} \neq \tau^{A} = \tau^{B}$ o atacante não poderá impedir a atualização dos pesos das outras redes. Ao invés disso, o atacante tentará corrigir a sua representação interna, para obter a mesma saída das outras redes, antes de aplicar a atualização dos pesos. Para isso, o atacante troca o valor da saída total (τ^{E}) com o valor de saída de um neurônio da camada escondida.
- Ataque de Maioria: o atacante tenta melhorar a capacidade de predição das representações internas utilizando um conjunto de *m tree parity machines*, em vez de uma única rede utilizada pelo ataque geométrico. No início do processo de sincronização, os pesos de todas as m redes são escolhidos aleatoriamente. Quando a saída de uma determinada rede τ_{i}^{E} for diferente de $\tau^{A/B}$, o atacante tenta corrigir a sua representação interna da mesma forma que o ataque geométrico. Após a correção, o atacante seleciona entre as m representações internas a mais comum. Esta representação será adotada por todas as redes na aplicação da regra da aprendizagem. Entretanto, as atualizações das redes de forma idêntica aumentam a correlação. Este fato reduz a eficiência do ataque. O problema da correlação será minimizado se o atacante utilizar o ataque de maioria e geométrico de forma alternada [Shacham et al. 2003].

Contra Medidas

Apresentado os ataques, agora iremos mostrar medidas que visam combater tais ataques. Para isso, uma característica especial do processo de sincronização precisa ser levada em consideração. O processo de treinamento é finalizado quando A e B obtiverem a

sincronização. Deste modo, o número de exemplos de treinamento para um atacante é limitado. Conseqüentemente, o atacante só obterá sucesso se ele descobrir a representação interna de A ou B antes do término do processo de sincronização.

Para dificultar a realização do ataque algumas características da rede neural deverão ser avaliadas. São elas [Ruttor 2007]:

- Quando $k \leq 3$, o tempo de sincronização de A e B é diretamente proporcional a l^2. Entretanto, o tempo de sincronização de um atacante cresce exponencialmente com o aumento de l, exceto para o ataque geométrico. Ou seja, quando $k \leq 3$, as redes A e B tendem sempre a sincronizar antes da rede atacante, exceto no caso do ataque geométrico;
- Quanto maior o valor de k, maior será a probabilidade de sucesso de um ataque simples. Além disso, neste caso o esforço para A e B obterem a sincronização cresce exponencialmente com o aumento de l, o que poderá resultar em uma grande quantidade de exemplos de treinamento. Em outras palavras, altos valores para a variável k não deverão ser utilizadas pois aumenta o tempo de para geração das chaves e facilita o ataque simples;
- No caso do ataque geométrico, a probabilidade de sucesso aumenta com a diminuição de k, porém diminui com o aumento de l, quando $k \geq 3$. Isto é, k deverá ter um valor maior ou igual a 3 para combater o ataque geométrico.

Desta forma $k = 3$ é a escolha ótima para a aplicação da *tree parity machine* na geração de chaves de criptografia [Ruttor 2007]. Deste modo, o nível de segurança desejado é obtido através da variável l, sendo o sistema considerado seguro se $l \rightarrow \infty$. Na prática, para conseguir qualquer nível de segurança desejado, basta aumentar o valor de l.

4.2 Protocolo de Diffie-Hellman

A finalidade do protocolo é permitir que dois usuários negociem uma chave com segurança, a qual pode então ser usada para a subseqüente criptografia das mensagens. O protocolo de Diffie-Hellman depende, para a sua segurança, da dificuldade de se calcular logaritmos discretos. Resumindo, podemos definir o logaritmo discreto da seguinte maneira. Primeiro, definimos uma raiz primitiva de um número primo p como aquela cujas potências módulo p geram todos os inteiros de 1 até $p-1$. Ou seja, se g é uma raiz primitiva do número primo p, então os números

$$g \mod p, g^2 \mod p, \ldots, g^{p-1} \mod p \tag{4.7}$$

são distintos e consistem nos inteiros de 1 até $p-1$ em alguma permutação.

Em outras palavras, para qualquer inteiro b, sendo $b < p$, e uma raiz primitiva g do número primo p, podemos encontrar um expoente exclusivo i tal que:

$$b = g^i \mod p. \tag{4.8}$$

O expoente i é referenciado como o logaritmo discreto de b na base g mod p. Expressamos esse valor como $dlog_{g,p}(b)$. Beth [Beth et al. 1992] apresentou um algoritmo

para o calculo do logaritmo discreto de um número primo. Porém, a complexidade deste algoritmo é da ordem de:

$$e^{((lnp)^{1/3}(ln(lnp))^{2/3})} \tag{4.9}$$

que é intratável para números p grandes.

Se a exponenciação fosse feita sobre os inteiros e depois reduzida módulo n, os valores intermediários seriam extremante grande. Por exemplo, considere $g = 3$, $p = 233$ e $i = 50$, obtemos:

$$\begin{aligned} b &= 3^{50} \ \texttt{mod}\ 233 \\ b &= 717.897.987.691.852.588.770.249 \ \texttt{mod}\ 233 \\ b &= 218 \end{aligned}$$

Felizmente, podemos utilizar uma propriedade da aritmética modular:

$$[(\alpha \ \texttt{mod}\ p)\, x\, (\beta \ \texttt{mod}\ p)] \ \texttt{mod}\ p = (\alpha\, x\, \beta) \ \texttt{mod}\ p. \tag{4.10}$$

Assim, podemos reduzir resultados intermediários.

Generalizando, suponha que queiramos encontrar o valor a^b mod p, com a e b inteiros positivos. Se expressarmos b como um número binário $b_k b_{k-1} \ldots b_0$, então temos:

$$b = \sum_{b_i \neq 0} 2^i \tag{4.11}$$

Portanto,

$$a^b = a^{\left(\Sigma_{b_i \neq 0} 2^i\right)} = \prod_{b_i \neq 0} a^{(2^i)} \tag{4.12}$$

$$a^b \ \texttt{mod}\ p = \left[\prod_{b_i \neq 0} a^{(2^i)}\right] \ \texttt{mod}\ p = \left(\prod_{b_i \neq 0} \left[a^{(2^i)} \ \texttt{mod}\ p\right]\right) \ \texttt{mod}\ p \tag{4.13}$$

Então, podemos utilizar o Algoritmo 2 para calcularmos a^b mod p.

4.2.1 Protocolo de Geração de Chaves

Existem dois números conhecidos publicamente: um número primo p e um inteiro g. Lembrando que g é uma raiz primitiva de p. Suponha que Alice e Bob desejem negociar uma chave secreta. Inicialmente, Alice seleciona um inteiro aleatório $X_A < p$ e calcula $Y_A = g^{X_A} mod\, p$. Da mesma maneira, Bob seleciona independentemente um inteiro aleatório $X_B < p$ e calcula $Y_B = g^{X_B} mod\, p$. Alice e Bob mantêm os valores de X privados e tornam os valores de Y públicos. Então, Alice calcula a chave como $K = (Y_B)^{X_A}$ mod p e Bob calcula a chave como $K = (Y_A)^{X_B}$ mod p. Esses dois cálculos produzem resultados idênticos. A Figura 4.3 ilustra as etapas necessárias para a obtenção da chave.

Pela regra da aritmética modular, poderemos provar que as chaves serão idênticas. Vejamos:

Algoritmo 2 Algoritmo para calcular $a^b \bmod p$

```
1: inputs: a, b e p; {O inteiro b é expresso como um número binário (b_k, b_{k-1}...b_0)};
2: c = 0;
3: f = 1;
4: for i = k; i > 0; i-- do
5:     c = 2c;
6:     f = (fxf) mod p;
7:     if b_i = 1 then
8:         c++;
9:         f = (fxa) mod p
10:    end if
11: end for
12: return f
```

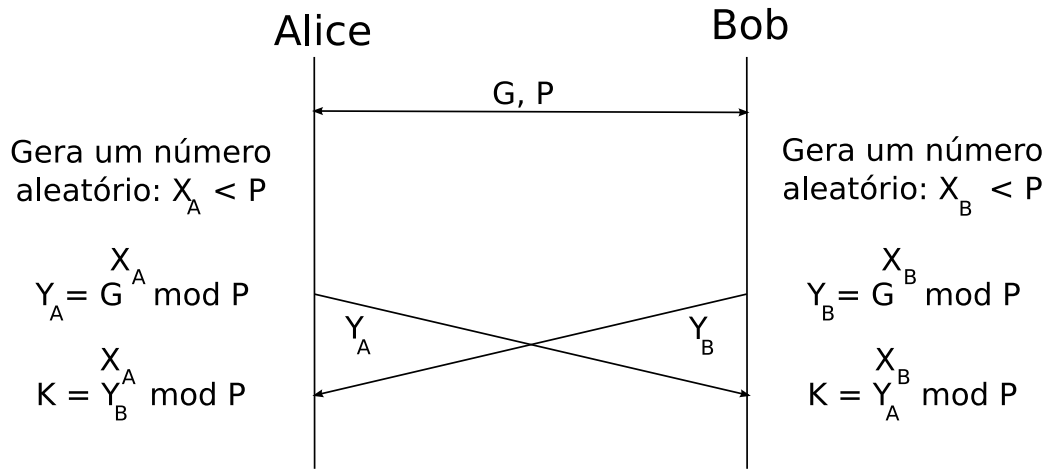

Figura 4.3: Acordo de chaves Diffie-Hellman.

$$\begin{aligned} K &= (Y_B)^{X_A} \bmod p \\ &= (g^{X_B} \bmod p)^{X_A} \bmod p \\ &= (g^{X_B} \bmod p)^{X_A} \bmod p \\ &= (g^{X_B})^{X_A} \bmod p \\ &= g^{X_B X_A} \bmod p \\ &= (g^{X_A})^{X_B} \bmod p \\ &= (g^{X_A} \bmod p)^{X_B} \bmod p \\ &= (Y_A)^{X_B} \bmod p. \end{aligned}$$

Como X_A e X_B são privados, um adversário só tem acesso a p, g, Y_A e Y_B. Deste modo, Eve é obrigado a calcular um logaritmo discreto para determinar a chave.

A segurança do protocolo está em definir um p grande, pois a segurança do sistema é baseado na dificuldade de fatorar números primos grandes. Além disso, Eve não teria tempo hábil porque Alice e Bob trocam a chave toda vez que eles estabelecem a comunicação.

Contudo, o protocolo de Diffie-Hellman é inseguro contra o ataque *man-in-the-middle*. Neste ataque, Eve não precisa determinar os valores de X_A e X_B para atacar o protocolo. Ela pode ludibriar Alice e Bob criando duas chaves: uma entre ela e Alice e outra entre ela e Bob. Portanto, deverá ser incluído algum tipo de autenticação para solucionar este ataque. Este serviço irá permitir que Alice e Bob verifiquem se eles estão realmente interagindo uns com os outros e não com Eve.

Capítulo 5

Implementação e Análise de Desempenho dos Protocolos em FPGA

A tecnologia RFID está sendo utilizada em diferentes aplicações da sociedade moderna, tais como: controle de acesso, aplicações médicas, eventos esportivos, identificação de animais, pagamento automatizado de pedágios e no transporte público. Porém, estas aplicações podem resultar em riscos a segurança e privacidade dos usuários.

Pesquisadores vêm propondo soluções para minimizar estes riscos, uma vez que as normas internacionais, relacionadas à tecnologia RFID, não abrangem especificações de criptografia, autenticação e gerenciamento de chaves. As dificuldades encontradas para a concepção de mecanismos de segurança são devidas a limitações na capacidade computacional, espaço de armazenamento e fornecimento de energia das *tags* RFID.

Atualmente, um dos principais desafios de segurança da informação em sistemas RFID trata-se do gerenciamento de chaves criptográficas. Entre os protocolos de geração de chaves presentes na literatura se destacam os de Criptografia Neural [Kinzel & Kanter 2002] e Diffie-Hellman [Diffie & Hellman 1976]. Neste capítulo iremos avaliar o desempenho destes protocolos na geração de chaves em sistemas RFID. Para os testes foi desenvolvido uma plataforma em FPGA (*Field-Programmable Gate Array*) com o processador embarcado Nios II.

Na seqüência será apresentada a plataforma de testes, as características da tecnologia FPGA e do processador Nios II, a descrição da implementação dos protocolos de Criptografia Neural e Diffie-Helman e os resultados dos testes realizados.

5.1 Plataforma de Testes

A computação reconfigurável é uma tecnologia que se baseia em dispositivos lógicos reprogramáveis. Aplicações que utilizam esta tecnologia podem atingir um desempenho elevado e, ao mesmo tempo, fornecer flexibilidade de programação. A computação reconfigurável está se tornando um novo paradigma para o desenvolvimento de *hardware* nos últimos anos.

As FPGAs (*Field-Programmable Gate Arrays*) são dispositivos típicos de *hardware* utilizados em computação reconfigurável. Estes dispositivos são formados por blocos funcionais interconectados. Estes blocos podem ser programados, e reprogramados, para

executar funções lógicas definidas pelo usuário. A tecnologia FPGA tem evoluído significativamente, alcançando elevados níveis de densidade, desempenho e baixo custo. Esses avanços tornam a tecnologia FPGA cada vez mais competitiva com relação à tecnologia ASIC (*Application Specific Integrated Circuit*), a qual por muitos anos tem liderado a fabricação de dispositivos.

O aumento da capacidade de processamento e a característica de reprogramação motivaram a utilização desta tecnologia no desenvolvimento do protótipo da plataforma de testes. Esta plataforma foi desenvolvida visando avaliar o tempo de processamento inerente aos protocolos de distribuição de chaves testados. Para mensurarmos o tempo total dos protocolos, se faz necessário a utilização de transmissores e receptores de rádio freqüência. Deste modo, o tempo total corresponde ao tempo de processamento mais o tempo de comunicação. A avaliação do tempo total esta fora do escopo deste trabalho.

A plataforma utilizada para avaliação dos protocolos consiste de um processador Nios II, uma memória SRAM (512 KB), um *Timer* e um componente JTAG UART, conforme ilustrado na Figura 5.1.

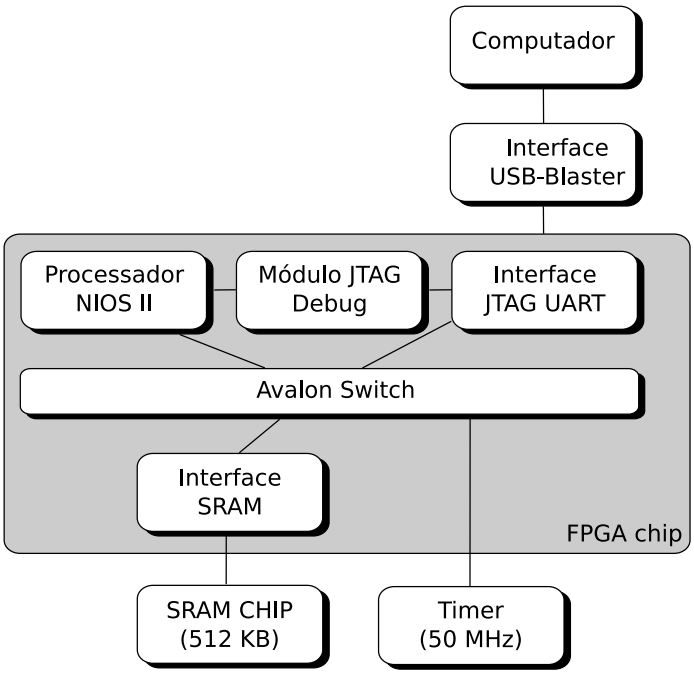

Figura 5.1: Plataforma de testes.

O Nios II consiste de um processador RISC de 32 *bits*, desenvolvido pela Altera® para atender uma grande escala de dispositivos embarcados. As principais características deste processador são: conjunto de instruções e espaço de endereçamento de 32 *bits*; 32 registradores de propósito geral; 32 fontes de interrupções externas; instruções dedicadas ao cálculo de multiplicações com 64 *bits* e 128 *bits*; instruções ponto-flutuante; acesso a uma variedade de periféricos *on-chip* e interfaces para acesso a memórias e periféricos *off-chip*. O Nios II foi escolhido porque ele oferece um curto ciclo de desenvolvimento e um completo ambiente de desenvolvimento integrado.

O componente JTAG UART é responsável pela interface entre o processador Nios II e um computador. Esta interface se faz necessário para a programação dos protocolos na plataforma e a aquisição dos dados necessários para a realização dos testes.

A plataforma foi embarcada na placa DE2, introduzida no mercado pela Altera. Entre os componentes da placa foi utilizado o FPGA (Cyclone II EP2C35F672C6), *clock* (50 MHz), chip de memória (SRAM) e a interface USB-Blaster. A Tabela 5.1 apresenta a quantidade de recursos utilizados pela plataforma e os recursos disponíveis pelo FPGA.

Recursos	Utilizados	Disponíveis
Elementos lógicos	1.746	33.216
Pinos	45	475
Bits de memórias	10.240	483.840

Tabela 5.1: Recursos utilizados pela plataforma embarcada e disponíveis pelo FPGA Cyclone II EP2C35F672C6.

Os protocolos foram escritos na linguagem C e compilado com o compilador Altera Nios II C2H *Compile*. Esta linguagem foi escolhida por ser uma linguagem de alto nível e permitir uma maior portabilidade em relação às linguagens Assembly, VHDL e Verilog.

5.2 Implementações dos Protocolos

Na modelagem dos protocolos na plataforma foram definidas duas entidades (leitor e *tag*) que se comunicam através de mensagens. As mensagens foram transmitidas através de referências a posições de memórias (ponteiros). Lembrando que a memória foi compartilhada entre as duas entidades.

Na seqüência serão apresentadas as principais características de implementação dos dos protocolos de Criptografia Neural e Diffie-Hellman.

5.2.1 Criptografia Neural

O protocolo de Criptografia Neural foi implementado em conjunto com o algoritmo de criptografia com chave simétrica AES. A utilização de um algoritmo de criptografia se fez necessário para a realização dos testes de sincronização das redes neurais.

No protocolo de Criptografia Neural, a *tag* e o leitor utilizam redes neurais (*tree parity machines*) com os mesmos atributos (k, l e n). O processo de geração de chaves começa com a atribuição de valores aleatórios aos pesos destas redes neurais. 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, saída obtida (τ^L) e uma seqüência cifrada de *bits*, conforme mostrado na Figura 5.2. Esta seqüência é obtida cifrando uma variável do sistema chamada TS. Nesta criptografia é utilizada 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.

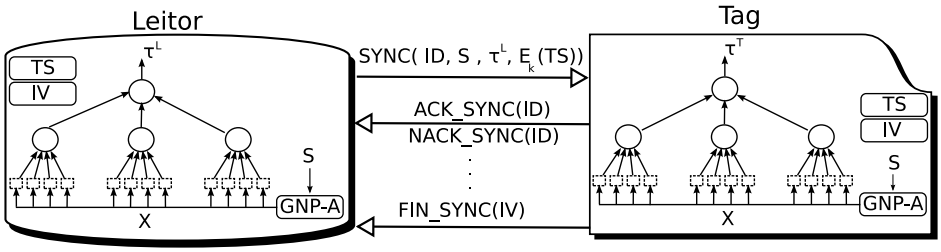

Figura 5.2: Mensagens utilizadas pelo protocolo de Criptografia Neural implementado.

Após o envio do quadro SYNC, o leitor dispara um cronômetro e fica aguardando a resposta da *tag*. Caso a *tag* não responda até um determinado tempo limite, o leitor reinicia o processo de sincronização, ou seja, cria um novo quadro de sincronização. Entretanto, o leitor só enviará uma nova mensagem se o número de tentativas não exceder certo limite. Além disso, o leitor deverá detectar a presença da *tag* antes de enviar a nova mensagem. Esta restrição se faz necessário para evitar que o leitor envie quadros de sincronização quando a *tag* não estiver mais presente.

Quando a *tag* receber a mensagem de sincronização, ela deverá 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 escolher aleatoriamente 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.

Ao término do processo de sincronização, o leitor e a *tag* deverão extrair a chave de criptografia dos neurônios. O tamanho do vetor de chaves geradas (δ) é dado pela fórmula:

$$\delta = \varepsilon * k * n \tag{5.1}$$

onde, ε, k e n representa a quantidade de *bits* necessário para representar o valor de l (máximo valor dos pesos em módulo) com o sinal, quantidade de neurônios da camada escondida e a quantidade de entradas para cada neurônio escondido. Por exemplo, Se $k = 3$, $l = 127 = (1111111)_2$ e $n = 32$ teremos $\varepsilon = 8$ e $\delta = 768$ *bits*. Supondo que desejamos obter uma chave de 128 *bits* então podemos dividir o vetor em um conjunto de 6 vetores de 128 *bits*. O IV irá distinguir um destes novos vetores, ou seja:

$$IV \in \{0,1,2,3,4,5\} \tag{5.2}$$

O outro campo da mensagem FIN_SYNC é o identificador (ID). Este campo tem a finalidade de informar ao leitor que esta mensagem é uma resposta a mensagem recebida

com o mesmo identificador, ou seja, uma mensagem recente. Por exemplo, o leitor envia uma mensagem SYNC com o identificador igual a 5, a *tag* processa a mensagem e envia uma resposta com o identificador também igual a 5.

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 (τ^T). Caso a saída da sua rede seja igual à 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 ajustar os pesos dele. Para informar ao leitor a *tag* utiliza a mensagem ACK_SYNC composta apenas do identificador, sendo este igual ao identificador da mensagem SYNC recebida. Se $\tau^T \neq \tau^L$, 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 verificar se o identificador do quadro recebido corresponde ao identificador do quadro esperado. Se a mensagem for uma mensagem antiga, o leitor retornará a aguardar novas mensagens 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.

Ao término do processo de treinamento as estruturas neurais da *tag* e do leitor apresentação as mesmas chaves de critpografia.

5.2.2 Protocolo de Diffie-Hellman

O leitor inicia o processo de geração das chaves obtendo, nos seus registradores, os valores de p e g. Estes valores são fixos e inseridos pelo administrador do sistema. Lembrando que p é um número primo e g é uma raiz primitiva modulo p. Na seqüência, o leitor gera um número aleatório (X_L), sendo que $X_L < p$. Após a geração do número, o leitor calcula a sua senha pública (Y_L), conforme mostrada na equação:

$$Y_L = g^{X_L} \mod p. \tag{5.3}$$

O leitor tem a atribuição de informar as variáveis g, p e Y_L a *tag*, através do quadro de sincronização (chamado de SYNC), conforme mostrado na Figura 5.3. O quadro SYNC tem ainda um identificador (ID). Este campo tem a finalidade de informar a *tag* qual é o identificador da mensagem. A este campo é atribuído um valor aleatório.

De forma semelhante ao protocolo de Criptografia Neural, após o envio do quadro SYNC, o leitor dispara um cronômetro e fica aguardando uma resposta da *tag*. Caso a *tag* não responda até um determinado tempo limite, o leitor reinicia o processo de sincronização. Porém, o leitor só enviará uma nova mensagem se o número de tentativas não

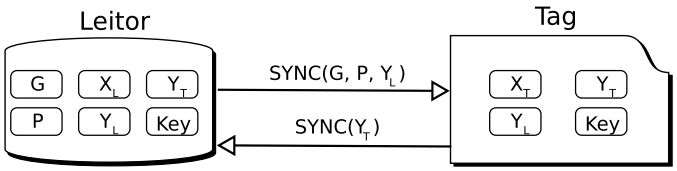

Figura 5.3: Mensagens utilizadas pelo protocolo de Diffie-Hellman implementado.

exceder certo limite. Além disso, antes de criar a nova mensagem o leitor deverá detectar a presença da *tag*. Esta restrição se faz necessário para evitar que o leitor envie quadros de sincronização quando a *tag* não estiver mais presente.

Quando a *tag* recebe a mensagem de sincronização, ela deverá gerar seu número aleatório (X_T), sendo que $X_T < p$. Na seqüência, a *tag* irá calcular a sua senha pública (Y_T), conforme mostrado na equação:

$$Y_T = g^{X_T} \mod p. \tag{5.4}$$

A *tag* deverá informar as sua senha pública Y_T ao leitor, através do quadro de sincronização (chamado de SYNC). O outro campo desta mensagem é o identificador (ID). Este campo tem o objetivo de informar ao leitor que esta mensagem é uma resposta a mensagem recebida com o mesmo identificador. Após o enviou do quadro SYNC a *tag* irá calcular a sua chave de criptografia (k_{LT}):

$$K_{LT} = Y_L^{X_T} \mod p \tag{5.5}$$

O leitor que estava aguardando a resposta da *tag*, ao receber a mensagem, irá verificar se o identificador do quadro recebido corresponde ao identificador do quadro esperado. Caso o identificador seja válido, o leitor irá calcular a sua chave criptográfica (K_{LT}):

$$K_{LT} = Y_T^{X_L} \mod p \tag{5.6}$$

O tamanho da chave gerado (δ) é igual à quantidade de *bits* necessários para representar o valor de p. Por exemplo, $p = 233$, temos $\delta = 8$. Caso o tamanho da chave gerada seja inferior ao tamanho da chave em *bits* necessárias para a criptografia, então mais de uma chave deverá ser gerada.

5.3 Resultados

O objetivo deste trabalho é avaliar o desempenho dos protocolos de Criptografia Neural e Diffie-Hellman na geração de chaves em sistemas RFID, através da implementação destes protocolos em uma plataforma embarcada. Os testes foram realizados visando mensurar o tempo de processamento destes protocolos. O tempo de processamento corresponde ao tempo necessário para a geração das chaves criptográficas desprezando o tempo de comunicação.

Lembrando que no protocolo de Criptografia Neural e Diffie-Hellman o nível de segurança é representado pelas variâveis l e p, respectivamente. No protocolo de Criptografia Neural existe ainda a variâvel n responsável por controlar a quantidade de chaves geradas. Na seqüência serão apresentados os resultados dos testes realizados.

A Figura 5.4 ilustra a quantidade média de mensagens com relação ao parâmetro l, sendo $k = 3$ e $n = 32$, em 10 amostras. Nos gráficos é possível observar que o aumento no nível de segurança (ou seja, o aumento na variável l) implica no aumento exponencial da quantidade de mensagens necessárias para a sincronização das redes neurais. Por exemplo, se utilizarmos $l = 25$ e $l = 100$ se fazem necessários aproximadamente 57.048 e 1.133.201 mensagens, respectivamente.

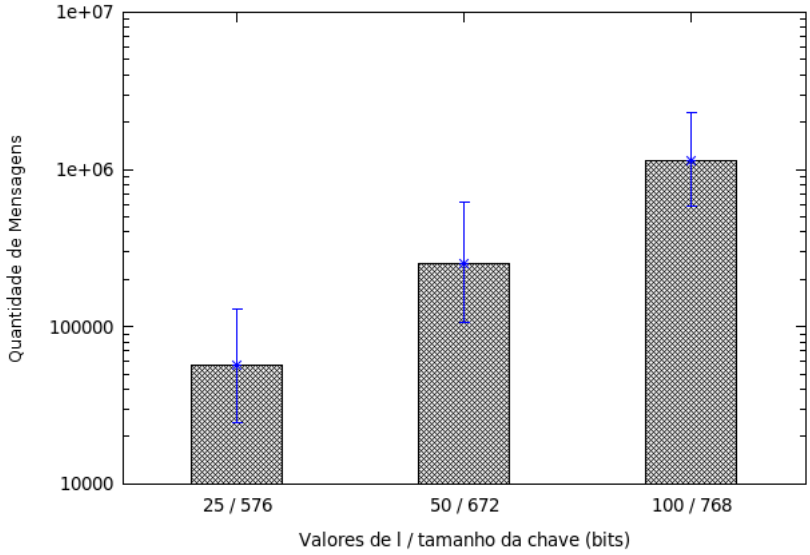

Figura 5.4: Quantidade média de mensagens em função de l, para $k = 3$ e $n = 5$, obtidos em 10 amostras.

De acordo com a Figura 5.4, uma desvantagem do protocolo de Criptografia Neural é a grande quantidade de mensagens necessárias para o protocolo. Lembrando que no protocolo de Diffie-Hellman a quantidade de mensagem é constante e igual a dois.

Os tempos médios de processamento com relação à variável l são mostrados na Figura 5.5. Nos gráficos é possível observar que o aumento na variável l, resultando em uma maior quantidade de mensagens, implica no aumento exponencial do tempo para a obtenção das chaves criptográficas. Por exemplo, se utilizarmos $l = 25$ e $l = 100$ se faz necessário aproximadamente 72,07 e 42.672,50 segundos, respectivamente. Portanto, o protocolo de Criptografia Neural não deverá ser utilizado em sistemas que necessitem de rigorosas exigências de segurança e possuam restrições temporais.

Enquanto o parâmetro l influência diretamente no nível de segurança, a variável n controla a quantidade de chaves geradas. Os tempos médios de processamento com relação à variável n são mostrados na Figura 5.6. Como podemos observar, o aumento no valor de n (ou seja uma maior quantidade de chaves geradas) implica em um aumento no tempo de processamento. Por exemplo, se utilizarmos $n = 11$ e $n = 22$ teremos gerado ao final

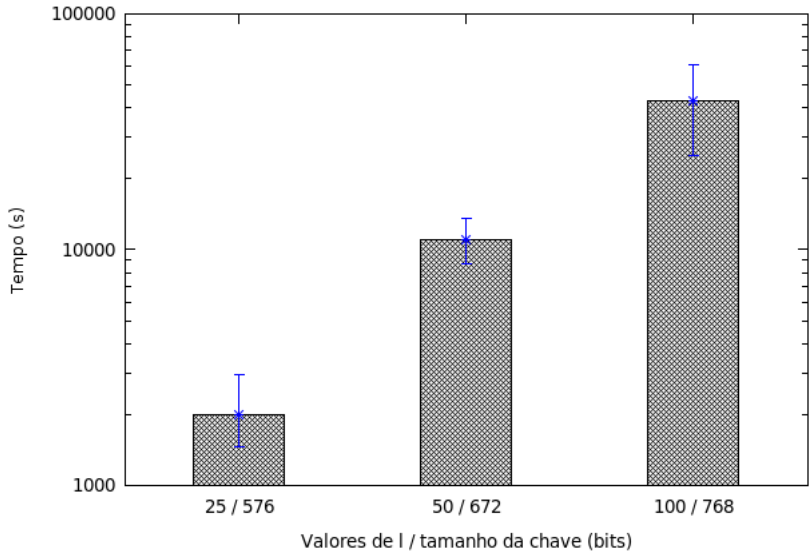

Figura 5.5: Tempo médio de processamento do protocolo de Criptografia Neural em função de l, para $k = 3$ e $n = 5$, obtidos em 10 amostras.

do processo uma chave de 132 *bits* e 264 *bits*, no tempo médio de 20,8 e 41,60 segundos, respectivamente.

Agora iremos apresentar os testes do protocolo de Diffie-Hellman. A Figura 5.7 ilustra o tempo médio de processamento em função dos valores de p, sendo $g = 3$, para a geração de chaves de 128 *bits*. Como os valores de p testados eram inferiores a 128 *bits*, se fez necessário a criação de mais de uma chave. Por exemplo, quando $p = 233$, (gerando chave de 8 *bits*) se faz necessário a criação de 16 chaves, resultando ao final um vetor de $8 * 16 = 128$ *bits*.

De acordo com a Figura 5.7, o aumento nos valores de p (ou seja, aumento no nível de segurança) resulta em uma pequena variação no tempo médio necessário para a geração das chaves. Por exemplo, quando $p = 233$ e $p = 4.294.960.049$ obtemos o tempo médio de 1,40 e 1,15 segundos, respectivamente.

O tempo médio de processamento em função de diferentes quantidades de chaves geradas, sendo $p = 65.419$ e $g = 3$, é mostrado na Figura 5.8. Pelo gráfico é possível observarmos que o aumento na quantidade de chaves geradas implica no aumento do tempo de processamento. Por exemplo, para gerarmos uma chave de 128 e 256 *bits*, se faz necessário criarmos 8 e 16 chaves, resultando no tempo de 1,43 e 2,82 segundos, respectivamente.

Com base nas Figuras 5.4 e 5.7, é possível observar que o protocolo de Criptografia Neural e mais sensível a variação do parâmetro relacionado à segurança do protocolo. Além disso, o tempo médio de processamento obtido foi maior no protocolo de Criptografia Neural. Podendo, dependo da aplicação, se tornar impraticável.

Conforme os resultados dos testes é possível concluir que o tempo de processamento do protocolo de Diffie-Hellman é menor que o de Criptografia Neural. Além disso, o protocolo de Diffie-Hellman é menos sensível a variações nos parâmetros de segurança

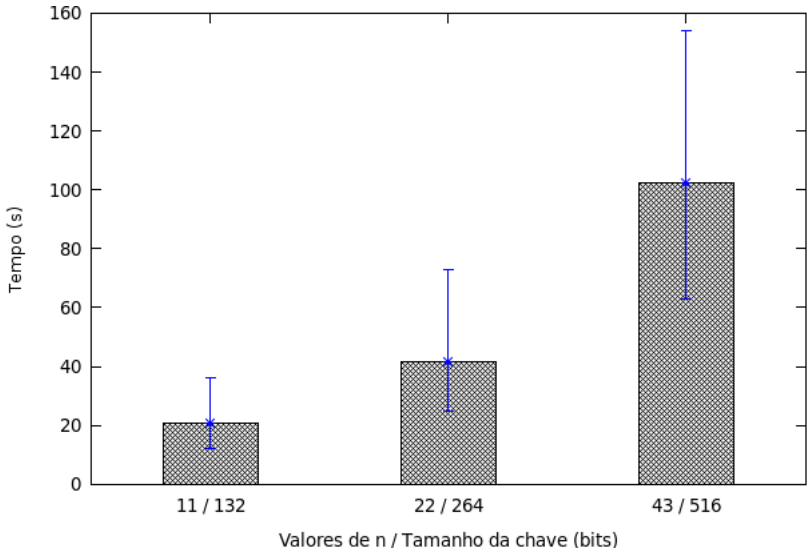

Figura 5.6: Tempo médio de processamento do protocolo de Criptografia Neural em função de n, para $k = 3$ e $l = 5$,obtidos em 10 amostras.

e tamanho da chave gerada, ou seja, com este protocolo é possível obter maiores níveis de segurança. Portanto, o protocolo de Diffie-Hellman é mais apropriado para sistemas RFID quando comparado com o protocolo de Criptografia Neural.

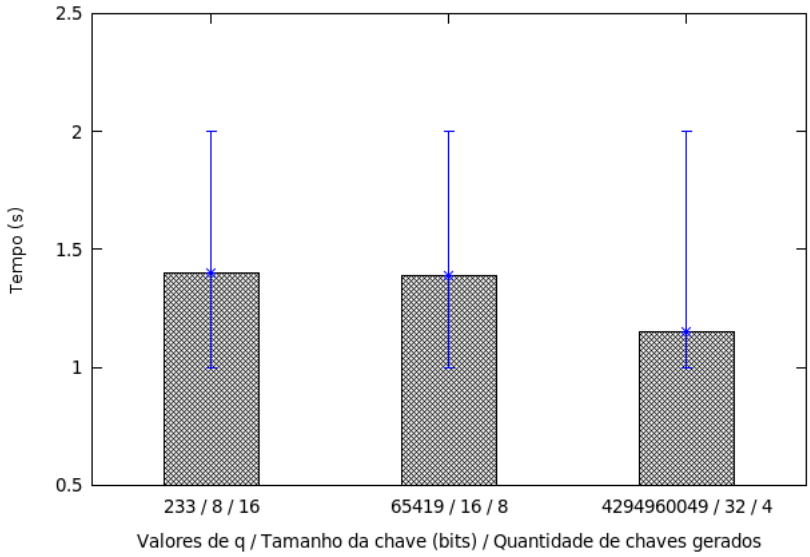

Figura 5.7: Tempo médio de processamento do protocolo de Diffie-Hellman em função de p, para $g = 3$, obtidos em 100 amostras.

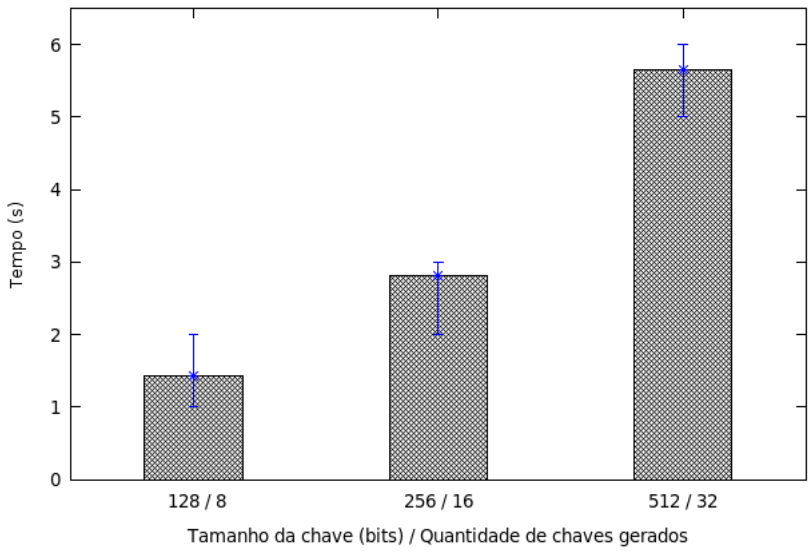

Figura 5.8: Tempo médio de processamento do protocolo de Diffie-Hellman em função de diferentes tamanhos de chave, sendo $p = 65.419$ e $g = 3$, obtidos em 100 amostras.

Capítulo 6

Conclusões e Trabalhos Futuros

Nesta dissertação aspectos de privacidade e segurança da informação em sistemas RFID foram analisados e discutidos. De acordo com estas análises foi possível concluir que para garantir uma comunicação segura entre componentes RFID se faz necessário a utilização de um protocolo de geração de chaves.

Entre os protocolos de geração de chaves presentes na literatura se destaca o protocolo de Criptografia Neural e Diffie-Hellman. No entanto, não existia na literatura implementações e análises de desempenho destes protocolos em sistemas RFID. Desta forma, o presente trabalho teve como objetivo realizar uma avaliação de desempenho dos protocolos de Criptografia Neural e Diffie-Hellman na geração de chaves em sistemas RFID. Para isso, foi mensurado o tempo de processamento destes protocolos. O tempo de processamento corresponde ao tempo necessário para a geração das chaves criptográficas desprezando o tempo de comunicação. O tempo de comunicação é função da quantidade e do tamanho das mensagens. Para mensuramos o tempo de comunicação se fazia necessário a utilização de transmissores e receptores de rádio freqüência. A análise do tempo de comunicação está fora do escopo deste trabalho pois o mesmo não depende do protocolo que está realizando a comunicação.

Para os testes foi desenvolvido uma plataforma em FPGA (*Field-Programmable Gate Array*) com o processador embarcado Nios II. Sobre esta plataforma foram utilizados os protocolos de Criptografia Neural e Diffie-Hellman no processo de geração de chaves criptográficas. A metodologia de pesquisa baseia-se na agregação de conhecimento ao desenvolvimento de novos sistemas RFID através de uma análise comparativa entre esses dois protocolos de segurança da informação.

Nos testes foi possível observar altos valores para o tempo de processamento do protocolo de Criptografia Neural. Portanto, este protocolo não deverá ser utilizado em sistemas que necessitem de rigorosas exigências de segurança e possuam restrições temporais. Por outro lado, o tempo de processamento do protocolo de Diffie-Hellman se apresentou de forma satisfatória. Além disso, o protocolo de Diffie-Hellman é menos sensível a variações nos parâmetros de segurança e tamanho da chave gerada, ou seja, com este protocolo é possível obter maiores níveis de segurança. Logo, o protocolo de Diffie-Hellman é mais apropriado para sistemas RFID quando comparado com o protocolo de Criptografia Neural.

As principais contribuições deste trabalho foram: levantamento bibliográfico de pesquisas relacionadas à segurança da informação em sistemas RFID; implementação dos

protocolos Diffie-Hellman e Criptografia Neural em uma plataforma embarcada em *hardware* reconfigurável e a avaliação de desempenho destes protocolos na plataforma;

A presente dissertação é o começo de uma proposta de projeto de pesquisa para a implementação de um Sistema de Monitoramento e Auditoria Hospitalar seguro que faz uso da tecnologia RFID. Para tal se faz necessário adquirir, previamente, os conhecimentos das vulnerabilidades de segurança e medidas que visam garantir a comunicação segura nesta tecnologia.

Outras sugestões para trabalhos futuros:

- Mensurar o tempo de comunicação entre as entidades RFID;
- Analisar outros protocolos de geração de chaves;
- Implementar os protocolos diretamente em *hardware*.

Referências Bibliográficas

Albassal, Ayman M. B. & Abdel-Moneim A. Wahdan (2004), 'Neural network based cryptanalysis of a feistel type block cipher', *Electrical, Electronic and Computer Engineering* pp. 231– 237.

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

Beth, Thomas, Markus Frisch & Gustavus J. Simmons, eds. (1992), *Public-Key Cryptography: State of the Art and Future Directions, E.I.S.S. Workshop, Oberwolfach, Germany, July 3-6, 1991, Final Report*, Vol. 578 de *Lecture Notes in Computer Science*, Springer.

Bono, Steve, Matthew Green, Adam Stubblefield, Ari Juels, Avi Rubin & Michael Szydlo (2005), Security Analysis of a Cryptographically-Enabled RFID Device, *em* 'In 14th USENIX Security Symposium', USENIX, pp. 1–16.

Chan, Chi-Kwong & L.M. Cheng (1998), 'Pseudorandom generator based on clipped hopfield neural network', *Circuits and Systems* **3**, 183–186.

Diffie, Whitfield & Martin E. Hellman (1976), 'New directions in cryptography', *IEEE Transactions on Information Theory* **IT-22**(6), 644–654.

EPCGlobal (2009), Electronic Product Code (EPC): An overview. url: http://www.epcglobalinc.org/public/ppsc_factsheets/epc_overview, último acesso em Junho de 2009.

Feldhofer, Martin, Sandra Dominikus & Johannes Wolkerstorfer (2004), Strong Authentication for RFID Systems Using the AES Algorithm, pp. 357–370.

Finkenzeller, Klaus (2003), *RFID Handbook : Fundamentals and Applications in Contactless Smart Cards and Identification*, 2ª edição, John Wiley & Sons.

Florentino, G. H. P., M. A. Xavier, H. Uchoa, A. Junior H. B., R. A. M. Valentim, Morais A. H. F., Ana Maria Guimarães Guerreiro, Glaucio Bezerra Brandão & Carlos Paz de . Araujo (n.d.), 'Hospital Automation System RFID-Based: Technology Embedded In Smart Devices (Cards, Tags and Bracelets)'.

Forouzan, Behrouz A. (2006), *Comunicação de Dados e Redes de Computadores*, 3ª edição, Bookman.

Guerreiro, Ana Maria G. & Carlos Paz de Araujo (2006), A Neural Key Generator for a Public Block Cipher, *em* 'SBRN '06: Proceedings of the Ninth Brazilian Symposium on Neural Networks', IEEE Computer Society, p. 25.

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

Heydt-Benjamin, Thomas, Daniel Bailey, Kevin Fu, Ari Juels & Tom O'hare (2008), Vulnerabilities in First-Generation RFID-enabled Credit Cards, pp. 2–14.

Jeng, Albert, Li-Chung Chang & Sheng-Hui Chen (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, Ari (2005), Minimalist Cryptography for Low-Cost RFID Tags (Extended Abstract), pp. 149–164.

Juels, Ari (2006), 'RFID Security and Privacy: A research Survey', *IEEE Journal on Selected Areas in Communication* **24**(2).

Kanter, I., W. Kinzel & E. Kanter (2002), 'Secure exchange of information by synchronization of neural networks', *Europhysics Letters* **57**, 141.

Karras, D.A. & V. Zorkadis (2003), 'Improving pseudorandom bit sequence generation and evaluation for secure internet communications using neural network techniques', *Neural Networks* **2**, 1367– 1372.

Kinzel, Wolfgang & Ido Kanter (2002), Neural cryptography, *em* 'in Proc. of the 9th International Conference on Neural Information Processing', pp. 18–22.

Lei, He, Gan Yong, Li Na-Na & Cai Zeng-Yu (2007), 'A Security-Provable Authentication and Key Agreement Protocol in RFID System', *IEEE - WiCom* pp. 2078 – 2080.

Molnar, David & David Wagner (2004), Privacy and security in library RFID: issues, practices, and architectures, *em* 'CCS '04: Proceedings of the 11th ACM conference on Computer and communications security', ACM, New York, NY, USA, pp. 210–219.

Osipov, Grigory V., Jurgen Kurths & Changsong Zhou (2007), *Synchronization in Oscillatory Networks*, Springer.

Piramuthu, Selwyn (2007), 'Protocols for RFID tag reader authentication', *Decis. Support Syst.* **43**(3), 897–914.

Rieback, Melanie, Bruno Crispo & Andrew S. Tanenbaum (2006*a*), Is Your Cat Infected with a Computer Virus?, *em* 'PERCOM '06: Proceedings of the Fourth Annual IEEE International Conference on Pervasive Computing and Communications', IEEE Computer Society, Washington, DC, USA, pp. 169–179.

Rieback, Melanie, Patrick Simpson, Bruno Crispo & Andrew S. Tanenbaum (2006), 'RFID malware: Design principles and examples', *Pervasive and Mobile Computing* **2**(4), 405–426.

Rieback, Melanie R., Bruno Crispo & Andrew S. Tanenbaum (2005), RFID guardian: A battery-powered mobile device for RFID privacy management, *em* 'Proc. 10th Australasian Conf. on Information Security and Privacy (ACISP 2005)', Vol. 3574 de *LNCS*, Springer-Verlag, pp. 184–194.

Rieback, Melanie R., Bruno Crispo & Andrew S. Tanenbaum (2006*b*), 'The Evolution of RFID Security', *IEEE Pervasive Computing* **5**(1), 62–69.

Ruttor, Andreas (2007), 'Neural Synchronization and Cryptography'.

Sarma, Sanjay, Stephen Weis & Daniel Engels (2002), RFID Systems and Security and Privacy Implications, *em* B.Kaliski, c.Kaya ço & C.Paar, eds., 'Cryptographic Hardware and Embedded Systems – CHES 2002', Vol. 2523 de *Lecture Notes in Computer Science*, Springer-Verlag, Redwood Shores, California, USA, pp. 454–469.

Shacham, L.N., E. Klein, R. Mislovaty, I. Kanter & W. Kinzel (2003), 'Cooperating attackers in neural cryptography'.

Singh, Simon (1999), *The Code Book: The Evolution of Secrecy from Mary, Queen of Scots, to Quantum Cryptography*, Doubleday, New York, NY, USA.

Stallings, William (2008), *Criptografia e Segurança de Redes: Princípios e Práticas*, 4ª edição, Prentice-Hall.

Stockman, Harry (1948), 'Communication by Means of Reflected Power', *Proceedings of the IRE* pp. 1196–1204.

Suliman, Shankarapani, Mukkamala & Sung (2008), 'RFID malware fragmentation attacks', *Collaborative Technologies and Systems* pp. 533–539.

Tanenbaum, Andrew S (2002), *Computer networks: 4nd edition*, Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

Valentim, R. A. M., Morais A. H. F., M. A. Xavier, Diego R. Carvalho, Glaucio Bezerra Brandão & Ana Maria Guimarães Guerreiro (2008), 'Rfid aplicado a automação hospitalar: Desenvolvimento de um sistema para automação de laboratórios de análise clínica', *XVII Congresso Brasileiro de Automática* **1**.

Volkmer, Markus & Sebastian Wallner (2005), 'Lightweight key exchange and stream cipher based solely on tree parity machines'.

Westhues, Jonathan (2005), Hacking the prox card, *em* 'RFID: Applications, Security, and Privacy', Addison-Wesley, pp. 169–179.

Yee, Liew Pol & L.C. de Silva (2002), 'Application of multilayer perceptron networks in public keycryptography', *Neural Networks* **2**, 1439–1443.