UNIVERSIDADE FEDERAL DO CEARÁ

CENTRO DE TECNOLOGIA

DEPARTAMENTO DE ENGENHARIA DE TELEINFORMÁTICA

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE TELEINFORMÁTICA

TIAGO MALVEIRA CAVALCANTE

AVALIAÇÃO DE DESEMPENHO DE MECANISMOS DE SEGURANÇA UTILIZADOS PARA PROVER OS SERVIÇOS DE CONFIDENCIALIDADE, INTEGRIDADE E AUTENTICAÇÃO EM REDES DE SENSORES SEM FIO

FORTALEZA

2012

TIAGO MALVEIRA CAVALCANTE

AVALIAÇÃO DE DESEMPENHO DE MECANISMOS DE SEGURANÇA UTILIZADOS PARA PROVER OS SERVIÇOS DE CONFIDENCIALIDADE, INTEGRIDADE E AUTENTICAÇÃO EM REDES DE SENSORES SEM FIO

Dissertação de mestrado apresentada ao Programa de Pós-Graduação em Engenharia de Teleinformática, da Universidade Federal do Ceará, como requisito parcial para obtenção do Título de Mestre em Engenharia de Teleinformática. Área de concentração: Sinais e Sistemas.

Orientadora: Profa. Dra. Rossana Maria de Castro Andrade.

FORTALEZA

2012

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca de Ciências e Tecnologia

C364a Cavalcante, Tiago Malveira.
Avaliação de desempenho de mecanismos de segurança utilizados para prover os serviços de confidencialidade, integridade e autenticação em redes de sensores sem fio / Tiago Malveira Cavalcante. – 2012.
126 f. : il., enc. ; 30 cm.

Dissertação (mestrado) – Universidade Federal do Ceará, Centro de Tecnologia, Departamento de Engenharia de Teleinformática, Programa de Pós-Graduação em Engenharia de Teleinformática, Fortaleza, 2012.
Área de Concentração: Sinais e Sistemas.
Orientação: Profa. Dra. Rossana Maria de Castro Andrade.

1. Redes sensoriais - Segurança. 2. Sistemas de comunicação sem fio – Medidas de segurança. I. Título.

CDD 621.38

TIAGO MALVEIRA CAVALCANTE

AVALIAÇÃO DE DESEMPENHO DE MECANISMOS DE SEGURANÇA UTILIZADOS PARA PROVER OS SERVIÇOS DE CONFIDENCIALIDADE, INTEGRIDADE E AUTENTICAÇÃO EM REDES DE SENSORES SEM FIO

Dissertação de mestrado apresentada ao Programa de Pós-Graduação em Engenharia de Teleinformática, da Universidade Federal do Ceará, como requisito parcial para obtenção do Título de Mestre em Engenharia de Teleinformática. Área de concentração: Sinais e Sistemas.

Aprovada em: ___/___/_____.

BANCA EXAMINADORA

_____

Profa. Dra. Rossana Maria de Castro Andrade (Orientadora) (PPGETI/UFC)

_____

Prof. Dr. Antonio Alfredo Ferreira Loureiro (DCC/UFMG)

_____

Prof. Dr. Pedro Fernandes Ribeiro Neto (MCC/UERN)

_____

Prof. Dr. Danielo Gonçalves Gomes (PPGETI/UFC)

A Deus.

Aos meus pais, Jucival e Antonia.

Aos meus irmãos, Jucianne e Jéfferson.

Ao meu avô, vovô Deca (*in memorian*).

Ao meu amigo Eleonardo (*in memorian*).

AGRADECIMENTOS

À Profa. Dra. Rossana Maria de Castro Andrade, pela valiosa orientação, ensino e incentivo no decorrer do curso de mestrado.

Ao Prof. Dr. Danielo Gonçalves Gomes, pelas excelentes sugestões e apoio durante o curso do mestrado.

À Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), pelo apoio financeiro com a manutenção da bolsa de auxílio.

À Universidade Federal do Ceará e em especial ao curso de mestrado em Engenharia de Teleinformática pelo conhecimento e experiência adquirida.

Ao Grupo de Redes de Computadores, Engenharia de Software e Sistemas (GREat) pela disponibilidade de equipamentos e ambiente de estudos.

Ao Laboratório de Microcontroladores (LABOMICRO) do Instituto Federal de Educação Tecnológica do Ceará (IFCE) pela disponibilidade de equipamentos utilizados neste trabalho.

Aos meus pais Antonia Malveira Cavalcante e Jucival Ferreira Cavalcante pelo encorajamento em todas as fases de minha vida, pelo seu amor e por suas orações.

Ao meu irmão e grande amigo Jéfferson Malveira que tanto me incentivou, ajudou e apoiou durante toda a minha vida.

À minha irmã Jucianne e aos meus sobrinhos Carlos Neto e Fernanda Lys pelo apoio e horas de descontração.

Aos meus avós Alzira e Sebastião pelo carinho, incentivo e apoio.

À Kelly Mineiro Cavalcante, pelo apoio incondicional, carinho e incentivo.

Aos meus tios e tias, em especial as tias Aurilênia, Alzirene e Fátima e ao tio Paulo, que tanto me incentivaram e apoiaram durante todo a minha vida.

Enfim, a todos que de alguma forma me ajudaram na realização deste trabalho.

“A persistência realiza o impossível.”

(Provérbio Chinês)

RESUMO

O crescimento e a diversidade de aplicações para Redes de Sensores sem Fio (RSSFs) trouxeram novos desafios e uma maior visibilidade para a necessidade de buscar soluções para a segurança dessas redes. Por exemplo, o meio físico sem fio torna essas redes vulneráveis a diversos tipos de ataques, sendo essencial o uso de pelo menos os serviços básicos de segurança, como a confidencialidade, a integridade e a autenticação. Entretanto, prover esses serviços em RSSF é um grande desafio devido às severas limitações de recursos de comunicação, processamento, memória e energia dos dispositivos utilizados nessas redes. Dessa forma, a seleção dos mecanismos de segurança mais apropriados levando em consideração o desempenho é essencial para prover a segurança da informação em RSSF. Por outro lado, os principais trabalhos encontrados na literatura que objetivam avaliar o desempenho de mecanismos de segurança no contexto de RSSFs apresentam grandes diferenças em termos de plataforma escolhida e métricas avaliadas, além de apresentar deficiências no processo de avaliação de desempenho utilizado. Sendo assim, este trabalho propõe uma avaliação de desempenho utilizando um processo sistemático, baseado em abordagens de avaliação de desempenho tradicionais, no qual os mecanismos de segurança são selecionados, implementados e validados. Os mecanismos de segurança para prover os serviços de confidencialidade, integridade e autenticação em RSSFs são considerados neste trabalho. Experimentos reais são realizados na plataforma MicaZ com o auxílio de um osciloscópio digital para medir o desempenho dos algoritmos utilizados. Com os resultados obtidos, espera-se oferecer um direcionamento quanto à escolha dos mecanismos de segurança mais apropriados para aplicações de RSSF.

Palavras-chave. Avaliação de Desempenho. Redes de Sensores Sem Fio. Segurança.

ABSTRACT

The number and diversity of applications for Wireless Sensor Networks (WSNs) have grown in the last decade what makes more relevant the concern about information security in such networks. For example, since the wireless physical medium makes these networks vulnerable to various types of attacks, there is then a need to use at least basic security services such as confidentiality, integrity, and authentication. However, providing security in WSN faces some challenges because of the severe limitations of processing, communication, power, and memory resources of the devices used in these networks. In this context, the selection of the most appropriate security mechanisms taking into consideration their performance is essential to provide information security in WSNs. On the other hand, the main results found in the literature that aim to assess the performance of security mechanisms in the context of a WSN differ in terms of the chosen platform and evaluated metrics, and present deficiencies in the performance assessment process used. This work then proposes a performance assessment using a systematic process, based on traditional performance analysis approaches, in which security mechanisms are selected, implemented and validated. The security mechanisms considered in this work are for that provide confidentiality, integrity, and authentication services in WSNs. Real experiments are carried out on the MicaZ platform with a digital oscilloscope to measure the performance of the chosen algorithms. With the obtained results a good direction for choosing the most appropriate security mechanisms for WSN applications is aimed as another contribution of this work.

Keywords. Performance Evaluation. Wireless Sensor Networks. Security.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ALGORITMOS

LISTA DE ABREVIATURAS E SIGLAS

AES	*Advanced Encryption Standard*
CBC	*Cipher-Block Chaining*
CBCMAC	*Cipher-Block Chaining Message Authentication Code*
CFB	*Cipher Feedback*
CMAC	*Cipher-Based Message Authentication Code*
CTR	*Counter Mode*
DARPA	*Defense Advanced Research Projects Agency*
DoS	*Denial of Service*
HMAC	*Hash-Based Message Authentication Code*
IDS	*Intrusion Detection System*
IV	*Initialization Vector*
MAC	*Message Authentication Code*
MD5	*Message Digest 5*
NSA	*National Security Agency*
OCB	*Offset Codebook Mode*
OFB	*Output Feedback*
PEG	*Pursuit-Evasion Game*
QoS	*Quality of Service*
RSSF	Redes de Sensores Sem Fio
SHA1	*Secure Hash Algorithm*
SOSUS	*Sound Surveillance System*
WBAN	*Wireless Body Area Networks*
WSAN	*Wireless Sensor and Actor Networks*
WSN	*Wireless Sensor Networks*

SUMÁRIO

1 INTRODUÇÃO

Esta dissertação apresenta uma avaliação de desempenho de mecanismos de segurança para Redes de Sensores Sem Fio (RSSF), na qual são considerados mecanismos de criptografia, modos de operação criptográficos e algoritmos de autenticação.

Na seção 1.1 são apresentadas a caracterização do problema e a motivação que influenciaram o desenvolvimento deste trabalho. Em seguida, o objetivo geral e os objetivos específicos são indicados na seção 1.2 e a metodologia científica adotada é descrita na seção 1.3. Finalmente, a seção 1.4 encerra o capítulo introdutório descrevendo a estrutura restante deste documento.

1.1 Caracterização do Problema e Motivação

Como já bastante discutido na literatura (AKYILDIZ *et al.*, 2002; GARCIA-HERNANDEZ *et al*., 2007; ZHAO; GUIBAS, 2004), os avanços recentes da microeletrônica proporcionaram o desenvolvimento de pequenos sensores que, em conjunto com dispositivos com capacidades de processamento e recursos de computação limitadas e providos de comunicação sem fio, formam um nó sensor. Uma coleção desses nós sensores trabalhando cooperativamente forma uma Rede de Sensores Sem Fio (RSSF).

Uma RSSF possui um grande potencial de aplicações, sendo utilizada geralmente para coletar dados em um determinado ambiente com o objetivo de realizar tarefas de monitoramento e/ou de rastreamento. Dentre as inúmeras aplicações das RSSF, destacam-se o monitoramento ambiental (SZEWCZYK *et al.*, 2004; WERNER-ALLEN *et al.*, 2006; COLONNA *et al.*, 2011), as aplicações para detecção de incêndios (LI *et al.*, 2006; HEFEEDA; BAGHERI, 2007), as aplicações médicas (JANANI *et al.*, 2011), a aquisição de dados na indústria (LOW *et al,*. 2005; SHEN *et al.,* 2004) e aplicações de tempo real (BORGES NETO *et al.*, 2010a).

Nos últimos anos, várias pesquisas têm buscado resolver os problemas relacionados às severas limitações de recursos inerentes às RSSF, principalmente as limitações de processamento, armazenamento e energia. Entretanto, o surgimento de novos campos de

aplicação, como a computação ubíqua, introduziu novos problemas relacionados a fatores como a alta mobilidade dos nós sensores e requisitos de qualidade de serviço (*QoS – Quality of Service*) (BORGES NETO *et al.*, 2010b).

Vale ressaltar ainda que determinados tipos de aplicações podem necessitar de serviços de segurança devido a fatores como o ambiente no qual a rede foi implantada, as ameaças esperadas, o nível de criticidade dos dados que trafegam na rede ou o ganho esperado por um adversário ao atacar a rede (SABBATH *et al.*, 2006).

Dessa forma, aplicações de RSSF com a finalidade de realizar o monitoramento de espécies animais ou aplicações implantadas em ambientes controlados, por exemplo, podem não exigir serviços de segurança em nível prioritário. Mas a segurança é essencial em aplicações de RSSF empregadas em áreas como a medicina. Por exemplo, a operação de uma RSSF dotada de nós sensores com capacidade de atuação utilizada para realizar o monitoramento de órgãos vitais humanos pode resultar em consequências trágicas na ocorrência de um ataque, caso a rede não utilize serviços básicos de segurança.

Outro exemplo de categoria de aplicações de RSSF que podem necessitar prioritariamente de segurança dos dados consiste em aplicações militares. Nesse tipo de aplicação, em geral, o acesso não autorizado por inimigos a informações trafegadas entre os nós sensores pode resultar no comprometimento da rede.

Entretanto, a segurança da informação em RSSF é um dos principais problemas encontrados nesse tipo de rede, sendo o foco de diversos pesquisadores da área nos últimos anos (WALTERS *et al.*, 2006). Assim como os sistemas de comunicação sem fio em geral, as RSSF utilizam o ar como meio de propagação do sinal. Essa característica torna essas redes vulneráveis a diversos tipos de ataques (e.g., *eavesdropping*, modificação de mensagens e injeção de mensagens), evidenciando a necessidade do emprego de mecanismos de segurança em RSSF (CIRQUEIRA *et al.*, 2011).

Se por um lado o aumento das aplicações de RSSF, bem como a diversificação das mesmas, enseja a preocupação com a segurança dos dados coletados, por outro, a segurança é um aspecto que vem sendo considerado um desafio em RSSF devido às suas limitações de memória, processamento e energia (KAPS *et al.*, 2007).

Dessa forma, para garantir a segurança dos dados nas aplicações de RSSF, uma primeira alternativa deve ser a utilização de mecanismos para prover essas redes com pelo menos os serviços básicos de segurança**,** tais como confidencialidade, integridade e autenticação.

Além disso, vale ressaltar que a seleção dos mecanismos de segurança mais apropriados para uma RSSF é essencial, uma vez que a utilização de mecanismos inadequados pode acarretar um grande impacto negativo em uma RSSF, podendo resultar no aumento do consumo de energia e na perda de desempenho da rede, entre outras consequências negativas (SIMPLÍCIO; BARRETO, 2010).

Portanto, a avaliação de desempenho de mecanismos de segurança possui importância fundamental para orientar o projeto de RSSF na escolha dos mecanismos mais apropriados a serem empregados em aplicações de RSSF.

1.2 Objetivos

1.2.1 Objetivo Geral

O objetivo deste trabalho é avaliar o desempenho de mecanismos de segurança capazes de prover os serviços de confidencialidade, integridade e autenticação para RSSF. Com os resultados obtidos nessa avaliação de desempenho, este trabalho tem o intuito de fornecer um direcionamento na escolha dos mecanismos de segurança mais apropriados para serem empregados em aplicações de RSSF, inclusive no desenvolvimento e evolução de protocolos de segurança específicos para tais redes.

1.2.2 Objetivos Específicos

Para alcançar o objetivo geral desta dissertação é necessária a realização das seguintes tarefas específicas:

a) elaborar um processo de avaliação de desempenho de mecanismos de segurança para RSSF baseado em abordagens de avaliação de desempenho tradicionais;

b) escolher os mecanismos de segurança a serem avaliados, dentre os mecanismos disponíveis na literatura;

c) avaliar o desempenho dos algoritmos criptográficos, modos de operação e algoritmos de autenticação por meio de experimentos reais utilizando nós sensores MicaZ, uma das plataformas mais utilizadas no contexto de RSSF atuais.; e

d) analisar e discutir os resultados obtidos na execução do processo de avaliação de desempenho.

1.3 Metodologia

A metodologia científica utilizada neste trabalho é caracterizada resumidamente nos itens a seguir.

a) Revisão de Literatura

Inicialmente é realizada uma revisão bibliográfica sobre os principais conceitos e desafios na área de RSSF, a fim de investigar se a segurança como um dos principais desafios atuais dessa área. Dessa forma, a revisão inclui um estudo sobre conceitos tradicionais de segurança, envolvendo mecanismos, serviços e protocolos de segurança empregados nesse tipo de rede. Além disso, são estudados diversos trabalhos relacionados com avaliação de desempenho de mecanismos de segurança para RSSF.

b) Elaboração do Processo de Avaliação de Desempenho

Em seguida é elaborado um processo sistemático de avaliação de desempenho de mecanismos de segurança para RSSF, seguindo os conceitos e princípios da abordagem de avaliação de desempenho proposta por Jain (1991). Na execução do processo elaborado, são escolhidos os principais mecanismos de

segurança publicados na literatura para realizar a avaliação de desempenho proposta neste trabalho. O nível de segurança e as características de projeto são os principais critérios de escolha dos mecanismos de segurança avaliados nesta dissertação. Além disso, outros mecanismos tradicionais foram incluídos para verificar a viabilidade de operação no contexto atual de RSSF. Vale ressaltar que as métricas de avaliação são determinadas levando-se em consideração as principais limitações de recursos em RSSF.

c) Medição e Avaliação dos Resultados

Seguindo a execução do processo de avaliação de desempenho, experimentos reais são realizados na plataforma MicaZ. Em seguida, os resultados obtidos são analisados e apresentados. O último passo consiste na comparação e discussão sobre o desempenho dos diversos mecanismos de segurança considerados neste trabalho.

1.4 Organização do Texto

O restante desta dissertação está organizado conforme as descrições dos capítulos a seguir.

- Capítulo 2

No Capítulo 2 são apresentados as principais características e conceitos relacionados com as RSSF, incluindo a descrição de diversas aplicações reais desse tipo de rede.

- Capítulo 3

O Capítulo 3 apresenta os principais conceitos e as principais características envolvidas com a segurança da informação no contexto de RSSF relacionados com esta dissertação.

- Capítulo 4

  O Capítulo 4 aborda os mecanismos de segurança utilizados na avaliação de desempenho proposta nesta dissertação. Além disso, são discutidos os principais trabalhos relacionados e os protocolos de segurança específicos para RSSF.

- Capítulo 5

  No Capítulo 5 é apresentada a descrição completa do processo utilizado para avaliar o desempenho dos mecanismos de segurança no contexto de RSSF, foco principal desta dissertação.

- Capítulo 6

  Finalmente, o Capítulo 6 finaliza a dissertação enfatizando as conclusões, as contribuições e as perspectivas de trabalhos futuros.

2 REDES DE SENSORES SEM FIO

Neste capítulo são apresentados as principais características e conceitos envolvidos com as Redes de Sensores Sem Fio para facilitar a compreensão dos capítulos posteriores. A seção 2.1 define as RSSF, discutindo sua origem. Em seguida, a seção 2.2 discorre sobre as principais características desse tipo de rede e a seção 2.3 exemplifica várias aplicações reais de RSSF. Finalmente, a seção 2.4 conclui o capítulo.

2.1 Introdução

Conforme mencionado por Romer e Mattern (2004), o surgimento das RSSF foi motivado principalmente por aplicações militares. Uma das primeiras aplicações conhecidas foi implantada durante a guerra fria e se chamava SOSUS (*Sound Surveillance System*). Ela consistia de um sistema de sensores acústicos (hidrofones) implantados em localizações estratégicas no fundo do oceano para rastrear submarinos soviéticos. Além disso, uma série de projetos financiados pela agência norte americana DARPA (*Defense Advanced Research Projects Agency*) é considerada como uma das principais precursoras das pesquisas que envolvem esse paradigma de comunicação (SHONG; KUMAR, 2003).

Embora haja outras definições disponíveis na literatura, esta dissertação considera, por ser mais completa, a definição de Borges Neto (2010) que, baseando-se nos conceitos definidos por Loureiro *et al.* (2003) e Ribeiro Neto (2006), define RSSF da seguinte forma:

> São redes compostas por dispositivos dotados de limitações computacionais, denominados sensores inteligentes, capazes de monitorar e, em alguns casos, controlar o ambiente onde estão inseridos. Estes sensores se comunicam entre si para, de forma colaborativa, realizar tarefas pré-estabelecidas e prolongar o seu tempo de funcionamento.

É importante observar que, de acordo com a definição anterior, os dispositivos de uma RSSF, em alguns casos, podem ser capazes de controlar o ambiente onde estão inseridos. Essa

característica indica que uma RSSF pode ser composta por dispositivos com capacidades de atuação. Assim, vale ressaltar que as RSSF que possuem, simultaneamente, sensores e atuadores são também chamadas na literatura de WSANs (*Wireless Sensor and Actor Networks*) (MELODIA, 2007).

2.2 Desafios e Características

Uma RSSF possui características intrínsecas desafiantes que as diferem das demais redes de comunicações. Além disso, o surgimento de novos campos de aplicação, como a computação ubíqua, introduziu novos desafios a serem enfrentados por pesquisadores da área de RSSF. Os principais desafios e características envolvidos com as RSSF serão discutidos nas próximas subseções.

2.2.1 Limitações

A principal característica das RSSF está nas suas severas limitações de recursos computacionais, comunicação e, principalmente, fonte de energia. Conforme pode ser observado na definição de RSSF da seção 1.1, essas redes são formadas por dispositivos chamados sensores inteligentes[1]. Esses dispositivos são compostos basicamente por quatro subsistemas (BHARATHIDASAN; PONDURU, 2003):

a) Processamento – consiste de um microcontrolador que é responsável principalmente pelo controle dos sensores e execução dos protocolos de comunicação. Vale lembrar que os dispositivos de armazenamento fazem parte desse subsistema.

b) Comunicação – é formado por um transceptor que fornece a interface de comunicação sem fio para o dispositivo, permitindo a comunicação na RSSF.

[1] O termo *nó sensor* também é utilizado na literatura para referenciar os sensores inteligentes. Além disso, o termo *sensor* é, de forma genérica, bastante utilizado na literatura como sinônimo de tais dispositivos.

c) Sensor – consiste em um grupo de sensores e atuadores utilizados, respectivamente, para coletar informações do ambiente no qual estão inseridos e realizar uma atuação nesse ambiente.

d) Fonte de energia – é composto por uma bateria que fornece a energia para o funcionamento do sensor inteligente.

A redução do tamanho dos sensores inteligentes, que pode variar de alguns centímetros para um tamanho microscópico, implicou na redução de tamanho e capacidade de seus componentes (LOUREIRO *et al.*, 2003), sendo esse o fator determinante das limitações das RSSF.

2.2.2 Tolerância a Falhas

Em uma RSSF, um ou mais sensores inteligentes podem se tornar inoperantes devido a uma série de fatores. Por exemplo, um dispositivo pode esgotar sua bateria, sofrer interferência de ruído do ambiente ou sofrer algum dano físico, uma vez que é comum o uso de RSSF em ambientes hostis, conforme será discutido na seção 2.3. Assim, a falha de um nó sensor não deve comprometer o funcionamento da rede. Logo uma RSSF deve possuir a capacidade de tolerar falhas, ou seja, ela deve ser capaz de manter as funcionalidades da rede sem qualquer interrupção devido a falhas de sensores inteligentes (AKYILDIZ *et al.*, 2001).

É bom lembrar ainda que o projeto de mecanismos e protocolos para RSSF deve considerar o nível de tolerância a falhas requerida por essas redes (AKYILDIZ *et al.*, 2001). Por exemplo, em uma aplicação residencial para realizar o controle de temperatura, o nível de tolerância a falhas pode ser baixo, uma vez que o ambiente no qual a RSSF está implantada dificilmente causa alguma interferência de ruído e os sensores inteligentes dificilmente podem sofrer danos físicos. Por outro lado, caso um RSSF seja implantada em um campo de batalha, é bem provável que haja destruição de sensores inteligentes. Nesse caso, um nível elevado de tolerância a falhas deve ser considerado no projeto de mecanismos para RSSF.

2.2.3 Escalabilidade e Densidade

Uma RSSF pode possuir uma pequena quantidade de sensores inteligentes, assim como também pode ser formada por centenas ou milhares de nós sensores. Dependendo da aplicação, esse número pode chegar à casa de milhões de sensores inteligentes (AKYILDIZ *et al.*, 2001; LOUREIRO *et al.*, 2003). Sendo assim, o projeto de mecanismos de RSSF, em geral, deve levar em consideração o tipo de aplicação e as quantidades mínimas e máximas de nós sensores a serem utilizados.

Outro aspecto relacionado com a escalabilidade é a densidade de uma RSSF. Essa característica está relacionada com a proporção do número de sensores em uma determinada área na qual a rede é implantada. De acordo com (BULUSU *et al.*, 2001), a densidade $\mu(N)$ de uma RSSF pode ser estimada a partir da Equação 1, em que N é o número de nós sensores dispersos em uma região de área A e R é o alcance de transmissão do rádio, considerado idealmente circular.

$$\mu(N) = \frac{N \times \pi \times R^2}{A} \quad (1)$$

Com um grande número de nós sensores, as RSSF podem apresentar um alto valor de densidade, exigindo mecanismos e protocolos de roteamento que atendam o nível de complexidade imposto por essa característica da rede.

2.2.4 Comunicação e Colaboração

A comunicação entre os sensores inteligentes em uma RSSF é realizada por meio de componentes de rádio, utilizando-se o ar como meio de propagação do sinal. Além disso, os nós sensores possuem alcance de transmissão limitado e, na maioria das aplicações, um nó sensor não pode se comunicar diretamente (dentro do raio de transmissão) com todos os demais nós sensores da rede. Diante dessas características, as RSSF utilizam o processo de comunicação *multi-hop*, que consiste na transferência de dados de um nó sensor para outro, que está fora de seu alcance de transmissão, utilizando nós sensores intermediários (LOUREIRO *et al.*, 2003). Vale ressaltar que esse tipo de comunicação é essencial para a escalabilidade de uma RSSF, pois

diferentemente do processo de comunicação *single-hop*, as redes não são limitadas espacialmente pelo alcance do rádio e podem abranger longas distâncias com fontes de energia limitadas.

Na maioria das aplicações de RSSF, os dados coletados pelos nós sensores são enviados para um dispositivo responsável pela comunicação da rede com o mundo exterior, chamado de ponto de acesso (RUIZ *et al.*, 2004). Esse ponto de acesso é implementado em um nó diferenciado da RSSF chamado de nó sorvedouro[2], pois esse nó geralmente possui menores limitações de recursos computacionais do que os demais sensores inteligentes da rede. Um diagrama simplificado de uma RSSF contendo nós sensores e um nó sorvedouro é mostrado na Figura 1, no qual os segmentos de reta representam a conectividade entre os sensores inteligentes da rede.

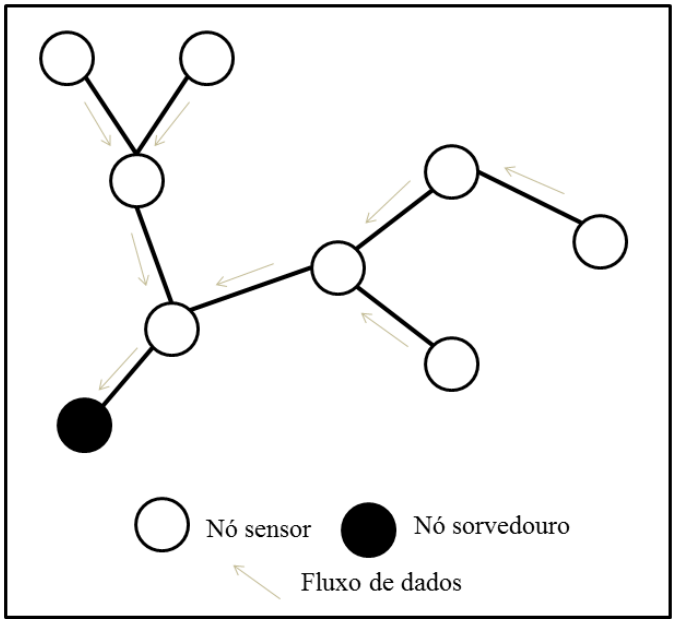

Figura 1. Diagrama simplificado de uma RSSF.

Conforme pode ser observado na figura 1, o nó sorvedouro recebe as informações coletadas pelos sensores inteligentes da rede. Esse modelo é muito utilizado em aplicações de RSSF, evidenciando o caráter colaborativo das RSSF, uma vez os nós sensores devem trabalhar de forma cooperativa para que uma mensagem percorra da origem até o destino com sucesso. Vale ressaltar que existem diversas tarefas para diminuir o consumo de energia na RSSF que

[2] Tradução do termo em inglês *sink node*. Esse nó também é conhecido como “estação base”.

exigem o trabalho de forma colaborativa, entre as quais se destacam a agregação de dados, monitoramento de objetos móveis e a sincronização de tempo (BORGES NETO, 2010).

2.2.5 Mobilidade

De acordo com Bhattacharyya *et al.* (2010), as RSSF podem ser estáticas, quando os sensores inteligentes que compõem a rede não possuem capacidade de movimentação, ou móveis, quando os nós sensores podem se movimentar no ambiente na qual a rede foi implantada. Existe ainda a possibilidade de a rede ser composta por ambos os tipos de dispositivo, resultando em uma classificação híbrida. A capacidade de mobilidade dos nós sensores em uma RSSF implica na possibilidade de alteração frequente da configuração topológica da rede. Isso aumenta bastante a complexidade no projeto de mecanismos ou protocolos para tratamento de questões de roteamento e transporte das mensagens na rede. Vale ressaltar que o aumento da complexidade de mecanismos utilizados em RSSF geralmente implica no aumento do consumo de energia da rede.

Além disso, a característica de mobilidade de uma RSSF pode afetar a manutenção da qualidade de serviço, uma vez que a mobilidade pode causar uma falha nas rotas *multi-hop* e, consequentemente, perdas de comunicação (BORGES NETO, 2010). É bom lembrar que o fornecimento de QoS em RSSF é um fator difícil de ser estabelecido tendo em vista as dificuldades no gerenciamento de parâmetros de QoS (por exemplo, conectividade da rede), o que implica em gasto adicional de energia (LOUREIRO *et al.*, 2003). Sendo assim, em aplicações de RSSF que exigem garantias de QoS devem ser tratadas as questões envolvidas com a mobilidade na rede.

2.2.6 Segurança

Conforme mencionado na seção 1.1, as RSSF utilizam o ar como meio de propagação do sinal de comunicação. Dessa forma, essas redes são vulneráveis a diversos tipos de ataques. Por exemplo, um adversário pode facilmente recuperar informações valiosas a partir do conteúdo dos dados que estão sendo transmitidos em uma RSSF (*eavesdropping*). De outra maneira, um adversário pode interceptar e modificar o conteúdo de um pacote transmitido para um nó sorvedouro ou nó sensor intermediário (modificação de mensagens), além de poder retransmitir

um pacote legítimo em um tempo posterior (repetição). Em outro caso, um adversário pode enviar um pacote falso, se passando por um nó sensor legítimo da rede, com a finalidade de distorcer os dados coletados pelos sensores inteligentes (injeção de mensagens) (CASTRO, 2008).

As possibilidades de ataques descritas anteriormente evidenciam a necessidade de serviços de segurança para garantir o funcionamento de diversas aplicações de RSSF. Por exemplo, a ocorrência de um ataque de *eavesdropping* em uma aplicação militar pode comprometer o objetivo da RSSF, uma vez que o inimigo pode monitorar o tráfego de dados da rede e obter informações privilegiadas.

Por outro lado, existem aplicações que não necessitam de um nível alto de segurança (e.g., aplicações de monitoramento de espécies animais). Sendo assim, o nível de segurança a ser estabelecido em uma RSSF depende fortemente da aplicação, dependendo de diversos fatores como: o ambiente no qual a rede será instalada, o ambiente, as ameaças esperadas, o nível de criticidade dos dados e o ganho esperado de um ataque à rede. (SABBATH *et al.*, 2006).

As características discutidas nesta subseção, assim como outros aspectos de segurança em RSSF, assunto principal desta dissertação, são apresentados e discutidos de forma mais detalhada no Capítulo 3.

2.3 Aplicações

As RSSF possuem um grande potencial de aplicações, podendo ser aplicadas nas mais variadas áreas que vão do monitoramento ambiental (SZEWCZYK *et al.*, 2004) à medicina (JANANI *et al.*, 2011). Muitas vezes, elas são implantadas em locais de difícil acesso onde não há possibilidade de intervenção humana seja para recarregamento de baterias ou para movimentação física dos sensores inteligentes (LOUREIRO *et al.*, 2003), demonstrando a importância da minimização do consumo de energia em RSSF.

Nas próximas subseções, alguns exemplos práticos de aplicações reais de RSSF publicadas na literatura são apresentados e discutidos, evidenciando as vantagens da utilização desse tipo de rede. Além disso, as aplicações mencionadas estão categorizadas de acordos com a área na qual a RSSF foi implantada.

2.3.1 Monitoramento Ambiental

As aplicações de monitoramento ambiental geralmente consistem na coleta de dados do ambiente (por exemplo, temperatura, pressão atmosférica ou umidade) pelos sensores inteligentes e o envio desses dados para uma estação central, responsável pelo tratamento dos dados. Vale ressaltar que o monitoramento ambiental engloba também aplicações de monitoramento e rastreamento de espécies animais e o monitoramento de condições climáticas.

Conforme indicado por (SZEWCZYK *et al.*, 2004), as RSSF apresentam grandes vantagens para o monitoramento ambiental. Entre elas, pode-se citar:

- as RSSF podem ser compostas por vários sensores inteligentes para cobrir uma grande área geográfica, aumentando a confiança estatística dos dados;

- os sensores inteligentes podem ser suficientemente pequenos para que sejam discretamente implantados sem influenciar o comportamento do ambiente nem comprometer os dados coletados; e

- a RSSF pode ser implantada em ambientes hostis, onde inexiste a possibilidade de intervenção humana.

Uma aplicação de RSSF de monitoramento ambiental, talvez a aplicação mais conhecida de RSSF, foi descrita em (MAINWARING *et al.*, 2002) e realizada com a parceria de biólogos. Essa aplicação foi implantada em *Great Duck Island*, localizada no estado de Maine (Estados Unidos), para realizar o monitoramento de pássaros marinhos chamados *Leach's Storm Petrel*. A aplicação tinha como objetivo estudar o comportamento dessas aves no período de reprodução, englobando o padrão de escolha de ninhos, a entrada e saída dos ninhos, bem como a influência causada no ambiente dentro e fora de suas tocas.

Dessa forma, foram implantados 32 nós sensores dentro e na superfície dos ninhos, sendo que os sensores inteligentes eram dotados com capacidade de sensoriamento de temperatura,

umidade, pressão e intensidade de luz. Esses nós sensores trabalhavam cooperativamente, enviando as informações coletadas para uma estação central.

Outro exemplo prático de monitoramento ambiental é descrito em (WERNER-ALLEN *et al.*, 2006). Essa aplicação consiste no monitoramento de um vulcão ativo localizado no Equador e chamado de *Reventador*. Uma vez que sistemas típicos de monitoramento de vulcão empregam equipamentos volumosos e difíceis de implantar, Werner-Allen *et al.* (2006) utilizaram uma RSSF composta por 16 sensores inteligentes com capacidade de medição de dados sísmicos e acústicos para monitorar o estado do vulcão equatoriano.

Enfim, os dois exemplos citados ilustram o potencial de aplicação de RSSF para realizar atividades de monitoramento ambiental e demais atividades similares. Além disso, as RSSF podem ser empregadas para resolver tarefas complexas com eficiência que seriam difíceis, ou até impossíveis, de realizar com equipamentos convencionais.

2.3.2 Operações Militares

Conforme mencionado na seção 2.1, o potencial de uso em aplicações militares foi o principal fator que impulsionou o surgimento de RSSF. Nessa área, as RSSF podem ser utilizadas para realizar a vigilância em campos de batalha, a detecção, rastreamento e classificação de inimigos (Li *et al.*, 2002), entre outras aplicações.

Uma demonstração, conforme descrito em (SINOPOLI *et al.*, 2003), aborda o rastreamento de veículos em um jogo de perseguição-evasão (*PEG - pursuit-evasion game*). Nessa abordagem, há duas equipes competitivas, os perseguidores e os evasores. Além disso, há uma terceira parte formada por uma RSSF, a qual é responsável por ajudar os perseguidores a localizar seus adversários. Assim, a posição relativa e a movimentação das unidades inimigas são informadas aos perseguidores pela RSSF. Dessa forma, a rede atua aumentando a "visão" da equipe de perseguidores sobre os inimigos. Os autores avaliaram o sistema por meio do *testbed* mostrado na Figura 2.

Outro exemplo de aplicação militar (SIMON *et al.*, 2004) objetiva a localização de atiradores e a definição da trajetória das balas. O sistema é formado por nós sensores que podem detectar a explosão do disparo e a onda de choque e medir o tempo de chegada por meio de

sensores acústicos. Ao comparar esse tempo de chegada à RSSF, o atirador pode ser localizado com uma precisão de um metro e latência de menos de dois segundos.

Enfim, a área militar continua impulsionando o emprego da tecnologia de RSSF para realizar operações afins, reforçando o potencial desse paradigma de comunicação para uma grande diversidade de aplicações.

Figura 2. Testbed utilizado por (SINOPOLI et al., 2003).

2.3.3 Saúde

As RSSF também podem ser utilizadas em várias situações na área de saúde como, por exemplo, no monitoramento integrado de pacientes, na administração de drogas em hospitais ou o monitoramento de médicos e pacientes dentro de um hospital (AKYILDIZ *et al.*, 2001).

A tecnologia sem fio ajuda a resolver alguns inconvenientes associados aos sensores com fio comumente utilizados em salas de emergências e hospitais para o acompanhamento de pacientes. A grande quantidade de fios ligados a um paciente é problemática tanto para o paciente, por causar desconforto e restringir sua mobilidade, quanto para os profissionais da saúde, pela dificuldade de gerenciar os equipamentos. Assim os sensores inteligentes diminuem o desconforto dos pacientes por serem menos perceptíveis, ajudam a reduzir o emaranhado de fios, além de reduzir a ocorrência de erros (KO *et al.*, 2010b).

Um exemplo prático desse tipo de aplicação é o MEDiSN (KO *et al.*, 2010a), uma RSSF para automatizar o processo de monitoramento de pacientes. Esse sistema é composto por nós sensores dotados de capacidade de sensoriamento de nível de oxigenação do sangue, pulso radial, eletrocardiograma, pressão arterial, entre outros. Os autores mostraram que o MEDiSN proporciona medições de sinais vitais que são estatisticamente idênticos aos gerados pelos equipamentos convencionais com fios (*wired*).

O uso da tecnologia de RSSF na área médica está em constante evolução. De acordo com Darwish e Hassanien (2011), as RSSF poderão ser implantadas na superfície e até mesmo dentro do corpo humano formando uma WBAN (*Wireless Body-Area Network*). As WBANs poderão realizar o monitoramento de sinais vitais e prover *feedback* em tempo real para permitir o diagnóstico de pacientes. Além disso, o desenvolvimento de sensores inteligentes em escala nanométrica[3] proporcionará, por exemplo, o monitoramento de componentes do sangue, monitoramento do colesterol e detecção de agentes infecciosos dentro do corpo humano, representando um grande avanço para área médica (KUMAR *et al.*, 2011).

2.4 Conclusão

As RSSF são redes compostas por dispositivos providos de comunicação sem fio com capacidade de processamento e recursos de computação limitados. Essas redes trabalham de forma colaborativa com o objetivo de resolver um problema específico. Além das limitações, as RSSF podem possuir nós sensores móveis e são potencialmente escaláveis. Além do mais, vale ressaltar que as aplicações de RSSF devem ser tolerantes a falhas e podem necessitar de serviços de segurança da informação. Dessa forma, o projeto de mecanismos e protocolos para RSSF deve levar em consideração essas características, principalmente a limitação de fonte de energia disponível.

É bom lembrar ainda que as RSSF possuem um grande potencial de aplicações, podendo ser empregadas em áreas que vão desde o monitoramento ambiental até a assistência médica. Exemplos práticos publicados na literatura demonstram a eficiência do emprego das RSSF para realizar tarefas de monitoramento e controle, apresentando grandes vantagens com relação às tecnologias convencionais.

Portanto, as RSSF estão cada vez mais consolidadas, possibilitando a execução de aplicações desafiadoras e podendo ser integradas a sistemas maiores. Além disso, o emprego dessa tecnologia no futuro na área da saúde poderá causar grande impacto na melhoria da qualidade vida das pessoas.

[3] O sensores inteligentes em escala nanométrica são também chamados de nanosensores.

3 SEGURANÇA EM REDES DE SENSORES SEM FIO

Este capítulo apresenta os principais conceitos e as principais características envolvidas com a segurança da informação no contexto de Redes de Sensores Sem Fio essenciais para o desenvolvimento deste trabalho. A seção 3.1 apresenta uma visão geral da segurança em RSSF. Em seguida, os principais serviços de segurança são descritos na seção 3.2. Os mecanismos de segurança são apresentados na seção 3.3 e os ataques à segurança são caracterizados na seção 3.4. Por fim, a seção 3.5 encerra o capítulo.

3.1 Introdução

Conforme caracterizado na seção 1.1, o aumento de aplicações de RSSF, bem como a diversificação das mesmas, enseja a preocupação com a segurança da informação nessas redes. Além disso, a utilização do ar como meio de propagação do sinal torna as RSSF vulneráveis a diversos tipos de ataques, conforme indicado na subseção 2.2.6.

Os mecanismos de segurança são projetados com a finalidade de detectar, impedir ou até permitir a recuperação de um ataque à segurança, oferecendo um ou mais serviços de segurança (STALLINGS, 2010). Por exemplo, a criptografia é um mecanismo de segurança utilizado para fornecer o serviço de confidencialidade de um sistema.

Por outro lado, as limitações características das RSSF (ver subseção 2.2.1) fazem com que o provimento de serviços de segurança nessas redes seja desafiante. Dessa forma, o desempenho pode atuar como fator limitante da utilização de mecanismos de segurança, uma vez que o funcionamento de mecanismos convencionais pode não ser viável no contexto de RSSF.

3.2 Serviços de Segurança

Um ataque pode comprometer a segurança da informação de sistema como, por exemplo, uma RSSF. Assim, de acordo com Stallings (2010), um serviço de segurança pode ser entendido como um serviço de processamento ou comunicação capaz de aumentar a segurança dos sistemas

de processamento de dados bem como as transferências de informação de uma organização, servindo para frustar ataques à segurança da informação. No contexto de RSSF, os serviços básicos de segurança, tais como a criptografia e a autenticação, podem ser necessários em grande parte das aplicações. Esses e outros serviços são caracterizados nas próximas subseções.

3.2.1 Confidencialidade

A confidencialidade é um serviço que fornece condições de garantir que uma informação será acessada somente por dispositivos autorizados, sendo um serviço essencial para muitas aplicações, como as militares (SARAOGI, 2004). Assim, em uma RSSF que oferece o serviço de confidencialidade, mesmo que um atacante intercepte um pacote da rede, ele não deve ser capaz de entender o seu conteúdo.

É importante ressaltar que a confidencialidade também pode incluir o serviço de *segurança semântica*, quando ela evita não apenas o entendimento da informação completa, mas também o entendimento parcial da informação (MENEZES *et al.*, 1996).

3.2.2 Integridade

A integridade é um serviço que fornece condições de identificar se o conteúdo de uma mensagem foi alterado após o seu envio (WALTERS *et al.*, 2006). Dessa forma, em uma RSSF que provê o serviço de integridade dos dados, um sensor inteligente ao receber uma mensagem poderá descartá-la caso identifique que o conteúdo da mensagem recebida sofreu alteração ou aceitá-la, caso contrário. Além disso, o serviço de integridade pode ser utilizado para detectar erros de transmissão na recepção de mensagens.

3.2.3 Autenticação

A autenticação é o serviço que fornece condições de garantir que um determinado participante da rede é autêntico (WALTERS *et al.*, 2006). Dessa forma, em uma troca de mensagens, o serviço de autenticação fornece ao destinatário a possibilidade de garantir a autenticidade do remetente.

Com o serviço de autenticação, os nós sensores de uma RSSF terão capacidade de verificar e aceitar as mensagens de participantes legítimos da rede e rejeitar essas mensagens, caso contrário. Isso é extremamente importante principalmente durante atividades como, por exemplo, a reprogramação dos sensores inteligentes ou o envio de comandos de atuação em uma RSSF.

3.2.4 Disponibilidade

O serviço de disponibilidade consiste na garantia da capacidade dos dispositivos de fornecer as funcionalidades de acordo com o esperado. De acordo com Zhou e Fang (2009), a disponibilidade é provavelmente o serviço mais abrangente, pois envolve quase todos os aspectos de uma rede. Sendo assim, qualquer problema em uma rede pode resultar na degradação do funcionamento da rede e, assim, comprometer a sua disponibilidade, o que pode ser causado por ataques conhecidos como negação de serviço (*DoS - Denial of Service*).

3.2.5 Freshness

Em uma RSSF, um adversário poderá interceptar mensagens e armazená-las para reenviá-las em um momento futuro com o objetivo de causar distorção nos dados recebidos ou até mesmo uma atuação equivocada em redes dotadas de nós sensores atuadores. Esse ataque é conhecido como ataque do tipo *replay* ou ataque de repetição (CASTRO, 2008).

Dessa forma, somente autenticação, integridade e confidencialidade não são suficientes para proteger contra esse tipo de ataque, uma vez que as mensagens serão decifradas e autenticadas como mensagens legítimas, pois não sofrem modificações. Uma forma de proteção contra ataques de repetição é atribuir um valor numérico diferente a cada mensagem. Por exemplo, usando-se um contador, incrementado a cada mensagem enviada, é possível estabelecer uma relação entre o "frescor" da mensagem e o valor do contador incluído na mensagem. Esse tipo de proteção é chamado de *freshness* (SARAOGI, 2004).

3.2.6 Outros Serviços de Segurança

É bom salientar que existem outros serviços de segurança, mas que ou são muito improváveis para RSSF ou exigem mecanismo muito dispendiosos para o contexto de RSSF. Um exemplo é o serviço de irretratabilidade (ou não-repúdio), que consiste na capacidade de impedir que um dos participantes da rede envolvidos em uma comunicação negue ter participado de toda ou parte da comunicação (BRAGA, *et al.*, 1998). Esse serviço exige protocolos complexos com grande *overhead* de comunicação e processamento que são inviáveis para o contexto de RSSF atuais, compostas por dispositivo com recursos limitados (ZHOU; FANG, 2009).

3.3 Mecanismos de Segurança

Conforme indicado por Stallings (2010), um mecanismo de segurança é um processo que é projetado para detectar, impedir ou permitir a recuperação de um ataque à segurança. Assim, um ou mais mecanismos de segurança são utilizados para prover serviços de segurança, visando combater determinados tipos de ataques. Nas próximas subseções são apresentados os conceitos e as características que envolvem os mecanismos de segurança estudados nesta dissertação.

3.3.1 Criptografia de Chave Pública

A criptografia de chave pública funciona conforme o esquema mostrado na Figura 3. Nesse esquema, cada participante possui duas chaves, sendo uma privada secreta, utilizada para decriptografar mensagens, e outra pública não secreta, usada na encriptação. Baseando-se na Figura 3, quando o dispositivo A envia uma mensagem para B deve ocorrer o seguinte fluxo de eventos:

- o dispositivo A criptografa a mensagem com a chave pública do dispositivo B, lembrando que a chave pública de B não é secreta e é conhecida por A;

- o dispositivo A envia a mensagem que foi criptografada com a chave pública de B para o dispositivo B; e

- o dispositivo B decriptografa a mensagem usando a sua chave privada, conhecida apenas por ele mesmo. Vale ressaltar que essa mensagem somente pode ser decriptografada por essa chave privada de B.

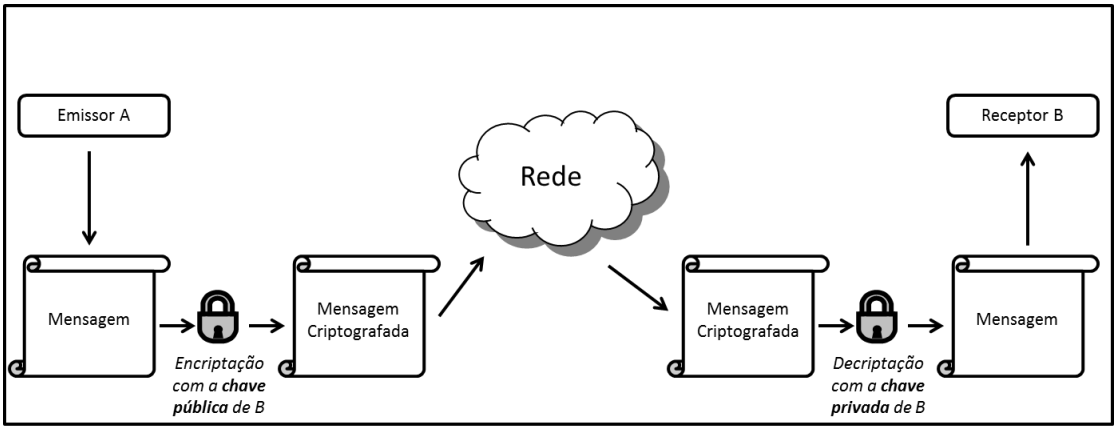

Figura 3. Criptografia assimétrica. Fonte: adaptado de (KAHATE, 2008).

O envio de mensagens do dispositivo B para o dispositivo A ocorre de forma análoga à anteriormente descrita.

Uma vez que os mecanismos de chave assimétrica apresentam, em geral, uma maior complexidade e, consequentemente, um baixo desempenho quando comparados com os algoritmos de chave simétrica (SARAOGI, 2004), esses últimos são mais adequados para prover confidencialidade em RSSF. Por exemplo, o mecanismo de criptografia RSA, considerado a técnica de uso geral mais aceita e implementada para a criptografia de chave pública (STALLINGS, 2010), apresenta péssimos resultados experimentais, requerendo quantidades excessivas de energia para realizar suas operações no contexto de RSSF (AMIN *et al.*, 2008). Dessa forma, este trabalho foca nos mecanismos de criptografia simétrica, os quais são discutidos na próxima subseção.

3.3.2 Criptografia Simétrica

De acordo com Stallings (2010), a criptografia simétrica era o único tipo de criptografia utilizado antes do surgimento da criptografia de chave pública (ver subseção 1.3.1) na década de

1970 e continua sendo o mais usado dentre os dois tipos de criptografia. Ao contrário do que ocorre na criptografia assimétrica, uma única chave é utilizada na encriptação e decriptação de uma mensagem na criptografia simétrica.

Antes de definir formalmente um esquema de criptografia simétrica, cujo modelo simplificado é mostrado na Figura 4, é importante o esclarecimento sobre os seguintes ingredientes que os compõem (STALLINGS, 2010):

- **texto claro** é a mensagem original não criptografada e inteligível;

- **algoritmo de criptografia, cifragem ou encriptação** é o algoritmo responsável pelo processo de converter um texto claro em um texto cifrado;

- **chave secreta**: é a entrada para o algoritmo de criptografia, sendo um valor independente do texto claro e do algoritmo, produzirá uma saída diferente para cada valor de chave específica utilizada;

- **texto cifrado**: é a mensagem codificada; e

- **algoritmo de decriptografia, decifragem ou decriptação**: é o algoritmo responsável pelo processo inverso do algoritmo de criptografia que restaura o texto claro a partir do texto cifrado.

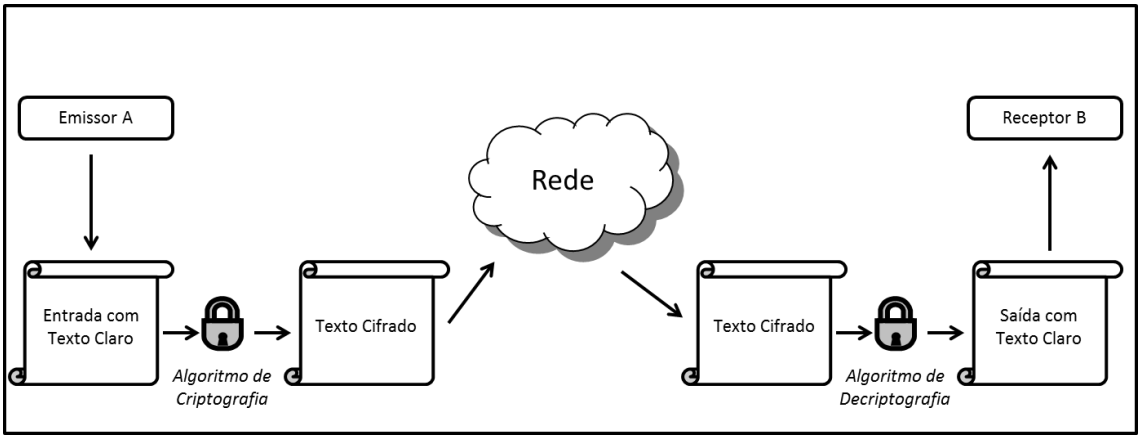

Figura 4. Modelo simplificado de criptografia simétrica. Fonte: adaptado de (STALLINGS, 2010).

Um esquema de criptografia simétrica, também chamado de cifra simétrica, pode ser definido, baseando-se nas definições de Hayashi e Tanaka (2006) e Delfs e Knebl (2007), da seguinte maneira:

> **Definição 1**. *Seja $\mathcal{M}$ um conjunto de textos claros, $\mathcal{K}$ um conjunto de chaves e C um conjunto de textos cifrados. Um esquema de criptografia simétrica $\prod = \{\mathcal{E},\mathcal{D}\}$ consiste em dois algoritmos:*
>
> - *o algoritmo de encriptação $\mathcal{E}$, um algoritmo determinístico que recebe como entrada um texto claro $m \in \mathcal{M}$ e uma chave $k \in \mathcal{K}$ e devolve um texto cifrado $c \in C$, denotado por $c \leftarrow \mathcal{E}_k(m)$; e*
>
> - *o algoritmo de decriptação $\mathcal{D}$, um algoritmo determinístico que recebe como entrada um texto cifrado $c \in C$ e uma chave $k \in \mathcal{K}$ e devolve um texto claro $m \in M$ correspondente, denotado por $m \leftarrow \mathcal{D}_k(c)$;*
>
> *Além disso, é necessário que: se $m \in \mathcal{M}$, $k \in \mathcal{K}$, e $c \leftarrow \mathcal{E}_k(m)$, então $m \leftarrow \mathcal{D}_k(c)$.*

É importante ressaltar que a operação dos algoritmos simétricos, assim como todos os algoritmos de criptografia, é baseada nos princípios gerais de substituição, em que cada elemento no texto claro é mapeado em outro elemento, e transposição, em que os elementos no texto claro são reorganizados (STALLINGS, 2010). De outra forma, para referenciar os dois componentes básicos de qualquer sistema criptográfico, Shanon (1949) introduziu os termos confusão, que busca tornar o relacionamento entre as estatísticas do texto cifrado e o valor da chave de criptografia o mais complexo possível, e difusão, que busca maximizar a complexidade do relacionamento estatístico entre o texto claro e o texto cifrado.

As cifras simétricas podem ser classificadas em duas categorias: cifras de fluxo (*stream cipher*) e cifras de bloco (*block cipher*), discutidas nas próximas subseções.

3.3.2.1 Cifra de Fluxo

Para definir uma cifra de fluxo é importante caracterizar o *one-time pad* ou cifra de chave de uso único, pois, conforme indicado por Stallings (2010), as cifras de fluxo funcionam de modo semelhante.

O *one-time pad* consiste em um sistema que produz uma saída aleatória que não possui nenhum relacionamento estatístico com o texto claro, sendo dessa forma um sistema de

criptografia incondicionalmente seguro. Isso ocorre devido ao *one-time pad* utilizar uma chave aleatória tão grande quanto o tamanho do texto claro. Porém, esse sistema é praticamente inviável por dois motivos principais: a incapacidade de geração de chaves verdadeiramente aleatórias e a complexidade gigantesca de distribuição de chaves (STALLINGS, 2010).

A diferença entre as cifras de fluxo e o *one-time pad* é que este último usa um fluxo de chaves verdadeiramente aleatórias, enquanto as cifras de fluxo utilizam uma chave de tamanho fixo para gerar um fluxo de chaves de forma pseudoaleatória. Formalmente, baseado na definição de uma cifra de fluxo de Delfs e Knebl (2007), as cifras de fluxo podem ser definidas da seguinte forma:

> **Definição 2**. *Seja $\mathcal{M}$ um conjunto de textos claros, $\mathcal{K}$ um conjunto de chaves e $\mathcal{C}$ um conjunto de textos cifrados. Uma cifra de fluxo é um esquema de criptografia simétrica* $\prod = \{\mathcal{E},\mathcal{D}\}$, *tal que:*
>
> - *o algoritmo $\mathcal{E}$ recebe como entrada duas sequências de caracteres, $m_1m_2m_3... \in \mathcal{M}$ e $k_1k_2k_3... \in \mathcal{K}$, chamado keystream, e devolve uma sequência de caracteres de texto cifrado $c_1c_2c_3... \in \mathcal{C}$, em que $c_i \leftarrow \mathcal{E}_{ki}(m_i)$. i =1, 2, ...; e*
>
> - *o algoritmo $\mathcal{D}$ recebe como entrada duas sequências de caracteres, $c_1c_2c_3... \in \mathcal{C}$ e $k_1k_2k_3... \in \mathcal{K}$, e devolve uma sequência de caracteres de texto claro $m_1m_2m_3... \in \mathcal{M}$, em que $m_i \leftarrow \mathcal{D}_{ki}(c_i)$. i =1, 2, ...*

As cifras de fluxo geralmente são mais rápidas dos que as cifras de bloco. Entretanto, a segurança das cifras de fluxo depende fortemente da geração de um fluxo pseudoaleatório de chaves. Dessa forma, elas são pouco recomendadas para aplicações nas quais a repetição de chaves após um curto período de tempo seja uma realidade, como geralmente é o caso das RSSF. Assim, as cifras de bloco são mais atrativas para RSSF (Simplício; 2008) e são abordadas na próxima subseção.

3.3.2.2 Cifra de Bloco

Ao contrário das cifras de fluxo, que codifica um fluxo de dados digital um *bit* ou byte de cada vez, as cifras de bloco dividem uma mensagem em blocos de tamanho fixo e realizam a operação de cifragem em cada bloco gerando blocos de texto cifrado do mesmo tamanho.

Baseando-se na definição de Delfs e Knebl (2007) e Santos (2009), uma cifra de bloco pode ser definida da seguinte maneira:

> **Definição 3**. *Seja $\mathcal{M}=\{0,1\}^n$ um conjunto de textos claros, $\mathcal{K}=\{0,1\}^r$ um conjunto de chaves e $C=\{0,1\}^n$ um conjunto de textos cifrados. Uma cifra de bloco é um esquema de criptografia simétrica $\prod = \{\mathcal{E},\mathcal{D}\}$, tal que:*
>
> - *o algoritmo $\mathcal{E}$ implementa a seguinte função: $E : \{0,1\}^r \times \{0,1\}^n \rightarrow \{0,1\}^n$, tal que para cada $k \in \mathcal{K}$, a função $E_k : \{0,1\}^n \rightarrow \{0,1\}^n$ é inversível e denotada por $c \leftarrow E_k(m)$, em que $m \in \mathcal{M}$ e $c \in C$;*
> - *e o algoritmo $\mathcal{D}$ implementa a função inversa de E, denotada por $m \leftarrow E_k^{-1}(c)$.*
>
> *OBS 1. $\{0,1\}^n$ representa uma cadeia binária de tamanho n.*

De acordo com a Definição 3, numa cifra de bloco utilizando uma chave secreta *k* de tamanho *r*, o algoritmo $\mathcal{E}$ criptografa blocos de texto claro *m* de tamanho *n* gerando um bloco de texto cifrado *c* de mesmo tamanho *n*. Nesse caso, *n* é chamado de tamanho do bloco (*block length*) (DELFS; KNEBL, 2007). Além disso, os algoritmos de grande parte das cifras de bloco consistem em várias iterações (*rounds*) de uma mesma função. Em cada repetição, uma nova sub-chave, derivada da chave principal, é utilizada por meio do processo de escalonamento de chaves (DELFS; KNEBL, 2007), também conhecido como expansão da chave ou ainda *key setup*.

3.3.3 Modo de Operação

Os modos de operação são utilizados com as cifras de bloco com o objetivo de suportar a encriptação de mensagens cujo tamanho seja maior que o tamanho de bloco padrão da cifra. Eles geralmente utilizam um vetor de inicialização (IV – *initialization vector*) que é o primeiro bloco a ser utilizado na encriptação e decriptação correspondente (DOWRKING, 2001) e gerado frequentemente de forma pseudoaleatória para garantir a segurança semântica, evitando que uma mesma mensagem criptografada seguidas vezes resulte em textos cifrados diferentes.

3.3.4 Código de Autenticação de Mensagens

Um código de autenticação de mensagem (*MAC*[4] *– message authentication code*) é um algoritmo utilizado para autenticar mensagens, provendo o serviço de autenticação. Apesar de ser possível construir um MAC também com uma cifra de bloco simétrica[5] (SCHNEIER, 1996), este trabalho define um MAC, baseando-se nas definições de Delfs e Knebl (2007), a partir de uma *função hash* e da definição de colisão.

Definição 4. *Seja $\mathcal{M}=\{0,1\}^*$ um conjunto de textos claros de tamanho variável e $C=\{0,1\}^n$ um conjunto de códigos. Uma função hash é uma função $h : \{0,1\}^* \rightarrow \{0,1\}^n$, denotada por $c \leftarrow h(m)$, em que $m \in \mathcal{M}$ e $c \in C$;*

Além disso, uma função hash deve possuir as seguintes propriedades:

1. *Deve ser computacionalmente fácil obter calcular $h(m)$;*
2. *Dado $y \in C$, deve ser impraticável computacionalmente encontrar m tal que $y \leftarrow h(m)$.*

Definição 5. *Seja $\mathcal{M}=\{0,1\}^*$ um conjunto de textos claros de tamanho variável, $\mathcal{K}=\{0,1\}^r$ um conjunto de chaves e $C=\{0,1\}^n$ um conjunto de códigos. Um MAC é um algoritmo que implementa a função que combina uma função hash com uma chave $h_k : \{0,1\}^r \times \{0,1\}^* \rightarrow \{0,1\}^n$, denotada por $c \leftarrow h_k(m)$, em que $m \in \mathcal{M}$, $k \in \mathcal{K}$ e $c \in C$.*

Além disso, um MAC deve possuir as seguintes propriedades:

1. *Deve se computacionalmente fácil calcular $h_k(m)$ a partir de m e k;*
2. *Dado $y \in C$, deve ser impraticável computacionalmente encontrar m tal que $y \leftarrow h_k(m)$.*

De acordo com a Definição 5, um MAC recebe como entrada uma chave e uma mensagem a ser autenticada e fornece como saída um código, muitas vezes chamado de *tag*, que possui uma relação unívoca com a entrada. Assim, a aplicação de um MAC em uma mensagem resulta em uma *tag* única. Além disso, dado um $h(m)$, um adversário não deve ter chance de encontrar uma $m' \neq m$ tal que $h(m) = h(m')$. Tal ocorrência é chamada de *colisão* (DELFS; KNEBL, 2007).

[4] Embora o termo MAC possua outro significado na área tradicional de redes (*Media Access Control*), o mesmo termo é utilizado nesta dissertação com o significado *Message Authentication Code*, por ser bastante difundido na literatura.

[5] Dois exemplos de MACs criados a partir de cifras de bloco simétricas são descritos no Capítulo 4.

Dessa forma, caso uma mensagem seja alterada após sua transmissão ou enviada de um dispositivo não autêntico que não possui a chave secreta compartilhada, o receptor pode verificar a autenticidade da mensagem simplesmente aplicando o MAC e comparando a *tag*, possibilitando, de acordo com o resultado dessa comparação, validar ou rejeitar a mensagem, conforme mostrado na Figura 5. É importante observar, então, que um MAC, além de prover autenticação, também fornece o serviço de integridade.

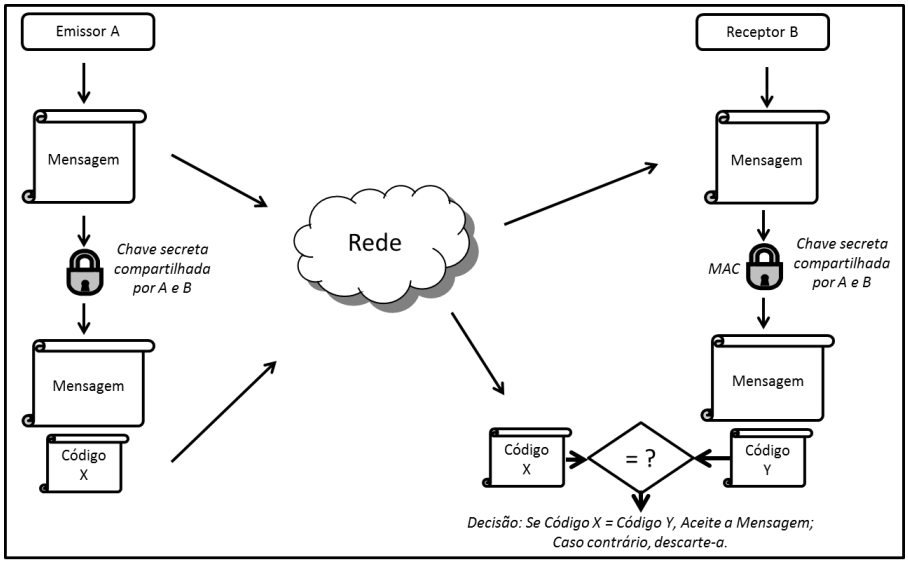

Figura 5. Modelo simplificado típico do uso de um MAC. Fonte: adaptado de (SANTOS, 2009).

3.3.5 Sistemas de Detecção de Intrusão

Os mecanismos descritos anteriormente objetivam reduzir a probabilidade de possíveis adversários causarem algum tipo de prejuízo à rede por meio de um ataque. Entretanto, uma vez que muitas vezes os nós sensores são dispostos em áreas abertas sem proteção física, é possível que um atacante tenha acesso físico a um nó sensor e consiga capturar seus dados armazenados (chaves criptográficas, por exemplo) (LEMOS *et al.*, 2009).. Nesse caso é importante o emprego de um Sistema de Detecção de Intrusão (IDS – *Intrusion Detection System*) com a finalidade de detectar eventuais nós sensores não legítimos que consigam burlar os serviços básicos de segurança da rede (SILVA *et al.*, 2005).

Vale ressaltar que, apesar da importância de um IDS para elevar o nível de segurança em uma RSSF, o foco deste trabalho são os mecanismos capazes de prover os serviços básicos de segurança: confidencialidade, integridade e autenticação.

3.4 Ataques de segurança

Nesta seção são apresentados os principais aspectos sobre os tipos de ataques possíveis aos mecanismos de segurança e à RSSF.

3.4.1 Modelo de Adversário

Um adversário, ou atacante, pode ser caracterizado de acordo com o tipo de ataque que ele pode realizar (HU; SHARMA, 2005):

- **adversário passivo** – esse tipo de atacante ataca o serviço de confidencialidade, podendo escutar e monitorar as transmissões na RSSF com o objetivo de obter os dados trafegados na rede; e

- **adversário ativo** – esse atacante busca realizar alguma modificação do fluxo de dados que trafegam na RSSF, podendo realizar, por exemplo, ataques de repetição ou injeção de mensagens (ver subseção 1.2.6).

3.4.2 Sistemas de criptografia

Normalmente, o objetivo de atacar um sistema de criptografia é determinar a chave secreta que está sendo utilizada, em vez de simplesmente determinar o texto claro de um único texto cifrado. A posse da chave por um adversário representa o total comprometimento da rede, uma vez que todas as mensagens futuras e passadas, codificadas com essa chave, podem ser facilmente determinadas. De acordo com Stallings, (2010), existem duas técnicas gerais para atacar um esquema de criptografia simétrica:

- **criptoanálise –** os ataques criptoanalíticos exploram as características do algoritmo para tentar deduzir um texto claro específico ou deduzir a chave utilizada; e

- **ataque por força bruta** – nesse tipo de ataque, o atacante experimenta exaustivamente cada chave possível em um trecho de texto cifrado até obter uma tradução inteligível para texto claro. Para se obter sucesso, é necessário, na média, experimentar metade de todas as chaves possíveis.

Entre os principais ataques criptoanalíticos estão: o *ataque de texto cifrado*, no qual o criptoanalista conhece apenas o algoritmo de criptografia e o texto cifrado; o *ataque de texto claro conhecido*, no qual, além do algoritmo de criptografia e texto cifrado, o adversário conhece um ou mais pares de texto claro e texto cifrado com a chave secreta; e o *ataque de texto claro escolhido*, no qual o adversário pode obter o texto cifrado correspondentes de textos claros escolhidos. Em geral, um algoritmo de criptografia é projetado para resistir a um ataque de texto escolhido (STALLINGS, 2010).

3.4.3 Falsificação de MACs

Conforme indicado na subseção 3.3.4, o objetivo de um atacante consiste em encontrar colisões. Dessa forma, os ataques contra MACs podem ser classificados da seguinte forma (MENEZES *et al.*, 1996), sendo que a última delas é a mais crítica:

- ataque de texto conhecido, no qual o adversário possui um ou mais pares de texto claro e código correspondente;

- ataque de texto escolhido, onde o adversário possui um ou mais pares de texto claro escolhido e código correspondente; e

- ataque de texto escolhido adaptativo, no qual o texto claro pode ser escolhido como o caso anterior, mas ainda é possível escolhas subsequentes com base nos resultados anteriores.

Além disso, quando um adversário consegue falsificar um código MAC, a severidade do ataque depende de como ele pode falsificá-lo, conforme a seguinte classificação (MENEZES *et al.*, 1996):

- **falsificação seletiva**, que é caracterizada quando o adversário está apto a produzir um novo código a partir de um texto claro a sua escolha; e

- **falsificação existencial**, que ocorre quando o adversário consegue produzir um novo código, mas sem controle sobre o texto claro correspondente.

Obviamente, a possibilidade de falsificação seletiva consiste na ameaça mais crítica à segurança de uma RSSF com relação ao MAC, uma vez que esse tipo de ataque é equivalente ao fato de um adversário possuir conhecimento da chave secreta utilizada pelos sensores inteligentes para autenticar as mensagens da rede.

3.4.4 Ataques de segurança em RSSF

Além dos ataques mencionados na subseção 2.2.6, as RSSF são vulneráveis a diversos outros tipos de ataques em virtude não apenas de suas limitações de recursos mas também das características do meio sem fio e do fato de que, muitas vezes, os sensores inteligentes são dispostos em ambientes hostis sem proteção física, reforçando a importância do uso de mecanismos para neutralizar essas vulnerabilidades. Alguns exemplos dos principais ataques em RSSF disponíveis na literatura (SARAOGI, 2004; HU; SHARMA, 2005) são ilustrados na Figura 6 e descritos a seguir:

- comprometimento de nó sensor, no qual um adversário captura um sensor inteligente com o objetivo de extrair informações armazenadas, por exemplo, sobre a chave secreta, comprometendo o nó sensor.

- buracos negros (*black holes*), onde um nó sensor adversário introduz a melhor rota em vários destinos passando por ele, possibilitando ao nó sensor malicioso modificar ou até descartar pacotes da rede;

- *loops* ou desvios, no qual um adversário introduz *loops* ou desvios na rede para que os pacotes circulem sem chegar ao destinatário final, podendo gerar um excessivo consumo de energia nos sensores inteligentes da rede; e

- inundação *(flooding)*, onde um nó sensor malicioso pode injetar muitos pacotes na rede causando congestionamento e perda de mensagens, levando a um consumo exaustivo de energia pelos sensores inteligentes da rede.

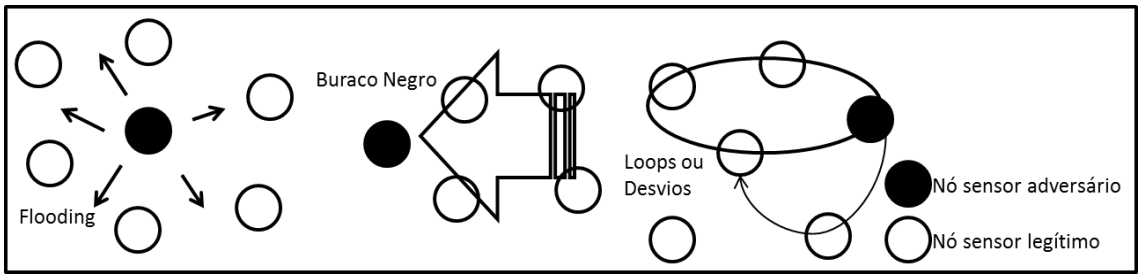

Figura 6. Ataques de segurança em RSSF. Fonte: adaptado de (MARGI et al., 2009).

3.5 Conclusão

Neste capítulo foi apresentado que a utilização do ar como meio de propagação do sinal e as severas limitações de recursos de armazenamento, processamento e comunicação tornam as RSSF vulneráveis a diversos tipos de ataques de segurança. Para combater essas vulnerabilidades, diversas aplicações de RSSF podem necessitar de determinados serviços de segurança como a confidencialidade dos dados, a integridade, a autenticação dos pacotes que trafegam na rede e a garantia de *freshness* das mensagens.

Além disso, foi visto que um ou mais mecanismos de segurança são responsáveis pelo fornecimento de serviços de segurança. Por exemplo, os mecanismos de criptografia fornecem o serviço de confidencialidade dos dados e os mecanismos de autenticação MACs proporcionam os serviços de integridade dos dados e a autenticação das mensagens da RSSF.

Além do mais, foi discutido como um adversário geralmente pode atacar mecanismos de segurança utilizados em uma RSSF por meio das técnicas de criptoanálise, ataque por força bruta ou falsificação de MACs e diversos outros ataques às aplicações de RSSF como a formação de buracos negros e o comprometimento dos sensores inteligentes da rede.

Portanto, as severas limitações apresentadas pelas RSSF, assim como as vulnerabilidades a diversos tipos possíveis de ataques de segurança apresentados neste capítulo demonstram a fundamental importância de pesquisas na área de segurança em RSSF, especialmente referente ao desempenho de seus mecanismos, o qual é o foco principal desta dissertação.

4 DESCRIÇÃO DOS MECANISMOS DE SEGURANÇA UTILIZADOS E DOS TRABALHOS RELACIONADOS

Neste capítulo são apresentadas as principais definições e características dos mecanismos de segurança avaliados nesta dissertação e são discutidos os principais trabalhos relacionados com esta dissertação encontrados na literatura. A seção 4.1 apresenta algumas notações importantes utilizadas para descrever os mecanismos de segurança. Em seguida, as seções 4.2, 4.3 e 4.4 definem e caracterizam, respectivamente, os mecanismos de criptografia, os modos de operação e os algoritmos de autenticação utilizados neste trabalho. A seção 4.5 caracteriza os principais protocolos de segurança para RSSF e a seção 4.6 discute os principais trabalhos relacionados encontrados na literatura. Por fim, a seção 4.7 conclui o capítulo.

4.1 Introdução

Ao longo deste capítulo algumas notações específicas são utilizadas e, para facilitar a compreensão das mesmas e evitar a repetição dos significados, elas são apresentadas a seguir.

- *w* – uma determinada quantidade de *bits*;
- *word* – uma sequência de *w bits*;
- $\boxplus$,$\boxminus$ – adição e subtração de duas *words* módulo 2^w;
- $\hookleftarrow$,$\hookrightarrow$ – rotação de *bit* à esquerda e à direita, respectivamente;
- $\oplus$ – operação de *bit ou exclusivo* (XOR);
- | – concatenação;
- O termo $E_k(M)$ e o termo $D_k(M)$ representam uma operação de encriptação e de decriptação de um bloco de texto claro M, utilizando uma chave *k* e uma cifra de bloco, respectivamente.

Outras notações não tão abrangentes são oportunamente explicadas no decorrer de cada seção deste capítulo.

4.2 Cifras de Bloco

Nesta seção, são descritas e caracterizadas as cifras de bloco consideradas neste trabalho, bem como são discutidos aspectos sobre o nível de segurança dessas cifras.

4.2.1 AES

Em 1997, o *National Institute of Standards and Technology* (NIST) realizou um concurso mundial para escolher um novo padrão de criptografia, sendo que a cifra de bloco Rijndael foi a vencedora tornando-se o Advanced Encryption Standard (AES) (NIST, 2001). O AES é uma cifra de bloco de domínio público e amplamente utilizada por sistemas de segurança, possuindo os seguintes parâmetros: tamanho do bloco de 128 bits; tamanho da chave de 128, 192 ou 256 bits; e número de rounds igual a 10, 12 ou 14.

O AES manipula *words de 32 bits*, sendo que em cada *round* ele opera um resultado intermediário chamado matriz *state*, o qual é um bloco de *words*. O funcionamento do AES é descrito nos Algoritmos 1 e 2, baseado nas definições de (NIST, 2001; DELFS; KNEBL, 2007).

No Algoritmo 1, após inicializar a matriz *state* e somá-la (XOR) com uma subchave gerada pelo processo de expansão da chave que será descrito mais adiante nesta subseção, o AES realiza as operações *SubBytes*, *ShiftRows* e *MixColumns* em cada *round*.

Algoritmo 1 AES – Encriptação	**Algoritmo 2** AES – Decriptação
***Entrada**: um bloco de texto claro de 128 bits M, o número de rounds r e as subchaves $k_0...k_{r-1}$.* ***Saída**: o bloco de texto criptografado C de 128 bits.*	***Entrada**: um bloco de texto cifrado de 128 bits C, o número de rounds r e as subchaves $k_0...k_{r-1}$.* ***Saída**: o bloco de texto claro M de 128 bits.*
1. *Inicialize a matriz state* *state ← M;* *state ← state⊕k_0;*	1. *Inicialize a matriz state* *state ← C;* *state ← state⊕k_r;*
2. *Processe da seguinte forma:* ***para** i ← 1 **até** r-1 **faça*** *SubBytes(state);* *ShiftRows(state);* *MixColumns(state);* *state ← state⊕k_i;* *SubBytes(state);* *ShiftRows(state);* *state ← state⊕k_r;*	2. *Processe da seguinte forma:* ***para** i ← r-1 **até** 1 **faça*** *InvShiftRows(state);* *InvSubbytes(state);* *state ← state⊕k_i;* *InvMixColumns(state);* *InvShiftRows(state);* *InvSubBytes(state);* *state ← state⊕k_0;*
3. *Devolva C ← state.*	3. *Devolva M ← state.*

A operação *SubBytes* consiste em realizar substituições de acordo com uma tabela chamada S-box (*Substitution box*), um componente muito utilizado em cifras de bloco. A tabela S-box do AES pode ser obtida em (NIST, 2001). Por outro lado, a *ShiftRows* realiza operações de deslocamento na matriz *state*. E, por último, a função *MixColumns* realiza permutações em *state*.

As operações do AES são baseadas no conceito matemático de corpos finitos *Galois Fields* (GF 2^8) (NIST, 2001). Assim, cada *byte* nessa cifra é tratado como um elemento de um campo finito representado de forma polinomial, permitindo a descrição de operações matemáticas como adição e multiplicação na forma de operações polinomiais (NIST, 2001). Na função *MixColumns*, por exemplo, é realizada uma combinação linear utilizando a aritmética GF 2^8.

Todas as funções apresentadas no Algoritmo 1 são invertíveis e, portanto, o processo de decriptação (Algoritmo 2) pode ser realizado a partir das funções inversas correspondentes *invSubBytes*, *invShiftRows* e *InvMixColoumns* e da utilização das subchaves na ordem inversa à utilizada na encriptação. Apesar disso, o algoritmo de decriptação AES não é igual ao de encriptação, consequência da estrutura específica do AES (STALLINGS, 2010).

O Algoritmo 3 apresenta o processo de expansão da chave AES com uma chave principal de tamanho quatro *words* (AES-128) que produz um vetor linear de 44 *words*, sendo suficiente para oferecer uma subchave para a inicialização e para cada *round* da cifra. A operação *SubWord* realiza uma substituição em cada byte de sua *word* de entrada usando a S-box do AES e um vetor Rcon[i], uma constante *word* diferente para cada *round* definida como (RC[i], 0,0,0), em que o cálculo de RC é baseado em operações sobre o corpo finito GF 2^8.

Algoritmo 3 AES – Expansão da chave

***Entrada**: um bloco de texto claro de 128 bits M e uma chave de 128 bits K[0], .., K[15].*
***Saída**: as subchaves k_0,..., k_{44} de 16 bits cada.*

1. *Processe as 4 primeiras subchaves*
   para** i ← 0 **até** 3 **faça
   k_i = (K[4i] | K[4i+1] | K[4i+2] | K[4i+3]);
2. *Processe as demais subchaves*
   para** i ← 4 **até** 44 **faça
   temp ←k_{i-1}; //tamanho de temp é 16 bits
   ***se** (i mod 4 = 0) **então** temp ← SubWord(temp ↩1) ⊕ Rcon[i/4];*
   k_i = k_{i-4} ⊕ temp;
3. *Devolva k_0,..., k_{44}.*

4.2.2 Skipjack

O Skipjack é uma cifra de bloco desenvolvida na década de 80 e tornada pública em 1998 (NIST, 1998), sendo concebida especialmente para dispositivos com capacidade de processamento limitada. Os parâmetros dessa cifra são caracterizados da seguinte forma: tamanho do bloco de 64 *bits*, tamanho da chave de 80 *bits* e 32 *rounds*. Cada *round* consiste na aplicação das regras dispostas nos algoritmos 4, 5, 6 e 7, em que *x+1* corresponde ao número do *round* atual e G a uma função de permutação conhecida como estrutura de Feistel.

A função de permutação G é composta por quatro *rounds*, sendo que em cada *round* é utilizada uma subchave de oito *bits* derivada da chave principal e uma função F que consiste em uma tabela S-box. O processo de expansão da chave do Skipjack consiste em formar as subchaves utilizadas na função G e cada subchave consiste no [(4k+i-3) mod 10]-ésimo *byte* da chave principal. A tabela F está disposta em (NIST, 1998) e G é definida no algoritmo 8.

Algoritmo 4 Skipjack – Regra_A

***Entrada**: uma sequência de 16 bits cada w_1, w_2, w_3 e w_4, um índice x e a chave K.*
***Saída**: w_1, w_2, w_3 e w_4 e c processados.*

1. $\boldsymbol{w_1^{x+1} \leftarrow G^x(w_1^x, x) \oplus w_4^x \oplus (x+1)}$;
2. $\boldsymbol{w_2^{x+1} \leftarrow G^x(w_1^x, x, K)}$;
3. $\boldsymbol{w_3^{x+1} \leftarrow w_2^x}$;
4. $\boldsymbol{w_4^{x+1} \leftarrow w_3^x}$.

Algoritmo 5 Skipjack – Regra_B

***Entrada**: uma sequência de 16 bits cada w_1, w_2, w_3 e w_4, um índice x e a chave K.*
***Saída**: w_1, w_2, w_3 e w_4 e c processados.*

1. $w_1^{x+1} \leftarrow w_4^x$;
2. $w_2^{x+1} \leftarrow G^x(w_1^x, x)$;
3. $w_3^{x+1} \leftarrow w_1^x \oplus w_2^x \oplus (x+1)$;
4. $w_4^{x+1} \leftarrow w_3^x$.

Algoritmo 6 Skipjack – Regra_A^{-1}

***Entrada**: uma sequência de 16 bits cada w_1, w_2, w_3 e w_4, e um índice x.*
***Saída**: w_1, w_2, w_3 e w_4 e c processados.*

1. $\boldsymbol{w_1^{x-1} \leftarrow [G^{x-1}]^{-1}(w_2^x, x)}$;
2. $\boldsymbol{w_2^{x-1} \leftarrow w_3^x}$;
3. $\boldsymbol{w_3^{x-1} \leftarrow w_4^x}$;
4. $\boldsymbol{w_4^{x-1} \leftarrow w_1^x \oplus w_2^x \oplus (x-1)}$.

Algoritmo 7 Skipjack – Regra_B^{-1}

***Entrada**: uma sequência de 16 bits cada w_1, w_2, w_3 e w_4, e um índice x.*
***Saída**: w_1, w_2, w_3 e w_4 e c processados.*

1. $w_1^{x-1} \leftarrow [G^{x-1}]^{-1}(w_2^x, x)$;
2. $w_2^{x-1} \leftarrow [G^{x-1}]^{-1}(w_2^x, x) \oplus w_3^x \oplus (x-1)$;
3. $w_3^{x-1} \leftarrow w_4^x$;
4. $w_4^{x-1} \leftarrow w_1^x$.

Algoritmo 8 Skipjack – Função G

***Entrada**: uma sequência de 16 bits w = g1|g2, uma chave K_1,..., K_{10} e um índice x.*
***Saída**: uma sequência de 16 bits processada.*

1. ***para** i=3 **até** 6 **faça***
2. $\boldsymbol{g_i \leftarrow F(g_{i-1} \oplus K_{(4x+i-3)\ mod\ 10}) \oplus g_{i-1}}$;
3. *Devolva g5|g6.*

Portanto, considerando os algoritmos anteriores e baseando-se nas definições de (NIST, 1998; MATEUS, 2009), a encriptação e a decriptação com o Skipjack são descritos conforme os algoritmos 9 e 10.

Algoritmo 9 Skipjack – Encriptação	**Algoritmo 10** Skipjack – Decriptação
***Entrada**: um bloco de texto claro de 64 bits M = (w_1|w_2|w_3|w_4) e a chave K de 10 bytes.* ***Saída**: o bloco de texto criptografado C de 64 bits.*	***Entrada**: um bloco de texto cifrado de 64 bits C = (w_1|w_2|w_3|w_4) e a chave K de 10 bytes.* ***Saída**: o bloco de texto M de 64 bits.*
1. *Aplique as Regras A e B da seguinte forma:* ***para** i=0 **até** 7 **faça*** *Regra_A(w_1,w_2,w_3,w_4,i,K);* ***para** i=8 **até** 15 **faça*** *Regra_B(w_1,w_2,w_3,w_4,i,K);* ***para** i=16 **até** 23 **faça*** *Regra_A(w_1,w_2,w_3,w_4,i,K);* ***para** i=24 **até** 31 **faça*** *Regra_B(w_1,w_2,w_3,w_4,i,K);* 2. *Devolva C ← w_1|w_2|w_3|w_4.*	1. *Aplique as Regras A^{-1} e B^{-1} da seguinte forma:* ***para** i=31 **até** 24 **faça*** *Regra_B^{-1}(w_1,w_2,w_3,w_4,i,K);* ***para** i=15 **até** 8 **faça*** *Regra_A^{-1}(w_1,w_2,w_3,w_4,i,K);* ***para** i=23 **até** 16 **faça*** *Regra_B^{-1}(w_1,w_2,w_3,w_4,i,K);* ***para** i=15 **até** 0 **faça*** *Regra_A^{-1}(w_1,w_2,w_3,w_4,i,K);* 2. *Devolva M ← w_1|w_2|w_3|w_4.*

4.2.3 RC5

O RC5 é uma cifra de bloco patenteada e concebida para, entre outros objetivos, requerer pouca quantidade de memória com alto nível de segurança (RIVEST, 1994), sendo esse nível definido pela escolha de seus parâmetros.

Algoritmo 11 RC5 – Encriptação	**Algoritmo 12** RC5 – Decriptação
***Entrada**: um bloco de texto claro de 2w bits M = (A|B),em que A e B são duas words de w bits; o número de rounds r e a chave de b bytes K[0]...K[b-1].* ***Saída**: o bloco de texto criptografado C de 2w bits.*	***Entrada**: um bloco de texto cifrado de 2w bits C = (A|B),em que A e B são duas words de w bits; o número de rounds r e a chave de b bytes K[0]...K[b-1].* ***Saída**: o bloco de texto claro M de 2w bits.*
1. *Processe da seguinte forma:* *$A \leftarrow A \boxplus K_0$, $B \leftarrow B \boxplus K_1$;* ***para** i=1 **até** r **faça*** *$A \leftarrow ((A \oplus B) \lll B) \boxplus K_{2i}$;* *$B \leftarrow ((B \oplus A) \lll A) \boxplus K_{2i+1}$;* 2. *Devolva C ← (A,B).*	1. *Processe da seguinte forma:* *para i=r até 1 faça* *$B \leftarrow ((B \boxminus K_{2i+1}) \lll A) \boxplus A$;* *$A \leftarrow ((A \boxminus K_{2i}) \lll B) \boxplus B$;* *$B \leftarrow B \boxminus K_1$, $A \leftarrow A \boxminus K_0$;* 2. *Devolva M ← (A,B).*

As operações básicas do RC5 consistem em manipulação de *words* de *w bits* (16, 32 ou 64), sendo que o tamanho do bloco de texto claro e do bloco de texto cifrado possui tamanho 2*w*. Além disso, o RC5 usa uma tabela que armazena as subchaves derivadas da chave secreta principal. O tamanho dessa tabela é definido por *2(numeroDeRounds+1) words*, significando que

a escolha de um alto valor do número de *rounds* implica no aumento do nível de segurança ao custo do aumento de utilização da memória.

Algoritmo 13 RC5 – Expansão da chave

***Entrada**: o tamanho de uma word w; o número de rounds r e a chave K[0]...K[b-1].*
***Saída**: as subchaves K_0, ..., K_{2r+1}, em que K_i possui w bits.*

1. *Copie a chave secreta K[0], ... K[b-1] em um vetor L[0], ..., L[c-1]de c = b/u words, em que u = w/8 é o número de bytes por word. Ao final, qualquer byte não preenchido de L é zerado;*
   ***para** i←b-1 **até** 0 **faça** L[i/u] ← (L[i/u] ↪ 8) + K_i;*

2. *Seja P_w ← Odd((e – 2)2^w e Q_w ← Odd((**ϕ** – 1)2^w) constantes mágicas[6]. Nesta segunda fase, as subchaves são inicializadas com um padrão de bit pseudoaleatório independente da chave, utilizando progressão aritmética módulo 2w determinada por P_w e Q_w. OBS. Odd(X)o número inteiro ímpar mais próximo de X;*
   K_0 ← P_w;
   ***para** i←r **até** 2r + 1 **faça** K_i ← K_{i-1} ⊞ Q_w; i ← 0, j ← 0; A ← 0, B ← 0;*
   para** h←1 **até** 3*max(2(r+1),c) **faça
   K_i ← (K_i ⊞ A ⊞ B) ↩ 3, A ← K_i, i + 1 mod 2(r+1);
   Lj ← (Kj ⊞ A ⊞ B) ↩ (A ⊞ B), B ← Lj, j ← j + 1 mod c;

3. *Devolva K_0, ..., K_{2r+1}.*

Os parâmetros do RC5 podem ser caracterizados da seguinte forma: tamanho do bloco de 32, 64 ou 128 *bits*; tamanho da chave variável de 0 a 255 *bytes* e 0 a 255 *rounds*. Assim, baseado nas definições disponíveis em (RIVEST, 1994; MENEZES *et al.*, 1996), o RC5 pode ser descrito conforme os algoritmos 11, 12 e 13.

4.2.4 Klein

O Klein é uma cifra de bloco de domínio público projetada para dispositivos com capacidade de processamento limitado, como é o caso das RSSF. Essa cifra possui como principais características: tamanho do bloco de 64 *bits*, tamanho da chave de 64, 80 ou 96 *bits* e 12, 16 ou 20 *rounds* de acordo com a chave, respectivamente (GONG *et al.*, 2011). Em cada *round*, essa cifra de bloco realiza operações de substituição e permutação de forma semelhante a outras cifras como o AES. O funcionamento do Klein é descrito nos algoritmos 14 e 15, baseado nas definições de (GONG *et al.*, 2011).

[6] *e é o número de Euler (2,71828182...) e **ϕ** é a razão áurea (1,61803399...).*

Algoritmo 14 Klein – Encriptação	**Algoritmo 15** Klein – Decriptação
***Entrada**: um bloco de texto claro de 64 bits M, o número de rounds r, o tamanho t da chave e as subchaves $k_0 ... k_{r-1}$.* ***Saída**: o bloco de texto criptografado C de 64 bits.*	***Entrada**: um bloco de texto cifrado de 64 bits C, o número de rounds r, o tamanho t da chave e as subchaves $k_0 ... k_{r-1}$.* ***Saída**: o bloco de texto claro M de 64 bits.*
1. *Inicialize o vetor state* *state ← C;*	1. *Inicialize o vetor state* *state ←M;*
2. *Processe da seguinte forma:* ***para** i ← 0 **até** r-1 **faça*** *state ← state$\bigoplus k_i$;* *SubNibbles(state);* *RotateNibbles(state);* *MixNibbles(state);*	2. *Processe da seguinte forma:* ***para** i ← r-1 **até** 0 **faça*** *state ← state$\bigoplus k_i$;* *MixNibbles(state);* *RotateNibbles(state);* *SubNibbles(state);*
3. *Devolva C ← state$\bigoplus k_{r+1}$.*	3. *Devolva C ←state$\bigoplus k_0$.*

A operação de encriptação do Klein, mostrada no Algoritmo 14, inicia com a cópia do bloco de texto claro para um vetor de *bytes* chamado *state*. Em cada *round*, uma subchave é adicionada ao vetor *state* por meio de uma operação XOR e as funções *SubNibbles*, *RotateNibbles* e *MixNibbles* são executadas. Por fim, o bloco de texto cifrado é formado a partir da operação XOR entre o vetor *state* e uma subchave, sendo que as subchaves são geradas pelo processo de expansão da chave descrito no Algoritmo 16.

A função *SubNibble* consiste na realização de substituições de cada *nibble* (sequência de quatro *bits*) do vetor *state* por um valor correspondente em uma tabela S-box do Klein, a qual pode ser encontrada em (GONG *et al.*, 2011). Por outro lado, a função *RotateNibbles* realiza uma operação de deslocamento de 16 *bits* no vetor *state*. Por fim, a função *MixNibbles* processa o vetor *state* da mesma maneira que a função *MixColumns* do AES.

Também da mesma forma que o AES, todas as funções apresentadas no Algoritmo 14 são invertíveis e, portanto, o processo de decriptação pode ser realizado a partir das funções inversas correspondentes *InvSubNibbles*, *InvRotateNibbles* e *InvMixNibbles*, conforme mostrado no Algoritmo 15 e da utilização das subchaves na ordem inversa à ordem utilizada na encriptação.

O processo de expansão da chave do Klein, descrito no Algoritmo 16, gera as subchaves a serem utilizadas nos algoritmos 14 e 15. Conforme pode ser observado no Algoritmo 16, há um processo recursivo no passo 2, bem como a utilização da tabela S-box para realizar substituições de forma não linear para a obtenção das subchaves.

Algoritmo 16 Klein – Expansão da chave

***Entrada**: uma chave de t bytes K[0], .., K[t-1].*
***Saída**: as subchaves k_0,..., k_r.*

1. *Inicialize k_0*
   $k_0 \leftarrow K$;

2. *Processe da seguinte forma para obter as demais subchaves:*
   para** i ← 1 **até** r **faça
   para** j ← 0 **até** (t/2) -1 **faça
   $k_i[j] \leftarrow k_{i-1}[((j+1) \bmod t/2) + t/2]$;
   $k_i[j+(t/2)] = ki\text{-}1[((j+1) \bmod t/2] \oplus k_{i-1}[(j+1) \bmod t/2]$;
   $k_i[2] = S[k_i[5]] \oplus \mathbf{i}$;
   $k_i[5] = S[k_i[5]]$;
   $k_i[6] = S[k_i[6]]$;

3. *Devolva k_0,..., k_r.*

4.2.5 Present

O Present é uma cifra de bloco de domínio público projetada como alternativa ao AES para dispositivos com capacidade de processamento limitada como as RSSF (BOGDANOV *et al.*, 2007). Essa cifra possui como parâmetros um tamanho de bloco de 64 *bits*, tamanho da chave de 80 ou 128 *bits* e 31 *rounds*. Além disso, a sua operação de encriptação é descrita no Algoritmo 17, baseado em (BOGDANOV *et al.*, 2007).

Como mostrado no Algoritmo 17, da mesma forma que a cifra Klein, o Present opera sobre um vetor de *bytes* (*state*) que inicialmente recebe o bloco de texto claro e, ao final, recebe o bloco de texto cifrado. Em cada *round*, é realizada uma operação XOR entre o vetor *state* e uma subchave gerada pelo Algoritmo 18, seguido da aplicação das duas funções *sBoxLayer* e *pLayer*. A função sBoxLayer consiste em realizar substituições não lineares dos elementos de *state* por elementos correspondente da tabela S-box do Present, a qual pode ser encontrada em (BOGDANOV *et al.*, 2007). Por outro lado, a operação *pLayer* consiste em permutar a posição de cada elemento do vetor *state*.

O processo de expansão da chave realiza operações de permutação e substituição utilizando a tabela S-box do Present para formar as subchaves a serem utilizadas em cada *round* da encriptação, conforme mostrado no Algoritmo 18. Vale ressaltar que os autores da cifra recomendam o uso somente do processo de encriptação (*only-encryption*) do Present para obter uma melhor eficiência da cifra. Sendo assim, para realizar a decriptação, a cifra deve ser utilizada

juntamente com um modo de operação que não necessite do algoritmo de decriptação, conforme será visto na seção 4.3.

Algoritmo 17 Present – Encriptação

***Entrada**: um bloco de texto claro de 64 bits M e as subchaves chave $x_0...x_{31}$.*
***Saída**: o bloco de texto criptografado C de 64 bits.*

1. *Inicialize state*
   state ←C;
2. *Processe da seguinte forma:*
   para** i ← 0 **até** 30 **faça
   state ← state$\bigoplus x_i$;
   sBoxLayer(state);
   pLayer(state);
   state ← state$\bigoplus x_{32}$;
3. *Devolva C ← state.*

Algoritmo 18 Present – Expansão da chave

***Entrada**: uma chave $K = k_{t-1}...k_0$, em que t é 80 ou 128 bits.*
***Saída**: as subchaves $x_0, ..., x_{31}$.*

1. *Processe da seguinte forma:*
   para** i ← 1 **até** 32 **faça
   $K \leftarrow K \lhd 61$;
   $[k_{t-1}k_{t-2}k_{t-3}k_{t-4}] \leftarrow S[k_{t-1}k_{t-2}k_{t-3}k_{t-4}]$;
   $[k_{t-5}k_{t-6}k_{t-7}k_{t-8}] \leftarrow S[k_{t-5}k_{t-6}k_{t-7}k_{t-8}]$;
   $[k_{t-61}k_{t-62}k_{t-63}k_{t-64}] \leftarrow [k_{t-61}k_{t-62}k_{t-63}k_{t-64}] \bigoplus i$;
   $x_{i-1} \leftarrow k_{t-1}...k_{t-64}$;
2. *Devolva $x_0,...,x_{31}$.*

4.2.6 Nível de Segurança

Com relação ao nível de segurança, iniciando pelo AES, conforme publicado em (CRYPTREC, 2011), o qual apresentou uma criptoanálise com complexidade ligeiramente menor quando comparado com o ataque de força bruta. Entretanto, essa criptoanálise não é considerada realista, uma vez que exige uma grande quantidade de dados, o que demonstra o alto nível de segurança da cifra.

Já em relação ao Skipjack, em 2002 foi apresentada a primeira criptoanálise completa dos 32 *rounds* da cifra (PHAN, 2002), revelando a vulnerabilidade do algoritmo Skipjack, sendo a limitação da chave um dos fatores principais responsáveis pelo seu baixo nível de segurança.

Sobre o RC5, Rivest (1994) definiu a configuração RC5 64/12/16 (tamanho do bloco de 64 *bits*, chave de 16 *bytes* e 12 *rounds*) como padrão na publicação da cifra. Entretanto, para ser considerado seguro contra ataques de segurança, o número de *rounds* deve ser maior do que 16 (POTLAPALLY *et al.*, 2005).

Por outro lado, com relação ao Klein, Aumasson *et al.* (2011) apresentaram uma criptoanálise contra oito *rounds* do Klein-64. Por ser uma cifra relativamente nova não foram encontradas publicações sobre a criptoanálise dessa cifra com tamanhos de chave de 80 e 96 *bits*, entretanto, seus criadores apresentaram um estudo mostrando a robustez do Klein contra diversos

tipos de ataques (GONG *et al.*, 2011), revelando seu nível de segurança elevado para aplicações como RSSF.

Com relação ao Present, Borghoff *et al.* (2011) apresentaram uma criptoanálise contra até 28 *rounds* do Present. Assim, como o número de *rounds* da cifra é 32, o Present pode fornecer um nível alto de segurança para grande parte das aplicações de RSSF.

4.3 Modos de Operação

Para descrever os modos de operação serão utilizadas as seguintes notações:

- $M_1, M_2, \ldots, M_n$ é uma sequência de blocos de texto claro;
- $C_1, C_2, \ldots, C_n$ é uma sequência de blocos de texto cifrado;
- Os termos $LSB_s(X)$ e $MSB_s(X)$ representam os *s bits* menos significativos de X e os *s bits* mais significativos de X, respectivamente; e
- *I* denota a entrada do algoritmo de encriptação e *O* a saída desse algoritmo.

4.3.1 CBC

O modo CBC (DWORKING, 2001) realiza uma operação de XOR entre o bloco a ser cifrado e o bloco cifrado anteriormente. Assim, cada bloco de texto cifrado depende dos demais blocos cifrados anteriormente. O primeiro bloco a ser criptografado deve ser somado (XOR) com um IV para que blocos de texto claro idênticos gerem blocos cifrados diferentes, sendo que esse IV deve ser gerado preferencialmente de forma aleatória.

Algoritmo 19. CBC

***Entrada**: um conjunto de n blocos de texto claro M e um vetor de inicialização IV.*
***Saída**: um conjunto de n blocos de texto cifrado/claro C/M na encriptação/decriptação.*

1. *Encriptação* $C_1 \leftarrow E_k(M_1 \oplus IV)$; ***para*** $i \leftarrow 2$ ***até*** n ***faça*** $C_i \leftarrow E_k(M_i \oplus C_{i-1})$; 2. *Devolva* $C_1,\ldots,C_n$.	1. *Decriptação* $M_1 \leftarrow D_k(C_1) \oplus IV$ ***para*** $i \leftarrow 2$ ***até*** n ***faça*** $M_i \leftarrow D_k(C_i) \oplus C_{i-1}$; 2. *Devolva* $M_1,\ldots,M_n$.

O bloco criptografado no modo CBC possui tamanho múltiplo do tamanho do bloco padrão da cifra. Assim, caso o último bloco seja menor do que o tamanho padrão da cifra, ele

será preenchido. As operações de encriptação e decriptação de *n* blocos no modo CBC são denotadas de acordo com as equações dispostas no Algoritmo 19, baseado em (DWORKING, 2001).

4.3.2 CFB

O CFB (DWORKING, 2001) transforma uma cifra de bloco em um gerador de números pseudoaleatórios. O texto cifrado realimenta a cifra de bloco, sendo esse processo repetido para produzir um fluxo de *bits* pseudoaleatórios, de forma semelhante a uma cifra de fluxo. Além disso, o modo CFB requer um parâmetro *s*, tal que *s* indica o número de *bits* da saída do modo. Frequentemente, esse parâmetro é incorporado ao nome do modo, por exemplo, CFB-1, CFB-8, CFB-64.

Vale salientar que as operações de encriptação e decriptação utilizam apenas o algoritmo de encriptação da cifra de bloco, conforme mostrado no Algoritmo 20. Os termos $M^{\#}$ e $C^{\#}$ referem-se a segmentos de *s bits* de *M* e *C*, respectivamente.

Algoritmo 20. CFB

***Entrada**: um conjunto de n blocos de texto claro M e um vetor de inicialização IV.*
***Saída**: um conjunto de n blocos de texto cifrado/claro C/M na encriptação/decriptação.*

1. *Encriptação*
   $I_1 \leftarrow IV$;
   para $i \leftarrow 1$ ***até*** $n-1$ ***faça***
      $I_{i+1} \leftarrow LSB_{b-s}(I_i) \mid C^{\#}_i$;
      $O_i \leftarrow E_k(I_i)$; $C^{\#}_i \leftarrow M^{\#}_i \oplus MSB_s(O_i)$;
   $C^{\#}_n \leftarrow M^{\#}_n \oplus MSB_s(O_n)$;
2. *Devolva* $C^{\#}_1,...,C^{\#}_n$.

1. *Decriptação*
   $I_1 \leftarrow IV$;
   para $i \leftarrow 1$ ***até*** $n-1$ ***faça***
      $I_{i+1} \leftarrow LSB_{b-s}(I_i) \mid C^{\#}_i$;
      $O_i \leftarrow E_k(I_i)$;$M^{\#}_i \leftarrow C^{\#}_i \oplus MSB_s(O_i)$;
   $M^{\#}_n \leftarrow C^{\#}_n \oplus MSB_s(O_n)$;
2. *Devolva* $M^{\#}_1,...,M^{\#}_n$.

4.3.3 OFB

O modo OFB (DWORKING, 2001) é muito semelhante ao modo CFB. A principal diferença está na realimentação da saída da função de criptografia em vez do texto cifrado (saída da encriptação somado – XOR – com o texto claro). As operações do modo OFB são mostradas no Algoritmo 21, sendo que o termo *u* refere-se ao número de *bits* do último bloco, o qual pode ser menor que o tamanho padrão da cifra de bloco.

Algoritmo 21. OFB

***Entrada**: um conjunto de n blocos de texto claro M e um vetor de inicialização IV.*
***Saída**: um conjunto de n blocos de texto cifrado/claro C/M na encriptação/decriptação.*

1. *Encriptação*
   $I_1 \leftarrow IV$;
   para $i \leftarrow 1$ ***até*** $n-1$ ***faça***
      $O_i \leftarrow E_k(I_i)$; $I_{i+1} \leftarrow O_i$; $C_i \leftarrow M_i \oplus O_i$;
   $O_n \leftarrow E_k(I_n)$;
   $C^{\#}_n \leftarrow M^{\#}_n \oplus MSB_u(O_n)$;
2. *Devolva* $C^{\#}_1,...,C^{\#}_n$.

1. *Decriptação*
   $I_1 \leftarrow IV$;
   para $i \leftarrow 1$ ***até*** $n-1$ ***faça***
      $O_i \leftarrow E_k(I_i)$; $I_{i+1} \leftarrow O_i$; $M_i \leftarrow C_i \oplus O_i$;
   $O_{\boldsymbol{n}} \leftarrow E_k(I_{\boldsymbol{n}})$;
   $M^{\#}_n \leftarrow C^{\#}_n \oplus MSB_u(O_n)$;
2. *Devolva* $M^{\#}_1,...,M^{\#}_n$.

4.3.4 CTR

O modo CTR (DWORKING, 2001) utiliza um contador o qual deve ser utilizado como entrada na encriptação e decriptação de cada bloco. Esse contador deve possuir um valor diferente a cada mensagem criptografada. Além do contador, geralmente é utilizado um número chamado de *nonce*, o qual é similar ao IV nos demais modos. O valor *nonce* em geral é concatenado com o contador para produzir um bloco único. As operações do modo CTR são mostradas no Algoritmo 22.

Algoritmo 22. CTR

***Entrada**: um conjunto de n blocos de texto claro M e um valor nonce.*
***Saída**: um conjunto de n blocos de texto cifrado/claro C/M na encriptação/decriptação.*

1. *Encriptação*
   para $i \leftarrow 1$ ***até*** $n-1$ ***faça***
      $O_i \leftarrow E_k(T_i \mid nonce)$; $C_i \leftarrow M_i \oplus O_i$;
      $O_n \leftarrow E_k(T_n \mid nonce)$;
   $C^{\#}_n \leftarrow M^{\#}_n \oplus MSB_u(O_n)$;
2. *Devolva* $C^{\#}_1,...,C^{\#}_n$.

1. *Decriptação*
   para $i \leftarrow 1$ ***até*** $n-1$ ***faça***
      $O_i \leftarrow E_k(T_i \mid nonce)$; $M_i \leftarrow C_i \oplus O_i$;
      $O_i \leftarrow E_k(T_i \mid nonce)$;
   $\boldsymbol{M}^{\#}_n \leftarrow C^{\#}_n \oplus MSB_u(O_n)$**;**
2. *Devolva* $M^{\#}_1,...,M^{\#}_n$.

4.3.5 OCB

O modo OCB, além de fazer o papel de um modo de operação na encriptação dos dados, fornece o serviço de autenticação. O seu funcionamento é descrito nos algoritmos 23 e 24, baseado em (ROGWAY *et al.*, 2003). Geralmente um contador é utilizado como *nonce*, não sendo necessário que ele seja secreto ou imprevisível. A operação do OCB resulta em um conjunto de blocos de texto cifrado e uma *tag* de autenticação (ROGWAY *et al.*, 2003).

Algoritmo 23. OCB – Encriptação	**Algoritmo 24.** OCB – Decriptação
***Entrada**: um conjunto de n blocos de texto claro M e um valor nonce N.* ***Saída**: um conjunto de n blocos de texto cifrado C e um código (tag).*	***Entrada**: um conjunto de n blocos de texto cifrado C, um valor nonce N e um código (tag).* ***Saída**: Se tag válida, devolve um conjunto de n blocos de texto claro C e um código (tag). Caso contrário, devolve OK.*
1. *Processe da seguinte forma:* $L \leftarrow E_k(0^n)$; $R \leftarrow E_k(N \oplus L)$; ***para*** $i \leftarrow 1$ ***até*** m ***faça*** $Z_i \leftarrow \boldsymbol{\gamma}_i \cdot L \oplus R$; ***para*** $i \leftarrow 1$ ***até*** $m-1$ ***faça*** $C_i \leftarrow E_k(M_i \oplus Z_i) \oplus Z_i$; $X_m \leftarrow len(M_m) \oplus L \cdot x^{-1} \oplus Z_m$; $Y_m \leftarrow E_k(X_m)$; $C_m \leftarrow Y_m \oplus M_m$; $C \leftarrow C_1,...C_m$; $ck \leftarrow M_1 \oplus \ldots \oplus M_{m-1} \oplus C_m\ 0^* \oplus Y_m$; $tag \leftarrow E_k(ck \oplus Z_m)$ *primeiros* $\boldsymbol{\tau}$ *bits* 2. *Devolva* $C \leftarrow C \mid tag$; *Obs.* $\boldsymbol{\gamma}_i$ *é o i-ésimo número em notação de código de Gray.*	1. *Processe da seguinte forma:* $L \leftarrow E_k(0^n)$; $R \leftarrow E_k(N \oplus L)$; ***para*** $i \leftarrow 1$ ***até*** m ***faça*** $Z_i \leftarrow \gamma_i \cdot L \oplus R$; ***para*** $i \leftarrow 1$ ***até*** $m-1$ ***faça*** $M_i \leftarrow D_k(C_i \oplus Z_i) \oplus Z_i$; $X_m \leftarrow len(C_m) \oplus L \cdot x^{-1} \oplus Z_m$; $Y_m \leftarrow E_k(X_m)$; $M_m \leftarrow Y_m \oplus C_m$; $M \leftarrow C_1,...C_m$; $ck \leftarrow M_1 \oplus \ldots \oplus M_{m-1} \oplus C_m\ 0^* \oplus Y_m$; $tag` \leftarrow E_k(ck \oplus Z_m)$ *primeiros* $\tau$ *bits* 2. ***se*** $tag \neq tag`$ ***então*** *devolva FALHOU;* ***senão*** *Devolva M;*

4.3.6 Nível de Segurança

Com relação ao nível de segurança dos modos de operação, é importante discorrer sobre algumas características desses mecanismos. Os modos CFB, OFB, CTR e OCB funcionam de forma semelhante a uma cifra de fluxo. Assim, o tamanho da mensagem criptografada será o mesmo da mensagem não criptografada. Por outro lado, o modo CBC pode produzir textos cifrados com tamanho superior ao texto claro correspondente, uma vez que o tamanho da mensagem criptografada será múltiplo do tamanho da mensagem original. No contexto de RSSF, isso pode resultar em envio de *bytes* desnecessários pela rede e consequente perda de energia, mas uma técnica conhecida por *ciphertext stealing* (NIST, 2007) pode ser utilizada para evitar esse problema.

É bom lembrar ainda que os modos CFB, OFB e CTR utilizam apenas o algoritmo de encriptação tanto na encriptação como também na decriptação dos blocos, ao contrário dos demais modos apresentados. Essa característica traz benefícios com relação à quantidade de memória requerida, uma vez que é necessária uma quantidade relativamente menor de código.

Vale salientar que a geração dos IVs é a tarefa principal para garantir um nível alto de segurança na implantação de mecanismos de criptografia, uma vez que a utilização indevida de IVs pode comprometer a segurança do sistema. Os modos CBC e CFB exigem a

imprevisibilidade dos IVs, mas permitem a sua repetição, ao contrário dos demais modos. Além disso, como é necessário o uso do mesmo IV na encriptação e na decriptação de uma determinada mensagem, os participantes da rede ou devem possuir o IV ou devem possuir informações de como calculá-lo. Vale ressaltar ainda que o nível de segurança de um modo de operação está fortemente relacionado com o nível de segurança da cifra de bloco adjacente (DELFS; KNEBL, 2007).

4.4 Message Authentication Codes

Nesta seção, os *Message Authentication Codes* (MACs) utilizados neste trabalho são descritos e caracterizados. Além disso, são discutidos aspectos sobre o nível de segurança desses mecanismos.

4.4.1 CBCMAC

O CBCMAC (BELLARE *et al.*, 2000) é um algoritmo de autenticação baseado no modo CBC. A mensagem é criptografada no modo CBC utilizando uma cadeia de zeros como IV e o último bloco criptografado é a *tag* de autenticação. Diferentemente do CBC, o CBCMAC devolve apenas um bloco de dados, cujo tamanho é definido pelo tamanho de bloco padrão da cifra adjacente. Baseado na descrição apresentada em (BELLARE *et al.*, 2000), o funcionamento do CBCMAC é descrito no algoritmo 25.

Algoritmo 25. CBCMAC

***Entrada**: um conjunto de n blocos de texto M e uma chave k.*
***Saída**: uma tag de autenticação.*

1. *Inicialize C com uma cadeia de n zeros.*
   $C_0 \leftarrow 0^n$;
2. *Processe da seguinte forma:*
   para** i←1 **até** n **faça
   $C_i \leftarrow E_k(M_i \oplus C_{i-1})$
3. *Devolva tag* ← $\boldsymbol{C_n}$**;**

4.4.2 CMAC

O algoritmo CMAC (DWORKING, 2005) é uma evolução do CBCMAC, sendo recomendado pelo NIST. O CMAC utiliza um algoritmo que gera duas sub-chaves K1 e K2 a partir da chave principal, as quais serão utilizadas para autenticar mensagens com tamanho múltiplo ou não do tamanho de bloco padrão da cifra de bloco adjacente, respectivamente. Baseado na descrição encontrada em (DWORKING, 2005), o funcionamento do CMAC é descrito nos algoritmos 26 e 27

Algoritmo 26. CMAC – Geração das subchaves

***Entrada**: uma chave k.*
***Saída**: as subchaves K1 e K2.*

1. *Inicialize L*
   $L \leftarrow C_k(0^b)$;
2. *Processe da seguinte forma:*
   ***se** $MSB_1(L) = 0$ **então** $K1 \leftarrow L << 1$;*
   ***senão** $K1 \leftarrow (L<<1) \oplus R_b$;*
   ***se** $MSB_1(K1) = 0$ **então** $K2 \leftarrow K1 << 1$;*
   ***senão** $K2 \leftarrow (K1<<1) \oplus R_b$;*
3. *Devolva K1, K2;*

Algoritmo 27. CMAC – Geração da *tag*

***Entrada**: um conjunto de n blocos de texto claro M e uma chave k.*
***Saída**: uma tag de autenticação.*

1. *Inicialização*
   ***se** $M_{len} = 0, n=1$ **senão** $n = \lceil M_{len}/b \rceil$;*
   Seja $M \leftarrow M_1|...|M_n$, em que $M_1,...M_{n-1}$ são blocos completos de M;
2. *Processe da seguinte forma:*
   ***se** Mn é um bloco completo **então** $M_n \leftarrow K1 \oplus M_n$;*
   ***senão** $j \leftarrow nb\text{-}M_{len}\text{-}1$; $M_n \leftarrow K2 \oplus (M_n/10^j)$;*
   $C_0 \leftarrow 0^b$;
   ***para** $i \leftarrow 1$ **até** n **faça** $C_i \leftarrow C_k(C_{i-1}) \oplus M_i$;*
3. *Devolva tag $\leftarrow MSB_{Tlen}(C_n)$;*

4.4.3 HMAC

O HMAC (NIST, 2002) é um algoritmo de autenticação baseado em funções *hash*. Assim, ele utiliza uma função *hash*, em vez de uma cifra de bloco, e uma chave para autenticar uma mensagem. Antes de definir o HMAC, as funções *hash* MD5 e SHA1, as quais são utilizadas por ele, são caracterizadas nos Algoritmos 28 e 29, respectivamente.

O *Message Digest 5* (MD5) foi desenvolvida por Rivest (1992), sendo amplamente utilizada na prática para fornecer o serviço de integridade dos dados em aplicações convencionais. O funcionamento dessa função *hash* é descrito no Algoritmo 28, baseado nas definições de (MENEZES *et al.*, 1996; RIVEST, 1992), sendo que a sua execução gera um código de 128 *bits*.

Algoritmo 28. MD5

***Entrada**: um bloco de texto M.*
***Saída**: um código hash H.*

1. *Preencha x de tal forma que seu tamanho se torne congruente a 448 módulo 512 e faça x ←x|(b mod 2^{64}). Seja m múltiplo de 512, x = x_0,...x_{16m-1}, em que x_i possui 32 bits;*
2. *Inicialize A, B, C e D, buffers de 32 bits cada, da seguinte forma:*
   A ← 0x01234567; B ← 0x89ABCDEF; C ← 0xFEDCBA98; D ← 0x76543210;
3. *Defina um vetor de constantes y, tal que y_i ←valor_absoluto(seno(i+1)), 0 ≤ **i** ≤ 63;*
4. *Defina a ordem de acesso para o processamento:*
   z[0…15] ←[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15];
   z[16…31] ←[1,6,11,0,5,10,15,4,9,14,3,8,13,2,7,12];
   z[32…47] ←[5,8,11,14,1,4,7,10,13,0,3,6,9,12,15,2];
   z[48…63] ←[0,7,14,5,12,3,10,1,8,15,6,13,4,11,2,9];
5. *Finalmente, defina o número de posições de bits a ser deslocados a esquerda:*
   s[0…15] ←[7,12,17,22,7,12,17,22,7,12,17,22,7,12,17,22];
   s[16…31] ←[5,9,14,20,5,9,14,20,5,9,14,20,5,9,14,20];
   s[32…47] ←[4,11,16,23,4,11,16,23,4,11,16,23,4,11,16,23];
   s[48…63] ←[6,10,15,21,6,10,15,21,6,10,15,21,6,10,15,21];
6. *Processe da seguinte forma:*
   para** i←0 **até** m-1 **faça
   ***para** j←0 **até** 15 **faça** X[j] ← x[16i +j];*
   (AA,BB,CC,DD) ← (A,B,C,D);
   Round 1:
   ***para** j←0 **até** 15 **faça** t ← (A + f(B,C,D) + X[z[j]] + y[j]);*
   (A,B,C,D) ← (D,B + (t ↩ s[j],B,C);
   Round 2:
   ***para** j←16 **até** 31 **faça** t ← (A + g(B,C,D) + X[z[j]] + y[j]);*
   (A,B,C,D) ← (D,B + (t ↩ s[j],B,C);
   Round 3:
   ***para** j←32 **até** 47 **faça** t ← (A + h(B,C,D) + X[z[j]] + y[j]);*
   (A,B,C,D) ← (D,B + (t ↩ s[j],B,C);
   Round 4:
   ***para** j←48 **até** 63 **faça** t ← (A + l(B,C,D) + X[z[j]] + y[j]);*
   (A,B,C,D) ← (D,B + (t ↩ s[j],B,C);
   (A,B,C,D) ← (A + AA, B + BB, C + CC, D + DD);
7. *Devolva o valor hash H←A|B|C|D.*

Por outro lado, o *Secure Hash Algorithm* 1 (SHA1) foi desenvolvida pela agência norte-americana *National Security Agency* (NSA), sendo também largamente utilizado na prática, assim como o MD5. O seu funcionamento possui semelhanças ao MD5, como pode ser observado no Algoritmo 29, o qual é baseado nas descrições de (MENEZES *et al.*, 1996; EASTLAKE; JONES, 2001). Entretanto, ao fim de sua execução, o algoritmo SHA1 gera um código de 160 *bits*. Utilizando funções como o MD5 e o SHA1 e uma chave, o HMAC opera conforme o Algoritmo 30, baseado em (NIST, 2002), para gerar uma *tag* de autenticação cujo tamanho depende também da função *hash* utilizada.

Algoritmo 29. SHA1

***Entrada**: um bloco de texto M.*
***Saída**: um código hash H.*

1. *Preencha x de tal forma que seu tamanho se torne congruente a 448 módulo 512 e faça x ←x|b. Seja m múltiplo de 512, $x = x_0,...x_{16m-1}$, x_i de 32 bits;*
2. *Inicialize A, B, C e D, buffers de 32 bits cada, da seguinte forma:*
   $h_0 \leftarrow 0x67452301$; $h_1 \leftarrow 0xEFCDAB89$; $h_2 \leftarrow 0x98BADCFE$; $h_3 \leftarrow 0x10325476$; $h_4 \leftarrow 0xC3D2E1F0$;
3. *Defina um vetor de constantes y:*
   $y_1 \leftarrow 0x5A827999$; $y_2 \leftarrow 0x6ED9EBA1$; $y_3 \leftarrow 0x8F1BBCDC$; $y_4 \leftarrow 0xCA62C1D6$;
4. *Processe da seguinte forma:*
   para** i←0 **até** m-1 **faça
   ***para** j←0 **até** 15 **faça** X[j] ← x[16i +j];*
   ***para** j←16 **até** 79 **faça** $X[j] \leftarrow ((X_{j-3} \oplus X_{j-8} \oplus X_{j-14} \oplus X_{j-16}) \circlearrowleft 1)$;*
   $(A,B,C,D,E) \leftarrow (h_0,h_1,h_2,h_3,h_4)$;
   Round 1:
   ***para** j←0 **até** 19 **faça** $t \leftarrow ((A \circlearrowleft 5) + f(B,C,D) + E + H[j] + y_1)$;*
   $(A,B,C,D,E) \leftarrow (t,A,B \circlearrowleft 30,C,D)$;
   Round 2:
   ***para** j←20 **até** 39 **faça** $t \leftarrow ((A \circlearrowleft 5) + h(B,C,D) + E + X[j] + y_2)$;*
   $(A,B,C,D,E) \leftarrow (t,A,B \circlearrowleft 30,C,D)$;
   Round 3:
   ***para** j←40 **até** 59 **faça** $t \leftarrow ((A \circlearrowleft 5) + g(B,C,D) + E + X[j] + y_3)$;*
   $(A,B,C,D,E) \leftarrow (t,A,B \circlearrowleft 30,C,D)$;
   Round 4:
   ***para** j←60 **até** 79 **faça** $t \leftarrow ((A \circlearrowleft 5) + h(B,C,D) + E + X[j] + y_4)$;*
   $(A,B,C,D,E) \leftarrow (t,A,B \circlearrowleft 30,C,D)$;
   $(A,B,C,D,E) \leftarrow (h_0 + h_1 + B, h_2 + C, h_3 + D, h_4 + E)$;
5. *Devolva o valor hash H←A|B|C|D|E.*

Algoritmo 30. HMAC

***Entrada**: um bloco de texto M e uma chave K. (OBS. hash é uma função hash com tamanho de saída padrão igual a B e len(X) é o tamanho de X.)*
***Saída**: uma tag de autenticação.*

1. ***Inicialização***
   ipad ← 0x36 repetido B vezes;
   opad ← 0x5C repetido B vezes;
2. ***Processe da seguinte forma:***
   se len(K) ≠B então
   ***se** len(K) > B **então** $K \leftarrow hash(K)|0^{B-l}$;*
   ***se** len(K) < B **então** $K \leftarrow K|0^{B-len(K)}$;*
3. *Devolva tag← hash((K0⊕opad)|H((K0⊕ipad)|M));*

4.4.4 Nível de Segurança

Um dos principais fatores relacionados ao nível de segurança de um MAC é o tamanho *len* da *tag* de autenticação gerada, sendo definido de tal forma que a probabilidade de um adversário falsificar um código válido é dada por 2^{-len}. Além disso, caso necessário, o tamanho da *tag* poderá ser truncado, considerando apenas os *n* primeiros bits mais significativos. Entretanto, é recomendado o uso de um tamanho de no mínimo L/2, em que L é o tamanho em *bits* do código padrão do algoritmo MAC, podendo ser relaxado para no mínimo 32 *bits* para aplicações nas quais o número possível de tentativas de falsificações repetidas por um adversário seja limitado pela largura de banda de comunicação (NIST, 2002), como é o caso das RSSF.

Considerando o contexto atual de RSSF, no qual os dispositivos possuem capacidade de transmissão em torno de 250 *kbps*, uma *tag* de 32 *bits* – tamanho padrão utilizado na maioria dos protocolos de segurança para RSSF (KARLOF *et al.*, 2004; LUK *et al.*, 2004) – poderá ser forjada por um adversário em apenas 90 dias de contínuas tentativas (JINWALA *et al.*, 2008). Assim, utilizar uma *tag* maior pode ser uma alternativa para aumentar o nível de segurança em RSSF ao custo do aumento do consumo de energia.

O algoritmo CBCMAC apresenta vulnerabilidades para autenticar mensagens de tamanho variável sendo necessário o uso de alguma estratégia como, por exemplo, a encriptação do tamanho da mensagem seguida da soma (XOR) com o primeiro bloco de texto claro (BELLARE *et al.*, 2000). Já o CMAC foi desenvolvido para suprir as deficiências do CBCMAC, sendo recomendado pelo NIST. Além disso, o nível de segurança do CMAC, assim como do CBCMAC, depende do nível de segurança das cifras de bloco adjacentes.

Com relação ao HMAC, embora tenha sido encontradas vulnerabilidades no MD5 e SHA1, o HMAC não é afetado por elas (SCHNEIER, 2005), relevando o potencial nível de segurança para RSSF.

4.5 Protocolos de Segurança

Os principais protocolos de segurança desenvolvidos para RSSF são o Tinysec (KARLOF *et al.*, 2004) e o Minisec (LUK *et al.*, 2004).

O protocolo TinySec foi o primeiro protocolo de segurança específico para RSSF totalmente implementado. Esse protocolo foi desenvolvido com o objetivo de prover os serviços de segurança confidencialidade, integridade e autenticação para RSSF por meio de duas configurações. Uma de suas configurações consiste em prover somente os serviços de integridade e autenticação, enquanto que a outra versão fornece também, além de integridade e autenticação, o serviço de confidencialidade.

Para fornecer esses serviços de segurança, o Tinysec implementa os seguintes mecanismos de segurança: a cifra de bloco Skipjack com o modo de operação criptográfico CBC (confidencialidade) e o algoritmo CBCMAC (integridade e autenticação). Por outro lado, a ausência de mecanismos de proteção contra ataques de repetição (*replay*) e o baixo nível de segurança oferecido pela cifra de bloco padrão são os principais fatores limitantes desse protocolo. Entretanto, apesar das limitações, o Tinysec é o principal protocolo de segurança utilizado em RSSF. Ele é integrado à versão 1.x do sistema operacional TinyOS e, inclusive, foi utilizado comercialmente em dispositivos comercializados pela Bosch (BOYLE; NEWE, 2008).

De outra forma, o protocolo Minisec foi desenvolvido com o objetivo de fornecer não só os serviços de confidencialidade, integridade e autenticação, mas também a proteção contra ataques de repetição. O Minisec possui duas configurações as quais utilizam a cifra de bloco Skipjack em conjunto com o modo de operação criptográfico OCB para fornecer os três serviços básicos de segurança (confidencialidade, integridade e autenticação).

Por outro lado, o Minisec oferece diferentes mecanismos de sincronização para fornecer a proteção contra ataques de repetição, conforme a versão do protocolo. Uma descrição completa desses mecanismos pode ser encontrada em (LUK *et al.*, 2004).

Vale ressaltar que ambos os principais protocolos específicos para RSSF, caracterizados nesta seção, empregam a cifra Skipjack. Entretanto, conforme visto na seção 4.2, esse mecanismo oferece um baixo nível de segurança, o que ratifica a fundamental importância da avaliação de desempenho em busca de mecanismos de segurança alternativos para o contexto de RSSF.

4.6 Trabalhos Relacionados

Diversos trabalhos têm avaliado o desempenho de mecanismos de segurança, entre os quais se destacam (GUIMARÃES *et al.*, 2005; LAW *et al.*, 2006; GROBSCHADL *et al.*,2007;

BAUER *et al.*, 2009; CASADO; TSIGAS, 2009; ÇAKIROGLU *et al.*, 2010; LEE *et al.*, 2010; SZALACHOWSKI *et al.*, 2010; CAVALCANTE *et al.*, 2011).

Vale ressaltar que a maior parte dos trabalhos relacionados publicados na literatura concentra-se na avaliação de cifras de bloco. Por outro lado, existem poucos trabalhos disponíveis na literatura que objetivam avaliar o desempenho de modos de operação criptográficos, assim como algoritmos de autenticação para RSSF.

Por exemplo, Guimarães *et al.* (2005) avaliaram o desempenho das cifras simétricas Skipjack, RC5, RC6, TEA e DES com o objetivo de verificar o impacto no processamento e na quantidade de memória requerida por cada algoritmo em plataformas Mica2. Os autores concluíram que a cifra RC5 é a mais eficiente. Entretanto, eles utilizaram uma configuração do RC5 com apenas 12 *rounds*, a qual é considerada insegura (POTLAPALLY *et al.*, 2005).

Já Law *et al.* (2006) estudaram e avaliaram o desempenho dos algoritmos de criptografia RC5, RC6, Rijndael, MISTY1, KASUMI e Camelia, além dos modos de operação CBC, OFB, CFB e CTR em microcontroladores MSP430F149 da *Texas Instruments*. Considerando as métricas avaliadas, a quantidade de memória requerida e o consumo de energia das cifras, os autores concluíram que o Rijndael (AES) é a opção mais eficiente para o uso em RSSF. Com relação aos modos, o OFB apresentou a menor quantidade de memória requerida e o menor consumo de energia, enquanto o CBC apresentou o pior desempenho em termos de consumo de energia. Entretanto, os autores não consideraram mecanismos de autenticação.

Grobschadl *et al.* (2007) também apresentaram um estudo avaliando o desempenho de cifras de bloco para dispositivos com recursos de computação limitados. Os autores avaliaram o desempenho dos algoritmos criptográficos RC6, AES, Serpent, Twofish e XTEA por meio de simulações, utilizando os parâmetros do processador StrongARM AS-1100. No entanto, os modos de operação e os algoritmos de autenticação não foram considerados. Além disso, os autores não indicaram detalhes importantes do processo de simulação, como as medidas estatísticas utilizadas e o intervalo de confiança.

Por outro lado, Bauer *et al.* (2009) avaliaram o desempenho dos mecanismos de segurança capazes de fornecer encriptação e autenticação OCB, EAX, CCFB+H e GCM na plataforma MicaZ com relação à quantidade de memória RAM requerida e o tempo de execução. Entretanto, os autores avaliaram o comportamento desses mecanismos variando a quantidade de

bytes de entrada até no máximo 29 *bytes*. Além disso, não foram considerados os mecanismos de criptografia.

Um outro trabalho relacionado é o de Casado e Tsigas (2009), que avaliaram os algoritmos de criptografia AES, RC5, Skipjack, Triple-DES, Twofish e XTEA e os mecanismos de autenticação CMAC e OCB utilizando o AES como cifra de bloco adjacente para gerar uma tag de 4 *bytes* a partir de diferentes tamanhos de entrada. As métricas utilizadas foram a quantidade de memória requerida, o tempo de execução e o consumo de energia, por meio de experimentos reais na plataforma MSB-430, utilizando o sistema operacional Contiki. Os autores concluíram que o Skipjack é a cifra mais eficiente em termos de energia. No entanto, os autores não explicitaram a configuração da cifra RC5 utilizada nos experimentos.

Além dos trabalhos citados anteriormente, Çakiroglu *et al.* (2010) apresentaram um estudo avaliando os algoritmos criptográficos SEA, RC6 e AES, sendo que as métricas consideradas pelos autores foram a quantidade de memória requerida e o tempo de execução de cada mecanismo. Os autores realizaram a avaliação por meio de simulações. Entretanto, não foram considerados modos de operação criptográficos nem algoritmos de autenticação. Além disso, o consumo de energia também não foi avaliado.

Ainda importante mencionar, o trabalho de Szalachowski *et al.* (2010), os quais apresentaram um estudo no qual avaliaram os códigos de autenticação de mensagens CMAC, CCM e GMAC em plataformas Iris utilizando o sistema operacional *LiteOS*. Os autores consideram as métricas quantidade de memória requerida e o tempo de execução de cada mecanismo. No entanto, os autores não consideraram mecanismos de criptografia nem modos de operação criptográficos.

Finalmente, Lee *et al.* (2010) realizaram um trabalho denso sobre segurança em RSSF, avaliando mecanismos por meio de experimentos reais em plataformas MicaZ e TelosB, considerando as métricas quantidade de memória requerida e consumo de energia. Os algoritmos analisados foram as cifras Skipjack, RC5, AES e o XXTEA, os modos CBC, CFB, OFB, CTR e OCB e os MACs CBCMAC, XMAC, CMAC e HMAC-MD5.

De acordo com os resultados obtidos pelos autores, o Skipjack e o AES apresentaram o melhor e o pior desempenho em termos de consumo de energia, respectivamente. Além disso, o RC5-64/18/16 apresentou melhor desempenho que o Skipjack e o AES com relação à quantidade

de memória requerida. Já o modo de operação CFB apresentou o melhor desempenho entre os demais modos, sendo que o modo CBC foi o que apresentou o pior desempenho.

Já com relação aos MACs, o HMAC-MD5 foi o que apresentou o melhor desempenho com relação à memória RAM, enquanto que o CMAC apresentou a menor quantidade requerida memória ROM. Com relação ao consumo de energia, o CBCMAC apresentou o melhor desempenho para todos os tamanhos de entrada avaliados, com exceção do tamanho aproximado de 128 *bytes*, em que o HMAC-MD5 levou vantagem. Entretanto, os autores não apresentaram informações estatísticas importantes como a quantidade de experimentos realizados ou o intervalo de confiança das medições.

Durante o desenvolvimento desta dissertação, resultados preliminares desta pesquisa foram publicados em (CAVALCANTE *et. al.*, 2011). Esses resultados foram alcançados na avaliação de desempenho dos algoritmos criptográficos Skipjack e RC5 na plataforma MicaZ para RSSF, na qual foram encontradas diferenças significativas com relação aos resultados apresentados nos trabalhos anteriormente citados. Entretanto, não foi considerada a variação da quantidade de *bytes* de entrada dos algoritmos nem modos de operação e mecanismos de autenticação.

Em resumo, as principais diferenças entre os trabalhos relacionados e esta proposta estão dispostas na tabela 1.

Tabela 1. Comparativo entre os trabalhos relacionados.

Trabalho	**Avaliou**			**Considerou**	**Realizou**	**Indicou**	
	Cifras de bloco?	**Modos de operação?**	**MACs?**	**Diferentes tamanhos de entrada?**	**Experimentos reais?**	**Número de repetições dos experimentos?**	**Intervalo de confiança?**
BAUER *et al.*, 2009	Não	Sim	Não	Sim	Não	-	Não
ÇAKIROGLU *et al.*, 2010	Sim	Não	Não	Não	Não	-	Não
CASADO; TSIGAS, 2009	Sim	Sim	Sim	Não	Sim	Não	Não
CAVALCANTE *et al.*, 2011	Sim	Não	Não	Não	Sim	Sim	Sim
GROBSCHADL *et al.*,2007	Sim	Não	Não	Não	Não	-	Não
GUIMARÃES *et al.*, 2005	Sim	Não	Não	Não	Sim	Sim	Não
LAW *et al.*, 2006	Sim	Sim	Não	Não	Sim	Não	Não
LEE *et al.*, 2010	Sim	Sim	Sim	Sim	Sim	Não	Não
SZALACHOWSKI *et al.*, 2010	Não	Sim	Sim	Sim	Sim	Não	Não
Esta proposta	**Sim**	**Sim**	**Sim**	**Sim**	**Sim**	**Sim**	**Sim**

4.7 Conclusão

Este Capítulo descreveu todos os mecanismos de segurança utilizados na avaliação de desempenho proposta nesta dissertação, inclusive discutindo aspectos sobre o nível de segurança apresentado por cada mecanismo.

Além disso, os dois principais protocolos de segurança específicos para RSSF (TinySec e Minisec) foram caracterizados, ressaltando-se que esses protocolos empregam mecanismos que apresentam baixo nível de segurança.

Ainda neste capítulo, foi visto que a literatura inclui diversos trabalhos que objetivam estudar a viabilidade e identificar os mecanismos de segurança mais adequados para prover os serviços de confidencialidade, integridade e autenticação no contexto de RSSF. Essa diversidade de trabalhos apresentam muitas vezes grandes diferenças em termos de, por exemplo: a plataforma escolhida para realizar os experimentos; as métricas selecionadas na avaliação; ou a metodologia empregada para realizar a avaliação de desempenho.

Além disso, muitos trabalhos não fornecem informações precisas sobre os dados obtidos nas medições ou simulações realizadas como a repetição de experimentos ou o intervalo de confiança dos dados, dificultando a realização de uma comparação minuciosa entre os mecanismos considerados. Dos diversos trabalhos discutidos nesta seção, apenas o trabalho de Cavalcante *et. al.* (2011), o qual é fruto das pesquisas desta dissertação, incluiu o intervalo de confiança dos dados nas informações sobre o processo de medição e obtenção dos resultados do trabalho.

Portanto, a inclusão de mecanismos não avaliados nos trabalhos anteriores, a descrição detalhada do processo de medição e o fornecimento de informações sobre as medidas estatísticas utilizadas, inclusive seguindo os princípios de avaliação de desempenho convencionais, ressaltam a importância e o caráter diferencial desta dissertação com relação aos trabalhos relacionados disponíveis na literatura.

5 AVALIAÇÃO DE DESEMPENHO DE MECANISMOS DE SEGURANÇA PARA RSSF

Este capítulo apresenta a descrição completa do processo, baseado em uma abordagem sistemática existente na literatura, utilizado para avaliar o desempenho dos mecanismos de segurança no contexto de RSSF. A seção 5.1 introduz o capítulo com uma visão geral do processo de avaliação adotado nesta dissertação. Logo após, as seções numeradas de 5.2 até 5.7 abordam cada fase do processo de avaliação de desempenho utilizado neste trabalho. Em seguida, a seção 5.8 finaliza o capítulo com uma conclusão.

5.1 Introdução

A avaliação de desempenho realizada nesta dissertação é baseada nos princípios e técnicas de análise de desempenho dispostos na abordagem proposta por Jain (1991). Essa abordagem é uma das principais referências voltadas para a avaliação de desempenho de sistemas computação e o seu uso é justificado pela capacidade de ajudar a evitar problemas comuns envolvidos com o processo de avaliação como, por exemplo, a escolha inadequada da carga de trabalho (*workload*), o não estabelecimento de objetivos, o estabelecimento de objetivos tendenciosos, ou até mesmo a escolha incorreta de métricas de desempenho e a seleção de cargas de trabalho não representativas.

Dessa forma, este trabalho segue uma abordagem sistemática adaptada, conforme ilustrado no diagrama da Figura 7, a partir dos conceitos e princípios de avaliação de desempenho de Jain (1991). Conforme essa abordagem elaborada nesta dissertação e mostrada na figura 7, o início do processo de avaliação é determinado pelo estabelecimento dos objetivos a serem alcançados com a análise de desempenho proposta e a definição do sistema a ser avaliado (a).

Em seguida, as funcionalidades oferecidas pelo sistema objeto de avaliação, neste caso os mecanismos de segurança para RSSF, são caracterizados para ajudar na seleção das métricas a serem utilizadas para comparar o desempenho dos mecanismos de segurança entre si (b). Com as métricas devidamente selecionadas (c), a técnica de avaliação e a carga de trabalho são escolhidas

para definir a estratégia a ser utilizada na realização dos experimentos (d). Por fim, após a etapa de experimentação (e), os resultados obtidos são analisados e apresentados (f).

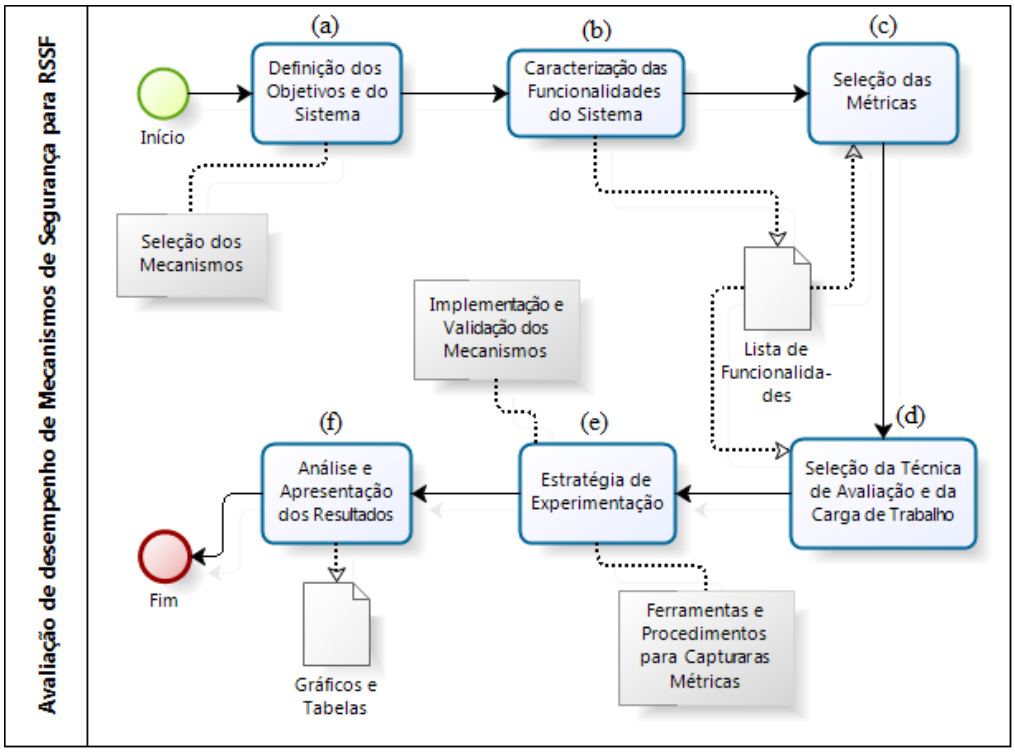

Figura 7. Diagrama da abordagem de avaliação baseada em (JAIN, 1991).

Portanto, considerando a abordagem mencionada anteriormente, as tarefas e procedimentos realizados para alcançar o objetivo geral deste trabalho são apresentados e caracterizados nas próximas seções.

5.2 Definição dos Objetivos e do Sistema (a)

O objetivo principal do processo de avaliação de desempenho dos mecanismos de segurança utilizados para fornecer os serviços de confidencialidade, integridade e autenticação em RSSF, que também é o objetivo geral desta dissertação, conforme indicado no Capítulo 1, foi definido da seguinte forma:

- *Determinar os mecanismos de segurança mais adequados, com relação ao desempenho, para fornecer os seguintes serviços de segurança em RSSF: confidencialidade, integridade e autenticação;*

Considerando o objetivo estabelecido anteriormente, a proposta desta dissertação consiste na realização de uma análise de desempenho, em uma plataforma utilizada como sensores inteligentes no contexto atual de RSSF, de um número variado de mecanismos de segurança capazes de fornecer os serviços de confidencialidade, integridade e autenticação.

Dentro do contexto desta avaliação, o sistema a ser avaliado compreende aos mecanismos de segurança. Assim, o processo de seleção dos mecanismos de segurança que participaram da avaliação de desempenho proposta neste trabalho é discutido nas próximas subseções e a plataforma selecionada para realizar os experimentos é caracterizada mais adiante neste capítulo (ver seção 5.5).

5.2.1 Seleção dos Mecanismos de Segurança

Para alcançar o objetivo geral citado anteriormente, é necessária a realização de um processo de escolha dos mecanismos de segurança a serem avaliados. Nesse processo, foram estabelecidos alguns critérios com o objetivo de nortear a filtragem dos diversos mecanismos de segurança existentes na literatura para realizar a avaliação de desempenho proposta neste trabalho. Esses critérios são listados de acordo com o tipo de mecanismo de segurança e justificados nas próximas subseções.

5.2.1.1 Criptografia

Os principais critérios de seleção dos mecanismos criptográficos a serem avaliados nesta dissertação são listados a seguir:

a) *mecanismos considerados seguros*. Os mecanismos criptográficos escolhidos não devem apresentar ataques conhecidos com relação à criptoanálise ou, caso haja trabalhos disponíveis, os melhores ataques correspondentes publicados na literatura

não devem indicar o comprometimento do nível de segurança oferecido pela cifra de bloco; ou

b) *avaliação das cifras de bloco utilizadas nos principais protocolos de segurança para RSSF*. As cifras de bloco já empregadas nos principais protocolos de segurança específicos para o contexto de RSSF serão incluídas na avaliação de desempenho; ou

c) *tamanhos dos principais parâmetros de uma cifra de bloco*. Outro critério utilizado para filtrar as diversas cifras de bloco existentes foi a limitação do tamanho dos principais parâmetros de uma cifra de bloco. Assim, optou-se por escolher cifras de bloco com um tamanho de bloco padrão de 64 *bits* e tamanho da chave de até 128 *bits*.

Vale ressaltar que os parâmetros mencionados na alínea *d* exercem uma grande influência em termos de desempenho (tempo de execução e energia) e nível de segurança, pois em geral quanto maior forem esses parâmetros, maior será o nível de segurança oferecido pela cifra ao custo de uma degradação no desempenho. Entretanto, fatores como a manipulação de registradores internos e de resultados intermediários no microcontrolador afetam significativamente o desempenho de dispositivos com capacidade de computação limitada, como é o caso dos sensores inteligentes (GONG *et al.*, 2011).

Por outro lado, com relação à chave criptográfica utilizada por uma cifra de bloco, é recomendada a utilização de chaves com tamanho próximas de 128 *bits* para garantir segurança computacional suficiente contra ataques de força bruta no contexto atual (JINWALLA *et al.*, 2008).

Portanto, buscando satisfazer o maior número de critérios relacionados anteriormente, foram escolhidos cinco mecanismos criptográficos, os quais são dispostos e caracterizados resumidamente na tabela 2. Os valores destacados na tabela 2 representam os parâmetros[7] determinados para o processo de avaliação, os quais foram determinados levando em consideração os critérios *a* e *c*. Assim, por exemplo, o AES foi utilizado com a configuração 128,

[7] A fase de identificação dos parâmetros foi omitida no processo de avaliação elaborado, pois eles foram fixados. Da mesma forma, a seleção dos fatores (parâmetros que variam durante a avaliação) também foi omitida, pois eles não foram identificados.

128, e 10 relacionados ao tamanho do bloco padrão, tamanho da chave e o numero de *rounds*, respectivamente.

Tabela 2. Características das cifras de bloco avaliadas.

Cifra de bloco	**Tamanho do bloco (*bits*)**	**Tamanho da Chave (*bits*)**	**Número de *rounds***
AES	**128**	**128**, 196 ou 256	**10**, 12 ou 14
Skipjack	**64**	**80**	**32**
Klein	**64**	64, 80 ou **96**	12, 16 ou **20**
RC5	32, **64** ou 128	0 a 2040 (**128**)	0 a 255 (**18**)
Present	**64**	80 ou **128**	**32**

Conforme pode ser observado na tabela 2, as cifras de bloco AES, Skipjack e Klein não obedecem à totalidade dos critérios definidos anteriormente, pois o AES possui tamanho de bloco padrão de 128 *bits* e o Skipjack e o Klein possuem tamanho de chave criptográfica menor do que 128 *bits*. Entretanto, a inclusão dessas cifras de bloco na avaliação é justificada pelos motivos descritos a seguir.

Em primeiro lugar, conforme indicado na seção 4.2.1, o AES é a cifra padrão mundial de criptografia e utilizada universalmente em sistemas para fornecer o serviço de confidencialidade, o que torna uma opção atrativa para realizar a comparação de desempenho com outras cifras.

Em segundo lugar, o Skipjack é a cifra de bloco padrão dos principais protocolos de segurança específicos para RSSF, o Tinysec e o Minisec. Sendo assim, da mesma forma que o AES, é interessante comparar o seu desempenho com relação às demais cifras de bloco na avaliação.

Por outro lado, embora a cifra de bloco Klein apresente um tamanho de chave menor do que 128 *bits*, não foram encontrados ataques criptoanalíticos contra a configuração do algoritmo utilizando o tamanho de chave criptográfica igual a 96 *bits*, justificando a sua inclusão na avaliação de desempenho proposta neste trabalho.

5.2.1.2 Modos de Operação

Conforme mencionado na seção 4.6, existem poucos trabalhos disponíveis na literatura que objetivam avaliar o desempenho de modos de operação criptográficos para RSSF. Levando esse fato em consideração, o seguinte critério principal foi estabelecido na escolha dos modos de operação a serem avaliados:

a) avaliação de modos de operação utilizados nos principais protocolos de segurança para RSSF. Os modos de operação criptográficos utilizados nos principais protocolos de segurança específicos para RSSF serão incluídos na avaliação de desempenho; ou

b) Modos de operação padronizados. Os modos de operação criptográficos padronizados por autoridades competentes serão incluídos na avaliação de desempenho.

Os modos de operação escolhidos de acordo com o primeiro critério mencionado anteriormente foram o CBC, empregado no protocolo Tinysec e o modo OCB, utilizado no protocolo Minisec. A caracterização de ambos os modos pode ser observada na seção 4.3.

Além disso, os modos CFB, OFB e CTR foram incluídos na avaliação de desempenho por serem modos de operação, assim como o modo CBC, padronizados e recomendados pelo instituto norte americano NIST para uso em sistemas criptográficos.

5.2.1.3 Códigos de Autenticação de Mensagens

Da mesma maneira que ocorre com os modos de operação criptográficos, há um número reduzido de trabalhos publicados na literatura dedicados a avaliar o impacto de mecanismos de autenticação para RSSF, conforme indicado na seção 4.6. Levando isso em consideração, os seguintes critérios orientaram a escolha dos códigos de autenticação de mensagens a serem avaliados nesta dissertação:

a) MACs considerados seguros. Os MACs escolhidos não devem apresentar vulnerabilidades que prejudiquem seu nível de segurança ofertado ou, caso ataques

tenham sido descobertos, os melhores ataques correspondentes publicados na literatura não devem indicar o comprometimento do nível de segurança oferecido pelo código de autenticação de mensagens; ou

b) avaliação dos MACs utilizados nos principais protocolos de segurança para RSSF. Os MACs já empregados nos principais protocolos de segurança específicos para RSSF serão incluídos na avaliação de desempenho; ou

c) *MACs padronizados.* Os MACs padronizados por autoridades competentes serão incluídos na avaliação de desempenho.

Considerando os critérios estabelecidos para a escolha dos MACs a serem avaliados neste trabalho relacionados anteriormente, os seguintes MACs foram selecionados: CBCMAC, CMAC e HMAC-SHA1 e HMAC-MD5.

O CBCMAC atende ao critério indicado na alínea *b*, uma vez que ele é o código de autenticação de mensagens empregado no protocolo Tinysec, o qual utiliza o Skipjack como cifra de bloco adjacente, conforme mencionado na seção 4.5.

Por outro lado, o CMAC foi incluído devido ao fato de que, além de satisfazer o critério indicado na alínea *a*, ele é um MAC recomendado pelo instituto norte americano NIST e apresenta melhorias com relação ao algoritmo CBCMAC, conforme pode ser observado na subseção 4.4.4.

O HMAC foi selecionado porque ele obedece às alíneas *a* e *b*, uma vez que ele é um código de autenticação de mensagens aprovado pelo instituto norte americano NIST. É bom lembrar ainda que o HMAC foi avaliado utilizando as duas principais funções *hash* utilizadas em sistemas de segurança, os quais são o MD5 e o SHA1, as duas principais funções *hash* existentes na literatura, cujos funcionamentos são descritos no Capítulo 4.

5.3 Caracterização das Funcionalidades do Sistema (b)

Conforme indicado por Jain (1991), os sistemas em geral são capazes de prover um conjunto de funcionalidades, em que a identificação clara dos possíveis resultados dessas

funcionalidades é uma tarefa importante para ajudar na seleção das métricas de desempenho a serem consideradas na avaliação.

Dessa forma, baseado nos conceitos apresentados por Jain (1991), as seguintes funcionalidades dos mecanismos de segurança foram identificadas no contexto da análise de desempenho proposta nesta dissertação:

a) a encriptação fornecida por uma cifra de bloco;

b) a decriptação fornecida por uma cifra de bloco;

c) a expansão da chave criptográfica realizada por uma cifra de bloco;

d) a encriptação fornecida pelo conjunto formado por uma cifra de bloco e um modo de operação;

e) a decriptação fornecida pelo conjunto formado por uma cifra de bloco e um modo de operação; e

f) a autenticação fornecida por um código de autenticação de mensagens.

Uma vez identificadas as principais funcionalidades oferecidas pelos mecanismos de segurança estudados neste trabalho, o processo de seleção das métricas é realizado, conforme discutido na próxima seção.

5.4 Seleção das Métricas (c)

Após a etapa de identificação dos serviços oferecidos pelos mecanismos de segurança estudados nesta dissertação, foi realizada a seleção do critério a ser utilizado para comparar o desempenho entre os mecanismos de segurança. Tal critério de desempenho é conhecido como métrica. Para realizar a definição das métricas foram levadas em consideração, além dos serviços

listados na seção anterior, as principais características limitantes apresentadas pelos sensores inteligentes no contexto de uma RSSF.

Conforme apresentado no Capítulo 2, os nós sensores empregados em uma RSSF apresentam como principais características: capacidade limitada de armazenamento (memória ROM e RAM), processamento, e fonte de energia.

Portanto, de acordo com as considerações mencionadas anteriormente, as métricas selecionadas para realizar na avaliação de desempenho dos mecanismos de segurança são listadas a seguir: quantidade de memória ROM requerida, quantidade de memória RAM requerida, tempo de execução e consumo de energia.

5.5 Seleção da Técnica de Avaliação e da Carga de Trabalho (d)

Conforme explicado por Jain (1991), existem três técnicas que podem ser utilizadas em avaliação de desempenho, as quais são a modelagem analítica, a simulação e a medição. Além disso, a seleção da técnica de avaliação correta depende de fatores como o tempo e os recursos disponíveis para resolver o problema de avaliação e do nível de confiança desejado.

Uma vez que existia a disponibilidade dos recursos necessários para realizar experimentos reais e a necessidade de um alto nível de confiança dos resultados, a medição foi a técnica de avaliação selecionada neste trabalho.

Dessa forma, foram elaboradas as estratégias de medição, apresentadas na próxima seção, para avaliar o desempenho dos mecanismos de segurança na plataforma MicaZ, uma plataforma muito utilizada em aplicações de RSSF, sendo bastante popular (LIU; NING, 2008). A plataforma MicaZ, ilustrada na Figura 8, foi desenvolvida pela empresa *Crossbow Technology* e suas principais características são apresentadas na tabela 3.

Figura 8. Plataforma MicaZ (CROSSBOW, 2006).

Tabela 3. Características da plataforma MicaZ (CROSSBOW, 2006).

Especificação	**MicaZ**
Processador	ATMEGA 128L
Memória ROM	128 KB
Memória RAM	4 KB
Alimentação *default*	2.7 – 3 V
Radio	CC2420 2.4 GHz
Máxima taxa de transmissão	250 *kbps*
Canais A/D	8 de 10 *bits*

Com relação à seleção da carga de trabalho, considerando a identificação dos serviços mencionada na seção 5.3 e a escolha das métricas de desempenho, foram estabelecidas as seguintes requisições para realizar o processo de medição com relação à confidencialidade (*a*, *b*, *c*, *d*, *e*), integridade (*f*) e autenticação (*f*):

a) criptografar uma determinada quantidade de *bytes* com uma cifra de bloco;

b) decriptografar uma determinada quantidade de *bytes* com uma cifra de bloco;

c) inicializar uma chave com uma cifra de bloco;

d) criptografar uma determinada quantidade de *bytes* pelo conjunto formado por uma cifra de bloco e um modo de operação;

e) decriptografar uma determinada quantidade de *bytes* pelo conjunto formado por uma cifra de bloco e um modo de operação; e

f) autenticar uma determinada quantidade de *bytes* com um código de autenticação de mensagens.

5.6 Estratégias de Experimentações (e)

Nesta seção são apresentados aspectos de implementação e validação dos mecanismos de segurança, a aplicação desenvolvida e as estratégias de obtenção das métricas escolhidas para avaliar os algoritmos estudados neste trabalho, além da descrição dos experimentos realizados.

5.6.1 Implementação dos Mecanismos de Segurança

Para realizar a avaliação de desempenho dos mecanismos de segurança, foram utilizadas implementações na linguagem *nesC*, uma variação da linguagem C desenvolvida para incorporar os conceitos e o modelo de execução do *TinyOS*, sendo uma linguagem específica para esse sistema operacional (GAY *et al.*, 2005).

O *TinyOS* é um sistema operacional desenvolvido em 2000 na UC Berkeley para dispositivos com recursos limitados utilizados em RSSF, sendo um dos principais sistemas operacionais utilizados em RSSF (DONG *et al.*, 2010). Atualmente, ele é *open-source* e possui uma comunidade internacional de desenvolvedores e usuários (TINYOS DOCUMENTATION, 2010). Entre as principais características do *TinyOS*, pode-se citar: a arquitetura baseada em componentes, o modelo de concorrência simples baseado em eventos e as operações divididas em fases (*split-phase*) (GAY *et al.*, 2005). Vale ressaltar que, embora a versão atual do *TinyOS* seja a 2.x, a versão 1.x é a mais estável.

Dessa forma, com o objetivo de evitar retrabalho foram realizadas buscas por implementações dos mecanismos de segurança na própria linguagem *nesC*. Entretanto, devido à

indisponibilidade de implementações nessa linguagem, as implementações de alguns mecanismos de segurança foram portados e adaptados a partir do código escrito na linguagem C. Além disso, devido à indisponibilidade de códigos nas linguagens *nesC* e C, alguns mecanismos foram implementados diretamente na linguagem *nesC* a partir da descrição original de alto nível.

A indicação de cada mecanismo de segurança e da fonte de sua implementação são relacionadas na tabela 4. Nessa tabela, as implementações indicadas com o termo "Código em *nesC*" foram reutilizadas integralmente, enquanto que as demais implementações são resultados da portabilidade, modificação ou implementação a partir da descrição de alto nível referenciada.

Todas as implementações utilizadas na avaliação de desempenho proposta neste trabalho podem ser encontradas na página *web*:

https://sourceforge.net/projects/evaluationwsn/

Tabela 4. Implementação dos mecanismos de segurança.

Mecanismo	**Implementação**
AES	Código em *nesC* (IIS, 2011)
Skipjack	Código em *nesC* (LUK *et al.*, 2007)
Klein	Código em *nesC* (GONG *et al.*, 2010)
RC5	Código em *nesC* (KARLOF *et al.*, 2004)
Present	Portado do código em *C* (CIS LAB, 2011)
CBC, CFB, OFB, CTR	Implementado a partir de (DWORKING, 2001)
OCB	Código em *nesC* (LUK *et al.*, 2007)
CBCMAC	Adaptado do código em *nesC* (KARLOF *et al.*, 2004)
CMAC	Implementado a partir de (DWORKING, 2005)
HMAC-MD5, HMAC-SHA1	Portado do código em *C* (MD5DEEP, 2011)

5.6.2 Validação dos Mecanismos de Segurança

Após a etapa de implementação, todos os mecanismos de segurança avaliados neste trabalho passaram por uma fase de testes utilizando o modelo de testes de caixa preta orientado à entrada e saída para validar as implementações a serem avaliadas. Esse modelo de testes foi escolhido devido a sua eficácia e simplicidade de execução. O modelo de *teste de caixa preta* consiste no fornecimento de dados de entrada para um sistema, os quais serão processados durante a execução do teste e, em seguida, o resultado obtido é comparado a um resultado esperado previamente conhecido (SOMMERVILLE, 2011).

Dessa forma, foram realizados exaustivos testes utilizando valores conhecidos de entradas e saídas correspondentes para validar as implementações dos mecanismos na linguagem *nesC*. Vale ressaltar que essas entradas e saídas correspondentes conhecidas geralmente são disponibilizadas na publicação do mecanismo de segurança e são frequentemente chamados de vetores de teste (*test vectors*).

A indicação de cada mecanismo de segurança e a referência, que contém uma lista de valores conhecidos de entradas e saídas correspondentes utilizados no processo de validação dos mecanismos, estão dispostas na tabela 5.

Tabela 5. Validação dos mecanismos de segurança.

Mecanismo	**Validação**
AES	(DWORKING, 2001)
Skipjack	(NIST, 1998)
Klein	(GONG *et al.*, 2010)
RC5	(RIVEST, 1994)
Present	(BOGDANOV *et al.*, 2007)
CBC, CFB, OFB, CTR	(DWORKING, 2001)
OCB	(ROGAWAY, 2011)
CBCMAC	(DWORKING, 2001)
CMAC	(DWORKING, 2005)

É importante ressaltar que o processo de testes foi dividido em duas etapas, sendo a primeira etapa realizada com o auxílio do simulador TOSSIM, o qual foi utilizado por sua simplicidade e praticidade para executar os testes de caixa preta, e a etapa final realizada diretamente na plataforma MicaZ, a qual é caracterizada na seção 5.5.

O TOSSIM (LEVIS; LEE, 2003; LEVIS *et al.*, 2003) é um simulador de eventos de RSSF que incorpora o modelo de execução do sistema operacional *TinyOS*. Essa ferramenta já vem integrada com a versão 1.x do *TinyOS* e uma vez compilado o código de uma aplicação para um nó sensor real, é possível compilar o mesmo código fonte utilizando uma ferramenta do simulador TOSSIM e executar em um PC.

Além disso, o TOSSIM se apresenta aos desenvolvedores de aplicações de RSSF como uma interessante ferramenta capaz de auxiliar a etapa de desenvolvimento, oferecendo facilidades de depuração do código, a realização de testes e a análise de algoritmos em um ambiente controlado e repetível.

Entretanto, embora o simulador TOSSIM capture um comportamento de baixo nível do sistema operacional *TinyOS*, ele não oferece a capacidade de modelagem precisa de tempo nem de consumo de energia devido a abstrações próprias da arquitetura interna desse simulador de eventos para RSSF, não devendo ser empregado com essa finalidade (LEVIS; LEE, 2003).

5.6.3 Aplicação Utilizada na Avaliação

Para avaliar o desempenho dos mecanismos de segurança foi necessário o desenvolvimento de uma aplicação, chamada *CryptoTest*, utilizando a linguagem *nesC* para a versão mais estável do sistema operacional *TinyOS* (1.x). Além disso, todos os experimentos foram realizados em um nó sensor MicaZ.

Essa aplicação, a qual é descrita no Algoritmo 31, fornece condições para a medição do tempo de execução das operações dos mecanismos por um processo que será descrito mais adiante na seção 5.6.3, além de possibilitar a estimação de memória ROM e RAM requeridos por cada mecanismo. Os mecanismos de segurança a serem utilizados na aplicação podem ser selecionados em tempo de compilação, assim como a variação da carga de trabalho.

No Algoritmo 31, as instruções localizadas entre as linhas 5 e 7 e entre as linhas 12 e 14 determinam as operações de encriptação e autenticação, respectivamente, cujo tempo será

medido. Nesse caso, supondo que o mecanismo a ser avaliado seja uma cifra de bloco, as execuções das instruções 5, 6 e 7 possibilitam a medição do tempo de execução da operação de encriptação da cifra de bloco. Por outro lado, caso um mecanismo de autenticação seja utilizado, a execução das instruções 12, 13 e 14 possibilitam a medição do tempo de execução da operação de autenticação.

Algoritmo 31 *CryptoTest*

Entrada: *textoClaro, um bloco de texto claro; chave, uma chave secreta; e tipo, o tipo de mecanismo de segurança a ser utilizado.*
Saída: *textoCifrado, um bloco criptografado, caso seja utilizada uma cifra de bloco ou um modo de operação; uma tag de autenticação, caso um mecanismo de autenticação seja utilizado; e a alteração lógica do bit 4 da porta E do microcontrolador.*

```
1.  //O mecanismo de segurança é definido em tempo de compilação
2.  Altere o estado do bit 4 da porta E para 0
3.  se tipo = cifra de bloco ou tipo = modo de operação então
4.      inicializaChave(chave);
5.      altere o estado do bit 4 da porta E para 1;
6.      textoCifrado ← criptografaBloco(textoClaro, chave);
7.      altere o estado do bit 4 da porta E para 0;
8.      textoClaro ← decriptografaBloco(textoCifrado, chave);
9.      escreva(textoClaro, textoCifrado);
10.     Devolva textoCifrado;

11. se tipo = MAC então
12.     altere o estado do bit 4 da porta E para 1;
13.     tag ← autenticaBloco(textoClaro, chave);
14.     altere o estado do bit 4 da porta E para 0;
15.     escreva(tag)
16.     Devolva tag;
```

Confidencialidade

Integridade e autenticação

Dessa forma, com uma pequena alteração no Algoritmo 31 (posição das linhas 5 e 7) é possível avaliar as operações de inicialização da chave e de decriptação referentes às cifras de bloco e modos de operação criptográficos de forma análoga à encriptação.

Além disso, ao criptografar, decriptografar ou autenticar uma mensagem, a aplicação *CryptoTest* envia para um microcomputador, via USB, o conteúdo da mensagem, permitindo a depuração durante o desenvolvimento do código. O mecanismo de segurança a ser utilizado na aplicação pode ser selecionado em tempo de compilação, podendo ser escolhido cada um dos mecanismos selecionados para a avaliação de desempenho proposta nesta dissertação.

5.6.4 Quantidade de Memória ROM e RAM

A quantidade de memória ROM requerida por cada algoritmo foi obtida a partir do próprio processo de compilação do código fonte no *TinyOS*, uma vez que ao final do processo de compilação, o *TinyOS* determina a quantidade total de memória ROM requerida pela aplicação compilada. Vale ressaltar que essa indicação de memória ROM é geralmente fornecida por ambientes utilizados no desenvolvimento de sistemas embarcados, como, por exemplo, o MPLAB (MICROCHIP, 2011).

Por outro lado, a medição da quantidade de memória RAM requerida é uma tarefa mais complicada por causa da sua característica dinâmica. Uma maneira típica empregada para realizar medições da quantidade de memória RAM em uso é utilizar um depurador de tempo real juntamente com uma configuração de *breakpoints* disponibilizados no código fonte da aplicação (CASADO; TSIGAS, 2009). Entretanto, não há ferramenta disponível capaz de fornecer essa funcionalidade específico para a plataforma MicaZ, a qual foi utilizada na realização dos experimentos.

Diante desse contexto, da mesma forma que a quantidade de memória ROM, o compilador do sistema operacional *TinyOS* também oferece uma estimativa correta da quantidade memória RAM requerida pelas aplicações (TINYOS DOCUMENTATION, 2010). Portanto, essa foi a estratégia adotada para realizar as medições de utilização de memória RAM.

O resultado do processo de compilação com a indicação das quantidades de memória RAM e memória ROM requeridas é ilustrado na Figura 9.

Figura 9. Ilustração do processo de compilação, destacando a quantidade de memória ROM e RAM requerida pela aplicação.

5.6.5 Tempo de Execução

As medições do tempo de execução foram realizadas com o auxílio de um osciloscópio digital *Tektronix TDS 210* (TEKTRONIX, 1999)[8]. Conforme pode ser observado no esquema mostrado na figura 10, um pino do microcontrolador *ATMega128L* da plataforma MicaZ foi conectado a um canal do osciloscópio para monitorar o seu nível lógico. Embora tenha sido escolhido o pino E4 do microcontrolador para realizar a conexão com o osciloscópio, vale ressaltar que é possível utilizar qualquer pino de entrada/saída que não esteja sendo empregado para outra função.

Para medir o tempo de execução, o nível lógico desse pino foi alterado de 0 para 1 antes e de 1 para 0 após cada operação realizada pelo mecanismo de segurança, conforme o Algoritmo 31. Por exemplo, considere que a aplicação utilize o CMAC para autenticar uma determinada quantidade de *bytes*. Antes do processo de autenticação ser iniciado pela aplicação, ou seja, antes do algoritmo CMAC ser executado, o nível lógico do pino E4 é alterado de 0 para 1. Assim, depois de realizada a operação de autenticação, o nível lógico do pino E4 é novamente alterado de 1 para 0.

Dessa forma, o tempo total no qual o pino permanece com o nível lógico 1 representa o tempo de execução de cada operação que pode ser medido no osciloscópio. No caso do exemplo citado anteriormente, o tempo durante o qual o pino E4 permanece com estado lógico 1 representa o tempo de execução da operação de autenticação do código de autenticação de mensagens CMAC.

Após a realização do procedimento mostrado no esquema da Figura 10, a medição do tempo de execução pode ser realizada a partir do resultado que pode ser observado no osciloscópio, conforme a ilustração real apresentada na Figura 11.

É bom lembrar ainda que a conexão USB entre a plataforma MicaZ e um PC permitiu o processo de depuração na etapa de implementação, o que permitiu a captura de dados a serem analisados, assim como a realização da segunda etapa de validação das implementações dos mecanismos de segurança, conforme indicado na subseção 5.2.3.

[8] Todos os experimentos realizados com o osciloscópio *Tektronix TDS 210* para a medição de tempo foram realizados no Laboratório de Microcontroladores (LABOMICRO) localizado no Instituto Federal de Educação, Ciência e Tecnologia do Ceará (IFCE).

Na última etapa de validação mencionada anteriormente, foi utilizada uma ferramenta de monitoração serial utilizada para realizar a comunicação entre a plataforma MicaZ e o PC chamada *Docklight v.9.1.21* (DOCKLIGHT, 2011).

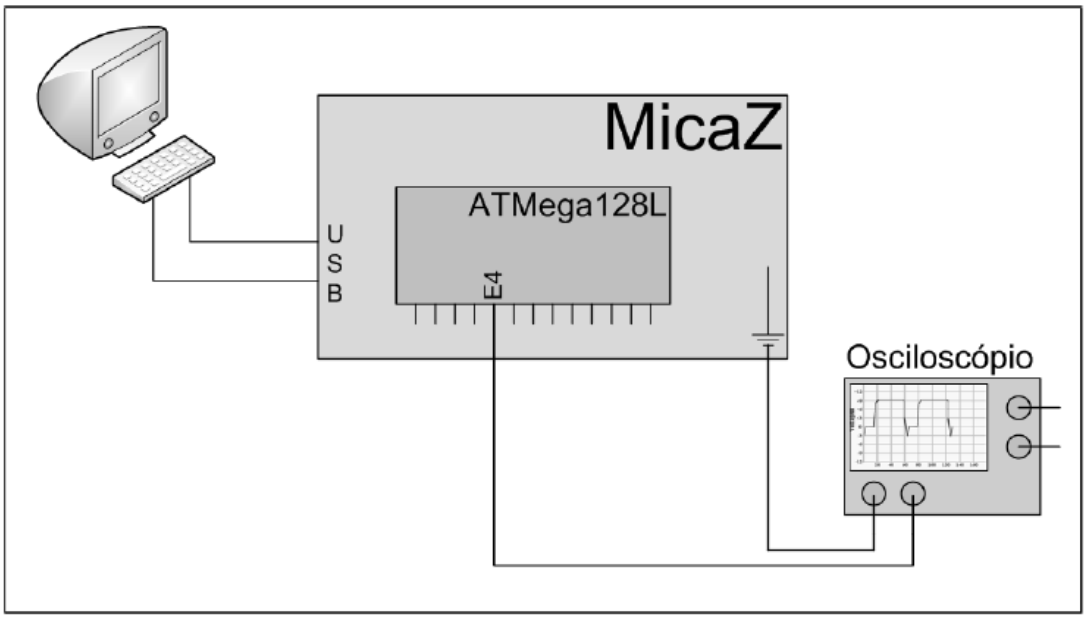

Figura 10. Esquema de medição do tempo de execução.

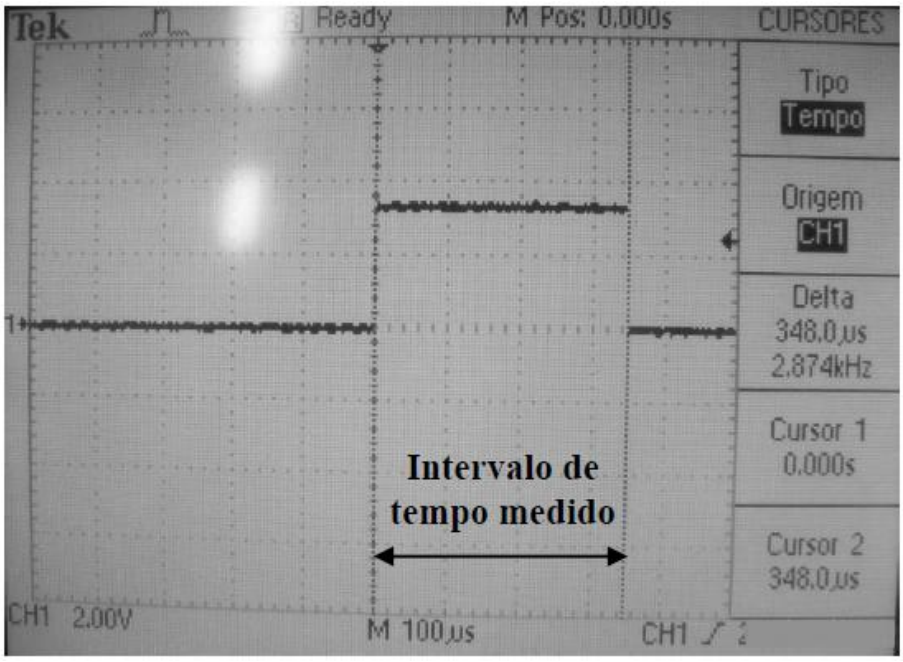

Figura 11. Imagem ilustrativa da medição do tempo de execução no osciloscópio.

5.6.6 Consumo de Energia

O consumo de energia foi obtido a partir das Equações 1 e 2. Os valores de tensão da bateria e de corrente de operação foram obtidos a partir do documento de especificação da plataforma MicaZ (*datasheet*), que são 3V e 8mA, respectivamente.

$$Potência = tensãoBateria * correnteOperação \quad (1)$$

$$energiaConsumida = Potência * tempoExecução \quad (2)$$

Assim, a potência foi calculada, utilizando-se a Equação 1 da seguinte forma:

$$Potência = 3*8*10\text{-}3 = 24\ mW$$

A partir da potência calculada, a quantidade de energia requerida por cada algoritmo foi obtida de acordo com a Equação 2. Ressalta-se que, utilizando essa estratégia de medição, obtém-se a energia gasta por cada algoritmo de forma aproximada, uma vez que não são utilizados os valores reais de tensão e corrente. Entretanto, Lee *et al.*, (2010) verificaram em seus trabalhos que a diferença entre os valores reais e teóricos não é significativa e, portanto, a comparação do desempenho dos algoritmos estudados neste trabalho não é afetada.

Vale ressaltar que esse método fornece medições precisas uma vez que o controle do pino do microcontrolador da plataforma MicaZ não afeta significativamente a latência do microcontrolador ou o consumo de energia da plataforma.

5.7 Análise e Apresentação dos Resultados (f)

Nesta seção são apresentados e discutidos os resultados obtidos após a realização de experimentos reais com relação às métricas quantidade de memória requerida, tempo de execução e consumo de energia, relacionados a cada mecanismo de segurança estudado nesta dissertação.

5.7.1 Quantidade de Memória ROM e RAM

Os resultados relacionados à quantidade de memória requerida são apresentados nas figuras 12, 13, 14, 15, 16 e 17 e comentados nas próximas subseções. Vale ressaltar que os valores levam em consideração também a aplicação desenvolvida para realizar as medições, o que não afeta a comparação, pois o único fator diferente na aplicação consiste apenas no mecanismo de segurança a ser utilizado.

5.7.1.1 Cifras de Bloco

As quantidades de memória ROM e RAM requeridas por cada cifra de bloco são apresentadas nas figuras 12 e 13, respectivamente. Conforme mostrado na figura 12, a cifra de bloco Skipjack é a cifra que requer a menor quantidade de memória ROM, enquanto a cifra AES requer a maior quantidade com relação às demais cifras, o que já era esperado uma vez que o AES é a cifra mais complexa e a que requer a maior quantidade de dados (tabelas Sbox) a serem armazenados.

Já as cifras Klein, RC5 e Present apresentaram uma quantidade cerca de 7%, 8% e 26% mais ROM que o Skipjack, respectivamente, enquanto que a quantidade requerida pelo AES foi 64% maior que a quantidade requerida pelo Skipjack.

Por outro lado, conforme pode ser observado na figura 13, o RC5 foi o algoritmo que apresentou a menor quantidade requerida de memória RAM, seguido pelo Skipjack, que apresentou uma quantidade cerca de 80% maior, pelo Klein, que apresentou uma quantidade mais de duas vezes maior do que a quantidade requerida pelo RC5 e pelo Present, que apresentou uma quantidade requerida mais de quatro vezes maior que a quantidade requerida pelo RC5.

Já o AES apresentou mais uma vez o pior desempenho com relação às demais cifras, uma vez que a sua quantidade de memória RAM requerida foi mais de sete vezes maior que a quantidade requerida pelo RC5, representando quase 58% do total disponível na MicaZ (4KB).

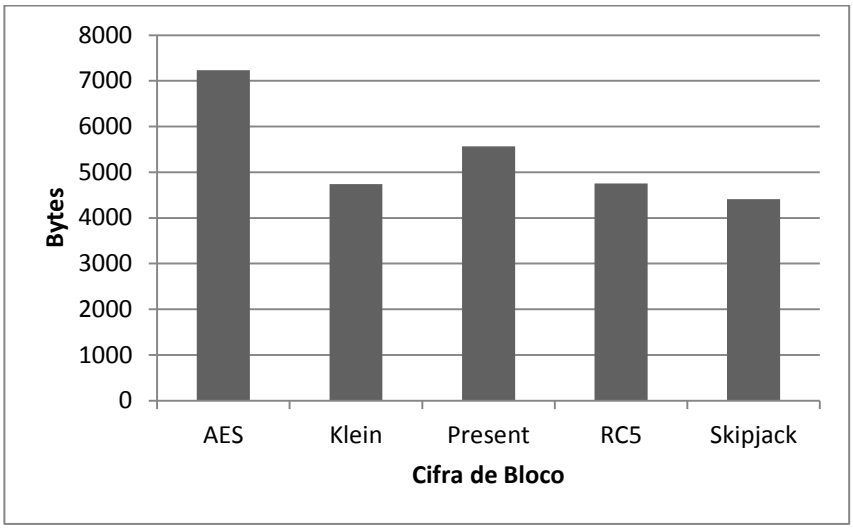

Figura 12. Quantidade de memória ROM requerida pelas cifras de bloco.

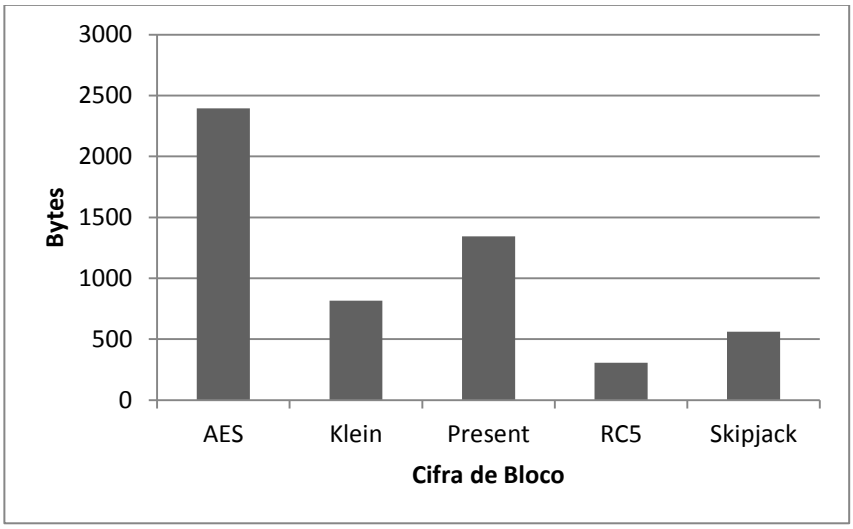

Figura 13. Quantidade de memória RAM requerida pelas cifras de bloco.

5.7.1.2 Modos de Operação

As figuras 14 e 15 apresentam a quantidade de memória ROM e RAM requeridas pelos modos de operação utilizando a cifra AES-128 como cifra de bloco adjacente, respectivamente. O AES-128 foi escolhido para realizar a avaliação dos modos de operação devido à fácil disponibilidade na literatura de vetores de teste dessa cifra utilizando cada um dos modos avaliados neste trabalho, o que facilitou o processo de validação dos mecanismos conforme apresentado na subseção 5.2.3.

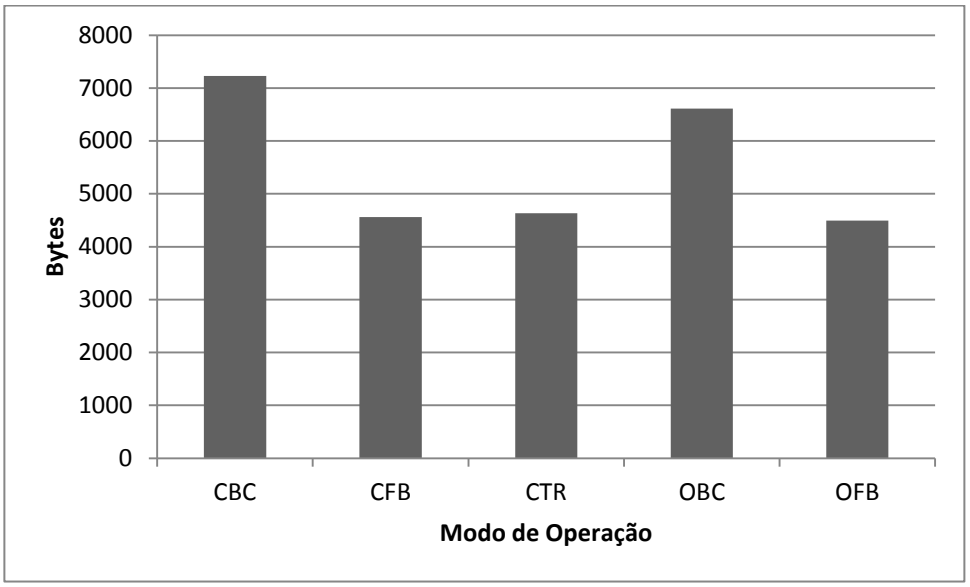

Figura 14. Quantidade de memória ROM requerida pelos modos de operação.

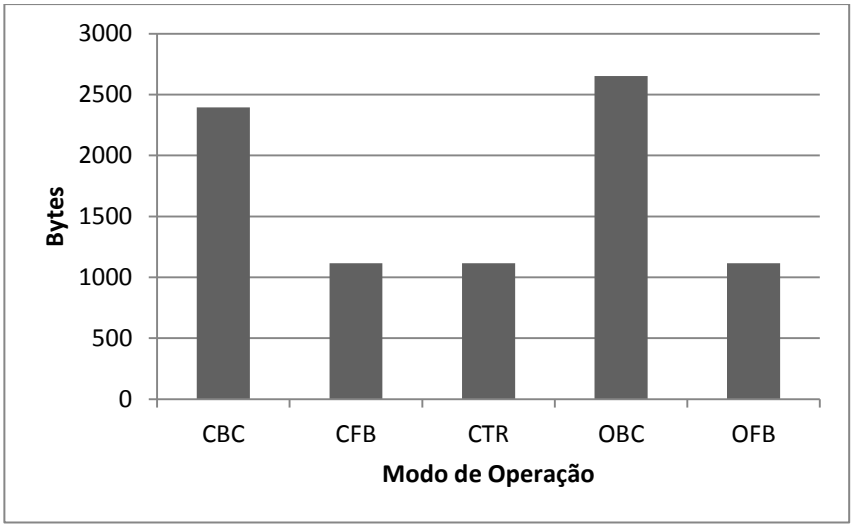

Figura 15. Quantidade de memória RAM requerida pelos modos de operação.

Conforme mostrado na figura 14, os modos de operação CTR, OFB e CFB apresentaram resultados muito próximos com relação à quantidade de memória ROM requerida, com uma ligeira vantagem do modo OFB, o que pode ser explicado pela necessidade de utilização de apenas do algoritmo de encriptação tanto para criptografar quanto para decriptografar os dados, conforme indicado na subseção 4.3.6.

Já os modos CBC e o OCB apresentaram uma quantidade requerida em torno de 60% e 47% maiores do que a quantidade requerida pelo modo OFB, respectivamente. Além disso, a quantidade requerida pelo modo CBC, que apresentou o pior desempenho dentre os demais modos com relação a memória ROM, representou cerca de 6% do total disponível na plataforma MicaZ (128KB).

Por outro lado, com relação ao consumo de memória RAM, os modos CTR, OFB e CFB também apresentaram resultados bastante próximos. Entretanto, o modo CTR foi o que apresentou a menor quantidade requerida.

Já os modos CBC e OCB apresentaram quantidades requeridas de memória RAM em torno de 2,1 e 2,3 vezes maiores do que a quantidade requerida pelo modo CTR, respectivamente. Além disso, a quantidade requerida pelo modo OCB, que apresentou o pior desempenho dentre os demais modos com relação a memória RAM, representou cerca de 6% do total disponível na plataforma MicaZ (128KB).

5.7.1.3 Códigos de Autenticação de Mensagens

As quantidades de memória ROM e RAM requeridas por cada mecanismo de autenticação são apresentadas nas figuras 16 e 17, respectivamente. Vale ressaltar que os mecanismos CBCMAC e CMAC foram avaliados utilizando a cifra AES-128 como cifra de bloco adjacente pelo mesmo motivo pelo qual ele foi utilizado na avaliação dos modos de operação (ver subseção 5.7.1.2).

Além disso, o modo OCB foi incluído na avaliação dos mecanismos de autenticação devido a sua característica de projeto que permite além de criptografar, realizar a operação de autenticação, conforme mencionado no Capítulo 4.

De acordo com a figura 16, o CBCMAC foi o MAC que apresentou a menor quantidade de memória ROM requerida, seguido pelos mecanismos CMAC e OCB, que apresentaram quantidades de memória ROM requeridas cerca de 9% e 63% maiores do que a quantidade requerida pelo CBCMAC.

Por outro lado, o algoritmo HMAC, utilizando o SHA1 como função *hash* adjacente, foi o MAC que apresentou o pior desempenho em termos de memória ROM, requerendo uma

quantidade significativamente superior aos demais algoritmos, representando em torno de 21% do total disponível na MicaZ (128KB).

Já o algoritmo HMAC-MD5 apresentou um desempenho melhor que o HMAC-SHA1, requerendo cerca de duas vezes menos memória ROM que o HMAC-SHA1. Entretanto, o HMAC-MD5 apresentou uma quantidade de memória ROM requerida 2,7 vezes maior que o mecanismo CBCMAC.

Com relação à memória RAM, conforme mostrado na figura 17, o modo OCB foi o algoritmo que apresentou a maior quantidade requerida, representando uma quantidade em torno de 64% do total de memória RAM disponível na plataforma MicaZ (4KB), enquanto que o HMAC-MD5 e o HMAC-MD5 foram os MACs que apresentaram o melhor desempenho dentre os demais mecanismos de autenticação, requerendo quantidades de memórias RAM muito próximas, com ligeira vantagem para o HMAC-MD5.

Já os algoritmos CBCMAC e CMAC apresentaram uma quantidade praticamente igual de memória RAM, requerendo uma quantidade cerca de cinco vezes maior do que a quantidade requerida pelo HMAC-MD5.

A quantidade de memória RAM requerida pelo mecanismo OCB girou em torno de 13,7 vezes maior do que a quantidade requerida pelo HMAC-MD5, o qual apresentou a menor quantidade.

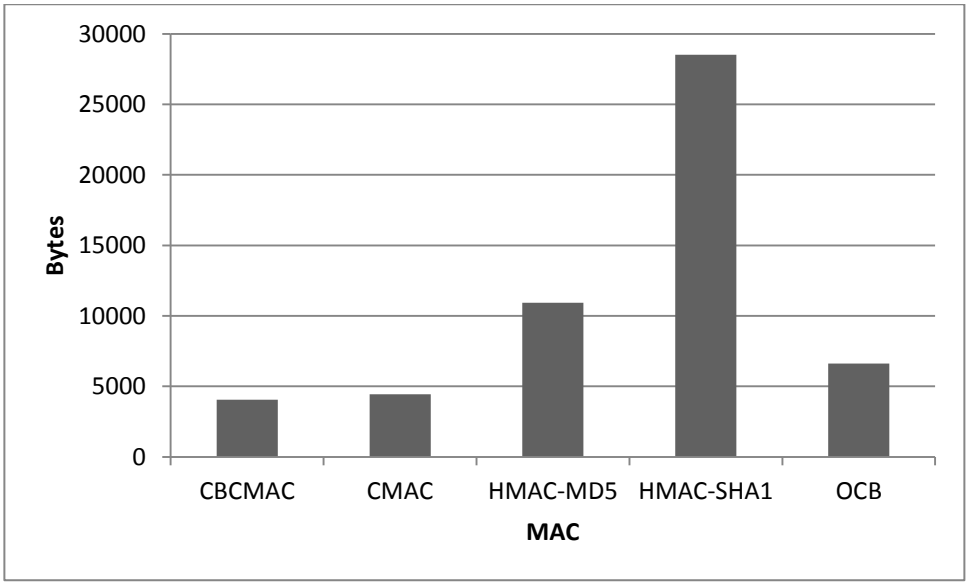

Figura 16. Quantidade de memória ROM requerida pelos MACs.

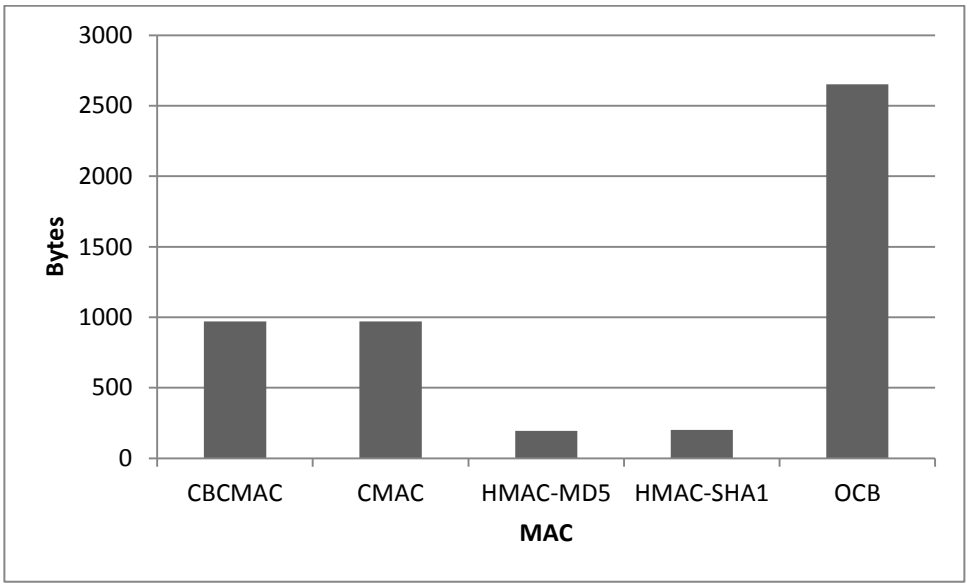

Figura 17. Quantidade de memória RAM requerida pelos MACs.

5.7.2 Tempo de Execução

Os resultados relacionados à métrica tempo de execução dos mecanismos são apresentados nas figuras 18, 19, 20 e 21, e comentados nas próximas subseções. Os gráficos apresentam a média de tempo de processamento a partir das 31 medições com intervalo de confiança de 95%. Além disso, devido à pequena magnitude do intervalo de confiança, alguns intervalos não são visíveis nos gráficos.

5.7.2.1 Cifras de Bloco

O tempo de execução de cada cifra de bloco nas operações de inicialização da chave criptográfica, encriptação e decriptação de dados são mostrados nas figuras 18 e 19, respectivamente. Vale ressaltar que as cifras de bloco foram avaliadas utilizando-se um mesmo modo de operação (CBC).

Conforme mostrado na figura 18, a cifra Skipjack foi a que apresentou o menor tempo de execução durante a etapa de inicialização da chave criptográfica, enquanto que o RC5 apresentou

o pior desempenho dentre os demais algoritmos. O Skipjack levou em torno de 0,18 (± 0,014) milissegundos para inicializar a chave, enquanto que o RC5 apresentou um tempo de execução mais de 17 vezes maior do que o tempo necessário para o Skipjack realizar essa operação.

Com relação às demais cifras de bloco, o AES, apresentou um tempo de execução para inicializar a chave criptográfica 55% maior do que o tempo levado pelo Skipjack para realizar essa operação. Além disso, o tempo de execução das cifras Klein e Present foram 2,7 e 11,7 vezes maiores do que o tempo de execução apresentado pelo Skipjack.

De acordo com a figura 19, o AES foi o algoritmo que apresentou o pior desempenho em termos de tempo de execução tanto nas operações de encriptação quanto nas operações de decriptação, enquanto que os algoritmos Skipjack e RC5 apresentaram os melhores de desempenhos, com ligeira vantagem para o Skipjack.

Vale ressaltar que o tempo de execução nas operações de encriptação e decriptação do Skipjack foram praticamente idênticos. Assim, apenas o tempo de encriptação é mostrado no gráfico da figura 18. De forma semelhante, apenas a operação de encriptação do algoritmo Present é mostrada no gráfico devido a suas características de projeto mencionadas no Capítulo 4.

Com relação às demais cifras, o Klein levou em torno de 4,86 milissegundos para criptografar um bloco de 64 *bytes* e 5,82 milissegundos para decriptografar um bloco de mesmo tamanho, representando 46% e 57% mais tempo do que o RC5 para realizar essas operações, respectivamente. Já o Present apresentou um desempenho superior apenas à cifra AES, levando aproximadamente 7,09 milissegundos para criptografar um bloco de 64 *bytes*.

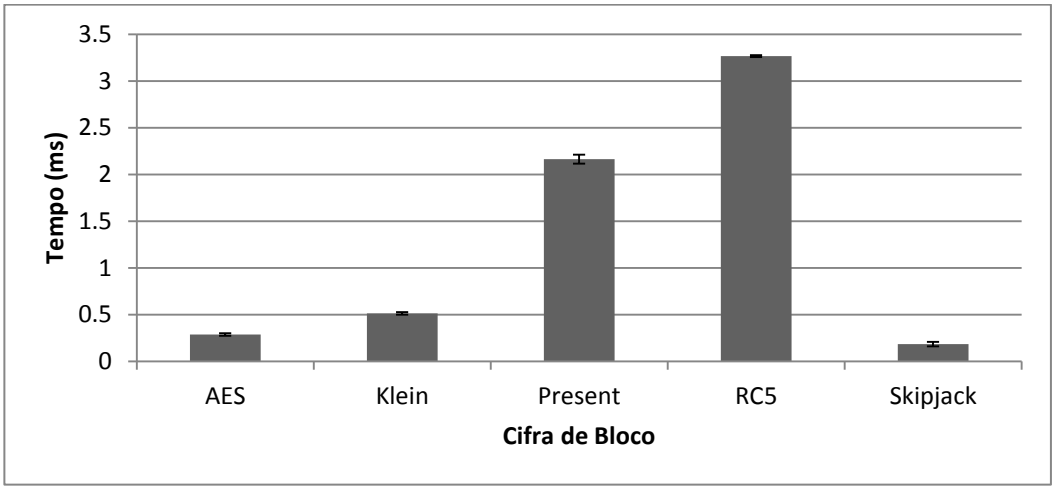

Figura 18. Tempo de execução na inicialização da chave criptográfica com intervalo de confiança de 95%.

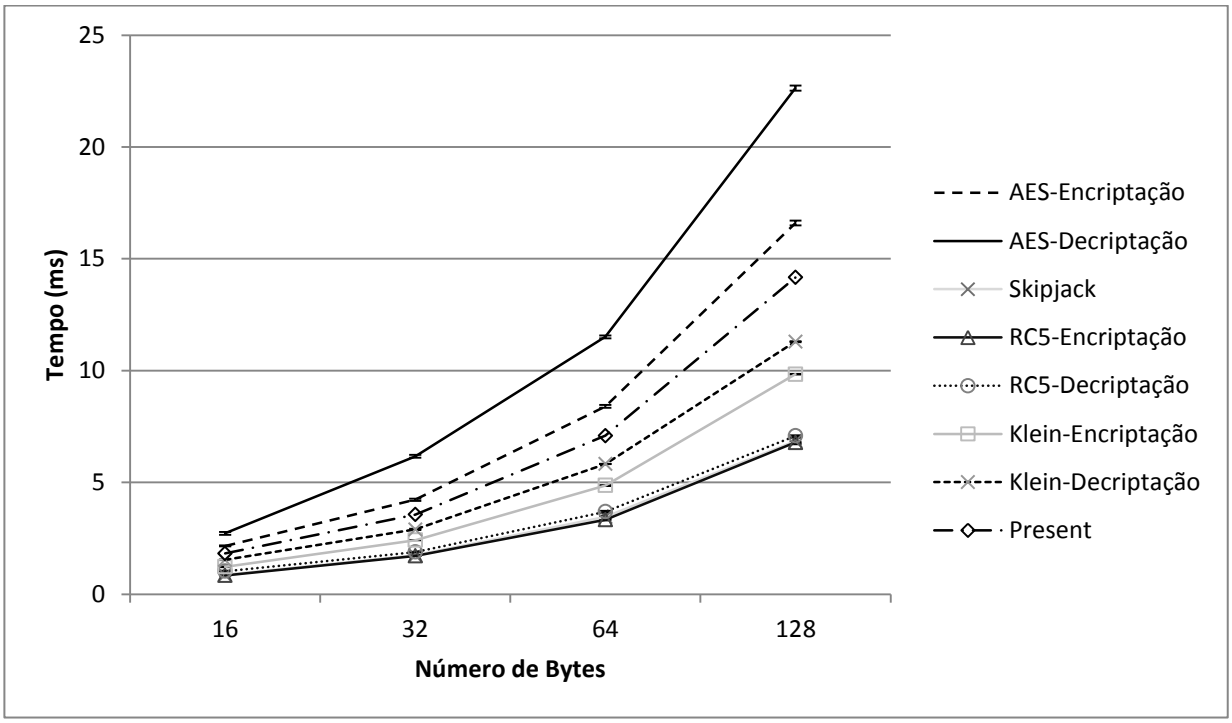

Figura 19. Tempo de execução das cifras de blocos no modo CBC com intervalo de confiança de 95%.

5.7.2.2 Modos de Operação

Os tempos de execução de cada modo de operação para diferentes tamanhos de bloco de entrada são mostrados nas figuras 20. Vale ressaltar que os modos OFB, CTR e CFB necessitam apenas do algoritmo de encriptação para realizar as operações de encriptação de decriptação dos dados, conforme visto Capítulo 4.

Conforme mostrado na figura 20, o CBC e o OCB foram os modos de operação que apresentaram os piores desempenhos em termos de tempo de execução, sendo que o OCB levou vantagem na operação de maiores quantidades de *bytes* com relação ao modo CBC, que apresentou melhor desempenho que o OCB para operar pequenas quantidades de *bytes*.

Por outro lado, os modos CFB, OFB e CTR apresentaram os melhores desempenhos com relação ao tempo de execução, sendo que o CTR foi o que apresentou o menor valor de tempo de execução em todos os experimentos.

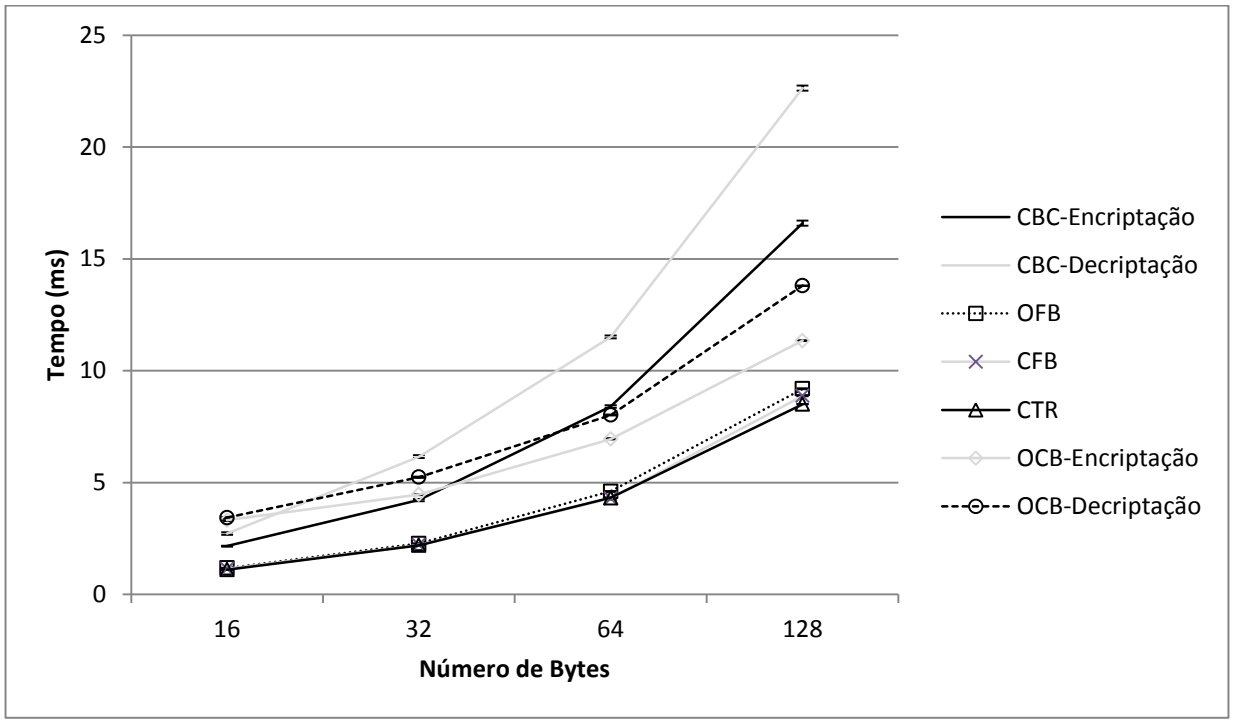

Figura 20. Tempo de execução dos modos de operação com a cifra AES-128 com intervalo de confiança de 95%.

Por exemplo, o CTR levou em torno de 1,11 milissegundos para criptografar um bloco de 16 *bytes*, enquanto que o CFB levou 1,15 milissegundos, o OFB levou 1,18 milissegundos, o CBC levou 2,15 milissegundos e o OCB levou 3,31 milissegundos.

5.7.2.3 Códigos de Autenticação de Mensagens

Os tempos de execução de cada mecanismo de autenticação para diferentes tamanhos de bloco de entrada são mostrados nas figuras 21.

Conforme pode ser observado na figura 21, o CBCMAC foi mecanismo de autenticação que apresentou o menor tempo de execução para autenticar os diferentes tamanhos de bloco avaliados, em contraste com o HMAC-SHA1 que apresentou o pior desempenho dentre os demais mecanismos.

Por exemplo, para autenticar um bloco de 64 *bytes*, o CBCMAC levou em torno de 4,43 milissegundos, enquanto que o CMAC, OCB e HMAC-MD5 levaram quantidades 31%, 56% e 3 vezes maiores do que a quantidade de tempo levada pelo CBCMAC para autenticar um bloco de mesmo tamanho, respectivamente.

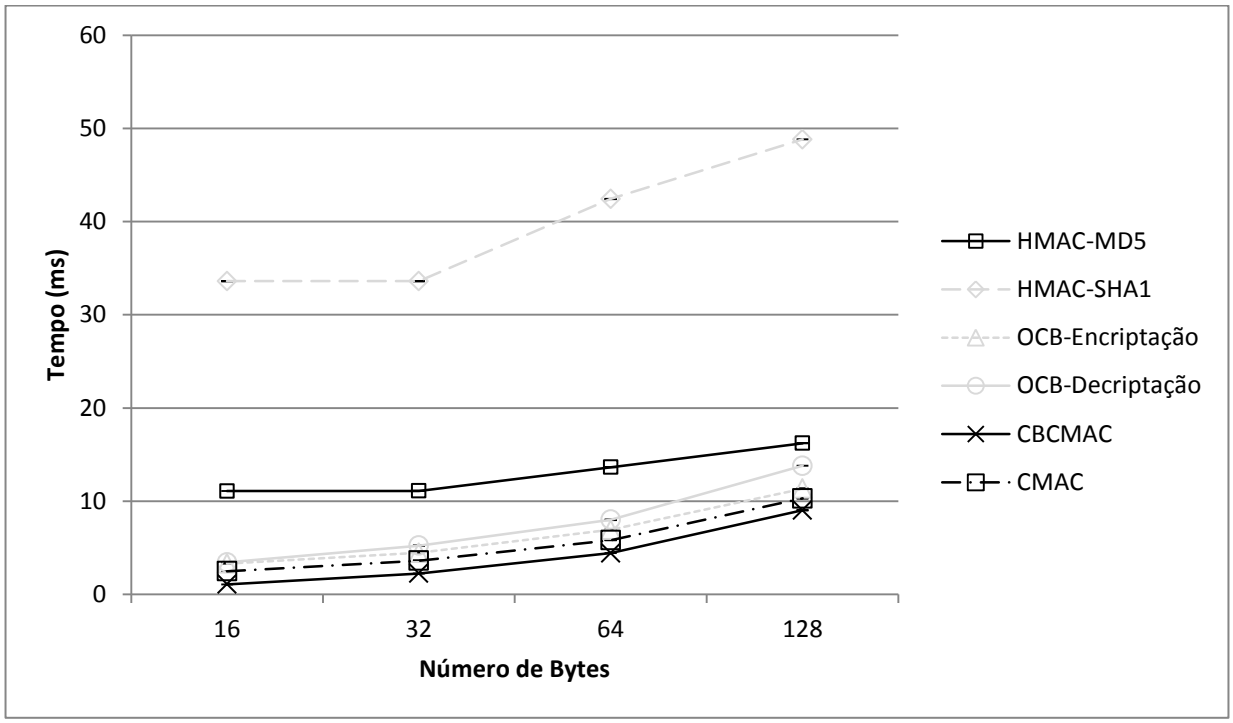

Figura 21. Tempo de execução dos códigos de autenticação de mensagens com intervalo de confiança de 95%.

5.7.3 Consumo de Energia

Os resultados relacionados à métrica de consumo de energia dos mecanismos são apresentados nas figuras 22, 23, 24 e 25, e comentados nas próximas subseções.

Conforme mencionado anteriormente, o consumo de energia foi obtido a partir do tempo de execução dos algoritmos e através da aplicação da Equação 2, uma vez que foram utilizados valores teóricos para o cálculo da potência. Os gráficos apresentam a média dos valores de energia calculados a partir das 31 medições de tempo de processamento com intervalo de confiança de 95%. Devido à pequena magnitude do intervalo de confiança, alguns intervalos não são visíveis nos gráficos.

5.7.3.1 Cifras de Bloco

O consumo de energia para realizar a inicialização da chave criptográfica dos algoritmos criptográficos está mostrado na figura 22 com intervalo de confiança de 95%.

Como mostrado na figura 22, o RC5 apresentou um consumo de energia significativamente maior que as demais cifras na inicialização da chave, embora essa etapa não seja determinante, uma vez que é realizada apenas uma vez na inicialização da aplicação. Por outro lado, o Skipjack, AES, Klein e o Present foram as cifras que apresentaram os menores valores, nessa ordem.

Já na figura 23, é possível observar que o AES é o algoritmo criptográfico que apresentou o pior desempenho em termos de consumo de energia tanto na encriptação quanto na decriptação de diferentes tamanhos de blocos, uma vez que consumiu a maior quantidade de energia comparando com os demais algoritmos, o que era esperado por ser o mais complexo.

Por outro lado, o Present apresentou um consumo intermediário entre o AES e as demais cifras nas operações de encriptação e decriptação. Já o Skipjack e o RC5 foram os algoritmos que apresentaram o melhor desempenho em termos de consumo de energia.

O algoritmo Skipjack apresentou praticamente o mesmo desempenho nas duas operações e, por isso, apenas uma operação é mostrada na figura 23. Já o RC5 apresentou um consumo cerca de 1% menor do que o Skipjack na encriptação e cerca de 4% maior do que o Skipjack na decriptação. Dessa forma, o Skipjack levou ligeira vantagem sobre o RC5 no geral.

Com relação às demais cifras, para criptografar e decriptografar um bloco de 128 *bytes*, por exemplo, a cifra Klein requereu em torno de 43% e 66% a mais que o Skipjack, respectivamente. Já o Present apresentou um consumo duas vezes maior de energia que o algoritmo Skipjack.

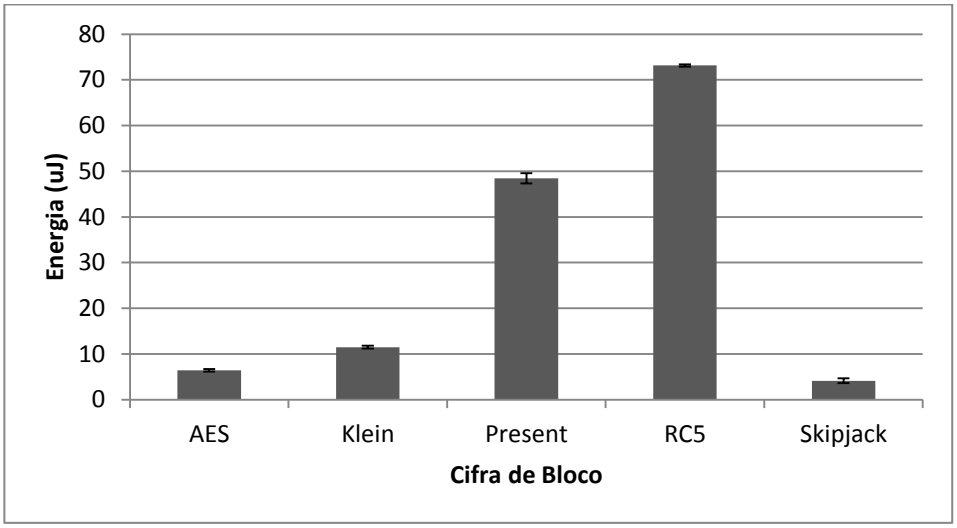

Figura 22. Consumo de energia na inicialização da chave criptográfica com intervalo de confiança de 95%.

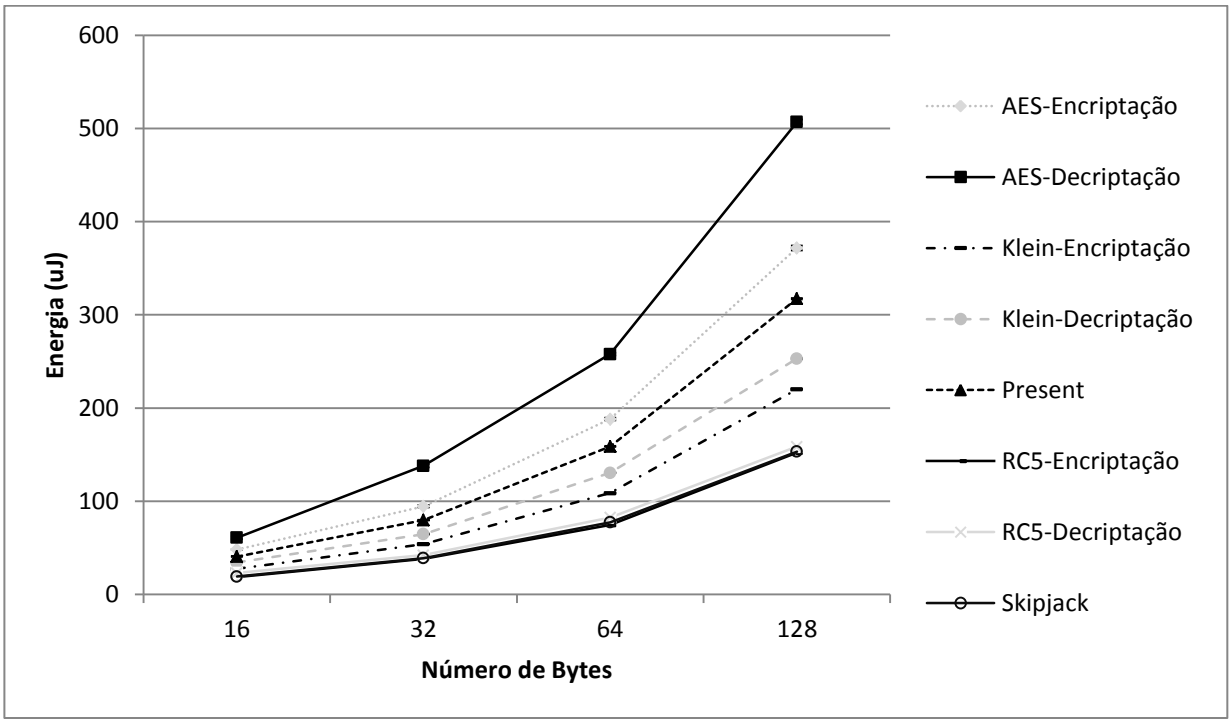

Figura 23. Consumo de energia das cifras de blocos no modo CBC com intervalo de confiança de 95%.

5.7.3.2 Modos de Operação

O consumo de energia relacionado a cada modo de operação para diferentes tamanhos de bloco de entrada são mostrados na figura 24 com intervalo de confiança de 95%, ressaltando que a cifra AES-128 foi a cifra de bloco adjacente utilizada pelos mesmos motivos apresentados na subseção 5.7.1.2.

Conforme pode ser observado na figura 24 o CBC e o OCB foram os modos que apresentaram os maiores consumos de energia, nessa ordem, sendo que o OCB menos energia para criptografar e decriptografar quantidades maiores quantidades de *bytes* com relação ao modo CBC, que apresentou melhor desempenho que o OCB para operar pequenas quantidades de *bytes*.

Por outro lado, os modos OFB, o CFB e o CTR apresentaram desempenhos bastante próximos, ressaltando que esses três modos utilizam basicamente o mesmo algoritmo para criptografar e decriptografar as mensagens e, por isso, apenas uma operação é mostrada na figura 23.

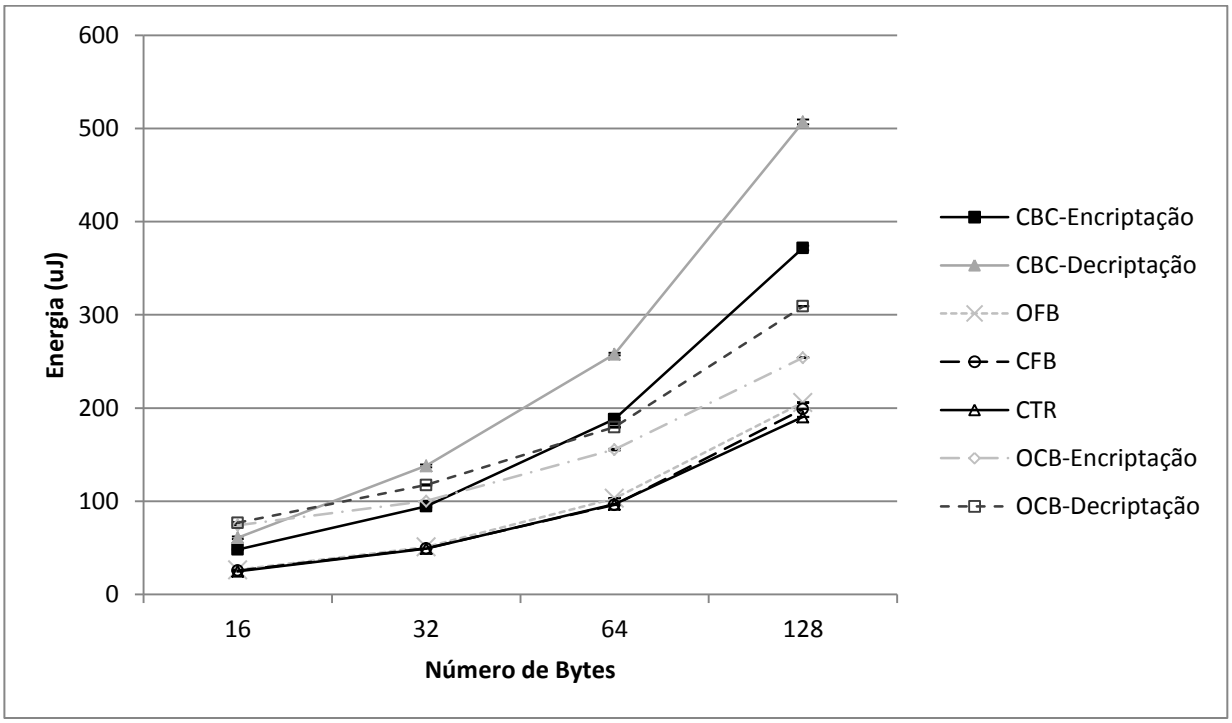

Figura 24. Consumo de energia dos modos de operação com o AES-128 com intervalo de confiança de 95%.

Embora os modos CFB, OFB e CTR tenham apresentado resultados próximos, o CTR levou ligeira vantagem sobre os outros dois, sendo o modo que apresentou o menor consumo de energia. Para criptografar um bloco de 128 *bytes*, por exemplo, o CTR apresentou desempenho 4% e 8% melhor do que o desempenho apresentado pelos modos CFB e OFB, respectivamente.

5.7.3.3 Códigos de Autenticação de Mensagens

O consumo de energia relacionado aos mecanismos de autenticação para autenticar diferentes tamanhos de blocos está mostrado na figura 25 com intervalo de confiança de 95%.

Conforme mostrado na figura 25, os algoritmos de autenticação CBCMAC e CMAC apresentaram os menores consumos de energia, nessa ordem. Já o OCB, também considerando uma mensagem de 128 *bytes*, consumiu em torno de 26% mais energia para criptografar e autenticar a mensagem e em torno de 53% mais energia para decriptografar e verificar a *tag* de autenticação do que o algoritmo CBCMAC.

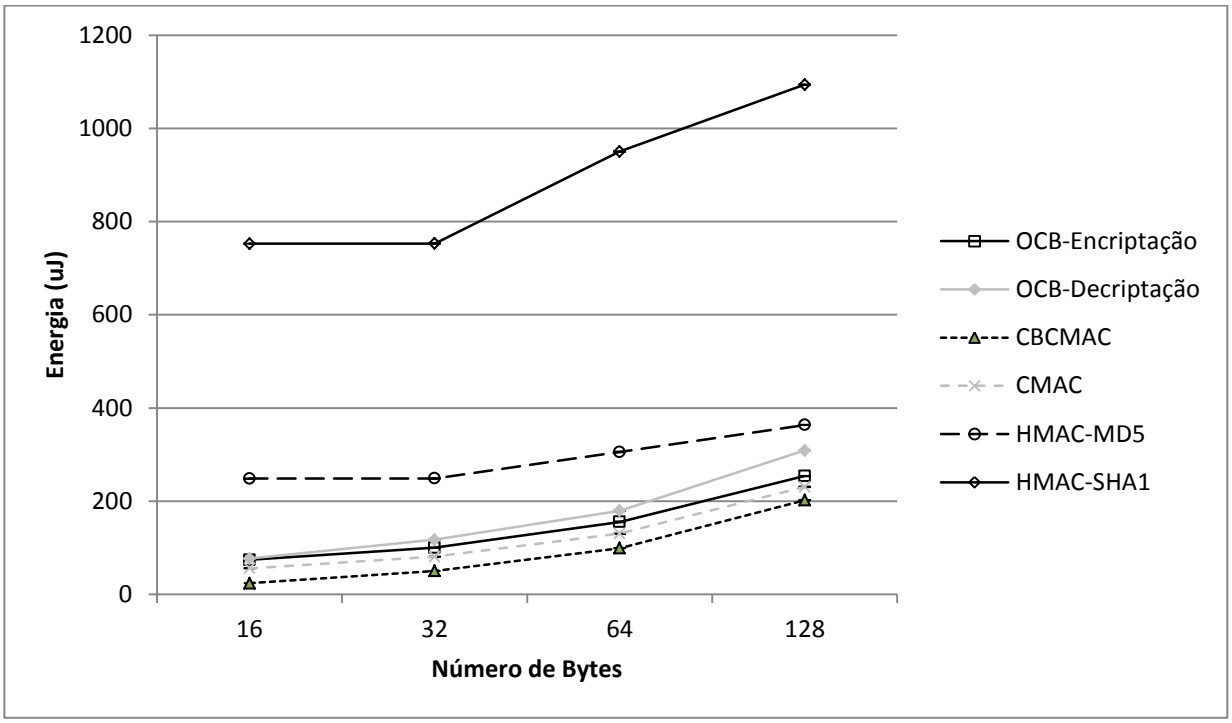

Figura 25. Consumo de energia dos códigos de autenticação de mensagens com intervalo de confiança de 95%.

Por outro lado, o HMAC-MD5 apresentou desempenho inferior aos apresentados pelos mecanismos CBCMAC, OCB e CMAC, consumindo em torno de 79%, 43% e 57% mais energia para autenticar um bloco de 128 *bytes*, respectivamente. Além disso, vale ressaltar que o HMAC-MD5 foi o algoritmo que apresentou a menor variação em decorrência do aumento do número de *bytes* a ser autenticados.

O HMAC-SHA1 apresentou consumo significativamente maior que os demais algoritmos. Por exemplo, para criptografar um bloco de 128 *bytes*, ele apresentou consumos em torno de 3 vezes, 4,3 vezes, 4,7 vezes e 5,4 vezes maiores do que a quantidade de energia consumida pelo CBCMAC para autenticar uma mensagem de 128 *bytes*.

5.7.4 Discussão dos Resultados

Os resultados obtidos neste trabalho podem contribuir para a escolha apropriada de mecanismos de segurança a serem empregados em aplicações de RSSF, bem como no desenvolvimento e evolução de protocolos de segurança para tais redes.

Primeiramente, com relação aos algoritmos de criptografia, o RC5 (avaliado com a configuração RC5 64/18/16) mostrou-se como a melhor opção para uso em RSSF devido ao fato que, além de apresentar um alto nível de segurança, ele apresentou um desempenho muito próximo do melhor desempenho, apresentado pelo algoritmo Skipjack. Este último deve ser evitado, pois, apesar de apresentar o melhor desempenho entre as demais cifras de bloco, oferece um nível reduzido de segurança, conforme mencionado no Capítulo 4.

Entretanto, devido ao fato do RC5 possuir registro de patente, motivo pelo seu uso a longo prazo é dificultado, o algoritmo Klein pode ser uma solução atrativa para aplicações de RSSF, uma vez que ele é um algoritmo de domínio público e apresentou um desempenho intermediário entre o RC5 e o Skipjack com relação aos demais algoritmos.

Com relação aos modos de operação, os modos CTR, CFB e OFB apresentaram os melhores desempenhos tanto em termos de memória quanto de consumo de energia. No caso do CFB, como ele permite o reuso do *IV*, ele pode ser uma boa opção para RSSF. Entretanto, o uso do CTR de forma apropriada pode ser a melhor opção para entre os demais modos, uma vez que ele foi o modo que apresentou o melhor desempenho.

Por outro lado, embora o OCB apresente um consumo de energia superior ao CTR e CFB, ele é uma solução atrativa quando se deseja confidencialidade e autenticação ao mesmo tempo. Entretanto, vale lembrar que o OCB possui registro de patente e seu uso é condicionado à concessão de licença.

Com relação aos mecanismos de autenticação, o CBCMAC e o CMAC apresentaram os melhores desempenhos entre os algoritmos de autenticação avaliados neste trabalho, sendo os mais recomendados para RSSF.

No caso dos algoritmos HMAC-SHA1 e HMAC-MD5, embora eles sejam bastante eficiente em plataformas de 32 *bits*, esses algoritmos apresentaram desempenho bastante inferior aos demais na plataforma MicaZ, que possui arquitetura de 8 *bits*. Entretanto, o HMAC-MD5 pode ser uma solução atrativa para as novas gerações de sensores inteligentes em um futuro próximo, devido ao fato de que seu consumo de energia é menos afetado do que os demais com relação à variação do tamanho do bloco de entrada a ser autenticado.

Com relação aos trabalhos relacionados indicados no Capítulo 4, esta dissertação apresentou alguns diferenciais, os quais são mencionados a seguir.

Em contrapartida dos trabalhos de Guimarães et al. (2005), esta proposta avaliou uma configuração do RC5 de 18 rounds que apresenta alto nível de segurança e realizou os experimentos na plataforma MicaZ.

Ao contrário do trabalho de Law *et al.* (2006), Grobschadl et al. (2007), Casado e Tsigas (2009) e Szalachowski *et al.* (2010), esta proposta, além de avaliar outros mecanismos de segurança, utilizou uma plataforma popular (MicaZ) no contexto atual de RSSF.

Diferentemente do trabalho de Bauer et al. (2009), esta proposta, além de avaliar o OCB e diferentes mecanismos dedicados de autenticação, considerou uma maior faixa de tamanho de entrada para permitir uma conclusão mais precisa acerca do desempenho dos mecanismos avaliados.

Diferentemente do trabalho de Lee et al. (2010), esta proposta definiu precisamente as medidas estatísticas utilizadas e avaliou outras cifras de blocos além do Skipjack, RC5 e o AES. Além disso, os resultados desta dissertação apresentam diferenças significativas com relação aos resultados obtidos nos experimentos realizados por Lee et al. (2010). Entretanto, os detalhes sobre o processo de medições e apresentação dos resultados em (LEE *et al.*, 2010) não são suficientes para justificar as diferenças encontradas nesta dissertação com precisão.

Com relação às diferenças com relação aos resultados obtidos em (CAVALCANTE *et al.*, 2011) e os obtidos nesta dissertação sobre as cifras RC5 e Skipjack, vale ressaltar nesta dissertação foi avaliada uma versão otimizada da cifra Skipjack e utilizada uma versão mais estável do sistema operacional TinyOS, obtendo, assim, alguns resultados diferentes dos apresentados por Cavalcante et al. (2011).

Conforme os resultados obtidos com relação aos mecanismos de criptografia, modos de operação e algoritmos de autenticação, um *ranking* de classificação está disposto nas tabelas 6, 7, e 8, respectivamente. Nesses *rankings*, o índice 1 representa o mecanismo que apresentou o melhor desempenho, enquanto o índice 2 indica o que apresentou o 2º melhor desempenho, e sucessivamente. Vale ressaltar que a cifra Skipjack foi omitida devido ao seu baixo nível de segurança oferecido, conforme indicado na seção 4.2.

Tabela 6. *Ranking* das cifras de bloco.

	AES	**Klein**	**Present**	**RC5**
Memória ROM	4	1	3	2
Memória RAM	4	2	3	1
Tempo de Execução	4	2	3	1
Consumo de Energia	4	2	3	1

Tabela 7. *Ranking* dos modos de operação.

	CBC	**CFB**	**CTR**	**OCB**	**OFB**
Memória ROM	5	2	3	4	1
Memória RAM	4	1	1	5	1
Tempo de Execução	5	2	1	4	3
Consumo de Energia	5	2	1	4	3

Tabela 8. *Ranking* dos MACs.

	CBCMAC	**CMAC**	**HMAC-MD5**	**HMAC-SHA1**	**OCB**
Memória ROM	1	2	4	5	3
Memória RAM	3	3	1	2	5
Tempo de Execução	1	2	4	5	3
Consumo de Energia	1	2	4	5	3

5.8 Conclusão

A avaliação de desempenho realizada nesta dissertação consistiu na utilização de uma abordagem sistemática de avaliação de desempenho baseada em técnicas e estratégias tradicionais.

O processo de avaliação foi iniciado com as etapas de definição dos objetivos e do sistema e a caracterização dos serviços do sistema. Em seguida, foram realizadas as etapas de seleção das métricas, da técnica de avaliação e da carga de trabalho. Por fim, foi elaborada a estratégia de experimentação, seguido da análise e apresentação dos resultados.

Conforme discutido nas análises realizadas neste capítulo, os mecanismos que apresentam os melhores desempenhos em termos de memória ROM foram as cifras de bloco Skipjack e Klein, os modos de operação OFB e CFB, e os MACs CBCMAC e CMAC. Já com relação à memória RAM, os melhores desempenhos foram apresentados pelas cifras de bloco RC5 e Skipjack, pelos modos de operação OFB, CFB e CTR, e pelos MACs HMAC-MD5 e HMAC-SHA1.

Além disso, dentre os mecanismos avaliados, a cifra RC5, o modo de operação CTR e o algoritmo CBCMAC são as melhores opções em termos de tempo de execução e consumo de energia.

Vale ressaltar que os resultados obtidos neste trabalho podem contribuir para a escolha apropriada de mecanismos de segurança a serem empregados em aplicações de RSSF, bem como no desenvolvimento e evolução de protocolos de segurança para tais redes, uma vez que a escolha dos mecanismos de segurança mais adequados é essencial para o balanceamento do *trade-off* entre a segurança e o desempenho em RSSF.

6 CONCLUSÕES

Esta dissertação apresentou uma avaliação de desempenho de mecanismos de segurança para Redes de Sensores Sem Fio (RSSF), considerando mecanismos de criptografia, modos de operação criptográficos e algoritmos de autenticação.

Na seção 6.1 são apresentadas as conclusões sobre esta dissertação, discutindo aspectos sobre os resultados obtidos com o processo de avaliação dos mecanismos de segurança utilizado no contexto de RSSF, caracterizando as principais contribuições alcançadas com a conclusão deste trabalho. A seção 6.2 lista os trabalhos publicados no decorrer do desenvolvimento desta dissertação. Finalmente, a seção 6.3 encerra este capítulo com a indicação de perspectivas de trabalhos futuros decorrentes desta pesquisa.

6.1 Resultados Alcançados

Esta dissertação apresentou uma avaliação de desempenho de mecanismos de segurança, incluindo algoritmos de criptografia (AES, Skipjack, Klein, RC5 e Present), modos de operação criptográficos (CBC, CFB, OFB, CTR e OCB) e algoritmos de autenticação (CBCMAC, CMAC, HMAC-MD5 e HMAC-SHA1) no contexto de RSSF.

Para alcançar o objetivo principal deste trabalho, foi utilizado um processo de análise de desempenho sistemático baseado em uma abordagem genérica tradicional, que foi definido nas seguintes etapas: definição dos objetivos e do sistema; caracterização das funcionalidades do sistema; seleção das métricas de desempenho; seleção da técnica de avaliação e da carga de trabalho; estratégia de experimentação; e apresentação e análise dos resultados.

Seguindo o processo, os principais mecanismos de segurança disponíveis na literatura foram selecionados. Em seguida, as métricas quantidade de memória ROM e RAM requerida, tempo de execução e consumo de energia foram selecionadas para avaliar o desempenho das funcionalidades oferecidas por cada mecanismo de segurança, os quais foram implementados e validados em uma linguagem de programação específica para o sistema operacional de RSSF TinyOS. A técnica de avaliação escolhida foi a realização de experimentos reais, os quais foram

executados na plataforma de RSSF MicaZ com o auxílio do compilador do TinyOS e de um osciloscópio digital.

Em relação aos resultados obtidos, a cifra Skipjack, o modo OFB e o algoritmo CBCMAC apresentaram os melhores desempenhos de memória ROM requerida, enquanto que a cifra RC5, os modos OFB, CFB e CTR, e o algoritmo HMAC-MD5 apresentaram os melhores desempenhos de memória RAM requerida. Já em termos de tempo de execução e consumo de energia, os resultados indicam que a cifra RC5, o modo de operação CTR e o algoritmo CBCMAC são as melhores opções, dentre os mecanismos avaliados.

Vale ressaltar que os resultados obtidos nesta dissertação podem contribuir para a escolha apropriada de mecanismos de segurança a serem empregados em aplicações de RSSF que necessitam de segurança dos dados, bem como no desenvolvimento e evolução de protocolos de segurança para tais redes, uma vez que a escolha dos mecanismos de segurança mais adequados é essencial para o balanceamento do *trade-off* entre a segurança e o desempenho em RSSF.

Além disso, pode-se citar as seguintes contribuições secundárias:

- o estudo de viabilidade de diferentes mecanismos de segurança para o contexto de RSSF, incluindo mecanismos capazes de fornecer os serviços de confidencialidade, integridade e autenticação; e

- a descrição detalhada do processo de medições e disponibilização de todos os códigos utilizados na avaliação, facilitando a continuação deste trabalho e eventuais trabalhos relacionados.

É importante ainda salientar que, após exaustivas buscas na literatura, acredita-se que este é o primeiro trabalho a avaliar o desempenho da cifra de bloco Klein na plataforma Micaz.

6.2 Produção Científica

Durante o desenvolvimento desta dissertação de mestrado, os seguintes trabalhos científicos foram aceitos para publicação:

- ✓ **Cavalcante, T. M.**, Garcia, F. P., Gomes, D. G., e Andrade, R. M. C. Avaliação de desempenho dos algoritmos criptográficos skipjack e rc5 para redes de sensores sem fio. *In*: SIMPÓSIO BRASILEIRO DE COMPUTAÇÃO UBÍQUA E PERVASIVA (SBCUP), p.1103-1112, 2011.

- ✓ **Cavalcante, T. M.**, Garcia, F. P., e Andrade, R. M. C. Avaliação de Mecanismos de Segurança para Redes de Sensores Sem Fio. *In*: SIMPÓSIO BRASILEIRO DE REDES DE COMPUTADORES E SISTEMAS DISTRIBUÍDOS (SBRC), p. 277-290, 2012.

6.3 Trabalhos Futuros

Os principais direcionamentos com relação às perspectivas de trabalhos futuros decorrentes desta pesquisa são listados a seguir:

- avaliar o impacto e verificar a viabilidade de outros mecanismos de segurança em RSSF, assim como avaliar outras métricas como, por exemplo, a latência da rede em decorrência do uso de tais mecanismos;

- aprimorar o processo de avaliação de desempenho elaborado, explorando a identificação e a variação de parâmetros e fatores que podem afetar o desempenho; e

- analisar os mecanismos de segurança com relação às quantidades de operações primitivas aritméticas, lógicas, entre outras, visando uma maior facilidade de comparação de desempenho dos mecanismos com relação a diferentes plataformas de RSSF.

Além do que foi citado anteriormente e conforme discutido na análise dos resultados obtidos no capítulo 5, os projetistas de RSSF devem levar em consideração as prioridades das aplicações com relação a cada uma das métricas avaliadas neste trabalho visando o melhor balanceamento do *trade-off* entre a segurança e o desempenho em RSSF. Diante dessa

constatação, um interessante trabalho futuro seria o desenvolvimento de um protocolo de segurança específico para RSSF capaz de oferecer certo grau de flexibilidade com relação à escolha dos mecanismos e serviços de segurança a serem utilizados. Com essa flexibilidade, os projetistas de RSSF poderiam selecionar uma combinação de mecanismos e serviços de segurança mais apropriada a uma determinada aplicação por meio de diferentes configurações do protocolo.

Finalmente, outra forma interessante de fornecer a flexibilidade mencionada anteriormente é por meio de uma camada de adaptação de mecanismos de segurança, conforme o nível de energia e o consumo de memória da RSSF. Nesse caso, o principal objetivo seria a busca pelo melhor balanceamento entre o desempenho da RSSF e o nível de segurança exigido.

REFERÊNCIAS

AKYILDIZ, I. F.; SU, W.; SANKARASUBRAMANIAM, Y.; CAYIRCI, E. Wireless sensor networks: a survey. **Computer Networks**, v. 38, p. 393-422, 2002.

AMAUSSON, J. P.; PLASENCIA, M. N.; SAARINEN, M. J. O. Pratical attack on 8 rounds of the lightweight block cipher. *In*: INTERNATIONAL CONFERENCE ON CRYPTOLOGY IN INDIA, 2011.

AMIN, F.; JAHANGIR, A. H., RASIFARD, H. Analysis of public-key cryptography for wireless sensor networks security. *In:* WORLD ACADEMY OF SCIENCE, ENGINEERING AND TECHNOLOGY, 2008.

BHARATHIDASAN, A.; PONDURU, V. **Sensor network**: an overview. Technical Report, University of California, Davis, 2003.

BHATTACHARYYA, D.; KIM, T.; PAL, S. A comparative study of wireless sensor networks and their routing protocols. *In*: **Sensors**, p. 10506-10523, 2010.

BELLARE, M.; KILIAN, J.; ROGAWAY, P. The security of the cipher block chaining message authentication code. In: JOURNAL OF COMPUTER AND SYSTEM SCIENCES, vol 61, 2000.

BRAGA, A. M.; RUBINA C. M. F.; DAHAB, R. Tropyc: a pattern language for cryptographic software. In: PATTERN LANGUAGES OF PROGRAMS, 1998.

BOGDANOV, A.; KNUDSEN, L. R.; LEANDER, G.; POSCHMANN, A.; ROBSHAW, M. J. B.; SEURIN, Y.; VIKKELSOE, C. Present: an ultra-lightweight block cipher. In: CRYPTOGRAPHIC HARDWARE AND EMBEDDED SYSTEMS, 2007.

BORGES NETO, J. B.; GOMES, D. G.; RIBEIRO NETO, P. F.; ANDRADE, R. M. C. Integração de protocolos de roteamento e de sincronização de tempo para redes de sensores sem fio. *In*: EURO AMERICAN CONFERENCE ON TELEMATICS AND INFORMATION SYSTEMS, 2010a.

BORGES NETO, J. B.; GOMES, D. G.; RIBEIRO NETO, P. F.; ANDRADE, R. M. C. Wireless sensor networks advances for ubiquitous computing. In: Mendes Neto, F. M., Ribeiro Neto, P. F. **Designing solutions-based ubiquitous and pervasive computing**: new issues and trends. Hershey-PA: IGI Global, p. 175-189, 2010b.

BORGES NETO, J. B. **Prost**: Um protocolo de roteamento sensível ao tempo para redes de sensores sem fio. 2010. 141 f. Dissertação (Mestrado em Engenharia de Teleinformática) – Centro de Tecnologia, Universidade Federal do Ceará, Fortaleza, 2010.

BORGHOFF, J.; Knudsen, L. R.; Leander, G.; Thomsen, S. S. Cryptanalysis of present-like ciphers with secret s-boxes. *In*: FAST SOFTWARE ENCRYPTION, v. 6733, 2011.

BOYLE, D.; NEWE, T. Securing wireless sensor networks: security architectures. *In*: JOURNAL OF NETWORKS, 3, p. 65-77, 2008.

BULUSU, N.; ESTRIN, D.; GIROD, L. Scalable coordination for wireless sensor networks: self-configuring localization systems. *In*: INTERNATIONAL SYMPOSIUM ON COMMUNICATION THEORY AND APPLICATIONS, Ambleside, 2001.

CASTRO, R. R. **Application-driven security in wireless sensor networks**. Thesis 223 f. (Doctor in Computer Science) – University of Malaga, Malaga, 2008.

CAVALCANTE, T. M.; GARCIA, F. P.; GOMES, D. G.; ANDRADE, R. M. C. Avaliação de desempenho dos algoritmos criptográficos skipjack e rc5 para redes de sensores sem fio. *In*: SIMPÓSIO BRASILEIRO DE COMPUTAÇÃO UBÍQUA E PERVASIVA, p.1103-1112, 2011.

CIRQUEIRA, A. C.; ANDRADE, R. M. C.; CASTRO, M. F. Um mecanismo de segurança com adaptação dinâmica em tempo de execução para dispositivos móveis. In: SEMINÁRIO INTEGRADO DE SOFTWARE E HARDWARE, p. 1337-1351, 2011.

COLONNA, J. G.; NAKAMURA, E. F.; SANTOS, E. M. Classificação de anuros baseado em vocalizações para monitoramento ambiental pervasivo. In: SIMPÓSIO BRASILEIRO DE COMPUTAÇÃO UBÍQUA E PERVASIVA, p. 1093-1102, 2011.

CROSSBOW. **MPR-MIB user manual**, Rev June, PN 7430-0021-07, 2006. Disponível em: <http://bit.ly/wPUViu> Acesso em: jan. 2012.

CRYPTREC, Cryptography research and evaluation commitees. Security of 128-bit block cipher AES, Disponível em: <http://bit.ly/t0fJV8>, Acesso em: out. 2011.

DARWISH, A.; HASSANIEN, A. E. Wearable and implantable wireless sensor network solutions for healthcare monitoring. **Sensors**, p. 5561-5595, 2011.

DELFS, H.; KNEBL, H. **Introduction to cryptography**: principles and applications. 2th Edition, Springer, 2007.

DOCKLIGHT. **Rs232 terminal / rs232 monitor**. Disponível em: < http://bit.ly/ahmrpZ>. Acesso em: jan. 2012.

DONG, W.; CHEN, C.; LIU, X.; BU, J. Providing os support for wireless sensor networks: challenges and approaches. In: IEEE COMMUNICATIONS SURVEYS & TUTORIALS, v. 12, no. 4, ch. 4, p. 519-530, 2010.

DWORKING, M. **Recommendation for block cipher modes of operation,** 2001. Disponível em: <http://1.usa.gov/gu49gK>. Acesso em: out. 2011 .

DWORKING, M. **Recommendation for block cipher modes of operation**: the cmac mode for authentication, 2005. Disponível em: <http://1.usa.gov/uqxrRA>. Acesso em: out. 2011.

GARCIA-HERNANDEZ, C. F.; IBARGUENGOYTIA-GONZALEZ, P. H.; GARCIA-HERNANDEZ, J.; PEREZ-DIAZ, J. A. Wireless sensor networks and applications: A survey. *In*: INTERNATIONAL JOURNAL OF COMPUTER SCIENCE AND NETWORKS SECURITY, v. 7, p. 264-273, 2007.

GONG, Z.; NIKOVA, S.I.; LAW, Y.W. Klein: A new family of lightweight block ciphers. In: WORKSHOP ON RFID SECURITY, 2011.

HAYASHI, R.; TANAKA, K. Pa in the two-key and generic conversion for encryption with anonymity. In Information Security and Privacy, **Lecture Notes in Computer Science**, vol. 4058, p. 271-282, 2006.

HEFEEDA, M.; BAGHERI, M. Wireless sensor networks for early detection of forest fires. In: CONFERENCE ON MOBILE ADHOC AND SENSOR SYSTEMS, 2007.

HU, F.; SHARMA, N. K. Security considerations in ad hoc sensor networks. **Ad Hoc Networks**, vol 3, p. 69-89, 2005.

JAIN, R. **The art of computer systems performance analysis**: techniques for experimental design, measurement, simulation, and modeling. Wiley-Interscience, 1991.

JANANI, K.; DHULIPALA, V. R. S.; CHANDRASEKARAN, R. M. A wsn based framework for human health monitoring. In: INTERNATIONAL CONFERENCE ON DEVICES AND COMMUNICATIONS, 2011.

JINWALA, D. C.; PATEL, D. R.; DASGUPTA, K. S. Configurable link layer security architecture for wireless sensor networks. In: WORLD CONGRESS ON ENGINEERING, 2008.

KAHATE, A. **Cryptography and network security**. 2th Edition, The McGraw-Hill, 2008.

KAPS, J. P.; GAUBATZ, G.; SUNAR, B. Cryptography on a speck of dust. **Computer**, v. 40, p. 38-44, 2007.

KARLOF, C.; SASTRY, N.; WAGNER, D. Tinysec: a link layer security architecture for wireless sensor networks. *In*: INTERNATIONAL CONFERENCE ON EMBEDDED NETWORKED SENSOR SYSTEMS, p. 162-175, 2004.

KO, J.; LIM, J. H.; CHEN, Y.; MUSALOIU-E, R.; TERZIS, A.; MASSON, G. M.; GAO, T.; DESTLER, W.; SELAVO, L.; DUTTON, R. P. Medisn: medical emergency detection in sensor networks. *In*: ACM TRANSACTIONS ON EMBEDDED COMPUTING SYSTEMS, 2010a.

KO, J.; LU, C.; SRIVASTAVA, M. B.; STANKOVIC, J. A.; TERZIS, A.; WELSH, M. Wireless sensor networks for healthcare. *In*: PROCEEDINGS OF THE IEEE, p. 1947-1960, 2010b.

KUMAR, N.; KUMAR, A.; CHAUDHRY, D. A novel approach to use nano-sensor in wsn applications. In: INTERNATIONAL JOURNAL OF COMPUTER APPLICATIONS, p. 31-34, 2011.

LEMOS, M. V. S.; LEAL, L. B.; FILHO, R. H. Detecção de intrusão em redes de sensores sem fio utilizando uma abordagem colaborativa e cross-layer. *In*: SIMPÓSIO BRASILEIRO DE REDES DE COMPUTADORES, p. 247-260, 2009.

LEVIS, P.; LEE, N. **Tossim**: a simulator for tinyos networks, 2003. Technical Report, University of California, Berkeley. Disponível em <http://bit.ly/z6aVky> Acesso em: jan. 2012.

LEVIS, P.; LEE, N.; WELSH, M.; CULLER, D. Tossim: accurate and scalable simulation of entire tinyos applications. In: INTERNATIONAL CONFERENCE ON EMBEDDED NETWORKED SENSOR SYSTEMS, p. 126-137, 2003.

LI, D.; WONG, K. D.; HU, Y. H.; SAYEED, A. M. Detection, classification, and tracking of targets. **IEEE Signal Processing Magazine**, p. 17-29, 2002.

LI, W.; WANG, Z.; SONG, Y. Wireless sensor network design for wildfire monitoring. *In*: WORLD CONGRESS ON INTELLIGENT CONTROL AND AUTOMATION, 2006.

LIU, A.; NING, P. TinyECC: A cofigurable library for elliptic curve cryptography in wireless sensor networks. *In*: 7TH INTERNATIONAL CONFERENCE ON INFORMATION PROCESSING IN SENSOR NETWORKS, p. 245-256, 2008.

LOUREIRO, A. A. F.; NOGUEIRA, J. M. S., RUIZ, L. B., MINI, R. A., NAKAMURA, E. F., FIGUEIREDO, C. M. S. Redes de sensores sem fio. *In*: SIMPÓSIO BRASILEIRO DE REDES DE COMPUTADORES, p. 179-226, 2003.

LOW, K. S.; WIN, W. N. N.; ER, M. J. Wireless sensor networks for industrial environments. *In*: WORLD CONGRESS ON INTELLIGENT CONTROL AND AUTOMATION, p. 271-276, 2005.

LUK, M.; MEZZOUR, G.; PERRIG, A.; AND GLIGOR, V. D. Minisec: a secure sensor network communication architecture. *In*: IEEE INTERNATIONAL CONFERENCE ON INFORMATION PROCESSING IN SENSOR NETWORKS, 2007.

MAINWARING, A.; CULLER, D.; POLASTRE, J; SZEWCZYK, R.; ANDERSON, J. Wireless sensor networks for habitat monitoring. *In*: ACM INTERNATIONAL WORKSHOP ON WIRELESS SENSOR NETWORKS AND APPLICATIONS, p. 88-97, 2002.

MARGI, C. B.; SIMPLICIO, M. A. J.; BARRETO, P. S. L. M.; CARVALHO, T. C. M. B. **Segurança em redes de sensores sem fio**. In: SANTIN, A.; NUNES, R. C.; DAHAB, R. (Org), 1 Edição, Porto Alegre: Sociedade Brasileira de Computação, p. 149-194, 2009.

MELODIA, T. **Communication and coordination in wireless multimedia sensor and actor networks**. Thesis 190 f. (Doctor in Philosophy) – School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, 2007.

MENEZES, A. J.; OORSCHOT, P. C.; VANSTONE, S. **A. Handbook of applied cryptography**. CRC Press, 1996.

MICROCHIP. **Mplab**: integrated development environment. Disponível em: <http://bit.ly/yOi22Y>. Acesso em: jan. 2012.

NIST, National Institute of Standards and Technology. **Advanced enhanced encryption, 2001**. FIPSP 197. Disponível em: < http://1.usa.gov/8Y4V6U >. , Acesso em: out 2011.

_____, National Institute of Standards and Technology. **Skipjack and kea algorithm specifications, 1998**. Disponível em: <http://bit.ly/uCWxLY>. , Acesso em: dec 2011.

_____. **The keyed-hash message authentication code, 2002**. FIPSP 198. Disponível em: <http://1.usa.gov/gjg0co>. Acesso em: out. 2011.

PHAN, R. C. Cryptanalysis of full Skipjack block cipher. **Electronics Letters**, 38, 2, p. 69-71, 2002.

POTLAPALLY, N. R.; RAVI, S.; RAGHUNATHAN, A.; JHA, N K. A study of the energy consumption characteristics of cryptographic algorithms and security protocols, *In*: IEEE TRASACTIONS ON MOBILE COMPUTING, p. 128–143, 2005.

RIBEIRO NETO, P. F. **Mecanismos de qualidade de servico para o gerenciamento de dados e transações em tempo real**. Tese (Doutorado em Ciências no Domínio da Engenharia Elétrica) – Universidade Federal de Campina Grande, Campina Grande, 2006.

RIVEST, R. L. The RC5 encryption algorithm. . In FAST SOFTWARE ENCRYPTION, p. 86–96, 1994.

ROGAWAY, P.; BELLARE, M.; BLACK, J. OCB: a block-cipher mode of operation for efficient authenticated encryption. In: ACM TRANSACTIONS ON INFORMATION AND SYSTEM SECURITY, vol. 6, p. 365-403, 2003.

ROMER, K.; MATTERN, F**. The Design Space of Wireless Sensor Networks.** In: IEEE WIRELESS COMMUNICATIONS, p. 54-61, 2004.

RUIZ, L. B.; CORREIA, L. H. A.; VIEIRA, L. F.; MACEDO, D.; NAKAMURA, E. F.; FIGUEIREDO, C. M. S.; VIEIRA, M. A. M.; MECHELANE, E. H.; CAMARA, D.; LOUREIRO, A. A. F.; NOGUEIRA, J. M. S.; SILVA JR, D. C. Arquitetura para redes de sensores sem fio. In: SIMPÓSIO BRASILEIRO DE REDES DE COMPUTADORES, 2004.

SABBATH, E.; MAJEED, M.; KANG, K.; LIU, K.; ABDUL-GHAZALEH, N. An application-driven perspective on wireless sensor network security. *In*: SYMPOSIUM ON QOS AND SECURITY FOR WIRELESS AND MOBILE NETWORKS, 2006.

SANTOS, M. A. S. **Análise comparativa de protocolos de segurança para redes de sensores sem fio**. Dissertação 113 p. (Mestrado em Ciências) – Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2009.

SARAOGI, M. Security in Wireless Sensor Networks. *In*: ACM CONFERENCE ON EMBEDDED NETWORKED SENSOR SYSTEMS, 2004.

SCHNEIER, B. **Applied cryptography**: protocols, algorithms, and source code in c. 2th Edition, John Wiley Sons, 1996.

SCHNEIER, B. **Sha-1 broken**, 2005. Disponível em: <http://bit.ly/rqS8Nc>, Acesso em: out. 2011.

SHANON, C. **Communication theory of secrecy systems**. Bell Systems Technical Journal, n. 4, 1949.

SHEN, X.; WANG, Z.; SUN, Y. Wireless sensor networks for industrial applications. *In*: WORLD CONGRESS ON INTELLIGENT CONTROL AND AUTOMATION, p. 3636-3640, 2004.

SHONG, C.; KUMAR, S. P. Sensor networks: evolution, opportunities, and challenges. *In*: PROCEEDINGS OF THE IEEE, p. 1247-1256, 2003.

SILVA, A. P. R. D.; MARTINS, M. H. T.; ROCHA, B. P. S.; LOUREIRO, A. A. F.; RUIZ, L. B.,; WONG, H. C. Decentralized intrusion detection in wireless sensor networks.
In Proceedings of the 1st ACM international workshop on Quality of service & security in wireless and mobile networks, pages 16–23, New York, NY, USA. ACM, 2005.

SIMON, G.; MAROTI, M.; LEDECZI, A.; BALOGH, G.; KUSY, B.; NADAS, A.; PAP, G.; SALLAI, J.; FRAMPTON, K. Sensor network-based countersniper system. *In*: INTERNATIONAL CONFERENCE ON EMBEDDED NETWORKED SENSOR SYSTEMS, 2004

SIMPLÍCIO, M. A. J. **Algoritmos criptográficos para redes de sensores sem fio**. Dissertação 190 p. (Mestrado em Engenharia Elétrica), Escola Politécnica da Universidade de São Paulo, São Paulo, 2008.

SIMPLÍCIO, M. A. J.; BARRETO, P. S. L. Algoritmos criptográficos para redes de sensores. In: SIMPÓSIO BRASILEIRO DE EM SEGURANÇA DA INFORMAÇÃO E SISTEMAS COMPUTACIONAIS, p. 523-530, 2010.

SINOPOLI, B.; SHARP, C.; SCHENATO, L.; SHAFFERT, S.; SASTRY, S. S. Distributed control applications within sensor networks. *In*: INTELLIGENT DISTRIBUTED SURVEILLANCE SYSTEMS, 2004.

SOMMERVILLE, I. **Software engineering**. 9 ed., Addison-Wesley, 2011.

STALLINGS, W. **Cryptography and network security**: principles and practice, Prentice Hall, 5th Edition, 2010.

SZEWCZYK, R.; MAINWARING, A.; POLASTRE, J.; ANDERSON, J.; CULLER, D. An analysis of a large scale habitat monitoring application. *In*: CONFERENCE ON EMBEDDED NETWORKED SENSOR SYSTEMS, p. 214-226, 2004.

TEKTRONIX. Osciloscópios digitais de tempo real tds 210, tds 220 e tds 224. Disponível em: <http://bit.ly/IDG5OD> Acesso em: out. 2011.

WALTERS, J. P.; LIANG, Z.; SHI, W.; CHAUDHARY, V. **Wireless sensor network security**: survey. In XIAO, Y. Security in distributed, grid, mobile and pervasive computing, p. 367-417, 2006.

WERNER-ALLEN, G.; LORINCZ, K.; JHONSON, J.; LEES, J.; WELSH, M. Fidelity and yield in a volcano monitoring sensor network. *In*: SYMPOSIUM ON OPERATING SYSTEMS DESIGN AND IMPLEMENTATION, p. 381-396, 2006.

ZHAO, F.; GUIBAS, L. J. **Wireless sensor networks**: an information processing approach. San Francisco: Morgan Kaufmann, 2004.

ZHOU, Y.; FANG, Y. **Network security and attack defense**. In ZHENG, J.; JAMALIPOUR, A. Wireless sensor networks: a networking perspective. Wiley, 2009.

APÊNDICES

APÊNDICE A – Memória, tempo de execução e consumo de energia das cifras de bloco.

Tabela 9. Quantidade de memória ROM e RAM das cifras de bloco.

Mecanismo	ROM (*bytes*)	RAM (*bytes*)
AES	7230	2395
Skipjack	4412	561
RC5	4756	305
Klein	4736	817
Present	5568	1345

Tabela 10. Tempo de Execução de inicialização da chave das cifras de bloco com intervalo de confiança de 95%.

Mecanismo	Tempo (ms)
AES	0.2867 ± 0,0133
Klein	0.5128 ± 0.0098
Present	2.1622 ± 0.0232
RC5	0.5128 ± 0.0490
Skipjack	0.1841 ± 0.0144

Tabela 11. Tempo de Execução de encriptação das cifras de bloco com intervalo de confiança de 95%.

Mecanismo	16 *bytes* (ms)	32 *bytes* (ms)	64 *bytes* (ms)	128 *bytes* (ms)
AES	2.1587 ± 0.0211	4.2235 ± 0.0500	8.4003 ± 0.0615	16.5967 ± 0.1096
Klein	1.2251 ± 0.0116	2.4167 ± 0.0101	4.8619 ± 0.0110	9.8354 ± 0.0134
Present	1.8212 ± 0.0137	3.5622 ± 0.0110	7.0906 ± 0.0229	14.1671 ± 0.0330
RC5	0.8400 ± 0.0490	1.7138 ± 0.0082	3.3290 ± 0.0241	6.7883 ± 0.0335
Skipjack	0.8606 ± 0.0107	1.7383 ± 0.0228	3.4548 ± 0.0436	6.8451 ± 0.1404

Tabela 12. Tempo de Execução de decriptação das cifras de bloco com intervalo de confiança de 95%.

Mecanismo	16 *bytes* (ms)	32 *bytes* (ms)	64 *bytes* (ms)	128 *bytes* (ms)
AES	2.7216 ± 0.0599	6.1619 ± 0.0651	11.5116 ± 0.0636	22.6303 ± 0.1118
Klein	1.5345 ± 0.0147	2.8941 ± 0.0276	5.8241 ± 0.0137	11.2925 ± 0.0241
RC5	1.0261 ± 0.0313	1.8867 ± 0.0140	3.6870 ± 0.0424	7.0796 ± 0.0209
Skipjack	0.8529 ± 0.3068	1.7383 ± 0.0242	3.4812 ± 0.3118	6.7890 ± 0.1487

Tabela 13. Consumo de energia na inicialização da chave das cifras de bloco no modo CBC com intervalo de confiança de 95%.

Mecanismo	Energia (µJ)
AES	6.4237 ± 0.2994
Klein	11.4875 ± 0.2212
Present	48.4345 ± 0.5217
RC5	73.1468 ± 1.0978
Skipjack	4.1259 ± 0.3237

Tabela 14. Consumo de energia na encriptação das cifras de bloco no modo CBC com intervalo de confiança de 95%.

Mecanismo	16 *bytes* (µJ)	32 *bytes* (µJ)	64 *bytes* (µJ)	128 *bytes* (µJ)
AES	48.3551 ± 0.4730	94.6074 ± 1.1206	188.1672 ± 1.3795	371.7676 ± 2.4557
Klein	27.4436 ± 0.2614	54.1357 ± 0.2261	108.9073 ± 0.2860	220.3148 ± 0.3004
Present	40.7968 ± 0.3079	79.7945 ± 0.2382	158.8304 ± 0.5150	317.3430 ± 0.7406
RC5	18.8160 ± 0.1303	38.3907 ± 0.1839	74.5703 ± 0.5403	152.0586 ± 0.7512
Skipjack	19.2784 ± 0.2416	38.9398 ± 0.5110	77.3883 ± 0.9770	153.3316 ± 3.1466

Tabela 15. Consumo de energia na decriptação das cifras de bloco no modo CBC com intervalo de confiança de 95%.

Mecanismo	16 *bytes* (µJ)	32 *bytes* (µJ)	64 *bytes* (µJ)	128 *bytes* (µJ)
AES	60.9641 ± 1.3425	138.0273 ± 1.4600	257.8600 ± 1.4258	506.9191 ± 2.5057
Klein	34.3731 ± 0.3307	64.8299 ± 0.6911	130.4619 ± 0.3086	252.9537 ± 0.5417
RC5	22.9852 ± 0.7019	42.2637 ± 0.3150	82.5909 ± 0.9508	158.5847 ± 0.4702
Skipjack	19.1050 ± 0.3068	38.9398 ± 0.5433	77.9808 ± 1.3118	152.0743 ± 3.3318

APÊNDICE B – Memória, tempo de execução e consumo de energia dos modos de operação.

Tabela 16. Quantidade de memória ROM e RAM dos modos de operação.

Mecanismo	ROM (*bytes*)	RAM (*bytes*)
CBC	7230	2395
OFB	4495	1115
CFB	4562	1115
CTR	4630	1115
OBC	6614	2651

Tabela 17. Tempo de execução dos modos de operação na encriptação com o AES-128 com intervalo de confiança de 95%.

Mecanismo	16 *bytes* (ms)	32 *bytes* (ms)	64 *bytes* (ms)	128 *bytes* (ms)
CBC	2.1587 ± 0.0211	4.2235 ± 0.0500	8.4003 ± 0.0615	16.5967 ± 0.1096
CFB	1.1458 ± 0.0144	2.2080 ± 0.0059	4.2893 ± 0.0526	8.8796 ± 0.0178
CTR	1.1129 ± 0.0037	2.1990 ± 0.0041	4.3219 ± 0.0092	8.5074 ± 0.0079
OCB	3.3170 ± 0.0051	4.4735 ± 0.0100	6.9451 ± 0.0394	11.3422 ± 0.0159
OFB	1.1827 ± 0.0179	2.2761 ± 0.0088	4.6096 ± 0.0074	9.1948 ± 0.0280

Tabela 18. Tempo de execução dos modos de operação na decriptação com o AES-128 com intervalo de confiança de 95%.

Mecanismo	16 *bytes* (ms)	32 *bytes* (ms)	64 *bytes* (ms)	128 *bytes* (ms)
CBC	2.7216 ± 0.0599	6.1619 ± 0.0651	11.5116 ± 0.0636	22.6303 ± 0.1118
OCB	3.4377 ± 0.0352	5.2416 ± 0.3164	8.0161 ± 0.0224	13.8009 ± 0.0051

Tabela 19. Consumo de energia dos modos de operação na encriptação com o AES-128 com intervalo de confiança de 95%.

Mecanismo	16 *bytes* (µJ)	32 *bytes* (µJ)	64 *bytes* (µJ)	128 *bytes* (µJ)
CBC	48.3551 ± 0.4730	94.6074 ± 1.1206	188.1672 ± 1.3795	371.7676 ± 2.4557
CFB	25.6660 ± 0.3230	49.4606 ± 0.1326	96.0815 ± 1.1799	198.9047 ± 0.6284
CTR	24.9290 ± 0.0844	49.2583 ± 0.0918	96.8113 ± 0.2080	190.5661 ± 0.1786
OCB	74.3029 ± 0.1155	100.2074 ± 0.2253	155.5716 ± 0.8837	254.0666 ± 0.3577
OFB	26.4320 ± 0.4015	50.9852 ± 0.1991	103.2567 ± 0.1672	205.9643 ± 0.6284

Tabela 20. Consumo de energia dos modos de operação na decriptação com o AES-128 com intervalo de confiança de 95%.

Mecanismo	**16 *bytes* (µJ)**	**32 *bytes* (µJ)**	**64 *bytes* (µJ)**	**128 *bytes* (µJ)**
CBC	60.9641 ± 1.3425	138.0273 ± 1.4600	257.8600 ± 1.4258	506.9191 ± 2.5057
OCB	77.0054 ± 0.7900	117.4121 ± 0.7088	179.5612 ± 0.5027	309.1417 ± 0.1158

APÊNDICE C – Quantidade de memória, tempo de execução e consumo de energia dos MACs.

Tabela 21. Quantidade de memória ROM e RAM dos MACs.

Mecanismo	ROM (*bytes*)	RAM (*bytes*)
OCB	6614	2651
CBCMAC	4058	971
CMAC	4448	971
HMAC-MD5	10936	193
HMAC-SHA1	28508	201

Tabela 22. Tempo de execução dos MACs com intervalo de confiança de 95%.

Mecanismo	16 *bytes* (ms)	32 *bytes* (ms)	64 *bytes* (ms)	128 *bytes* (ms)
CBCMAC	1.0809 ± 0.0039	2.2487 ± 0.0063	4.4387 ± 0.0126	9.0367 ± 0.0254
CMAC	2.4970 ± 0.0151	3.6251 ± 0.0133	5.8241 ± 0.0159	10.3087 ± 0.0235
HMAC-MD5	11.0983 ± 0.0187	11.1064 ± 0.0185	13.6503 ± 0.0159	16.2267 ± 0.0144
HMAC-SHA1	33.6029 ± 0.0120	33.6071 ± 0.0151	42.4274 ± 0.0139	48.8254 ± 0.0149

Tabela 23. Consumo de energia dos MACs com intervalo de confiança de 95%.

Mecanismo	16 *bytes* (µJ)	32 *bytes* (µJ)	64 *bytes* (µJ)	128 *bytes* (µJ)
CBCMAC	24.2136 ± 0.0895	50.3711 ± 0.1421	99.4271 ± 0.2826	202.4237 ± 0.5695
CMAC	55.9349 ± 0.3401	81.2036 ± 0.2991	130.4619 ± 0.3583	230.9151 ± 0.5284
HMAC-MD5	248.6039 ± 0.4138	248.7844 ± 0.4150	305.7671 ± 0.3718	363.4796 ± 0.3234
HMAC-SHA1	752.7049 ± 0.2694	752.7990 ± 0.3389	950.3742 ± 0.3134	1093.6907 ± 0.3338