UNIVERSIDADE FEDERAL DE SERGIPE

CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Análise Arquitetural de Algoritmos Criptográficos Assimétricos em Plataformas Embarcadas usadas em RSSF

Gustavo da Silva Quirino

SÃO CRISTÓVÃO/ SE

2013

UNIVERSIDADE FEDERAL DE SERGIPE

CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Gustavo da Silva Quirino

Análise Arquitetural de Algoritmos Criptográficos Assimétricos em Plataformas Embarcadas usadas em RSSF

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação (PROCC) da Universidade Federal de Sergipe (UFS) como parte de requisito para obtenção do título de Mestre em Ciência da Computação.

Orientador: Prof. Dr. Edward David Moreno Ordonez

SÃO CRISTÓVÃO/ SE

2013

Gustavo da Silva Quirino

Análise Arquitetural de Algoritmos Criptográficos Assimétricos em Plataformas Embarcadas usadas em RSSF

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação (PROCC) da Universidade Federal de Sergipe (UFS) como parte de requisito para obtenção do título de Mestre em Ciência da Computação.

BANCA EXAMINADORA

Prof. Dr. Edward David Moreno Ordonez, Presidente

Universidade Federal de Sergipe (UFS)

Prof. Dr. Ricardo P. B. Salgueiro, Membro

Universidade Federal de Sergipe (UFS)

Prof. Dr. Eduardo Freire Nakamura, Membro

Universidade Federal do Amazonas (UFAM) / FUCAPI

Análise Arquitetural de Algoritmos Criptográficos Assimétricos em Plataformas Embarcadas usadas em RSSF

Este exemplar corresponde à Dissertação de Mestrado do aluno **Gustavo da Silva Quirino** para ser avaliada pela Banca examinadora.

São Cristóvão - SE, 13 de março de 2013

Prof. Dr. Edward David Moreno Ordonez

Orientador

Prof. Dr. Ricardo P. B. Salgueiro

Membro

Prof. Dr. Eduardo Freire Nakamura

Membro

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA CENTRAL - UFS

Q8a

QUIRINO, Gustavo da Silva
Análise arquitetural de algoritmos criptográficos assimétricos em plataformas embarcadas usadas em RSSF / Gustavo da Silva Quirino; orientador: Edward David Moreno Ordonez – São Cristóvão, 2013.
103f. : il.

Dissertação (Mestrado em Ciência da Computação) – Universidade Federal de Sergipe, 2013

1. Criptografia de dados (Computação). 2. Certificado Digital.
3. Rede de sensores sem fio. 4. Desempenho
I. Moreno Ordonez, Edward David, orient. II. Título

CDU: 004.056.55

Dedicatória

Aos meus irmãos Éden Felipe da Silva Quirino e Lucas Christiano da Silva Quirino;

À minha namorada Tamires Fernandes Ramos;

Agradecimentos

Aos colegas: Leila Matos, Luana Barreto, Marcelino Oliveira e Wanderson Roger;

Aos professores: Ricardo Salgueiro, Tarcísio Rocha e Leila Maciel;

Aos colaboradores: Prof. Eduardo Nakamura (UFAM), Prof. Diego Aranha (UnB), Daniel Fernando Pigatto (USP), Ricardo Maia(USP);

Especialmente ao meu orientador **Edward David Moreno Ordonez**

Resumo

Esse trabalho consiste na avaliação de desempenho de algoritmos criptográficos assimétrico em plataformas embarcadas usadas em Redes de Sensores Sem Fio (RSSF). Os dispositivos sensores têm baixa capacidade computacional e energética, portanto técnicas de segurança não devem consumir grande quantidade de recursos. Os algoritmos criptográficos assimétricos RSA, ECC e MQQ foram avaliados nas plataformas Desktop, ARM, MSP430 e AVR. A avaliação de desempenho foi realizada com auxílio dos simuladores SimpleScalar, SimPanalyzer, MSPsim e AVRORA. Os critérios de avaliação foram: tempo de processamento, uso de memória e processador, além do consumo de energia. Os dados revelaram que o algoritmo MQQ foi mais eficiente que os algoritmos RSA e ECC na maioria dos critérios de avaliação, além disso, o algoritmo ECC obteve os melhores resultados quando comparado com o algoritmo RSA.

Palavras Chave: Criptossistemas de chave pública, Rede de Sensores sem Fio, Avaliação de Desempenho.

Abstract

This work consists in a performance evaluating of Asymmetrical cryptographic algorithms in embedded platforms used in Wireless Sensor Networks (WSN). The sensor devices have low computing power and energy, therefore safety techniques should not consume large amounts of resources. The asymmetric cryptographic algorithms RSA, ECC and MQQ were evaluated on platforms Desktop, ARM, MSP430 and AVR. The evaluations were performed using the SimpleScalar simulators, Sim-Panalyzer, MSPsim and AVRORA . The evaluation criteria were: processing time, memory usage and processor, as well as energy consumption. The data showed that the algorithm MQQ was more efficient than RSA and ECC algorithms in most assessment criteria, in addition, the ECC algorithm obtained the best results when compared with the RSA algorithm.

Keywords: Public-key Cryptosystems, Wireless Sensor Network, Performance Evaluation.

Lista de Figuras

Lista de Tabelas

Lista de Siglas

AEAD - *Authenticated Encryption with Associated Data*

ANSI - *American National Standards Institute*

CBC - *Cipher-Block Chaining*

CMAC - *Cipher-based MAC*

DARPA - *Advanced Research Projects Agency*

DoS - *Denial of Service*

ECC - *Elliptic Curve Cryptography*

ECDH - *Elliptic Curve Diffie-Hellman*

ECDSA - *Elliptic Curve Digital Signature Algorithm*

ECIES - *Elliptic Curve Integrated Encryption Standard*

ECMQV - *Elliptic Curve Menezes-Qu-Vanstone*

ECNR - *Elliptic Curve Nyberg Rueppel Signatures*

ECPVS - *Elliptic Curve Pintsov Vanstone*

FIPS - *Federal Information Processing Standard*

GCC - *Gnu Compiler Collection*

GMAC - *Galois Message Authentication Code*

GNFS - *General Number Field Sieve*

HECC - *Hyperelliptic Curve Cryptography*

IEEE - *Institute of Electrical and Electronics Engineers*

ISA - *Instruction Set Architecture*

ISO - *International Organization for Standardization*

MAC - *Message Authentication Code*

MQQ - *Multivariate Quadratic Quasigroup*

MWSN - *Mobile Wireless Sensor Network*

NIST - *National Institute of Standards and Technology*

OCB - *Offset Codebook*

PKC - *Public Key Cryptosystem*

PMAC - *Parallelizable MAC*

RFID - *Radio-Frequency IDentification*

RSA - *Rivest, Shamir, Adleman*

RSSF - *Rede de Sensores Sem Fio*

SPINS - *Security Protocols for Sensor Networks*

WSN - *Wireless Sensor Network*

Sumário

Capítulo 1

Introdução

Uma rede de sensores é formada por dispositivos autônomos denominados nós sensores, que de modo geral apresentam baixa capacidade computacional, limitação na capacidade de transmissão de dados e restrições de energia. Uma rede de sensores sem fio (RSSF) consiste em nós sensores capturando informações de um ambiente, processando os dados e transmitindo-os através de sinais de rádio. As RSSF estão cada vez mais presentes em nosso dia-a-dia, podendo ser encontradas na área ambiental em medições climáticas, presença de fumaça, na área da saúde em medição dos sinais vitais, temperatura, automação residencial em sistemas detectores de movimento e de imagem, entre outras diversas áreas de atuação. Geralmente as RSSF não têm estrutura fixa, tampouco existe monitoramento dos nós durante a vida operacional da rede, neste caso devem ser implantados mecanismos de auto-configuração e adaptação em caso de falhas, inclusão ou exclusão de um nó. Um exemplo óbvio de RSSF que requer segurança na transmissão de dados são as aplicações militares, além disso, neste tipo de aplicação existe a necessidade de sigilo de localização e resistência a tentativas de destruição da rede. Segundo Amara, Beghdad e Oussalah (2013), outras aplicações também apresentam requisitos de segurança como: Monitoramento de Desastres, onde é importante proteger os dados da rede, principalmente de terroristas. Segurança Pública, em aplicações onde produtos químicos, biológicos, ou outras ameaças ambientais são monitoradas, é vital que a disponibilidade da rede nunca esteja ameaçada. Saúde, onde a privacidade das informações dos pacientes, disponibilidade da rede e autenticação dos profissionais devem ser garantidas.

As RSSF têm requisitos de segurança relativamente similares às redes convencionais, portanto parâmetros como confidencialidade, integridade, disponibilidade e autenticação de-

vem ser levados em conta na criação de um ambiente seguro. Devido às limitações das RSSF, nem todas as soluções de segurança criadas para redes convencionais podem ser aplicadas diretamente neste tipo de rede. A criptografia é o método de segurança mais utilizado em sistemas computacionais. Essa técnica consiste basicamente em transformar uma informação da sua forma original para um formato ilegível usando-se chaves criptográficas, de forma que somente o destinatário consiga decifrá-la. Na criptografia simétrica apenas uma chave criptográfica é utilizada, em contrapartida, a criptografia assimétrica ou criptografia de chave pública faz uso de duas chaves distintas. Durante muito tempo acreditou-se que a criptografia de chave pública não era adequada para redes de sensores pois exigia alto poder de processamento, porém após estudos realizados com algoritmos de criptografia baseados em curvas constatou-se a viabilidade dessa técnica nas RSSF (AHMAD; BEG; ABBAS, 2010). Os algoritmos assimétricos mais avaliados em sistemas embarcados são *Rivest Shamir Adleman* (RSA) (RIVEST; SHAMIR; ADLEMAN, 1978) e *Elliptic Curve Cryptography* (ECC) (MILLER, 1986; KOBLITZ, 1987). Alguns trabalhos consideram o *Hyperelliptic Curve Cryptography* (HECC) (MENEZES; ZUCCHERATO; WU, 1996), e pelo fato de ter sido criado recentemente, poucos trabalhos avaliaram o algoritmo *Multivariate Quadratic Quasigroup* (MQQ) (GLIGOROSKI; MARKOVSKI; KNAPSKOG, 2008).

O algoritmo criptográfico RSA é atualmente o mais utilizado dentre os assimétricos, funcionando a partir da dificuldade de fatoração de números primos. Padronizado pelo *National Institute of Standards and Technology* (NIST)[1], este algoritmo é bastante utilizado em transações na internet. Os algoritmos ECC e HECC foram criados na década de 80, e são baseados na dificuldade de se resolver o problema do logaritmo discreto sobre curvas elípticas e hiperelípticas, respectivamente. Apesar de sua complexidade, os algoritmos baseados em curvas elípticas e hiperelípticas têm sido estudados e avaliados por pesquisadores. Recentemente o algoritmo de chave pública denominado Multivariado Quadrático Quase Grupo (MQQ) foi proposto no meio acadêmico, esse algoritmo é baseado em polinômios multivariados quadráticos e transformações de *strings* quase grupos (EL-HADEDY; GLIGOROSKI; KNAPSKOG, 2008; GLIGOROSKI; MARKOVSKI; KNAPSKOG, 2008). Os algoritmos envolvidos neste estudo são assimétricos, porém cada um trabalha com um modo matemá-

[1]NIST - Agência Americana de tecnologia que mantém parceria com a indústria para desenvolver e aplicar tecnologias, medidas e padrões. Maiores informações em: www.nist.gov

tico de encriptação específico.

Muitas pesquisas avaliaram o desempenho de algoritmos criptográficos em redes de sensores, porém não existe uma padronização na forma da análise. Margi, Jr e Barreto (2009) afirmam que os trabalhos de avaliação de desempenho de algoritmos criptográficos para redes de sensores são muitas vezes bastante distintos em termos de metodologia, plataforma, métricas e foco da análise, o que dificulta uma comparação direta entre os resultados obtidos. Neste contexto, este trabalho avalia o desempenho dos algoritmos criptográficos RSA, ECC e MQQ em plataformas usadas em RSSF, seguindo a mesma metodologia de avaliação com enfase no consumo de memória, energia e tempo de processamento.

O algoritmo RSA é estudado, uma vez que é o algoritmo assimétrico padrão na maioria das aplicações de segurança. Por outro lado, o algoritmo ECC é avaliado por apresentar bons resultados em plataformas embarcadas. Finalmente, consideramos o algoritmo MQQ por apresentar excelentes resultados quando comparado com o RSA em trabalhos anteriores. As próximas seções abordam questões metodológicas da fase de projeto.

1.1 Justificativa

Um clássico da comparação de algoritmos criptográficos para redes de sensores foi realizado pelos autores Gura et al. (2004). Esta avaliação envolveu os algoritmos de chave pública RSA e ECC aplicados nas plataformas 8051 baseado no Chipcon CC1010, e Atmel AVR Atmega128. Os parâmetros de comparação foram tamanho de código e consumo de memória. Os resultados mostraram que o ECC apresentou desempenho 2 vezes superior ao RSA. Um aprimoramento da função de multiplicação do ECC melhorou o consumo de memória em 25%. Outro trabalho que comparou os algoritmos RSA e ECC foi realizado por Ahmad, Beg e Abbas (2010), porém neste caso foi avaliado somente o consumo de energia aplicado na plataforma Mica2dot. Os resultados mostraram que o ECC consumiu 4 vezes menos energia que o RSA, porém o ambiente criado pelos autores Ahmad, Beg e Abbas (2010) não é fácil de ser replicado no mundo real, pois segundo eles o esquema só funciona em RSSF compostas por nós com alto recurso energético, como na plataforma do sensor iMote. Além disso, eles mostraram que o consumo de energia aumenta de forma proporcional à quantidade de

sensores.

Os autores Boyle e Newe (2008) informaram que a utilização de RSSF irá crescer consideravelmente nos próximos anos devido sua aplicação em todas as áreas que permitem sensoriamento de dados.

Recentemente foi criado o algoritmo MQQ, que parece mais simples que outros algoritmos assimétricos, apesar de sua complexidade matemática. Neste sentido, experimentos realizados na plataforma FPGA e Desktop mostraram que o MQQ é mais veloz que algoritmos como RSA e ECC, e em análises recentes, o MQQ ofereceu a mesma segurança de tradicionais *Public Key Cryptosystem* (PKC) consumindo menos recursos computacionais (GLIGOROSKI; MARKOVSKI; KNAPSKOG, 2008)(EL-HADEDY; GLIGOROSKI; KNAPSKOG, 2008)(MAIA, 2010), porém não há estudos para plataformas embarcadas e para RSSF usando o promissor MQQ.

A partir das considerações anteriores, observa-se que existe uma busca constante por algoritmos criptográficos que sejam eficientes e que consumam o mínimo de recursos possíveis em uma RSSF. Porém, a maioria dos autores avaliaram o desempenho dos algoritmos, sem preocupação com a influência que os parâmetros de arquitetura das plataformas podem acarretar na avaliação, como uso de memória cache, número de instruções processadas, perdas de dados, entre outros. Além disso, o MQQ tornou-se uma promessa para plataformas de processamento limitado (GLIGOROSKI; MARKOVSKI; KNAPSKOG, 2008)(EL-HADEDY; GLIGOROSKI; KNAPSKOG, 2008). Uma melhor descrição das principais avaliações de desempenho de algoritmos criptográficos assimétricos em sistemas embarcados está presente no capítulo 4. É importante ainda salientar que a segurança em RSSF está presente nos grandes desafios da pesquisa em computação no Brasil (CARVALHO; LEON et al., 2006).

1.2 Problemática e Hipótese

Em geral técnicas de segurança aplicadas em dispositivos computacionais são consumidoras de recursos físicos “limitados” como memória, processador e energia. Devido a alimentação dos sensores por meio de uma bateria, o consumo energético de algoritmos criptográficos

limita o tempo de vida da rede. Além disso, consumo de memória e processador de uma técnica de segurança dificulta a comunicação eficiente da rede, pois a medida que os nós estão gastando recursos com a segurança, eles deixam de transportar os dados obtidos do sensoriamento. Devido a criação recente do algoritmo MQQ (GLIGOROSKI; MARKOVSKI; KNAPSKOG, 2008), não existem trabalhos que realizaram um comparativo detalhado dos algoritmos criptográficos RSA, ECC e MQQ em uma mesma plataforma. Neste sentido surge um questionamento: Dentre os algoritmos RSA, ECC e MQQ, qual seria o mais eficiente em plataformas embarcadas usadas em RSSF, tendo como parâmetros de comparação consumo de energia, memória e tempo de processamento ?

Avaliações realizadas com algoritmos baseados em curvas elípticas apontam que o ECC é o algoritmo de chave pública mais rápido em plataformas com restrições computacionais, porém espera-se verificar através de um estudo comparativo se o MQQ apresenta melhores resultados frente ao RSA e ECC em relação a consumo de energia, memória e tempo de processamento quando analisado em plataformas específicas usadas em RSSF.

1.3 Objetivos

O objetivo principal desta dissertação é avaliar o desempenho dos algoritmos criptográficos de chave pública RSA, ECC e MQQ em plataformas embarcadas usadas em RSSF e determinar o algoritmo mais eficiente, segundo critérios de consumo de energia, uso de memória e tempo de processamento.

Objetivos Específicos

1. Avaliar o desempenho dos algoritmos criptográficos RSA, ECC e MQQ em plataformas embarcadas usadas em RSSF;

2. Realizar um comparativo entre algoritmos criptográficos assimétricos que considere os aspectos arquiteturais das plataformas embarcadas usadas em RSSF.

3. Determinar qual algoritmo é mais eficiente em plataformas embarcadas usadas em RSSF;

1.4 Metodologia

Segundo Wazlawick (2008) esta pesquisa pode ser enquadrada no estilo "Apresentação de algo presumivelmente melhor" pois a abordagem apresentada para se chegar ao melhor algoritmo criptográfico de chave pública, dentre os estudados, segundo os parâmetros pré-estabelecidos foi comparada **quantitativamente** com outras da literatura. Neste sentido o método utilizado para se chegar aos resultados foi dividido em duas partes, descritas a seguir.

A primeira parte da pesquisa consistiu em uma **pesquisa bibliográfica**, que é definida por Severino (2007) como sendo aquela que se realiza a partir do registro disponível, decorrente de pesquisas anteriores, em documentos impressos, como livros, artigos, teses etc. Utiliza-se de dados ou de categorias teóricas já trabalhadas por outros pesquisadores e devidamente registrados.

A segunda parte da pesquisa consistiu em uma **pesquisa experimental**, que é definida por Severino (2007) como sendo aquela que toma o próprio objeto em sua concretude como fonte e o coloca em condições técnicas de observação e manipulação experimental em equipamentos de um laboratório, onde são criadas condições adequadas para seu tratamento.

Neste estudo, os algoritmos criptográficos RSA e ECC foram escritos com base na biblioteca Miracl - *Multiprecision Integer and Rational Arithmetic C/C++ Library*. Segundo Scott (2003) a biblioteca Miracl é referência nas ferramentas criptográficas por apresentar facilidades de implementação de algoritmos em aplicações de segurança no mundo real. Além disso, a Miracl disponibiliza códigos compactos, rápidos e eficientes, apresentando alto desempenho em qualquer processador ou plataforma. Estudos realizados por Uto e Reis (2005), Pigatto, Silva e Branco (2011) e Pigatto (2012) mostraram que códigos baseados na Miracl obtém desempenho superior à códigos baseados em outras bibliotecas criptográficas, como a RELIC, Crypto++, LibTomCrypt, OpenSSL e LiDIA+GMP.

Em relação ao algoritmo MQQ, foi utilizada a versão escrita por Maia (2010) em sua dissertação de mestrado e no artigo (MAIA; BARRETO; OLIVEIRA, 2010).

Inicialmente, a plataforma desktop foi utilizada nas avaliações de medição de tempo,

como forma de validar a eficiência dos algoritmos. Em seguida as avaliações foram realizadas na plataforma ARM, que está presente na maioria dos dispositivos embarcados. Foram realizadas também avaliações na plataforma MSP430, que está presente nos sensores Tmote Sky e TelosB. Por fim, a avaliação foi feita na plataforma AVR, que está presente em sensores como MicaZ e Mica2. Segundo Pesovic et al. (2012), os processadores mais usados em dispositivos de RSSF são MSP430 e AVR Atmega128.

Devido ao alto custo de plataformas reais de sensores e da possibilidade de se avaliar diversos modelos de sensores através de programas computacionais eficientes, optou-se por realizar a avaliação de desempenho por meio de simuladores. A simulação é uma técnica que consiste em imitar o comportamento dinâmico de um sistema usando-se de um programa de computador. No contexto da avaliação de desempenho, a simulação é útil na realização de experiências que não seriam possíveis com o sistema real (LAW; KELTON; KELTON, 1991). Os simuladores SimpleScalar (AUSTIN; LARSON; ERNST, 2002) e Sim-Panalyzer (AUSTIN; LARSON; ERNST, 2002) da plataforma ARM, o simulador Avrora (TITZER; LEE; PALSBERG, 2005) da plataforma AVR e o simulador MSPsim (ERIKSSON et al., 2007) da plataforma MSP430 foram utilizados devido ao grande número de trabalhos acadêmicos que utilizam essas ferramentas em avaliações de desempenho.

Segundo Austin, Larson e Ernst (2002), o Sim-Panalyzer é um simulador de consumo de energia baseado no simulador arquitetural SimpleScalar. O SimpleScalar é um simulador de arquitetura computacional que modela um computador virtual com CPU, memórias caches (de dados e instruções), memória RAM (Random Access Memory) e toda a hierarquia das memórias. Com base neste modelo, os simuladores SimpleScalar e Sim-Panalyzer conseguem simular programas reais executando sobre tais plataformas.

A ferramenta Avrora foi produzida pela ATmel para simulação de programas voltados para o microcontrolador AVR e plataforma de sensor Mica2. Segundo Titzer, Lee e Palsberg (2005), o Avrora têm a capacidade de realizar a medição de tempo de fenômenos em redes de sensores, com eficiência similar e em alguns critérios superior a simuladores como ATEMU (POLLEY et al., 2004) e TOSSIM (LEVIS et al., 2003).

O MSPsim é escrito na linguagem JAVA e simula processadores da família MSP430, além de algumas plataformas de sensores.

O tempo de execução foi medido e analisado para verificar a viabilidade dos algoritmos RSA, ECC e MQQ em plataformas embarcadas usadas em RSSF. Devido a quantidade restrita de memória em plataformas embarcadas, o consumo de memória também foi avaliado. A maioria dos dispositivos usados em RSSF são alimentadas por fonte esgotável de energia, portanto a avaliação e análise do consumo energético foi realizada, com detalhamento do consumo por componentes arquiteturais.

1.5 Contribuições

As contribuições deste trabalho consistem em:

- Criação de uma metodologia que engloba plataformas com alto e baixo recurso computacional, além da avaliação de algoritmos com simuladores de plataformas embarcadas. Esse modelo de análise deve servir de base para novas pesquisas.
- Indicação do algoritmo criptográfico de chave pública mais eficiente em plataformas embarcadas usados em RSSF, segundo os parâmetros pré-estabelecidos, de forma que essa contribuição seja revertida em melhor análise das técnicas de segurança para que mais pessoas possam utilizar esse tipo de tecnologia;
- Publicação de trabalhos acadêmicos, relatando os resultados obtidos neste estudo, conforme apresentado na página 90 desta dissertação.

1.6 Organização

Este documento está organizado em 5 capítulos. O capítulo 2 descreve alguns conceitos fundamentais de redes de sensores sem fio, os tipos de ataques mais comuns nesse tipo de rede, bem como as técnicas de segurança aplicáveis a RSSF com foco nos algoritmos criptográficos assimétricos. No capítulo 3 está descrito o histórico, aplicações e funcionamento dos algoritmos RSA, ECC e MQQ. O capítulo 4 aborda o estado da arte das avaliações de desempenho de algoritmos criptográficos em plataformas embarcadas usadas em RSSF. O

capítulo 5 mostra os resultados e análise dos testes. O capítulo 6 traz as conclusões, bem como os trabalhos futuros.

Capítulo 2

Rede de Sensores Sem Fio (RSSF)

Sensores são dispositivos eletrônicos que têm entre seus principais componentes unidades de armazenamento, processamento, sensoriamento e de transmissão. Geralmente esses dispositivos têm baixa capacidade computacional, porém desempenham um papel importante na computação ubíqua, pois têm a função de coletar dados em um determinado ambiente. Segundo Loureiro et al. (2003), as Redes de Sensores Sem Fio (RSSF) podem ser vistas como um tipo especial de rede móvel *ad hoc (MANET - MobileAd hoc Network)* que tendem a executar uma função colaborativa onde os elementos (sensores) provêem dados, que são processados (ou consumidos) por nodos especiais chamados de sorvedouros *(sink nodes)*. Em uma RSSF a transmissão de dados entre os componentes da rede é realizada usando sinais de rádio. A área de aplicação deste tipo de rede é abrangente, podendo ser utilizada em monitoramento ambiental, controle de temperatura e umidade, controle de tráfego de veículos, monitoramento de órgãos do corpo humano, entre outros. A Figura 2.1 descrita por Le et al. (2010) ilustra um cenário de utilização de redes de sensores na área médica, onde os pacientes que estão sendo monitorados podem estar no hospital, na residência, ou em outro lugar qualquer, realizando uma atividade rotineira. Os dados do sensoriamento que neste caso podem ser batimento cardíaco, pressão ou temperatura são encaminhados aos profissionais da saúde por meio da Internet.

Algumas áreas de aplicação de redes de sensores requerem segurança no transporte das informações, a exemplo do cenário ilustrado pela Figura 2.1 em que os sensores implantados no corpo humano transmitem dados para o hospital. Segundo Le et al. (2010) no campo da saúde, autenticação e controle de acesso são os principais desafios de uma topologia de rede dinâmica, móvel e com recursos limitados. Além do setor médico, diversas outras áreas tam-

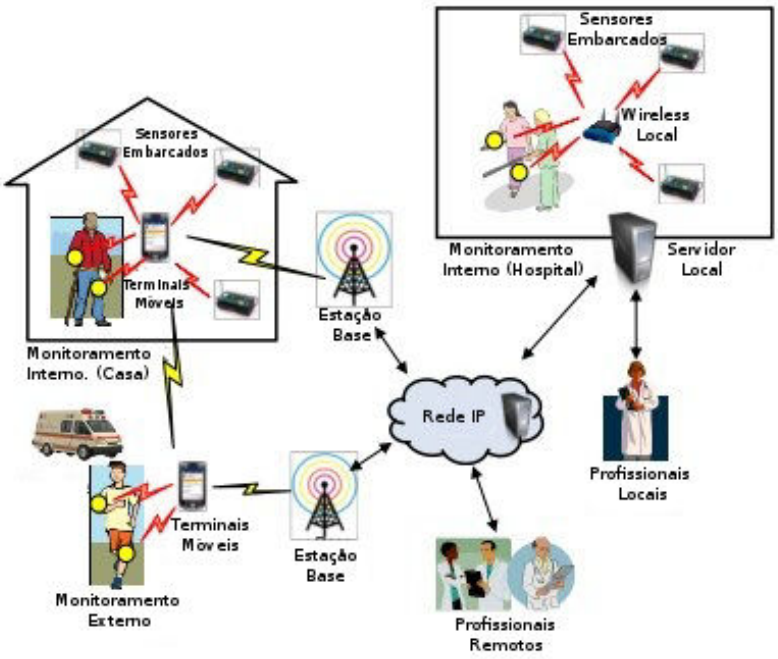

Figura 2.1: Exemplo de um cenário de RSSF (LE et al., 2010)

bém necessitam de segurança em suas transmissões como: industria, segurança patrimonial e aplicações militares. A agência de segurança americana denominada DARPA[1] desenvolve inúmeras pesquisas envolvendo segurança em redes de sensores para fins militares.

As redes de sensores podem ser aplicadas em diversas áreas, atuando na captura e transmissão de dados. Além dos ambientes citados no inicio do capitulo, segundo Loureiro et al. (2003) as RSSF podem ser utilizadas nas situações descritas a seguir.

- **Ambiente -** Monitoramento de variáveis ambientais como prédios, residências e locais externos como oceanos, vulcões,desertos, etc.
- **Tráfego -** Monitoramento de tráfego de veículos em rodovias, ferrovias, rios, oceanos, etc.
- **Segurança -** Para prover segurança em residências, centros comerciais, fazendas, entre outros.
- **Militar -** Para detectar presença de inimigo, explosões, presença de material perigoso como gás venenoso e radiação.

[1] *Advanced Research Projects Agency*. Mais informações em: http://www.darpa.mil

Este capítulo está dividido da seguinte forma: A seção 2.1 descreve características dos dispositivos sensores mais utilizados atualmente. A seção 2.2 aborda as vulnerabilidades de segurança que podem ocorrer em uma RSSF. Na seção 2.3, são definidos os mecanismos de defesa aplicáveis a RSSF. Por fim, a seção 2.4 traz as considerações finais do capítulo.

2.1 Dispositivos Sensores

Os dispositivos sensores são formados basicamente por uma parte computacional responsável pelo armazenamento e transmissão dos dados, além de uma parte de sensoriamento que pode ser composta por um ou vários sensores, tais como acústico, sísmico, infravermelho, vídeo-câmera, calor, temperatura e pressão (LOUREIRO et al., 2003). Em geral dois formatos de modulação são disponíveis: *Frequency-Shift-Keyed* (FSK) operando a 433 e 868-915 MHz e o *direct sequence spread spectrum* (DSSS) operando a 2.4 GHz, faixa em que transmitem os padrões 802.15.4 e ZigBee (GIACOMIN; VASCONCELOS, 2006). O alcance dos rádios varia entre 10 e 100 metros. A configuração da antena pode fazer com que as taxas de transmissão variem entre 19.2kbps a 240kbps (MEMSIC, 2012).

Atualmente os estados de funcionamento dos sensores podem variar entre o modo atividade, inatividade *(idle)* e baixo consumo *(sleep)* de forma a economizar energia. A questão energética é importante, pois a maioria dos sensores são alimentados por baterias. Os principais sensores disponíveis atualmente no mercado são LOTUS, IRIS, MICAz, MICA2, TELOSB e CRICKET (MEMSIC, 2012).

Figura 2.2: Exemplo de plataformas de sensores [Memsic 2012]

A Figura 2.2 ilustra o formato real dos sensores IRIS, MicaZ e TelosB. Os sensores

IRIS e MicaZ utilizam o processador AVR. O sensor TelosB utiliza o processador MSP430. Durante o desenvolvimento desta dissertação não foram encontradas empresas brasileiras que comercializassem plataformas específicas de sensores como MicaZ e Mica2. Um orçamento realizado na empresa chinesa Mensic[2] em fev/2013 mostrou que um Micaz custava U$114,00 e um TelosB U$160,00, sem contar com as taxas de importação, além disso as compras mínimas deveriam ser de U$1.000,00.

2.2 Vulnerabilidades de Segurança

Na maioria das aplicações de redes de sensores sem fio, os dispositivos ficam espalhados em grandes áreas, o que dificulta a aplicação de um controle individual dos componentes da rede. Além disso, a comunicação sem fio permite que um possível invasor dispare ataques sem ter acesso físico ao dispositivo, portanto segundo Shi e Perrig (2004) os ataques a redes de sensores podem ser divididos em três principais tipos:

Ataque contra autenticação e confidencialidade: Consiste em ataques de alteração, repetição ou modificação de pacotes.

Ataque de disponibilidade da rede: Geralmente conhecidos como ataques de Denial of Service (DoS) ou negação de serviço, esse ataque consiste na aplicação de técnicas que tornem a rede indisponível.

Ataque contra a integridade: Neste tipo de ataque o objetivo do invasor é injetar dados falsos na rede, mantendo a rede disponível, porém trafegando dados fictícios.

A Tabela 2.1 descrita por Wang, Attebury e Ramamurthy (2006) ilustra os tipos mais comuns de ataques em RSSF segundo a camada em que atuam, destacando os ataques na camada de rede.

Na camada física podem ocorrer os ataques (i) *jamming* e (ii) *tampering*. O ataque *jamming* consiste na interferência do sinal de radiofrequência que os nodos utilizam para se comunicar. O ataque *Tampering* ocorre devido a vulnerabilidade física dos nodos espa-

[2]Mensic - Empresa chinesa especializada em dispositivos eletrônicos. Maiores informações em: http://www.memsic.com/

Tabela 2.1: Principais ataques a uma RSSF

Camada	**Tipos de Ataques**
Camada Física	*jamming* ou ataque de interferência
Camada de Enlace	*collision, exhaustion, unfairness*
Camada de Rede	*spoofed routing information and selective forwarding, sinkhole, sybil, wormhole, Hello flood, Ack Flooding,*
Camada de Transporte	*Flooding De-synchronization*
Camada de Aplicação	*Malicious Node*

lhados em grandes áreas, portanto suscetíveis a captura, quebra dos circuitos, modificação da configuração ou até mesmo substituição de um nodo da rede por um sensor malicioso (WANG et al., 2005). Na camada de enlace os ataques podem ser do tipo *collision*, quando dois nodos tentam transmitir na mesma frequência simultaneamente, neste caso o pacote é descartado e necessita ser retransmitido (WOOD; STANKOVIC, 2002). O atacante pode causar colisões propositais por meio de um nodo malicioso. A repetição das colisões pode levar à exaustão dos recursos, tornando os nodos indisponíveis. Ainda na camada de enlace o ataque *unfairness* é um tipo de DoS em que o adversário provoca degradação das aplicações de tempo real executada em outros nós por interrupção intermitente das transmissões de seus quadros. Ataques do tipo DoS consistem em inundar o receptor com requisições para que nenhuma outra comunicação possa ser realizada durante o ataque, deixando os nodos envolvidos indisponíveis para novas conexões.

Na camada de rede podem ocorrer ataques do tipo *Spoofed routing information*, onde o invasor altera informações da tabela de roteamento. As rotas falsas fazem com que os pacotes não cheguem ao destino correto, ou até mesmo fazem com que o encaminhamento consuma mais recursos que o normal (KARLOF; WAGNER, 2003). O ataque *Selective forwarding* consiste no comprometimento de um nodo por meio de o invasor que faz com que algumas mensagens sejam encaminhadas e outras descartadas (KARLOF; WAGNER, 2003). No ataque *sinkhole* o atacante faz com que um nodo compromissado seja visto como rota mais eficiente para o *sink* da rede, desta forma os nodos vizinhos sempre usarão o invasor para encaminhar seus dados (NEWSOME et al., 2004)(KARLOF; WAGNER, 2003)(WOOD; STANKOVIC, 2002).

O *sybil attack* acontece quando um nodo malicioso assume mais de uma identidade na rede. Segundo Douceur (2002) este ataque foi originalmente destinado a sistemas distri-

buídos de armazenamento de dados redundantes, porém ele também é eficiente contra algoritmos de roteamento, agregação de dados, alocação de recursos, entre outros. O ataque *Wormhole* consiste em um elo de baixa latência entre dois nodos de uma rede através do qual um atacante gera reenvio de mensagens com intuito de esgotar os recursos dos dispositivos (KARLOF; WAGNER, 2003). No ataque *Hello flood* o invasor pode usar um transmissor de alta potência para enganar um grande número de nós, fazendo-os acreditar que eles estão próximos (KARLOF; WAGNER, 2003). Posteriormente o atacante transmite um falso caminho mais curto para a estação base, e todos os nós que receberam os pacotes *Hello*, tentam transmitir através do nó atacante. No entanto, estes nós estão fora do alcance do rádio do nodo malicioso. Alguns algoritmos de roteamento utilizam informações do estado de funcionamento dos nós. O ataque *Acknowledgment spoofing* consiste na disseminação de falsas informações sobre os estados dos nodos vizinhos realizada por um nodo malicioso de forma a impedir que os pacotes cheguem a seus destinos (KARLOF; WAGNER, 2003).

Já na camada de transporte o ataque *Flooding* consiste na inundação de requisições para novas conexões de forma a esgotar os recursos de memória e impedir o fechamento de requisições legítimas. O ataque *De-synchronization* refere-se a interrupção de uma conexão existente (WOOD; STANKOVIC, 2002). Neste ataque o invasor captura mensagens forçando o emissor a reenviá-las gastando energia de forma desnecessária.

Existem ainda ataques que exploram vulnerabilidades de autenticação e sigilo dos dados. O ataque de replicação consiste na implantação de um nodo malicioso que assume a identidade de um nodo da rede. Esse falso nodo pode corromper ou encaminhar pacotes em falsas rotas. Se o invasor tiver acesso físico a rede, ele pode copiar as chaves criptográficas e utilizá-las em falsas mensagens. Além disso o invasor pode implantar o nodo malicioso em locais estratégicos de forma a segmentar a rede de sensores. A preservação de privacidade nas transmissões de dados em RSSF é desafiador, uma vez que esse tipo de rede permite acesso remoto. Além disso, um único adversário pode monitorar várias redes simultaneamente (CHAN; PERRIG, 2003). *Eavesdropping and passive monitoring* (Escutas e monitoramento passivo) é a forma mais comum e mais fácil de ataque à privacidade de dados. Neste tipo de ataque o espião monitora as transferências de dados e pode ter acesso a seu conteúdo caso não haja nenhum mecanismo de criptografia implantado na rede que está

sendo monitorada. O *Traffic analysis* (análise de tráfego) geralmente é aplicado em conjunto com o ataque de escuta e monitoramento passivo. Consiste na análise prévia do tráfego da rede de forma a identificar os nodos que estão gerando troca de dados que interessam ao invasor. Por fim, o ataque *camouflage*, em que o invasor implanta um nodo malicioso na rede que encaminha pacotes para nodos que estão sendo monitorados.

Dessa maneira, com essa análise é possível perceber que existe uma gama de ataques destinados a RSSF em todas as camadas da pilha de protocolos TCP/IP. Além disso é notório que um ponto em comum na maioria dos ataques é a exploração da baixa capacidade computacional dos sensores, uma vez que dados falsos são injetados e rotas são alteradas sempre com o intuito de ocupar a baixa capacidade de transmissão dos sensores, ou ainda eliminar sua reserva de energia. Outros ataques ainda não identificados podem ocorrer em RSSF, e proteger a rede dessas ameaças pode ser uma tarefa difícil.

2.3 Mecanismos de Defesa

Os diferentes tipos de aplicações em RSSF requerem diferentes requisitos de segurança. Em um ambiente de monitoramento de temperatura, onde pesquisadores coletam dados para pesquisas, pode ser que requisitos de segurança não sejam de extrema importância, porém o monitoramento de radiação nas proximidades de uma usina nuclear requer garantia de autenticidade, confidencialidade, disponibilidade e integridade. Algumas arquiteturas foram desenvolvidas para provêr segurança em redes de sensores, dentre elas destacam-se: SPINS,TinySec e MiniSec, além dessas o padrão IEEE 802.15.4 inclue um *framework* de segurança para atender os serviços de integridade dos dados, confidencialidade e autenticidade (MARGI; JR; BARRETO, 2009). O SPINS *(Security Protocols for Sensor Networks)* desenvolvido por (PERRIG et al., 2002) consiste em um conjunto de protocolos de segurança que atua usando criptografia e códigos de autenticação de mensagem. O TinySec foi projetado e implementado no sistema operacional TinyOS para ser um mecanismo de provimento de confidencialidade, integridade e autenticidade na camada de enlace de dados (KARLOF; SASTRY; WAGNER, 2004). Utiliza o modo de operação Cipher-block chaining (CBC) que pode ser combinado com algoritmos criptográficos simétricos baseados em cifras de bloco

como RC5 e skipjack (KARLOF; SASTRY; WAGNER, 2004). O MiniSec é um protocolo de camada de segurança para RSSF que utiliza o modo OCB (*Offset Codebook*) de operação de cifra de bloco, que elimina a necessidade de adicionar enchimento aos blocos de texto claro (LUK et al., 2007). O padrão IEEE 802.15.4 provê integridade, controle de acesso, confidencialidade e proteção contra repetição na camada de enlace. O algoritmo criptográfico utilizado neste padrão é o AES (IEEE, 2011), que é um algoritmo simétrico, e portanto precisa de uma única chave criptográfica.

2.4 Considerações Finais do Capítulo

Uma RSSF tende a ser autônoma e requer um alto grau de cooperação para executar as tarefas definidas para a rede. Isto significa que algoritmos distribuídos tradicionais, como protocolos de comunicação e eleição de líder, devem ser revistos para esse tipo de ambiente antes de serem usados diretamente (LOUREIRO et al., 2003; XIAO et al., 2009). Levando ainda em consideração a limitação computacional e principalmente de energia dos dispositivos é possível deduzir que nem tudo que funciona com eficiência nas redes tradicionais pode ser utilizado nas RSSF. As limitações computacionais de um dispositivo restringem a escolha de algoritmos criptográficos e protocolos de segurança. Além disso, o tempo de vida das baterias impede que se aplique técnicas complexas de segurança, pois isso diminuiria drasticamente o tempo de vida da rede (IEEE, 2011). Apesar dos padrões de segurança sugerirem algoritmos simétricos nos protocolos de segurança para RSSF, pesquisas recentes têm mostrado que alguns algoritmos de criptografia de chave pública são eficientes em dispositivos com baixa capacidade computacional, incluindo sensores. Este tipo de algoritmo provê maior eficiência por usar chaves assimétricas, e alguns deles como o ECC e MQQ têm alcançado a mesma velocidade de algoritmos simétricos. O próximo capitulo apresenta definições e características de criptografia, especificamente dos algoritmos criptográficos assimétricos RSA, ECC e MQQ. É importante enfatizar que o algoritmo RSA fez parte deste estudo por ser o mais utilizado dentre os assimétricos, portanto considerado padrão na criptografia de chave pública. O algoritmo ECC foi avaliado, pois apresenta os melhores resultados em plataformas embarcadas. Por fim, o MQQ foi avaliado por ter sido criado recentemente e

apresentar bons resultados em hardware, além disso não existem avaliações de desempenho em RSSF que tenham comparado este algoritmo com o ECC.

Capítulo 3

Algoritmos Assimétricos

Para um melhor entendimento das considerações realizadas neste trabalho, faz-se necessário apresentar alguns conceitos básicos de criptografia, bem como a definição e diferenciação dos métodos simétrico e assimétrico. Os algoritmos envolvidos na pesquisa também são apresentados e classificados de acordo com suas respectivas técnicas. A seção 3.1 define criptografia e suas funcionalidades. A seção 3.2 aborda sobre as classes de algoritmos criptográficos. As seções 3.3, 3.4 e 3.5 descrevem as características dos algoritmos RSA, ECC e MQQ. Finalmente, a seção 3.6 traz as considerações finais do capítulo.

3.1 Comunicação Segura

A criptografia de dados surgiu muito antes da invenção do computador. Diplomatas, amantes e principalmente militares contribuíram para a evolução dessa arte que consiste em distorcer uma informação que está sendo transportada, de forma que somente o receptor autorizado possa decifrá-la. Neste sentido, um algoritmo criptográfico pode ser definido como uma função que transforma mensagens claras em mensagens cifradas e vice-versa, utilizando para isso uma chave criptográfica.

A maioria dos algoritmos criptográficos são públicos, uma vez que segundo Tanembaum (2003), mantendo o algoritmo público o criador se livra de criptólogos ansiosos por decodificar o sistema, afim de publicar artigos, além de que passados cinco anos de sua exposição e nenhuma decodificação tenha obtido sucesso, o algoritmo passa a ser considerado sólido. O sigilo está na chave que tem a função de parametrizar a função criptográfica, isto é, somente

com a chave é possível cifrar ou decifrar uma mensagem em tempo hábil.

Outro fator importante é que a chave tem a capacidade de mudar a saída do algoritmo, portanto a cada mudança de chave o algoritmo criptográfico gera uma nova mensagem cifrada. O tamanho da chave é fator crucial em um projeto, pois quanto maior for a chave, mais trabalho terá o cripto-analista para tentar decifrar a mensagem. Em geral as chaves criptográficas apresentam tamanho de 64, 128, ou 256 bits, podendo ser maiores ou menores, de acordo com a necessidade de segurança.

Atualmente, além da confidencialidade, a criptografia também opera nos campos da integridade e autenticação, descritas a seguir:

- **Confidencialidade:** Garantia de que somente emissor e receptor tem a capacidade de entender a mensagem que está sendo trocada.
- **Integridade:** Possibilidade de verificar se uma mensagem apresenta alteração durante a transmissão.
- **Autenticação:** Mecanismo que permite comprovar a identidade de um indivíduo na comunicação.

Segundo Boyle e Newe (2008) a criptografia é o método padrão para defender uma rede de sensores da maioria dos ataques possíveis, e os vários níveis de criptografia implicam em variações de *overhead* na forma de crescimento do tamanho do pacote de dados, tamanho do código, uso de processador, memória, etc. A escolha de um algoritmo criptográfico eficiente para rede de sensores é o grande debate entre pesquisadores da área. Segundo Chen et al. (2009) os métodos criptográficos utilizados em RSSF devem atender às restrições computacionais dos dispositivos, além de passarem por avaliação antes de serem implantados.

3.2 Classes de Algoritmos Criptográficos

Tradicionalmente os usuários de criptografia utilizavam algoritmos simples, porém atualmente o objetivo é tornar o algoritmo tão complexo, que sem a chave fica praticamente impossível extrair alguma informação por meio de uma criptoanalise. As classes de algoritmos

criptográficos dizem respeito a como uma chave criptográfica é trocada e também a quantidade de chaves envolvidas na aplicação do método. A maioria dos algoritmos criptográficos existentes podem ser classificados em simétricos ou assimétricos.

3.2.1 Criptografia Simétrica

Criptografia simétrica ou **criptografia de chave secreta** consiste na utilização de apenas uma chave, tanto na cifragem quanto na decifragem dos dados. Até o ano de 1976 esse era o único método conhecido de utilização da criptografia, porém para que ele seja eficiente é necessário um canal seguro de comunicação em que a chave criptográfica possa ser trocada.

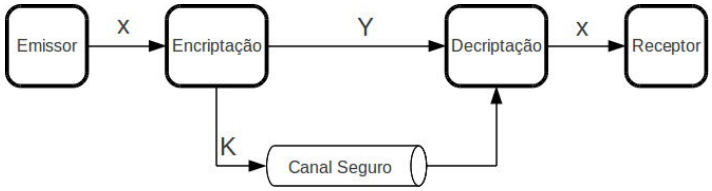

Figura 3.1: Funcionamento da Criptografia Simétrica

A Figura 3.1 ilustra uma comunicação que usa criptografia simétrica. O texto X é encriptado e transformado na mensagem Y por meio do algoritmo de criptografia e da chave K. A mensagem Y é enviada para o receptor, que utiliza a chave K para descriptografá-la, transformando-a novamente no texto X. Ainda de acordo com a Figura 3.1 é possível perceber que a chave K é transportada por um canal seguro, pois de posse dela, um possível atacante poderia facilmente fazer a leitura do texto original. Os algoritmos AES *Advanced Encryption Standard* e DES *Data Encryption Standard* são dois exemplos de algoritmos que fazem parte da classe simétrica.

3.2.2 Criptografia Assimétrica

A **criptografia de chave pública** ou **criptografia assimétrica** surgiu com uma mudança radical de paradigmas. Segundo Stallings (1995) algoritmos de chave pública são baseados em funções matemáticas, em vez de substituição e permutação. Além disso o fator mais importante é que a criptografia de chave pública é assimétrica, envolvendo o uso de duas chaves

distintas, em contraste com a criptografia simétrica convencional, que utiliza apenas uma chave. O uso de duas chaves tem consequências profundas nas áreas de confidencialidade, distribuição de chaves e autenticação. O principal diferencial da criptografia assimétrica, é que ela permite o estabelecimento de uma comunicação segura entre indivíduos, sem a necessidade do compartilhamento prévio de uma única chave criptográfica.

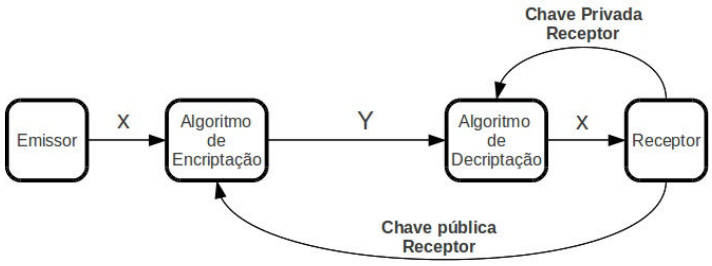

Figura 3.2: Funcionamento da Criptografia Assimétrica

Nesta classe de algoritmos criptográficos são utilizadas duas chaves distintas para cifragem e decifragem, melhor conhecidas como chave pública e sua correspondente chave privada. Neste modelo e de acordo com a Figura 3.2 o receptor divulga sua chave pública para que o emissor possa encriptar a mensagem, porém somente a chave privada do receptor, que é mantida em sigilo, é capaz de decifrá-la.

3.2.3 Técnica Simétrica x Técnica Assimétrica

O padrão IEEE 802.15.4 de 2011 define parâmetros para redes pessoais de baixo alcance (LR-WPANs). A primeira versão desse padrão foi publicada em 2003, e segundo Boyle e Newe (2008) foi constituído para ser o protocolo padrão de comunicação em redes de sensores sem fio. O mecanismo de criptografia indicado no padrão IEEE 802.15.4 é baseado na criptografia de chave simétrica. Porém, de acordo com Sen (2009), estudos recentes têm mostrado que é possível aplicar criptografia de chave pública em redes de sensores utilizando a seleção correta de algoritmos e parâmetros associados, otimização e técnicas de baixa potência. Em alguns casos a criptografia de chave pública obteve eficiência semelhante e até superior à criptografia de chave simétrica utilizando chaves menores. Segundo Struik (2011) já está provado que algoritmos de chave pública desenvolvidos em *hardware* são adequados para redes de sensores.

3.3 Algoritmo RSA

No artigo de apresentação do RSA, os autores Rivest, Shamir e Adleman (1978) propuseram um método para implementar um sistema de encriptação de chave pública cuja segurança repousa na dificuldade de se fatorar números inteiros grandes. Por meio dessa técnica é possível criptografar dados, bem como criar assinaturas digitais. Atualmente o RSA é o algoritmo de chave pública mais utilizado no mundo.

Segundo Moreno, Pereira e Chiaramonte (2005), quanto maior os números primos utilizados para a criação da chave, maior é a segurança proporcionada por esse algoritmo. Atualmente os números primos utilizados têm geralmente 512 bits de comprimento e combinados formam chaves de 1024 bits. Ainda de acordo com Moreno, Pereira e Chiaramonte (2005), com o passar do tempo, a tendência é que o comprimento da chave aumente cada vez mais. Esse fenômeno acontece, em grande parte, pelo avanço nos sistemas computacionais que acompanham o surgimento de computadores que são capazes de fatorar chaves cada vez maiores em um tempo muito baixo.

Os algoritmos para a geração das chaves pública e privada usadas para cifrar e decifrar as mensagens são descritos a seguir (RIVEST; SHAMIR; ADLEMAN, 1978).

1. Escolhem-se dois números primos grandes (**“p”** e **“q”**)
2. Gera-se um número **“n”** por meio da multiplicação dos números escolhidos anteriormente (n = p . q);
3. Escolhe-se um número **“d”**, tal que **“d”** é menor que **“n”** e **“d”** é relativamente primo à (p-1)(q-1)
4. Escolhe-se um número **“e”** tal que (ed-1) seja divisível por (p-1).(q-1). Para realizar esse cálculo é necessário o algoritmo de Euclides estendido.
5. Os valores **“e”** e **“d”** são chamados de expoentes público e privado, respectivamente. O par (**“n”**,**“e”**) é a chave pública e o par (**“n”**,**“d”**), a chave privada. Os valores p e q devem ser mantidos em segredo ou destruídos.

Algoritmo 1: Encriptação básica RSA

Entrada: RSA Chave pública (n,e), texto puro m $\in$ [0, n-1]
Saída: Texto cifrado c
início
1. Computar c = m^e mod n
2. Retorna c.
fim

Algoritmo 2: Decriptação básica RSA

Entrada: RSA chave pública (n,e),RSA chave privada d, Texto cifrado c
Saída: Texto puro m
início
1. Computar m = c^d mod n
2. Retorna m.
fim

O esquema de encriptação e assinatura do RSA utiliza o fato de que

$$m^{ed} \equiv m(mod\, n) \tag{3.1}$$

para todos os inteiros **"m"**. Os esquemas de encriptação e decriptação são apresentados nos algoritmos 1 e 2. A decriptação funciona porque $c^d \equiv (m^e)^d \equiv m(mod\, n)$. A segurança repousa na dificuldade de se computar um texto puro **"m"** a partir de um texto crifrado $c = m^e mod\, n$ e os parâmetros públicos **"n"** e **"e"** (HANKERSON; VANSTONE; MENEZES, 2004).

3.4 Algoritmos Baseados em Curvas Elípticas - ECC

A idéia principal deste tipo de algoritmo criptográfico é construir um conjunto de pontos de uma curva elíptica para o qual o problema do logaritmo discreto seja intratável. Segundo Blake, Seroussi e Smart (1999) sistemas criptográficos baseados em curvas elípticas atingem o mesmo nível de segurança de sistemas como o RSA, mas utilizando chaves menores, e portanto, consumindo menos recursos de memória e processador. Isso os torna ideais para uso em *smart cards* e outros ambientes onde os recursos como armazenamento, processamento, tempo e energia são restritos.

Em meados da década de 80 Koblitz (1987) e Miller (1986) propuseram um método de

criptografia baseado nas curvas elípticas ECC [1]. Segundo os criadores do ECC, uma curva elíptica é uma curva plana definida pela seguinte equação:

$$y^2 = x^3 + ax + b \tag{3.2}$$

A eficiência deste algoritmo baseia-se em encontrar um logaritmo discreto de um elemento aleatório que faça parte de uma curva elíptica. Para se ter uma ideia da aplicabilidade dos algoritmos baseados em curvas elípticas em dispositivos com restrições computacionais Chatterjee, De e Gupta (2011) afirmam que a eficiência do algoritmo criptográfico ECC com tamanhos de chave de aproximadamente 160 bits é a mesma obtida utilizando o algoritmo RSA com chave de 1024 bits. Algoritmos de diversas funcionalidades são baseados em curvas elípticas, incluindo gerenciamento de chaves, encriptação e assinatura digital. Algoritmos de gerenciamento de chaves são utilizados para compartilhar chaves secretas, algoritmos de encriptação habilitam uma comunicação confidencial e algoritmos de assinatura digital autenticam um participante da comunicação além de validar a integridade da mensagem.

Assim, o cenário de utilização dos algoritmos criptográficos de chave pública vêm mudando, pois de acordo com Koc (2009) em termos de criptografia de chave pública o algoritmo RSA continua a liderar o número de implementações, porém o número de aplicações que estão utilizando algoritmos de curvas elípticas está aumentando consideravelmente graças à padronização realizada pelo *National Institute of Standards and Technology* (NIST).

Os algoritmos baseados em curvas são padronizados de acordo com o *American National Standards Institute* (ANSI X9.62), *Federal Information Processing Standard* (FIPS 186-2), *Institute of Electrical and Electronics Engineers* (IEEE 1363-2000) e *International Organization for Standardization* (ISO/IEC 15946-2). Esses padrões definem como usar curvas elípticas sobre corpos primos GF(p) e sobre corpos binários GF(2^m). Nas curvas elípticas GF(p) as operações são realizadas sobre um número primo "p". Nas curvas elípticas GF(2^m) as operações são feitas sobre um polinômio irredutível F(t). Segundo Amin, Jahangir e Rasifard (2008) a criptografia de chave pública inclui algoritmos para acordo de chaves, en-

[1] ECC - Elliptic curve cryptography. Maiores informações sobre o ECC e seus algoritmos em: http://www.certicom.com/index.php/ecc

criptação e assinaturas digitais. Entre os algoritmos que atuam no acordo de chaves pode-se citar o *Elliptic Curve Diffie-Hellman* (ECDH), na encriptação de dados o *Elliptic Curve Integrated Encryption Standard* (ECIES) e na geração de assinaturas digitais o *Elliptic Curve Digital Signature Algorithm* (ECDSA), *Elliptic Curve Pintsov Vanstone* (ECPVS) e *Elliptic Curve Nyberg Rueppel Signatures* (ECNR) (AMIN; JAHANGIR; RASIFARD, 2008).

Os procedimentos de encriptação e decriptação usando curvas elípticas são similares ao esquema de encriptação ElGamal e estão descritos nos algoritmos 3 e 4. O texto puro **"m"** é primeiramente representado como um ponto **"M"**, e depois encriptado através da adição com **"kQ"**, onde **"k"** é um inteiro escolhido randomicamente, e **"Q"** é a chave pública (HANKERSON; VANSTONE; MENEZES, 2004).

Algoritmo 3: *ElGamal elliptic curve encryption*

Entrada: Parâmetros de domínio da curva elíptica (p, E, P, n), Chave pública Q, Texto puro m
Saída: Texto cifrado (C_1, C_2)
início
1. Representar a mensagem m como um ponto M em E (F_p)
2. Selecionar $k \in R^{[1,n-1]}$.
3. Calcular $C_1 = kP$
4. Calcular $C_2 = M + kQ$.
5. Retornar (C_1, C_2)
fim

Algoritmo 4: *ElGamal elliptic curve decryption*

Entrada: Parâmetros de domínio da curva elíptica (p, E, P, n), Chave privada d, Texto cifrado (C_1, C_2)
Saída: Texto puro m
início
1. Calcular $M = C_2 - dC_1$, e extrair m de M.
2. Retornar (m).
fim

O emissor transmite os pontos $C_1 = kPeC_2 = M + kQ$ para o receptor que usa sua chave privada **"d"** para calcular:

$$dC_1 = d(kP) = k(dP) = kQ, \tag{3.3}$$

e depois calcular $M = C_2 - kQ$. Um invasor que deseja fazer a leitura de **"M"** necessita calcular **"kQ"**.

Este modelo de algoritmo tem sido estudado por cientistas, pois segundo Amin, Jahangir e Rasifard (2008), nos últimos anos o ECC têm atraído atenção como solução de segurança para redes sem fio, devido a utilização de pequenas chaves e baixo *overhead* computacional.

3.5 Algoritmo MQQ

Os algoritmos criptográficos apresentados anteriormente têm sua segurança baseada em problemas matemáticos intratáveis computacionalmente: eficiência computacional do cálculo do logaritmo discreto e fatoração de inteiros (GLIGOROSKI; MARKOVSKI; KNAPSKOG, 2008). Em 2008 foi criado um novo esquema de chave pública denominado multivariado quadrático quase grupo, mais conhecido como MQQ. Segundo Gligoroski, Markovski e Knapskog (2008) este algoritmo baseia-se em polinômios multivariados quadráticos e transformações de quase grupos, possuindo as seguintes propriedades.

1. O MQQ é um algoritmo Pós-Quântico. Quanto a essa propriedade, segundo Nielsen e Chuang (2010) não está longe o momento em que computadores quânticos serão concebidos para resolver problemas práticos e serão utilizados para quebrar muitos algoritmos criptográficos existentes. Quando esse momento chegar, será necessário o projeto de sistemas criptográficos baseados em problemas intratáveis num computador quântico.

2. Na encriptação a velocidade é comparável a outros criptosistemas de chave pública baseados em multivariados quadráticos (MAIA, 2010).

3. Na decriptação a velocidade é de uma típica cifra de bloco simétrica (MAIA; BARRETO; OLIVEIRA, 2010).

4. Altamente paralelizável ao contrário de outros algoritmos que são essencialmente sequenciais.

Segundo El-Hadedy, Gligoroski e Knapskog (2008) experimentos realizados em hardware mostraram que o MQQ pode ser tão rápido quanto uma típica cifra de bloco simétrica, sendo várias ordens de grandeza mais rápido que os algoritmos RSA e ECC. Este fato foi

confirmado por Gligoroski et al. (2010) quando concluiu que em *software* a assinatura digital realizada pelo MQQ é de 300 á 7.000 vezes mais veloz que a assinatura dos algoritmos RSA e ECC. Já em *hardware*, a superioridade do MQQ pode chegar a 10.000 vezes. Além disso, segundo Ahlawat, Gupta e Pal (2009), o algoritmo MQQ dá uma nova direção para o campo da criptografia, podendo ser utilizado para desenvolver novos criptosistemas de chave pública, bem como melhorar esquemas criptográficos existentes.

Algoritmo 5: Mapeamento não linear

Entrada: Inteiro n onde n = 5k, k$\geq$ 28 e o vetor x = $(f_1, ..., f_n)$ de n funções Booleanas lineares com n variáveis

Saída: Oito quase-grupos $*_1, ..., *_8$ e n polinômios multivariados quadráticos $P'_i(x_1, ..., x_n)$, i = 1,...,n

início

Pré Processamento Chamar os procedimentos MQQ(4,1) e MQQ(5,0) gera dois grandes conjuntos $Quad_4Lin_1$ e $Quad_5Lin_0$ (com mais de 2^{20} elementos cada conjunto) de MQQ do tipo $Quad_4Lin_1$ e do tipo $Quad_5Lin_0$ tal que a classificação mínima dos seus polinômios quadráticos quando representado na forma de matriz é de no mínimo 8; Transformar por permutação as coordenadas de todos quase-grupos no conjunto $Quad_4Lin_1$ tal que a primeira coordenada seja linear.

1. Represente um vetor $x = (f_1, ..., f_n)$ de n funções booleanas lineares de n variáveis $x_1, ..., x_n$ como string x = $X_1, ..., X_k$ onde X_i são vetores de dimensão 5.

2. Escolha aleatoriamente diferentes quase-grupos $*_1, *_2 \in Quad_4Lin_1$ e diferentes quase-grupos $*_3, *_4, *_5, *_6, *_7, *_8 \in Quad_5Lin_0$

3. Defina uma tupla *I* = $(i_1, i_2, ..., i_{k-1})$ onde $i_j \in 1, 2, ..., 8$ que seria usado como um conjunto de índices para determinar qual quase-grupo seria utilizado na transformação não linear de y. O requisito para este conjunto de índices é que o número total de índices que são referenciados por um quase-grupo de classe $Quad_4Lin_1$ seja 8.

4. y = $Y_1...Y_k$ onde $Y_1 = X_1, Y_{j+1} = X_j * X_{j+1}$, para j = 1,2,...,k-1.

5. Z = $Y_1 \| Y_{\mu 1}1 \| Y_{\mu 2}1 \| ... \| Y_{\mu 8}1$ onde todos 13 componentes são funções booleanas lineares. A notação $Y_{\mu j}1$ significa a primeira coordenada do vetor $Y_{\mu j}$.

6. Transforme Z por bijeção de Dobbertin: W = Dob(Z).

7. $Y_1 = (W_1, W_2, W_3, W_4, W_5), Y_{\mu 1,1} = W_6, Y_{\mu 2,1} = W_7, Y_{\mu 3,1} = W_8, Y_{\mu 4,1} = W_9, Y_{\mu 5,1} = W_{10}, Y_{\mu 6,1} = W_{11}, Y_{\mu 7,1} = W_{12}, Y_{\mu 8,1} = W_{13}$

8. Saída: Quase-grupos $*_1, ..., *_8$ e y como n polinômios multivariados quadráticos $P'_i(x_1, ..., x_n), i = 1, ..., n$

fim

Uma descrição genérica para o esquema MQQ é um típico sistema multivariado quadrático $T \circ P' \circ S : \{0,1\}^n \rightarrow \{0,1\}^n$ onde T e S são duas transformações lineares não singulares e P' é um mapeamento bijetivo multivariado quadrático sobre $\{0,1\}^n$. O mapea-

Algoritmo 6: Algoritmo MQQ para decriptar e assinar com chave privada T, S, *1,...,*8

Entrada: Um vetor $y = y_1, ..., y_n$
Saída: Um vetor $x = x_1, ..., x_n$ tal que P(x) = y
início
1. y' = $T_{-1}(y)$.
2. W = $y'_1, y'_2, y'_3, y'_4, y'_5, y'_6, y'_{11}, y'_{16}, y'_{21}, y'_{26}, y'_{31}, y'_{36}, y'_{41}$.
3. Z = $Z_1, Z_2, Z_3, Z_4, Z_5, Z_6, Z_7, Z_8, Z_9, Z_{10}, Z_{11}, Z_{12}, Z_{13} = Dob^{-1}(W)$.
4. $y'_1 \longleftarrow Z_1, y'_2 \longleftarrow Z_2, y'_3 \longleftarrow Z_3, y'_4 \longleftarrow Z_4, y'_5 \longleftarrow Z_5, y'_6 \longleftarrow Z_6, y'_{11} \longleftarrow Z_7, y'_{16} \longleftarrow Z_8, y'_{21} \longleftarrow Z_9, y'_{26} \longleftarrow Z_{10}, y'_{31} \longleftarrow Z_{12}, y'_{41} \longleftarrow Z_{13}$.
5. $y' = Y_1, ..., Y_k$ onde Y_i são vetores de dimensão 5.
6. Sendo $*_i$, i = 1,...,8, obter x' = $X_1, ..., X_k$, de modo que,
$X_1 = Y_1, X_2 = X_{1\backslash 1}Y_2, X_3 = X_{2\backslash 2}Y_3, X_1 = X_{i-1_{\backslash 3+((i+2)mod6)}}Y_i$
7. x = $S^{-1}(x')$
fim

mento $P' : \{0,1\}^n \rightarrow \{0,1\}^n$ é definido no algoritmo 5.

O algoritmo para encriptação com a chave pública é a aplicação direta do conjunto de n polinômios multivariados $P = \{P_i(x_1, ..., x_n)|i = 1, ..., n\}$ sobre o vetor $x = (x_1, ..., x_n)$, ou seja $y = P(x)$. Podendo ser representada como $y = P(x) \equiv y \equiv A.X$. No algoritmo 6 descreve-se a decriptação e assinatura usando a chave privada $(T, S, *_1, ..., *_8)$.

Os trabalhos de Maia (2010) e Maia, Barreto e Oliveira (2010) avaliaram os tempos de encriptação e decriptação dos algoritmos MQQ e RSA nas plataformas MicaZ e TelosB, sendo que o MQQ exibiu os tempos de 825,1ms para encriptar e 116,6 ms para decriptar no TelosB e 445 ms para decriptar no MicaZ. Ainda de acordo com Maia (2010), o MQQ 160 bits é 909 vezes mais rápido na encriptação e 5470 vezes mais rápido na decriptação quando comparado ao RSA 8 bits.

3.6 Considerações Finais do Capítulo

O melhor algoritmo conhecido para fatoração de inteiros é o *general number field sieve* (GNFS) , que leva tempo $O(e^{(\frac{64}{9})\frac{1}{3}(n.log2)\frac{1}{3}(log(n.log2))\frac{2}{3}})$ para fatorar um inteiro de n-bit. No entanto, o melhor algoritmo quântico conhecido para este problema, o algoritmo de Shor, é executado em tempo polinomial. Infelizmente, este fato não diz muito sobre onde está o problema com relação a classes de complexidade não-quântica (PEREIRA, 2008). Para

Torres (2007) os problemas de fatoração de inteiros (RSA) admitem, em geral, algoritmos que executam em tempo sub-exponencial.

Segundo Oliveira (2010), o melhor algoritmo para resolução do problema do logaritmo discreto (ECC) tem complexidade exponencial, o que confere um alto grau de segurança ao sistema. Esse fato é confirmado por Gouvea (2010a) quando informa que um subgrupo gerado por um ponto cuja ordem r tem k bits fornece $\frac{k}{2}$ bits de segurança, tal valor está relacionado ao fato que o melhor algoritmo para resolver o problema do logaritmo discreto em uma curva elíptica possui complexidade $O(2^{k/2})$. Segundo Stapleton (2012), ao contrário do caso do problema de fatoração de inteiros, não há algoritmo de tempo sub-exponencial conhecido para o problema de logaritmo discreto sobre curvas elípticas. O melhor algoritmo conhecido até o momento tem tempo exponencial. Realizar as operações necessárias para soma de pontos em curvas elípticas é um processo mais lento do que efetuar a exponenciação módulo um primo, que é a operação utilizada nos sistemas tradicionais. No entanto, como o problema do logaritmo discreto para curvas elípticas é mais complexo, o mesmo nível de segurança pode ser conseguido com uma chave menor, portanto, uma chave menor implica em operações mais rápidas, o que na prática compensa a maior complexidade das operações.

Segundo Faugere et al. (2010) a complexidade computacional do algoritmo MQQ é polinomial com grau de regularidade D_{reg}, mais precisamente a complexidade é: $O(N^{\omega D_{reg}})$, que consiste basicamente na complexidade em se reduzir uma matriz de tamanho $\approx N^{D_{reg}}$.

A utilização de sistemas embarcados vêm crescendo rapidamente nos últimos anos, dispositivos como *mobile phones*, PDAs, *smart cards*, e mais recentemente RFID e sensores estão cada vez mais presentes em nossas vidas. Grande parte dos dispositivos citados têm baixa capacidade computacional, porém são implantados em ambientes que geralmente requerem segurança na transmissão de dados. No que tange a criptografia de chave pública em RSSF, o algoritmo RSA é o mais utilizado atualmente, está padronizado e consegue uma eficiência satisfatória. Os algoritmos baseados em curvas elípticas e hiperelípticas têm sido estudados no meio acadêmico como alternativa ao RSA, e os resultados mostram que é possível alcançar bons resultados com chaves menores. O algoritmo MQQ foi desenvolvido recentemente e apresentou resultados expressivos quando comparado com RSA e ECC, tomando como parâmetros a autenticação e a assinatura digital. O algoritmo MQQ é pós-quântico, e pode ser

uma boa solução mesmo quando a computação quântica estiver padronizada. Apesar dos resultados satisfatórios do MQQ frente ao RSA e ECC, não existe ainda um trabalho avaliação de desempenho específico para encriptação e decriptação de dados, tampouco um trabalho de avaliação que leve em consideração os algoritmos ECC e MQQ em plataformas usadas em RSSF. O próximo capítulo mostra o que vem sendo feito no meio acadêmico como forma de apontar algoritmos criptográficos eficientes em RSSF.

Capítulo 4

Segurança em RSSF - Estado da Arte

A disseminação da computação ubíquia motivou pesquisadores na investigação de técnicas de segurança que pudessem garantir privacidade, integridade e disponibilidade na troca de dados entre dispositivos de baixa capacidade computacional. A maioria das avaliações de desempenho levavam em consideração apenas algoritmos criptográficos de chave privada, pois acreditava-se que este tipo de cifra era mais adequada a dispositivos com restrições computacionais. Porém, atualmente, observa-se uma certa tendência a utilização de algoritmos de chave pública, uma vez que pesquisas recentes mostraram que algumas cifras de chave públicas alcançam a mesma eficiência das cifras simétricas utilizando chaves menores. Este capítulo apresenta as principais avaliações de desempenho que vêm sendo feitas nos últimos anos tendo como foco os algoritmos RSA, ECC e MQQ aplicados em RSSF. As avaliações estão divididas por tema. O primeiro tema aborda segurança em RSSF (seção 4.1). O segundo descreve avaliações de desempenho de algoritmos criptográficos simétricos (seção 4.2). O terceiro tema descreve as avaliações de desempenho de algoritmos criptográficos de chave pública em RSSF (seção 4.3), e finalmente o quarto tema aborda sobre avaliações de desempenho de protocolos de segurança em RSSF (seção 4.4).

4.1 Segurança em RSSF

Os autores Boyle e Newe (2008) descreveram em seu trabalho que a utilização de redes de sensores sem fio (RSSF) tende a crescer consideravelmente nos próximos anos devido sua aplicação em todas as áreas que permitem sensoriamento de dados. Em muitos casos

essas redes necessitam de segurança, e portanto o trabalho deles comparou arquiteturas de segurança disponíveis para RSSF.

Os autores Boyle e Newe (2008) definiram rede de sensores sem fio como sendo um grupo de nós independentes se comunicando sem fio através de frequência e largura de banda limitada. A diferença dessa rede para as tradicionais é que as RSSF necessitam de implantação e coordenação para executar suas tarefas. Devido às restrições computacionais dos sensores, tópicos como gerenciamento de energia, roteamento e segurança estão sempre em evidência em pesquisas recentes.

É notório que quanto mais complexa é a técnica de segurança implantada em um sistema, mais recursos são exigidos. No que diz respeito às redes de sensores, limitações de recursos como baixa capacidade de memória, processamento, além de fonte finita de energia fazem com que as técnicas de criptografia aplicáveis nessas redes tenham que ser implementadas visando consumir o mínimo possível de recursos. Segundo Boyle e Newe (2008) a segurança deve estar integrada em todos os nós de uma rede de sensores, caso contrário um integrante desprotegido pode se tornar uma brecha para um eventual ataque.

Ainda segundo Boyle e Newe (2008) a criptografia é método padrão para defender uma rede de sensores da maioria dos ataques possíveis, e os vários níveis de criptografia implicam em vários níveis de *overhead* na forma de crescimento do tamanho do pacote de dados, tamanho do código, uso de processador, memória, etc. A escolha de um algoritmo eficiente para rede de sensores continua sendo o grande debate entre pesquisadores da área.

As áreas militar e de saúde são críticas em termos de segurança, neste sentido os autores Boyle e Newe (2008) citaram que pesquisas têm sido realizadas pelo *Defense Advanced Research Projects Agency* (DARPA) a fim de desenvolver transferência segura de dados em comunicações de redes de sensores. Além disso, aplicações onde os sensores estão implantados no corpo humano, monitorando sinais vitais e enviando dados para profissionais de saúde também requerem segurança no transporte de dados. Além das áreas citadas, várias outras aplicações de aplicação das redes de sensores como indústria, comércio, automação residencial requerem algum nível de segurança implantado.

Os autores Boyle e Newe (2008) citaram brevemente as formas de comunicação dos sen-

sores, destacando *Bluetooth, ZigBee* e o padrão IEEE 802.15.4. Devido à baixa utilização do *Bluetooth* em 2008, os autores não relataram nenhuma primitiva de segurança aplicada nesse padrão, porém trabalhos atuais como (TAN; MASAGCA, 2011) mostram que o *Bluetooth* vêm sendo bastante utilizado e necessita de técnicas de segurança como qualquer outro método de comunicação. Em relação ao IEEE 802.15.4, existe uma gama de *suites* de segurança para esse padrão, destacando o algoritmo de encriptação AES definido pelo NIST provendo controle de acesso e encriptação de dados, além da autenticação realizada usando cifra de bloco CBC-MAC. As chaves criptográficas podem ser de 32, 64 e 128 bits. O padrão *ZigBee* adota algumas funções extras de segurança, porém é baseado no padrão IEEE 802.15.4.

Segundo Boyle e Newe (2008) existe um grande número de ataques contra redes de sensores. Esses ataques podem ser classificados como: ataque contra a privacidade dos dados, negação de serviço, ataques de replicação e ataques físicos. Algumas características dos principais ataques a RSSF são listadas a seguir:

- **DOS -** Ataque que pode assumir várias formas, minando a capacidade da rede através de inundação de dados.
- **Sybil -** Consiste em injetar um nó malicioso que pode assumir diversas identidades, podendo desviar o curso normal dos pacotes na rede.
- **Segurança física -** Devido a impossibilidade de monitoramento dos nós em alguns ambientes, estes são suscetíveis a ataques físicos como sequestro, alteração ou destruição de dispositivos.

A tabela 4.1 ilustra um comparativo das arquiteturas de segurança para redes de sensores, e nela é possível observar que a maioria das soluções utilizam criptografia simétrica.

Tabela 4.1: Comparação de Arquiteturas de Segurança para RSSF. [Boyle e Newe 2008]

	Encriptação	Cifra	Tamanho de código	Custo Energético	Atualização
SPINS	CTR-mode	RC5	2674B (máx)	7,2ms / 20%	2002
TINYSEC	CBC	Skipjack	7146B (máx)	0,38ms / 9,1%	2004
LEAP / LEAP++	RC5	RC5	17,8Kb	Variável	2006
SM	ECC	N/A	N/A	Variável	2006
ZigBee	AES	AES 128	N/A	Sob investigação	2005

Os autores Margi, Jr e Barreto (2009) realizaram uma abordagem acerca da segurança

em RSSF, na qual conceituaram os componentes que fazem parte das principais plataformas e relacionaram os principais ataques que uma rede de sensores pode sofrer, levando em consideração as vulnerabilidades deste tipo de rede. Os autores também definiram os principais algoritmos de criptografia e protocolos de autenticação que podem ser aplicados em redes de sensores, além disso descreveram testes de mecanismos de segurança. Segundo Margi, Jr e Barreto (2009) as redes de sensores são formadas por dispositivos de baixa capacidade computacional, com recursos limitados de processamento, memória, comunicação e energia. Os nós sensores TelosB e MicaZ, ambos da Crossbow, têm sido amplamente utilizados em *testbeds* e implantação de RSSF.

Em uma RSSF, diferentes aplicações requerem diferentes níveis de segurança. Na área da saúde por exemplo, os dados devem trafegar de forma criptografada para evitar a leitura por parte de um atacante, além disso deve existir um mecanismo de autenticação para confirmar a identidade dos nós participantes da rede e da informação que está sendo transmitida. Segundo Margi, Jr e Barreto (2009) além dos serviços de segurança como integridade dos dados, confidencialidade, autenticidade e disponibilidade, é importante também garantir que os dados enviados sejam recentes *(data freshness)*, ou seja, que nenhum intruso está replicando dados antigos. Esse tipo de ataque é conhecido como ataque de reprodução.

As vulnerabilidades das RSSF podem estar em todas as camadas da pilha de protocolo de rede destes dispositivos, portanto brechas podem aparecer na camada física, MAC, rede ou aplicação. Neste contexto, em (MARGI; JR; BARRETO, 2009) informa-se que os principais ataques a RSSF envolvem a captura de nós, o manuseio indevido dos mesmos *(tampering)* e negação de serviço (com a criação de buracos negros, *wormholes*, introdução de desvios ou *loops*, sequestro de nós e criação de problemas na cobertura da aplicação).

Em relação ao esquema de gerenciamento de chaves para RSSF, deve-se priorizar a flexibilidade, uma vez que este tipo de rede pode variar em relação à quantidade de nós. Entre os principais esquemas de distribuição destacam-se os de pré-distribuição onde as chaves são distribuídas antes mesmo da rede ser montada, esquemas-arbitrários em que a responsabilidade do gerenciamento fica por conta de dispositivos mais robustos computacionalmente, como estações-base ou *cluster-heads*, e por fim esquemas auto-regulados que utilizam algoritmos assimétricos para estabelecimento de chaves entre pares de forma dinâmica, após a

implantação da rede. Segundo Margi, Jr e Barreto (2009) esta estratégia elimina a existência de pontos centrais confiáveis que se tornariam alvos preferenciais de ataques. Ainda de acordo com Margi, Jr e Barreto (2009) o recente desenvolvimento de soluções assimétricas bem mais leves do que algoritmos tradicionais, como é o caso da criptografia de curvas elípticas (ECC), fez com que os esquemas auto-regulados passassem a receber uma maior atenção por parte dos pesquisadores. Margi, Jr e Barreto (2009) deixaram uma contribuição sobre o uso de algoritmos assimétricos em RSSF afirmando que ainda era cedo para dizer que todas as questões de eficiência relacionadas aos esquemas auto-regulados foram resolvidas. No entanto, os avanços recentes nesta área deixam claro o grande potencial desta abordagem, que tende a ocupar um espaço cada vez mais importante dentre as soluções preferenciais de gerenciamento de chaves em RSSF. Margi, Jr e Barreto (2009) também enfatizaram os desafios de uma análise de desempenho de algoritmos e mecanismos de segurança em RSSF. Os autores afirmaram que uma análise simples, focada no nó sensor, medirá o tempo e o consumo de energia de transmissão e recepção de um pacote maior. Porém, analisar o impacto de algoritmos e mecanismos de segurança para a RSSF como um todo não é trivial. Esta análise é dependente da aplicação, dos protocolos (enlace, roteamento, sincronização, localização, vizinhança) em uso na RSSF, e em qual camada os mecanismos estão sendo aplicados

Os autores Margi, Jr e Barreto (2009) afirmaram que devido a limitação de recursos dos sensores, os mecanismos de segurança devem ser escaláveis, em termos de energia e atraso. Além disso, pelo fato de terem desenvolvido o algoritmo Curupira (SIMPLICIO, 2008), os autores focaram o trabalho na análise de cifras simétricas, deixando espaço para trabalhos futuros focados na análise de cifras de chave pública.

O trabalho de Ren et al. (2011) discutiu os desafios de segurança, modelos de ameaças e soluções existentes para redes de sensores sem fio móveis (*Mobile Wireless Sensor Networks* - MWSNs). Além disso, os autores apresentam pesquisas que estão sendo desenvolvidas nessa área e questões abertas passiveis de exploração. A principal diferença entre WSN e MWSN é que em MWSNs o nó *sink* é móvel, portanto a maior parte do trabalho da rede é feita por esse dispositivo, fazendo com que haja economia de energia nos dispositivos sensores que integram a rede (MUNIR et al., 2007). Devido a natureza aberta de uma MWSN, os

autores afirmaram que os ataques podem ser realizados diretamente no sensor, danificando o dispositivo, espionando a frequência de rádio utilizada na comunicação, alterando suas configurações originais ou mesmo implantando nós para atrapalhar o correto funcionamento da rede. A falha de um nó em uma rede MWSN é considerada mais prejudicial que em uma rede WSN devido ao armazenamento de dados na memória do sensor. Alguns requisitos de segurança como: confidencialidade, integridade, autenticação, controle de acesso, disponibilidade, não repudio, auditoria, privacidade e anonimato, foram descritos por Ren et al. (2011) como primordiais em uma rede MWSN.

Em relação à autenticação dos dados, Ren et al. (2011) informaram que o uso de algoritmos criptográficos assimétricos não é viável devido à restrição de recursos computacionais dos sensores. Os autores citam ainda o protocolo uTesla de autenticação que funciona em WSN, mas não têm a mesma eficiência em MWSN pois boa parte do processamento fica por conta das estações base, que não estão presentes durante todo o tempo de vida da rede em MWSN. Para alcançar eficiência computacional, encaminhamento seguro e autenticação, os autores Ren et al. (2011) sugerem um novo esquema de assinatura digital desenvolvido por Yavuz e Ning (2009) e conhecido como *Hash-based Sequential Aggregate and Forward Secure Signatures (HaSAFSS)*.

Esse esquema pode usar chaves simétricas, *Sym-HaSAFSS*, ou utilizar o algoritmo ECC, conhecido como *ECC-HaSAFSS*. A propriedade principal desse método é criptografar dados de forma que as partes envolvidas na comunicação não podem decriptar os dados antes de um tempo pré-definido, portanto mesmo que um remetente seja comprometido, o atacante não conseguirá obter informações daquele dispositivo. Para prover controle de acesso, os autores Ren et al. (2011) citaram a criptografia de chave simétrica e a criptografia de chave pública como alternativas, porém na chave pública os autores criticam a necessidade de se armazenar muitas chaves, causando um *overhead* de armazenamento e um *overhead* computacional na rede. Uma solução para este problema é proposta por Asztor et al. (2008) denominada *Fine-grained Distributed data Access Control (FDAC)*. Neste esquema os dados cifrados são associados a uma série de atributos e a chave privada associada a uma estrutura de acesso. Os dados somente podem ser decriptados se seus atributos satisfazerem a estrutura de acesso da chave privada.

Outras questões de segurança em MWSN foram expostas pelos autores Ren et al. (2011), porém não dizem respeito a criptografia de chave pública. Uma questão relevante que os autores colocaram como primordial é o desenvolvimento de mecanismos de segurança para MWSN que consuma o mínimo de energia possível, uma vez que os dispositivos desse tipo de rede geralmente estão implantados em área de difícil acesso e sem nenhuma fonte externa de energia. Outra questão em aberto citada pelos autores é o desenvolvimento de controle de acesso de dados baseado em chave simétrica que seja eficiente, neste contexto o autor considera que as soluções baseadas em chaves públicas consomem grande quantidade de recursos e portanto não são indicadas para MWSN.

4.2 Estudo de Algoritmos Simétricos

Os autores Margi, Jr e Barreto (2009) relacionaram algumas cifras simétricas dedicadas a sistemas embarcados, entre elas destacam-se CURUPIRA, PRESENT, HIGHT, SEA, mCrypton, Trivium e Grain. O projeto destes algoritmos leva em consideração as limitações dos sensores, portanto trabalham com operações simples e fazem com que estas soluções sejam mais eficientes e compactas que algoritmos de uso geral. Diversos trabalhos de análise de desempenho foram realizados a fim de comparar as cifras, porém segundo Margi, Jr e Barreto (2009) estes trabalhos são muitas vezes distintos em termos de metodologia, plataforma, métricas e foco da análise, o que dificulta uma comparação direta entre os resultados obtidos. Porém, através destas análises é possível "filtrar"grupos de algoritmos que satisfaçam necessidades de um ambiente específico.

A autenticação pode ser importante em algumas aplicações de RSSF, uma vez que a introdução de dados falsos podem levar a consequências graves. Segundo Margi, Jr e Barreto (2009) são raras as soluções otimizadas de autenticação, com baixo custo de memória, processamento e livre de patentes. Os autores citam ALRED e o modo Cipher-State como esquemas de autenticação voltados para RSSF. Uma análise de desempenho de mecanismos de autenticação foi realizada por Bauer, Potisk e Tillich (2009) onde os autores verificaram a ocupação de memória RAM de quatro esquemas de *Authenticated Encryption with Associated Data* (AEAD): OCB, EAX, *Galois/Counter Mode of Operation* (GCM) e *Counter-*

CipherFeedBack with Header (CCFB+H), todos usando o AES como cifra de bloco subjacente. O CCFB+H foi indicado pelos autores como mais eficaz em RSSF devido ao melhor desempenho e melhor uso de memória para diferentes tamanhos de mensagens. Devido à escassez de análises de desempenho de algoritmos MAC, os autores Margi, Jr e Barreto (2009) realizaram um estudo comparativo com os esquemas *cipher-based MAC* (CMAC), *Parallelizable MAC* (PMAC), *Galois Message Authentication Code* (GMAC) e Marvin. Porém, utilizando o Curupira como cifra de bloco subjacente.

O número de aplicações envolvendo RSSF vêm crescendo consideravelmente nos últimos anos, portanto faz-se necessário análises de técnicas de segurança mais eficientes neste tipo de rede. O trabalho desenvolvido por Cavalcante et al. (2011) avaliou o desempenho dos algoritmos criptográficos Skipjack e RC5 objetivando determinar o mais eficiente para RSSF.

Os algoritmos Skipjack e RC5 são simétricos, porém são amplamente utilizado em RSSF. Segundo Cavalcante et al. (2011) a escolha destes algoritmos é justificada pelo fato de que ambos foram desenvolvidos para dispositivos com capacidade limitada de processamento e memória. O Skipjack trabalha com blocos de 64 bits, chave de 80 bits e 32 *rounds*. Já o RC5 trabalha com blocos de 32, 64 ou 128 bits, tamanho de chave variável ente 0 e 2048 bits, e *rounds* variáveis entre 0 e 255.

Para avaliar o desempenho dos algoritmos, os autores Cavalcante et al. (2011) desenvolveram uma aplicação na linguagem nesC chamada criptotest para o sistema operacional TinyOS 2.1. Essa aplicação têm a função de medir o tempo de execução dos algoritmos criptográficos usando uma interface onde é possível enviar uma mensagem criptografada com Skipjack, RC5 ou sem segurança. Os algoritmos criptográficos foram baseados no TinySec com algumas modificações para serem executados no TinyOS 2.X. As métricas utilizadas foram quantidade de memória RAM e ROM, tempo de execução e consumo de energia. Segundo Cavalcante et al. (2011) essas métricas foram escolhidas levando-se em consideração as principais limitações de recursos computacionais em RSSF. A plataforma escolhida foi a MicaZ com 4KB de memória RAM e 128KB de memória ROM.

A quantidade de memória RAM e ROM consumida pelos algoritmos foi obtida a partir do processo de compilação do código no TinyOS. O próprio sistema operacional informa

a quantidade de memória consumida na compilação. As medições do tempo de execução foram realizadas com auxílio de osciloscópio digital conectado ao MicaZ. O consumo de energia foi obtido usando fórmulas de potência e energia consumida, onde:

$$Potencia = TensaoBateria * correnteOperacao \tag{4.1}$$

$$EnergiaConsumida = potencia * TempoExecucao \tag{4.2}$$

Os resultados mostraram que o Skipjack requer 7% mais memória ROM que o algoritmo RC5, representando 16% da quantidade de memória ROM disponível na plataforma. Por outro lado o RC5 requer 9% mais memória RAM que o Skipjack representando 25% da quantidade de memória disponível no MicaZ. Considerando o tempo de encriptação e decriptação o RC5 apresentou desempenho 1% melhor que o Skipjack. Em relação ao consumo de energia, o RC5 consome 1% menos energia que o Skipjack nas operações de encriptação e decriptação.

4.3 Estudo de Algoritmos Assimétricos

Uma RSSF pode ser considerada um tipo especial de rede Ad-Hoc, porém, aplicações de segurança não podem ser aplicadas diretamente nessa rede devido a limitação de seus recursos computacionais. O uso de RSSF está em ascensão nos mais variados tipos de aplicações. As características de um sistema amplamente distribuído em que os nodos participantes têm recursos limitados de processamento, memória e energia fazem com que protocolos e arquiteturas de segurança devam ser projetados especificamente para atender esse tipo de rede. Alguns pesquisadores não recomendam a criptografia de chave pública para dispositivos de baixa capacidade computacional por ser considerada dispendiosa, ou seja, necessitar de dispositivos com relativa potência computacional.

A criptografia de chave pública pode garantir um alto grau segurança e também maior flexibilidade e gerenciamento que a criptografia de chave privada. Os algoritmos *Rivest-Shamir-Adelman* (RSA) e *Elliptic Curve Cryptography* (ECC) são utilizados em sistemas de segurança, sendo que o ECC é baseado na estrutura algébrica de curvas elípticas sobre cam-

pos finitos. O algoritmo ECC é promissor em RSSF devido sua maior velocidade utilizando menores tamanhos de chaves que o RSA.

É cada vez maior o número de dispositivos computacionais, especificamente sensores, que vem sendo utilizados nas mais diversas áreas como agricultura, saúde e climatologia. A implantação desses dispositivos em ambientes acessíveis gera uma vulnerabilidade de segurança que deve ser minimizada através de técnicas de segurança, como a criptografia. Aspectos como autenticação, autorização, confidencialidade e integridade podem ser garantidos através da criptografia na transferência dos dados. A criptografia simétrica pode ser utilizada em dispositivos de baixa capacidade computacional, porém é inflexível no gerenciamento de chaves, diferentemente da criptografia de chave pública, que é flexível, porém requer maior poder computacional.

4.3.1 ECC vs RSA

Os autores Gura et al. (2004) realizaram um estudo comparativo dos algoritmos criptográficos RSA e ECC aplicados nos microcontroladores de 8-bits Chipcon CC1010 e Atmel AVR Atmega128. Gura et al. (2004) afirmaram que comparado com RSA, o ECC apresenta bons resultados em relação a tamanho de chave, velocidade de processamento, consumo de memória e energia, sendo recomendado para dispositivos com restrições computacionais. Um coprocessador poderia ser adicionado para tratar especificamente de criptografia, porém isso geraria aumento de custo nos dispositivos. O trabalho de Gura et al. (2004) teve como foco aspectos de implementação do RSA e ECC padronizados pelo NIST/SECG e avaliação destes algoritmos em relação a performance, tamanho de código e uso memória.

A operação fundamental do RSA é exponenciação modular em inteiros, e sua segurança está na dificuldade de fatoração de inteiros grandes. O ECC tem sua segurança advinda da rigidez do problema logarítmico da curva elíptica discreta. Somente algoritmos exponenciais são conhecidos para fatorar números em ECDLP. Isso permite ao ECC alcançar bons níveis de segurança com chaves pequenas. No estudo de Gura et al. (2004) foram avaliados e comparados os algoritmos ECC-160 com RSA-1024 e ECC-224 com RSA-2048.

A operação fundamental do ECC é a multiplicação por pontos, que é definida por finitas

operações. Todas as operações são definidas a partir de números primos inteiros ou *binary polynomial fields*. Várias técnicas têm sido propostas com o intuito de acelerar o ponto de multiplicação do ECC. Os autores Gura et al. (2004) mostraram as principais técnicas aplicáveis ao ECC padronizado pelo NIST e SECG.

Entre os métodos citados pelos autores destacam-se o método *Non–Adjacent Forms* que trabalha com a recodificação de k em um ponto de multiplicação kp para reduzir o número de *non-zero bits* e portanto o número de pontos de adição. Outro método é o *Curve-Specific Optimations* que otimiza o desempenho do ECC. Como resultado dessas otimizações, o ECC apresentou relativa vantagem frente ao RSA em processadores de 8 bits.

Os autores informam que a multiplicação modular e radiciação de inteiros grandes são pontos críticos para RSA e ECC. Algumas estratégias de otimização da multiplicação escalar como *Row-Wise Multiplication, Column-Wise Multiplication, Hybrid Multiplication* são sugeridas para aumentar a performance dos algoritmos estudados.

Os resultados mostraram que o ponto de multiplicação do ECC-160 é 2 vezes mais rápido que o RSA-1024. Além disso foi constatado pelos autores que a relativa vantagem do ECC aumentava a medida que o tamanho de palavra do processador diminuía. Técnicas que reduzem o número de multiplicações fizeram com que o ECC tivesse um aumento de performance na ordem de 25% quando aplicado na plataforma AVR. Apesar do excelente trabalho desenvolvido por Gura et al. (2004), os autores não avaliaram questões importantes no desempenho de algoritmos criptográficos, como tempo de processamento.

Em relação ao gerenciamento de chaves, os autores Amin, Jahangir e Rasifard (2008) informam que esta é a parte do sistema criptográfico no qual as chaves são geradas, armazenadas, protegidas, transferidas, lidas, usadas e destruídas, portanto é de fundamental importância no estabelecimento de uma política de segurança em redes de sensores sem fio. Neste contexto o trabalho de (AMIN; JAHANGIR; RASIFARD, 2008) comparou o custo energético de algoritmos de chave pública, bem como esquemas de gerenciamento de chaves baseados em criptografia de chave pública. Foram implantados de 250 a 640 sensores em uma topologia randômica. A plataforma utilizada foi a Mica2dot baseada no microntrolador ATmega128L da ATMEL. Os autores Wander et al. (2005) proveram medições detalhadas da plataforma Mica2dot, especificamente através do consumo de energia para operações de

geração/verificação de assinaturas no cliente e também no servidor.

Os autores Amin, Jahangir e Rasifard (2008) utilizaram o modelo de consumo energético apresentado por Wander et al. (2005) para estimar o tempo consumido na plataforma Mica2dot. Esses dados foram utilizados para estimar o tempo e o consumo energético para todos os outros sensores da rede. Os possíveis modos da plataforma, bem como seu consumo energético foram obtidos usando a tabela criada em (CORPORATION, 2006). Nesta tabela, visualizam-se os modos Ativo, Inativo e Econômico, bem como seus respectivos valores de consumo de potência. Um exemplo prático do uso da tabela foi dado: Se o microcontrolador está funcionando em modo Ativo, a 8Mhz e 3.3V, a energia consumida é: 3.3V * 10mA(tabela) = 33mW, logo 33mW / 8Mhz = 4.124nW/clock por ciclo (nWs). Os tempos gastos para geração de assinaturas, verificação e troca de chaves entre clientes e servidores quando o consumo de potência ativo é igual a 13.8mJ (PIOTROWSKI; LANGENDOERFER; PETER, 2006).

Baseado nas informações em (CORPORATION, 2006) e (PIOTROWSKI; LANGENDOERFER; PETER, 2006) os autores calcularam o consumo energético para geração de assinaturas e verificação/troca de chaves, porém com consumo de potência ativo em 33mJ. O resultado final é apresentado na Tabela 4.2, onde se observa e pode-se constatar que o RSA não é indicado para RSSF.

Tabela 4.2: Estimativa de consumo de potência (mWs) dos algoritmos RSA e ECC (AHMAD; BEG; ABBAS, 2010)

Algoritmo	Tamanho de Chave	Troca de Chave		Assinatura	
		Cliente	Servidor	Entrada	Verificação
RSA	1024	39,96	726,99	726,99	28,38
	2048	136,62	5.506,06	5.506,06	128,37
ECC	160	53,46	53,46	54,45	107,91
	224	144,54	144,54	147,18	291,72

A comparação entre o ECC(160) com o RSA(1024) indica que a troca de chaves no servidor é 16 vezes mais eficiente no ECC. Além disso, a criptografia de chave pública pode ser utilizada para outras funções de segurança, como conexão segura de dispositivos com a internet e distribuição de assinaturas. Os autores Amin, Jahangir e Rasifard (2008) ficaram limitados em somente uma plataforma e não demonstraram como foi feita a simulação da

rede. Estudos com TelosB e MicaZ poderiam ser feitos para comprovar a real eficiência do ECC.

Segundo Ahmad, Beg e Abbas (2010) um dos grandes desafios na segurança das redes de sensores é o desenvolvimento de protocolos de segurança que não necessitem de auxílio externo, uma vez que as redes de sensores apresentam limitações como baixa quantidade de memória, poder de processamento, limitação de banda, falta de segurança física e facilidade de acesso por parte de invasores. Para Ahmad, Beg e Abbas (2010) a tendência é que a segurança em redes de sensores seja implantada por meio da criptografia de chave pública, porque é mais fácil de distribuir as chaves em relação à criptografia simétrica. Os motivos são a implantação aleatória dos sensores na rede e a dificuldade de quebra da autenticação por meio de invasores na criptografia de chave pública.

Em qualquer mecanismo de segurança de redes, a prevenção de ataques irá requerer serviços com autenticação, confidencialidade, integridade e não-repúdio. Os ataques a redes de sensores podem ser passivos, quando o invasor apenas captura dados que estão trafegando na rede, ou ativos, quando o invasor também altera ou insere novas informações na rede, como na negação de serviço. Além dos ataques comumente conhecidos, uma rede de sensores pode sofrer modificação no roteamento dos pacotes. Existe uma gama de variações para esse tipo de ataque, porém, basicamente o invasor altera a rota contida nos sensores para que as mensagens não consigam chegar ao destino.

Nos trabalhos relacionados Ahmad, Beg e Abbas (2010) citam o SPINS como protocolo de segurança que utiliza dois blocos de segurança: SNEP e uTesla. O modo Snep provê primitivas de segurança, como confidencialidade dos dados e autenticação. uTesla é um novo protocolo que provê *broadcast* autenticado em dispositivos com recursos limitados. Os autores afirmam que apesar da literatura informar que a criptografia de chave pública seja complexa, lenta, consumir maior quantidade de energia e não ser recomendada para dispositivos de recursos limitados como sensores, é possível utilizá-la em RSSF fazendo a escolha certa do algoritmo e dos parâmetros associados, realizando otimizações e utilizando técnicas de baixo consumo energético.

Os autores Ahmad, Beg e Abbas (2010) propõem a utilização do algoritmo ECC com base na Tabela 4.3, que compara o consumo de energia do algoritmo RSA e do ECC apli-

Tabela 4.3: Energia consumida em uJ dos algoritmos RSA e ECC na plataforma Mica2dot.

Algoritmo	Cliente	Servidor
RSA-1024	397,7	390,3
ECC-160	93,7	93,9

cados na plataforma Mica2dot. Na tabela se visualiza um melhor desempenho do algoritmo ECC em comparação com o algoritmo RSA. Em seu trabalho Ahmad, Beg e Abbas (2010) consideraram que cada sensor teria um único identificador e a rede seria homogênea, portanto formada por sensores do mesmo tipo e, além disso, a rede seria estática. Foi considerado também que o invasor não atacaria imediatamente após a implantação dos sensores. O esquema foi baseado no algoritmo ECC melhorado para redes de sensores criado por Blake, Seroussi e Smart (1999). O esquema proposto trabalha em três fases, antes da implantação, depois da implantação e na adição de um novo sensor. Antes da implantação a estação base gera os números que serão utilizados no protocolo. Depois da implantação cada nó sensor troca sua chave com seus vizinhos. Uma mensagem *HELLO* é trocada entre nós vizinhos, juntamente com o "id"e um contador. Qualquer mensagem que venha a ser trocada pelos sensores incrementa o contador, evitando *replay attack*. Quando um novo nó é adicionado para substituir um nó da rede, os mesmos valores do nodos substituídos são configurados no novo nó, e o processo de troca de mensagem com seus vizinhos citado é iniciado.

Segundo Ahmad, Beg e Abbas (2010) o sistema proposto é melhor que os convencionais *Diffie-Hellman* e RSA, porque no *Diffie-Hellman* a função para gerar a chave secreta envolve pelo menos duas fases que consomem grande quantidade de energia, assim como o RSA que contém uma função consumidora de energia que é acionada na encriptação e decriptação dos dados. No esquema proposto as funções que consomem muita energia não são realizadas pelos sensores, e sim pela estação base que contém fonte externa de energia. Os autores identificaram as fases em que era possível transferir o trabalho pesado dos esquemas de segurança dos sensores para a estação base. Seu esquema proposto mostrou que era possível utilizar algoritmos de chave pública em rede de sensores com baixo consumo de energia. Eles citaram duas limitações do seu trabalho: (1) O esquema só funciona em redes com plataformas que têm maior recurso energético, como o iMote, e (2) O consumo de energia aumenta com a quantidade de nós, portanto, a quantidade de nós é limitada.

O trabalho desenvolvido por Casola et al. (2011) teve o intuito de estudar problemas relacionados com a troca de dados entre nodos sensores e a avaliação de dois sistemas criptográficos utilizados para garantir requisitos de confidencialidade, autenticidade e integridade. Segundo Casola et al. (2011) o problema de gerenciamento de chaves e recursos necessários para autenticação e encriptação dos dados em RSSF não estão totalmente resolvidos, e há poucas soluções propostas na literatura.

Devido ao baixo custo computacional a criptografia simétrica é amplamente utilizada em protocolos de segurança para RSSF como MiniSec, TinySec e ZigBee. Porém, esta técnica requer um complexo método de gerenciamento e distribuição de chaves. Além disso, de acordo com Casola et al. (2011), a criptografia simétrica abrange de forma satisfatória apenas o requisito de confidencialidade, não satisfazendo requisitos de autenticação e integridade.

O trabalho desenvolvido por Casola et al. (2011) analisou o desempenho das bibliotecas WM-ECC e TinyPairing. A WM-ECC é uma biblioteca pública que implementa o ECC-169 bits para as plataformas MicaZ, TelosB e Tmote Sky. Algumas operações foram feitas na linguagem assembly, de forma a aumentar o desempenho. O TinyPairing é uma biblioteca criptográfica para RSSF modelada para reduzir os custos de memória RAM e ROM. Essa biblioteca utiliza emparelhamento de funções atreladas a operações de curvas elípticas.

A avaliação de desempenho foi realizada na plataforma TelosB acoplada ao microcontrolador MSP430 4.15MHZ, um rádio CC2420, 10kB de memória RAM e 48kB de memória Flash. Foram utilizados três sensores que se comunicavam com um mediador, porém, em um deles a arquitetura de segurança não estava implantada. O objetivo era obter a latência de cada protocolo de segurança, o tamanho do pacote, além do uso de memória RAM e ROM.

Os resultados mostraram que em relação à latência de inicialização, assinatura, verificação, encriptação e decriptação de dados, o WM-ECC foi mais rápido em todos os aspectos. O tamanho do pacote do WM-ECC foi de 80, enquanto que o TinyPairing utilizou pacotes de 60 bytes. Em relação à ocupação de memória RAM, o WM-ECC ocupou mais espaço que o TinyPairing, porém em relação à ocupação de memória ROM, o TinyPairing ocupou um espaço maior que o WM-ECC. Os resultados de ocupação de memória foram praticamente idênticos tanto nos nodos quanto no mediador.

4.3.2 ECC vs HECC

É notório que a segurança de dados têm papel fundamental nos sistemas de tecnologia da informação. A utilização de algoritmos criptográficos de chave pública em sistemas embarcados foi repudiada durante muito tempo, porém o trabalho de (WOLLINGER et al., 2004) surgiu como a primeira comparação entre o algoritmo de curva elíptica (ECC) e o algoritmo de curva hiperelíptica (HECC) em processadores embarcados, especificamente ARM7, ColdFire e PowerPC. Os autores analisaram a influência do tipo de processador, recursos e arquitetura no desempenho dos algoritmos.

Segundo Wollinger et al. (2004) o bom desempenho dos algoritmos criptográficos baseados em curvas em plataformas com restrições computacionais deve-se ao tamanho dos operandos utilizadas por esses algoritmos. Considerando o mesmo nível de segurança, enquanto o ECC trabalha com operandos de aproximadamente 160 bits, o RSA funciona com operandos de aproximadamente 1024 bits. As técnicas utilizadas para melhorar o desempenho dos algoritmos foram: Troca de inversões simultâneas das operações Montgomery, Reordenamento da etapa de normalização, Cálculo do resultante r de u1 e u2 para adição de grupo, bem como u1 e h+2v1 usando matrix de Berzout, e por fim escolha do HECC com algumas propriedades.

Os diferentes tipos de implementações e plataformas influenciam no desempenho dos algoritmos, e a performance dos sistemas baseados em curvas elípticas e hiperelípticas dependem não somente da especificação do algoritmo, mas também dos detalhes de implementação e do tipo de processador utilizado. Foram codificados diferentes variações dos algoritmos ECC e HECC. Especificamente para o ECC foi realizada uma implementação baseada no padrão IEEE P1363. Ainda segundo os autores, quanto mais específica uma implementação, maior a sua eficiência. Para o HECC foram realizadas implementações genus 2 e genus 3, com $h(x) \neq 1$. A influência da cache de dados e da cache de instruções foi analisada especificamente na plataforma PowerPC. O tempo da multiplicação escalar melhorou em um fator de 7.7 quando utilizou-se a cache de dados. Portanto, os autores indicam o uso de cache nas implementações do ECC e HECC.

Analisando os testes realizados na plataforma ARM, a multiplicação modular e a ra-

diciação chegaram a ser 4 e 2 vezes mais rápidas em relação a implementações padrão e otimizadas respectivamente. Em relação à multiplicação escalar, a implementação especial chegou a ser 50% mais rápida que a implementação padrão. Segundo Wollinger et al. (2004) contrariamente ao que se acreditava, o HECC genus 3 com h(x)=1 pode superar o ECC, além disso as curvas genus 3 são mais rápidas que curvas genus 2. Portanto, os algoritmos ECC e HECC são indicados para segurança de sistemas embarcados. O HECC genus 3 obteve melhores resultados em praticamente todas as plataformas quando comparado com o ECC.

Os autores Wollinger et al. (2004) finalizam afirmando que o HECC é adequado para plataformas com restrições computacionais, pois como foi apresentado, alcançou as mesmas taxas do ECC. Além disso, rotinas especiais podem reduzir em até 50% o tempo dos algoritmos e o uso de cache pode melhorar o desempenho de forma considerável. Apesar dos resultados satisfatórios do HECC com relação ao tempo de processamento, outros quesitos como consumo de memória e energia deveriam ter sido avaliados por Wollinger et al. (2004).

O trabalho realizado por Batina et al. (2007) teve o intuito de criar implementações dos algoritmos *Elliptic/Hyperelliptic Curve Cryptography (ECC/HECC)* da forma mais compacta possível, para que pudessem ser utilizados em sistemas embarcados. Os autores avaliaram implementações anteriores e fizeram testes com os algoritmos propostos.

Segundo Batina et al. (2007), as curvas elípticas podem ser consideradas como um caso especial das curvas hiperelípticas, onde as curvas elípticas são curvas hiperelípticas de genus g = 1. Neste trabalho os autores consideram as curvas hiperelípticas com genus g = 2, onde $GF(2^n)$. Para o ponto de multiplicação do ECC foi utilizado o algoritmo de Montgomery (1987) que mantém a relação entre $P_2 - P_1$ como invariantes. Nessa representação os cálculos são realizados apenas na coordenada x, esse fato permite economizar registradores, principal critério para obtenção de uma implementação compacta. A implementação do HECC utilizou curvas do tipo II, e como ponto de partida para operações de divisão foi utilizada a fórmula de Hodjat et al. (2006), pois permite maior velocidade na duplicação das curvas mantendo o mesmo nível de segurança.

A solução proposta por Batina et al. (2007) consistiu em uma unidade de controle (FSM), uma unidade aritmética modular (MALU), memória ROM e RAM. As tarefas de cada componente da arquitetura foram divididas de forma à aumentar a performance dos algoritmos

ECC e HECC. Os resultados finais para um ponto de multiplicação mostraram que o ECC obteve um tempo de 210 ms sobre $F_{2^{67*2}}$ e o HECC obteve um tempo de 546 ms com $F_{2^{67}}$, ambos com tamanho de campo (d = 8).

Os autores utilizaram algoritmos HECC e ECC sobre campos compostos e fizeram otimizações em relação ao número de registradores, resultando em uma solução de área otimizada. Os resultados mostraram que a unidade aritmética obteve melhores resultados em relação às implementações padronizadas do ECC, porém requisitou uma quantidade de memória relativamente maior. Os autores também afirmam que com os resultados obtidos é possível aplicar criptografia de chave pública baseada em curva em dispositivos como *Radio-Frequency IDentification* (RFID), porém novos estudos devem ser realizados com estimativas mais precisas de utilização de memória e de consumo de energia. Apesar dos resultados satisfatórios obtidos por Batina et al. (2007), o trabalho não considerou uma plataforma específica de sensores, além disso, analisa apenas tempo computacional.

O trabalho desenvolvido por Amin, Jahangir e Rasifard (2008) analisou algoritmos de chave pública em rede de sensores sem fio usando simulações e dados obtidos de trabalhos anteriores. A plataforma escolhida foi a Mica2dot e os algoritmos comparados foram o RSA e o *Elliptic Curve Cryptosystem* ECC. Eles afirmam que o ECC têm atraído muita atenção como solução de segurança para RSSF devido seu pequeno tamanho de chave e baixo *overhead* computacional.

Uma grande limitação das RSSF é quanto a restrição de recursos computacionais e energia, portanto, nem todos os algoritmos criptográficos são aplicáveis nesse tipo de rede. Os autores Chatterjee, De e Gupta (2011) informaram que a criptografia baseada em curvas é mais eficiente em sistemas embarcados, pois este tipo de técnica exige operadores menores que os exigidos pelo algoritmo RSA e oferecem o mesmo nível de segurança. Portanto, ECC e HECC são mais indicados em ambientes com restrição computacional, desde que técnicas de aprimoramento possam ser utilizadas no desenvolvimento desses algoritmos.

Os autores Chatterjee, De e Gupta (2011) estudaram as principais funções que compõem os algoritmos ECC e HECC, fazendo um comparativo de tempo das funções que exigem mais processamento. Os algoritmos ECC e HECC são baseados na estrutura algébrica das curvas elípticas (ECC) e também hiperelípticas (HECC) sobre campos finitos e utilizam chaves

menores que as utilizadas no algoritmo RSA. Atualmente a comunicação digital utiliza a criptografia de chave pública, pois esta permite comunicação segura por canais inseguros sem a troca prévia de chaves, e ainda permite a utilização de assinaturas digitais. Segundo Chatterjee, De e Gupta (2011) a maioria dos produtos e padrões que utilizam criptografia de chave pública para encriptação e assinaturas digitais preferem o RSA, porém o RSA não é adequado para ambientes com restrição de hardware, pois utiliza chaves grandes.

Para se ter uma idéia da aplicabilidade dos algoritmos ECC e HECC em dispositivos com restrições computacionais, Chatterjee, De e Gupta (2011) afirmaram que a eficiência do algoritmo criptográfico ECC com tamanhos de chave de aproximadamente 160 bits é a mesma obtida utilizando o algoritmo RSA com chave de 1024 bits.

A avaliação foi realizada com o algoritmo HECC genus 2 (campos binários) em grupos de 162, 166, 176 e 182. O ECC utilizando métodos binários foi utilizado segundo recomendação do NIST, com n = 163, 233 e 283. A máquina de teste foi um PC Core 2DUO CPU T6400@2.00GHz com 4GB de RAM e sistema operacional *windows vista* utilizando jdk1.6. Os resultados mostraram que a multiplicação escalar do HECC é mais veloz que a multiplicação escalar do ECC, (ver Tabela 4.4).

Tabela 4.4: Tempo das operações de multiplicação escalar em curvas Elípticas e Hiperelípticas (CHATTERJEE; DE; GUPTA, 2011)

Curva	Campo	Grupo	Multiplicação Escalar(ms)
Curvas Hiperelípticas (genus 2)	F_2^{81}	2^{162}	2,12
Curvas Elípticas (genus1)	F_2^{163}	2^{162}	4,24

Os autores Chatterjee, De e Gupta (2011) também realizaram a encriptação e decriptação de um arquivo de texto com 899 Bytes para comparar o tempo dos algoritmos ECC e HECC. Os resultados mostraram que os dois algoritmos gastam mais tempo na encriptação, com aumento de tempo de acordo com o crescimento dos campos. Já a decriptação do HECC é relativamente mais rápida que a decriptação do ECC com o mesmo nível de segurança. Eles concluíram que o HECC genus 2 mostrou-se ligeiramente mais rápido que o ECC (genus1). No cenário estudado isso acontece porque o HECC utiliza pequenos operandos, além de requerer menos tempo de processamento em operações básicas como encriptação e decriptação. Portanto, segundo Chatterjee, De e Gupta (2011) o HECC é mais indicado para dispositivos com restrição de hardware. Os resultados mostraram também que a encriptação

do HECC é ligeiramente mais lenta que o ECC. Além disso por se tratar de uma possível indicação desse algoritmo para dispositivos com recursos limitados, seria importante a análise do consumo energético destes algoritmos.

4.3.3 MQQ em RSSF

O trabalho de El-Hadedy, Gligoroski e Knapskog (2008) comparou o algoritmo MQQ 160 com o RSA 1024 na plataforma PC e FPGA. Os dados mostraram que a quantidade de núcleos de processamento influenciou nos resultados, uma vez que na plataforma PC com um processador, o algoritmo MQQ realizou a cifragem em 140.485 ciclos. Já na plataforma PC com dois processadores, o MQQ realizou a cifragem em 80.105 ciclos. O RSA 1024, em um único núcleo realizou a cifragem em 119.800 ciclos. Em relação à decifragem, o algoritmo MQQ se mostrou 275 vezes mais rápido que o RSA. Segundo os autores, na plataforma FPGA, o algoritmo MQQ se mostrou 10.000 vezes mais rápido que o RSA, apresentando velocidade comparável com a do algoritmo simétrico AES. El-Hadedy, Gligoroski e Knapskog (2008) não consideraram plataformas específicas de sensores, além de não envolver algoritmos baseados em curvas, que são apontados pela literatura como mais indicados para plataformas com restrições computacionais.

Em outro trabalho, Maia (2010) avaliou o desempenho dos algoritmos RSA e MQQ nas plataformas de sensores TelosB e MicaZ. Os códigos foram escritos em nesC e os tempos foram medidos usando componentes da própria linguagem e também com auxílio de um multímetro. Os resultados mostraram que na plataforma TelosB o MQQ foi aproximadamente 909 vezes mais rápido que o RSA na cifragem e 5.470 vezes mais veloz na decifragem (MAIA; BARRETO; OLIVEIRA, 2010). O autor ainda informa que a mensagem utilizada pelo MQQ foi de 160 bits contra uma mensagem de 8 bits usada pelo RSA, portanto caso fosse utilizada mensagens do mesmo tamanho, essa diferença aumentaria consideravelmente. Apesar de ter comparado os algoritmos RSA e MQQ na plataforma TelosB, Maia (2010) apenas estimou o tempo de processamento do algoritmo MQQ na plataforma Micaz. Como trabalhos futuros da dissertação de Maia (2010), ele sugere a comparação entre MQQ e ECC.

Os autores Branovic, Giorgi e Martinelli (2004) analisaram os algoritmos RSA e ECC

através do SimpleScalar. O chip ARM considerado foi o XScale, que contém 32KB na camada de instrução e dados da L1. Nesse trabalho, o algoritmo RSA 1024 apresentou uma quantidade de ciclos inferior ao obtido pelo ECC. A quantidade de misses por instruções do RSA 1024 também foi menor que a quantidade apresentada pelo ec-elg_p192.

4.4 Protocolos

No trabalho de Meulenaer et al. (2008) os autores analisaram o consumo de energia dos protocolos criptográficos Kerberos e *Elliptic Curve Diffie-Hellman* com autenticação provida pelo *Elliptic Curve Digital Signature Algorithm* (ECDH-ECDSA) aplicados nas plataformas de sensores MicaZ e TelosB. A análise foi feita utilizando modelos energéticos dos sensores baseados em medições.

As principais contribuições desse trabalho foram: (i) Uma metodologia para apontar o real custo da criptografia em sensores, possibilitando determinar a relação entre custo computacional e comunicação; (ii) Estimativa dos protocolos de gerenciamento de chaves para MicaZ e TelosB, permitindo comparar técnicas criptográficas para soluções simétricas e assimétricas.

Segundo Meulenaer et al. (2008), um grande número de trabalhos acadêmicos têm sido desenvolvidos tendo como foco o uso de algoritmos criptográficos em RSSF. Algoritmos simétricos como o AES foram discutidos em (HEALY et al., 2007; LAW; DOUMEN; HARTEL, 2006). Comparações de performance foram realizadas em (GURA et al., 2004; WANDER et al., 2005). No trabalho de Piotrowski, Langendoerfer e Peter (2006) os pesquisadores descobriram que a energia consumida na transmissão de dados era, em ordem de magnitude, menor que a energia consumida pelas operações de criptografia.

Os autores Hodjat e Verbauwhede (2002) compararam o Kerberos com o ECDH na plataforma de sensores WINS (32-bit, 133Mhz), e concluíram que o Kerberos consome de 1 a 2 vezes menos energia. Em (GROBSCH; SZEKELY; TILLICH, 2007) os autores também compararam o Kerberos, porém com uma nova versão *Diffie-Hellman*: a *Elliptic Curve Menezes-Qu-Vanstone* (ECMQV), e descobriram que o consumo de energia do ECMQV foi

de até o dobro do consumo do Kerberos. Para quantificar o consumo energético, os dois trabalhos citados usaram o custo de transmissão e recepção por bit, baseado em medições, porém excluíram elementos práticos como o estado de escuta do sensor. Os autores Meulenaer et al. (2008) acreditam que os trabalhos relacionados subestimaram o consumo dos algoritmos por não levar em conta todos os modos de operação. Portanto, o trabalho dos autores Meulenaer et al. (2008) apresentou o consumo energético dos protocolos de segurança Kerberos e ECDH-ECDSA levando em consideração os modos de operação *Transmit*, *Listen*, *Receive*, *Compute* e *Sleep*.

Segundo Meulenaer et al. (2008) o custo de uma função específica pode ser obtido através da média de energia consumida por ciclos e o total de ciclos gastos pela função. O modo *listening* consome energia equivalente ao modo de transmissão devido ao *transceiver* ficar ativo nos dois modos. O modo *listen* pode causar um consumo exagerado de energia caso os sensores necessitem ficar nesse modo por muito tempo. Alguns protocolos economizam energia fazendo com que os sensores fiquem em modo *listen* o minímo possível, como o *Low Power Listening* (LPL). O protocolo LPL foi considerado nas avaliações.

Meulenaer et al. (2008) encontraram o custo computacional de cada operação criptográfica utilizando os modelos de energia de sensores e o número de ciclos computacionais de implementações conhecidas. Por exemplo, para encriptação simétrica no Kerberos, Meulenaer et al. (2008) utilizaram os resultados da implementação de Healy et al. (2007). O resultado final do consumo energético pode ser visto na Tabela 4.5.

Tabela 4.5: Estimativa de custo energético em mJ do Kerberos e ECDH-ECDSA nas plataformas MicaZ e TelosB (MEULENAER et al., 2008)

	MicaZ	TelosB
Kerberos (mJ)	14,4	12,64
ECDH-ECDSA (mJ)	283	130,9

Os autores de (MEULENAER et al., 2008) informam que outros autores chegaram a resultados semelhantes, porém testando em outra plataforma como a WINS. Os pesquisadores Hodjat e Verbauwhede (2002) mostraram que é preciso 140 mJ para executar o ECDH no WINS, e com essa mesma quantidade é possível executar o ECDH-ECDSA no TelosB. Portanto, o TelosB consome menos energia que o WINS. Outro fator importante é que outros autores não consideravam o modo *listen* em suas análises ou ainda utilizaram versões me-

lhoradas dos protocolos que não estão disponíveis publicamente. O trabalho de Meulenaer et al. (2008) apresentou uma metodologia para obter o real custo energético da criptografia em rede de sensores. As estimativas foram comprovadas, uma vez que o Kerberos se mostrou de 10 a 20 vezes mais rápido que o ECDH-ECDSA nas plataformas MicaZ e TelosB. Essa diferença pode permanecer significativa, mesmo com a utilização do LPL. Uma análise completa sobre o consumo de energia dessas técnicas foi sugerida como trabalho futuro. Utilizando o modelo criado pelos autores é possível analisar outros fatores que influenciam no desempenho de transmissões de dados em RSSF, como consumo de memória e velocidade de processamento.

Os autores Le et al. (2010) propuseram um esquema seguro e escalável de autenticação chamado de *Mutual Authentication and Access Control* baseado em *Elliptic Curve Cryptography* (MAACE). Esse protocolo promete garantir autenticação mutua entre os nós sensores e as estações base que são acessadas por profissionais de saúde. Le et al. (2010) utilizaram o algoritmo de chave pública ECC em seu protocolo, pois ele provê escalabilidade e menor consumo de memória em relação a algoritmos de chave privada, além disso, é fácil de ser implementado em plataformas de sensores. A análise da eficiência do MAACE foi apresentada e comparada com outros protocolos existentes para comprovar a vantagem do esquema proposto.

Na última década aumentou consideravelmente o número de pesquisas que estudam politicas de segurança em RSSF, porém poucas são aplicáveis na área da saúde. Os autores Le et al. (2009), Du et al. (2006), Le et al. (2008) estudaram aspectos de segurança baseado na localização do sensor, este aspecto não se aplica em monitoramento de pacientes, pois os nós são móveis e mudam constantemente de posição, gerando *overhead* na rede e crescimento do consumo de energia. Outros autores como Wang, Sheng e Li (2006) e Le et al. (2009) propuseram esquemas baseados em ECC para prover segurança de forma mais eficiente que algoritmos de chave simétrica.

Segundo Le et al. (2010) a hierarquia da rede de sensores aplicada na saúde consiste em três camadas: Camada de rede de sensores (SN), Camada de coordenação de rede (CN), e *Back-end Network layer* (BN). A camada de rede de sensores diz respeito aos sensores implantados no corpo do paciente. Estes dispositivos podem transmitir dados usando-se as

tecnologias *ZigBee* (IEEE 802.15.4) ou *Bluetooth* (IEEE 802.15.1), e por geralmente terem uma comunicação de curto alcance que varia entre 10-100m, necessitam de conexão com outros dispositivos de maior potência encontrados na camada de coordenação. A camada de coordenação de rede é composta por dispositivos como PDA, celular e *laptop*. Esses dispositivos são mais potentes que os sensores em relação a alcance de transmissão, capacidade de armazenamento e autonomia energética. Eles têm a missão de transmitir os dados capturados pelos sensores para estações que compõem a camada BN. A camada *Back-end* é composta por estações locais e servidores que são acessadas por profissionais da saúde.

Em relação a autenticação mutua, Le et al. (2010) afirmaram que o ECC é mais escalável que algoritmos de chave simétrica e requer menor quantidade de memória para armazenar as chaves, apresenta baixo *delay* na comunicação e é fácil de ser implantado. Além disso, o ECC requer pouca capacidade de processamento e pequeno tamanho de chave comparado com as soluções disponíveis de criptografia de chave pública como o RSA. A distribuição de chaves é feita pelo (KDC) *Key Distribution Center*, que é responsável por gerar todo material de segurança, como as chaves, certificados, além de revogar privilégios de acesso dos usuários. Apesar de ser eficiente, no ambiente de saúde, o KDC introduz alguns obstáculos como aumento de *delay*, congestionamento na rede e aumento no consumo de energia.

O trabalho de Le et al. (2010) comparou a eficiência do MAACE em relação ao HBQ e ao ENABLE, considerando que os dispositivos da camada CN e BN são mais robustos. Essa avaliação teve como foco apenas os sensores, tomando como base implementações práticas utilizando a plataforma Mica2. Os três protocolos avaliados por Le et al. (2010) são baseados no algoritmo ECC.

Tabela 4.6: Comparação de Performance dos protocolos MAACE, ENABLE e HBQ [Le et al. 2010]

	MAACE	ENABLE	HBQ
Tempo (ms)	10,136	13,256	2.415,04
Consumo Energético (mJ)	0,240	0,381	58,82

É possível observar na Tabela 4.6 que o protocolo MAACE conseguiu o melhor resultado em relação a tempo computacional, frente aos protocolos ENABLE e HBQ. Para calcular o consumo de energia dos protocolos, os autores Le et al. (2010) utilizaram a fórmula: E = U*I*t, e consideraram os seguintes valores para a plataforma Mica2 para o processo ativo: I

= 8 mA, U = 3V. Além disso foi considerado o uso de 2 baterias AA novas.

O trabalho de Le et al. (2010) mostrou que o protocolo MAACE é 238 vezes mais rápido que o HBQ e 1.3 vezes mais rápido que o ENABLE. Além de consumir 0.41% e 75% da energia consumida pelo HBQ e ENABLE, respectivamente. Uma questão importante levantada pelos autores é que a multiplicação do ECC consumiu um tempo significante e consequentemente aumentou o gasto energético.

Nos últimos anos houve um crescimento no número de usuários que utilizam dispositivos com restrição computacional para acessar a internet. Aparelhos como PDA, telefone celular e *pages* estão cada vez mais substituindo os computadores pessoais no acesso a web, porém a maioria destes dispositivos apresentam baixa quantidade de memória, pouco poder de processamento, além de restrições energéticas. Segundo Ganesan (2010) a internet é uma rede aberta e insegura, portanto soluções de criptografia e protocolos de autenticação devem ser capazes de garantir confidencialidade, integridade e autenticação nas transmissões. Neste contexto, as técnicas de segurança desenvolvidas para dispositivos móveis devem levar em consideração todas as limitações dessas plataformas.

A criptografia de chave pública têm sido utilizada em aplicações de comércio eletrônico. Na maioria das soluções o algoritmo RSA é utilizado por apresentar eficiência nas transações eletrônicas. Porém, esse algoritmo é considerado robusto, e não é indicado para dispositivos com restrições computacionais. Ganesan (2010) sugeriu o desenvolvimento de um protocolo de autenticação usando o algoritmo de curvas hiperelípticas (HECC) que fosse apropriado para dispositivos móveis. O algoritmo HECC utiliza chaves menores que o RSA, portanto espera-se que esse algoritmo tenha um melhor desempenho em plataformas restritas.

O protocolo proposto é baseado em curvas hiperelípticas sobre campos finitos *genus* 2, com chaves de 80 bits. O algoritmo foi desenvolvido em *J2ME wireless tool kit 2.5.1* pois a linguagem Java é multiplataforma. As comparações foram feitas com o algoritmo RSA utilizando chaves de 512 bits. Os resultados mostraram que a geração de chaves do RSA demorou 165.000 ms, enquanto o HECC levou 120.000 ms. A geração de sinal no RSA demorou 5.000 ms, e no HECC 3.600 ms, sendo portanto o HECC 27,3% mais veloz que o RSA na geração de chaves e 28% mais rápido na geração de sinal. Por fim, a verificação do sinal no RSA demorou 640 ms, contra 520ms do HECC, sendo portanto o HECC 18,7%

mais veloz que o RSA na verificação de sinal.

É perceptível que mesmo com chaves menores, o HECC alcançou melhores resultados que o RSA, sendo indicado pelo autor como um algoritmo apropriado para dispositivos com restrições computacionais. Porém, a plataforma de teste utilizada para o RSA foi a PALM III, diferente da plataforma de teste do HECC, que foi a *wireless tolkit*. Portanto, sugere-se o desenvolvimento de uma pesquisa mais completa, utilizando a mesma plataforma de teste para ambos os algoritmos.

4.5 Considerações Finais

Ainda hoje, grande parte dos pesquisadores não consideram viável a utilização de criptografia de chave pública em dispositivos com restrições computacionais, pois afirmam que este tipo de algoritmo têm um alto custo de implementação e consome grande quantidade de energia. Porém, existe uma linha de pesquisa que estuda implementações compactas de algoritmos de chave pública para aplicações de baixo custo como RFID e sensores. Os autores Aranha et al. (2008) desenvolveram técnicas de otimização para algoritmos de aritmética em corpos binários, incluindo cálculo de quadrado, multiplicação e redução modular. Os algoritmos resultantes minimizam a quantidade de acessos à memória e se mostraram como os algoritmos mais eficientes para a plataforma de sensor Micaz já publicados. Ainda de acordo com Aranha et al. (2008), a eficiência destes algoritmos culmina em uma multiplicação de ponto com desempenho superior em até 39% à melhor implementação conhecida para corpos binários para o mesmo nível de segurança. Os resultados também mostram que na plataforma Micaz, curvas definidas sobre corpos binários podem ter desempenho idêntico ou superior a curvas definidas sobre corpos primos. Além disso, muitas pesquisas envolvendo avaliação de desempenho de algoritmos criptográficos vêm sendo realizadas a fim de apontar aquele que seja mais eficiente em RSSF.

É possível observar na Tabela 4.7 que nenhum trabalho apresentou um estudo unificado de desempenho dos algoritmos RSA, ECC e MQQ em plataformas de RSSF. Além disso, a maioria dos trabalhos avalia apenas uma variável de desempenho, como consumo energético ou tempo computacional. Neste sentido, essa dissertação de mestrado objetiva avaliar o

Tabela 4.7: Restrições das principais avaliações de desempenho

Trabalho	Algoritmos	Aspectos Analisados	Plataforma	Limitações
(GURA et al., 2004)	(ECC - 160) (RSA - 1024)	Tamanho de código e Consumo de memória	(8051 - Chipcom) (AVR Atmega128)	Não avaliou o tempo de processamento
(WOLLINGER et al., 2004)	(ECC) e (HECC)	Tempo computacional	ARM7, PowerPC e ColdFire	Considerou apenas um aspecto de comparação (tempo). Avaliou apenas a função de multiplicação escalar
(BRANOVIC; GIORGI; MARTINELLI, 2004)	(ECC) (RSA)	Tempo e Consumo de Memória	ARM	Não avaliou MQQ
(BATINA et al., 2007)	(ECC) (HECC)	Tempo Computacional	Plataforma Específica	Não considera plataforma específica de sensores. Analisa apenas tempo computacional.
(AMIN; JAHANGIR; RASIFARD, 2008)	(ECC) (RSA)	Consumo energético	Mica2dot	Apenas uma plataforma foi considerada. As funções consideradas foram troca de chaves e assinatura digital.
(EL-HADEDY; GLIGOROSKI; KNAPSKOG, 2008)	(MQQ) (RSA) (AES)	Tempo de processamento em ciclos, frequência e throughput	Desktop e FPGA	Não envolveu algoritmos baseados em curvas elípticas
(AHMAD; BEG; ABBAS, 2010)	(ECC - 160) (RSA - 1024)	Consumo de Energia	Mica2dot	Não realizou testes reais ou simulação
(MAIA, 2010)	(MQQ) (RSA)	Tempo de processamento e utilização de memória	TelosB e MicaZ	Não envolveu algoritmos baseados em curvas elípticas
(CHATTERJEE; DE; GUPTA, 2011)	(ECC) (HECC)	Tempo de Encriptação, Decriptação, Mult. Escalar	jdk 1.6	Não avaliou o consumo de energia

desempenho dos principais algoritmos assimétricos para RSSF de forma a sanar algumas brechas deixadas pelos trabalhos relacionados que foram apresentados nesse capítulo. O próximo capítulo apresenta os testes dos algoritmos criptográficos RSA, ECC e MQQ nas plataformas desktop, ARM, MSP430 e AVR, bem como a análise dos resultados.

Capítulo 5

Testes e Análise comparativa

Como já foi exposto anteriormente, os componentes de uma RSSF não são dotados de grande capacidade computacional, tampouco de capacidade energética, portanto é constante a busca por soluções de segurança que sejam eficientes, rápidas e que consumam o mínimo de recursos. Neste contexto, este capítulo apresenta as avaliações de desempenho em plataformas embarcadas usadas em RSSF, que foram realizadas com os algoritmos RSA, ECC e MQQ. A seção 5.1 mostra quais plataformas utilizadas nos experimentos desta dissertação. A seção 5.2 define quais ferramentas foram utilizadas nas simulações e avaliação de desempenho. A seção 5.3 descreve quais os parâmetros e cargas utilizados nas simulações. A seção 5.4 descreve os resultados obtidos na plataforma Desktop. A seção 5.5 mostra os resultados obtidos na plataforma ARM. A seção 5.6 mostra os resultados obtidos na plataforma MSP430. A seção 5.7 mostra os resultados obtidos na plataforma AVR Atmega128. Por fim, a seção 5.8 faz um comparativo entre os resultados obtidos nesta avaliação com os trabalhos presentes na literatura.

5.1 Plataformas e Bibliotecas Criptográficas

As plataformas utilizadas nesta avaliação foram Desktop, ARM, MSP430 e AVR. A plataforma Desktop consiste em um computador do tipo Notebook, processador Intel core2duo 1,83Ghz, memória RAM de 2GB.

Segundo Austin (2012) e de acordo com a Tabela 5.1 e Figura 5.1, a plataforma sa1core utilizada no simulador ARM é compatível com a plataforma StrongArm SA-110 que utiliza

processador RISC de 32 bits 200MHz, 16k na cache L1 de dados e 16k na cache L1 de instruções.

Figura 5.1: Plataforma StrongArm (WIKIPEDIA, 2013).

Tabela 5.1: Arquitetura Simulada - ARM

Fetch queue (instructions)	8
Branch prediction	nottaken
Fetch & decode width	1
Issue width	2
ITLB	32-entry, fully associative
DTLB	32-entry, fully associative
Functional units	1 int ALU / 1 int MUL/DIV
Instruction L1 cache	16KB, 32 way
Data L1 cache	16KB, 32 way
L1 cache hit latency	1 cycle
L1 cache block size	16B
L2 cache	none
Memory latency (cycles)	64 , 1
Memory bus width (bytes)	4

A família MSP430 é composta por microcontroladores de 16 bits da Texas Instruments. A arquitetura da CPU é RISC e suporta instruções de 8 e 16 bits, permitindo migração para a maioria das plataformas de tamanho similar (NAGY, 2003). Os microcontroladores da família MSP430 possuem várias características em comum como serem de 16 bits e possuírem o mesmo conjunto de 27 instruções e 12 registradores de propósito geral. A frequência de clock e tamanhos de ROM e RAM variam para cada membro - o MSP430F1611, utilizado pelos sensores Tmote Sky e TelosB, possui clock de 8MHz, 48KB de ROM e 10KB de RAM (GOUVEA, 2010b). A imagem do sensor TelosB que utiliza o processador MSP430 pode ser visualizada na Figura 5.2.

A plataforma Atmega 128 é baseada na arquitetura AVR. Esse microcontrolador é de 8 bits, contém 128KB de memória flash e 4KB de memória de dados (YANG; HUA, 2012). Nesta avaliação, foi considerado um processador de 16Mhz. Essa plataforma é amplamente

Figura 5.2: Sensor Telosb com processador MSP430 (BERKELEY.EDU, 2013).

utilizada em RSSF, incluindo dispositivos como Mica2, MicaZ e IRIS. A imagem do sensor MicaZ que utiliza o processador AVR Atmega128 pode ser visualizada na Figura 5.3.

Figura 5.3: Sensor MicaZ com processador AVR Atmega128 (LORINCZ, 2006).

Na avaliação de desempenho na plataforma ARM, os algoritmos criptográficos RSA e ECC foram escritos com base na biblioteca Miracl - *Multiprecision Integer and Rational Arithmetic C/C++ Library*. Segundo Scott (2003) a biblioteca Miracl é referência entre ferramentas criptográficas por apresentar facilidades de implementação de algoritmos em aplicações de segurança no mundo real. Além disso, a Miracl disponibiliza códigos compactos, rápidos e eficientes, apresentando alto desempenho em qualquer processador ou plataforma. Estudos realizados por Uto e Reis (2005), Pigatto, Silva e Branco (2011) e Pigatto (2012) mostraram que códigos baseados na Miracl obtem desempenho superior a códigos baseados em outras bibliotecas criptográficas, como Crypto++, LibTomCrypt, OpenSSL e LiDIA+GMP. Em relação ao algoritmo MQQ, o código baseia-se na implementação escrita por Maia (2010) e Maia, Barreto e Oliveira (2010).

Na plataforma MSP430, os códigos dos algoritmos RSA e ECC foram baseados na biblioteca RELIC (ARANHA; GOUVEA, 2012), pois a biblioteca MIRACL não oferece suporte a essa plataforma. O algoritmo MQQ não sofreu alterações, uma vez que não depende de bibliotecas externas da linguagem ANSI C.

5.2 Ferramentas

A avaliação do tempo de processamento dos algoritmos criptográficos na plataforma Desktop foi realizada através da própria linguagem de programação C, pelo compilador *Gnu Compiler Collection* (gcc) com uso da biblioteca "time.h".

A avaliação de desempenho na plataforma ARM foi realizada usando-se o simulador SimpleScalar (AUSTIN; LARSON; ERNST, 2002), que retornou informações sobre uso de memória, processador e tempo de processamento. A avaliação do consumo de energia foi realizada usando o simulador Sim-Panalyzer (AUSTIN; LARSON; ERNST, 2002), que retornou informações do consumo energético dos componentes arquiteturais da plataforma ARM. Segundo Pourpeighambar e Sabaei (2012), o Sim-Panalyzer é um simulador de consumo de energia baseado no simulador arquitetural SimpleScalar. O SimpleScalar é um simulador de arquitetura computacional que modela um computador virtual com CPU, memórias caches (de dados e instruções), memória RAM (*Random Access Memory*) e toda a hierarquia das memórias, podendo simular programas reais executados sobre tais plataformas.

O Sim-Panalyzer consegue modelar de forma detalhada tanto a potência dinâmica quanto a potência de fuga (SHARMA et al., 2012). Para gerar os resultados, o simulador é dividido em componentes que simulam partes distintas do computador e ao final da simulação mostra o consumo de energia de cada um desses componentes, tais como: cache, caminho de dados, unidade de execução, circuitos de *clock* e unidades de entrada e saída (I/O). Os parâmetros para a geração destes componentes são configurados como entrada para o simulador Sim-Panalyzer que juntamente com o simulador SimpleScalar geram os padrões do consumo de energia da arquitetura modelada.

Na plataforma MSP430, a avaliação de desempenho foi realizada usando o simulador

MSPsim (ERIKSSON et al., 2007) que mede o número de ciclos gastos por cada função, tornando fácil o cálculo do tempo de uma simulação. O MSPsim é escrito na linguagem JAVA e simula processadores da família MSP430, além de algumas plataformas de sensores. É possível observar na Figura 5.4 que o MSPsim pode mostrar representações gráficas dos sensores que estão sendo simulados, além disso, informações sobre funções que estão sendo executadas são apresentadas no monitor do programa. A versão do simulador utilizada nos testes foi modificada por Conrado P. L. Gouvêa e disponibilizada em seu site: http://conradoplg.cryptoland.net/software/custom-mspsim/. Essa versão customizada foi utilizada por oferecer facilidades na execução de algoritmos baseados na biblioteca RELIC.

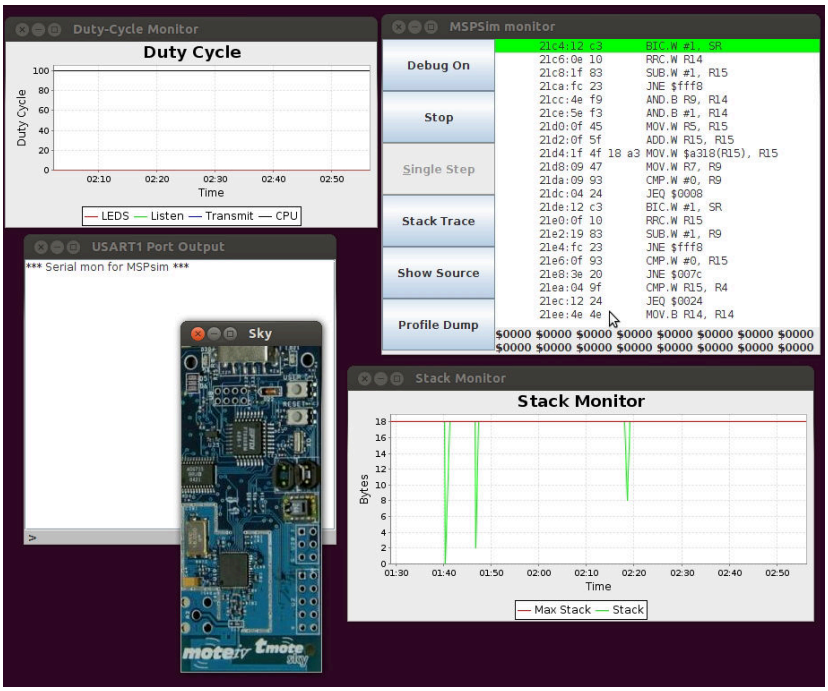

Figura 5.4: Simulador MSPsim

Na plataforma AVR Atmega128, a avaliação de desempenho foi realizada usando o simulador Avrora (TITZER; LEE; PALSBERG, 2005). O simulador Avrora retorna entre outras informações, o número de ciclos de CPU e o throughput dos programas simulados (OTHMAN; TRAD; YOUSSEF, 2012).

5.3 Parâmetros e Carga

Os algoritmos RSA, ECC e MQQ foram avaliados segundo tempo de processamento, uso de memória, processador e consumo de energia. Os algoritmos foram escritos na mesma linguagem de programação e submetidos a cargas de mesmo tamanho. A mensagem encriptada em todos os testes foi de 44 bytes. Enquanto Maia (2010) usou uma mensagem de 20 bytes, Chatterjee, De e Gupta (2011) usaram mensagens de 899 bytes. Os autores Refatti e Pazeto (2011), avaliaram RSSF para o corpo humano e a partir da literatura concluíram que neste tipo de aplicação, 32 bits são necessários para transmitir dados. Os autores Oliveira e Albuquerque (2007) avaliaram aplicações multimídia em RSSF e utilizaram pacotes de 200 bytes nas simulações. Os autores Wander et al. (2005), consideraram um pacote de 41 bytes para aplicações de RSSF que necessitam de segurança, sendo 32 bytes referentes ao *payload* e 9 bytes referentes ao cabeçalho. A partir das considerações acima, é perceptível que não existe uma padronização do tamanho da mensagem utilizada em aplicações de RSSF que requerem segurança. Portanto, optou-se por utilizar uma mensagem que fosse maior que a utilizada por Maia (2010), uma vez que apenas esse autor realizou avaliações com o MQQ em plataformas embarcadas usadas em RSSF e que tivesse o tamanho aproximado da mensagem utilizada por Wander et al. (2005).

A equivalência entre tamanhos de chaves dos algoritmos pode ser visualizada na Tabela 5.2 (BRANOVIC; GIORGI; MARTINELLI, 2004). Essa Tabela traz a seguinte informação: O nível de segurança obtido com o algoritmo RSA 1024 bits é o mesmo obtido com os algoritmos ECC 192 (prime field - corpos primos), ECC 163 (binary field - corpos binários) e MQQ 160 bits.

Tabela 5.2: Equivalência de tamanho de chave para RSA, ECC e MQQ.

RSA	1024	2048	3072	7680	15360
ECC (Prime field)	192	224	256	384	521
ECC (Binary field)	163	233	283	408	571
MQQ 160	160	—	—	—	—

A avaliação de desempenho desenvolvida nesta dissertação não levou em consideração o algoritmo RSA 3072, pois os houve estouro de *buffer* na tentativa de realizar simulações com chaves maiores que 2048.

Na plataforma MSP430 o tamanho das chaves dos algoritmos criptográficos é limitado,

devido a plataforma ser de apenas 16 bits. Portanto, foram realizadas avaliações apenas com o algoritmo RSA 354 bits, ECC 158 bits e MQQ 160 bits.

Na plataforma AVR, apenas o algoritmo MQQ foi simulado, devido as restrições da plataforma, como processador de apenas 8 bits. Os resultado obtidos com o MQQ 160 foram comparados com a avaliação do ECC realizada por Yan e Shi (2006).

Alguns gráficos estão divididos em (A) e (B). Essa divisão foi necessária, pois alguns dados do MQQ não eram visíveis quando comparados com o RSA. Além disso, a divisão permite uma melhor comparação direta entre resultados do algoritmo ECC e MQQ.

Os simuladores SimpleScalar, Sim-Panalyzer, MSPsim e Avrora são considerados arquiteturais, portanto, independente da máquina em que as simulações forem executadas, os resultados serão os mesmos, logo, para esses simuladores não houve cálculo do intervalo de confiança da média das amostras.

5.4 Resultados Obtidos na Plataforma Desktop

Os dados obtidos na plataforma desktop são importantes para mostrar o correto funcionamento dos algoritmos alvo de estudo. Para realizar a aferição do tempo, variáveis do tipo clock_t da biblioteca time.h foram adicionadas às funções de cifragem e decifragem dos algoritmos. Os testes na plataforma Desktop foram repetidos 30 vezes, de forma que a avaliação pudesse seguir estudos de confiabilidade usando a distribuição normal. As médias das amostras apresentadas na Tabela 5.3 e Figuras 5.5 e 5.6 estão sob um intervalo de confiança de 95%.

Tabela 5.3: Intervalo de confiança de 95%

Algoritmo	Média (cifragem)	IC (cifragem)	Média (decifragem)	IC (decifragem)
RSA 1024	21,4	(21,37 - 21,43)	7,1	(7,06 - 7,14)
RSA 2048	149	(148,49 - 149,51)	42,8	(42,66 - 42,94)
ec-elg_p192	7,6	(7,50 - 7,70)	3,9	(3,83 - 3,97)
ec-elg_p224	9,0	(8,58 - 9,42)	4,7	(4,59 - 4,81)
ec-elg_p256	10,4	(10,30 - 10,50)	5,3	(5,21 - 5,39)
ec-elg_p384	16,8	(16,55 - 17,05)	8,6	(8,47 - 8,73)
ec-elg_p521	25,3	(24,75 - 25,85)	12,9	(12,47 - 13,33)
MQQ 160	0,3	(0,296 - 0,304)	0,08	(0,077 - 0,083)

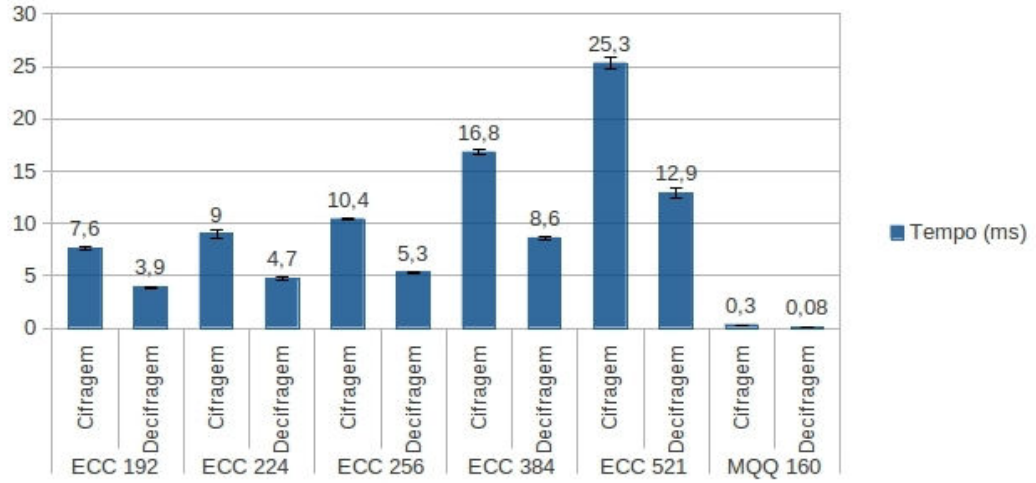

Figura 5.5: Tempo de processamento em plataforma Desktop - Algoritmos ECC e MQQ

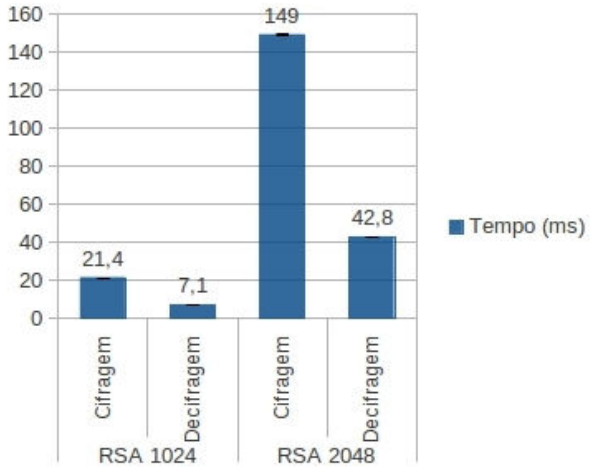

Figura 5.6: Tempo de processamento em plataforma Desktop - Algoritmo RSA

É possível observar na Tabela 5.3, que o tempo de cifragem e decifragem dos algoritmos baseados em curvas atinge desempenho superior ao RSA. Por outro lado, na Figura 5.5 se observa que o tempo de processamento do MQQ 160 na plataforma Desktop é menor que qualquer variação de chave do ECC. Neste comparativo, o MQQ 160 se mostrou 25 vezes mais rápido que o ec-elg_p192 na cifragem e aproximadamente 49 vezes mais rápido na decifragem. Em relação ao RSA 1024, o MQQ 160 foi 71 vezes mais rápido na cifragem e 88 vezes mais rápido na decifragem. Esses dados estão em correspondência com o que foi mostrado em outros trabalhos como em (EL-HADEDY; GLIGOROSKI; KNAPSKOG, 2008), (MAIA; BARRETO; OLIVEIRA, 2010) e (AHMAD; BEG; ABBAS, 2010), onde se destaca que o MQQ parece ser promissor, uma vez que é mais rápido que o ECC, e o ECC

mais rápido que o RSA.

Tabela 5.4: Porcentagem de tempo por função - ARM

	RSA 1024	**RSA 2048**	**ec-elg_p192**	**ec-elg_p224**	**ec-elg_p256**	**ec-elg_p384**	**ec-elg_p521**	**MQQ 160**
Cifragem	74,8%	77,7%	66,1%	65,7%	66,2%	66,1%	66,2%	78,9%
Decifragem	25,2%	22,3%	33,9%	34,3%	33,8%	33,9%	33,8%	21,1%

A Tabela 5.4 mostra que em todos os algoritmos estudados, a cifragem consome mais tempo que a decifragem. A cifragem do algoritmo RSA 1024 consome 74,8% do tempo, enquanto que a decifragem consome 25,2%. Já a cifragem do algoritmo ec-elg_p192 consome 66,1% do tempo, contra 33,9% da decifragem. Por outro lado, a cifragem do algoritmo MQQ 160 consome 78,9% do tempo, contra 21,1% da decrifragem. A geração de chave não foi avaliada nesse estudo, pois as funções de cifragem e decifragem são consideradas mais importantes em sistemas embarcados.

5.5 Resultados Obtidos na Plataforma ARM

Os mesmos códigos utilizados na plataforma Desktop foram transformados de modo a compilar e executar em plataforma ARM usando-se o compilador arm-linux-gcc[1]. A partir da compilação, o código ARM pôde ser executado no SimpleScalar e Sim-Panalyzer.

5.5.1 Tempo de Execução

Na plataforma ARM, à exceção do tempo de execução, os algoritmos foram analisados como um todo, ou seja, sem separação entre encriptação e decriptação. Considerando a frequência dos processador StrongArm sendo 200MHz e tendo a quantidade de ciclos de cada algoritmo e suas variações de chave dado pela Tabela 5.5, foi possível calcular o tempo de execução dos algoritmos em milissegundos (ms).

A Figura 5.7 mostra que na plataforma ARM, de forma similar ao resultado obtido em desktop, o tempo de processamento do algoritmo RSA foi maior que o ECC e MQQ em todas as variações de chave, além disso, observa-se também que o MQQ é mais rápido que

[1]arm-linux-gcc: Compilador gcc para plataforma ARM. Maiores informações em: http://www.gnuarm.com/

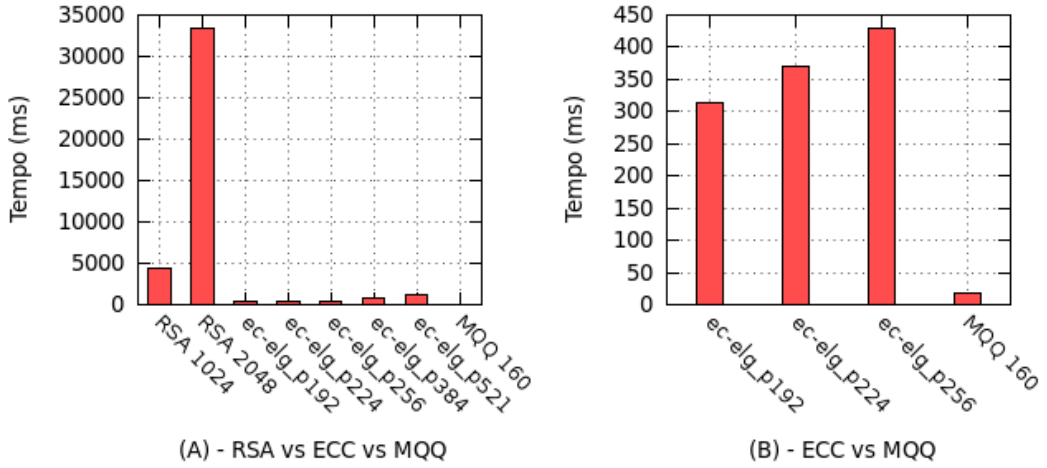

Figura 5.7: Tempo de processamento na plataforma ARM

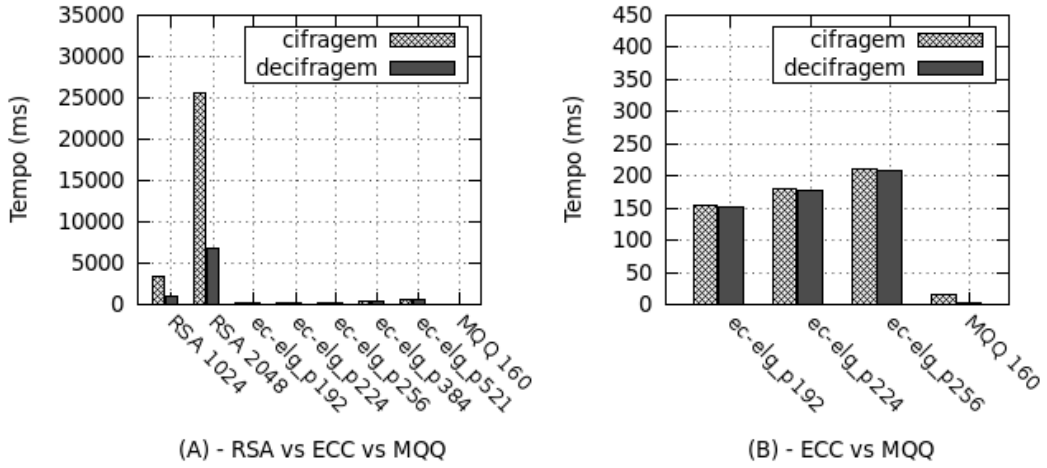

Figura 5.8: Tempo de processamento na plataforma ARM

o ECC. O MQQ 160 obteve o melhor desempenho dentre os algoritmos analisados, uma vez que apresentou um tempo de 19,2ms, contra 314,9ms do ec-elg_p192 e 4431,8ms do RSA 1024. Esses dados mostram que o MQQ na plataforma ARM foi 16 vezes mais rápido que o ec-elg_p192 e 230 vezes mais rápido que o RSA 1024. A Figura 5.8 mostra que nos algoritmos RSA e MQQ, a diferença entre a cifragem e decifragem segue os dados obtidos na plataforma Desktop, porém o algoritmo ECC apresentou o tempo de decifragem bem próximo ao tempo de cifragem. Neste caso, na plataforma ARM, os tempos de cifragem e decifragem dos algoritmos baseados em curvas elípticas são praticamente idênticos.

Tabela 5.5: Tempo de algoritmos em ciclos - ARM

Algoritmo	RSA 1024	RSA 2048	ec-elg_p192	ec-elg_p224	ec-elg_p256	ec-elg_p384	ec-elg_p521	MQQ 160
Ciclos	886.377.829	6.672.424.548	62.991.420	73.850.282	86.039.537	141.018.957	236.882.209	3.856.778

5.5.2 Uso do Processador

A avaliação de desempenho do processador na plataforma ARM levou em consideração critérios como quantidade de leituras e escritas, bem como a quantidade de *branches*. *Branches* são sequências de códigos que são condicionalmente executados de acordo com um controle de fluxo. Quanto mais *branches* em um código, menor o seu desempenho, pois testes condicionais exigem tempo de processamento e não tiram proveito do pipeline do processador, além de exigirem técnicas diferentes de escalonamento.

Um dos critérios utilizados para avaliação do uso de processador é a quantidade de ciclos por instrução (CPI). Esse dado refere-se a média de ciclos do processador que são gastos para executar uma instrução, portanto, quanto maior o número de CPI, mais complexas são as instruções do algoritmo e isso acarretará em maior uso do processador. Em relação à quantidade de ciclos por instrução (CPI), a Figura 5.9 mostra que o algoritmo baseado em curvas apresentou as maiores médias, seguido pelo algoritmo RSA e por fim, o algoritmo MQQ. As médias apresentadas nesse quesito confirmam os dados apresentados no final do capítulo 4 sobre a complexidade dos algoritmos, onde o algoritmo ECC apresenta alta complexidade, porém, por usar pequenas chaves apresenta melhor desempenho de tempo de processamento que o algoritmo RSA.

A Tabela 5.6 mostra o número de instruções executadas pelo processador nas diferentes variações de chave dos algoritmos. O algoritmo RSA 2048 apresentou um número de instruções aproximadamente 7 vezes maior que o RSA 1024, mostrando mais uma vez que o algoritmo RSA não é escalável, pois sua complexidade é exponencial. Em relação ao ec-elg_p192, o algoritmo RSA 1024 executou aproximadamente um número de instruções 20 vezes maior. Um comparativo entre ECC e MQQ, mostra que o MQQ (160) executou um número de instruções aproximadamente 13 vezes menor que o ec-elg_p192.

Tabela 5.6: Número de Instruções - ARM

Algoritmo	RSA 1024	RSA 2048	ec-elg_p192	ec-elg_p224	ec-elg_p256	ec-elg_p384	ec-elg_p521	MQQ 160
Instruções (bilhões)	1,110	8,426	0,055	0,067	0,080	0,140	0,248	0,004

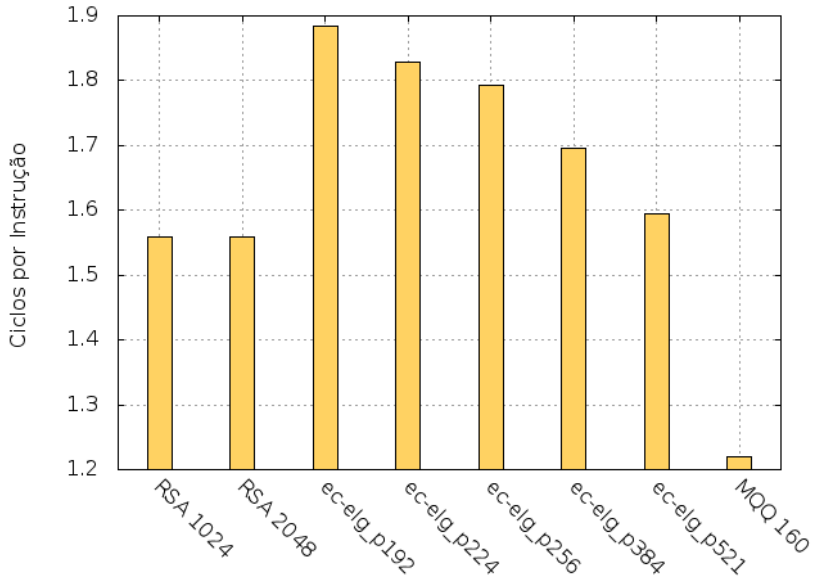

Figura 5.9: Ciclos por instrução (CPI) - ARM

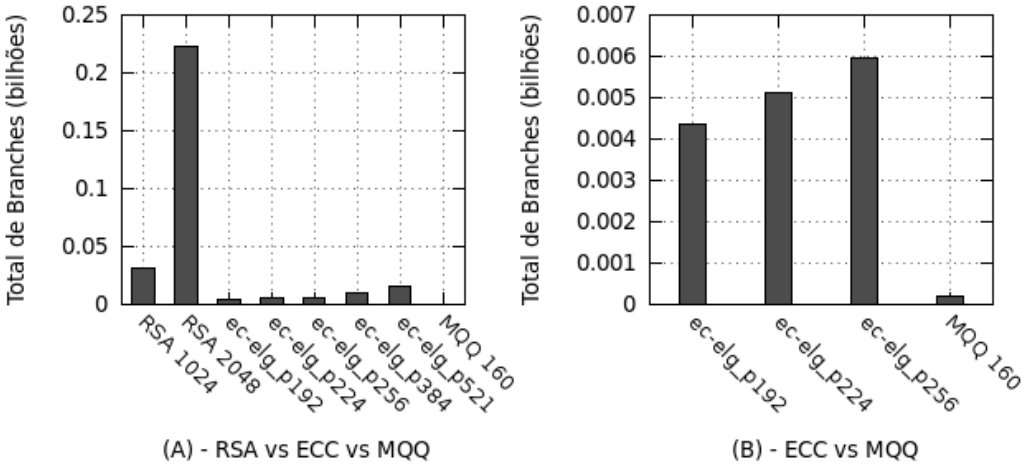

Figura 5.10: Quantidade de branches - ARM

A Figura 5.10 mostra que o algoritmo RSA 2048 apresentou 0,22 bilhões de branches, contra 0,03 bilhões do RSA 1024. Os algoritmos baseados em curvas, bem como o MQQ apresentaram números de branches menores. O ec-elg_p192 apresentou 4,3 milhões frente a 0,2 milhões do MQQ (160). Os números mostram que o MQQ (160) apresentou um número de branches 21 vezes menor que o apresentado pelo ec-elg_p192 e 150 vezes menor que o RSA 1024.

A verificação da quantidade de leituras e escritas realizadas por cada algoritmo na plata-

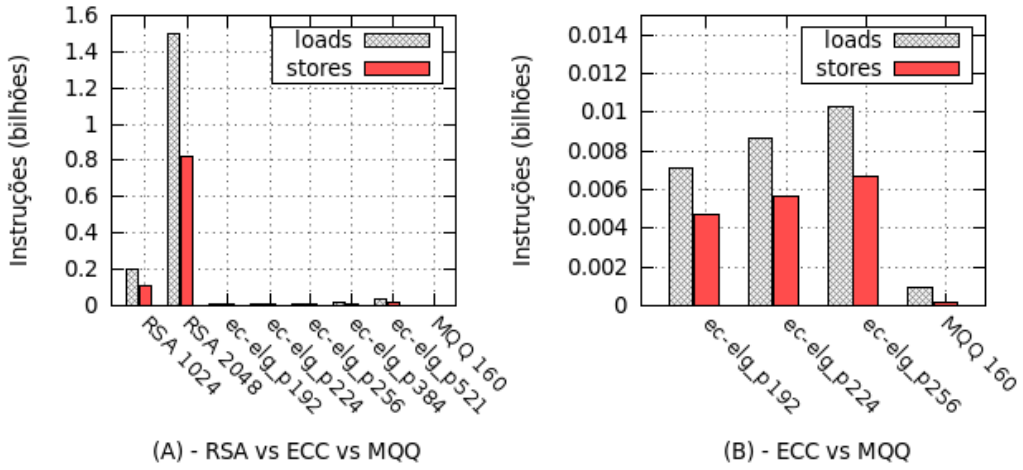

Figura 5.11: Quantidade de leituras e escritas - ARM

forma ARM não modificou o cenário da avaliação de desempenho, uma vez que de acordo com a Figura 5.11, o algoritmo RSA realizou mais operações desse tipo independente da variação de chave dos algoritmos estudados.

Dessa maneira, explica-se que o MQQ é promissor, pois apresenta melhores características de execução do que o ECC. Ainda assim, o ECC continua sendo padrão para sistemas embarcados, dado seu nível de segurança. O RSA não se mostrou viável para sistemas embarcados, pois além de não ser escalável, apresenta alto consumo de processador. Esses dados são confirmados nessa dissertação pelo menor uso de recursos arquiteturais, tais como loads, stores, branches e número de instruções executadas.

5.5.3 Consumo de Memória

A avaliação de consumo de memória na plataforma ARM teve o intuito de estudar não somente a quantidade de memória ocupada por cada algoritmo, como também questões específicas de cache que podem determinar qual algoritmo é mais indicado para RSSF. Entre esses quesitos mostramos o número de *replacements*. *Replacements* ou substituições, funcionam da seguinte forma: quando ocorre um *cache miss*, uma nova linha deve ser trazida da memória principal para a cache. A estratégia de substituição define qual linha será tirada da cache para dar lugar a nova. Outra quesito importante são os *write-backs*. *Write-backs* é

a transferência de um dado que se encontrava na memória principal para a memória cache. Tanto os *replacements* quanto *write-backs* são prejudiciais ao desempenho do sistema, pois implicam em maior uso de memória e processador.

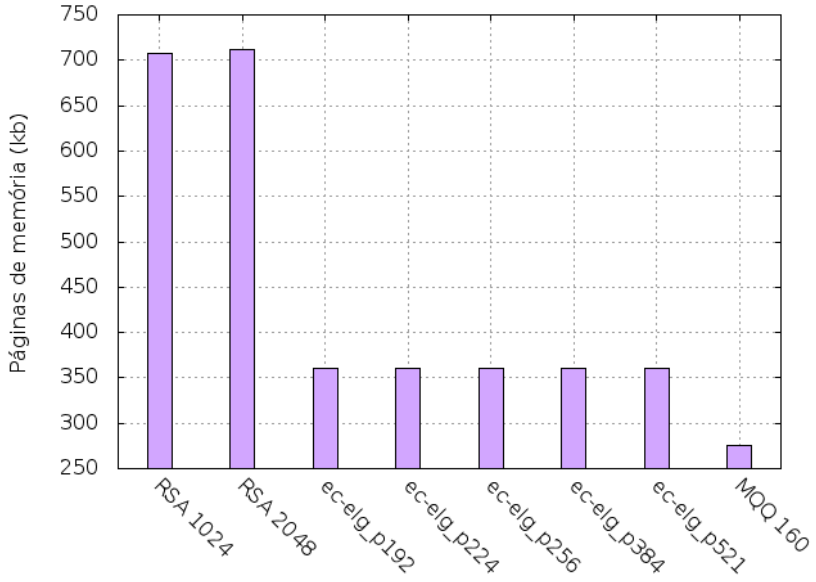

Figura 5.12: Tamanho total de páginas de memória alocadas - ARM

Em relação ao tamanho total de páginas de memória alocadas na plataforma ARM, a Figura 5.12 ilustra que independente da variação de tamanho de chave, os algoritmos baseados em fatoração de números primos e algoritmos baseados em curvas elípticas obtiveram resultados semelhantes em suas respectivas categorias. O algoritmo RSA 1024 alocou 708 KB de páginas de memória, enquanto que o RSA 2048 alocou 712 KB de memória. Quanto ao algoritmo ECC, todas as variações de chave alocaram 360 KB de páginas de memória. O algoritmo MQQ 160 foi o que menos alocou memória, apesentando ocupação de somente 276 KB. Neste sentido, o algoritmo MQQ 160 ocupou 76% da memória exigida pelo ec-elg_p192 e 39% da memória ocupada pelo RSA 1024.

Analisando a quantidade de páginas de memórias utilizadas por cada algoritmo, nota-se através da Figura 5.13 que o RSA 1024 ocupou 177 páginas e o RSA 2048 ocupou 178 páginas, apenas uma página a mais que o RSA 1024. Todas as variações de chave do ECC ocuparam 90 páginas e o MQQ ocupou 69 páginas. Esses números revelam que o MQQ 160 ocupa 61% de memória a menos do que a exigida pelo RSA 1024 e 23% a menos que a exigida pelo ec-elg_p192, e que o ec-elg_p192 usa 49% menos memória que a exigida pelo

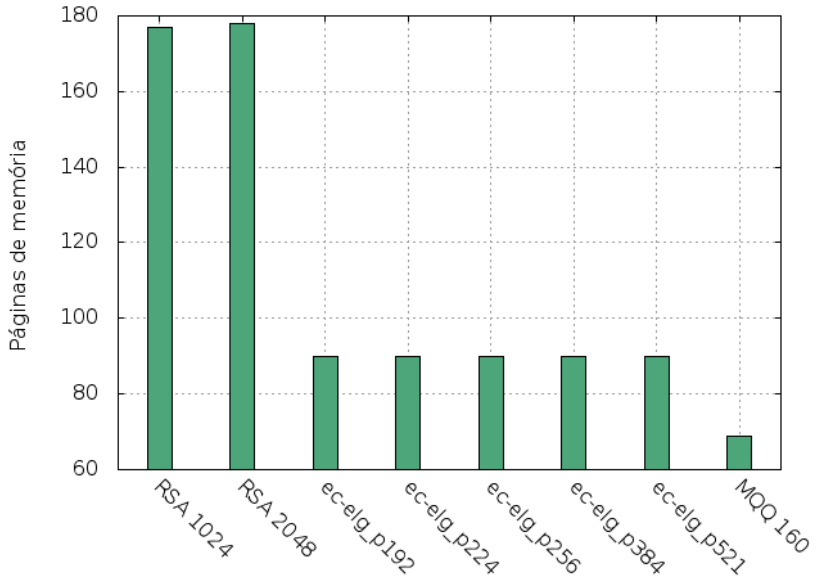

Figura 5.13: Número de páginas de memória - ARM

RSA 1024.

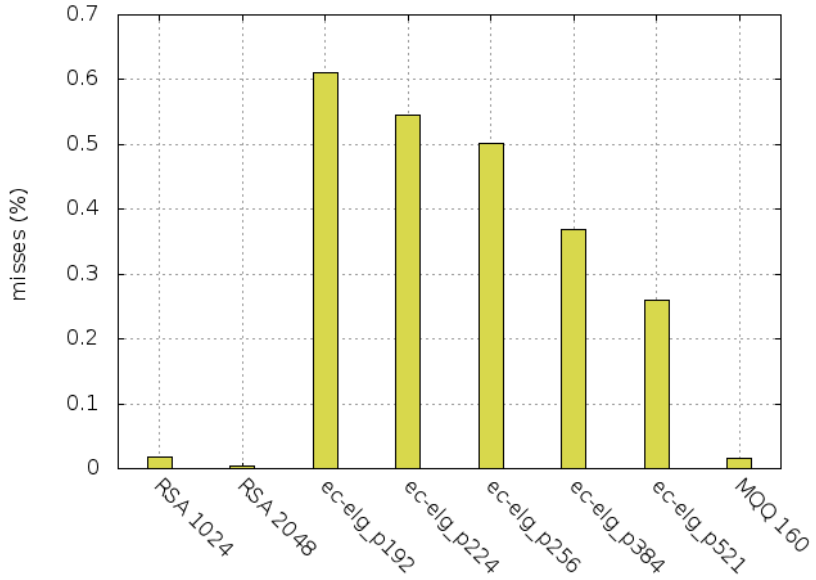

Figura 5.14: Porcentagem de *misses* na cache de instruções - ARM

A Figura 5.14 refere-se à porcentagem de consultas do processador que foram consideradas *miss*, ou seja, falta ou perda na memória cache de instruções, isto é, quando os dados não estão disponíveis na memória cache de instruções, fazendo com que a consulta tenha que ser realizada diretamente na memória RAM. É possível observar que a maioria das per-

das por acesso na cache de instruções ocorrem no algoritmo baseado em curvas. Porém, em geral todos os algoritmos apresentaram baixo percentual de perdas por acesso na cache de instruções. O algoritmo RSA 1024 apresentou uma porcentagem de perdas de 0,018%, frente a 0,61% do ec-elg_p192 e 0,015% do MQQ 160.

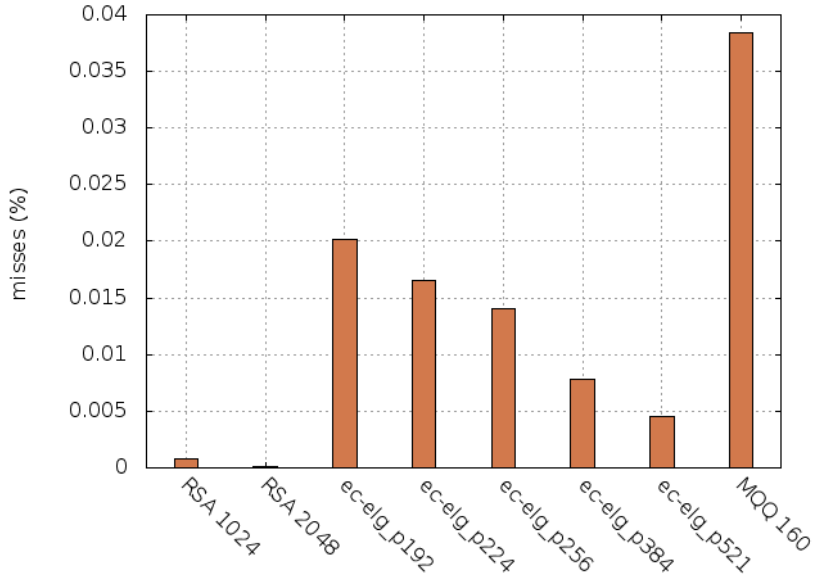

Figura 5.15: Porcentagem de *misses* na cache de dados - ARM

Diferente da porcentagem de *misses* na cache de instruções, o algoritmo MQQ apresenta a maior porcentagem de perdas por acesso na cache de dados. Porém, como pode ser confirmado na Figura 5.15, os níveis alcançados por todos os algoritmos são considerados irrelevantes, pois representam menos de 1% em relação à quantidade de acessos (DIAZ; POSADAS; VILLAR, 2010).

O número de acessos à memória de dados que pode ser visualizado na Figura 5.16 foi consideravelmente maior no algoritmo RSA. Esse número é resultado da soma dos acessos satisfatórios *hits* com os considerados *misses*. Como o número de *misses* foi baixo nesta avaliação, optou-se por não mostrar o número de *hits*, uma vez que esse número é bem próximo do número de acessos. Através da Figura 5.16 é possível visualizar que o número de acessos na cache de instruções foi maior que na cache de dados. Ainda de acordo com a Figura 5.16 e tomando como foco os algoritmos ECC e MQQ, é perceptível que o MQQ teve o menor número de acessos dentre todos os algoritmos avaliados. Ao ser comparado com o algoritmo baseado em curvas, o MQQ 160 apresentou apenas 8% da quantidade de acessos

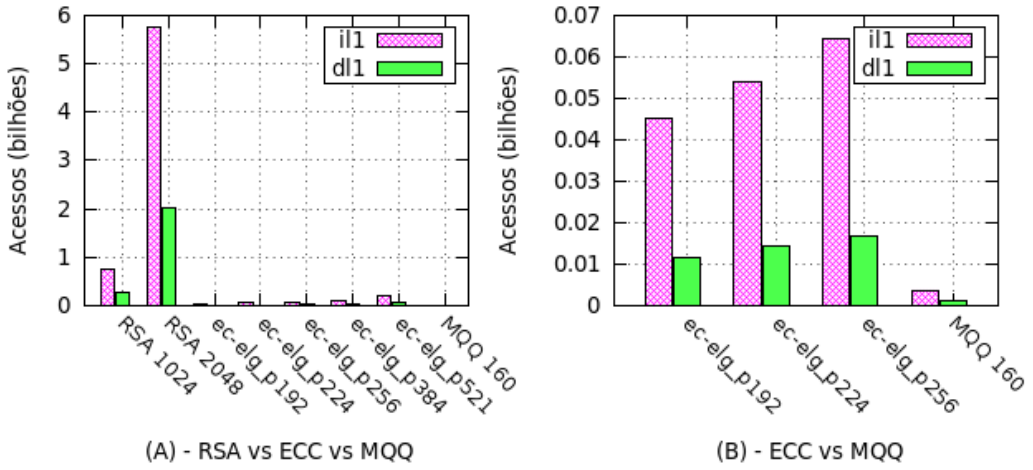

Figura 5.16: Número de Acessos - ARM

do algoritmo ec-elg_p192 na cache de instruções e 9% do número de acessos na cache de dados.

Os dados apresentados sobre uso da cache de instruções e da cache de dados mostram que os programas avaliados não precisam de caches maiores que as usadas na plataforma ARM, e ainda, que há uma boa localização espacial nesses algoritmos. O número elevado de acessos do algoritmo RSA mostra que ele não é adequado para aplicações embarcadas que utilizam a plataforma ARM, e seria ainda mais crítico para RSSF, onde as plataformas têm menor poder computacional. Isso permite concluir que para RSSF, os algoritmos mais adequados seriam o ECC e o MQQ.

As substituições (*replacements*) tanto na cache de instrução (il1) quanto na cache de dados (dl1) foram mínimas, uma vez que a Figura 5.17 e a Figura 5.18 mostram apenas os algoritmos RSA e ECC com dados computáveis nesse critério de avaliação, ainda assim, os valores estão na escala de 10^{-3} e 10^{-5}. Isso equivale dizer que o número de substituições do algoritmo RSA 2048 na cache de dados foi de aproximadamente 0,002 para cada 1000 acessos. Na cache de instrução, o algoritmo ECC apresenta um nível de substituições crescente de acordo com tamanho da chave, além disso, o ECC sempre apresenta níveis maiores que o RSA. Na camada de dados, todos os algoritmos analisados apresentam um número aproximado de substituições, a exceção do MQQ, que nas duas caches analisadas obteve níveis tão baixos que não aparecem nos gráficos.

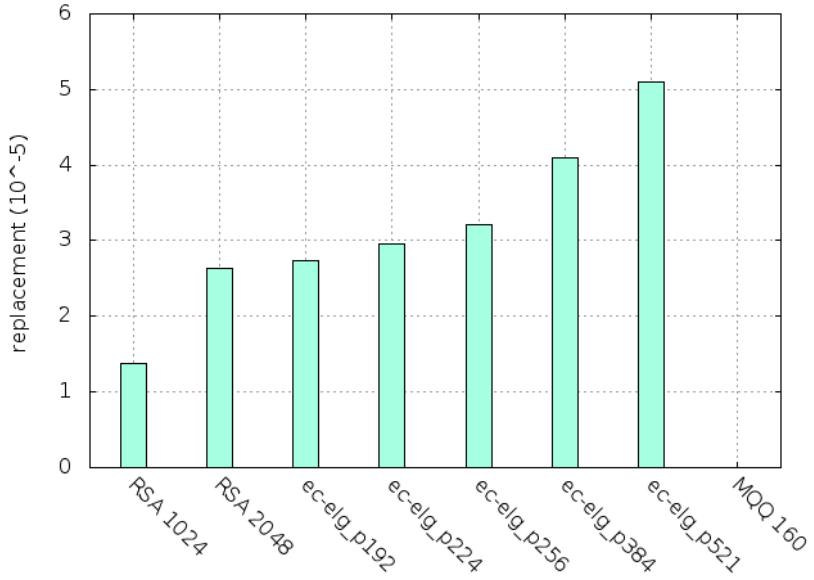

Figura 5.17: Substituições na cache de instruções il1 do ARM

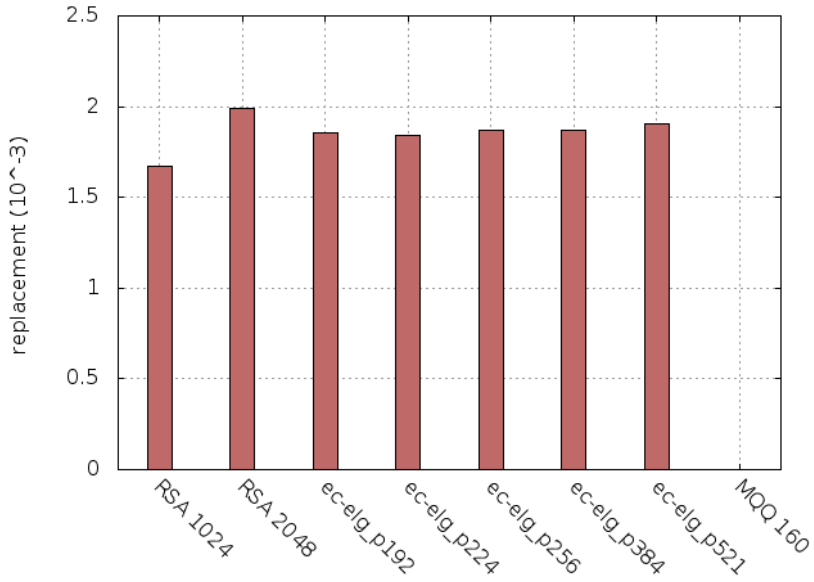

Figura 5.18: Substituições na cache de dados dl1 do ARM

Em relação aos *write-backs* e de acordo com a Figura 5.19, os algoritmos RSA 1024 e ECC mantiveram variação entre 0,014 e 0,016. O algoritmo RSA 2048 obteve níveis de aproximadamente 0,017 e o algoritmo MQQ não apresentou *write-backs*. Os valores referentes a quantidade de *write-backs* também são considerados baixos, uma vez que acontecem a cada 1000 acessos.

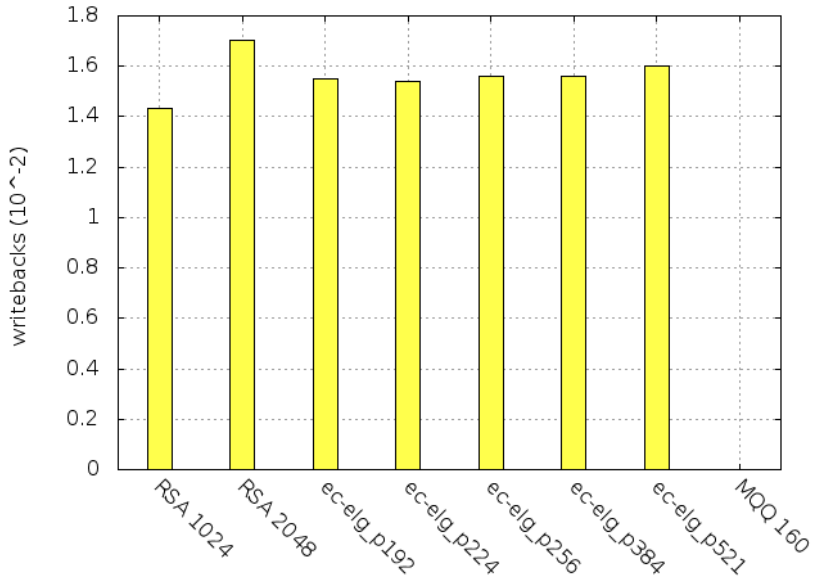

Figura 5.19: write-backs na cache de dados dl1 do ARM

5.5.4 Consumo de Energia

A ferramenta de análise do consumo de energia na plataforma ARM (Sim-Panalyzer) foi desenvolvida com base no conjunto ISA (*Instruction Set Architecture*) da família dos processadores ARM e Alpha, obtendo bons resultados em simulações deste tipo (LIM et al., 2012). Cada componente da arquitetura consome uma quantidade de energia que contribui com o total de energia dissipada na execução de um algoritmo. A Tabela 5.7 mostra os dezenove componentes do simulador Sim-Panalyzer que consomem mais energia na arquitetura.

Para cada algoritmo criptográfico foram realizadas duas análises: a primeira mostra o consumo de energia de cada componente da arquitetura ARM, conforme descritos na Tabela 5.7, com a unidade de medida em *Joule* x ciclos/segundo; a segunda análise mostra o consumo total de energia (real) para cada algoritmo, com a unidade em *Joules*, conforme a Equação 5.1.

$$Consumo\,Total\,(J) = \frac{(Joule * \frac{ciclos}{segundo}) * tempo\,de\,execucao\,(s)}{ciclos} \tag{5.1}$$

A Equação 5.1 mostra que para obtermos o consumo de energia dos componentes em *Joules*, foi preciso multiplicar os resultados da simulação pelo tempo de execução, e então

Tabela 5.7: Componentes Arquiteturais do Sim-Panalyzer

Componentes da Arquitetura	**Significados**
aio	Barramento de endereço da unidade de entrada e saída
dio	Barramento de dados da unidade de entrada e saída
irf	Registradores de inteiro
fprf	Registradores de ponto flutuante
il1	Cache de instruções nível 1
dl1	Cache de dados nível 1
il2	Cache de instruções nível 2
dl2	Cache de dados nível 2
itlb	Tabela de instruções do TLB
dtlb	Tabela da dados do TLB
btb	*Branch Target Buffer*
bimod	Previsor de desvio
ras	Retorno da pilha de endereço
lógico	Circuito lógico aleatório
clock	Relógio gerador de clock do sistema
uarch	Microarquitetura
fpu	Unidade de ponto flutuante
mult	Unidade de multiplicação
alu	Unidade lógica aritmética

dividir pelo número de ciclos executados. O consumo real em *Joule* é mostrado na Tabela 5.9 e nas Figuras 5.22 e 5.23.

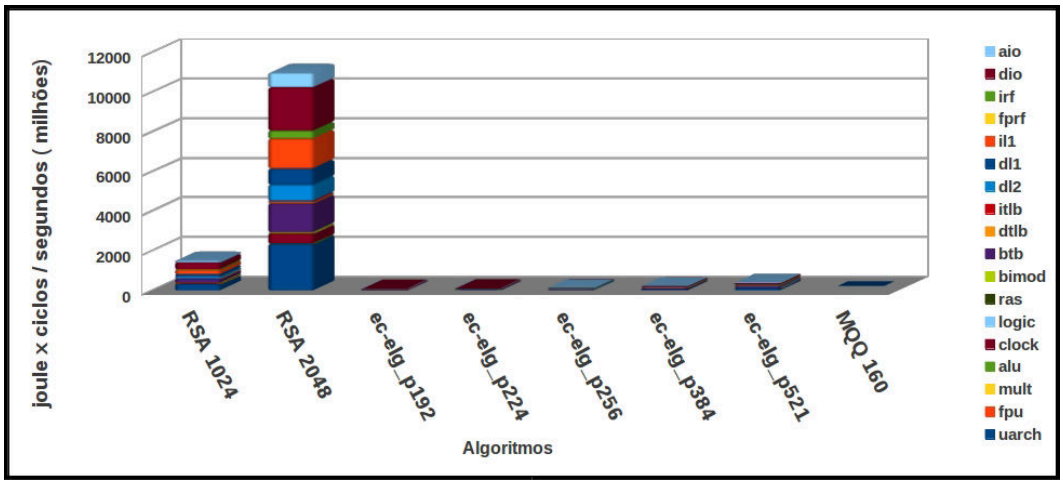

Figura 5.20: Consumo energético dos componentes da arquitetura ARM

É possível observar na Figura 5.20, que o consumo de energia é dividido conforme seu uso nos dezoito diferentes componentes arquiteturais definidos na Tabela 5.7 para o processador ARM. O algoritmo RSA consome quantidade de energia consideravelmente superior ao ECC e MQQ, independente do tamanho das chaves dos algoritmos. Na Figura 5.20 é possível observar também que os principais responsáveis pelo consumo energético do algoritmo RSA 2048 são o DIO (Barramento de dados), IL1 (cache nível 1 de instruções), BTB (Buf-

Tabela 5.8: Porcentagem do consumo energético dos principais componentes da arquitetura

	uarch	**btb**	**il1**	**dio**	**Total**
RSA 1024	21,1%	13,1%	13,5%	21,9%	**69,6%**
RSA 2048	21,5%	13,4%	13,8%	19,9%	**68,6%**
ec-elg_p192	22,8%	14,2%	12,7 %	19,2%	**68,9%**
ec-elg_p224	22,6%	14,1 %	12,6%	20,1%	**69,4%**
ec-elg_p256	22,8%	14,1%	12,7%	19,5%	**69,1%**
ec-elg_p384	22,7%	14,0%	12,9%	19,4%	**69,0%**
ec-elg_p521	22,7%	14,1%	12,9 %	18,8%	**68,5%**
MQQ 160	24,8%	13,2%	11,1%	19,9%	**69,0%**

fer de Predição de Branch) e UARCH (relacionada com a implementação e uso do ISA do processador). A Tabela 5.8 mostra que todos os algoritmos avaliados apresentam resultados semelhantes em relação aos componentes que mais consomem energia. Os dados mostram que os quatro componentes relacionados ao uso de memória IL1 e BTB, transferência de dados pelo barramento DIO e micro arquitetura UARCH, consomem juntos entre 68 e 70% da energia requerida pelos algoritmos analisados.

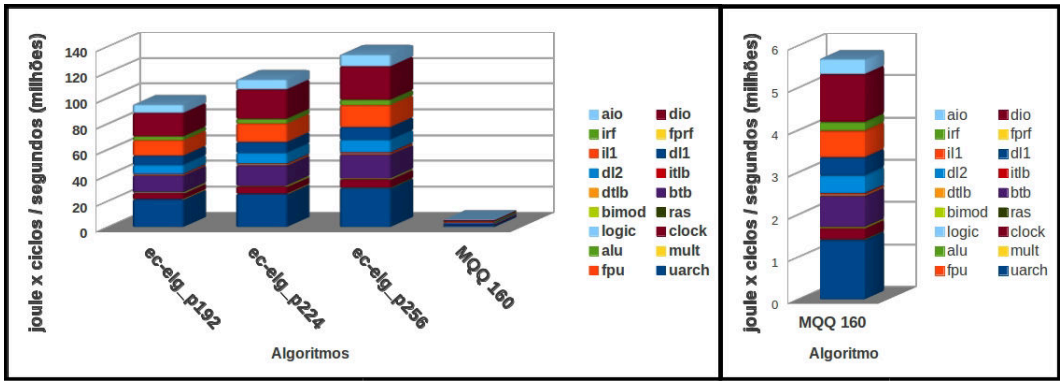

Figura 5.21: Consumo energético dos componentes da arquitetura ARM com foco no ECC e MQQ.

A Figura 5.21 apresenta de forma ampliada o consumo energético dos algoritmo ECC com chave 192, 224 e 256 bits e do algoritmo MQQ 160 bits. É possível observar na Figura 5.21 que os principais consumidores de energia do algoritmo ECC e MQQ também são os componentes DIO, IL1, BTB e UARCH.

A Figura 5.22 e a Tabela 5.9 mostram que o consumo energético total dos algoritmos criptográficos em *Joule* mantém o cenário apresentado anteriormente, uma vez que o algoritmo RSA continua apresentando altos níveis de consumo. Além disso, é importante destacar que o algoritmo RSA não é indicado para sistemas embarcados por não ser escalável, uma vez que o RSA 2048 consome uma quantidade de energia 637% maior que o RSA

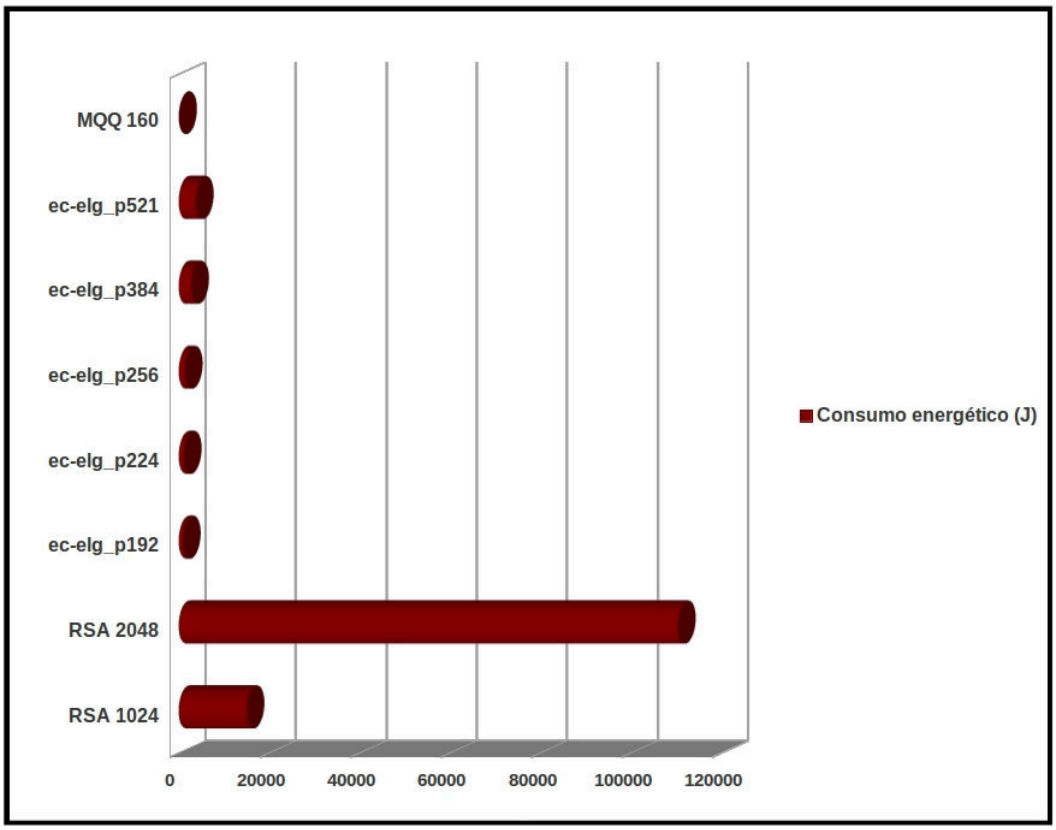

Figura 5.22: Consumo energético dos algoritmos criptográficos (J)

1024. Para fins de comparação, a Tabela 5.9 e a Figura 5.23 mostram que o ec-elg_p224 consome apenas 22,4% mais energia que o ec-elg_p192 e o ec-elg_p256 consome 21,84% mais energia que o ec-elg_p224. Segundo Wang et al. (2012) a implementação do RSA em dispositivos embarcados não é flexível o suficiente para suportar escalabilidade. Os dados revelam que o algoritmo RSA 1024 bits consome aproximadamente 17 vezes mais energia que o algoritmo ec-elg_p192. Em relação ao MQQ 160 bits, o RSA 1024 bits consome uma quantidade de energia 243 vezes superior.

Tabela 5.9: Consumo energético em Joule

RSA 1024	**RSA 2048**	**ec-elg_p192**	**ec-elg_p224**	**ec-elg_p256**	**ec-elg_p384**	**ec-elg_p521**	**MQQ 160**
14985,71	110459,57	889,98	1090,64	1328,90	2732,33	3759,98	61,60

É possível observar na Figura 5.23 o consumo energético dos algoritmos em *Joule* de forma ampliada, tendo como foco os algoritmos ec-elg_p192, 224 e 256 bits e o algoritmo MQQ 160 bits. Os dados revelam que o algoritmo MQQ 160 bits consome apenas 7% da energia consumida pelo algoritmo ec-elg_p192.

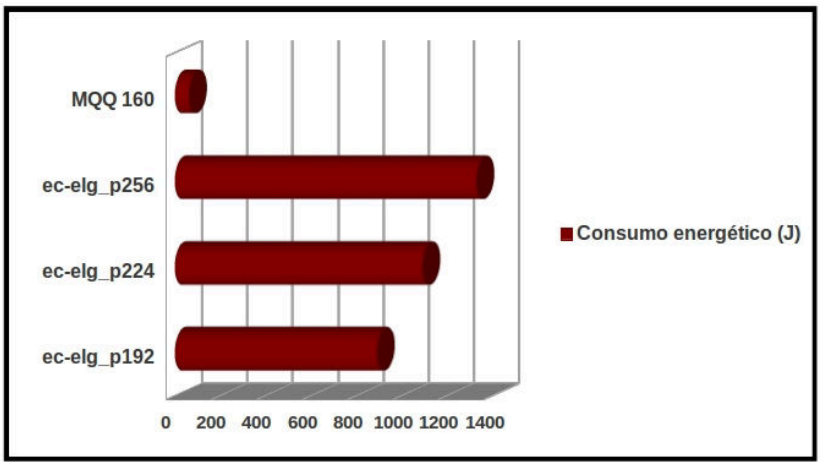

Figura 5.23: Consumo energético dos algoritmos criptográficos com foco no ECC e MQQ (J)

5.6 Resultados Obtidos na Plataforma MSP430

Para simulação do algoritmo MQQ na plataforma MSP430, foi preciso compilar o código escrito na linguagem C utilizando o compilador msp-gcc[2]. A compilação gera um arquivo ".elf"que pode ser lido pelo simulador MSPsim (ERIKSSON et al., 2007). No caso dos algoritmo RSA e ECC que foram baseados na biblioteca RELIC (ARANHA; GOUVEA, 2012), foi necessário executar um script disponível pela própria biblioteca, que invoca o compilador msp-gcc e gera os arquivos compilados dentro da pasta "bin". O tempo de processamento dos algoritmos em segundos, foi calculado supondo um clock de 8 Mhz. Vale notar que sensores mais modernos, como o TinyNode, utilizam microcontroladores MSP430 com 16 MHz de clock (GOUVEA, 2010a). Assim, o tempo necessário para o cálculo dos algoritmos em tais sensores corresponde à metade do cálculo aqui apresentado.

A Figura 5.24 mostra que a quantidade de ciclos do algoritmo RSA 354 bits foi 4 vezes maior que a quantidade de ciclos do algoritmo ECC 158 bits. O algoritmo MQQ 160 bits apresentou uma quantidade de ciclos aproximadamente 25 vezes menor que o algoritmo ECC 158 bits.

O tempo em segundos apresentado pela Tabela 5.10 confirma novamente o melhor desempenho do algoritmo MQQ frente aos algoritmos RSA e ECC, e do ECC frente ao RSA. Ainda de acordo com a Tabela 5.10 o tempo de processamento do algoritmo RSA 354 foi

[2]Compilador msp para códigos escritos na linguagem C. Informações em: http://mspgcc.sourceforge.net/

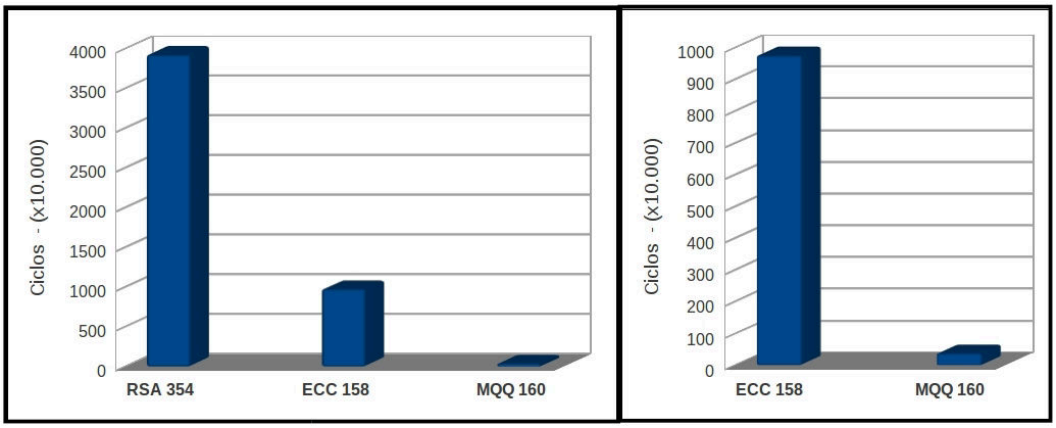

Figura 5.24: Quantidade de ciclos na plataforma MSP430

Tabela 5.10: Tempo de execução na plataforma MSP430

Algoritmo	Tempo (s)
RSA 354	4,91
ec-elg_p158	1,22
MQQ 160	0,05

aproximadamente 300% superior ao tempo de processamento do algoritmo ec-elg_p158. O algoritmo MQQ 160 apresentou um tempo de processamento 98 vezes menor que o tempo apresentado pelo algoritmo RSA 354 e aproximadamente 24 vezes menor que o tempo do algoritmo ec-elg_p158.

5.7 Resultados Obtidos na Plataforma AVR Atmega128

Nesta seção, foi avaliada a implementação do MQQ 160 na plataforma AVR Atmega 128 com processador de 8 bits. Devido a restrições da plataforma e limitações do simulador AVRORA ao executar os códigos escritos com base na biblioteca RELIC, não foi possível simular os algoritmos RSA e ECC, portanto, nesta seção comparamos os resultados obtidos na simulação do MQQ 160, com os dados obtidos na literatura, da avaliação de desempenho com o algoritmo ECC, realizada por Yan e Shi (2006).

Para simular o código do algoritmo MQQ no Avrora foi preciso compilar o código escrito na linguagem C através do compilador avr-gcc[3]. Nesta compilação, o parametro "-

[3]Compilador AVR para códigos escritos na linguagem C. Maiores informações em: http://www.nongnu.org/avr-libc/

mmcu=atmega128" garante que esse código será voltado para plataformas Atmega128. Ao final, um arquivo binário ".elf" é gerado. O arquivo ".elf" deve ser transformado em ".od" através da ferramenta avr-objdump[4] que faz parte da biblioteca avr-libc. Essa ferramenta têm a função de transformar o arquivo assembler em um arquivo de texto suportado pelo simulador Avrora.

O trabalho de Yan e Shi (2006) avaliou o algoritmo ECC na plataforma AVR Atmega128. Os autores também utilizaram o simulador Avrora na avaliação de desempenho. Considerando a plataforma Atmega128 com 16 Mhz, os dados mostraram que o ECC 163 (YAN; SHI, 2006) foi executado em 6,94s. A Figura 5.25 mostra que o MQQ 160 foi executado em 0,21s, o que equivale a um tempo aproximadamente 33 vezes menor que o tempo do ECC.

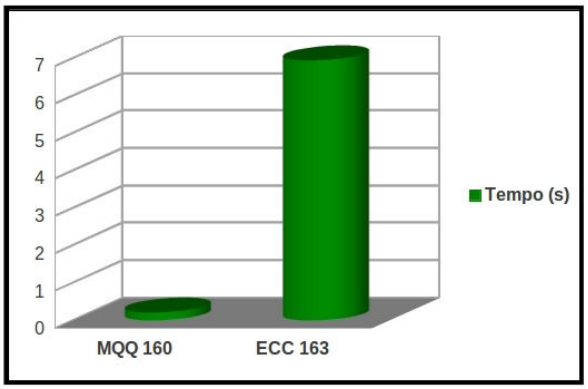

Figura 5.25: Tempo na plataforma AVR Atmega128

5.8 Comparação dos Resultados

A Tabela 5.11 faz um comparativo entre os tempos dos algoritmos criptográficos nas plataformas Desktop, ARM, MSP430 e AVR. Os dados mostram que em todas as plataformas, o algoritmo MQQ apresentou melhores resultados que os algoritmos RSA e ECC. A menor distância entre os tempos do MQQ ocorreu na comparação com o ec-elg_p192 na plataforma ARM, onde o algoritmo MQQ foi 16 vezes mais rápido. A maior relação ocorreu na comparação com o RSA 1024 também na plataforma ARM, onde o algoritmo MQQ apresentou um tempo de execução 230 vezes menor. O algoritmo ec-elg_p192 se mostrou mais rápido

[4]http://www.nongnu.org/avr-libc/

que o RSA 1024 em todas as plataformas, sendo que a maior relação ocorreu na plataforma ARM, onde o ec-elg_p192 foi 14 vezes mais rápido.

Tabela 5.11: Comparação do tempo nas plataformas Desktop, ARM, MSP430 e AVR

Plataforma	Algoritmos	Tempo	Relação com MQQ	ECC vs RSA
Desktop	RSA 1024	28,5	75	2,5
	ec-elg_p192	11,5	30	–
	MQQ 160	0,38	–	–
ARM	RSA 1024	4431,8	230	14
	ec-elg_p192	314,9	16	–
	MQQ 160	19,2	–	–
MSP430	RSA 354	4,91	98	4
	ec-elg_p158	1,22	24	–
	MQQ 160	0,05	–	–
AVR	ECC 163 (YAN; SHI, 2006)	6,94	33	–
	MQQ 160	0,21	–	–

A Figura 5.26 mostra os tempos de execução dos algoritmos em todas as plataformas avaliadas. Nesse gráfico, o tempo de execução em cada plataforma foi dividido pelo número de bytes da mensagem, de forma que esses valores possam ser comparados com outros trabalhos. É possível observar na Figura 5.26, que as distâncias entre os tempos de execução dos algoritmos são similares independente das plataformas, ou seja, o RSA apresenta maiores tempos e o MQQ apresenta os menores tempos. O algoritmo ECC sempre obteve melhores resultados que o RSA, porém apresentou tempo maior que o MQQ.

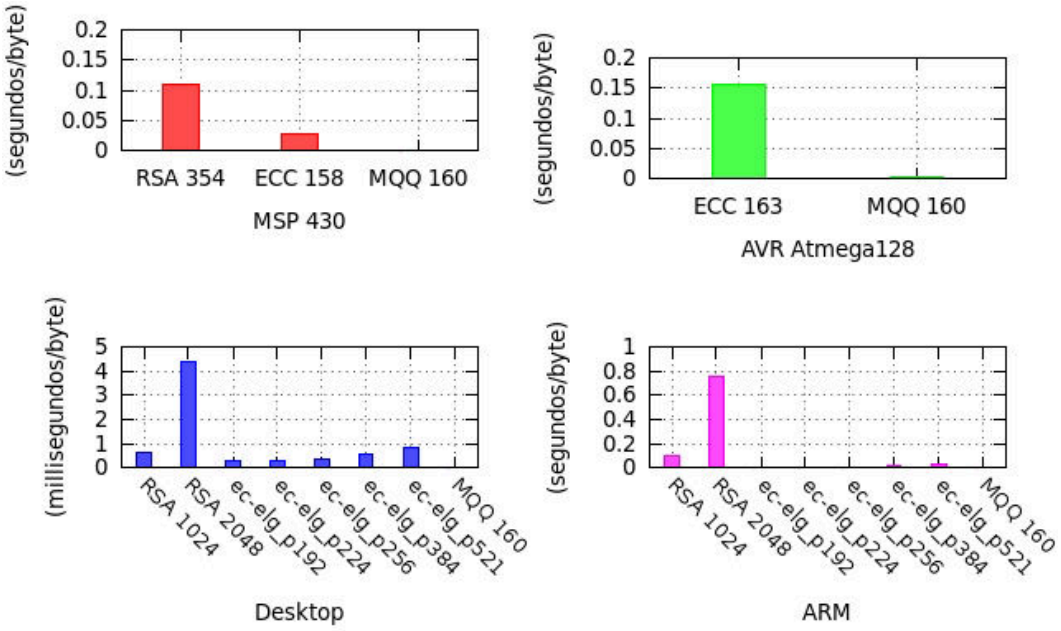

Figura 5.26: Tempo de execução divido por byte

Os autores Branovic, Giorgi e Martinelli (2004) analisaram os algoritmos RSA e ECC

usando o SimpleScalar. O chip ARM considerado foi o XScale, que contém 32KB nas caches de instrução e dados da L1. No trabalho (BRANOVIC; GIORGI; MARTINELLI, 2004), o algoritmo RSA 1024 apresentou uma quantidade de ciclos inferior ao obtido pelo ECC. A quantidade de *misses* por instruções do RSA 1024 também foi menor que a quantidade apresentada pelo ec-elg_p192. Neste último item, os resultados apresentados nesta dissertação foram similares. Branovic, Giorgi e Martinelli (2004) simularam os algoritmos na plataforma ARM através do SimpleScalar. A arquitetura simulada foi um processador StrongArm SA-1110 com modificações no simulador para adicionar uma cache da arquitetura Intel XScale. As medições de tempo são dadas em bilhões de ciclos. Os algoritmos comparados foram ec-elg_p192 e RSA 1024. Os resultados obtidos por Branovic, Giorgi e Martinelli (2004) mostram que o algoritmo RSA 1024 foi mais rápido que o algoritmo ec-elg_p192.

Chatterjee, De e Gupta (2011) usaram a plataforma Desktop com a configuração (Core 2 DUO CPU T6400@2.00GHz com 4GB RAM), além disso sua simulação considerou os algoritmos de curvas elípticas baseados em corpos binários. Os algoritmos comparados foram ec-elg_p192 e ECC 163.

O trabalho desenvolvido por Maia (2010) avaliou o desempenho dos algoritmo RSA e MQQ em plataformas embarcadas de sensores TelosB e MicaZ. Os resultados mostraram que na plataforma TelosB, o MQQ foi aproximadamente 909 vezes mais rápido que o RSA na cifragem e 5.470 vezes mais veloz na decifragem. O autor ainda informa que a mensagem utilizada pelo MQQ foi de 160 bits contra uma mensagem de 8 bits usada pelo RSA, portanto caso fosse utilizada mensagens do mesmo tamanho, essa diferença aumentaria consideravelmente. Maia (2010) usou a plataforma de sensor TelosB, e realizou as medições de tempo através de um multimetro digital. Os algoritmos comparados foram RSA 1024 e MQQ 160.

Os autores El-Hadedy, Gligoroski e Knapskog (2008) avaliaram o algoritmo MQQ 160 com o RSA 1024 na plataforma Desktop e FPGA. Os dados mostraram que a quantidade de núcleos influenciaram nos resultados, uma vez que na plataforma Desktop com um processador, o algoritmo MQQ realizou a cifragem em 140.485 ciclos. Já na plataforma desktop com dois processadores, o MQQ realizou a cifragem em 80.105 ciclos. O RSA 1024 realizou a cifragem em 119.800 ciclos. Em relação à decifragem, o algoritmo MQQ se mostrou

275 vezes mais rápido que o RSA. Segundo El-Hadedy, Gligoroski e Knapskog (2008) na plataforma FPGA o algoritmo MQQ se mostrou 10.000 vezes mais rápido que o RSA, apresentando velocidade comparável com a do algoritmo simétrico AES.

É possível observar através da Tabela 5.12 que de forma geral, os resultados obtidos neste trabalho, nas comparações entre RSA e MQQ são similares aos encontrados nas avaliações de desempenho da literatura, isto é, o desempenho do MQQ supera em várias ordens de grandeza o desempenho do RSA, e na maioria dos casos o ECC também supera o desempenho do RSA. Porém, nenhum autor comparou o desempenho do MQQ com o ECC, tornando inviável uma comparação direta entre os resultados obtidos neste estudo com a literatura. Apesar da dificuldade em se comparar diretamente os resultados dessa dissertação com a literatura, a Tabela 5.12 foi construída de forma a comparar os trabalhos que utilizaram plataformas similares. Os dados da Tabela 5.12 foram obtidos através da divisão do tempo de execução de cada algoritmo pela quantidade de bytes das mensagens usadas na cifragem e decifragem, portanto a maioria dos dados estão em **"ms/byte"**, á exceção dos dados na plataforma ARM, que estão em **"ciclos/byte"**. Essa conversão foi realizada para equalizar os diferentes tamanhos de mensagem usadas pelos autores que fazem parte dessa comparação.

Tabela 5.12: Comparação com resultados obtidos na literatura - Tempo de processamento/byte

Trabalho	Plataforma	RSA	ECC	MQQ	Relação (tempo)
(MAIA, 2010)	TelosB	72540	–	48,4	1486,4
Resultados obtidos nesse trabalho	MSP430	111	–	1,13	98,23
(CHATTERJEE; DE; GUPTA, 2011)		—	1,19	—	
(EL-HADEDY; GLIGOROSKI; KNAPSKOG, 2008)	Desktop	3,07	–	0,15	20,3
Resultados obtidos nesse trabalho		0,64	–	0,008	75
(BRANOVIC; GIORGI; MARTINELLI, 2004)	ARM	0,08	0,3	—	3,75
Resultados obtidos nesse trabalho		0,88	0,06	—	14,6

É importante salientar que o principal dado da Tabela 5.12 é a relação de tempo entre os algoritmos, uma vez que o objetivo dessa dissertação é comparar o desempenho entre algoritmos criptográficos, e não entre implementações do mesmo algoritmo. Os resultados desse trabalho, quando comparados a outras avaliações presentes na literatura, apresentaram o seguinte cenário: O pesquisador Maia (2010) afirmou que o MQQ 160 na plataforma TelosB é aproximadamente 1486 vezes mais rápido que o RSA 1024, porém nossa comparação mostrou que essa diferença é de aproximadamente 98 vezez. Em contrapartida, El-Hadedy, Gligoroski e Knapskog (2008) informou que na plataforma Desktop, o MQQ 160 é aproxi-

madamente 20 vezes mais veloz que o RSA 1024, porém os dados obtidos nesta avaliação mostraram que essa relação chega a ser de 75 vezes. Branovic, Giorgi e Martinelli (2004) mostrou que na plataforma ARM, o algoritmo RSA foi mais rápido que o ECC, porém este trabalho mostrou que na mesma plataforma, o ECC foi 14,6 vezes mais rápido que o RSA. As diferenças de tempo entre as implementações do mesmo algoritmo e entre os diferentes trabalhos, possivelmente são fruto das próprias implementações, bibliotecas, plataformas, entre outras variáveis que podem influenciar no tempo de execução.

Capítulo 6

Conclusões

O primeiro capítulo descreveu quais os objetivos, justificativa e metodologia deste trabalho, além dos passos a serem seguidos a fim de apontar o algoritmo criptográfico de chave pública que seja mais eficiente em RSSF em relação a consumo energético, tempo de processamento e consumo de memória. No segundo capítulo, foram definidas as características de uma RSSF, bem como suas aplicações, vulnerabilidades de segurança, além de apontar as principais medidas de segurança atualmente utilizadas nesse tipo de rede. O terceiro capítulo abordou sobre o surgimento e uso dos algoritmos criptográficos RSA, ECC e MQQ, bem como o funcionamento de cada algoritmo e as funções matemáticas envolvidas na cifragem e decifragem de mensagens. O quarto capítulo trouxe uma análise dos trabalhos acadêmicos que vêm sendo realizados a fim de apontar algoritmos criptográficos assimétricos que sejam adequados a RSSF.

A avaliação de desempenho dos algoritmos criptográficos RSA, ECC e MQQ foi realizada nas plataformas Desktop, ARM, MSP430 e AVR Atmega128. A plataforma MSP430 é usada em sensores como TelosB e Tmote Sky. A plataforma AVR é usada em sensores como MicaZ, Mica2 e IRIS. Apesar da plataforma ARM não ser efetivamente usadas em RSSF, pesquisas vêm sendo desenvolvidas para possibilitar o uso destes dispositivos nessas redes, logo, a avaliação feita nessa plataforma poderá servir de base para pesquisas futuras. A avaliação de desempenho realizada nas plataformas Desktop e ARM mostrou que o algoritmo MQQ obteve os melhores resultados frente aos algoritmos RSA e ECC. Os dados revelam que o MQQ 160 é mais veloz, consome menor quantidade de memória e requer menor poder de processamento que seus concorrentes RSA 1024 e ec-elg_p192. Além disso, este trabalho mostrou que o algoritmo recém criado MQQ consome apenas 7% da energia consumida

pelo algoritmo ECC, que atualmente é considerado o algoritmo criptográfico assimétrico mais indicado para sistemas embarcados. Os dados revelaram também que os componentes arquiteturais DIO, IL1, BTB e UARCH são críticos na plataforma ARM, indicando que possíveis otimizações voltadas a esses componentes podem melhorar consideravelmente o desempenho de algoritmos criptográficos. Na plataforma MSP430, o algoritmo MQQ foi 98 vezes mais rápido que o RSA e 24 vezes mais rápido que o ECC. Nesta mesma plataforma, o algoritmo ECC foi 4 vezes mais rápido que o RSA. Os resultados mostraram também que na plataforma AVR, o MQQ se mostrou 33 vezes mais rápido que o ECC.

A avaliação de desempenho realizada nessa dissertação levou em consideração quatro plataformas distintas, comparando algoritmos assimétricos padronizados (RSA), algoritmos cujo uso em sistemas embarcados está crescendo rapidamente (ECC), além do promissor (MQQ). Os resultados mostraram que o ECC têm motivos para ser amplamente utilizado em plataformas embarcadas usadas em RSSF, pois além de ser mais rápido que o RSA, consumir menor quantidade de energia e memória, esse algoritmo já foi avaliado por criptógrafos de todo o mundo, portanto é um algoritmo eficiente e seguro. O algoritmo MQQ se mostrou promissor, uma vez que alcançou excelentes níveis de consumo de energia, uso de processador e memória na plataforma ARM e tempo de processamento em todas as plataformas avaliadas, porém devido sua recente publicação, ainda não pode ser considerado realmente seguro.

No que tange a algoritmos criptográficos assimétricos para plataformas embarcadas usadas em RSSF: (i) O algoritmo RSA não é indicado, (ii) O algoritmo ECC é eficiente e está padronizado; (iii) O algoritmo MQQ é promissor.

6.1 Produção Científica

- Gustavo S. Quirino, Admilson R. L. Ribeiro and Edward David Moreno (2012). Asymmetric Encryption in Wireless Sensor Networks, Wireless Sensor Networks - Technology and Protocols, Mohammad A. Matin (Ed.), ISBN: 978-953-51-0735-4, InTech, Available from: http://www.intechopen.com/books/wireless-sensor-networks-technology-and-protocols/asymmetric-encryption-in-wireless-sensor-networks

- Gustavo S. Quirino and Edward David Moreno (2013). Architectural Evaluation of Algorithms RSA, ECC and MQQ in ARM Processors - International Journal of Computer Networks & Communications (IJCNC). Volume 5, Número 2. ISSN 0975 - 2293. Março de 2013.

- Gustavo S. Quirino and Edward David Moreno (2013). Architectural Evaluation of Asymmetric Algorithms in ARM Processors - 2nd International Conference on Solid-State and Integrated Circuit (ICSIC 2013). Vancouver, Canada; April 13-14 2013. http://www.icsic.org. **(Aceito. Esse artigo será publicado também no International Journal of Electronics and Electrical Engineering (IJEEE,ISSN: 2301-380X))**

6.2 Trabalhos Futuros

Como trabalhos futuros planejamos (i) estudar estratégias de implementação da geração de chave em plataformas embarcados, uma vez que esta análise considerou os processos de cifragem e decifragem (ii) estudar técnicas de programação que diminuam o consumo de energia dos componentes apresentados nesse trabalho como críticos na plataforma ARM (iii) Comparar os resultados obtidos nas simulações em plataformas reais (iv) avaliar protocolos de segurança que utilizem os algoritmos RSA, ECC e MQQ,e (v) estudos mais detalhados do nível de segurança do algoritmo MQQ.

REFERÊNCIAS

AHLAWAT, R.; GUPTA, K.; PAL, S. From mq to mqq cryptography: Weaknesses new solutions. In: *Western European Workshop on Research in Cryptology*. [S.l.: s.n.], 2009.

AHMAD, S.; BEG, M. R.; ABBAS, Q. Energy Saving Secure Framework for Sensor Network using Elliptic Curve Cryptography. *IJCA Special Issue on Mobile Ad-hoc Networks*, p. 167–172, 2010.

AMARA, S. O.; BEGHDAD, R.; OUSSALAH, M. Securing wireless sensor networks: A survey. *EDPACS*, Taylor & Francis, v. 47, n. 2, p. 6–29, 2013.

AMIN, F.; JAHANGIR, A. H.; RASIFARD, H. Analysis of Public-Key Cryptography for Wireless Sensor Networks Security. *World Academy of Science, Engineering and Technology*, v. 31, n. July, p. 530–535, 2008.

ARANHA, D. et al. Implementação eficiente de criptografia de curvas elípticas em sensores sem fio. In: SBC. *VIII Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais*. [S.l.], 2008. p. 173–186.

ARANHA, D. F.; GOUVEA, C. P. L. *RELIC is an Efficient LIbrary for Cryptography*. 2012. `http://code.google.com/p/relic-toolkit/`.

ASZTOR, B. P. et al. Selective code dissemination in mobile wireless sensor networks. In: *ACM/IFIP/USENIX Middleware - Leuven, Belgium*. [S.l.: s.n.], 2008. p. 113–115.

AUSTIN, T. An armload of simplescalar/arm. Este release está disponível em: http://www.simplescalar.com/docs/ANNOUNCE-ARM.txt. setembro 2012.

AUSTIN, T.; LARSON, E.; ERNST, D. Simplescalar: An infrastructure for computer system modeling. *Computer*, IEEE, v. 35, n. 2, p. 59–67, 2002.

BATINA, L. et al. Public-Key Cryptography on the Top of a Needle. In: *2007 IEEE International Symposium on Circuits and Systems*. Ieee, 2007. p. 1831–1834. ISBN 1-4244-0920-9. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4253017>.

BAUER, G.; POTISK, P.; TILLICH, S. Comparing block cipher modes of operation on MICAz sensor nodes. In: *17th Euromicro International Conference on Parallel, Distributed and Network-based Processing - Washington, DC, USA*. [S.l.]: IEEE Computer Society, 2009. p. 371–378.

BERKELEY.EDU. *OpenWSN Implementing the Internet of Things*. 2013. `http://openwsn.berkeley.edu/wiki/TelosB`.

BLAKE, I.; SEROUSSI, G.; SMART, N. *Elliptic Curves in Cryptography*. [S.l.]: Cambridge, 1999.

BOYLE, D.; NEWE, T. Securing Wireless Sensor Networks: Security Architectures. *Journal of Networks*, v. 3, n. 1, p. 65–77, jan. 2008. ISSN 1796-2056. Disponível em: <http://www.academypublisher.com/ojs/index.php/jnw/article/view/998>.

BRANOVIC, I.; GIORGI, R.; MARTINELLI, E. A Workload Characterization of Elliptic Curve Cryptography Methods in Embedded Environments. *ACM SIGARCH Computer Architecture News*, v. 32, n. 3, 2004.

CARVALHO, A.; LEON, F. Ponce de et al. Grandes desafios da pesquisa em computação no brasil–2006–2016. *SEMINÁRIO - GRANDES DESAFIOS DE PESQUISA EM COMPUTAÇÃO NO BRASIL*, 2006. Disponível em: <www.gta.ufrj.br/rebu/arquivos/SBC-Grandes.pdf>.

CASOLA, V. et al. Analysis and comparison of security protocols in wireless sensor networks. In: IEEE. *30th IEEE Symposium on Reliable Distributed Systems Workshops (SRDSW)*. [S.l.], 2011. p. 52–56.

CAVALCANTE, T. M. et al. Avaliação de Desempenho dos Algoritmos Criptográficos Skipjack e RC5 para Redes de Sensores sem Fio. In: *SBCUP - III Simpósio Brasileiro de Computação Ubíqua e Pervasiva*. Natal-RN: [s.n.], 2011.

CHAN, H.; PERRIG, A. Security and privacy in sensor networks. *Computer*, IEEE, v. 36, n. 10, p. 103–105, 2003.

CHATTERJEE, K.; DE, A.; GUPTA, D. Software Implementation of Curve based Cryptography for Constrained Devices. *International Journal of Computer Applications*, v. 24, n. 5, p. 18–23, 2011.

CHEN, X. et al. Sensor network security: a survey. *IEEE Communications Surveys & Tutorials*, v. 11, n. 2, p. 52–73, 2009. ISSN 1553-877X. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5039583>.

CORPORATION, A. *ATmega128L data Sheet*. 2006. Disponível em: <http://www.atmel.com/dyn/resources/prod_documents/doc2467.pdf>.

DIAZ, L.; POSADAS, H.; VILLAR, E. Obtaining memory address traces from native co-simulation for data cache modeling in systemc. In: *XXV Conference on Design of Circuits and Integrated Systems (DCIS2010)*. [S.l.: s.n.], 2010.

DOUCEUR, J. The sybil attack. *Peer-to-peer Systems*, Springer, p. 251–260, 2002.

DU, W. et al. Key Predistribution Scheme for Sensor Networks Using Deployment Knowledge. In: *IEEE Trans. Depend. Secure 2006*. [S.l.: s.n.], 2006. p. 62–77.

EL-HADEDY, M.; GLIGOROSKI, D.; KNAPSKOG, S. High performance implementation of a public key block cipher-mqq, for fpga platforms. In: IEEE. *Reconfigurable Computing and FPGAs, 2008. ReConFig'08. International Conference on*. [S.l.], 2008. p. 427–432.

ERIKSSON, J. et al. Mspsim - an extensible simulator for msp430-equipped sensor boards. In: *European Conference on Wireless Sensor Networks (EWSN)*. Delft, The Netherlands: [s.n.], 2007.

FAUGERE, J. et al. Analysis of the mqq public key cryptosystem. *Cryptology and Network Security*, Springer, p. 169–183, 2010.

GANESAN, S. P. An Authentication Protocol For Mobile Devices Using Hyperelliptic Curve Cryptography. *International J. of Recent Trends in Engineering and Technology*, v. 3, n. 2, p. 2–4, 2010.

GIACOMIN, J.; VASCONCELOS, F. Qualidade da medição de intensidade de sinal nas comunicações de uma rede de sensores sem fios: uma abordagem da camada física. In: *The 25th Conference on Computer Communications Sponsored by IEEE Communications Society, Barcelona, Catalunya, Spain*. [S.l.: s.n.], 2006.

GLIGOROSKI, D. et al. The digital signature scheme mqq-sig. Arxiv preprint arXiv:1010.3163, 2010.

GLIGOROSKI, D.; MARKOVSKI, S.; KNAPSKOG, S. A public key block cipher based on multivariate quadratic quasigroups. *Arxiv preprint arXiv:0808.0247*, 2008.

GOUVEA, C. L. P. *Implementação em Software de Criptografia Baseada em Emparelhamentos para Redes de Sensores Usando o Microcontrolador MSP430*. 106 p. Dissertação (mestrado) — Unicamp, 2010.

GOUVEA, C. P. L. Implementação em Software de Criptografia Baseada em Emparelhamentos para Redes de Sensores Usando o Microcontrolador MSP430. *X Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais*, p. 531–538, 2010.

GROBSCH, J.; SZEKELY, A.; TILLICH, S. The Energy Cost of Crypto-graphic Key Establishment in Wireless Sensor Networks. In: *2nd ACM Symposium on Information, Computer and Communications Security*. [S.l.: s.n.], 2007. p. 380–382.

GURA, N. et al. Comparing Elliptic Curve Cryptography and RSA on 8-bit CPUs. In: *In Proceedings of the 2004 Workshop on Cryptographic Hardware and Embedded Systems (CHES 2004), Boston Marriott Cambridge Cambridge (Boston)August*. [S.l.: s.n.], 2004.

HANKERSON, D.; VANSTONE, S.; MENEZES, A. *Guide to elliptic curve cryptography*. [S.l.]: Springer-Verlag New York Inc, 2004.

HEALY, M. et al. Efficiently securing data on a wireless sensor network. *Journal of Physics Conference Series*, v. 76, 2007.

HODJAT, A. et al. HW-SW Codesign of a Hyperelliptic Curve Cryptosystem using a uCode Instruction Set Coprocessor. *Elsevier Science Integration the VLSI Journal*, v. 40, p. 45–51, 2006.

HODJAT, A.; VERBAUWHEDE, I. The energy cost of secrets in ad-hoc networks. In: *IEEE Circuits and Systems Workshop on Wireless Communications and Networking*. [S.l.: s.n.], 2002. p. 4.

IEEE. *IEEE 802.15.4 - Wireless Medium Access Control (MAC)and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs)*. [S.l.], 2011. Disponível em: <http://standards.ieee.org/getieee802/download/802.15.4-2006.pdf>.

KARLOF, C.; SASTRY, N.; WAGNER, D. Tinysec: a link layer security architecture for wireless sensor networks. In: ACM. *Proceedings of the 2nd international conference on Embedded networked sensor systems*. [S.l.], 2004. p. 162–175.

KARLOF, C.; WAGNER, D. Secure routing in wireless sensor networks: Attacks and countermeasures. In: IEEE. *1st IEEE International Workshop on Sensor Network Protocols and Applications*. [S.l.], 2003. p. 113–127.

KOBLITZ, N. Elliptic curve cryptosystems. *Mathematics of computation*, v. 48, n. 177, p. 203–209, 1987.

KOC, C. *About Cryptographic Engineering*. [S.l.]: Springer, 2009.

LAW, A.; KELTON, W.; KELTON, W. *Simulation modeling and analysis*. [S.l.]: McGraw-Hill New York, 1991.

LAW, Y. W.; DOUMEN, J.; HARTEL, P. Survey and benchmark of block ciphers for wireless sensor networks. *ACM Transactions on Sensor Networks*, v. 2, n. 1, p. 65–93, fev. 2006. ISSN 15504859. Disponível em: <http://portal.acm.org/citation.cfm?doid=1138127.1138130>.

LE, X. et al. An Energy-Efficient Access Control Scheme for Wireless Sensor Networks based on Elliptic Curve Cryptography. *Journal of Communications and Networks, Special Issues on Secure Wireless Networking*, v. 1, 2009.

LE, X.-H. et al. An Energy- Efficient Secure Routing and Key Management Scheme for Mobile Sinks in Wireless Sensor Networks using Deploying Knowledge. *Journal of Sensor, Special Issue Wireless Sensor Technologies and Applications*, v. 8, p. 7753–7782, 2008.

LE, X.-H. et al. A Secure Coordination - based Data Dissemination for Mobile sinks in Sensor Networks. In: *IEICE Transaction on Communication*. [S.l.: s.n.], 2009. p. Vol E92–B(01).

LE, X. H. et al. Public Key Cryptography - based Security Scheme for Wireless Sensor Networks in Healthcare. In: *4th International Conference on Ubiquitous Information Management and Communication*. [S.l.: s.n.], 2010. p. 1–7. ISBN 9781605588933.

LEVIS, P. et al. Tossim: Accurate and scalable simulation of entire tinyos applications. In: ACM. *Proceedings of the 1st international conference on Embedded networked sensor systems*. [S.l.], 2003. p. 126–137.

LIM, C. et al. New techniques for improving the practicality of an svm-based speech/music classifier. In: IEEE. *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*. [S.l.], 2012. p. 1657–1660.

LORINCZ, K. *MoteTrack User's Manual v2.1*. [S.l.], julho 2006. Disponível em: <http://www.eecs.harvard.edu/ konrad/projects/motetrack/manual/MoteTrack-Manual-2.1.html>.

LOUREIRO, A. et al. Redes de sensores sem fio. In: *Simpósio Brasileiro de Redes de Computadores (SBRC)*. [S.l.: s.n.], 2003. p. 179–226.

LUK, M. et al. Minisec: a secure sensor network communication architecture. In: IEEE. *Information Processing in Sensor Networks, 2007. IPSN 2007. 6th International Symposium on*. [S.l.], 2007. p. 479–488.

MAIA, R. *Análise da viabilidade da implementação de algoritmos pós-quânticos baseados em quase-grupos multivariados quadráticos em plataformas de processamento limitadas.* Dissertação (Mestrado) — USP, 2010.

MAIA, R.; BARRETO, P.; OLIVEIRA, B. Implementation of multivariate quadratic quasigroup for wireless sensor network. *Transactions on computational science XI*, Springer, p. 64–78, 2010.

MARGI, M. S.; JR, M.; BARRETO, T. C. M. B. C. Segurança em Redes de Sensores Sem Fio. In: *Simpósio Brasileiro em Segurança da Informação*. [s.n.], 2009. p. 149–194. Disponível em: <www.lbd.dcc.ufmg.br/colecoes/sbseg/2009/042.pdf>.

MEMSIC. *Mote Processor Radio Mote Interface Boards User Manual - Document Part Number: 7430-0021-09 Rev A*. jan. 2012. Disponível em: <http://www.memsic.com/support/documentation/wireless-sensor-networks/category/6-user-manuals.html>.

MENEZES, A.; ZUCCHERATO, R.; WU, Y. *An elementary introduction to hyperelliptic curves*. [S.l.]: Faculty of Mathematics, University of Waterloo, 1996.

MEULENAER, G. et al. On the Energy Cost of Communication and Cryptography in Wireless Sensor Networks. In: *2008 IEEE International Conference on Wireless and Mobile Computing, Networking and Communications*. Ieee, 2008. p. 580–585. ISBN 9780769533933. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4654302>.

MILLER, V. Use of elliptic curves in cryptography. In: SPRINGER. *Advances in Cryptology - CRYPTO85 Proceedings*. [S.l.], 1986. p. 417–426.

MONTGOMERY, P. Speeding the Pollard and Elliptic Curve Methods of Factorization. *Mathematics of Computation*, v. 48, p. 243–264, 1987.

MORENO, E. D.; PEREIRA, F. D.; CHIARAMONTE, R. B. *Criptografia em Software e Hardware*. 1º. ed. [S.l.: s.n.], 2005. ISBN:85-7522-069-1.

MUNIR, S. et al. Mobile wireless sensor network: Architecture and enabling technologies for ubiquitous computing. In: IEEE. *Advanced Information Networking and Applications Workshops, 2007, AINAW'07. 21st International Conference on*. [S.l.], 2007. v. 2, p. 113–120.

NAGY, C. *Embedded Systems Design Using the TI MSP430 Series*. 1º. ed. new York, USA: Elsevier, 2003. 282 p. ISBN 075067623X.

NEWSOME, J. et al. The sybil attack in sensor networks: Analysis and defenses. In: ACM PRESS. *3rd International Symposium on Information Processing in Sensor Networks*. [S.l.], 2004. p. 259–268.

NIELSEN, M.; CHUANG, I. *Quantum computation and quantum information*. [S.l.]: Cambridge university press, 2010.

OLIVEIRA, E. C. R. de; ALBUQUERQUE, C. V. N. de. Avaliação de protocolos de roteamento para redes ad hoc e rssf aplicados à tv digital interativa e cidades digitais. In: *XXXIII Conferencia Latinoamericana de Informática (CLEI?2007)*. [S.l.: s.n.], 2007.

OLIVEIRA, L. F. de. Criptografia com o uso de curvas elipticas. ITA - Instituto Tecnológico de Aeronáutica. Testes de Software. Abril 2010. Disponível em: <https://sites.google.com/site/lucienefo/disciplinas/ce-281/criptografia-comcurvas-elipticas>.

OTHMAN, S. B.; TRAD, A.; YOUSSEF, H. Performance evaluation of encryption algorithm for wireless sensor networks. In: IEEE. *Information Technology and e-Services (ICITeS), 2012 International Conference on*. [S.l.], 2012. p. 1–8.

PEREIRA, S. R. *O sistema criptográfico de chave pública RSA*. Dissertação (Mestrado) — Universidade Católica de Santos, 2008.

PERRIG, A. et al. Spins: Security protocols for sensor networks. *Wireless networks*, Kluwer Academic Publishers, v. 8, n. 5, p. 521–534, 2002.

PESOVIC, U. et al. Benchmarking performance and energy efficiency of microprocessors for wireless sensor network applications. In: IEEE. *MIPRO, 2012 Proceedings of the 35th International Convention*. [S.l.], 2012. p. 743–747.

PIGATTO, D.; SILVA, N.; BRANCO, K. Performance evaluation and comparison of algorithms for elliptic curve cryptography with el-gamal based on miracl and relic libraries. *Journal of Applied Computing Research*, v. 1, n. 2, p. 95–103, 2011.

PIGATTO, D. F. *Segurança em sistemas embarcados críticos - utilização de criptografia para comunicação segura*. Dissertação (Mestrado) — USP, 2012.

PIOTROWSKI, K.; LANGENDOERFER, P.; PETER, S. How public key cryptography influences wireless sensor node lifetime. In: *fourth ACM workshop on Security of ad hoc and sensor networks*. [S.l.: s.n.], 2006. p. 169–176.

POLLEY, J. et al. Atemu: A fine-grained sensor network simulator. In: IEEE. *Sensor and Ad Hoc Communications and Networks, 2004. IEEE SECON 2004. 2004 First Annual IEEE Communications Society Conference on*. [S.l.], 2004. p. 145–152.

POURPEIGHAMBAR, S. B.; SABAEI, M. Data aggregation of moving object with hybrid clustering in wireless sensor networks. In: IEEE. *Recent Advances in Computing and Software Systems (RACSS), 2012 International Conference on*. [S.l.], 2012. p. 158–163.

REFATTI, L. F.; PAZETO, T. A. Análise e impacto das fontes para redes de sensores para o corpo humano. *Revista Brasileira de Inovação Tecnológica em Saúde*, v. 4, n. 4, p. 13–22, 2011.

REN, Y. et al. Security in Mobile Wireless Sensor Networks - A Survey. *Journal of Communications*, v. 6, n. 2, p. 128–142, abr. 2011. ISSN 1796-2021. Disponível em: <http://ojs.academypublisher.com/index.php/jcm/article/view/4179>.

RIVEST, R.; SHAMIR, A.; ADLEMAN, L. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, ACM, v. 21, n. 2, p. 120–126, 1978.

SCOTT, M. Miracl–multiprecision integer and rational arithmetic c/c++ library. *Shamus Software Ltd, Dublin, Ireland, URL< http://www. shamus. ie*, 2003.

SEN, J. A Survey on Wireless Sensor Network Security. *International Journal of Communication Networks and Information Security (IJCNIS)*, v. 1, n. 2, p. 55–78, 2009.

SEVERINO, A. *Metodologia do trabalho científico*. [S.l.]: Cortez São Paulo, 2007.

SHARMA, H. et al. Optimisation of energy efficiency in wireless sensor networks using error control codes. In: IEEE. *Engineering and Systems (SCES), 2012 Students Conference on*. [S.l.], 2012. p. 1–6.

SHI, E.; PERRIG, A. Designing secure sensor networks. *Wireless Communication Magazine*, v. 11, n. 6, p. 38–43, 2004.

SIMPLICIO, M. *Algoritmos criptográficos para redes de sensores*. Dissertação (Mestrado) — Escola Politécnica at the University of São Paulo, Abril 2008. Disponível em: <http://www.teses.usp.br/teses/disponiveis/3/3141/ tde-30092008-182545/>.

STALLINGS, W. *Network and internetwork security: principles and practice*. [S.l.]: Prentice-Hall, Inc., 1995.

STAPLETON, J. Information security management handbook. In: ____. 6. ed. USA: CRC Press, 2012. cap. Elliptic Curve Cryptosystems.

STRUIK, R. Cryptography for highly constrained networks. In: *NIST - CETA Workshop 2011*. [s.n.], 2011. Disponível em: <http://csrc.nist.gov/groups/ST/CETA_2011/presentations/struik.pdf>.

TAN, M.; MASAGCA, K. A. An investigation of bluetooth security threats. In: IEEE. *International Conference on Information Science and Applications (ICISA)*. [S.l.], 2011. p. 1–7.

TANEMBAUM, A. S. Redes de computadores. In: ____. 4ª edição. ed. Rio de Janeiro: Elsevier, 2003.

TITZER, B.; LEE, D.; PALSBERG, J. Avrora: scalable sensor network simulation with precise timing. *IPSN 2005. Fourth International Symposium on Information Processing in Sensor Networks, 2005.*, Ieee, p. 477–482, 2005. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1440978>.

TORRES, I. S. Elliptical curve cryptography - segurança e privacidade em sistemas de armazenamento e transporte de dados. MSDPA, Univ. do Minho. junho 2007.

UTO, N.; REIS, D. Um survey sobre bibliotecas criptográficas com suporte à criptografia de curvas elípticas. *Cad. CPqD Tecnologia*, v. 1, n. 1, p. 133–141, 2005.

WANDER, A. et al. Energy analysis of public-key cryptography for wireless sensor networks. In: *Third IEEE International Conference on Pervasive Computing and Communications*. [S.l.: s.n.], 2005. p. 324–328.

WANG, H.; SHENG, B.; LI, Q. Elliptic curve cryptography- based access control in sensor networks. *Int. J. Security and Networks*, v. 1, n. 3/4, p. 127–137, 2006.

WANG, X. et al. *Search-based physical attacks in sensor networks: Modeling and defense*. [S.l.], 2005.

WANG, Y.; ATTEBURY, G.; RAMAMURTHY, B. A survey of security issues in wireless sensor networks. *IEEE Communications Surveys and Tutorials*, v. 8, n. 2, p. 2–23, 2006.

WANG, Z. et al. Asip-based design and implementation of rsa for embedded systems. In: IEEE. *High Performance Computing and Communication & 2012 IEEE 9th International Conference on Embedded Software and Systems (HPCC-ICESS), 2012 IEEE 14th International Conference on*. [S.l.], 2012. p. 1375–1382.

WAZLAWICK, R. S. *Metodologia de Pesquisa para Ciência da Computação*. [S.l.]: Elsiever Rio de Janeiro, 2008.

WIKIPEDIA. *StrongARM*. 2013. `http://en.wikipedia.org/wiki/StrongARM`.

WOLLINGER, T. et al. Elliptic and hyperelliptic curves on embedded μP. *ACM Transactions on Embedded Computing Systems*, v. 3, n. 3, p. 509–533, ago. 2004. ISSN 15399087. Disponível em: <http://portal.acm.org/citation.cfm?doid=1015047.1015051>.

WOOD, A.; STANKOVIC, J. Denial of service in sensor networks. *IEEE Computer*, v. 35, n. 10, p. 54–62, 2002.

XIAO, Y. et al. Evaluate reliability of wireless sensor networks with obdd. In: IEEE. *Communications, 2009. ICC'09. IEEE International Conference on*. [S.l.], 2009. p. 1–5.

YAN, H.; SHI, Z. J. Studying software implementations of elliptic curve cryptography. In: IEEE. *Information Technology: New Generations, 2006. ITNG 2006. Third International Conference on*. [S.l.], 2006. p. 78–83.

YANG, J.; HUA, M. A popularizing movable environmental detection and regulating system based on smart home technology. In: IEEE. *Software Engineering and Service Science (ICSESS), 2012 IEEE 3rd International Conference on*. [S.l.], 2012. p. 165–168.

YAVUZ, A.; NING, P. Hash-based sequential aggregate and forward secure signature for unattended wireless sensor networks. In: *6th Int. Conf. on Mobile and Ubiquitous Systems: Computing, Networking and Services - Toronto, Canada*. [S.l.]: MobiQuitous, 2009. p. 1–10.