DANIEL NOGUEIRA DE OLIVEIRA COSTA

ADAPTAÇÃO DE MECANISMOS DE SEGURANÇA PARA COMUNICAÇÃO EM AMBIENTES MÓVEIS

Belo Horizonte
31 de março de 2007

Universidade Federal de Minas Gerais
Instituto de Ciências Exatas
Programa de Pós-Graduação em Ciência da Computação

ADAPTAÇÃO DE MECANISMOS DE SEGURANÇA PARA COMUNICAÇÃO EM AMBIENTES MÓVEIS

Dissertação apresentada ao Curso de Pós-Graduação em Ciência da Computação da Universidade Federal de Minas Gerais como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

DANIEL NOGUEIRA DE OLIVEIRA COSTA

Belo Horizonte
31 de março de 2007

UNIVERSIDADE FEDERAL DE MINAS GERAIS

FOLHA DE APROVAÇÃO

Adaptação de Mecanismos de Segurança para Comunicação em Ambientes Móveis

DANIEL NOGUEIRA DE OLIVEIRA COSTA

Dissertação defendida e aprovada pela banca examinadora constituída por:

Prof. Antonio Alfredo Ferreira Loureiro – Orientador
Universidade Federal de Minas Gerais

Profa. Fátima de Lima Procópio D. Figueiredo
PUC/MG - Pontifícia Universidade Católica de Minas Gerais

Prof. Carlos Maurício Seródio de Figueiredo
FUCAPI - Fundação Centro de Análise, Pesquisa e Inovação tecnológica

Belo Horizonte, 31 de março de 2007

Resumo

Os recentes avanços tecnológicos possibilitaram um grande crescimento da utilização de dispositivos móveis nos últimos anos. Conseqüentemente, a demanda por aplicações seguras nesse ambiente também cresceu. Segurança em redes tradicionais (cabeadas) já é um tema de reconhecida complexidade. O ambiente móvel, entretanto, introduz ainda mais complicações, como o meio de comunicação aberto que facilita a interceptação dos dados transmitidos, e as restrições dos dispositivos móveis, que geralmente possuem menos recursos computacionais, o que impede a utilização de algoritmos mais sofisticados de segurança.

Este trabalho propõe uma adaptação para os métodos de segurança em ambientes móveis, na qual os métodos de segurança são escolhidos dinamicamente de acordo com o valor semântico associado aos dados transmitidos. O objetivo é realizar transmissões utilizando métodos de segurança consistentes com o valor do dado transmitido. Dessa forma, é possível utilizar menos recursos computacionais nas transmissões que não necessitam de segurança, a fim de utilizá-los nas transmissões que realmente precisam de segurança.

A solução proposta isola completamente a camada de aplicação dos detalhes de implementação dos algoritmos de segurança. Ela só precisa informar os requisitos de segurança e qualidade de serviço associados a cada transmissão, ficando a cargo da camada de segurança a responsabilidade de escolher e aplicar o método de segurança mais adequado a cada transmissão. Os resultados experimentais mostraram que é possível obter ganhos significativos de performance, principalmente quando parte dos dados é transmitida sem utilizar nenhum método de segurança. Os experimentos também mostraram que a camada de segurança é capaz de se adaptar bem aos requisitos estabelecidos para cada transmissão.

Abstract

Recent technological advances have enabled a great growth in mobile devices utilization on the last few years. Consequently, there has also been an increasing demand for secure applications in this environment. Security in standard wired networks is already a theme of known complexity. The mobile environment, however, introduces further complications, such as an open communication medium, that makes data interception easier, and the restrictions imposed by mobile devices, which usually have fewer computational resources, and therefore can have restrictions while using certain sophisticate security algorithms.

This work proposes an adaptation for security methods on mobile environments; in witch the security methods are chosen dynamically according to the semantic worth of transmitted data. The goal is to realize transmissions using security methods consistent with the worth of sent data. This way, it's possible to use less computational resources in communications that don't need special security configurations and use them on communications that really need to.

The proposed solution departs the applications layer from security algorithms implementations details. This layer only need to set the security and QoS requirements associated with each transmission, being responsibility of the security layer to choose and apply the most appropriate security method for each transmission. Experimental results showed it's possible to achieve significant performance gains, mainly when part of data is sent without security algorithms. Besides that, experiments showed that security layer is able to adapt properly to requirements established to each transmission.

A Marlene, Pedro Paulo, Calebe e Renata.

Agradecimentos

Agradeço a Deus por tudo que tem feito na minha vida, pelo grande amor, cuidado, e pelas incontáveis e imerecidas bençãos que tem me concedido a cada dia. A Ele toda a honra, glória e louvor, para todo o sempre.

Aos meus pais que foram meus grandes professores e me ensinaram os valores mais importantes da vida. Obrigado pelo grande carinho, apoio e cuidado em todos os momentos. Ao meu irmão Calebe, pela amizade e companheirismo. À minha querida Renata pelo carinho, amor, compreensão, e pelas palavras sábias nos momentos mais difíceis. Não tenho palavras pra agradecer a todos vocês! Muito obrigado!

Agradeço em especial ao Loureiro pelo cuidado, conselhos, e amizade desde os primeiros passos da minha vida acadêmica. Aos demais professores e funcionários do DCC, pelos ensinamentos, todo auxílio e amizade durante esses anos.

Aos colegas de faculdade pela amizade, companheirismo e grande auxílio durante os últimos anos. Aos amigos da ATAN pela amizade, compreensão e companheirismo. Agradeço também aos demais amigos que, até mesmo indiretamente, me ajudaram em toda a minha vida até aqui. Muito obrigado a todos!

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Os recentes avanços tecnológicos têm possibilitado o desenvolvimento de dispositivos computacionais cada vez menores e com maiores recursos computacionais, como celulares, PDA's (do inglês, *Personal Digital Assistants*) e palm tops (computadores de mão), entre outros. Além disso, o desenvolvimento de tecnologias para comunicação sem fio, como Bluetooth [9], Wi-Fi [5] e WiMax [25], dentro outras, possibilitaram o estabelecimento de redes sem fio, permitindo a comunicação entre os mais diversos dispositivos móveis. Essas tecnologias possibilitam a comunicação entre dispositivos móveis em redes de tamanho variável, que vão de redes pessoais de pequeno alcance (1 a 10 metros) até redes de longa distância (que podem cobrir países ou continentes).

Juntamente com o crescimento da utilização dos dispositivos móveis, surge uma demanda pelo desenvolvimento de aplicações mais sofisticadas com maiores garantias de qualidade de serviço e segurança. Aplicações bancárias, por exemplo, necessitam de oferecer garantias de segurança a seus usuários. Aplicações militares possuem requisitos de segurança ainda mais rígidos. Já aplicativos de transmissão de vídeo, podem desejar oferecer determinadas garantias de qualidade de serviço.

O ambiente móvel, entretanto, traz consigo uma série de fatores que complicam o desenvolvimento desse tipo de aplicações. A largura de banda variável somada à mobilidade, por exemplo, adicionam complexidades suficientes para impossibilitar a garantia de qualquer qualidade de serviço. Se um usuário está em uma conversa multimídia através de um dispositivo móvel, e vai para uma área que possui menor capacidade de rede, é necessário renegociar os recursos alocados para a conexão e, ao mesmo tempo, é desejável que o usuário perceba o mínimo de impacto no seu aparelho [6]. Por outro lado, se o usuário se move para um local em que não há infra-estrutura de rede ou nós intermediários capazes de possibilitar o roteamento, é impossível manter a conexão ativa.

Além da qualidade de serviço, oferecer garantias de segurança em ambientes móveis não é uma tarefa fácil. Segurança em redes tradicionais (com fio) já é um tema por si só complexo, profundamente debatido na literatura, e ainda alvo de constantes estudos financiados por grandes empresas e instituições governamentais. Se garantir segurança nas redes tradicionais já é uma tarefa complexa, garantir segurança em redes sem fio é uma tarefa ainda mais difícil.

O meio de transmissão aberto facilita, por exemplo, o ataque passivo de espionagem, no qual uma entidade maliciosa se coloca fisicamente no raio de transmissão e, configurando sua freqüência de escuta, consegue ler todos os dados transmitidos [39].

Além disso, apesar do desenvolvimento de dispositivos com poderes computacionais cada vez maiores, as restrições desses dispositivos ainda representam um problema quando o objetivo é fornecer garantias de segurança. Isso porque fornecer segurança exige tipicamente um gasto considerável de processamento e energia que não pode ser desconsiderado [30, 41]. Diferente das estações de trabalho e dos computadores pessoais modernos, os dispositivos móveis não possuem grandes poderes computacionais de processamento, energia e memória. Por isso, as soluções de segurança de redes tradicionais não são aplicáveis às redes móveis e, assim, existe uma demanda por novos métodos de segurança para esse tipo de ambiente.

Um dos princípios básicos de segurança, denominado *Princípio da Proteção Adequada* [29], diz que um dado deve ser protegido com um grau consistente com seu valor. Ou seja, se um dado é muito importante, ele deve ser transmitido utilizando algoritmos de segurança mais fortes, mas se um dado não tem muito valor para a aplicação, ele deve ser transmitido com uma segurança mais baixa ou, até mesmo, sem segurança.

Nesse contexto, este trabalho propõe uma solução de segurança para redes móveis sem fio, na qual os métodos são escolhidos dinamicamente, a cada transmissão, de acordo com o valor semântico dos dados transmitidos. Ou seja, dados mais importantes são transmitidos com algoritmos mais fortes de segurança, enquanto dados menos importantes são transmitidos com algoritmos mais fracos, ou até mesmo sem nenhuma segurança. O objetivo é economizar os recursos tipicamente escassos dos dispositivos móveis para que eles sejam utilizados quando realmente for necessário.

A arquitetura proposta isola a camada de aplicação dos detalhes de implementação dos algoritmos de segurança, de forma que ela deve especificar apenas os requisitos de segurança e qualidade de serviço associados a cada transmissão, ficando a camada de segurança responsável por escolher o método de segurança mais adequado e aplicá-lo a cada transmissão. Também é proposta uma linguagem específica para a descrição do protocolo de comunicação entre as entidades envolvidas e definição dos requisitos associados a cada transmissão.

Como principais contribuições do trabalho estão:

- Alteração dinâmica dos algoritmos de segurança utilizados a cada transmissão, de acordo com a importância do dado transmitido;
- Definição de uma arquitetura que isola a camada da aplicação dos detalhes de implementação dos algoritmos de segurança;
- Definição de uma linguagem para descrição de protocolos de comunicação e associação de requisitos de segurança e qualidade de serviço a cada transmissão;
- Economia de recursos como: tempo de processamento, *throughput*, energia, latência, entre outros, quando dados são transmitidos com algoritmos de segurança mais fracos.

Figura 1.1: Problema de determinar o protocolo de segurança mais adequado a cada transmissão

1.1 Caracterização do Problema

Este trabalho propõe resolver o problema no qual uma aplicação deseja estabelecer uma comunicação segura com outra entidade em um ambiente móvel, mas os dispositivos envolvidos dispõem de poucos recursos computacionais (como processamento, memória e energia). Esse cenário é comum em redes móveis, onde os dispositivos geralmente não possuem grande capacidade de processamento, memória e energia.

A proposta é utilizar uma camada de segurança que transmita os dados com uma segurança consistente com o valor dos mesmos. Para isso, é proposta uma linguagem especial denominada LACS (Seção 3.2.1) para que a aplicação informe à camada de segurança os requisitos de segurança e qualidade de serviço associados a cada transmissão. A camada de segurança, por sua vez, é responsável por escolher, a cada transmissão, os algoritmos de segurança mais adequados, de acordo com os requisitos de segurança e QoS associados à transmissão.

A cada transmissão é necessário também resolver o problema de determinar o melhor protocolo de segurança. Conforme ilustrado na Figura 1.1, o algoritmo de tomada de decisão recebe como entrada os requisitos de segurança e qualidade de serviço associados à transmissão, e informações sobre o contexto, como mobilidade, interferência, entre outros. Um algoritmo de tomada de decisão é responsável por determinar o protocolo de segurança mais adequado em cada transmissão.

Neste trabalho, abordamos o problema em dois cenários distintos, cada um com um algoritmo de tomada de decisão diferente. A primeira, para redes ad hoc, é uma adaptação do algoritmo utilizado pelo *ASecMid* , desenvolvido pelo aluno de doutorado de TU Eindhoven Bruno P. S. Rocha [32]. Essa abordagem recebe como entrada os requisitos de segurança e qualidades de serviço associados às transmissões, e parâmetros do contexto, como mobilidade, interferência, memória disponível, bateria, etc. A segunda abordagem é um algoritmo de baixa complexidade de tempo ($O(1)$) que, portanto, consome ainda menos recursos de processamento e energia, sendo adequado para redes com maiores restrições, como redes de sensores sem fio. Essa alternativa, entretanto, não leva em consideração os parâmetros do meio, mas apenas os requisitos de segurança e qualidade de serviço associados a cada transmissão. Maiores detalhes sobre as duas soluções propostas podem ser encontrados na Seção 3.3.1 e 3.3.2. Apesar de serem propostas duas abordagens, esse trabalho é aplicável a qualquer tipo de rede sem fio.

Não faz parte do escopo avaliar se os algoritmos de segurança são seguros ou não. As duas

soluções propostas permitem a configuração de quais algoritmos de segurança serão utilizados, sejam eles para redes tradicionais ou redes móveis. O Apêndice B contém uma lista dos algoritmos de segurança utilizados pela camada de segurança nos experimentos realizados.

1.2 Organização do Texto

Este documento está organizado da seguinte maneira: o Capítulo 2 contém uma revisão da Literatura com trabalhos relacionados, o Capítulo 3 detalha a solução proposta, o Capítulo 4 detalha a implementação da Camada de Segurança, os experimentos e resultados são apresentados no Capítulo 5, e o Capítulo 6 apresenta as conclusões e trabalhos futuros.

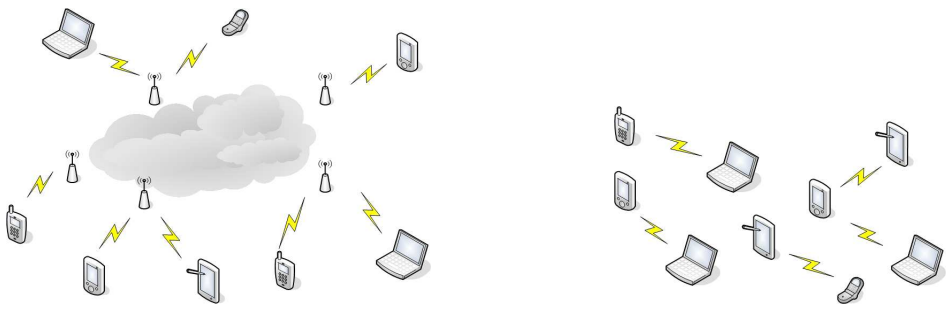
Capítulo 2

Revisão da Literatura

2.1 Ambientes Móveis

As redes móveis sem fio são compostas, tipicamente, por diversos dispositivos de diferentes capacidades computacionais, que comunicam entre si de duas formas possíveis. A primeira forma é através de estações base (Figura 2.1(a)), que fornecem uma infra-estrutura de comunicação que permite aos dispositivos se conectarem a ela e, assim, comunicarem entre si. A estação base geralmente é uma entidade fixa que possui conexão direta com uma rede maior e que, quando recebe uma conexão de um dispositivo móvel, permite o acesso desse dispositivo a essa rede maior. Esse tipo de rede é a mais comum, sendo utilizada, por exemplo, nas redes de telefonia celular.

A segunda forma de comunicação é denominada de redes ad hoc (Figura 2.1(b)). Nessas redes não há nenhuma infra-estrutura responsável pela comunicação entre os dispositivos, sendo a comunicação realizada diretamente entre os dispositivos. Um dos desafios mais comuns desse tipo de rede é o roteamento, pois nem sempre é possível transmitir as informações diretamente do dispositivo de origem para o destino. Outro desafio são as constantes mudanças de topologia que ocorrem devido à característica móvel dos dispositivos envolvidos. Na literatura existem diversos estudos sobre os desafios introduzidos por esse tipo de rede [34, 28, 15, 2, 17].



(a) Redes Infraestruturadas (b) Redes Ad Hoc

Figura 2.1: Classificação das redes sem fio em relação à infra-estrutura de comunicação

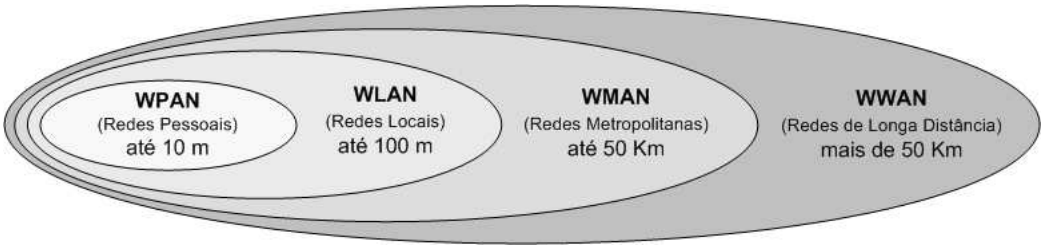


Figura 2.2: Classificação das redes sem fio em relação ao seu alcance

Atualmente existem diversas tecnologias para comunicação sem fio. Escolher a mais adequada depende das características da rede em questão, conforme ilustra a Figura 1.2. O Bluetooth [9], por exemplo, permite a comunicação a curta distância (no máximo 10 metros), sendo utilizado principalmente em periféricos como fones de ouvido, mouses e teclados sem fio, entre outros. Esse tipo de rede é denominado Rede Pessoal sem Fio ou WPAN (do inglês, *Wireless Personal Area Network*). Para esse tipo de rede existem também outros padrões como a o Wireless USB [22] (versão sem fio do tradicional USB), IEEE 802.15.4 [16], entre outros. Redes com um pouco mais de alcance são chamadas de Redes Locais sem Fio, ou WLAN (do inglês, *Wireless Local Area Network*), cujo objetivo é conectar computadores diversos (incluindo dispositivos móveis) em uma rede sem fio de no máximo 100 metros. Esse tipo de rede é mais utilizado em empresas, casas e edifícios em geral, nos quais se deseja formar uma rede local e fornecer acesso à internet a todos os computadores ligados a essa rede. Nesse cenário se destaca o protocolo IEEE 802.11, mais conhecido como Wi-Fi [5]. Existem ainda as Redes Metropolitanas sem Fio (WMAN, do inglês *Wireless Metropolitan Area Network*) que são redes com alcance maior que 100 metros (distância radial de até 50 km), utilizadas para formar redes em grandes áreas urbanas. Nesse cenário se destaca o padrão de comunicação IEEE 802.16, mais conhecido como WiMax [25]. Finalmente, as Redes sem Fio de Longa Distância (WWAN, do inglês *Wireless Wide Area Network*) são redes que cobrem países ou até continentes inteiros, como as redes de celulares.

Além das redes mencionadas, ainda existem outros tipos de redes sem fio, como as redes de sensores sem fio. Essas redes são compostas geralmente por dispositivos computacionais muito pequenos, com grandes restrições de processamento, memória e, principalmente energia [10]. Um cenário típico de uma rede de sensores sem fio é a instalação de uma grande quantidade de sensores em uma área que será monitorada, tipicamente inóspita e de difícil acesso. Conseqüentemente, os sensores são jogados nessas regiões e permanecem monitorando a região até que a sua bateria acabe. Existe um grande esforço na literatura no sentido de desenvolver técnicas que aumentem o tempo de vida dessas redes, ou seja, o tempo que ela permanece ativa antes de parar devido à falta de energia [19, 18, 4]. Fornecer segurança em um ambiente tão restrito é um desafio ainda maior.

A abordagem de segurança proposta neste trabalho independe do tipo da rede. Redes com restrições maiores, como é o caso de redes de sensores, podem utilizar métodos mais leves para determinar o melhor protocolo de segurança a cada transmissão. A Seção 3.3.2

detalha uma solução proposta para o ambiente de redes de sensores sem fio que é ótima na complexidade de tempo. A Seção 3.3.1 trata da solução proposta para ambientes menos restritos e com dispositivos com capacidades computacionais variáveis, como redes ad hoc. Nesse caso, a camada de segurança utiliza um algoritmo mais sofisticado para determinar o melhor protocolo de segurança a cada transmissão.

2.2 Segurança

Em computação, segurança pode ser tratada de diversas formas, através de diversas abordagens e técnicas. A segurança pode ser tratada a nível de software, no que diz respeito ao desenvolvimento de técnicas para detectar a existência de vírus, detecção de brechas, ou prevenção e remoção de arquivos infectados. A nível de sistema operacional, a segurança pode ser tratada no sentido de garantir controle de acesso aos dados da memória, ao próprio computador, entre outras.

Neste trabalho, a segurança será tratada do ponto de vista das redes de computadores, ou seja, do ponto de vista de garantir que os dados serão transmitidos com segurança, entre os dispositivos, sem que entidades não autorizadas tenham acesso aos dados transmitidos, e sem que os dados sejam perdidos ou modificados.

[27] fornece uma taxonomia para classificação dos tipos de ataques de segurança em redes WLANs, na qual os ataques são divididos em dois grupos: passivos ou ativos, conforme ilustrado na Figura 2.3. Os ataques passivos podem ser de dois tipos: Espionagem [1], no qual o nó malicioso intercepta o conteúdo transmitido entre dois nós da rede; ou Análise de Tráfego [2], no qual o nó malicioso ganha inteligência ao monitorar as transmissões entre dois nós da rede, analisando os padrões de comunicação. Já os ataques ativos, podem ser de quatro tipos: Mascaragem [3], no qual o nó malicioso se passa por um usuário autorizado, obtendo privilégios que não possui originalmente; Reenvio [4], no qual o nó malicioso monitora as transmissões e responde mensagens como se fosse um usuário legítimo; Modificação de Mensagem [5], no qual o nó malicioso altera uma mensagem transmitida por um usuário legítimo; ou Negação de Serviço [6], no qual o nó malicioso impede a utilização normal dos recursos de comunicação de um usuário legítimo.

Um mecanismo de segurança deve ter um objetivo definido, podendo ser para: detecção de ataques ou falhas de segurança; prevenção, no sentido de evitar a ocorrência de falhas; ou ação, no sentido de diminuir ou anular os efeitos do ataque, quando ele ocorre. O mecanismo de segurança proposto neste trabalho tem o objetivo principal de prevenir ataques.

Nos últimos anos, o assunto segurança tem recebido uma atenção especial, tanto da indústria quanto da academia. Entretanto, [14] destaca que, apesar do crescimento do interesse

[1] do inglês, *eavesdropping*
[2] do inglês, *traffic analysis*
[3] do inglês, *masquerading*
[4] do inglês, *replay*
[5] do inglês, *message modification*
[6] do inglês, *denial-of-service*

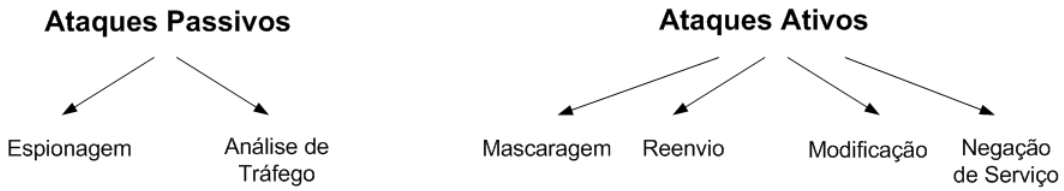


Figura 2.3: Tipos de Ataques

pela segurança, ainda existem poucos investimentos em segurança. Segundo ele, a segurança está relacionada principalmente com a informação, que pode estar disponível de diferentes formas: escrita em papel, eletrônica armazenada em algum dispositivo, em transmissões entre dispositivos, ou falada em conversas. O autor também menciona que a informação é um bem de alto valor na era em que vivemos e, portanto, é necessário protegê-la muito bem.

Muito esforço tem sido realizado no sentido de desenvolver padrões internacionais para questões de segurança. A ISO 27001 [11], por exemplo, é um padrão para gerência da segurança da informação. Publicado pela *International Organization for Standardization* (ISO) e *International Electrotechnical Commision* (IEC), é baseado no Modelo PDCA [7] de gestão de processos, e define um sistema padrão de Gestão de Segurança da Informação, determinando políticas, responsabilidades, práticas, procedimentos, entre outras normas. Ele define também o conceito de Segurança da Informação como a preservação de: confidencialidade (assegurar que a informação não seja acessível àqueles que não deveriam ter acesso), integridade (proteger a exatidão e a integridade da informação e dos métodos de processo) e disponibilidade (assegurar que usuários autorizados tenham acesso à informação e aos recursos associados quando requeridos). Além dessas, também pode ser necessário garantir outras propriedades, tais como: autenticação, responsabilidade, não repúdio e confiabilidade.

O Instituto Nacional de Padrões e Tecnologias dos Estados Unidos (NIST), em publicação especial, divulgou um documento (*NIST Special Publication 800-48*) com recomendações de segurança para o estabelecimento de redes sem fio [27]. O documento aborda detalhes sobre os padrões de comunicação IEEE 802.11 e Bluetooth, além de discutir benefícios e riscos de segurança decorrentes da utilização de dispositivos móveis. O documento destaca que em redes móveis é muito mais complexo oferecer garantias de segurança, pois todas as vulnerabilidades presentes em redes tradicionais (com fio) são aplicáveis, além de uma série de outras vulnerabilidades, como: informações que são transmitidas através da rede sem fio sem criptografia (ou com criptografia fraca) podem ser facilmente interceptadas devido ao meio de transmissão aberto; ataques do tipo D.o.S. (*Denial of Service*) podem ser dirigidos facilmente a dispositivos ou conexões sem fio; entidades maliciosas podem se passar por entidades legítimas (autorizadas) obtendo acesso e realizando ações para dano da rede ou simples coleta de informação; entidades maliciosas podem violar a privacidade de entidades autorizadas rastreando suas ações; e dispositivos móveis podem ser roubados de seu usuário verdadeiro, permitindo ao ladrão acesso a dados confidenciais, como o episódio ocorrido na Petrobrás em

fevereiro de 2008 [7] no qual laptops com informações sigilosas sobre pesquisas foram roubados.

2.3 Segurança em Redes sem Fio

Devido à complexidade inerente ao ambiente sem fio, existem na literatura diversas soluções para as questões de segurança. O padrão de comunicação sem fio IEEE 802.11, por exemplo, define um protocolo de segurança WEP, do inglês, *Wired Equivalent Privacy*. O protocolo WEP utiliza uma chave secreta que é compartilhada entre o dispositivo móvel e um ponto de acesso em uma estação base. A chave secreta é utilizada para criptografar os pacotes antes de serem transmitidos utilizando o protocolo RC4, e uma verificação de integridade (CRC-32) é utilizada para garantir que o pacote não foi modificado durante a transmissão. Para evitar que dois textos sejam criptografados com a mesma chave, um vetor de inicialização é utilizado para aumentar a chave compartilhada e produzir uma chave de RC4 diferente para cada pacote [26].

Embora o protocolo WEP tenha sido proposto como um protocolo padrão para redes sem fio, diversas publicações demonstraram a vulnerabilidade desse protocolo a certos tipos de ataques [26, 3, 1, 40, 33]. Uma deficiência, por exemplo, é o espaço limitado no vetor de inicialização para permutações do algoritmo de criptografia RC4 utilizado pelo protocolo. Em uma rede com tráfico pesado, um dispositivo malicioso através de um ataque passivo de espionagem, pode interceptar e ler pacotes até encontrar duas mensagens com o mesmo vetor de inicialização. O resultado de um XOR entre os dois pacotes pode ser utilizado para inferir sobre o conteúdo das duas mensagens. A redundância causada pelo protocolo IP também pode ser utilizada nesse propósito, assim como outros métodos, sendo que, em alguns casos, é possível obter o conteúdo exato da mensagem.

Diante das vulnerabilidades desse protocolo, surgem na literatura diversas soluções alternativas para as questões de segurança em redes sem fio. [26], por exemplo, propõe um mecanismo de segurança que utiliza uma técnica de cifra de fluxo que exige, para ser quebrado, um algoritmo de força bruta de complexidade temporal $\Omega(2^n)$. [24], por sua vez, propõe um método de detecção de intrusão multi-camada integrado, no qual mecanismos de detecção de intrusão são adicionados em todas as camadas de rede. Quando um nó detecta uma intrusão que afeta toda a rede, ele inicia um processo de re-autenticação para excluir o nó malicioso da rede. Quando uma intrusão local é detectada por uma camada superior, ela notifica às camadas inferiores. O autor também destaca que nenhuma rede aberta é imune a intrusões, e redes ad-hoc sem fio são, particularmente, vulneráveis devido ao meio de comunicação aberto, às mudanças dinâmicas de topologia e aos algoritmos cooperativos utilizados nesse ambiente. Também são examinadas as vulnerabilidades de uma rede ad-hoc sem fio, a necessidade de métodos de detecção de intrusão e os motivos pelos quais os métodos de segurança para redes tradicionais (com fio) não se aplicam diretamente no ambiente móvel.

Existem também soluções que exploram a capacidade variável dos dispositivos móveis em redes sem fio. [23] propõe um mecanismo de segurança adaptativo para redes *ad hoc* multi-

[7] http://www1.folha.uol.com.br/folha/dinheiro/ult91u372319.shtml (acessado em 06/03/2008)

níveis. O trabalho descreve uma rede composta de três níveis que variam de acordo com a capacidade dos nós em termos dos recursos disponíveis. No primeiro nível, encontram-se os nós regulares, em maior quantidade, cuja capacidade de energia, processamento e armazenamento são mais limitados. O segundo nível é composto por backbones móveis dotados de mais recursos computacionais. No terceiro nível estão localizados backbones com ainda maior capacidade computacional. A rede é organizada de forma hierárquica, onde os nós do primeiro nível são agrupados em *clusters* cujo nó cabeça é o backbone do segundo nível. Da mesma forma os nós do segundo nível estão agrupados. A rede proposta é capaz de operar em dois níveis diferentes dependendo da infra-estrutura disponível na rede. Quando os nós do terceiro nível estão disponíveis, a rede opera no modo *com infra-estrutura*, em que a certificação é realizada de forma centralizada nos nós do terceiro nível. Quando não há nós do terceiro nível, a rede opera no modo *sem infra-estrutura*, em que a certificação é realizada de forma distribuída na rede.

Alguns autores destacam ainda as restrições impostas pelo ambiente móvel. [30] destaca que a adição de mecanismos de segurança aumenta a necessidade de processamento devido aos algoritmos de criptografia e reduz do tempo de vida da bateria dos dispositivos. Como alternativa, os autores discutem abordagens que incluem algoritmos de criptografia de baixa complexidade, melhorias de segurança em processadores embutidos e arquiteturas avançadas para dispositivos sem fio disponíveis devido a novas metodologias de projetos a nível de sistema.

Devido às restrições dos ambientes móveis, alguns autores propõem a diminuição do nível de segurança em alguns casos. [36], por exemplo, descreve um esquema de proteção para transmissão de vídeos MPEG-4 para dispositivos móveis que transmite os quadros do vídeo MPEG de acordo com o seu tipo, criptografando apenas os quadros que carregam maior informação sobre o conteúdo do vídeo. Ele estuda a distorção causada pela criptografia de apenas alguns quadros do vídeo, para comparar e inferir quais tipos de quadro realmente precisam ser criptografados. Esse trabalho é semelhante ao proposto neste documento, mas aquele se limita às transmissões de vídeos MPEG, enquanto este trata de um mecanismo genérico, que funciona independente do tipo de dado transmitido e permite a configuração de diversos níveis de segurança para diferentes trechos do dado.

Nesse mesmo sentido, [37] propõe um método de segurança para redes de sensores sem fio, que classifica os dados transmitidos na rede e fornece níveis de segurança diferentes para cada tipo, a fim de economizar energia. Os dados que trafegam na rede são classificados em: código móvel, localização dos nós sensores e dado específico da aplicação, sendo que a segurança aplicada no primeiro tipo de dado é mais forte que a aplicada ao segundo, cuja segurança é maior que a aplicada ao terceiro. A diferença entre este e o trabalho de Slijepcevic é que o último trata exclusivamente de redes de sensores sem fio e limita o número de níveis de segurança em três níveis pré-definidos que são aplicados sempre sobre os mesmos tipos de dados. O primeiro, entretanto, pode ser aplicado em vários tipos de redes, e permite a utilização de níveis variados de segurança, que podem ser aplicados a quaisquer transmissões realizadas pela aplicação. Transmissões realizadas pela camada de roteamento não são

tratadas. A Seção 3.3.2 aborda mais detalhes sobre as adaptações da solução proposta para o ambiente de redes de sensores sem fio.

[32], em sua dissertação de mestrado, propõe um middleware adaptativo de segurança denominado *ASecMid*, que observa as condições do meio sem fio, das capacidades dos hardwares envolvidos, da utilização dos recursos e configurações da aplicação, a fim de realizar transmissões com segurança variável, de acordo com esses parâmetros. Ou seja, se o meio oferece boas condições de transmissão, será utilizado um método forte de segurança. Caso contrário, será escolhido o método de segurança mais adequado para as condições do ambiente no momento. O *ASecMid* utiliza seis parâmetros para caracterizar os protocolos de segurança: confidencialidade, integridade, autenticação, processamento, rede e memória. A cada transmissão, os protocolos são ranqueados através de uma formulação matemática que envolve os parâmetros do contexto, e o melhor protocolo é escolhido para a transmissão.

Este trabalho, por sua vez, propõe uma abordagem dinâmica para os parâmetros de segurança, na qual o nível de segurança de cada transmissão é determinado pelo valor semântico do dado transmitido. O *ASecMid* que, originalmente, considera os parâmetros de segurança definidos como parâmetros estáticos (que não variam a cada transmissão realizada pela camada de aplicação), foi modificado para que os parâmetros de segurança sejam variáveis a cada transmissão, de acordo com o protocolo definido pela aplicação. A intenção é utilizar menos recursos do dispositivo móvel escolhendo um método de segurança mais leve quando o dado não é muito importante, para permitir a escolha de níveis mais pesados quando realmente for necessário. Esse método é inspirado no *Princípio da Proteção Adequada* [29] que diz que a segurança deve ser fornecida para dados enquanto eles possuem algum valor e que, dados com valores diferentes, devem ser protegidos com níveis de segurança proporcionais ao seu valor.

Capítulo 3

Solução Proposta

Este trabalho propõe uma camada de segurança para ambientes móveis que permite à aplicação realizar transmissões com níveis de segurança que variam dinamicamente de acordo com a semântica dos dados transmitidos. Para isso, a aplicação utiliza uma linguagem denominada LACS (Linguagem de Anotação para Configuração de Segurança), através da qual define um padrão de comunicação e os níveis de segurança associados a cada transmissão. A definição dessa linguagem faz parte deste trabalho e pode ser vista em detalhes na Seção 3.2.1. A linguagem permite à aplicação abstrair os detalhes de implementação dos algoritmos de segurança, de forma que necessita apenas informar requisitos de segurança associados a cada transmissão.

A seguir, a Seção 3.1 detalha a arquitetura do trabalho proposto, e as Seções 3.2 e 3.3 detalham os dois módulos principais que compõem a solução de segurança proposta.

3.1 Arquitetura

No final da década de 70 e início da década de 80, o modelo OSI (*Open Systems Interconnection*) foi desenvolvido pela ISO (*International Organization for Standardization*) com o objetivo de criar um modelo de referência para interconexão de sistemas devido à diversidades de arquiteturas de comunicação que existiam na época. O modelo OSI define 7 camadas organizadas hierarquicamente, de forma que cada camada é responsável por prover um serviço para a camada superior utilizando os serviços da camada inferior [8].

O modelo TCP/IP tornou-se mais difundido que o modelo OSI devido ao crescimento da Internet, que utiliza essa arquitetura. Embora não siga o modelo ISO, ele é semelhante no sentido que define também uma pilha de camadas que comunicam entre si de forma interdependente. O modelo TCP/IP define 5 camadas : camada de aplicação, camada de transporte, internet, interface de rede e camada física.

A arquitetura proposta neste trabalho pode ser adaptada a qualquer um dos modelos mencionados, assim como a outros. A Figura 3.1 ilustra a arquitetura do trabalho proposto. Apenas a camada de aplicação possui informações suficientes sobre a semântica dos dados que serão transmitidos para poder determinar o nível de segurança de cada transmissão. Por isso,

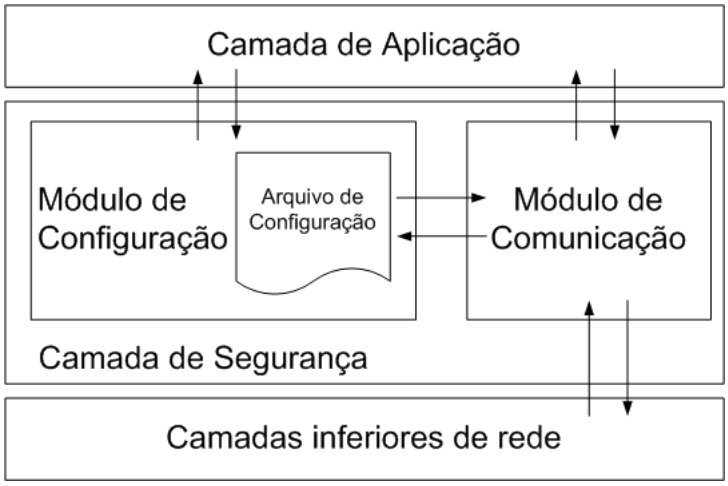


Figura 3.1: Arquitetura da camada de segurança proposta

a camada de aplicação deve passar essas informações para a camada de segurança através de um arquivo de configuração. A camada de segurança, por sua vez, é responsável por interpretar o arquivo de configuração e determinar o método de segurança mais adequado para cada transmissão.

Como a camada de segurança está localizada acima da camada de rede, ela abstrai os detalhes de roteamento dos pacotes transmitidos. Além disso, devido a limitações da linguagem LACS, a transmissão é realizada sempre entre duas entidades. Ou seja, este trabalho não trata de comunicações em *broadcast* ou *multicast*.

Conforme pode ser visto na Figura 3.1, a Camada de Segurança é dividida em dois módulos que comunicam entre si o tempo todo: o Módulo de Configuração e o Módulo de Comunicação. O Módulo de Configuração é responsável por interpretar o arquivo de configuração e informar, para cada transmissão, o nível de segurança e qualidade de serviço determinado pela aplicação. O Módulo de Comunicação, por sua vez, seleciona o protocolo de segurança mais adequado de acordo com os requisitos informados, e aplica-o na transmissão dos dados.

Caso não seja possível descrever o protocolo de comunicação utilizando a linguagem LACS, a camada de aplicação pode se comunicar diretamente com o módulo de configuração para configurar os requisitos associados a cada transmissão através da API da camada de segurança. Maiores detalhes sobre essa alternativa são tratados na Seção 3.2.2.

3.2 Módulo de Configuração de Segurança

Na camada de segurança, o Módulo de Configuração é responsável por interpretar o arquivo de configuração em LACS para informar ao Módulo de Comunicação, a cada transmissão, o nível de segurança definido pela aplicação. O nível de segurança de cada transmissão é definido através do que foi chamado de Configuração de Segurança. Uma Configuração de Segurança pode ser definida formalmente da seguinte forma:

Definição 1. *Uma **Configuração de Segurança** é um conjunto de requisitos que determinam o nível de segurança e qualidade de serviço de uma transmissão.*

De acordo com os requisitos de segurança definidos em uma Configuração de Segurança, a camada de segurança escolhe qual método de segurança mais adequado para cada transmissão. Neste trabalho, foram considerados os seguintes requisitos de segurança e qualidade de serviço:

Nível de Confidencialidade Determina o quanto o dado a ser transmitido é confidencial e, portanto, não pode ser acessado por alguém que não possua acesso ao conteúdo. Esse requisito previne principalmente ataques passivos de escuta e análise de tráfego. Os seguintes níveis de confidencialidade são considerados (ordenados do mais fraco para o mais forte): desabilitado, opcional, desejável, obrigatório [1] ou crítico.

Nível de Integridade Determina a importância de se garantir que o dado deve chegar ao destinatário exatamente como foi transmitido (sem alterações, maliciosas ou não). Esse requisito previne principalmente ataques de modificação, e pode assumir os mesmos valores de níveis definidos no requisito anterior.

Nível de Autenticação Define o quanto é importante garantir que o dado foi realmente transmitido pelo outro par, e não por alguma entidade maliciosa. Esse requisito previne principalmente ataques de reenvio e mascaragem, podendo também assumir os mesmos valores de níveis definidos nos requisitos anteriores.

Tempo Máximo de Latência Define um limite de tempo de uma transmissão. Em algumas aplicações a latência pode ser algo extremamente importante. Quando este requisito está presente na configuração de uma transmissão, o módulo de comunicação elimina os protocolos de segurança cujo tempo de processamento é maior que a latência máxima.

Taxa de Transmissão Mínima Em algumas aplicações pode ser necessário garantir um nível mínimo de taxa de transmissão de pacote.

Penalidade Máxima de Bytes por Pacote Alguns mecanismos de segurança adicionam alguns bytes em cada pacote. Esse requisito define um limite máximo para o *overhead* de bytes adicionados em cada transmissão.

Além das propriedades específicas de cada requisito (nível de segurança, tempo máximo de latência, taxa mínima de transmissão), todos os requisitos possuem também uma outra propriedade denominada prioridade.

Definição 2. *A **prioridade** de um requisito define a sua importância em relação aos demais requisitos, podendo assumir os valores: Baixa, Média ou Alta.*

Essa propriedade é utilizada pelo módulo de configuração quando existem dois requisitos conflitantes. Por exemplo, se uma configuração define um nível de autencidade alta e um

[1] do inglês, *mandatory*

valor de latência máxima muito baixo, esses requisitos claramente são conflitantes, no sentido de que pode ser impossível atender aos dois requisitos. A prioridade permite à módulo de comunicação definir qual requisito será atendido prioritariamente. Nesse exemplo, se a prioridade da latência for menor que a do nível de autenticidade, o módulo de comunicação deve utilizar um algoritmo mais forte de criptografia, mesmo que isso implique em não atender o requisito de latência. Caso todos os requisitos tenham a mesma prioridade, a camada de segurança seleciona um deles arbitrariamente.

3.2.1 LACS - Linguagem de Anotação para Configuração de Segurança

Conforme mencionado anteriormente, apenas a camada de aplicação possui informações suficientes sobre a semântica do dado para determinar se ele deve ser transmitido com muita ou pouca segurança. Para que essa informação chegue à camada de segurança, a camada de aplicação utiliza uma linguagem denominada LACS (Linguagem de Anotação para Configuração de Segurança). Nesta seção iremos definir essa linguagem formalmente.

O principal objetivo da linguagem LACS é permitir a definição de dois elementos básicos: o protocolo de comunicação entre duas entidades, e o nível de segurança associado a cada mensagem do protocolo. O protocolo é definido como uma seqüência de trocas de mensagens (envios e recebimentos), com condições e laços de repetição, semelhante a uma linguagem procedural. Os níveis de segurança são definidos através de Configurações de Segurança que devem ser declaradas e associadas às mensagens do protocolo.

Formalmente, um protocolo escrito em LACS pode ser definido da seguinte forma:

Definição 3. *Em LACS, um* ***protocolo*** P *é um par* $P = (D,T)$*, no qual* D *contém as declarações que contém as definições de configuração de segurança e* T *contém o modelo de transmissão, isto é, o protocolo de comunicação que descreve a comunicação entre um par de entidades.*

A Figura 3.2 mostra o esquema de um protocolo descrito em LACS. As palavras-chave da linguagem são escritas em inglês e serão detalhadas a seguir. Cada arquivo de configuração em LACS descreve um único protocolo de comunicação entre duas entidades. A descrição inicia com a palavra-chave **protocol-description** e encerra com a palavra-chave **end**. As próximas seções detalham cada parte da descrição de um protocolo em LACS. A primeira aborda as declarações em LACS, e a segunda o modelo de transmissão, que contém a definição do protocolo de comunicação entre as entidades.

3.2.1.1 Declarações

As declarações em LACS são definidas entre as palavras-chave **declaration** e **end declaration**, e podem ser divididas em dois grupos: definições de segurança e definições de mensagens, delimitados pelas palavras-chave **security-definition** e **end security-definition**, e **message-definition** e **end message-definition**, respectivamente. O primeiro contém a

```
protocol-description

    declaration

        security-definition
        // Configurações de Segurança
        end security-definition

        message-definition
        // Declaração de algumas mensagens do protocolo
        end message-definition

    end declaration

    transmission-model
    // Descrição do protocolo de comunicação
    end transmission-model

end protocol-description
```

Figura 3.2: Descrição de um protocolo em LACS

definição das Configurações de Segurança associadas às transmissões, enquanto o segundo (opcional), contém a definição de mensagens que serão transmitidas durante a comunicação.

As definições de segurança devem ser informadas entre as palavras-chave **security-definition** e **end security-definition**, e devem conter as Configurações de Segurança que serão associadas às transmissões. Em LACS, toda transmissão (envio ou recebimento de mensagem) deve estar obrigatoriamente associada a uma única configuração de segurança, que deve ser previamente declarada nas definições de segurança.

Conforme a Definição 1, uma Configuração de Segurança é composta de um conjunto de requisitos de segurança e qualidade de serviço. Além disso, em LACS, uma Configuração de Segurança possui um nome que é utilizado para associar uma mensagem a uma configuração de segurança. Esse nome, portanto, deve ser único para cada configuração. Os requisitos que compõem a configuração são os mesmos definidos na Seção 3.2, a saber: Confidencialidade (**confidentiality**), Integridade (**integrity**), Autenticidade (**authenticity**), Latência Máxima (**maximum-latency**), Taxa Mínima de Transmissão (**minimum-transmission-rate**) e Penalidade Máxima em Bytes por Pacote (**maximum-byte-overhead-per-packet**).

Cada requisito possui alguns parâmetros que devem ser informados para definir o valor do requisito que está sendo definido. Esses parâmetros variam de acordo com o requisito, podendo ser o nível de segurança, o tempo de latência máxima, o valor da taxa mínima de transmissão, etc.

Todos os requisitos têm um parâmetro em comum: a prioridade (**priority**), que pode assumir os valores **high**, **medium** e **low**. Conforme explicado na Definição 2, a prioridade define a importância de um requisito em relação ao outro. Caso não seja informado, o valor médio é atribuído, por padrão. Além desse, cada requisito possui outros parâmetros específicos, conforme detalhado a seguir:

Requisitos de Segurança Todos os requisitos de segurança (Confidencialidade, Autenti-

```
configuration ("segurança-média")

    authenticity (level=desired, priority=medium),
    confidentiality (level=Mandatory, priority=high),
    integrity (level=Optional, priority=low),
    maximum-latency (time=30000)

end configuration
```

Figura 3.3: Exemplo de Configuração em LACS

cação e Integridade) possuem o parâmetro obrigatório **level** que indica o nível de segurança associado ao requisito. Ele pode assumir os seguintes valores: **disabled**, **optional**, **desired**, **mandatory** ou **critical**.

Latência Máxima O parâmetro **time** é obrigatório e indica o tempo máximo de latência (em microssegundos) para a transmissão correspondente.

Taxa de Transmissão Mínima O parâmetro **rate** é obrigatório e indica a taxa mínima de transmissão. Além disso, o parâmetro **time-unit** também é obrigatório e indica a unidade da taxa informada, podendo assumir os seguintes valores: **byte-seconds**, **byte-microseconds** ou **byte-milliseconds**.

Penalidade Máxima em Bytes por Pacote O parâmetro **overhead** é obrigatório e indica a penalidade em bytes por pacote.

O exemplo da Figura 3.3 mostra a definição de uma Configuração de Segurança em LACS. O nome da configuração é *segurança-média* e os requisitos que a compõem são: Autenticidade, Confidencialidade, Integridade e Latência Máxima, cujos parâmetros são indicados entre parênteses separados por vírgula.

Além de Configurações de Segurança, também deve ser informado uma configuração de segurança especial, denominada Configuração Global.

Definição 4. ***Configuração Global*** *é uma configuração especial que define requisitos que serão válidos para todas as configurações de segurança.*

Ou seja, todas as configurações de segurança herdam os requisitos da Configuração Global. Entretanto, é possível redefinir algum requisito caso deseje alterar um requisito padrão. Não é possível associar uma transmissão do protocolo à configuração global. Ela somente serve de base para as demais configurações, sendo útil quando existem muitas configurações de segurança e é necessário definir um requisito comum a todas as configurações.

A Figura 3.4 mostra um exemplo de Configuração Global que define que todas as transmissões, por padrão, terão uma latência máxima de 70000 microssegundos e um nível de integridade obrigatório com prioridade alta. O exemplo também mostra uma Configuração denominada *segurança-baixa* que não define nenhum requisito e, portanto, é composta apenas pelos requisitos da Configuração Global. Se for definida no mesmo arquivo de configuração,

```
global-configuration()

    integrity (level=mandatory, priority=high),
    maximum-latency (time=70000)

end global-configuration

configuration("segurança-baixa")
end configuration
```

Figura 3.4: Exemplo de Configuração Global em LACS

a configuração *segurança-média* (Figura 3.3) redefinirá os níveis de integridade e latência máxima pré-estabelecidos na configuração global.

A linguagem LACS também permite a definição de várias configurações em uma única transmissão, através da Configuração de Segurança Composta.

Definição 5. *Uma **Configuração de Segurança Composta** é um conjunto de valores (F, C) em que F é o tamanho da parte do dado que será transmitido com a configuração de segurança C.*

A configuração composta é útil, por exemplo, quando se deseja transmitir um vídeo com níveis de segurança diferentes em cada trecho do vídeo. Ao invés da camada de aplicação particionar o vídeo em várias partes e enviá-las separadamente, basta enviar o vídeo inteiro e utilizar uma configuração de segurança composta. Nesse caso, a camada de segurança é responsável por particionar o dado e transmitir cada parte com a segurança correspondente.

A Figura 3.5 mostra um exemplo de Configuração Composta (**ComposedConfiguration**) denominada *tipo-composto*. Esse tipo de configuração é composto por faixas (**ranges**) que possuem um tamanho definido (**size**) e uma configuração de segurança associada (**configuration**). O tamanho deve ser informado em bytes ou porcentagem de acordo com o parâmetro informado na declaração da configuração (**byte** ou **percent**, respectivamente). A cláusula **Repeat** permite a definição de seqüências de faixas que se repetem por um número determinado de vezes, informado na declaração da cláusula entre parênteses. Se a unidade do tamanho for porcentagem, é importante que a soma dos tamanhos (considerando as repetições) seja exatamente 100%.

Para descrever o protocolo de comunicação entre as entidades, pode ser interessante realizar comparações para verificar se uma mensagem de um determinado tipo foi recebida ou enviada. Essa informação pode ser utilizada para definir fluxos de comunicação diferentes dependendo das mensagens que estão sendo trocadas entre as entidades. Para isso, a linguagem LACS permite a declaração de mensagens.

Definição 6. *Em LACS, uma **mensagem** m pode ser definida por um par $m = (n, cc)$ no qual n é o nome da mensagem e cc é um conjunto de comparadores que definem a mensagem.*

Nem todas as mensagens precisam ser declaradas. Para determinar se uma mensagem recebida ou enviada é uma mensagem previamente declarada, é necessário aplicar todos os

```
composed-configuration("tipo-composto", percent)

    repeat(2)

        range (configuration="segurança-baixa", size=10),
        range (configuration="segurança-média", size=10)

    end repeat

    range (configuration="segurança-baixa", size=60)

end composed-configuration
```

Figura 3.5: Exemplo de Configuração Composta em LACS

comparadores na mensagem e verificar se todos retornaram resultado positivo. Um comparador pode ser definido da seguinte forma:

Definição 7. *Em LACS, um* ***comparador*** *é um elemento que permite a comparação entre uma mensagem que trafega na rede e um tipo de mensagem pré-definido.*

A linguagem LACS define dois tipos de comparadores:

Comparador de Tamanho compara o tamanho em bytes de uma mensagem com um valor (**value**) determinado, utilizando um operador (**operator**) previamente definido. O operador pode assumir os seguintes valores: igual (equal), maior que (greater-than), menor que (less-than), maior ou igual (greater-or-equal), menor ou igual (less-or-equal), ou diferente (different).

Comparador de Máscara compara o conteúdo da mensagem com um determinado valor (**match-value**), aplicando uma máscara (**match-mask**) determinada. Nesse comparador, é necessário informar o primeiro byte (**first-byte**) a partir do qual a comparação será realizada, assim como o tamanho da máscara em bytes (**mask-size**).

O exemplo da Figura 3.6 ilustra a definição de uma mensagem em LACS de 5 bytes denominada *especial*, cujos 2^o e 3^o bytes possuem o valor 0103 em hexadecimal. O exemplo também inclui a declaração de uma mensagem de *fim de loop* cujo tamanho é menor que 3 bytes.

A Figura 3.7 ilustra o funcionamento do comparador de máscara. O atributo `first-byte=2` indica que a comparação ocorrerá a partir do segundo byte. O atributo `mask-size` indica o tamanho da máscara informada no atributo `match-mask`. A camada de segurança realiza uma operação de *and* binário do valor da máscara com o conteúdo da mensagem, e o resultado é comparado com o valor informado no atributo `match-value`. A máscara permite, portanto, a verificação de apenas alguns bits do conteúdo da mensagem. No exemplo, os valores não são iguais e, portanto, o comparador retorna o valor falso.

A Figura 3.8 mostra um exemplo completo de uma seção de declarações em LACS com os exemplos utilizados até aqui.

```
message-definition

    message ("especial")
        mask-comparator (first-byte=2, mask-size=2,
                         match-mask=0xFFFF, match-value=0x0103),
        size-comparator (operator=igual, value=5)
    end message

    message ("fim-loop")
        size-comparator (operator=less-than, value=3)
    end message

end message-definition
```

Figura 3.6: Exemplo de Definição de Mensagens em LACS

```
mask-comparator (first-byte=2, mask-size=2,
      match-mask=0x0F0F, match-value=0x0103)
```

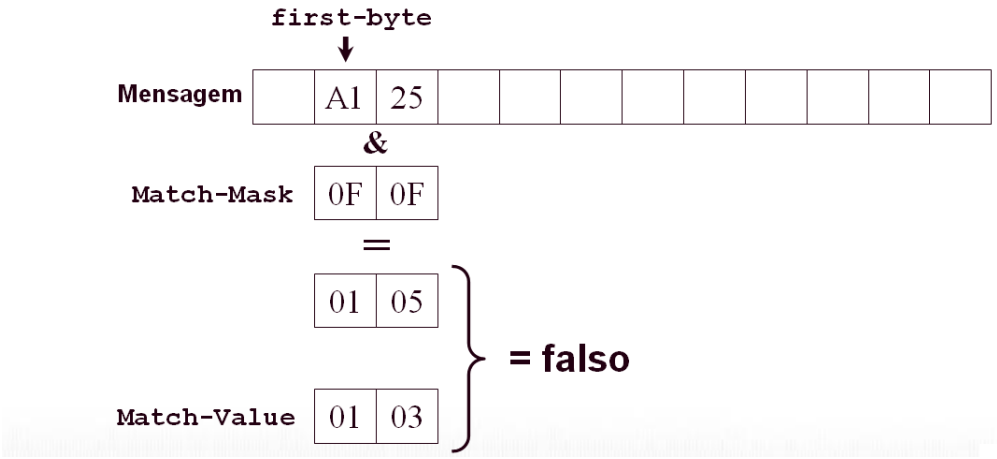


Figura 3.7: Exemplo de comparação através do comparador de máscara.

3.2.1.2 Modelo de Transmissão

O modelo de transmissão é composto por uma seqüência de comandos que descrevem trocas de mensagens realizada pelas entidades comunicantes. Formalmente podemos definir um modelo de transmissão em LACS da seguinte forma:

Definição 8. *Um **Modelo de Transmissão** em LACS é um conjunto de comandos que descrevem o protocolo de comunicação, ou seja, a seqüência de trocas de mensagens entre as duas entidades envolvidas.*

Os comandos podem ser tanto de primitivas de comunicação (*send* e *receive*) quanto de controle de fluxo de execução, isto é, comandos condicionais (*if*) ou de execução de laços (*loop*). Assim como os requisitos, os comandos também possuem atributos associados que variam de acordo com o comando.

```
declaration
  security-definition

    global-configuration()
      integrity (level=mandatory, priority=high),
      maximum-latency (time=70000)
    end global-configuration

    configuration ("segurança-baixa")
    end configuration

    configuration ("segurança-média")
      authenticity (level=desired, priority=medium),
      confidentiality (level=mandatory, priority=high),
      integrity (level=optional, priority=low),
      maximum-latency (time=30000)
    end configuration

    composed-configuration("tipo-composto", percent)

      repeat(2)
        range (configuration="segurança-baixa", size=10),
        range (configuration="segurança-média", size=10)
      end repeat

    range (configuration="segurança-baixa", size=60)
    end composed-configuration

  end security-definition

  message-definition

    message ("especial")
      mask-comparator (first-byte=2, mask-size=2,
        match-mask=0xFFFF, match-value=0x0103),
      size-comparator (operator=igual, value=5)
    end message

    message ("fim-loop")
      size-comparator (operator=less-than, value=3)
    end message

  end message-definition
end declaration
```

Figura 3.8: Exemplo de Declarações em LACS

Um modelo de transmissão em LACS pode conter os seguintes comandos:

Send Determina o envio de uma mensagem para a outra entidade. O comando suporta os seguintes atributos:

- Configuration obrigatório, indica o nome da Configuração de Segurança associada à transmissão;

- Label opcional, atribui um nome à mensagem enviada através desse comando;

- Size opcional, determina o tamanho (em bytes) da mensagem que será enviada, lançando exceção caso a mensagem não seja do tamanho especificado;

Receive Determina o recebimento de uma mensagem da outra entidade. Possui os mesmos atributos do comando acima.

Send-Receive Determina que uma mensagem será recebida ou enviada. Esse comando permite a definição de protocolos mais livres nos quais, em algum momento, não se sabe exatamente se haverá envio ou recebimento de mensagens, importando apenas que será utilizada uma determinada configuração de segurança. Os atributos são os mesmos dos comandos Send e Receive.

If Comando utilizado para determinar fluxos de execução diferentes de acordo com uma condição pré-determinada. É composta pelos seguintes elementos:

- Condições determinam as condições que determinam qual bloco de comandos deve ser executado (then ou else);

- Then bloco de comandos que é executado caso a condição seja verdadeira;

- Else bloco de comandos que é executado caso a condição seja falsa;

Loop Comando utilizado para estabelecer a repetição de um bloco de comandos. É possível determinar a condição de parada do loop de duas formas (que podem ser utilizadas em conjunto):

- Count adicionando esse atributo ao comando, determinando o número máximo de iterações do loop;

- Stop-condition adicionado antes do primeiro comando do bloco, contém um conjunto de condições que devem ser todas verdadeiras para que a condição de parada seja satisfeita.

Tanto a condição do **if** quanto o comando **stop-condition** contém um conjunto de condições que devem ser verificadas para determinar o fluxo de execução. Em LACS, o valor final de um conjunto de condições é a conjunção do valor das condições do conjunto, ou seja, um conjunto de condições é verdadeiro se e somente se todas as condições contidas no conjunto são satisfeitas. A linguagem LACS define três tipos de condições possíveis:

Compare-message Compara uma mensagem enviada ou recebida anteriormente, com uma mensagem previamente declarada na seção de definição de mensagens. O atributo **name** determina o nome da mensagem previamente declarada, enquanto o atributo **label** determina o nome da mensagem que foi transmitida ou recebida em algum momento.

Send-message Essa condição é verdadeira se e somente se o próximo evento for o envio de uma mensagem. Opcionalmente é possível verificar também se a mensagem enviada é de um tipo previamente declarado. Nesse caso, basta preencher o atributo **name** com o nome da mensagem declarada.

Receive-message Semelhante à condição anterior, mas verifica o recebimento de uma mensagem. O atributo **name** também pode ser utilizado para comparar a mensagem recebida com uma mensagem declarada previamente.

```
transmission-model(repeat=true)

  send (configuration="tipo-composto");
  receive (configuration="seguranca-baixa", size="4096");
  send (configuration="seguranca-media");
  receive (configuration="seguranca-baixa", size="4096");

  if
    send-message (name="mensagem-especial")
  then
    send (configuration="seguranca-media");
    receive (configuration="seguranca-baixa");
  else

    loop (count="10")
      stop-condition
        compare-message (label="msg-recebida", name="fim-loop")
      end stop-condition

      send (configuration="seguranca-baixa");
      receive (configuration="seguranca-baixa",
                  label="msg-recebida");
    end loop

  end if

end transmission-model
```

Figura 3.9: Exemplo de Modelo de Transmissão em LACS

O Modelo de Transmissão deve ser informado uma única vez por uma das entidades comunicantes. A outra entidade recebe o modelo e o interpreta, apenas trocando os comandos *send* por *receive* e vice-versa.

A seqüência de comandos do modelo repete por um número indefinido de vezes, por padrão. Contudo, é possível definir que os comandos não irão se repetir através do atributo `repeat` da palavra-chave `transmission-model`. Nesse caso, se houver alguma comunicação utilizando o módulo de comunicação após o fim do protocolo descrito, é lançada uma exceção.

A Tabela 3.9 mostra um exemplo de modelo de transmissão. O protocolo inicia com o envio de uma mensagem utilizando a configuração composta (tipo-composto), e o recebimento de uma mensagem de 4KB. Em seguida ocorre o envio de uma mensagem de segurança média e, novamente, uma mensagem de 4KB é recebida.

Na próxima comunicação, o módulo de configuração verifica se uma mensagem do tipo "mensagem-especial"está sendo enviada. Caso afirmativo, ela é enviada com uma segurança média e a entidade recebe uma mensagem com segurança baixa. Caso contrário, as entidades iniciam uma troca de mensagens de segurança-baixa que se repete no máximo 10 vezes. Se, durante essa troca de mensagens, a entidade receber uma mensagem de fim-loop, o loop é interrompido e a execução é reiniciada a partir do início.

3.2.2 Modelo de Transmissão para comunicações mais complexas

A linguagem LACS foi projetada para permitir a descrição de diversos protocolos de comunicação, sendo que os diversos comandos de transmissão, repetição ou condição servem para ampliar seu de descrição de protocolos. Por isso, ela é composta por comandos semelhantes a uma linguagem de programação, incluindo comandos próprios para comunicação entre entidades e verificação das mensagens trafegadas.

Embora a linguagem seja bastante flexível, uma limitação dessa abordagem é a necessidade de definir, a priori, a seqüência de transmissões (envios e recebimentos) que irá realizar. Pode haver casos em que a seqüência de transmissões é completamente imprevisível, ou até mesmo tão complexa que a linguagem LACS não consiga descrever.

Para esses casos, a camada de segurança permite uma configuração através da própria API do módulo de configuração. Para isso, a aplicação deve, primeiramente, definir a Configuração Global e cada Configuração de Segurança que irá utilizar em suas transmissões. Essa configuração é realizada através da classe *ApplicationSecurity* e, ao criar uma Configuração de Segurança, o módulo de configuração retorna uma entidade do tipo *ConfigurationHandler*, que faz referência à Configuração de Segurança criada.

Em seguida, basta que, ao realizar uma transmissão utilizando a camada de segurança, a aplicação passe como parâmetro o *ConfigurationHandler* relacionado à configuração desejada para aquela transmissão. Nesse caso, a camada de segurança realiza a transmissão considerando os requisitos definidos na Configuração de Segurança correspondente.

3.3 Módulo de Comunicação

O Módulo de Comunicação é o módulo responsável por aplicar efetivamente os métodos de segurança e transmitir conforme a configuração estabelecida no Arquivo de Configuração. O módulo contém uma lista de métodos de segurança e, a cada transmissão, escolhe um método de segurança mais adequado de acordo com a Configuração de Segurança estabelecido para aquela transmissão.

A interação entre o Módulo de Comunicação e o Módulo de Configuração é constante, de forma que o primeiro sempre informa ao segundo a ocorrência de eventos na rede de transmissão ou recebimento de mensagens pela aplicação. Dessa forma, o Módulo de Configuração consegue acompanhar a seqüência de mensagens através do Modelo de Transmissão (Seção 3.2.1.2) e, assim, determinar qual a Configuração de Segurança associada à transmissão corrente. Ou seja, a cada transmissão (eventos *send* ou *receive*) realizados pela aplicação), o módulo de comunicação informa a ocorrência desse evento ao módulo de configuração, que, por sua vez, retorna a Configuração de Segurança associada à transmissão.

Para escolher o método de segurança mais adequado para ser aplicado à transmissão, o Módulo de Comunicação deve possuir um algoritmo de escolha cuja responsabilidade é selecionar o método mais adequado para cada transmissão. Esse método pode ser mais sofisticado ou menos sofisticado, dependendo da rede em questão e da capacidade dos dispositivos envolvidos. Por exemplo, se o contexto é uma rede ad hoc sem fio com dispositivos que possuem

mais recursos computacionais, esse método pode ser mais complexo, e levar em conta até mesmo outros fatores além da configuração de segurança determinada no arquivo de configuração, como por exemplo, parâmetros do meio e características dos dispositivos envolvidos na comunicação. Entretanto, se o contexto for uma rede de sensores sem fio composta por dispositivos que possuem recursos muito mais restritos, esse algoritmo pode ser mais simples, de forma que esse processo de tomada de decisão não cause nenhum impacto significativo nos recursos de processamento, energia e memória.

Para ilustrar bem a aplicabilidade da solução proposta em diferentes cenários, neste trabalho dois cenários distintos serão detalhados. O primeiro envolve um ambiente de redes ad hoc sem fio, que são redes sem infra-estrutura cujos dispositivos possuem características diversas, podendo ser desde notebooks com muita capacidade de processamento, até celulares de pouca memória e capacidade computacional. O segundo cenário é uma rede de sensores sem fio, cujos dispositivos possuem uma restrição muito grande de processamento, memória, e energia. Cada um dos cenários será detalhado nas próximas seções.

3.3.1 Redes Ad Hoc Sem Fio

O ambiente de redes ad hoc sem fio é composto por dispositivos com capacidades muito distintas, que vão de celulares (com pouca capacidade computacional) a notebooks (com processadores de múltiplos núcleos). Para aproveitar essa heterogeneidade da rede, o ideal é que o método de segurança a ser utilizado nas transmissões leve em consideração a capacidade computacional de cada dispositivo. Assim, é possível utilizar algoritmos mais fortes de segurança nas comunicações entre dispositivos com maior capacidade, e algoritmos mais fracos quando dispositivos mais limitados estão envolvidos. Além disso, as redes móveis estão suscetíveis a diversas variações externas, como interferências, mobilidade, variação da largura de banda, etc. Uma solução de segurança mais sofisticada deve levar em consideração essas variáveis do ambiente em que está inserido.

Neste trabalho, o middleware de segurança *ASecMid* foi escolhido exatamente por possuir a característica de se adaptar ao contexto da rede. Desenvolvido por [32], o *ASecMid* é um middleware de segurança adaptativo que utiliza mecanismos de segurança diferentes dependendo da capacidade dos dispositivos envolvidos e das variáveis do meio como interferências, mobilidade, etc.

O *ASecMid* foi utilizado como base para a implementação do módulo de comunicação da camada de segurança proposta neste trabalho. Originalmente, esse middleware considera os parâmetros de segurança como uma configuração estática, realizada uma única vez no início da execução pelo administrador. Ou seja, a aplicação apenas definia o nível de segurança de suas transmissões durante a instalação, e esse nível era considerado para todas as suas transmissões. Para adaptá-lo a este trabalho, o *ASecMid* foi modificado e integrado ao módulo de configuração para considerar os parâmetros de configuração de segurança como dinâmicos. Ou seja, a cada transmissão, o middleware solicita ao Módulo de Configuração a configuração de segurança (que contém os requisitos de segurança), e considera esses requisitos durante a escolha do protocolo de segurança mais adequado para a transmissão.

3.3.1.1 Conceitos Básicos

Antes de aprofundar nos detalhes sobre o módulo de comunicação, é preciso entender a diferença entre um algoritmo de segurança e um protocolo de segurança. Embora esses conceitos possam ser interpretados da mesma forma em alguns contextos, a partir deste momento faremos distinção entre os termos, conforme as definições em [32].

Definição 9. *Um **algoritmo de segurança** é um processo computacional que representa uma operação atômica de segurança sobre um bloco de dado, com um único objetivo criptográfico. Por exemplo, codificar ou decodificar um dado, gerar um código hash, ou trocar uma ou mais chaves criptográficas.*

Definição 10. *Um **protocolo de segurança** é um conjunto de um ou mais algoritmos de segurança que são aplicados entre blocos de dados em comum acordo entre as entidades comunicantes, que, juntas, podem fornecer um ou mais serviços de segurança.*

Além dos termos acima, é preciso entender também a diferença entre parâmetros e métricas. O primeiro termo trata de uma informação do contexto ao qual o middleware está inserido, como informações sobre mobilidade, nível da bateria, nível de interferência do canal, etc. As métricas são medidas através das quais os algoritmos de segurança são comparados: nível de confidencialidade, integridade, autenticidade, e utilização de recursos de memória, processamento e rede.

Para o middleware, cada algoritmo está associado a seis métricas: três relacionadas a segurança (confidencialidade, integridade e autenticidade), e três relacionados à utilização de recursos (memória, tempo de processamento e overhead de rede). As métricas de segurança são representadas por números inteiros de 0 a 100 e devem ser determinadas pelo administrador do sistema durante a instalação do middleware para cada algoritmo de segurança utilizado pelo middleware. Por exemplo, o algoritmo AES [38] deve possuir um valor mais alto de confidencialidade do que o algoritmo DES [31], por ser conceitualmente mais forte e seguro. As demais métricas são armazenadas em tabelas com valores diferentes para entradas de tamanhos diferentes. Maiores detalhes serão abordados na próxima seção.

Além de alterar os valores das métricas de segurança para cada algoritmo, o administrador do sistema também pode configurar o *ASecMid* para não utilizar algum algoritmo que não seja de sua confiança.

Outro conceito importante é o de *Força Criptográfica*.

Definição 11. ***Força Criptográfica** representa a segurança de um algoritmo ou protocolo de segurança, e é calculado como a soma dos três valores das métricas de segurança: confidencialidade, integridade e autenticação.*

Quanto maior a Força Criptográfica, mais seguro é o algoritmo ou protocolo de segurança. Por definição, a Força Criptográfica depende das métricas de segurança e seu valor varia de 0 a 300, de acordo com os valores estabelecidos pelo administrador do sistema para cada métrica.

3.3.1.2 Escolha do Protocolo de Segurança

O processo completo de escolha do protocolo de segurança será detalhado, a seguir. Para facilitar o entendimento, o algoritmo será dividido em quatro etapas: instalação, inicialização, conexão e transmissão.

Primeira etapa - Instalação

Essa etapa ocorre apenas uma vez e é a fase computacionalmente mais pesada. Nela, cada algoritmo de segurança é testado com combinações diferentes de dados e tamanhos de chaves, a fim de calcular a quantidade média de utilização da memória, tempo de processamento e *overhead* de byte por pacote de cada algoritmo. Os resultados são armazenados em um arquivo para que esse processo não precise ser repetido. As métricas de utilização são armazenadas em um arquivo na forma de tabelas com os valores médios para cada tamanho da entrada utilizado.

Segunda etapa - Inicialização

Nessa fase, o arquivo de configuração gerado na etapa anterior é lido e o middleware gera um conjunto com todos protocolos de segurança possíveis através da combinação dos algoritmos de segurança suportados pela aplicação. Conforme mencionado anteriormente, neste trabalho consideramos um protocolo de segurança como um conjunto de algoritmos de segurança. Um cuidado especial é tomado nessa fase para não gerar protocolos que contém algoritmos com características redundantes. As métricas associadas aos algoritmos de segurança são agregadas para possibilitar a comparação entre os protocolos. Para as métricas de segurança (confidencialidade, integridade, autenticidade) são consideradas as maiores métricas de cada algoritmo, e as métricas de utilização de recursos (memória, processamento e rede) de cada algoritmo são somadas.

Terceita etapa - Conexão

Nessa fase, as entidades envolvidas na comunicação trocam informações entre si para determinar os protocolos e as chaves que serão utilizadas durante as transmissões. Para isso, os seguintes passos são executados:

1. As entidades trocam os parâmetros de configuração fixa (que não variam a cada transmissão) entre si, sendo escolhidos os valores mais restritivos de cada configuração;
2. As entidades selecionam um conjunto de protocolos que serão utilizados durante as transmissões;
3. As entidades selecionam algoritmos para troca de chaves;
4. As entidades geram e trocam chaves criptográficas.

No segundo passo, o objetivo do middleware é reduzir o número de protocolos de segurança disponíveis, a fim de diminuir o custo de escolher um protocolo em cada transmissão. Segundo [32], o primeiro conjunto de protocolos (gerado durante a instalação) pode chegar a mais de 7.000 protocolos devido ao grande número de combinações dos algoritmos de segurança. Por isso, no segundo passo, para cada configuração de segurança definida no arquivo de configuração, o middleware seleciona um conjunto de protocolos que melhor atende à configuração. Para selecionar cada conjunto, o *ASecMid* executa os seguintes passos:

2.a Primeiramente, são eliminados os protocolos com métricas que não satisfazem as restrições estabelecidas pelas configurações. Por exemplo, são eliminados os protocolos com tempo de processamento maior que o tempo de latência máxima definido na configuração de segurança; os protocolos que oferecem serviço de confidencialidade, no caso da configuração de segurança estabelecer o nível de confidencialidade desabilitado, etc.

2.b Após a eliminação acima, o middleware executa um algoritmo guloso para reduzir ainda mais o conjunto de protocolos possíveis. Para isso são selecionados os seguintes protocolos:

- Os N_1 protocolos com maiores valores de cada métrica de segurança (confidencialidade, integridade e autenticidade);
- Os N_2 protocolos que gastam menos memória;
- Os N_3 protocolos com menor tempo de processamento e menor overhead de rede para pequenas transmissões (10 bytes);
- Os N_4 protocolos com menor tempo de processamento e menor overhead de rede para grandes transmissões (128 KBytes);
- Os N_5 protocolos com maior soma das três métricas de segurança;
- Os N_6 protocolos com menor soma das três métricas de segurança.

O valor das constantes acima são configuráveis pelo administrador do sistema durante a instalação do middleware.

Em casos extremos, é possível que o passo 2.a elimine todos os protocolos disponíveis. Nesse caso, os requisitos menos prioritários (com prioridade baixa ou média) são descartados, no intuito de aumentar a lista de protocolos disponíveis. Se, ainda assim, todos os protocolos forem eliminados, o módulo de comunicação lança uma exceção informando que não foi possível encontrar um protocolo adequado para as configurações estabelecidas.

Os algoritmos de troca de chaves são selecionados de acordo com os protocolos selecionados no passo 2. Os passos acima são executados por apenas uma das entidades, que, no final, envia os algoritmos selecionados para a outra entidade atribuindo números identificadores a cada um deles. Maiores detalhes sobre o algoritmo de troca de chaves e sobre as restrições do item 2.a podem ser obtidos em [32].

Quarta etapa - Transmissão

A última etapa ocorre no momento da transmissão, quando a camada de segurança seleciona o melhor protocolo de segurança (dentro do conjunto pré-selecionado) para realizar a transmissão. Após escolher o melhor protocolo de segurança, o middleware envia seu número identificador e o tamanho de cada segmento concatenado ao pacote, para que o outro lado consiga realizar a operação reversa.

Antes de determinar o melhor protocolo, o *ASecMid* elimina do conjunto pré-selecionado os protocolos que estiverem violando as condições da configuração de segurança e qualidade de serviço estabelecidas para a transmissão corrente. Os métodos que necessitam de mais memória que a disponível, por exemplo, são eliminados, assim como os métodos cujo tempo de processamento for maior que o tempo máximo de latência, e etc.

Para selecionar o melhor protocolo de segurança, o middleware determina uma pontuação X_t para cada protocolo t e escolhe o protocolo com a maior pontuação, de acordo com a seguinte fórmula:

$$X_t = (c \times W_c + i \times W_i + a \times W_a) - (p \times W_p + r \times W_r + m \times W_m) \tag{3.1}$$

onde W_j é o peso da métrica j.

O valor das métricas é linearizado de forma que seus valores estejam entre 0 e 100. As métricas de segurança já possuem, por definição, o valor nessa faixa. O tempo de processamento (p) é calculado como a razão entre o tempo definido na tabela do protocolo (calculada na etapa de inicialização) para o tamanho de dado transmitido, dividido pela diferença entre a latência máxima e a latência de rede no momento. O overhead de rede (r) é calculado como a porcentagem do tamanho da mensagem atual, e a utilização de memória (m) como o menor valor entre a porcentagem da restrição de memória máxima e a porcentagem da memória disponível no sistema.

O peso W_i de cada métrica i é calculado através da seguinte equação:

$$W_i = \sum_j P_j \times K_{P_j,i} \tag{3.2}$$

onde P_j é o valor linearizado do parâmetro j e $K_{P_j,i}$ é o impacto do parâmetro j sobre a métrica i. Cada parâmetro influencia uma ou mais métricas, conforme exibido na Tabela 3.1.

Tanto o valor das constantes K quanto das constantes N, são configuráveis. Como não faz parte do escopo deste trabalho discutir o impacto da variação dessas constantes na escolha do melhor protocolo de segurança, foram utilizados os mesmos valores definidos por [32] em seu trabalho.

3.3.2 Redes de Sensores sem Fio

Embora o *ASecMid* seja muito bom para o ambiente de redes ad hoc sem fio, ele possui um algoritmo de tomada de decisão muito complexo e que, portanto, não é aplicável em redes com grandes restrições de energia e processamento, como é o caso de Redes de Sensores sem

Parâmetro / Métrica	Confidenc.	Integ.	Autent.	Proc.	Rede	Memória
Mobilidade	×		×			
Qualidade do Canal	×	×			×	
Latência				×		
Roteamento	×		×			
Capacidade de Memória						×
Informação do rádio	×	×	×			
Consumo de Memória						×
Uso de CPU				×		
Bateria				×	×	×
Latência Máxima				×		
Memória Máxima						×
Máx. Overhead de Rede					×	
Máx. Overhead em Negociações					×	
Nível de Confidencialidade	×					
Nível de Integridade		×				
Nível de Autenticação			×			

Tabela 3.1: Impacto de cada parâmetro nas métricas de segurança e QoS

Fio. Nesse caso, é necessário utilizar um Módulo de Comunicação mais leve, pois o processo de escolha do melhor método de segurança não pode consumir muita energia e tempo de processamento.

Uma alternativa é utilizar um algoritmo de tomada de decisão com baixa complexidade de tempo, conforme o algoritmo detalhado a seguir:

1. Na fase de configuração, o administrador determina estaticamente o protocolo de segurança que será executado para cada combinação dos valores dos requisitos da configuração de segurança. Os protocolos são carregados em uma matriz em que, cada dimensão da matriz é um requisito e os índices são mapeados nos valores possíveis de cada requisito.

2. A cada transmissão, o módulo de comunicação obtém a configuração de segurança associada à transmissão e obtém o protocolo de segurança que deve ser utilizado acessando a posição da matriz de acordo com o valor dos requisitos na configuração de segurança da transmissão correspondente.

O algoritmo acima é o mais rápido possível ($O(1)$) por que o primeiro acesso à matriz já retorna o método de segurança mais adequado. Entretanto, se forem considerados todos os valores possíveis para os requisitos, a complexidade espacial do algoritmo é exponencial e, portanto, inviável. Se todos os requisitos tivessem o mesmo número de valores possíveis (m, por exemplo), a complexidade espacial do algoritmo seria $O(m^n)$, onde n é o número de requisitos.

Embora a complexidade espacial seja exponencial, é possível simplificar os valores possíveis dos requisitos de forma a reduzir o espaço utilizado pela matriz. A configuração de segurança,

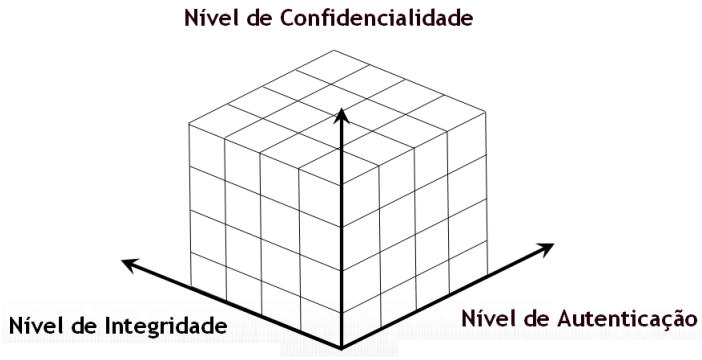


Figura 3.10: Exemplo da matriz do algoritmo para redes de sensores sem fio.

por exemplo, possui 6 requisitos de segurança e QoS. Se cada requisito possuir apenas dois valores possíveis (habilitado/desabilitado), a matriz irá possuir apenas 36 elementos. Também é possível reduzir a complexidade desprezando alguns dos 6 requisitos definidos anteriormente. A Figura 3.10 ilustra o exemplo de uma matriz contendo apenas três dimensões com os três requisitos de segurança. Outra opção é aumentar para três os valores possíveis no caso das métricas de segurança: nível alto, médio e baixo. Nesse caso o tamanho da matriz passa para $3^3 \times 2^3 = 216$ elementos.

Muitos outros algoritmos e métodos de segurança poderiam ser utilizados. O importante é que haja um algoritmo de tomada de decisão simples que, a partir das informações de uma Configuração de Segurança, selecione e aplique um método de segurança.

Capítulo 4

Implementação

O modelo de segurança proposto neste trabalho para redes ad hoc sem fio (descrito na Seção 3.3.1) foi implementado, para fins de análise do seu comportamento e comparação com outros métodos de segurança estáticos existentes. A camada de segurança foi toda implementada em C++, orientada a objetos.

O Módulo de Comunicação, conforme mencionado na Seção 3.3.1, foi implementado como uma adaptação do middleware de segurança *ASecMid*. Para isso, utilizamos a implementação de [32], que foi alterada para que os requisitos de segurança fossem obtidos do módulo de configuração e, a cada transmissão, considerados na escolha do método de segurança mais adequado. Os detalhes sobre o algoritmo de tomada de decisão que escolhe o protocolo de segurança mais adequado estão descritos na Seção 3.3.1.2.

A próxima seção explica a definição da linguagem LACS em XML, e a seção seguinte os detalhes sobre a implementação do Módulo de Configuração.

4.1 Definição da linguagem LACS em XML

A linguagem LACS foi definida em XML (*eXtensible Markup Language*). Um DTD (*Document Type Definition*) foi construído para definir formalmente a gramática da linguagem em função das tags XML. O DTD completo está disponível no Apêndice A. Nesta seção serão abordados alguns detalhes da linguagem em XML.

A tag **protocol-description** é a tag base de um protocolo descrito em LACS. Conforme mencionado na Definição 3, um protocolo em LACS é composto de duas partes: as declarações e o modelo de transmissão. Em XML, a tag **protocol-description** é composta por duas tags: a tag **declaration** que contém as declarações e a **transmission-model** que contém os comandos que compõem o modelo de transmissão do protocolo.

As declarações, por sua vez, são divididas em dois tipos: definições de segurança (**security-definition**) e definições de mensagens (**message-definition**). As definições de segurança incluem a definição obrigatória de uma configuração global (**global-configuration**), que será base para as demais, e a declaração de uma ou mais configurações de segurança (**configuration**).

Tanto a configuração global quanto as configurações de segurança, são compostas por

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE protocol-description SYSTEM "lacs.dtd">
<protocol-description>
  <declaration>
    <security-definition>
      <global-configuration>
        <requirements>
          <integrity level="Mandatory" priority="High" />
          <maximum-latency time="70000" />
        </requirements>
      </global-configuration>
      <configuration name="seguranca-baixa" />
      <configuration name="seguranca-media">
        <requirements>
          <authenticity    level="Desired" priority="Medium" />
          <confidentiality level="Mandatory" priority="High" />
          <integrity level="Optional" priority="Low" />
          <maximum-latency time="30000"/>
        </requirements>
      </configuration>
      <configuration name="tipo-composto" type="Composed" unit="Percent">
        <ranges>
          <repeat count="2">
            <range configuration="seguranca-baixa"  size="10" />
            <range configuration="seguranca-media" size="10" />
          </repeat>
          <range configuration="seguranca-baixa"  size="60" />
        </ranges>
      </configuration>
    </security-definition>
    <message-definition>
      <message name="especial">
        <mask-comparator first-byte="2" mask-size="2" match-mask="0xFFFF" match-value="0x0103" />
        <size-comparator operator="equal" value="5" />
      </message>
      <message name="fim-loop">
        <size-comparator operator="less-than" value="3" />
      </message>
    </message-definition>
  </declaration>
  <transmission-model ...
</protocol-description>
```

Figura 4.1: Exemplo em XML das Declarações em LACS

requisitos declarados dentro da tag **requirements**. Os requisitos são os mesmos definidos na Seção 3.2, cujas tags são: **authenticity**, **confidentiality**, **integrity**, **maximum-latency**, **minimum-transmission-rate** e **maximum-byte-overhead-per-packet**. Os atributos que esses requisitos possuem são os mesmos definidos na Seção 3.2.1.1. A linguagem LACS permite a definição de uma configuração sem requisitos. Nesse caso, ela irá apenas herdar os requisitos definidos na configuração global.

A Figura 4.1 mostra um exemplo de arquivo XML em LACS que descreve as mesmas configurações definidas na Figura 3.8. A tag **transmission-model** que contém a descrição do protocolo para facilitar o entendimento da estrutura do arquivo XML é omitida e será explicada posteriormente.

O exemplo da Figura 4.1 também contém a definição de mensagens. Conforme definido na Seção 3.2.1.1, cada mensagem contém um nome e um conjunto de comparadores, que podem ser de dois tipos: **mask-comparator** ou **size-comparator**. O primeiro permite a comparação do conteúdo da mensagem com um determinado valor utilizando uma máscara

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE protocol-description SYSTEM "lacs.dtd">
<protocol-description>
  <declaration>...
  <transmission-model  repeat="true">
    <send configuration="tipo-composto" />
    <receive configuration="seguranca-baixa" size="4096"/>
    <send configuration="seguranca-media" />
    <receive configuration="seguranca-baixa" size="4096"/>
    <if>
      <if-condition>
        <send-message name="especial"/>
      </if-condition>
      <if-then>
        <send configuration="seguranca-media" />
        <receive configuration="seguranca-baixa"/>
      </if-then>
      <if-else>
        <loop count="10">
          <stop-condition>
            <compare-message label="msg-recebida" name="fim-loop" />
          </stop-condition>
          <send configuration="seguranca-baixa" />
          <receive configuration="seguranca-baixa" label="msg-recebida" />
        </loop>
      </if-else>
    </if>
  </transmission-model>
</protocol-description>
```

Figura 4.2: Exemplo em XML do Modelo de Transmissão em LACS

que permite a comparação a nível de bits. Já o segundo, permite a comparação de mensagens através de seu tamanho.

A Figura 4.2 mostra o restante o arquivo XML com a definição do modelo de transmissão em LACS. Os comandos são os mesmos definidos na Seção 3.2.1.2, e os parâmetros são informados como atributos das tags XML.

4.2 Módulo de Configuração

Conforme mencionado na Seção 3.1, o Módulo de Configuração interage constantemente com o Módulo de Comunicação, e é responsável por interpretar o arquivo de configuração, a fim de informar, para cada transmissão, a configuração de segurança correspondente. A seguir serão abordados os detalhes de implementação do módulo de configuração. A Figura 4.3 mostra o diagrama de classes do Módulo de Configuração.

Na maioria das vezes, o Módulo de Configuração comunica-se apenas com o Módulo de Comunicação, que utiliza a classe *ConfigurationLayerInterface* para acessar as funcionalidades daquele módulo. O arquivo de configuração é lido durante a instanciação dessa classe, quando o Módulo de Comunicação deve passar como parâmetro o arquivo de configuração. Nesse instante, o Módulo de Configuração cria uma estrutura denominada *SecurityFlow*, que contém todas as informações do arquivo de configuração, isto é, as configurações de segurança que serão utilizadas e as informações sobre o protocolo de comunicação descrito em LACS.

Cada configuração de segurança é descrita pela entidade *SecurityConfiguration*, que possui

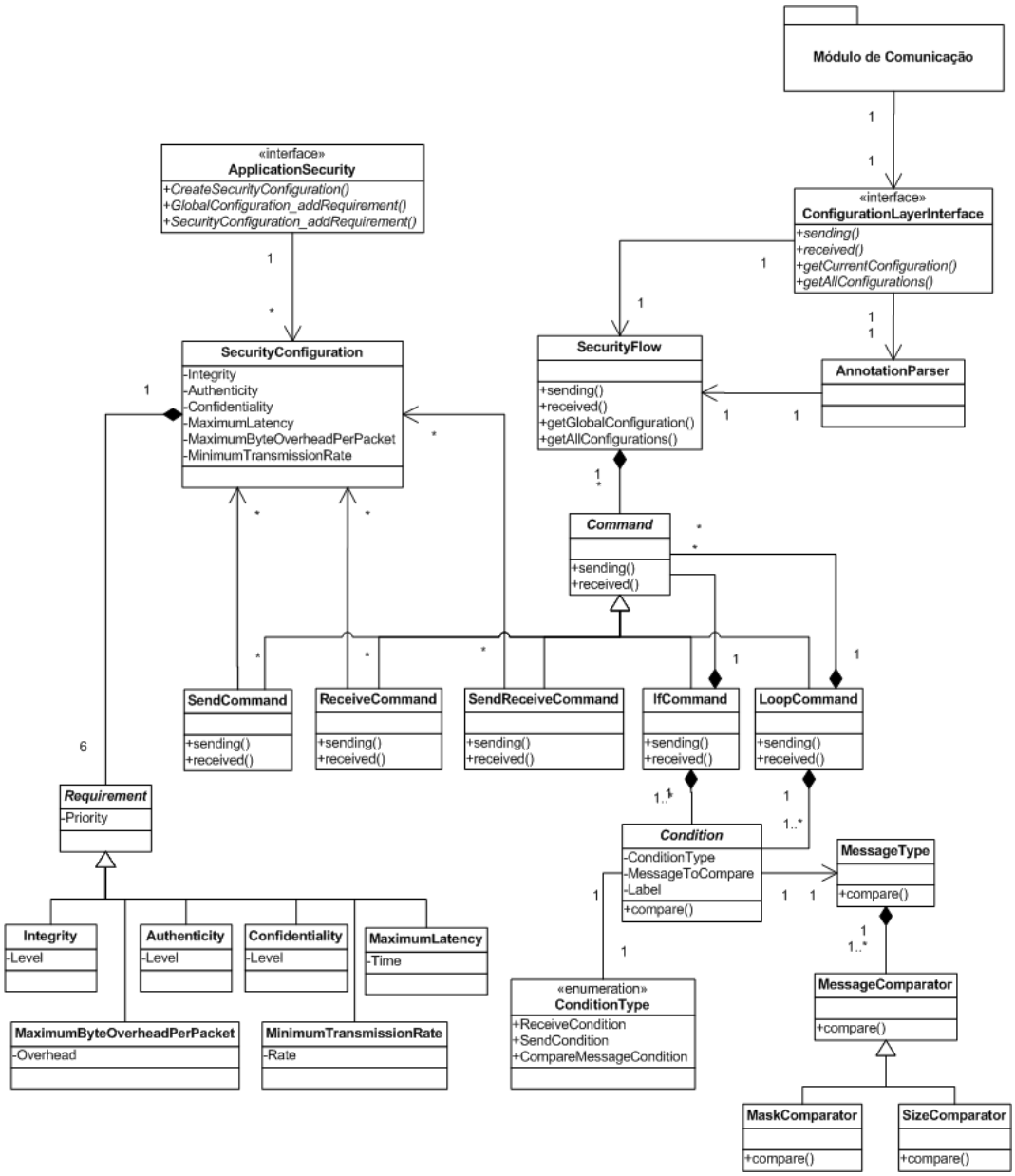


Figura 4.3: Diagrama de Classes da Camada de Segurança

um nome e os requisitos de: integridade, autenticidade, confidencialidade, latência máxima, *overhead* máximo por pacote e taxa de transmissão mínima. As mensagens definidas através da linguagem LACS são mapeadas nas classes *MessageType* que contém um conjunto de comparadores que, por sua vez, podem ser de dois tipos: comparadores de tamanho ou de máscara de bits.

O protocolo descrito no arquivo de segurança é armazenado na classe *SecurityFlow* em um conjunto de comandos. Os comandos podem ser dos cinco tipos definidos pela linguagem (Seção 3.2.1.2): *send*, *receive*, *send-receive*, *if* e *loop*. Os dois últimos comandos contêm um conjunto de condições que permitem a comparação das mensagens transmitidas com mensagens pré-definidas.

Durante a fase de conexão, o Módulo de Comunicação obtém todas as configurações de segurança, a fim de reduzir o conjunto de protocolos disponíveis e determinar o conjunto de protocolos que serão utilizados nas transmissões.

Durante as transmissões, o Módulo de Comunicação utiliza os métodos *sending* e *received* para informar que um dado está sendo enviado ou foi recebido. Esse comando é propagado pelo Módulo de Comunicação através do fluxo de execução do protocolo, de forma que o módulo de configuração consiga determinar a Configuração de Segurança associada à próxima transmissão. A função *getCurrentConfiguration* permite ao Módulo de Comunicação obter a configuração associada à transmissão corrente.

O módulo de configuração também pode ser utilizada no caso em que não é possível descrever o protocolo de comunicação entre as entidades, conforme mencionado na Seção 3.2.2. Nesse caso, a camada de aplicação deve utilizar a classe *ApplicationSecurity* para definir as Configurações de Segurança que irá utilizar, e os requisitos associados à configuração global. Além disso, a cada transmissão, a camada de aplicação precisa informar a configuração de segurança que deve ser considerada.

O módulo de configuração foi implementado de forma a garantir que o protocolo definido no arquivo de configuração está sendo executado rigorosamente. Quando uma inconsistência é encontrada (mensagem de envio quando deveria ser de recebimento, por exemplo), o módulo de configuração lança uma exceção informando a inconsistência, e a aplicação fica responsável por tratar essas exceções e tomar a atitude mais adequada de acordo com suas necessidades.

Capítulo 5

Experimentos e Resultados

Neste trabalho, realizar experimentos reais é uma tarefa difícil, pois o *ASecMid* depende de um número muito grande de variáveis que influenciam em seu comportamento. Por exemplo, a ocorrência de variações ou interferências na rede (comuns, em redes sem fio) são consideradas pela camada de segurança na hora de selecionar o método de segurança. O mesmo ocorre com outros parâmetros como: utilização de CPU, consumo de memória, mobilidade, etc. Enfim, devido ao grande número de variáveis, é difícil analisar se o impacto na escolha do método de segurança ocorreu devido à configuração de segurança associada à transmissão, ou devido a condições externas.

Portanto, neste trabalho foi adotada a estratégia de isolar o sistema, realizando transmissões através do middleware, mas utilizando variáveis de ambiente simuladas, semelhante a [32] em seu trabalho. Os experimentos foram realizados através de transmissões sucessivas durante 60 segundos, sem intervalo entre as transmissões. As transmissões foram realizadas entre computadores Intel(R) Pentium(R) 4 CPU 2.80GHz e 512MB de RAM. Como não se tratam de dispositivos que utilizam bateria, não foi possível obter a métrica de gasto energia.

As métricas coletadas foram: força criptográfica (Definição 11), que mede a segurança dos algoritmos de segurança utilizados; o *throughput*, que é o número de bytes transmitidos por segundo; e o tempo de processamento gasto para escolher um protocolo de segurança e aplicá-lo ao dado. A última métrica foi calculada como a soma do tempo durante o envio e durante o recebimento do dado.

As métricas de *throughput* e tempo de processamento são relacionadas e inversamente proporcionais. Ou seja, quanto menor o tempo gasto para processar o dado, maior o *throughput*. Além disso, os algoritmos de segurança com força criptográfica maior geralmente possuem um tempo de processamento maior, seja pela complexidade do método utilizado (por exemplo, o RSA [20] que é forte, mas é lento), ou pelo fato de usar uma chave de tamanho maior (um algoritmo com uma chave criptográfica de 128 bits é mais lento que um que utiliza uma chave de 256). Entretanto, por se tratar de uma métrica configurável, é possível que essa relação não seja necessariamente verdadeira. O algoritmo AES [38], por exemplo, é um algoritmo confiável, mas que tem um tempo de processamento pequeno. Ele é ideal, por exemplo, quando o requisito de segurança é alto, mas existe alguma restrição quanto ao tempo de latência.

Parâmetro	Valor
Capacidade de Memória	8 MB
Memória livre	1 MB
Bateria	100%
Utilização de CPU	40%
Tipo de Rádio	Wi-Fi
Roteamento	Direto
Mobilidade	10%
Qualidade do Canal	80%
Latência	1 ms

Tabela 5.1: Valor dos parâmetros simulados nos experimentos

As próximas seções apresentam os experimentos realizados para avaliar a solução proposta neste trabalho. Em alguns experimentos utilizamos métodos tradicionais de criptografia para comparar com os resultados obtidos pelo trabalho proposto. Para isso utilizamos os algoritmos: PGP [13] e S/MIME [21], utilizados para segurança de e-mails; o IPSec [12], que é amplamente utilizado em redes sem fio; e o protocolo SET [35], que é utilizado para transações bancárias. Esses algoritmos foram escolhidos por serem algoritmos amplamente utilizados em aplicações reais para ambientes móveis. Também fez parte dos experimentos comparar os resultados com a transmissão pura, ou seja, sem utilizar nenhum método de segurança. Dessa forma é possível ter uma referência sobre os valores limites (mínimos ou máximos) para as métricas avaliadas.

Todos os experimentos foram realizados 33 vezes, a fim de obter o valor médio para as métricas e o erro com um intervalo de confiança de 90%. Todos os experimentos foram realizados simulando uma transmissão FTP, na qual o transmissor envia 1 pacote de dados de 64 Kbytes e recebe uma mensagem de confirmação de 4 Kbytes. O conteúdo dos pacotes foi gerado aleatoriamente.

A Tabela 5.1 mostra os valores dos parâmetros que foram simulados e utilizados em todos os experimentos. Esses valores permaneceram fixos durante todas as transmissões. A mobilidade de 10% indica que há pouca mibilidade, a qualidade do canal em 80% indica que há pouca interferência, a utilização de CPU de 40%, por sua vez, indica baixa utilização. O roteamento direto indica que nenhuma outra entidade está sendo utilizada para rotear os pacotes, ou seja, a entidade que transmite está dentro do raio de alcance da entidade que recebe. Os demais parâmetros também foram escolhidos simulando condições normais de transmissão.

As próximas seções mostram mais detalhes sobre cada experimento. O Apêndice B mostra os algoritmos utilizados em cada experimento, assim como o número médio de transmissões realizadas por segundo com cada algoritmo. O Apêndice C mostra os arquivos de configuração completos em xml utilizados em cada experimento

N. Segurança	0	1	2	3	4	5	6	7	8	9	10
Desabilitado	100%	90%	80%	70%	60%	50%	40%	30%	20%	10%	0%
Máximo	0%	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%

Tabela 5.2: Experimento 1 - Porcentagem de dados transmitidos com cada nível de segurança em cada arquivo de configuração

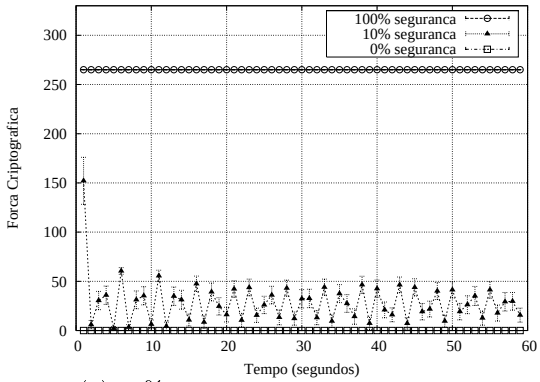
5.1 Variando a porcentagem de dados transmitidos com segurança

O objetivo deste experimento é avaliar o impacto da porcentagem de dados transmitidos utilizando um método forte de segurança em relação à quantidade de dados transmitidos sem segurança. Para isso, foram utilizados 11 arquivos de configuração diferentes (numerados de 0 a 10) que utilizam apenas duas configurações de segurança: uma com todos os níveis de segurança desabilitados (confidencialidade, integridade e autenticidade), e outra com esses requisitos com nível crítico. A porcentagem de dados transmitidos com segurança máxima varia de 0% a 100%, conforme detalha a Tabela 5.2. Cada uma das colunas representam um arquivo de configuração, informando a porcentagem de dados transmitidos com o nível de segurança máximo e desabilitado.

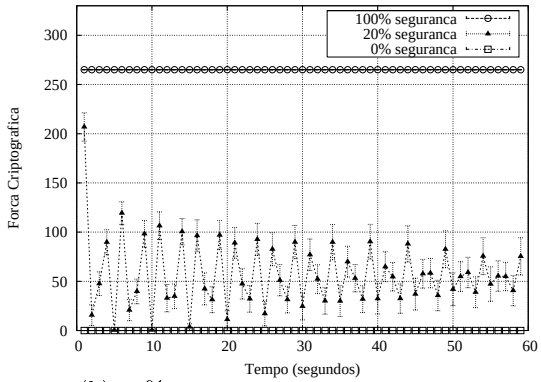
O Apêndice C contém cada arquivo de configuração utilizado nos experimentos. A configuração utilizada para transmissões sem segurança é composta por níveis de segurança (autenticidade, confidencialidade e integridade) desabilitados e requisitos de qualidade de serviço desabilitados. A configuração de segurança utilizada para transmissões seguras é composta por requisitos de segurança com nível crítico, requisito de latência máxima de 8000 microssegundos, e demais requisitos desabilitados. As Figuras 5.1, 5.2 e 5.3 apresentam os gráficos com a variação das métricas nos experimentos. Os experimentos com 40% foram omitidos por questões de espaço, mas apresentaram um comportamento semelhante aos demais, com valores médios entre os valores correspondentes dos experimentos com 30% e 50% dos dados transmitidos segurança.

A Figura 5.1 mostra o comportamento da força criptográfica nos experimentos realizados. A força criptográfica dos algoritmos varia entre 0 (transmissão nenhum algoritmo de segurança) e 265 (protocolo de segurança mais forte). Como era de se esperar, as transmissões com 0% e 100% de segurança ocorrem sempre com a mesma configuração e, portanto, não apresentam variação da força criptográfica. Já as demais transmissões, com segurança entre 10% e 90%, alternam entre transmissões com muita e nenhuma segurança e, por isso, apresentam uma curva que altera entre alta e baixa força criptográfica. Através dos experimentos, podemos concluir que a força criptográfica média geralmente cresce quando a porcentagem de dados transmitidos com segurança aumenta.

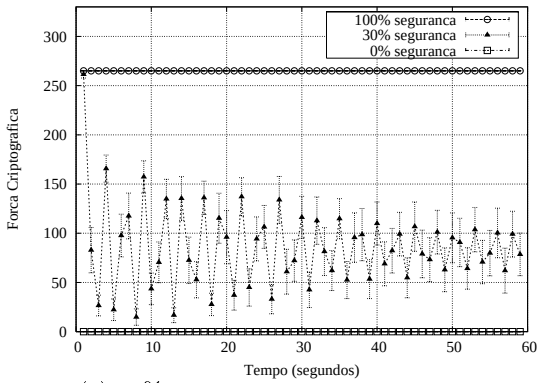
A Figura 5.2 apresenta o comportamento do *throughput* em relação à porcentagem de dados transmitidos com segurança. Conforme esperado, as configurações de 0% e 100% de segurança tiveram os *throughputs* de maior e menor valor médio, respectivamente. Ao contrário da



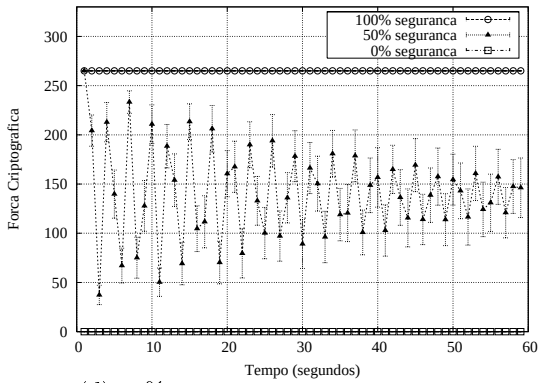
(a) 10% transmitidos com segurança



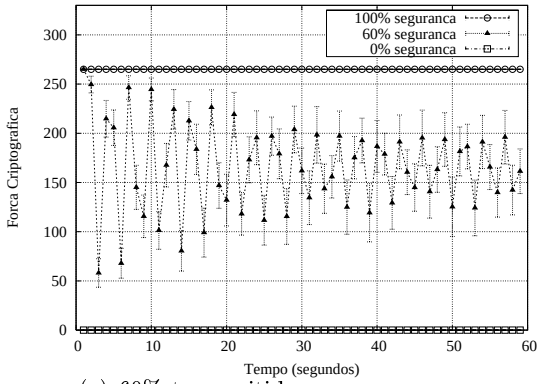
(b) 20% transmitidos com segurança



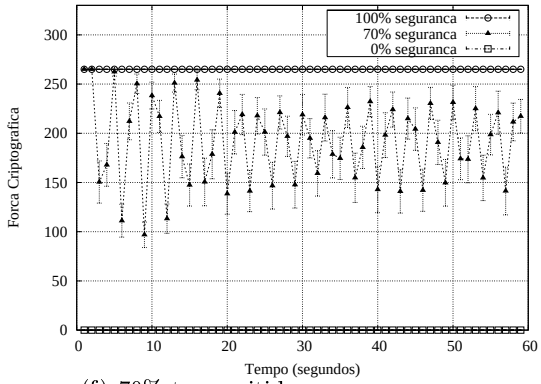
(c) 30% transmitidos com segurança



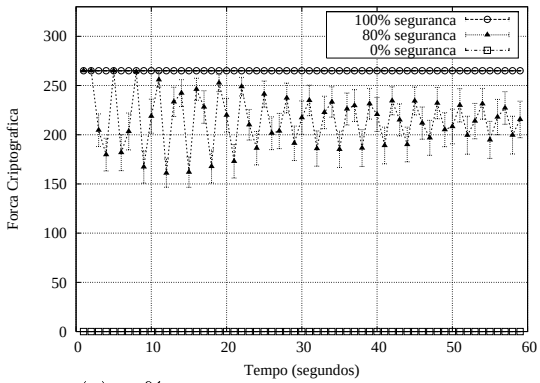
(d) 50% transmitidos com segurança



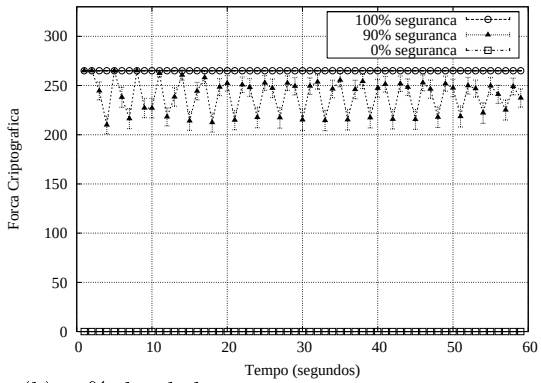
(e) 60% transmitidos com segurança



(f) 70% transmitidos com segurança

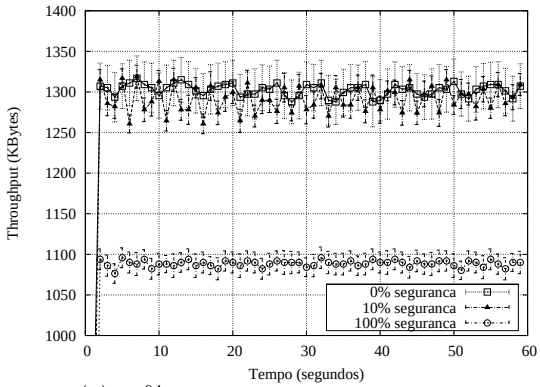


(g) 80% transmitidos com segurança

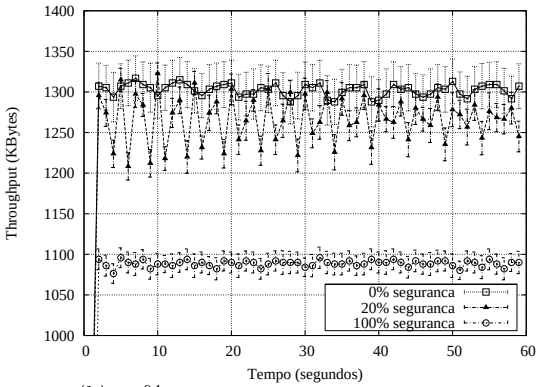


(h) 90% dos dados transmitidos com segurança

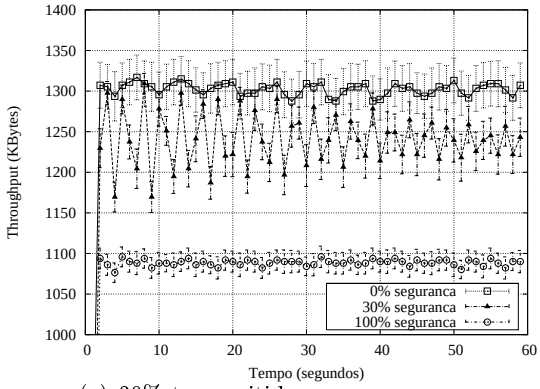
Figura 5.1: Experimento 1 - Variação da Força Criptográfica em relação à porcentagem de dados transmitidos com segurança



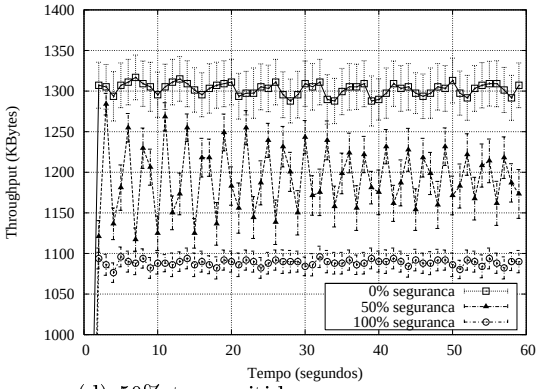
(a) 10% transmitidos com segurança



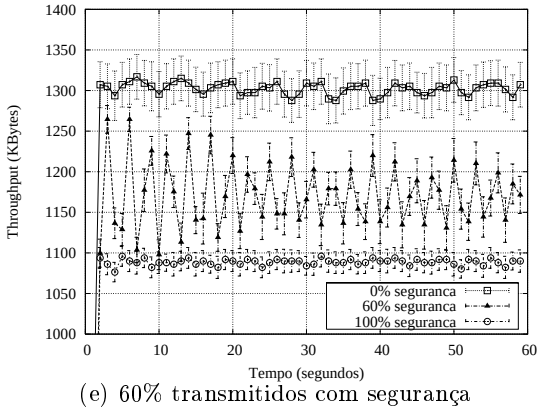
(b) 20% transmitidos com segurança



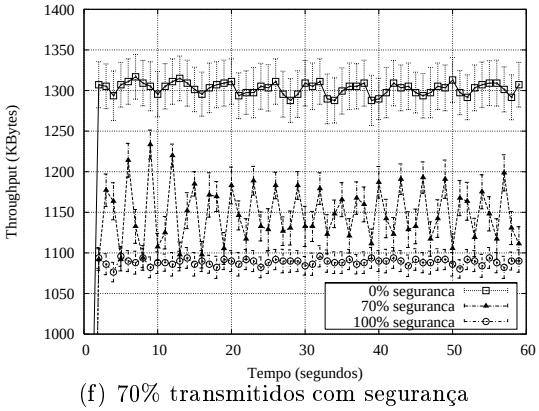
(c) 30% transmitidos com segurança



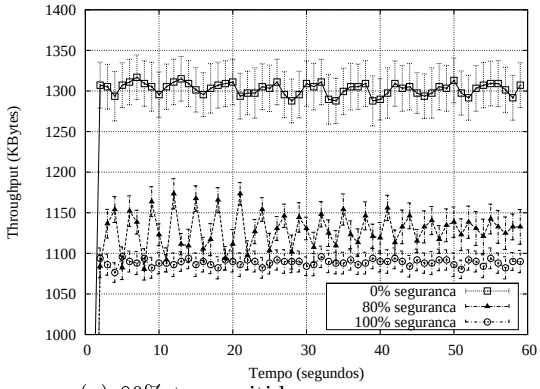
(d) 50% transmitidos com segurança



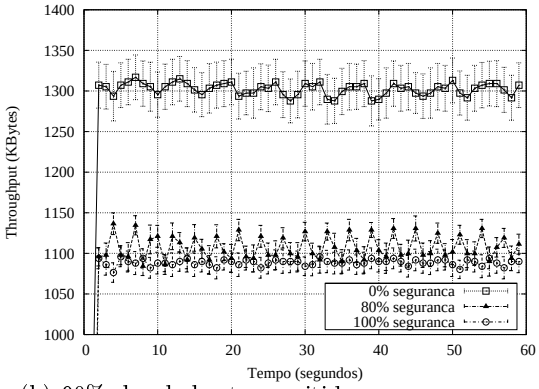
(e) 60% transmitidos com segurança



(f) 70% transmitidos com segurança

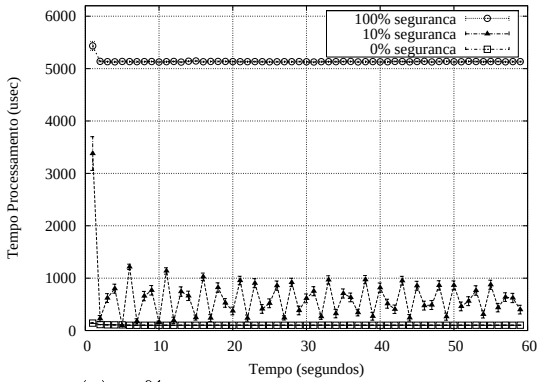


(g) 80% transmitidos com segurança

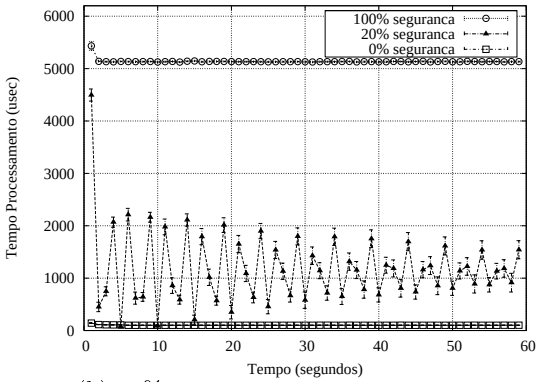


(h) 90% dos dados transmitidos com segurança

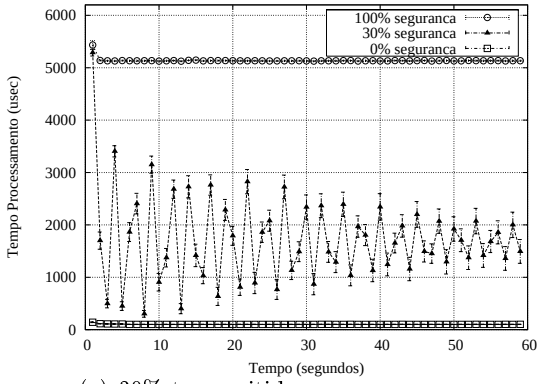
Figura 5.2: Experimento 1 - Variação do *throughput* em relação à porcentagem de dados transmitidos com segurança



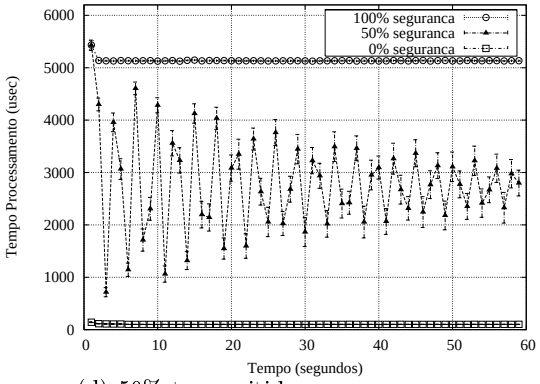
(a) 10% transmitidos com segurança



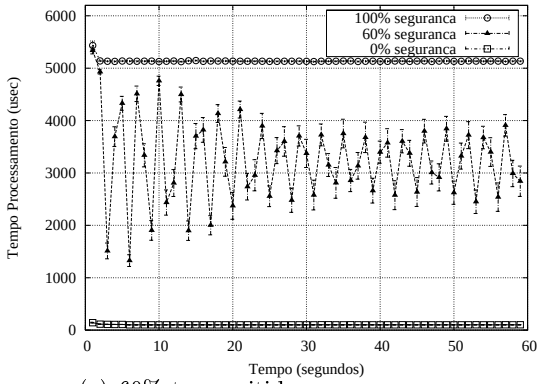
(b) 20% transmitidos com segurança



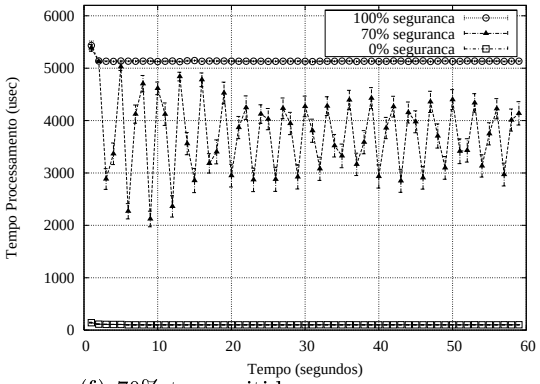
(c) 30% transmitidos com segurança



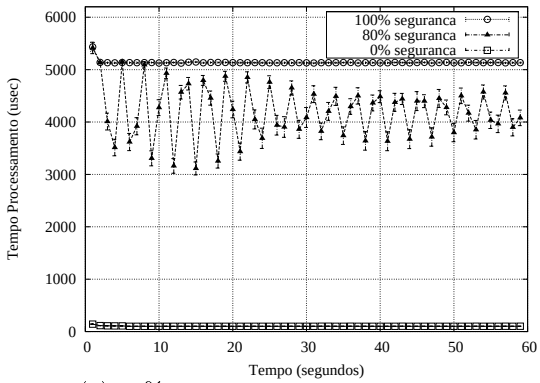
(d) 50% transmitidos com segurança



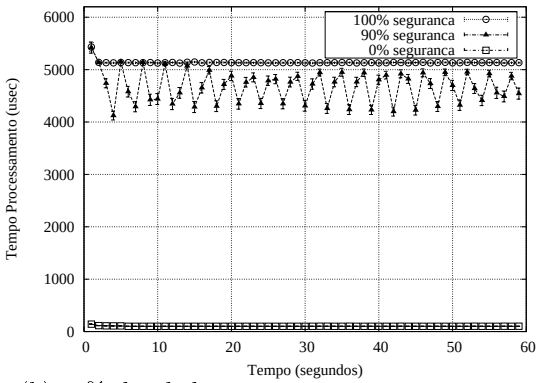
(e) 60% transmitidos com segurança



(f) 70% transmitidos com segurança



(g) 80% transmitidos com segurança



(h) 90% dos dados transmitidos com segurança

Figura 5.3: Experimento 1 - Variação do tempo de processamento em relação à porcentagem de dados transmitidos com segurança

força criptográfica, o valor do *throughput* é inversamente proporcional à quantidade de dados transmitidos com segurança. Isto porque a transmissão com 100% de segurança executa o processamento mais pesado e, portanto, apresenta a menor quantidade de dados transmitidos. As demais transmissões acompanham a tendência de aumentar o *throughput* na medida em que a porcentagem de dados transmitidos com segurança diminui.

Finalmente, a Figura 5.3 mostra a variação do tempo de processamento em relação à porcentagem de dados transmitidos com segurança. Através dos gráficos é possível perceber que o tempo de processamento é diretamente proporcional à quantidade de dados transmitidos com segurança. A curva da transmissão com 0% de segurança mostra o tempo gasto pela camada de segurança para escolher um protocolo de segurança, uma vez que nenhum tempo foi gasto para aplicar algoritmos de segurança.

Na maioria dos gráficos, os valores das métricas apresentaram uma variação da amplitude das oscilações. Essa variação ocorre porque, a cada segundo, o número de transmissões com alta ou baixa segurança é diferente, o que implica em diferentes valores médios da métrica a cada segundo.

5.2 Níveis de segurança variáveis

O objetivo deste experimento é avaliar o comportamento dos métodos de segurança quando a configuração de segurança varia ao longo do tempo. Para isso, foram realizados experimentos com diversos arquivos de configuração com níveis de segurança variáveis. Esses experimentos foram comparados com transmissões utilizando métodos tradicionais e transmissões sem utilizar a camada de segurança e nenhum método de segurança. Para avaliar o comportamento da camada de segurança, foram utilizados os seguintes arquivos de configuração:

Arquivo 0 Transmissão com todos os requisitos de segurança e qualidade de serviço desabilitados. Ou seja, transmissão sem utilizar nenhuma segurança.

Arquivo 1 Conforme mencionado anteriormente, todos os experimentos simulam uma transmissão FTP. Mas nesse arquivo de configuração específico, o dado é transmitido com segurança e a mensagem de confirmação é recebida sem utilizar nenhum algoritmo de segurança. Isso é razoável porque a mensagem de confirmação não contém nenhuma informação importante e serve apenas para controle dos pacotes transmitidos.

Arquivo 2 Transmissão em três níveis de segurança (crítico, opcional e desabilitado), sendo que a quantidade de dados enviado com segurança desabilitada (300 mensagens) é três vezes maior que a quantidade de dados enviados com segurança opcional (100 mensagens), que, por sua vez, é três vezes maior que a quantidade transmitida com segurança crítica (33 mensagens).

Arquivo 3 Transmissão com três níveis de segurança (crítico, opcional e desabilitado) com a mesma quantidade de transmissões para cada nível de segurança (300 mensagens).

Arquivo 4 Transmissão com dois níveis de segurança (crítico e desabilitado) com a mesma quantidade de transmissões para cada nível (300 mensagens).

Arquivo 5 Transmissão com dois níveis (crítico e desabilitado), com o número de transmissões (300 mensagens) sem segurança três vezes maior que com segurança desabilitada (100 mensagens).

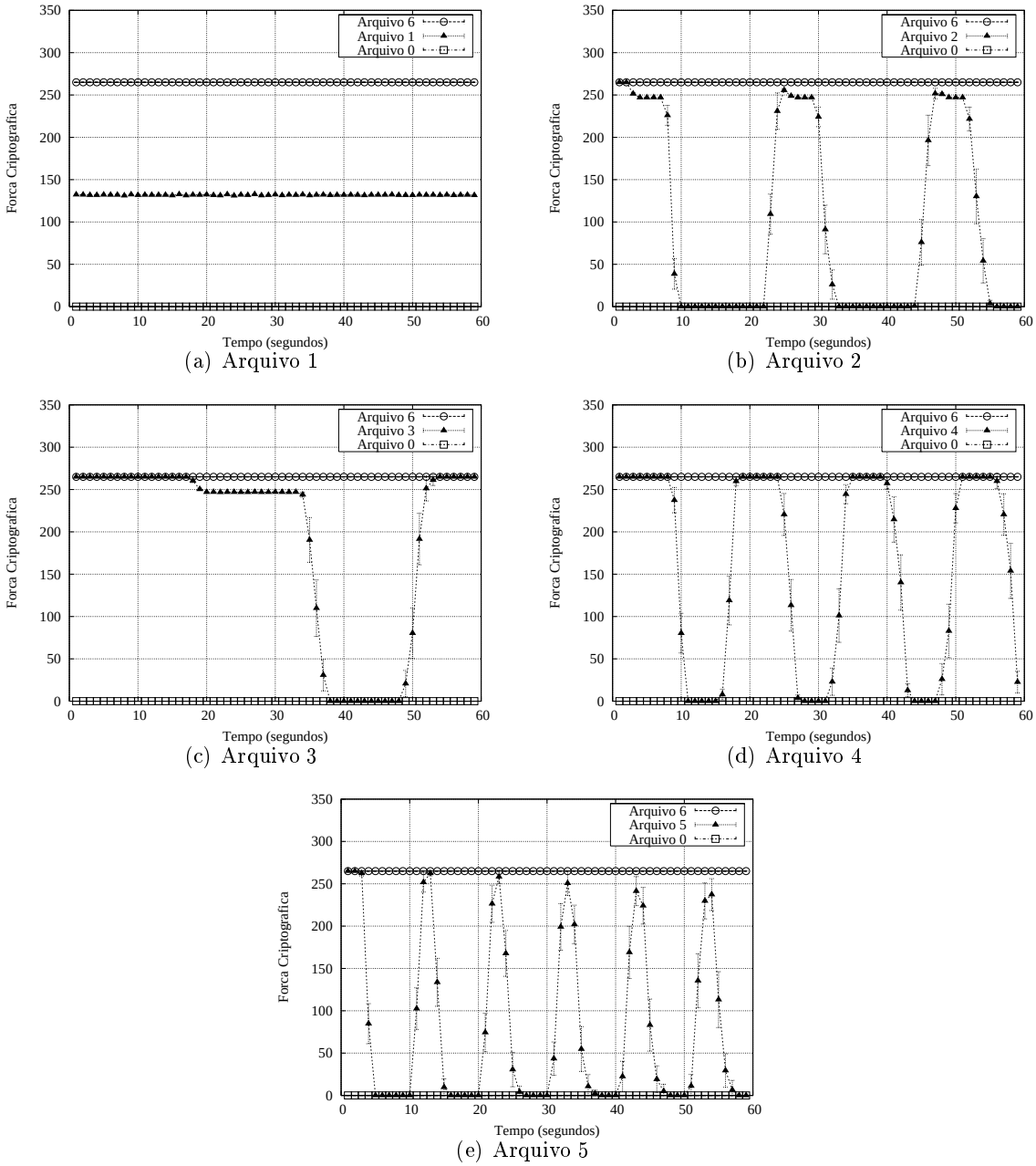
Arquivo 6 Transmissão apenas com segurança alta (nível crítico).

Em todas os arquivos acima, os níveis de segurança mencionados se referem aos três requisitos de segurança: confidencialidade, autenticidade e integridade. Os arquivos de configuração estão disponíveis no Apêndice C. Todas as configurações de segurança que possuem nível de segurança crítico e incluem o requisito de latência máxima de 8000 microssegundos, enquanto as configurações com nível de segurança opcional e incluem o requisito de latência máxima de 4000 microssegundos.

A Figura 5.4 mostra a variação da força criptográfica média das configurações utilizadas nos experimentos. Todos os gráficos mostram as transmissões utilizando os arquivos 0 e 6, que são referência por serem as configurações com nível de segurança mínimo e máximo, respectivamente. Como utilizam a mesma configuração de segurança durante todo o experimento, essas duas configurações mantém a força criptográfica constante ao longo do tempo. O arquivo 1 obteve um valor médio de força criptográfica, pois suas transmissões alternam entre uma transmissão com segurança alta (enviando o dado) e nenhuma segurança (recebendo o pacote de confirmação). As demais configurações, como são formadas por um número maior de transmissões com cada nível de segurança, mostram um gráfico que alterna entre força criptográfica alta, média (no caso dos Arquivos 2 e 3, que utilizam 3 configurações de segurança diferentes) e baixa. Pelo gráfico é possível perceber as mudanças da configuração de segurança utilizada nas transmissões. As configurações com três níveis realizaram transmissões com 3 valores diferentes para a força criptográfica: 265 (segurança alta), 250 (segurança média) e 0 (sem segurança). Os arquivos que possuem um número de transmissões maior em determinada configuração (arquivos 2 e 5), o gráfico mostra claramente o tempo maior realizando transmissões com essa configuração.

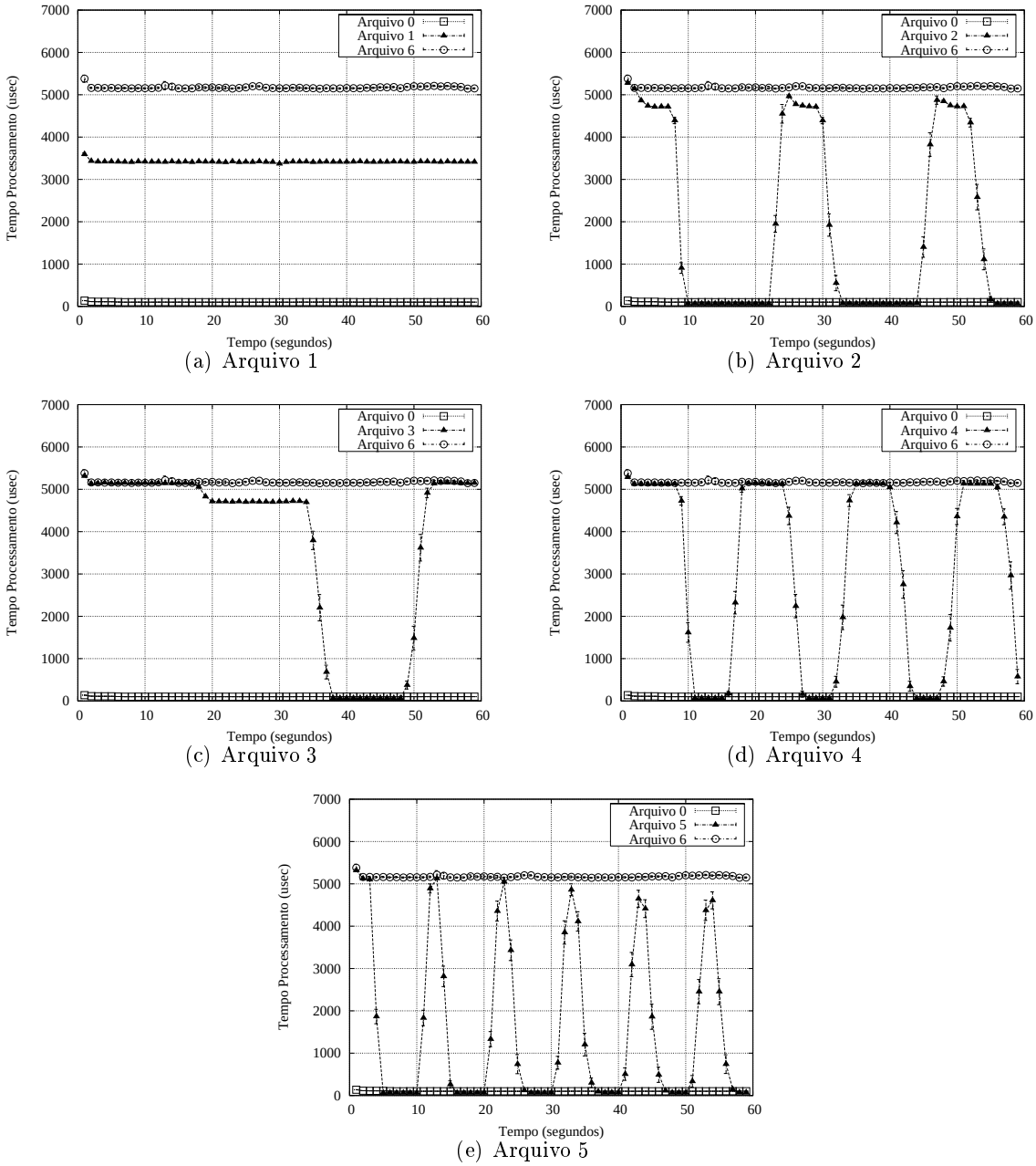
Os gráficos da Figura 5.5 mostram a variação do tempo de processamento gasto pelas transmissões utilizando a camada de segurança. A configuração do arquivo 1 (FTP) que transmite o pacote de confirmação sem nenhuma segurança, permitiu reduzir o tempo de processamento em cerca de 60%, o que indica que, mesmo transmitindo a maior parte dos dados utilizando um nível de segurança alto (pois o pacote de dados é maior que o de confirmação), é possível reduzir consideravelmente o tempo de processamento gasto com execução de algoritmos segurança. Os resultados dos experimentos com os demais arquivos de configuração mostram que é possível reduzir consideravelmente o tempo gasto com algoritmos de segurança, principalmente quando existem transmissões que não utilizam algoritmos de segurança, que permitem a redução do tempo de processamento a quase zero.

O gráfico da Figura 5.6 mostra o tempo de processamento dos métodos tradicionais de segurança comparados com o tempo gasto pelas transmissões utilizando a camada de segurança



(a) Arquivo 1 (b) Arquivo 2 (c) Arquivo 3 (d) Arquivo 4 (e) Arquivo 5

Figura 5.4: Experimento 2 - Força Criptográfica em relação à variação do nível de segurança



(a) Arquivo 1

(b) Arquivo 2

(c) Arquivo 3

(d) Arquivo 4

(e) Arquivo 5

Figura 5.5: Experimento 2 - Tempo de Processamento em relação à variação do nível de segurança

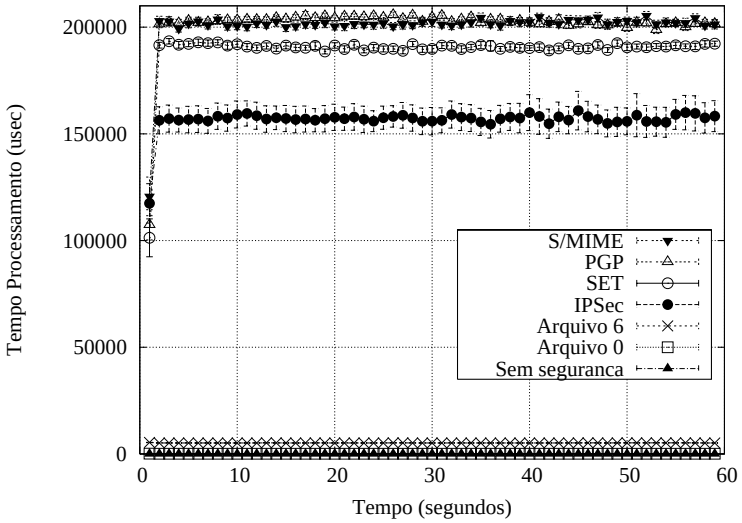
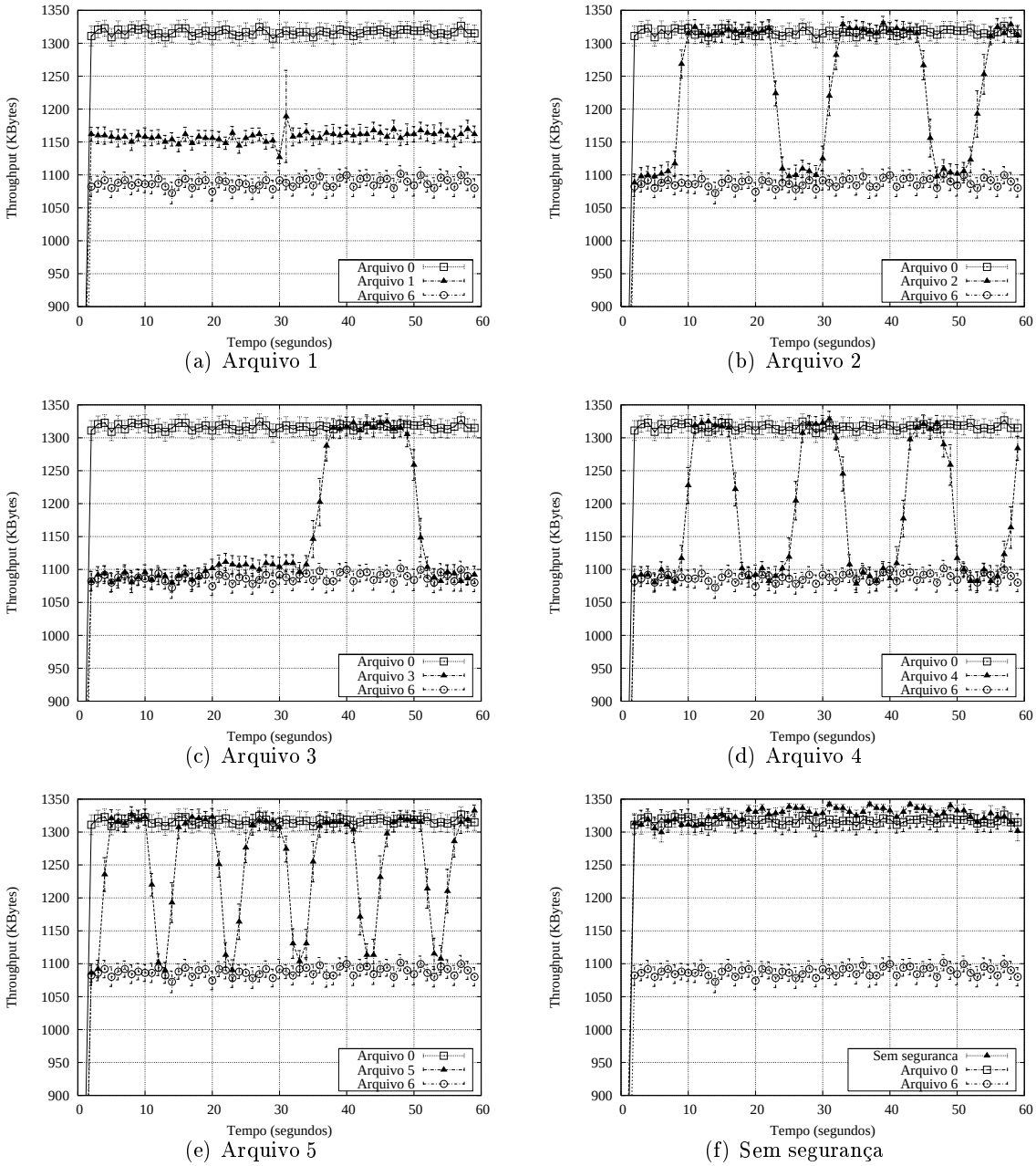


Figura 5.6: Experimento 2 - Tempo de Processamento - Comparação com métodos estáticos

com os arquivos 0 e 6, que representam o menor e maior tempo gasto. Como nas transmissões utilizando a camada de segurança, o requisito de latência foi configurado com o valor máximo de 8ms, as transmissões com os arquivos 0 e 6 não tiveram tempo de processamento maior que esse valor. Já os métodos tradicionais, tiveram tempo entre 150 e 210 milisegundos. A pequena diferença entre o tempo da transmissão pura (sem segurança) e o tempo do arquivo 0 (com os requisitos desabilitados) corresponde ao tempo gasto pela camada de segurança para determinar qual o melhor algoritmo de segurança, uma vez que nenhum tempo foi gasto aplicando algoritmos de segurança. Isso quer dizer que, em situações como as utilizadas nos experimentos (utilizando computadores com grande poder computacional), o *overhead* de tempo devido ao algoritmo de escolha do melhor protocolo de segurança em cada transmissão é muito pequeno.

A Figura 5.7 mostra os gráficos da variação do *throughput* nas transmissões utilizando os arquivos de configurações. Os gráficos mostram um aumento de cerca de 20% no *throughput* quando ocorrem transmissões com nível de segurança desabilitado. Conforme esperado, todos os gráficos mostraram o comportamento do *throughput* semelhante ao tempo de processamento, mas inversamente proporcional. O gráfico da Figura 5.7(f) mostra a pequena diferença do *throughput* de transmissões puras (sem segurança) em relação a transmissões utilizando o arquivo 0 (níveis de segurança desabilitados), o que comprova que o *overhead* causado pelo algoritmo de escolha do melhor protocolo de segurança em cada transmissão é muito pequeno.

Os gráficos da Figura 5.8 mostram ainda a comparação com os métodos tradicionais de segurança. A maioria desses métodos tiveram um *throughput* menor devido ao tempo de processamento maior. Entretanto, essa diferença não foi muito grande em relação à transmissão com o Arquivo 6 (segurança máxima), o que indica que os maiores ganhos em *throughput* ocorrem quando há configurações de segurança com níveis de segurança desabilitados. A utilização de protocolos de segurança mais fracos aumentam o *throughput* mas não muito, pois

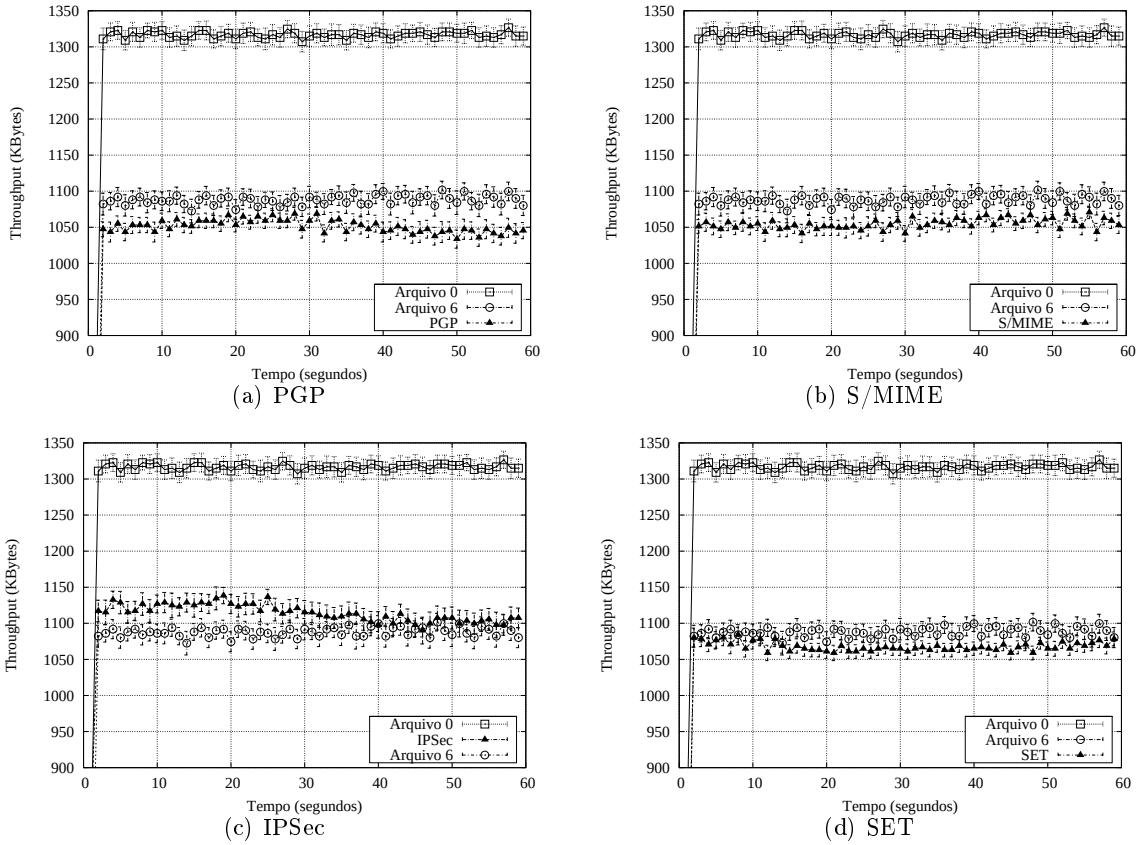


(a) Arquivo 1 (b) Arquivo 2

(c) Arquivo 3 (d) Arquivo 4

(e) Arquivo 5 (f) Sem segurança

Figura 5.7: Experimento 2 - Throughput em relação à variação do nível de segurança



(a) PGP (b) S/MIME

(c) IPSec (d) SET

Figura 5.8: Experimento 2 - Throughput - Comparação com métodos estáticos

mesmo os algoritmos mais fracos gastam um tempo considerável para aplicar o mecanismo de segurança. Essa constatação pode ser confirmada nos gráficos das Figuras 5.7(b) e 5.7(c), que indicam o pequeno ganho de *throughput* com as transmissões com nível médio (opcional) de segurança.

5.3 Variando o requisito de latência máxima

Para analisar o impacto do requisito de latência nas métricas avaliadas, realizamos um experimento comparando configurações de segurança de transmissões com diferentes valores para o requisito de latência máxima. A Tabela 5.3 mostra o valor do requisito de latência máxima para as configurações utilizadas no experimento. A configuração 0 não especifica a latência máxima e corresponde a uma configuração com todos os requisitos desabilitados (transmissão sem segurança). A configuração 5 também não define a latência máxima e possui os demais requisitos com os mesmos valores das demais configurações, para que seja possível encontrar o tempo de latência esperado se não houvesse a restrição de latência.

O Apêndice C mostra os arquivos de configurações utilizados no experimento. Nas configurações de segurança, o nível de integridade foi definido como desejável, e o nível de confidencialidade e autenticidade como obrigatório. Apenas a configuração com latência máxima de 1,5 milisegundos, por se tratar de uma restrição muito forte de latência, teve uma configu-

Configuração de Segurança	0	1	2	3	4	5
Latência Máxima (ms)	-	1,5	8	30	600	-

Tabela 5.3: Experimento 3 - Modelos de Transmissão

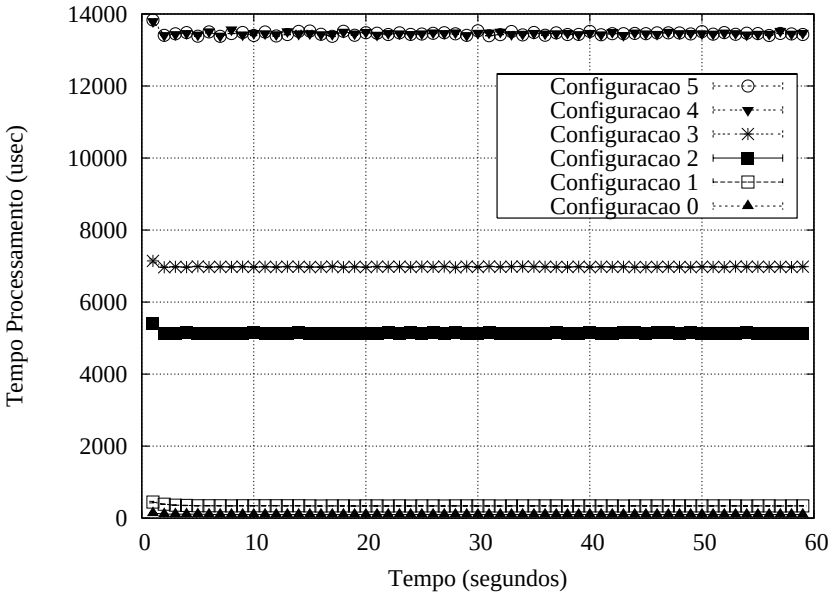


Figura 5.9: Experimento 3 - Variação do tempo de processamento em relação à Latência Máxima

ração de segurança mais fraca: níveis de confidencialidade e autenticidade opcionais, e nível de integridade desejável.

Nesse experimento, as configurações foram comparadas também com os algoritmos estáticos de segurança (PGP, S/MIME, IPSec e Set) e com a transmissão pura, isto é, sem nenhum algoritmo de segurança.

O gráfico da Figura 5.9 mostra o tempo de processamento gasto nas transmissões utilizando a camada de segurança com as configurações mencionadas. Nenhuma configuração obteve um tempo de processamento maior que o requisito estabelecido para a latência máxima, o que significa que a camada de segurança consegue se adequar bem ao requisito de latência, mesmo quando esse requisito representa fortes restrições. A configuração 5 (sem restrição de latência) obteve o mesmo tempo de processamento que a configuração 4 (que possuía uma restrição de 600ms), o que significa que a restrição de latência máxima de 600ms não influenciou na escolha do protocolo de segurança. As demais configurações, que possuem restrições mais fortes de latência, provocam a seleção de protocolos de segurança mais rápidos.

A Figura 5.10 mostra o gráfico que compara o tempo de processamento gasto pelo método mais demorado utilizando a camada de segurança (configuração 5) com os métodos tradicionais de segurança. Os métodos tradicionais tiveram um tempo de processamento maior que as transmissões utilizando a camada de segurança, por que o nível de segurança estabelecido nas configurações ocasionou a escolha de um protocolo de segurança mais fraco e, portanto, mais

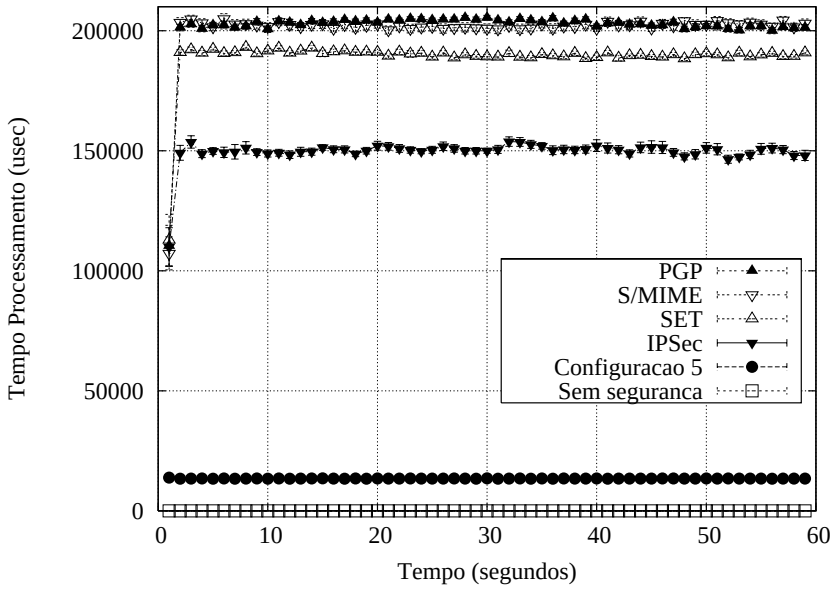


Figura 5.10: Experimento 3 - Variação do tempo de processamento em relação à Latência Máxima

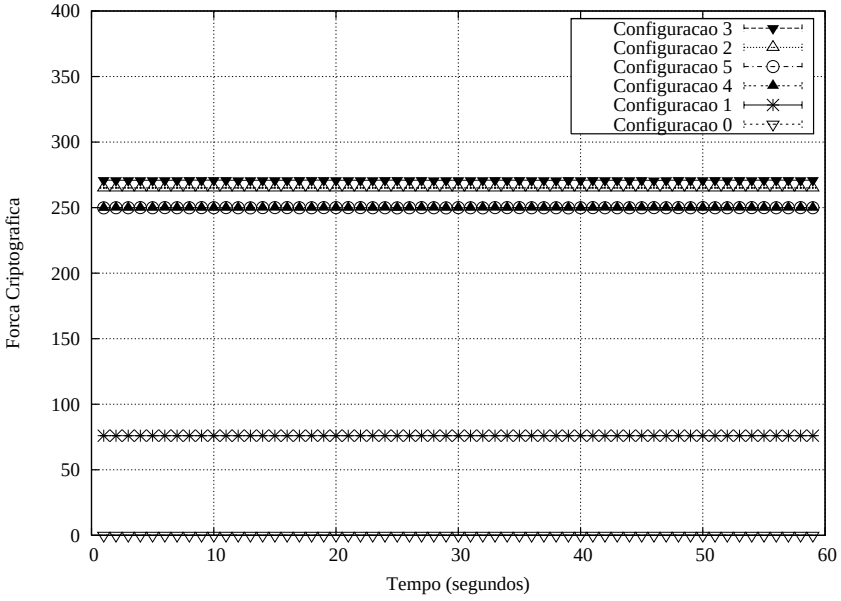
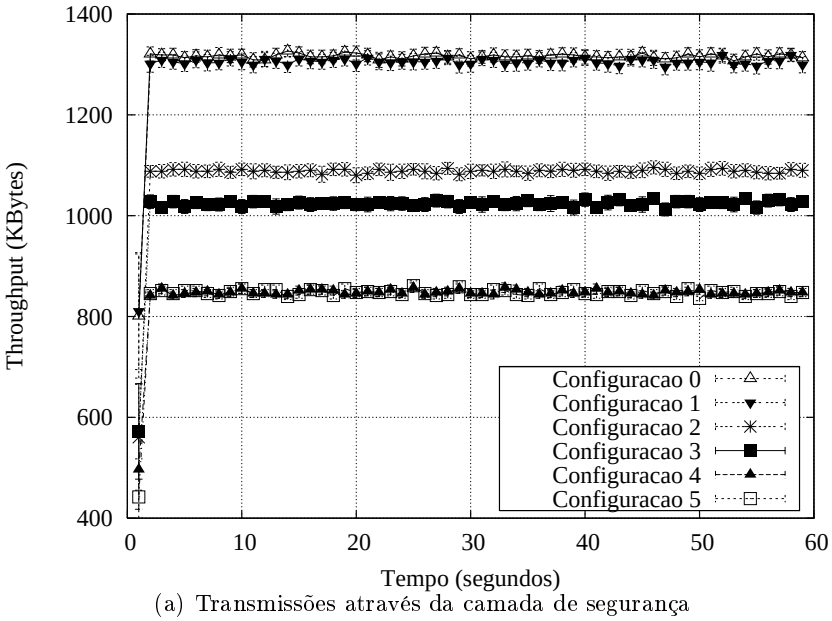
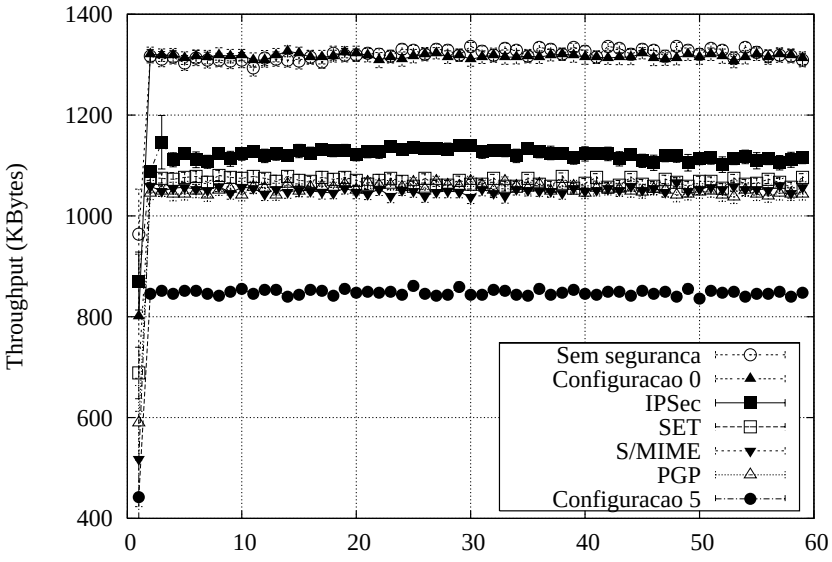


Figura 5.11: Experimento 3 - Variação da Força Criptográfica em relação à Latência Máxima



(a) Transmissões através da camada de segurança



(b) Comparação com métodos tradicionais

Figura 5.12: Experimento 3 - Variação do Throughput em relação à Latência Máxima

rápido.

A Figura 5.11 mostra a força criptográfica média das configurações utilizadas nos experimentos. Embora o tempo de processamento tenha diminuído com o aumento da latência máxima, a força criptográfica não possui uma relação tão direta com o requisito de latência máxima. Isso ocorre porque, não necessariamente, o protocolo mais demorado é o que possui maior força criptográfica. O AES, por exemplo, é um algoritmo que provê uma segurança considerável, mas possui pequeno tempo de latência [42].

A Figura 5.12 mostra a relação do *throughput* e a variação da latência máxima. Conforme esperado, o *throughput* é inversamente proporcional ao tempo de processamento. Ou seja,

os métodos com tempo máximo de latência maior utilizaram métodos mais demorados e, portanto, tiveram um *throughput* menor. A Figura 5.12(b) permite a comparação dos métodos de segurança tradicionais com as transmissões utilizando as configurações 0 e 5 utilizando a camada de segurança. Conforme constatado nos experimentos anteriores, a pequena diferença entre o *throughput* da configuração 0 e a transmissão pura (sem métodos de segurança) indica que o algoritmo de escolha do melhor método de segurança a cada transmissão não causa nenhum impacto significativo na quantidade de dados transmitidos por segundo.

Capítulo 6

Conclusões e Trabalhos Futuros

6.1 Conclusões

Este trabalho propôs uma adaptação para os mecanismos de segurança em ambientes móveis que permitiu a transmissão de dados com métodos de segurança que variam dinamicamente de acordo com o valor semântico do dado transmitido, conforme o *Princípio da Proteção Adequada* [29]. A solução proposta permitiu também a abstração da camada de aplicação em relação aos detalhes de implementação dos algoritmos de segurança, ficando a cargo da camada de segurança determinar e aplicar o protocolo mais adequado, de acordo com os requisitos estabelecidos pela aplicação para cada transmissão.

Este trabalho também definiu uma linguagem de anotação de segurança (LACS - Seção 3.2.1) própria para a descrição de protocolos de comunicação com requisitos de segurança e QoS associados a cada transmissão. A linguagem foi definida em um formato semelhante ao de uma linguagem procedural, permitindo a descrição da maior quantidade possível de protocolos de comunicação. Também foram definidos recursos para a definição de protocolos mais rígidos ou mais flexíveis, de acordo com a necessidade da aplicação.

Para tratar os casos em que a linguagem LACS não consegue descrever o protocolo de comunicação, foi proposta uma abordagem (Seção 3.2.2) que permite à aplicação informar a configuração de segurança associada à transmissão na própria diretiva de comunicação (*send* ou *receive*), no momento da transmissão. Dessa forma, é possível utilizar a camada de segurança para qualquer protocolo de comunicação entre duas entidades.

A utilização do *ASecMid* no módulo de comunicação tornou o módulo de comunicação ainda mais adaptativo, permitindo a adaptação dos métodos de segurança não só em relação à variação da configuração associada ao dado, mas também em relação ao contexto da rede e dos dispositivos envolvidos. Dessa forma, a camada de segurança tornou-se completamente adaptativa e sensível ao contexto, no sentido de se adaptar às condições em que está submetida.

Os resultados experimentais mostraram melhorias significativas no desempenho das transmissões, principalmente quando existiam transmissões sem nenhuma segurança. Uma transmissão em FTP, por exemplo, que enviava um dado de 64 KB com segurança máxima e recebia uma confirmação de 4KB sem segurança (o que é razoável, pois o conteúdo da confirmação

não é confidencial nem significativo para a transmissão), obteve uma melhoria de cerca de 60% no tempo gasto com processamento com algoritmos de segurança, e de quase 10% no *throughput*. Esse resultado nos permite concluir que uma redução de menos de 6% na quantidade de dados transmitidos com segurança, é capaz de produzir ganhos consideráveis nas métricas avaliadas.

6.2 Trabalhos Futuros

Um trabalho futuro imediato é avaliar o comportamento da camada de segurança proposta para redes ad hoc em transmissões reais entre dispositivos móveis, a fim de se avaliar os ganhos de energia, memória utilizada, entre outras métricas. Também é interessante avaliar o impacto das diversas variáveis do ambiente e dos dispositivos na escolha dos protocolos de segurança.

Outra direção possível é estudar o comportamento da solução proposta na Seção 3.3.2 para redes de sensores sem fio. Um estudo detalhado pode avaliar o impacto da solução proposta no consumo de energia dos nós e no tempo de vida da rede, entre outras variáveis.

Uma limitação importante deste trabalho é que ele só trata de comunicações entre apenas duas entidades, não sendo possível realizar transmissões em *multicast* ou *broadcast*. Trabalhos futuros podem ser feitos na direção de suportar comunicações em grupo ou em *broadcast*. Nesse caso, é preciso adaptar a linguagem LACS para considerar mais de duas entidades durante a comunicação.

Outro trabalho possível é pesquisar entre os diversos protocolos existentes na literatura, protocolos como o FTP, que possam ser adaptados para seja possível economizar recursos transmitindo algumas mensagens com um nível de segurança mais baixo, ou mesmo desabilitado.

Trabalhos futuros também podem ampliar o poder da linguagem de anotação de segurança para permitir que o próprio dado carregue as informações sobre a sua segurança. Dessa forma seria possível que, em uma transmissão de um vídeo, o próprio vídeo levasse consigo as informações sobre o nível de segurança associado a cada trecho. Dessa forma, cada dispositivo que recebesse o vídeo seria capaz de interpretar as configurações de segurança do trecho e aplicar o método de segurança reverso capaz de decodificar aquela parte do fluxo recebido. Além dessa, muitas outras modificações podem ainda ser realizadas no sentido de ampliar o poder da linguagem proposta.

Apêndice A

DTD do Arquivo de Configuração

Abaixo, o DTD (*Document Type Definition*) que descreve o XML do Arquivo de Configuração utilizado pela camada de segurança.

```
<!ELEMENT protocol-description (declaration,transmission-model)>

<!ELEMENT declaration (security-definition, message-definition?)>

<!ELEMENT security-definition (global-configuration, configuration+)>

<!ELEMENT global-configuration (requirements*)>
<!ELEMENT configuration (ranges|requirements*)>
<!ATTLIST configuration name ID #REQUIRED>
<!ATTLIST configuration type (Composed|Single) "Single">
<!ATTLIST configuration unit (Byte|Percent) "Percent">

<!ELEMENT ranges (range|repeat)+>
<!ELEMENT repeat (range)+>
<!ATTLIST repeat count CDATA #REQUIRED>

<!ELEMENT range EMPTY>
<!ATTLIST range configuration IDREF #REQUIRED>
<!ATTLIST range size CDATA #REQUIRED>

<!ELEMENT requirements (authenticity|confidentiality|integrity|maximum-latency|
minimum-transmission-rate|maximum-byte-overhead-per-packet)*>

<!ELEMENT authenticity EMPTY>
<!ATTLIST authenticity priority (Low|Medium|High) "Medium">
<!ATTLIST authenticity level
(Disabled|Optional|Desired|Mandatory|Critical) #REQUIRED>
```

```
<!ELEMENT confidentiality EMPTY>
<!ATTLIST confidentiality priority (Low|Medium|High) "Medium">
<!ATTLIST confidentiality level
(Disabled|Optional|Desired|Mandatory|Critical) #REQUIRED>

<!ELEMENT integrity EMPTY>
<!ATTLIST integrity priority (Low|Medium|High) "Medium">
<!ATTLIST integrity level
(Disabled|Optional|Desired|Mandatory|Critical) #REQUIRED>

<!ELEMENT maximum-latency EMPTY>
<!ATTLIST maximum-latency priority (Low|Medium|High) "Medium">
<!ATTLIST maximum-latency time CDATA #REQUIRED>

<!ELEMENT minimum-transmission-rate EMPTY>
<!ATTLIST minimum-transmission-rate priority (Low|Medium|High) "Medium">
<!ATTLIST minimum-transmission-rate rate CDATA #REQUIRED>
<!ATTLIST minimum-transmission-rate time-unit
(Byte-Seconds|Byte-Microseconds|Byte-Milliseconds) #REQUIRED>

<!ELEMENT maximum-byte-overhead-per-packet EMPTY>
<!ATTLIST maximum-byte-overhead-per-packet priority (Low|Medium|High) "Medium">
<!ATTLIST maximum-byte-overhead-per-packet overhead CDATA #REQUIRED>

<!ELEMENT message-definition (message)*>
<!ELEMENT message (mask-comparator|size-comparator)+>
<!ATTLIST message name ID #REQUIRED>

<!ELEMENT mask-comparator EMPTY>
<!ATTLIST mask-comparator first-byte CDATA #REQUIRED>
<!ATTLIST mask-comparator match-mask CDATA #REQUIRED>
<!ATTLIST mask-comparator mask-size CDATA #REQUIRED>
<!ATTLIST mask-comparator match-value CDATA #REQUIRED>

<!ELEMENT size-comparator EMPTY>
<!ATTLIST size-comparator operator
(equal|less-than|less-or-equal|greater-than|greater-or-equal|different) #REQUIRED>
<!ATTLIST size-comparator value CDATA #REQUIRED>

<!ELEMENT transmission-model (send|receive|send-receive|loop|if)*>
```

```
<!ATTLIST transmission-model repeat (true|false) "true">

<!ELEMENT send EMPTY>
<!ATTLIST send label ID #IMPLIED>
<!ATTLIST send configuration IDREF #REQUIRED>
<!ATTLIST send size CDATA #IMPLIED>

<!ELEMENT receive EMPTY>
<!ATTLIST receive label ID #IMPLIED>
<!ATTLIST receive configuration IDREF #REQUIRED>
<!ATTLIST receive size CDATA #IMPLIED>

<!ELEMENT send-receive EMPTY>
<!ATTLIST send-receive label ID #IMPLIED>
<!ATTLIST send-receive configuration IDREF #REQUIRED>
<!ATTLIST send-receive size CDATA #IMPLIED>

<!ELEMENT if (if-condition, if-then, if-else?)>

<!ELEMENT if-condition (compare-message|receive-message|send-message)+>
<!ELEMENT if-then (send|receive|send-receive|loop|if)*>
<!ELEMENT if-else (send|receive|send-receive|loop|if)*>

<!ELEMENT receive-message EMPTY>
<!ATTLIST receive-message name IDREF #IMPLIED>

<!ELEMENT send-message EMPTY>
<!ATTLIST send-message name IDREF #IMPLIED>

<!ELEMENT loop (stop-condition?,(send|receive|send-receive|loop|if)*)>
<!ATTLIST loop count CDATA #IMPLIED>

<!ELEMENT stop-condition (compare-message|receive-message|send-message)+>

<!ELEMENT compare-message EMPTY>
<!ATTLIST compare-message label IDREF #REQUIRED>
<!ATTLIST compare-message name  IDREF #REQUIRED>
```

Apêndice B

Algoritmos utilizados nos experimentos

Abaixo, uma tabela com os algoritmos utilizados em cada experimento.

B.1 Variando a porcentagem de dados transmitidos com segurança

A Tabela B.1 mostra a lista de protocolos utilizados nesse experimento. O Protocolo 0 indica a transmissão sem nenhum algoritmo de segurança.

Nome	Protocolo
Protocolo 0	-
Protocolo 1	AES CBC 256, CMAC AES 256, DSA
Protocolo 6	AES CFB 256, CMAC AES 256, DSA

Tabela B.1: Experimento 1 - Lista de Protocolos utilizados no experimento

A Tabela B.2 mostra o número médio de transmissões por segundo com cada protocolo.

Porcentagem	Sem segurança	Protocolo 1	Protocolo 2
0%	2420,11	0	0
10%	2083,20	118,76	122,45
20%	1871,90	239,18	238,94
30%	1551,33	338,90	349,09
40%	1256,87	429,84	457,45
50%	1093,20	557,78	557,36
60%	852,18	657,09	656,48
70%	630,14	747,66	747,39
80%	371,65	763,42	839,48
90%	201,80	925,15	924,66
100%	0	977,75	1008,03

Tabela B.2: Experimento 1 - Número médio de transmissões por segundo com cada protocolo

B.2 Níveis de Segurança Variáveis

A Tabela B.3 mostra os protocolos utilizados nesse experimento.

Nome	Protocolo
P0	-
P1	AES CBC 256, CMAC AES 256, DSA
P2	DES ECB, MD2,RSA HASH 1024
P3	RC4 128, SHA1, RSA HASH 1024
P4	DES ECB, SHA1, DSA
P5	DES CFB, CMAC DES, DSA
P6	AES CFB 256, CMAC AES 256, DSA

Tabela B.3: Experimento 2 - Lista de Protocolos utilizados no experimento

A Tabela B.4, por sua vez, mostra o número médio de transmissões por segundo com cada protocolo.

Configuração	P0	P1	P2	P3	P4	P5	P6
Arquivo 0	2448,08	0	0	0	0	0	0
Arquivo 1	1077,06	1078,06	0	0	0	0	0
Arquivo 2	0	0	300	399	1494,17	99	0
Arquivo 3	0	0	300	767,57	600	468,48	0
Arquivo 4	992,71	600	0	0	0	0	600
Arquivo 5	1716,59	300	0	0	0	0	300
Arquivo 6	0	978,69	0	0	0	0	1007,87

Tabela B.4: Experimento 2 - Número médio de transmissões por segundo com cada protocolo

B.3 Variando o requisito de latência

A Tabela B.5 mostra os protocolos utilizados nesse experimento.

Nome	Protocolo
P0	-
P1	AES CBC 256, CMAC AES 256, DSA
P2	RC4 128, SHA1, RSA HASH 1024
P3	RSA 1024, CMAC AES 256, DSA
P4	RC4 128, HMAC MD5, DSA
P5	RSA 2048, CMAC AES 256, DSA

Tabela B.5: Experimento 3 - Lista de Protocolos utilizados no experimento

A Tabela B.6, por sua vez, mostra o número médio de transmissões por segundo com cada protocolo.

Configuração	P0	P1	P2	P3	P4	P5
Arquivo 0	2444,84	0	0	0	0	0
Arquivo 1	0	2422,84	0	0	0	0
Arquivo 2	0	0	1008,42	1009,06	0	0
Arquivo 3	0	0	0	864,39	949,06	0
Arquivo 4	0	0	0	0	762,75	785,72
Arquivo 5	0	0	0	28,94	0	27,97
Arquivo 6	0	0	0	0	738,12	784,91

Tabela B.6: Experimento 3 - Número médio de transmissões por segundo com cada protocolo

Apêndice C

Arquivos de Configuração utilizados nos experimentos

Neste apêndice serão abordados os arquivos de configuração utilizados nos experimentos realizados. As próximas seções mostram os arquivos de configuração agrupados de acordo com os experimentos realizados.

C.0.1 Variando a porcentagem de dados transmitidos com segurança

Esta seção mostra os arquivos de configuração utilizados no experimento de variação da porcentagem de dados transmitidos com segurança (Seção 5.1).

C.0.1.1 Transmissão com 0% de segurança

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE protocol-description SYSTEM "anotacao.dtd">
<protocol-description>
  <declaration>
    <security-definition>
      <global-configuration>
        <requirements>
          <authenticity level="Disabled" priority="Low" />
          <confidentiality level="Disabled"  priority="Low" />
          <integrity level="Disabled" priority="Low"  />
          <maximum-latency priority="Low" time = "0"/>
          <minimum-transmission-rate  priority="Low" rate = "0"
              timeout="0" time-unit="Microseconds"/>
          <maximum-byte-overhead-per-packet  priority="Low" overhead="0"/>
        </requirements>
      </global-configuration>
      <configuration name="high_security">
```

```
        <requirements>
          <authenticity    level="Critical" priority="High" />
          <confidentiality level="Critical"  priority="High" />
          <integrity level="Critical" priority="High"  />
          <maximum-latency priority="High" time = "8000"/>
        </requirements>
      </configuration>
      <configuration name="low_security" />
    </security-definition>
  </declaration>

  <transmission-model repeat="true">
    <send-receive configuration="low_security" />
  </transmission-model>
</protocol-description>
```

C.0.1.2 Transmissão com 10% de segurança

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE protocol-description SYSTEM "anotacao.dtd">
<protocol-description>
  <declaration>
    <security-definition>
      <global-configuration>
        <requirements>
          <authenticity level="Disabled" priority="Low" />
          <confidentiality level="Disabled"  priority="Low" />
          <integrity level="Disabled" priority="Low"  />
          <maximum-latency priority="Low" time = "0"/>
          <minimum-transmission-rate  priority="Low" rate = "0"
              timeout="0" time-unit="Microseconds"/>
          <maximum-byte-overhead-per-packet  priority="Low" overhead="0"/>
        </requirements>
      </global-configuration>
      <configuration name="high_security">
        <requirements>
          <authenticity    level="Critical" priority="High" />
          <confidentiality level="Critical"  priority="High" />
          <integrity level="Critical" priority="High"  />
          <maximum-latency priority="High" time = "8000"/>
        </requirements>
      </configuration>
```

```
      <configuration name="low_security" />
    </security-definition>
  </declaration>

  <transmission-model repeat="true">
    <loop count="10">
      <send-receive configuration="high_security" />
    </loop>
    <loop count="90">
      <send-receive configuration="low_security" />
    </loop>
  </transmission-model>
</protocol-description>
```

C.0.1.3 Transmissão com 20% de segurança

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE protocol-description SYSTEM "anotacao.dtd">
<protocol-description>
  <declaration>
    <security-definition>
      <global-configuration>
        <requirements>
          <authenticity level="Disabled" priority="Low" />
          <confidentiality level="Disabled"  priority="Low" />
          <integrity level="Disabled" priority="Low"  />
          <maximum-latency priority="Low" time = "0"/>
          <minimum-transmission-rate  priority="Low" rate = "0"
              timeout="0" time-unit="Microseconds"/>
          <maximum-byte-overhead-per-packet  priority="Low" overhead="0"/>
        </requirements>
      </global-configuration>
      <configuration name="high_security">
        <requirements>
          <authenticity    level="Critical" priority="High" />
          <confidentiality level="Critical"  priority="High" />
          <integrity level="Critical" priority="High"  />
          <maximum-latency priority="High" time = "8000"/>
        </requirements>
      </configuration>
      <configuration name="low_security" />
    </security-definition>
```

```
  </declaration>

  <transmission-model repeat="true">
    <loop count="20">
      <send-receive configuration="high_security" />
    </loop>
    <loop count="80">
      <send-receive configuration="low_security" />
    </loop>
  </transmission-model>
</protocol-description>
```

C.0.1.4 Transmissão com 30% de segurança

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE protocol-description SYSTEM "anotacao.dtd">
<protocol-description>
  <declaration>
    <security-definition>
      <global-configuration>
        <requirements>
          <authenticity level="Disabled" priority="Low" />
          <confidentiality level="Disabled"  priority="Low" />
          <integrity level="Disabled" priority="Low"  />
          <maximum-latency priority="Low" time = "0"/>
          <minimum-transmission-rate  priority="Low" rate = "0"
              timeout="0" time-unit="Microseconds"/>
          <maximum-byte-overhead-per-packet  priority="Low" overhead="0"/>
        </requirements>
      </global-configuration>
      <configuration name="high_security">
        <requirements>
          <authenticity    level="Critical" priority="High" />
          <confidentiality level="Critical"  priority="High" />
          <integrity level="Critical" priority="High"  />
          <maximum-latency priority="High" time = "8000"/>
        </requirements>
      </configuration>
      <configuration name="low_security" />
    </security-definition>
  </declaration>
```

```
  <transmission-model repeat="true">
    <loop count="30">
      <send-receive configuration="high_security" />
    </loop>
    <loop count="70">
      <send-receive configuration="low_security" />
    </loop>
  </transmission-model>
</protocol-description>
```

C.0.1.5 Transmissão com 40% de segurança

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE protocol-description SYSTEM "anotacao.dtd">
<protocol-description>
  <declaration>
    <security-definition>
      <global-configuration>
        <requirements>
          <authenticity level="Disabled" priority="Low" />
          <confidentiality level="Disabled"  priority="Low" />
          <integrity level="Disabled" priority="Low"  />
          <maximum-latency priority="Low" time = "0"/>
          <minimum-transmission-rate  priority="Low" rate = "0"
              timeout="0" time-unit="Microseconds"/>
          <maximum-byte-overhead-per-packet  priority="Low" overhead="0"/>
        </requirements>
      </global-configuration>
      <configuration name="high_security">
        <requirements>
          <authenticity    level="Critical" priority="High" />
          <confidentiality level="Critical"  priority="High" />
          <integrity level="Critical" priority="High"  />
          <maximum-latency priority="High" time = "8000"/>
        </requirements>
      </configuration>
      <configuration name="low_security" />
    </security-definition>
  </declaration>

  <transmission-model repeat="true">
    <loop count="40">
```

```
      <send-receive configuration="high_security" />
    </loop>
    <loop count="60">
      <send-receive configuration="low_security" />
    </loop>
  </transmission-model>
</protocol-description>
```

C.0.1.6 Transmissão com 50% de segurança

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE protocol-description SYSTEM "anotacao.dtd">
<protocol-description>
  <declaration>
    <security-definition>
      <global-configuration>
        <requirements>
          <authenticity level="Disabled" priority="Low" />
          <confidentiality level="Disabled"  priority="Low" />
          <integrity level="Disabled" priority="Low"  />
          <maximum-latency priority="Low" time = "0"/>
          <minimum-transmission-rate  priority="Low" rate = "0"
              timeout="0" time-unit="Microseconds"/>
          <maximum-byte-overhead-per-packet  priority="Low" overhead="0"/>
        </requirements>
      </global-configuration>
      <configuration name="high_security">
        <requirements>
          <authenticity    level="Critical" priority="High" />
          <confidentiality level="Critical"  priority="High" />
          <integrity level="Critical" priority="High"  />
          <maximum-latency priority="High" time = "8000"/>
        </requirements>
      </configuration>
      <configuration name="low_security" />
    </security-definition>
  </declaration>

  <transmission-model repeat="true">
    <loop count="50">
      <send-receive configuration="high_security" />
    </loop>
```

```
    <loop count="50">
      <send-receive configuration="low_security" />
    </loop>
  </transmission-model>
</protocol-description>
```

C.0.1.7 Transmissão com 60% de segurança

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE protocol-description SYSTEM "anotacao.dtd">
<protocol-description>
  <declaration>
    <security-definition>
      <global-configuration>
        <requirements>
          <authenticity level="Disabled" priority="Low" />
          <confidentiality level="Disabled"  priority="Low" />
          <integrity level="Disabled" priority="Low"  />
          <maximum-latency priority="Low" time = "0"/>
          <minimum-transmission-rate  priority="Low" rate = "0"
              timeout="0" time-unit="Microseconds"/>
          <maximum-byte-overhead-per-packet  priority="Low" overhead="0"/>
        </requirements>
      </global-configuration>
      <configuration name="high_security">
        <requirements>
          <authenticity    level="Critical" priority="High" />
          <confidentiality level="Critical"  priority="High" />
          <integrity level="Critical" priority="High"  />
          <maximum-latency priority="High" time = "8000"/>
        </requirements>
      </configuration>
      <configuration name="low_security" />
    </security-definition>
  </declaration>

  <transmission-model repeat="true">
    <loop count="60">
      <send-receive configuration="high_security" />
    </loop>
    <loop count="40">
      <send-receive configuration="low_security" />
```

```
    </loop>
  </transmission-model>
</protocol-description>
```

C.0.1.8 Transmissão com 70% de segurança

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE protocol-description SYSTEM "anotacao.dtd">
<protocol-description>
  <declaration>
    <security-definition>
      <global-configuration>
        <requirements>
          <authenticity level="Disabled" priority="Low" />
          <confidentiality level="Disabled"  priority="Low" />
          <integrity level="Disabled" priority="Low"  />
          <maximum-latency priority="Low" time = "0"/>
          <minimum-transmission-rate  priority="Low" rate = "0"
              timeout="0" time-unit="Microseconds"/>
          <maximum-byte-overhead-per-packet  priority="Low" overhead="0"/>
        </requirements>
      </global-configuration>
      <configuration name="high_security">
        <requirements>
          <authenticity    level="Critical" priority="High" />
          <confidentiality level="Critical"  priority="High" />
          <integrity level="Critical" priority="High"  />
          <maximum-latency priority="High" time = "8000"/>
        </requirements>
      </configuration>
      <configuration name="low_security" />
    </security-definition>
  </declaration>

  <transmission-model repeat="true">
    <loop count="70">
      <send-receive configuration="high_security" />
    </loop>
    <loop count="30">
      <send-receive configuration="low_security" />
    </loop>
  </transmission-model>
```

```
</protocol-description>
```

C.0.1.9 Transmissão com 80% de segurança

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE protocol-description SYSTEM "anotacao.dtd">
<protocol-description>
  <declaration>
    <security-definition>
      <global-configuration>
        <requirements>
          <authenticity level="Disabled" priority="Low" />
          <confidentiality level="Disabled"  priority="Low" />
          <integrity level="Disabled" priority="Low"  />
          <maximum-latency priority="Low" time = "0"/>
          <minimum-transmission-rate  priority="Low" rate = "0"
              timeout="0" time-unit="Microseconds"/>
          <maximum-byte-overhead-per-packet  priority="Low" overhead="0"/>
        </requirements>
      </global-configuration>
      <configuration name="high_security">
        <requirements>
          <authenticity    level="Critical" priority="High" />
          <confidentiality level="Critical"  priority="High" />
          <integrity level="Critical" priority="High"  />
          <maximum-latency priority="High" time = "8000"/>
        </requirements>
      </configuration>
      <configuration name="low_security" />
    </security-definition>
  </declaration>

  <transmission-model repeat="true">
    <loop count="80">
      <send-receive configuration="high_security" />
    </loop>
    <loop count="20">
      <send-receive configuration="low_security" />
    </loop>
  </transmission-model>
</protocol-description>
```

C.0.1.10 Transmissão com 90% de segurança

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE protocol-description SYSTEM "anotacao.dtd">
<protocol-description>
  <declaration>
    <security-definition>
      <global-configuration>
        <requirements>
          <authenticity level="Disabled" priority="Low" />
          <confidentiality level="Disabled"  priority="Low" />
          <integrity level="Disabled" priority="Low"  />
          <maximum-latency priority="Low" time = "0"/>
          <minimum-transmission-rate  priority="Low" rate = "0"
              timeout="0" time-unit="Microseconds"/>
          <maximum-byte-overhead-per-packet  priority="Low" overhead="0"/>
        </requirements>
      </global-configuration>
      <configuration name="high_security">
        <requirements>
          <authenticity    level="Critical" priority="High" />
          <confidentiality level="Critical"  priority="High" />
          <integrity level="Critical" priority="High"  />
          <maximum-latency priority="High" time = "8000"/>
        </requirements>
      </configuration>
      <configuration name="low_security" />
    </security-definition>
  </declaration>

  <transmission-model repeat="true">
    <loop count="90">
      <send-receive configuration="high_security" />
    </loop>
    <loop count="10">
      <send-receive configuration="low_security" />
    </loop>
  </transmission-model>
</protocol-description>
```

C.0.1.11 Transmissão com 100% de segurança

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE protocol-description SYSTEM "anotacao.dtd">
<protocol-description>
  <declaration>
    <security-definition>
      <global-configuration>
        <requirements>
          <authenticity level="Disabled" priority="Low" />
          <confidentiality level="Disabled"  priority="Low" />
          <integrity level="Disabled" priority="Low"  />
          <maximum-latency priority="Low" time = "0"/>
          <minimum-transmission-rate  priority="Low" rate = "0"
              timeout="0" time-unit="Microseconds"/>
          <maximum-byte-overhead-per-packet  priority="Low" overhead="0"/>
        </requirements>
      </global-configuration>
      <configuration name="high_security">
        <requirements>
          <authenticity    level="Critical" priority="High" />
          <confidentiality level="Critical"  priority="High" />
          <integrity level="Critical" priority="High"  />
          <maximum-latency priority="High" time = "8000"/>
        </requirements>
      </configuration>
      <configuration name="low_security" />
    </security-definition>
  </declaration>

  <transmission-model repeat="true">
      <send-receive configuration="high_security" />
  </transmission-model>
</protocol-description>
```

C.0.2 Níveis de segurança variáveis

As próximas seções contém os arquivos de configuração utilizados no experimento com diversos arquivos com níveis de configuração variáveis (Seção 5.2).

C.0.2.1 Arquivo 0 - Transmissão sem segurança

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
```

```
<!DOCTYPE protocol-description SYSTEM "anotacao.dtd">
<protocol-description>
  <declaration>
    <security-definition>
      <global-configuration>
        <requirements>
          <authenticity level="Disabled" priority="Low" />
          <confidentiality level="Disabled"  priority="Low" />
          <integrity level="Disabled" priority="Low"  />
          <maximum-latency priority="Low" time = "0"/>
          <minimum-transmission-rate  priority="Low" rate = "0"
              timeout="0" time-unit="Microseconds"/>
          <maximum-byte-overhead-per-packet  priority="Low" overhead="0" />
        </requirements>
      </global-configuration>
      <configuration name="high_security">
        <requirements>
          <authenticity    level="Critical" priority="High" />
          <confidentiality level="Critical" priority="High" />
          <integrity       level="Critical" priority="High"  />
        </requirements>
      </configuration>
      <configuration name="low_security" />
    </security-definition>
  </declaration>

  <transmission-model repeat="true">
    <send-receive configuration="low_security" />
  </transmission-model>
</protocol-description>
```

C.0.2.2 Arquivo 1 - Transmissão FTP

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE protocol-description SYSTEM "anotacao.dtd">
<protocol-description>
  <declaration>
    <security-definition>
      <global-configuration>
        <requirements>
          <authenticity level="Disabled" priority="Low" />
          <confidentiality level="Disabled"  priority="Low" />
```

```
          <integrity level="Disabled" priority="Low"  />
          <maximum-latency priority="Low" time = "0"/>
          <minimum-transmission-rate  priority="Low" rate = "0"
              timeout="0" time-unit="Microseconds"/>
          <maximum-byte-overhead-per-packet  priority="Low" overhead="0" />
        </requirements>
      </global-configuration>
      <configuration name="data_message">
        <requirements>
          <authenticity    level="Critical" priority="High" />
          <confidentiality level="Critical"  priority="High" />
          <integrity level="Critical" priority="High"  />
          <maximum-latency priority="High" time="8000" />
        </requirements>
      </configuration>
      <configuration name="ack_message" />
    </security-definition>
  </declaration>

  <transmission-model repeat="true">
    <send configuration="data_message" />
    <receive configuration="ack_message" />
  </transmission-model>
</protocol-description>
```

C.0.2.3 Arquivo 2 - Três níveis com tamanhos diferentes

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE protocol-description SYSTEM "anotacao.dtd">
<protocol-description>
  <declaration>
    <security-definition>
      <global-configuration>
        <requirements>
          <authenticity level="Disabled" priority="Low" />
          <confidentiality level="Disabled"  priority="Low" />
          <integrity level="Disabled" priority="Low"  />
          <maximum-latency priority="Low" time = "0"/>
          <minimum-transmission-rate  priority="Low" rate = "0"
            timeout="0" time-unit="Microseconds"/>
          <maximum-byte-overhead-per-packet  priority="Low" overhead="0" />
        </requirements>
```

```
      </global-configuration>
      <configuration name="important_data_message">
        <requirements>
          <authenticity    level="Critical" priority="High" />
          <confidentiality level="Critical"  priority="High" />
          <integrity level="Critical" priority="High"  />
          <maximum-latency priority="High" time = "8000" />
        </requirements>
      </configuration>
      <configuration name="unimportant_message" />
      <configuration name="some_information">
        <requirements>
          <integrity level="Optional" priority="Medium" />
          <confidentiality level="Optional"  priority="Medium" />
          <authenticity    level="Optional" priority="Medium" />
          <maximum-latency priority="High" time = "4000" />
        </requirements>
      </configuration>
    </security-definition>
  </declaration>

  <transmission-model repeat="true">
    <loop count="33">
      <send configuration="important_data_message" />
      <receive configuration="important_data_message" />
    </loop>
    <loop count="100">
      <send configuration="some_information" />
      <receive configuration="some_information" />
    </loop>
    <loop count="300">
      <send configuration="unimportant_message" />
      <receive configuration="unimportant_message" />
    </loop>
  </transmission-model>
</protocol-description>
```

C.0.2.4 Arquivo 3 - Três níveis com mesmo tamanho

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE protocol-description SYSTEM "anotacao.dtd">
<protocol-description>
```

```
<declaration>
  <security-definition>
    <global-configuration>
      <requirements>
        <authenticity level="Disabled" priority="Low" />
        <confidentiality level="Disabled"  priority="Low" />
        <integrity level="Disabled" priority="Low"  />
        <maximum-latency priority="Low" time = "0"/>
        <minimum-transmission-rate  priority="Low" rate = "0"
            timeout="0" time-unit="Microseconds"/>
        <maximum-byte-overhead-per-packet  priority="Low" overhead="0" />
      </requirements>
    </global-configuration>
    <configuration name="important_data_message">
      <requirements>
        <authenticity    level="Critical" priority="High" />
        <confidentiality level="Critical"  priority="High" />
        <integrity level="Critical" priority="High"  />
        <maximum-latency priority="High" time = "8000" />
      </requirements>
    </configuration>
    <configuration name="unimportant_message" />
    <configuration name="some_information">
      <requirements>
        <integrity level="Optional" priority="Medium" />
        <confidentiality level="Optional"  priority="Medium" />
        <authenticity    level="Optional" priority="Medium" />
        <maximum-latency priority="High" time = "4000" />
      </requirements>
    </configuration>
  </security-definition>
</declaration>

<transmission-model repeat="true">
  <loop count="300">
    <send configuration="important_data_message" />
    <receive configuration="important_data_message" />
  </loop>
  <loop count="300">
    <send configuration="some_information" />
    <receive configuration="some_information" />
```

```
    </loop>
    <loop count="300">
      <send configuration="unimportant_message" />
      <receive configuration="unimportant_message" />
    </loop>
  </transmission-model>
</protocol-description>
```

C.0.2.5 Arquivo 4 - Dois níveis com mesmo tamanho

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE protocol-description SYSTEM "anotacao.dtd">
<protocol-description>
  <declaration>
    <security-definition>
      <global-configuration>
        <requirements>
          <authenticity level="Disabled" priority="Low" />
          <confidentiality level="Disabled"  priority="Low" />
          <integrity level="Disabled" priority="Low"  />
          <maximum-latency priority="Low" time = "0"/>
          <minimum-transmission-rate  priority="Low" rate="0"
              timeout="0" time-unit="Microseconds"/>
          <maximum-byte-overhead-per-packet  priority="Low" overhead="0" />
        </requirements>
      </global-configuration>
      <configuration name="data_message">
        <requirements>
          <authenticity    level="Critical" priority="High" />
          <confidentiality level="Critical" priority="High" />
          <integrity       level="Critical" priority="High"  />
          <maximum-latency priority="High" time = "8000" />
        </requirements>
      </configuration>
      <configuration name="unimportant" />
    </security-definition>
  </declaration>

  <transmission-model repeat="true">
    <loop count="300">
      <send-receive configuration="data_message" />
    </loop>
```

```
    <loop count="300">
      <send-receive configuration="unimportant" />
    </loop>
  </transmission-model>
</protocol-description>
```

C.0.2.6 Arquivo 5 - Dois níveis de tamanhos diferentes

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE protocol-description SYSTEM "anotacao.dtd">
<protocol-description>
  <declaration>
    <security-definition>
      <global-configuration>
        <requirements>
          <authenticity level="Disabled" priority="Low" />
          <confidentiality level="Disabled"  priority="Low" />
          <integrity level="Disabled" priority="Low"  />
          <maximum-latency priority="Low" time = "0"/>
          <minimum-transmission-rate  priority="Low" rate = "0"
              timeout="0" time-unit="Microseconds"/>
          <maximum-byte-overhead-per-packet  priority="Low" overhead="0" />
        </requirements>
      </global-configuration>
      <configuration name="data_message">
        <requirements>
          <authenticity    level="Critical" priority="High" />
          <confidentiality level="Critical"  priority="High" />
          <integrity level="Critical" priority="High"  />
          <maximum-latency priority="High" time = "8000" />
        </requirements>
      </configuration>
      <configuration name="unimportant" />
    </security-definition>
  </declaration>

  <transmission-model repeat="true">
    <loop count="100">
      <send-receive configuration="data_message" />
    </loop>
    <loop count="300">
      <send-receive configuration="unimportant" />
```

```
    </loop>
  </transmission-model>
</protocol-description>
```

C.0.2.7 Arquivo 6 - Segurança Máxima

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE protocol-description SYSTEM "anotacao.dtd">
<protocol-description>
  <declaration>
    <security-definition>
      <global-configuration>
        <requirements>
          <authenticity level="Disabled" priority="Low" />
          <confidentiality level="Disabled"  priority="Low" />
          <integrity level="Disabled" priority="Low"  />
          <maximum-latency priority="Low" time = "0"/>
          <minimum-transmission-rate  priority="Low" rate = "0"
              timeout="0" time-unit="Microseconds"/>
          <maximum-byte-overhead-per-packet  priority="Low" overhead="0" />
        </requirements>
      </global-configuration>
      <configuration name="high_security">
        <requirements>
          <authenticity    level="Critical" priority="High" />
          <confidentiality level="Critical" priority="High" />
          <integrity       level="Critical" priority="High"  />
          <maximum-latency priority="High" time = "8000" />
        </requirements>
      </configuration>
      <configuration name="low_security" />
    </security-definition>
  </declaration>

  <transmission-model repeat="true">
      <send-receive configuration="high_security" />
  </transmission-model>
</protocol-description>
```

C.0.3 Variando o requisito de latência máxima

As próximas seções contém os arquivos de configuração utilizados no experimento de variação de latência (Seção 5.3).

C.0.3.1 Configuração 0 - Transmissão sem segurança

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE protocol-description SYSTEM "anotacao.dtd">
<protocol-description>
  <declaration>
    <security-definition>
      <global-configuration>
        <requirements>
          <authenticity level="Disabled" priority="Low" />
          <confidentiality level="Disabled"  priority="Low" />
          <integrity level="Disabled" priority="Low"  />
          <maximum-latency priority="Low" time = "0"/>
          <minimum-transmission-rate  priority="Low" rate="0"
              timeout="0" time-unit="Microseconds"/>
          <maximum-byte-overhead-per-packet  priority="Low" overhead="0" />
        </requirements>
      </global-configuration>
      <configuration name="high_security">
        <requirements>
          <authenticity    level="Critical" priority="High" />
          <confidentiality level="Critical"  priority="High" />
          <integrity level="Critical" priority="High"  />
        </requirements>
      </configuration>
      <configuration name="low_security" />
    </security-definition>
  </declaration>

  <transmission-model repeat="true">
      <send-receive configuration="low_security" />
  </transmission-model>
</protocol-description>
```

C.0.3.2 Configuração 1 - Latência máxima de 1,5 ms

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE protocol-description SYSTEM "anotacao.dtd">
```

```
<protocol-description>
  <declaration>
    <security-definition>
      <global-configuration>
        <requirements>
          <authenticity level="Disabled" priority="Low" />
          <confidentiality level="Disabled"  priority="Low" />
          <integrity level="Disabled" priority="Low"  />
          <maximum-latency priority="Low" time = "0"/>
          <minimum-transmission-rate  priority="Low" rate="0"
              timeout="0" time-unit="Microseconds"/>
          <maximum-byte-overhead-per-packet  priority="Low" overhead="0" />
        </requirements>
      </global-configuration>
      <configuration name="important_data_message">
        <requirements>
          <authenticity    level="Mandatory" priority="High" />
          <confidentiality level="Mandatory"  priority="High" />
          <integrity level="Desired" priority="High"  />
          <maximum-latency priority="High" time="3000"/>
        </requirements>
      </configuration>
      <configuration name="some_information">
        <requirements>
          <integrity level="Desired" priority="Medium" />
          <confidentiality level="Optional"  priority="Medium" />
          <authenticity    level="Optional" priority="Medium" />
          <maximum-latency priority="High" time="1500"/>
        </requirements>
      </configuration>
      <configuration name="unimportant_message" />
    </security-definition>
  </declaration>

  <transmission-model repeat="true">
      <send configuration="some_information" />
      <receive configuration="some_information" />
  </transmission-model>
</protocol-description>
```

C.0.3.3 Configuração 2 - Latência máxima de 8 ms

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE protocol-description SYSTEM "anotacao.dtd">
<protocol-description>
  <declaration>
    <security-definition>
      <global-configuration>
        <requirements>
          <authenticity level="Disabled" priority="Low" />
          <confidentiality level="Disabled"  priority="Low" />
          <integrity level="Disabled" priority="Low"  />
          <maximum-latency priority="Low" time = "0"/>
          <minimum-transmission-rate  priority="Low" rate="0"
              timeout="0" time-unit="Microseconds"/>
          <maximum-byte-overhead-per-packet  priority="Low" overhead="0" />
        </requirements>
      </global-configuration>
      <configuration name="important_data_message">
        <requirements>
          <authenticity    level="Mandatory" priority="High" />
          <confidentiality level="Mandatory"  priority="High" />
          <integrity level="Desired" priority="High"  />
          <maximum-latency priority="High" time="8000"/>
        </requirements>
      </configuration>
      <configuration name="some_information">
        <requirements>
          <integrity level="Desired" priority="Medium" />
          <confidentiality level="Optional"  priority="Medium" />
          <authenticity    level="Optional" priority="Medium" />
          <maximum-latency priority="High" time="15000"/>
        </requirements>
      </configuration>
      <configuration name="unimportant_message" />
    </security-definition>
  </declaration>

  <transmission-model repeat="true">
      <send configuration="important_data_message" />
      <receive configuration="important_data_message" />
```

```
  </transmission-model>
</protocol-description>
```

C.0.3.4 Configuração 3 - Latência máxima de 30 ms

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE protocol-description SYSTEM "anotacao.dtd">
<protocol-description>
  <declaration>
    <security-definition>
      <global-configuration>
        <requirements>
          <authenticity level="Disabled" priority="Low" />
          <confidentiality level="Disabled"  priority="Low" />
          <integrity level="Disabled" priority="Low"  />
          <maximum-latency priority="Low" time = "0"/>
          <minimum-transmission-rate  priority="Low" rate="0"
              timeout="0" time-unit="Microseconds"/>
          <maximum-byte-overhead-per-packet  priority="Low" overhead="0" />
        </requirements>
      </global-configuration>
      <configuration name="important_data_message">
        <requirements>
          <authenticity    level="Mandatory" priority="High" />
          <confidentiality level="Mandatory"  priority="High" />
          <integrity level="Desired" priority="High"  />
          <maximum-latency priority="High" time="30000"/>
        </requirements>
      </configuration>
      <configuration name="some_information">
        <requirements>
          <integrity level="Desired" priority="Medium" />
          <confidentiality level="Optional"  priority="Medium" />
          <authenticity    level="Optional" priority="Medium" />
          <maximum-latency priority="High" time="15000"/>
        </requirements>
      </configuration>
      <configuration name="unimportant_message" />
    </security-definition>
  </declaration>

  <transmission-model repeat="true">
```

```
    <send configuration="important_data_message" />
    <receive configuration="important_data_message" />
  </transmission-model>
</protocol-description>
```

C.0.3.5 Configuração 4 - Latência máxima de 600 ms

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE protocol-description SYSTEM "anotacao.dtd">
<protocol-description>
  <declaration>
    <security-definition>
      <global-configuration>
        <requirements>
          <authenticity level="Disabled" priority="Low" />
          <confidentiality level="Disabled"  priority="Low" />
          <integrity level="Disabled" priority="Low"  />
          <maximum-latency priority="Low" time = "0"/>
          <minimum-transmission-rate  priority="Low" rate="0"
              timeout="0" time-unit="Microseconds"/>
          <maximum-byte-overhead-per-packet  priority="Low" overhead="0" />
        </requirements>
      </global-configuration>
      <configuration name="important_data_message">
        <requirements>
          <authenticity    level="Mandatory" priority="High" />
          <confidentiality level="Mandatory"  priority="High" />
          <integrity level="Desired" priority="High"  />
          <maximum-latency priority="High" time="600000"/>
        </requirements>
      </configuration>
      <configuration name="some_information">
        <requirements>
          <integrity level="Desired" priority="Medium" />
          <confidentiality level="Optional"  priority="Medium" />
          <authenticity    level="Optional" priority="Medium" />
          <maximum-latency priority="High" time="300000"/>
        </requirements>
      </configuration>
      <configuration name="unimportant_message" />
    </security-definition>
  </declaration>
```

```
  <transmission-model repeat="true">
      <send configuration="important_data_message" />
      <receive configuration="important_data_message" />
  </transmission-model>
</protocol-description>
```

C.0.3.6 Configuração 5 - Transmissão com segurança máxima

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE protocol-description SYSTEM "anotacao.dtd">
<protocol-description>
  <declaration>
    <security-definition>
      <global-configuration>
        <requirements>
          <authenticity level="Disabled" priority="Low" />
          <confidentiality level="Disabled"  priority="Low" />
          <integrity level="Disabled" priority="Low"  />
          <maximum-latency priority="Low" time = "0"/>
          <minimum-transmission-rate  priority="Low" rate="0"
              timeout="0" time-unit="Microseconds"/>
          <maximum-byte-overhead-per-packet  priority="Low" overhead="0" />
        </requirements>
      </global-configuration>
      <configuration name="important_data_message">
        <requirements>
          <authenticity    level="Mandatory" priority="High" />
          <confidentiality level="Mandatory"  priority="High" />
          <integrity level="Desired" priority="High"  />
        </requirements>
      </configuration>
      <configuration name="some_information">
        <requirements>
          <integrity level="Desired" priority="Medium" />
          <confidentiality level="Optional"  priority="Medium" />
          <authenticity    level="Optional" priority="Medium" />
        </requirements>
      </configuration>
      <configuration name="unimportant_message" />
    </security-definition>
  </declaration>
```

```
  <transmission-model repeat="true">
    <send configuration="important_data_message" />
    <receive configuration="important_data_message" />
  </transmission-model>
</protocol-description>
```

Referências Bibliográficas

[1] Russ Housley; William Arbaugh. Security problems in 802.11-based networks. *Commun. ACM*, 46(5):31–34, 2003.

[2] Manel Guerrero Zapata; N. Asokan. Securing ad hoc routing protocols. In *WiSE '02: Proceedings of the 1st ACM workshop on Wireless security*, pages 1–10, New York, NY, USA, 2002. ACM.

[3] John Bellardo. 802.11 denial-of-service attacks: Real vulnerabilities and practical solutions. In *Proceedings of the USENIX Security Symposium*, Washington, DC, USA, 2003. USENIX.

[4] M. Bhardwaj, T. Garnett, and A.P. Chandrakasan. Upper bounds on the lifetime of sensor networks. *Communications, 2001. ICC 2001. IEEE International Conference on*, 3:785–790 vol.3, 2001.

[5] Pablo Brenner. A technical tutorial on the ieee 802.11 protocol. BreezeCOM Wireless Communications, 1997. 1999 Edition.

[6] Andrew T. Campbell. Mobiware: Qos-aware middleware for mobile multimedia communications. In *HPN '97: Proceedings of the IFIP TC6 seventh international conference on High performance netwoking VII*, pages 166–183, London, UK, UK, 1997. Chapman & Hall, Ltd.

[7] Vicente Falconi Campos. *Gerenciamento da rotina do trabalho do dia-a-dia*. Edg, 2002.

[8] H. Day, J.D.; Zimmermann. The osi reference model. *Proceedings of the IEEE*, 71(12):1334–1340, Dec. 1983.

[9] Myra Dideles. Bluetooth: a technical overview. *Crossroads*, 9(4):11–18, 2003.

[10] E.J.; Mingyan Liu Duarte-Melo. Analysis of energy consumption and lifetime of heterogeneous wireless sensor networks. *Global Telecommunications Conference, 2002. GLOBECOM '02. IEEE*, 1:21–25 vol.1, 17-21 Nov. 2002.

[11] Edward H. Freeman. Holistic information security: Iso 27001 and due care. *Information Security Journal: A Global Perspective*, 16(5):291–294, 2007.

[12] Zhi Fu, Shyhtsun Felix Wu, He Huang, Kung Loh, Fengmin Gong, Ilia Baldine, and Chong Xu. Ipsec/vpn security policy: Correctness, conflict detection, and resolution. In *POLICY '01: Proceedings of the International Workshop on Policies for Distributed Systems and Networks*, pages 39–56, London, UK, 2001. Springer-Verlag.

[13] Simson Garfinkel. *PGP: Pretty Good Privacy.* O'Reilly & Associates, Inc., Sebastopol, CA, USA, 1996.

[14] K.S.; Jaquith A. Geer, D. Jr.; Hoo. Information security: why the future belongs to the quants. *Security e Privacy Magazine, IEEE*, 1(4):24–32, July-Aug. 2003.

[15] S.B. Goldsmith, A.J.;Wicker. Design challenges for energy-constrained ad hoc wireless networks. *Wireless Communications, IEEE [see also IEEE Personal Communications]*, 9(4):8–27, Aug. 2002.

[16] J.A. Gutierrez, M. Naeve, E. Callaway, M. Bourgeois, V. Mitter, and B. Heile. Ieee 802.15.4: a developing standard for low-power low-cost wireless personal area networks. *Network, IEEE*, 15(5):12–19, Sep/Oct 2001.

[17] A. Iwata, Ching-Chuan Chiang, Guangyu Pei, M. Gerla, and Tsu-Wei Chen. Scalable routing strategies for ad hoc wireless networks. *Selected Areas in Communications, IEEE Journal on*, 17(8):1369–1379, Aug 1999.

[18] L. Jae-Hwan Chang; Tassiulas. Energy conserving routing in wireless ad-hoc networks. *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, 1:22–31 vol.1, 2000.

[19] L. Jae-Hwan Chang; Tassiulas. Maximum lifetime routing in wireless sensor networks. *Networking, IEEE/ACM Transactions on*, 12(4):609–619, Aug. 2004.

[20] J. Jonsson and B. Kaliski. Public-key cryptography standards (pkcs) #1: Rsa cryptography specification version 2.1.

[21] Merike Kaeo. *Designing Network Security, Second Edition.* Cisco Press, 2003.

[22] R. Kolic. Wireless usb brings greater convenience and mobility to devices. http://deviceforge.com/articles/AT9015145687.html, 2004.

[23] J. Kong, H. Luo, K. Xu, D. Gu, M. Gerla, and S. Lu. Adaptive security for multi-level ad-hoc networks, 2002.

[24] Yongguang Zhang; Wenke Lee. Intrusion detection in wireless ad-hoc networks. In *MobiCom '00: Proceedings of the 6th annual international conference on Mobile computing and networking*, pages 275–283, New York, NY, USA, 2000. ACM Press.

[25] Govindan Nair, Joey Chou, Tomasz Madejski, Krzysztof Perycz, David Putzolu, and Jerry Sydir. IEEE 802.16 medium access control and service provisioning. 8(3):213–228, August 2004.

[26] Hamdy S. Soliman; Mohammed Omari. Application of synchronous dynamic encryption system in mobile wireless domains. In *Q2SWinet '05: Proceedings of the 1st ACM international workshop on Quality of service & security in wireless and mobile networks*, pages 24–30, New York, NY, USA, 2005. ACM Press.

[27] Tom Karygiannis; Les Owens. Nist special publication 800-48: Draft - wireless network security 802.11, bluetooth, and handheld devices. computer security division, national institute of standards and technology. disponível por www em: http://csrc.nist.gov/publications/drafts.html, 2002.

[28] Yih-Chun Hu; Adrian Perrig. A survey of secure wireless ad hoc routing. *IEEE Security and Privacy*, 2(3):28–39, 2004.

[29] Charles P. Pfleeger; Shari Lawrence Pfleeger. *Security in Computing*. Prentice Hall PTR, 2002.

[30] Srivaths Ravi, Anand Raghunathan, and Nachiketh Potlapally. Securing wireless data: system architecture challenges. In *ISSS '02: Proceedings of the 15th international symposium on System Synthesis*, pages 195–200, New York, NY, USA, 2002. ACM Press.

[31] Ronald L. Rivest. The md4 message digest algorithm. In *CRYPTO '90: Proceedings of the 10th Annual International Cryptology Conference on Advances in Cryptology*, pages 303–311, London, UK, 1991. Springer-Verlag.

[32] Bruno Pontes Soares Rocha. Middleware de Segurança Adaptativo para Computação Móvel. Master's thesis, Departamento de Ciência da Computação, Universidade Federal de Minas Gerais, Belo Horizonte, Minas Gerais, Brasil, Dezembro 2007.

[33] S.F. Russell. Wireless network security for users. In *Information Technology: Coding and Computing, 2001. Proceedings. International Conference on*, pages 172–177, Las Vegas, NV, USA, 2001.

[34] G. Sharma, R. Mazumdar, and B. Shroff. Delay and capacity trade-offs in mobile ad hoc networks: A global perspective. *Networking, IEEE/ACM Transactions on*, 15(5):981–992, Oct. 2007.

[35] M.H. Sherif, A. Serhrouchni, A.Y. Gaid, and F. Farazmandnia. Set and ssl: electronic payments on the internet. *Computers and Communications, 1998. ISCC '98. Proceedings. Third IEEE Symposium on*, pages 353–358, 30 Jun-2 Jul 1998.

[36] Gunhce Kim; Dongkyoo Shin; Dongil Shin. Intellectual property management on mpeg-4 video for hand-held device and mobile video streaming service. *Consumer Electronics, IEEE Transactions on*, 51(1):139–143, Feb. 2005.

[37] Sasha Slijepcevic, Miodrag Potkonjak, Vlasios Tsiatsis, Scott Zimbeck, and Mani B. Srivastava. On communication security in wireless ad-hoc sensor networks. In *WETICE*

'02: Proceedings of the 11th IEEE International Workshops on Enabling Technologies, pages 139–144, Washington, DC, USA, 2002. IEEE Computer Society.

[38] William Stallings. *Cryptography and Network Security: Principles and Practice.* Pearson Education, 2002.

[39] James P. G. Sterbenz, Rajesh Krishnan, Regina Rosales Hain, Alden W. Jackson, David Levin, Ram Ramanathan, and John Zao. Survivable mobile wireless networks: issues, challenges, and research directions. In *WiSE '02: Proceedings of the 1st ACM workshop on Wireless security*, pages 31–40, New York, NY, USA, 2002. ACM.

[40] Joseph Williams. The ieee 802.11b security problem, part 1. *IT Professional*, 03(6):96, 91–95, 2001.

[41] Hao Yang, Haiyun Luo, Fan Ye, Songwu Lu, and Lixia Zhang. Security in mobile ad hoc networks: Challenges and solutions. wireless communications, ieee, 2004.

[42] Xinmiao Zhang and Keshab K. Parhi. High-speed vlsi architectures for the aes algorithm. *IEEE Trans. Very Large Scale Integr. Syst.*, 12(9):957–967, 2004.