UNIVERSIDADE FEDERAL DO MARANHÃO
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE ELETRICIDADE

Cláudio Manoel Pereira Aroucha

Mecanismo de Segurança Ciente do Contexto para Computação em Nuvem Móvel

São Luís - MA
2015

Cláudio Manoel Pereira Aroucha

Mecanismo de Segurança Ciente do Contexto para Computação em Nuvem Móvel

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia de Eletricidade da Universidade Federal do Maranhão como requisito parcial para a obtenção do grau de MESTRE em Engenharia de Eletricidade.

Orientador: Zair Abdelouahab

Doutor em Ciência da Computação – UFMA

São Luís - MA

2015

Aroucha, Cláudio Manoel Pereira

Mecanismo de Segurança Ciente do Contexto para Computação em Nuvem Móvel / Cláudio Manoel Pereira Aroucha. – São Luís - MA, 2015.

88 f.

Orientador: Zair Abdelouahab.

Impresso por computador (fotocópia).

Dissertação (Mestrado) – Universidade Federal do Maranhão, Programa de Pós-Graduação em Engenharia de Eletricidade. São Luís - MA, 2015.

1. Computação em Nuvem. 2. Computação Ciente do Contexto. 3. Segurança em Redes. 4. Computação Móvel I. Abdelouahab, Zair, orient. II. Título.

CDU 004.65

MECANISMO DE SEGURANÇA CIENTE DO CONTEXTO PARA COMPUTAÇÃO EM NUVEM MÓVEL

Cláudio Manoel Pereira Aroucha

Dissertação aprovada em 27 de maio de 2015.

Prof. Zair Abdelouahab, Ph.D.
(Orientador)

Prof. Rafael Fernandes Lopes, Dr.
(Membro da Banca Examinadora)

Prof. Denivaldo Cicero Pavão Lopes, Dr.
(Membro da Banca Examinadora)

Aos meus pais, irmãos, amigos e a toda minha família que, com muito carinho e apoio, não mediram esforços para que eu chegasse até esta etapa de minha vida.

Resumo

O uso e a popularidade dos dispositivos móveis vem crescendo vertiginosamente, bem como a computação em nuvem como instrumento para armazenar dados e serviços, tornando-se viável a união dessas duas tecnologias. Um dos maiores obstáculos é a questão da segurança e ergonomia do dispositivo móvel que necessita provê-lo ao trafegar dados sem haver degradação de seu desempenho. Este trabalho propõe então um mecanismo de segurança ciente do contexto para Computação em Nuvem Móvel. Em adição, utiliza-se a tecnologia *Transport Layer Security* (TLS) para criar um canal seguro de envio de dados entre o cliente do dispositivo móvel e o servidor da Nuvem, autenticando-o por meio de criptografia de chave pública. Paralelamente, para prover um nível de segurança apropriado, analisa-se o contexto do dispositivo móvel a partir da lógica *Fuzzy*. O impacto do mecanismo no desempenho do dispositivo móvel foi medido por meio de experimentos.

Palavras-chaves: Computação em Nuvem, Computação Ciente do Contexto, Segurança em Redes, Computação Móvel.

Abstract

The use and popularity of mobile devices has been growing vertiginously, as well an the cloud computing as instrument to store data and services, making feasible the combination of these two technologies. One of the biggest obstacles is the mobile device's security and ergonomics to provide security for transporting its data without performance degradation. This work proposes a context-aware security mechanism for mobile devices in cloud computing. In addition, it is used the TLS technology to create a secure channel of data sent between mobile device's client and the cloud server, authenticating them via public key cryptography. At the same time, it provides an appropriate level of security, and analyses the context of the mobile device from the Fuzzy logic. The impact of the mechanism on the mobile device's performance were measured through experiments.

Keywords: Cloud Computing, Context Aware Computing, Network Security, Mobile Computing.

Agradecimentos

A Deus primeiramente por iluminar e guiar minha vida nesta empreitada, a meus pais Manoel de Jesus e Leila Raquel, as pessoas que me geraram e criaram contra todas adversidades da minha vida conduzindo-me a ser um homem integro que sou. Por sempre enfatizarem e apoiarem meus estudos como algo essencial.

Ao meu orientador Zair Abdelouhab por ter paciência e sabedoria a orientar-me nesta dissertação, prometendo-me um grande trabalho e aprendizado frente desta tarefa.

A minha namorada Letícia Rabelo, por me incentivar e apoiar nos momentos difíceis da produção desta dissertação que não foram poucos.

Aos meus colegas de laboratório que ajudaram-me nas correções ortográficas da dissertação e traduções de artigos.

Ao Willian por me auxiliar na produção deste mecanismo.

A todos que de alguma forma contribuíram com o meu trabalho e que certamente não serão esquecidos, meu sincero obrigado a todos.

"Maior que a tristeza de não haver vencido é a vergonha de não ter lutado."

Rui Barbosa

Lista de Figuras

Lista de Tabelas

Lista de Siglas

AES	*Advanced Encryption Standard*.
DES	*Data Encryption Standard*.
IaaS	*Infrastructure as a Service* .
JCE	*Java Cryptography Extension* .
JSSE	*JAVA SECURE SOCKET EXTENSION*.
MAC	Códigos de Autenticação de Mensagem.
MD5	*Message-Digest algorithm 5*.
NIST	*National Institute of Standards and Technology*.
OSI	*Open System Interconnection*.
PaaS	*Plataform as a Service* .
RSA	*Rivest, Shamir e Adleman*.
SaaS	*Software as a Service* .
SHA-1	*Secure Hash Algorithm 1*.
SSACC	Serviço de Segurança para Autenticação Ciente do Contexto: para Dispositivos Móveis no Paradigma da Computação em Nuvem.
TLS	*Transport Layer Security* .

Sumário

1 Introdução

Esta dissertação apresenta um mecanismo de segurança ciente do contexto para Computação em Nuvem Móvel, bem como uma ferramenta que auxilia no desenvolvimento deste mecanismo. Essa última realiza análises de recursos computacionais ao utilizar os algoritmos criptográficos em dispositivos móveis.

Este capítulo apresenta uma visão geral do trabalho. A seção 1.1 caracteriza a motivação e a justificativa para o desenvolvimento desta dissertação. A seção 1.2 expõe a problemática do trabalho. Descrevem-se os objetivos gerais e específicos na seção 1.3 e, por último, a seção 1.4 apresenta a estrutura na qual a dissertação está organizada.

1.1 Motivação e Justificativa

O surgimento dos dispositivos móveis, bem como sua grande aceitação e procura pela população, despertaram a necessidade do desenvolvimento de aplicações que satisfaçam as necessidades dos seus usuários. Entretanto, os recursos computacionais dos dispositivos móveis são escassos baixo poder de processamento de dados, bateria limitada e escasso limite de armazenamento, Satyanarayanan [46].

Para contornar esta limitação utiliza-se a Computação em Nuvem, Alzain [2], Rahimi [42], que fornece os serviços de computação, *software* e armazenamento que não exigem o conhecimento do usuário final da localização física e configuração do sistema, proporcionando uma redução dos gastos computacionais do dispositivo móvel. Segundo o *National Institute of Standards and Technology* (NIST), Computação em Nuvem é um modelo para permitir o acesso à rede de forma conveniente, ubíqua e sob demanda a um conjunto compartilhado de recursos computacionais configuráveis (por exemplo, redes, servidores, armazenamento, aplicações e serviços) que podem ser rapidamente provisionados e liberados com um mínimo de esforço de gerenciamento ou interação com o provedor de serviços, Mell [30].

O provedor de serviços da Nuvem por ser proprietário do ambiente fica responsável pela manutenção, atualização, armazenamento, etc. Segundo o autor Rong [45], é possível realizar uma analogia aos atuais sistemas de eletricidade e água corrente, em que os usuários finais podem obter as facilidades dos provedores de serviço da Nuvem sem se preocupar com a complexidade técnica por trás destes sistemas.

A aplicação no dispositivo móvel utilizará o servidor no ambiente de Computação em Nuvem para resolver o limite de armazenamento que possui. Para garantir que os dados trafegados entre a aplicação e o servidor sejam realizados de forma segura utiliza-se a criptografia e a autenticação. A criptografia será utilizada para codificar as mensagens enviadas do dispositivo móvel ao servidor da Nuvem de forma a garantir confidencialidade e a autenticação na sua comunicação.

A segurança dos dados dos dispositivo móveis é convencionalmente fornecida pelo uso de um único algoritmo de criptografia. Dessa forma utilizar um mecanismo que selecione diversos algoritmos que forneçam diferentes de níveis de segurança e adaptá-los aos mais diferentes contextos que os dispositivos móveis se encontram poderia proporcionar uma redução no uso de recursos. Mecanismos adaptativos devem evoluir constantemente às mudanças de contextos que ocorrem no dispositivo móvel. Visto que a complexidade de cada algoritmo está ligada diretamente ao seu consumo de recursos, um mecanismo que forneça economia destes é necessário. Nesse contexto as informações contextuais podem permitir uma melhor análise do estado do dispositivo para a adaptação dinâmica do seu nível de segurança.

Em resumo sabe-se que o dispositivo móvel possui capacidades limitadas, então levanta-se a questão de como fornecer a segurança sem degradar os seus recursos computacionais. Neste trabalho serão utilizados os algoritmos criptográficos do TLS, além de recurso de computação ciente de contexto, discutido pela primeira vez em Schilit [48], como um software que se adapta de acordo com sua localização de uso, o conjunto de pessoas e objetos próximos, assim como as alterações a esses objetos ao longo do tempo. Nesse caso, a segurança se adaptaria às necessidades do dispositivo móvel.

1.2 Problemática

Hoje os *softwares* dos dispositivos móveis estão sendo movidos para a Nuvem Alzain [2], Modi [31] que continuam sendo fornecidas aos seus usuários como serviços, por ser uma nova solução para sua capacidade limitada, como baixo poder de processamento de dados, bateria limitada e limite de armazenamento Satyanarayanan [46] como citado na seção anterior.

Visto que a comunicação entre a aplicação do dispositivo móvel com o servidor em Nuvem será realizado pela internet verifica-se a necessidade de fornecer segurança a eles. Segundo Kurose [27] e Soares [49], a segurança é uma característica essencial, por haver uma frequência nos números de ataques cibernéticos e protocolos inseguros, que deva ser realizada tanto em computadores tradicionais quanto servidores no ambiente de Computação em Nuvem, garantindo certas propriedades como: confidencialidade, autenticação, integridade e não-repudiação de mensagem, disponibilidade e controle de acesso.

Entretanto, surgiu a necessidade de criar um mecanismo que fornecesse esta segurança de forma ciente de contexto. A tentativa é reduzir os gastos de recursos computacionais ao máximo do dispositivo móvel sem a percepção do usuário da aplicação.

1.3 Objetivos Gerais e Específicos

Esta dissertação propõe no contexto de mecanismo de segurança ciente do contexto para Computação em Nuvem Móvel, bem como foram avaliados os algoritmos criptográficos para dispositivo móvel, que auxiliarão na mensuração dos recursos gastos para realizar a criptografia de dados. Este mecanismo deve ser capaz de identificar o contexto do dispositivo móvel e o tipo de rede que está conectado.

A partir das técnicas que serão utilizadas no decorrer da proposta, este mecanismo será capaz de realizar a segurança e assegurar a ergonomia do dispositivo móvel, sobretudo quando houver uma mudança de contexto do dispositivo móvel. O mecanismo é composto por um repositório de algoritmos criptográficos, um analisador de contexto do dispositivo móvel (avaliando memória, CPU e bateria) e pelo adaptador

de contexto baseado em *Fuzzy* que possui as regras para a tomada de decisão no cliente móvel. Por meio disto é possível: informar o tipo de criptografia que será utilizada no cliente do dispositivo móvel ao servidor em Nuvem; identificar a mudança de contexto no dispositivo móvel e realizar uma nova tomada de decisão (escolha do melhor algoritmo de criptografia para a nova situação).

Esta dissertação tem por objetivos específicos:

1. Apresentar uma avaliador de algoritmos criptográficos para dispositivos móveis. Para auxiliar na tomada de decisão do adaptador de contexto;

2. Apresentar um protótipo do mecanismo de segurança ciente do contexto por meio dos componentes analisador de contexto baseado em *Fuzzy*; repositório de algoritmos criptográficos, adaptador de contexto (elemento principal do mecanismo, responsável pela análise e tomada de decisão da melhor criptografia para a situação) e o componente de comunicação que informará o servidor em Nuvem da criptografia escolhida. Estes componentes irão analisar e prover a criptografia, segurança aos dados enviados do dispositivo móvel ao servidor em Nuvem;

3. Apresentar o resultado da avaliação, por meio de testes experimentais (e simulações computacionais).

1.4 Estrutura da Dissertação

Esta dissertação encontra-se organizada em 6 capítulos, descritos a seguir:

No capítulo 2, uma visão geral sobre computação verde, computação em nuvem, computação móvel em nuvem e computação ciente do contexto destacando suas características e definições, e em seguida, dando enfoque à segurança em redes, considerada neste trabalho. Por fim, os trabalhos relacionados sobre mecanismos de segurança ciente do contexto para dispositivos móveis são discutidos nessa seção.

O capítulo 3 apresenta, o mecanismo proposto. A arquitetura, funcionamento e todos os componentes são mostrados.

O capítulo 4 apresenta, as implementações e os resultados da dissertação.

Por fim o capítulo 5, apresenta as conclusões obtidas no desenvolvimento deste trabalho, possíveis trabalhos futuros. As considerações finais desta dissertação são apresentadas nesta seção.

2 Fundamentação Teórica

Neste capítulo são apresentados os conceitos fundamentais das tecnologias que servira de base para o desenvolvimento do projeto. O entendimento de cada tecnologia foi essencial para melhor integrá-las. A integração destas tecnologias é o alicerce do desenvolvimento dessa proposta.

2.1 Computação Verde

De acordo com Murugesan [32], Computação Verde ou Tecnologia de Informação Verde é o estudo, prática de desenvolvimento e uso de computadores, servidores, equipamentos eletrônicos e sistemas de comunicação, com o intuito de utilizar eficientemente os recursos com zero ou o minimo de impacto ao ambiente. Essa tecnologia também visa encorajar a melhora da eficiência da energia, do reuso e reciclagem de materiais menos prejudiciais para o benefício do ambiente. Com base nesse contexto foram criadas normas, padrões e políticas para a economia de produtos elétricos e eletrônicos, como os selos *Energy Stars* e o Procel.

Segundo Cirqueira [10], *Energy Stars* é um selo criado em 1992 pela Agência de Proteção Ambiental dos Estados Unidos para identificar aparelhos eletrônicos que possuíam eficiência energética. Popularizou-se e hoje está presente na maioria dos equipamentos de tecnologia da informação.

No Brasil um selo semelhante fora criado em 1993, o Procel Eletrobras de Economia de Energia, ou selo PROCEL. Ele tem como objetivo orientar os consumidores no ato da compra de equipamentos elétricos e eletrônicos a identificarem aqueles que possuem melhores níveis de eficiência energética dentro de cada categoria, para que haja uma economia de energia elétrica Piragibe [36].

Existem políticas para o uso da computação Verde. Um exemplo é o plano de ações para a tecnologia mais verde utilizado pela Dinamarca, no qual incluem-se quatro iniciativas para reduzir o impacto ambiental nas tecnologias de informação e comunicação Reimsbach [43]:

- Reduzir o impacto ambiental com o uso das tecnologias de informação e comunicação Ecológicas para as empresas.
- Promover o uso das tecnologias de informação verde aos mais jovens por meio de campanhas de conscientização.
- Orientar as autoridades públicas ao seu uso consciente.
- Fornecer o conhecimento básico para o cálculo de energia e CO2.

A computação Verde está presente também na otimização de aplicações que requerem recursos computacionais Cirqueira [10]. No caso de dispositivos móveis que possuem recursos limitados utilizar a computação Verde é de grande importância para o seu funcionamento. A bateria do dispositivo móvel é o principal recurso a ser economizado, visto a sua limitação e a necessidade sempre recarregá-la. Encontrar formas de prolongar o tempo de vida útil da bateria é um dos maiores ganhos que a computação Verde pode proporcionar.

2.2 Computação em Nuvem

De acordo com a definição do NIST, Computação em Nuvem é um modelo para permitir o acesso à rede conveniente, ubíquo e sob demanda a um conjunto compartilhado de recursos computacionais configuráveis (por exemplo, redes, servidores, armazenamento, aplicações e serviços) que podem ser rapidamente provisionados e liberados com um mínimo de esforço de gerenciamento ou interação com o provedor de serviços Mell [30].

Segundo Pedrosa [35], a palavra nuvem sugere a ideia de um ambiente desconhecido, em que todo o ambiente de infra-estrutura e recursos computacionais ficam "escondidos"do usuário, e este tendo apenas acesso a uma interface padrão por meio da qual é disponibilizado todo o conjunto de variadas aplicações e serviços.

A Computação em Nuvem é a próxima grande ruptura tecnológica, comparada e igualada à Revolução Industrial Kshetri [25]. Segundo Rahimi [42], esta funciona como uma utilidade água, luz, etc, pelo qual o usuário paga somente por aquilo que utiliza, nesse caso os serviços que a nuvem provê.

De acordo com Armbrust [5], a Computação em Nuvem refere-se tanto a aplicações entregues como serviços através da Internet e o hardware e software de sistemas nos data centers que fornecem esses serviços. Segundo Fernando [16], a virtualização dos recursos é o requisito fundamental para tornar a nuvem escalável, criando a ilusão de recursos computacionais infinitos ao usuário da nuvem.

A visão do NIST divide o modelo de Computação em Nuvem em cinco características essenciais, três modelos de serviço e quatro modelos de implantação, apresentado na figura 2.1 que será descrito nas seções seguintes.

Figura 2.1: Modelo visual da definição de Computação em Nuvem, Moraes [14]

2.2.1 Características Essenciais

Conforme a figura 2.1 as características essenciais definidas segundo o NIST são:

- **Amplo Acesso à Rede:** os recursos são disponibilizados pela rede e acessados por meio de mecanismos padronizados que promovem o uso de plataformas clientes heterogêneas (por exemplo, celulares, *tablets*, *notebooks* e estações de trabalho).

- **Elasticidade Rápida:** recursos podem ser elasticamente provisionados e liberados, em alguns casos automaticamente, caso haja a necessidade de escalar com o aumento da demanda, e liberados, na retração dessa demanda. Para o consumidor, as capacidades disponíveis para provisionamento frequentemente

parecem ser ilimitadas e podem ser apropriadas em qualquer quantidade e a qualquer momento.

- **Serviço Mensurado:** sistemas em nuvem automaticamente controlam e otimizam o uso dos recursos por meio de uma capacidade de medição em algum nível apropriado de abstração para o tipo de serviço (por exemplo, armazenamento, processamento, largura de banda e contas de usuários ativos). O uso de recursos pode ser monitorado, controlado e reportado, fornecendo transparência tanto ao provedor quanto ao consumidor do serviço utilizado.

- ***Self-Service* sob demanda:** um consumidor pode adquirir unilateralmente recursos computacionais, tais como tempo de servidor e armazenamento na rede, na medida que necessite e sem necessidade de interação humana com os provedores de cada serviço.

- **Conjunto de Recursos:** os recursos computacionais dos provedores de serviço são reunidos para servir a vários consumidores usando um modelo *multi-tenant* ou multi-inquilino, com diferentes recursos físicos e virtuais dinamicamente alocados e realocados de acordo com a demanda do usuário. Há um senso de independência de localização em que o cliente geralmente não tem o controle ou conhecimento sobre a exata localização dos recursos disponibilizados, mas podendo ser capaz de especificar o local em um nível mais alto de abstração (por exemplo, país, estado, ou *data center*). Exemplos de recursos incluem o armazenamento, processamento, memória e largura de banda de rede.

2.2.2 Modelos de Serviços

O ambiente da Computação em Nuvem é dividido em três modelos de serviços. A Figura 2.2 exibe estes modelos e a definição sobre estes, segundo o NIST.

- ***Infrastructure as a Service* (IaaS):** a capacidade fornecida ao consumidor é a provisão de processamento, armazenamento, redes, e outros recursos computacionais fundamentais a partir dos quais o consumidor é capaz de implantar e executar software arbitrariamente, que pode incluir sistemas

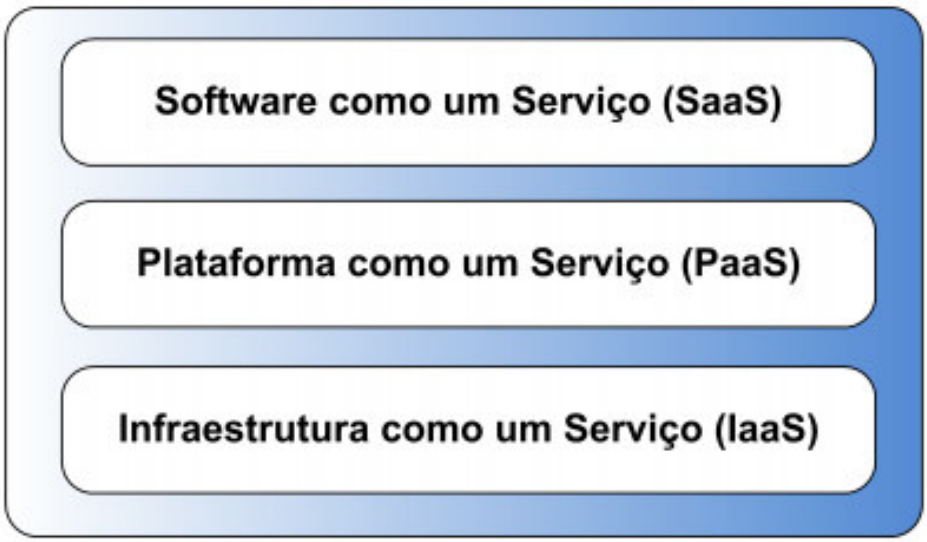

Figura 2.2: Modelos de Serviços, Sousa [50]

operacionais e aplicações. O consumidor não gerencia ou controla a infraestrutura subjacente da nuvem mas tem o controle sobre os sistemas operacionais, armazenamento, e aplicações implantadas; e possivelmente controle limitado a selecionar componentes de rede (por exemplo, *firewalls* do host).

- ***Plataform as a Service* (PaaS):** a capacidade fornecida ao consumidor é permitir implementar na infraestrutura da nuvem aplicações utilizando linguagens de programação, bibliotecas, serviços e ferramentas suportadas pelo provedor. O consumidor não gerencia ou controla a infraestrutura subjacente da nuvem incluindo rede, servidores, sistemas operacionais, ou armazenamento, mas tem controle sobre as aplicações implantadas e possivelmente definições de configuração do ambiente de hospedagem das aplicações.

- ***Software as a Service* (SaaS):** a capacidade fornecida ao consumidor é usar as aplicações do provedor executando na infraestrutura da nuvem. As aplicações são acessíveis a partir de vários dispositivos clientes através de uma interface cliente *thin*, tal como um navegador *web* (por exemplo, email baseado na web), ou uma *interface* de programa. O consumidor não gerencia ou controla a infra-estrutura subjacente da nuvem incluindo rede, servidores, sistemas operacionais, armazenamento, ou até mesmo recursos de aplicativos individuais, com a possível exceção das definições de configuração limitadas dos aplicativos

específicos do usuário. Como exemplo desse tipo de *softwares* como serviço temos:

- Google Apps (o mais utilizado);
- SalesForce (SFDC);
- NetSuite;
- Oracle;
- Microsoft.

2.2.3 Modelos de Implantação

De acordo com Buyya [8], embora a Computação em Nuvem surgisse a partir de utilitários de computação pública, outros modelos de implantação foram adotados. Portanto, independente de sua classe de serviço, a nuvem pode ser classificada em pública, privada, comunitária e híbrida.

- **Nuvem Pública:** é de propriedade de um provedor de serviços e os seus recursos vendidos ao público, Rong [45]. Segundo o NIST, esta pode ser de propriedade, gerenciada, e operada por uma empresa, instituição acadêmica ou organização governamental.

- **Nuvem Privada:** este termo refere-se a data centers de uma empresa ou organização que não está disponível para o público, Fox [17]. O autor Zhang [54] cita que a nuvem privada oferece maior grau de controle sobre desempenho, confiabilidade, e segurança, mas que é frequentemente criticada por ser similar à fazendas de servidores proprietários tradicionais.

- **Nuvem Comunitária:** é definido por Sousa [50], como uma nuvem que é compartilhada por várias empresas, que é suportada por uma comunidade específica que partilhou de seus interesses, missão, requisitos de segurança, política e sobre considerações de flexibilidade.

- **Nuvem Híbrida:** é definido pelo NIST como uma combinação de duas ou mais infraestruturas de nuvem distintas (privada, comunitária ou pública) que

permanecem entidades únicas, mas são unidas por tecnologia padronizada ou proprietária que permite a portabilidade de dados e aplicativo (por exemplo, uma nuvem estourando para balancear a carga entre as nuvens).

2.3 Computação Móvel em Nuvem

A computação móvel em nuvem é a combinação da Computação em Nuvem com a computação Móvel. Com o crescimento das aplicações que exigem mais recursos computacionais que o dispositivo móvel pode suportar tornou a computação em Nuvem em solução para este problema Dev [13] Popa [38] Dai [12].

A Figura 2.3 mostra a arquitetura da computação Móvel em Nuvem, que é dividida no ambiente de nuvem, com as maquinas virtuais que a compõe, interligada a um ponto de acesso que comunica os seus serviços aos aplicativos que localizam-se nos dispositivos móveis. Os modelos de aplicações para dispositivos móveis em nuvem são divididos em cliente, cliente/nuvem e nuvem.

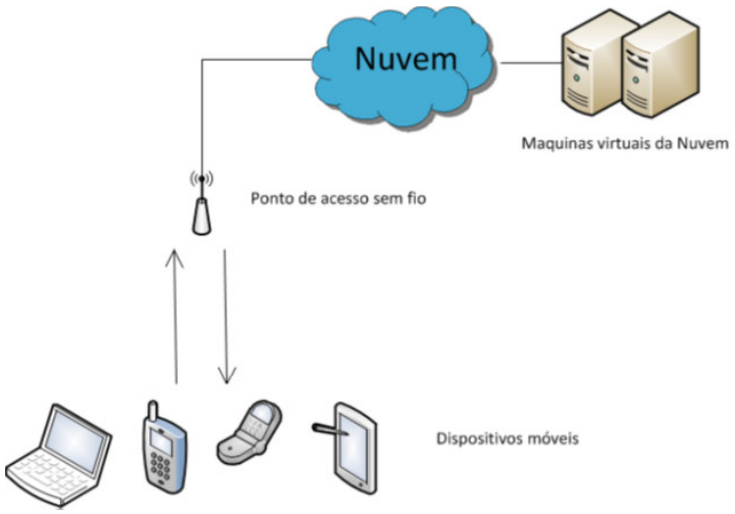

Figura 2.3: Arquitetura da computação Móvel em Nuvem

2.3.1 Modelos de aplicações para dispositivos móveis em nuvem

Segundo Popa [38], tem havido vários estudos sobre modelos de aplicações para dispositivos móveis em nuvem. Estes podem ser classificados em três categorias a seguir:

- **Modelo Cliente:** o dispositivo móvel é visto como uma forma mais conveniente de acessar os serviços da nuvem.

- **Modelo Cliente/Nuvem:** é dividido em aplicações em componentes e distribuídas entre os dispositivos móveis e a nuvem. Este modelo utiliza técnicas da nuvem como execução aumentada, elasticidade e mobilidade para superar as limitações dos dispositivos móveis.

- **Modelo Nuvem:** considera o dispositivo móvel como parte integral da Nuvem. O seu objetivo é fornecer uma infraestrutura distribuída de armazenamento e computação aos dispositivos, para que possa suportar novas aplicações.

Os modelos de aplicações para nuvens móveis se baseiam no modelo de serviços da nuvem que incluem PaaS, IaaS e SaaS Khan [24].

2.3.2 Vantagens e Desvantagens

Os autores Dai [12] e Dev [13], citam as vantagens do uso da computação Móvel em Nuvem como:

- **Estender o tempo de vida da bateria:** é uma das grandes preocupações para os dispositivos móveis. Soluções para reduzir o consumo de bateria requerem mudanças no *hardware* aumentando o seu custo ou sendo inviável para alguns dispositivos móveis. A técnica de descarregamento *offloadding* é proposta com o objetivo de migrar as computações complexas do dispositivo, que possui recursos limitados, para máquinas mais robustas (ou seja, servidores em nuvem).

- **Conveniência do compartilhamento de dados:** os dados são armazenados nos servidores finais da nuvem, permitindo o acesso conveniente. A largura de banda que o dispositivo móvel tem acesso é o que indica o quão fácil seria trabalhar com estes dados em nuvem.

- **Eliminação da regionalidade:** a computação Móvel em Nuvem torna possível o acesso das pessoas ao recursos que querem, a qualquer hora e lugar a partir da Internet móvel. Pode gerenciar o uso dos recursos em tempo real e realocar os recursos, quando for necessário.

- **Dispositivos móveis são independentes dos sistemas da computação Móvel em Nuvem:** as computações são migradas para os servidores finais em nuvem, não havendo necessidade de dispositivos móveis, tornando portável seu uso a qualquer dispositivo seja *smartphone* ou não.

- **Melhorar a confiabilidade:** os dispositivos móveis ao armazenarem os dados ou executarem os aplicativos em nuvem, apresenta uma melhora na sua confiabilidade, pois caso haja a perda de dados e aplicativos, estes podem ser recuperados.

- **Forte capacidade adaptativa:** a computação Móvel em Nuvem possui a capacidade de gerenciar vários tipos de cargas de trabalho, operações em *back-end* e aplicações interativas, além de ter redundância, auto-cura e modelo de programação escalável.

As desvantagens da computação Móvel em Nuvem de acordo com Guan [23] Qi [40] Qureshi [41] são:

- **Qualidade de comunicação:** nos dispositivos móveis, a capacidade de banda larga e a conectividade de rede são pobres. Os usuários dos dispositivos móveis, por se locomoverem e a rede sem fio ser descontínua impacta diretamente em alterações na banda larga e cobertura de rede.

- **Limitações dos dispositivos móveis:** possuem recursos limitados como CPU, memória, armazenamento, bateria e comunicação. O problema persiste na computação Móvel em Nuvem, pois a complexidade para realizar o desenvolvimento de aplicativos com estas limitações é muito alto.

- **Segurança e privacidade:** os dispositivos móveis, por possuírem limitado poder de processamento, muitas vezes ficam impedidos de utilizar algoritmos de segurança mais sofisticados. Por haver inúmeros tipos de dispositivos, torna-se difícil aplicar um mecanismo de proteção padronizado para que proteja a nuvem de ataques.

2.4 Computação Ciente do Contexto

Segundo Abowd [1], o contexto primeiramente é qualquer informação que possa caracterizar uma entidade, em que esta entidade pode ser uma pessoa, local, objeto físico ou computacional. Já a computação ciente do contexto ou computação sensível ao contexto é o uso das informações de contexto nas tarefas relevantes do usuário. Os "cinco W's"definem a computação ciente do contexto:

- ***Who* (Quem):** os sistemas atuais identificam as iterações de um usuário em particular, raramente incorporando outras informações do ambiente. Para os humanos recordar sobre eventos do passado é ter mais informações, detalhes, sobre o ocorrido.

- ***What* (O que):** Perceber e interpretar a atividade humana é um problema difícil. Os aplicativos dirigidos ao contexto provavelmente precisarão incluir interpretações humanas para serem capazes de fornecerem informações úteis.

- ***Where* (Onde):** este componente de contexto tem sido muito utilizado com o "Quando". Alguns sistemas de guia de turismo aprendem a partir do histórico dos movimentos (pontos turísticos já visitados pelo turista), para desta forma inferir em pontos turísticos de seu interesse na sua localização atual.

- ***When* (Quando):** utiliza-se bastante o tempo para indexar registros ou determinar quanto tempo uma pessoa ficou em um local, a maioria das aplicações orientadas ao contexto desconhecem a passagem do tempo. As mudanças relativas ao tempo ajudam a interpretar a atividade humana. Por exemplo, uma casa ciente de contexto pode perceber quando uma pessoa idosa desviou de sua rotina matinal tipicamente ativa.

- ***Why* (Por que):** Mais desafiador do que saber "O que"uma pessoa está fazendo é entender o "Por que". A percepção de outras informações contextuais podem dar uma indicação do estado de uma pessoa, como temperatura, frequência cardíaca e resposta galvânica da pele.

2.4.1 Classificação das Informações de Contexto

A computação ciente do contexto é entendida como a capacidade dos dispositivos adaptarem o seu comportamento ao meio ambiente, e desta forma melhorar a sua usabilidade, Gronli [22]. De acordo com Chen [9], as informações de contexto podem ser classificadas em:

- **Contexto Computacional:** informações sobre recursos computacionais e suas características. Por exemplo, conectividade de rede, custo de comunicação e largura de banda de comunicação;
- **Contexto do Usuário:** informações do usuário como localização, situação social e perfil do usuário;
- **Contexto Físico:** informações sobre o mundo físico. Por exemplo, temperatura, condição de tráfego e luminosidade;
- **Contexto Temporal:** informações sobre o tempo de um dia, mês e ano.

2.4.2 Aplicação da computação Ciente do Contexto

De acordo com [47], as aplicações cientes do contexto são descritas em quatro categorias:

- **Seleção por proximidade:** é uma técnica de interface de usuário em que objetos que estão próximos são escolhidos ou enfatizados mais facilmente.
- **Reconfiguração contextual automática:** é o processo de adição de novos componentes, remoção de componentes existentes, ou a alteração das conexões entre os componentes. No caso de sistemas cientes de contexto, o aspecto interessante é como as informações de contexto relativas ao histórico de suas ações podem trazer diferentes configurações de sistema.
- **Informação contextual e comandos:** as ações das pessoas podem ser frequentemente previstas. Tarefas realizadas com frequência como ir a cozinha e abrir a porta da geladeira. A informação contextual e comandos visa explorar isso.

- **Ações ativadas por contexto:** são simples regras se/então usadas para especificar como os sistemas cientes de contexto devem se adaptar. Informações sobre o contexto de uso de uma cláusula de condição que ao ser ativada desencadeia comandos consequentes; algo como viver em sistemas especialistas baseados em regras.

Um exemplo de aplicação ciente de contexto é o Prometheus [37], um serviço de segurança adaptativa para dispositivos móveis. Imagine o seguinte cenário um dispositivo móvel encontra-se com bateria em nível crítico e está conectado a um ambiente de rede com nível de segurança alto, então para economizar a bateria a aplicação seleciona uma criptografia de nível baixo que não consome tanto a bateria. Adaptar a segurança da aplicação ao ambiente de rede e ao estado do dispositivo móvel realiza um ganho em termos do consumo de bateria, que em determinadas situações nas quais políticas estáticas sejam empregadas, poderia haver um gasto desnecessário deste recurso que é bastante limitado.

Segundo Kudo [26], outros exemplos que também podem ser citados são os sistemas *Rememberer* e *CampusAware*. O primeiro tem como objetivo capturar experiências e estimular interações entre as pessoas por meio de dispositivos móveis. O segundo é um sistema de guia de turismo que identifica a localização do usuário, a partir do sensor de GPS, e fornece diversos serviços. Um destes serviços é o fornecimento de informações de como encontrar os pontos turísticos de interesse.

2.4.3 Captura de Informações de Contexto

De acordo com Brown [7], a informação de contexto pode ser capturada a partir de sensores, informações já existentes (diários, previsões de tempo ou valores de ações), do estado do dispositivo computacional ou pelo histórico da interação do usuário com o equipamento. Por exemplo, sensores de uma sala podem detectar que não há movimento, desta forma entendendo que não há pessoas no local e acionando o desligamento das lâmpadas para que haja economia de energia elétrica.

2.4.4 Lógica *Fuzzy*

A lógica *Fuzzy* é a lógica que busca o racionio pelo aproximado em vez do exato, Gomide [20]. Segundo Ortega [34], a lógica *Fuzzy* foi apresentada em 1964 por Lotfi A. Zadeh., quando trabalhava com problemas de classificação de conjuntos que não eram bem definidos (a transição de um conjunto a outro era lenta e não abrupta), por exemplo, o numero 3,8 pertence ou não aos números aproximados de 4. Neste caso pode-se possuir mais de uma resposta, que dependerá do tipo de problema que se busca resolver.

A lógica *Fuzzy* veio a solucionar problemas de controles de processo, reduzindo a sua complexidade, que eram intratáveis por técnicas tradicionais, Gomide [21]. Na lógica *Fuzzy* o termo linguístico é interpretado em um subconjunto *Fuzzy* do intervalo unitário (por exemplo, verdade, muito verdade, falso, ...). O uso de elementos qualitativos são mais comuns que os quantitativos.

Em vez de assumir valores numéricos, as variáveis linguísticas assumiriam instâncias linguísticas. Uma variável linguística tem característica de assumir valores dentro de conjuntos de termos linguísticos. Por exemplo, a variável linguística Temperatura pode vir a assumir um dos elementos do conjunto (baixa, média, alta), vide. Figura 2.4. Verifica-se que temperaturas entre 25°C e 50°C podem ser tanto baixas como medias, caso levássemos em conta o grau de pertinência, valor que descreve a pertinência a um grupo ou a outro, poderíamos dizer que os valores da temperatura poderiam pertencer mais a um conjunto e menos a outro.

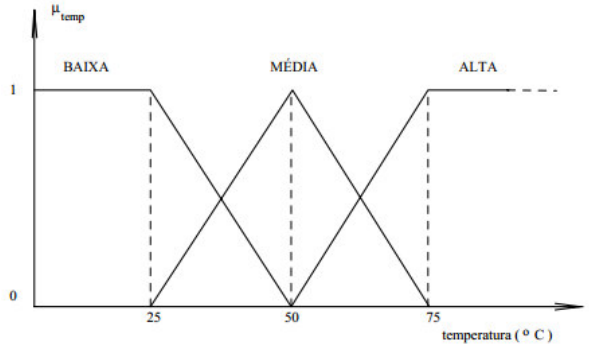

Figura 2.4: Variável linguística de Temperatura, Gomide [21].

Uma forma comum de representar conhecimento é através de regras de condição/ação. Nestas, as condições satisfeitas poderão manter ou levar a operações

desejadas. As regras se-então são chamadas de declarações condicionais, podendo ser chamadas de regras de controle ou modelagem *Fuzzy*. Por exemplo, na figura 2.5 a inferência *Fuzzy* do modus ponens generalizado é apresentada. A partir de uma regra estabelecida, condição se, a partir de um fato 'x' seja 'A', pode-se inferir em uma consequência que 'y' é 'B'.

Fato:	x é A'
Regra:	se (x é A) então (y é B)
Consequência:	y é B'

Figura 2.5: Regra de inferência decorrente do modus ponens, Gomide [21].

2.5 Segurança em Redes

Nos dias de hoje a Internet tem sido um dos grandes veículos de comunicação mundial tanto que no âmbito nacional vem crescendo de forma vertiginosa, seja pela necessidade ou pelo incentivo do governo federal. Os dados transmitidos pela Internet tornam-se vulneráveis a ponto de serem interceptados por usuários indevidos e não autorizados, sendo necessário um modo de garantir que essas informações não sejam modificadas. Dessa forma, é essencial garantir as características de integridade, confidencialidade e autenticação das mesmas.

2.5.1 Características da Comunicação Segura

Segundo Kurose [27], as características da comunicação segura são:

- **Integridade:** dois comunicantes (Alice e Bob) querem garantir que o conteúdo de sua comunicação não seja alterado, por acidente ou por má intenção. Uma das soluções é utilizar as extensões de soma de verificação que encontra-se nos protocolos de transporte e enlace para proporcionar a integridade da mensagem.

- **Confidencialidade:** é permitir que somente o remetente e destinatário possam entender o conteúdo da mensagem. Criptografa a mensagem para que o interceptador não possa entender o seu conteúdo.

- **Autenticação:** tanto o remetente como o destinatário precisam confirmar a identidade de cada um na comunicação, para assegurar que realmente é quem alega ser.

- **Segurança operacional:** hoje quase todas as organizações (empresas, universidades, etc) possuem suas redes conectadas à Internet. Estas redes podem ser comprometidas por atacantes, por exemplo, lançando ataques de negação de serviço ou adquirindo segredos da organização. Os mecanismos operacionais são utilizados para deter estes ataques como o *firewall* e sistemas de detecção de invasão, alertando os administradores da rede de alguma atividade suspeita.

De acordo com William [53], além destas características citadas há também o controle de acesso. Serviço que realiza o impedimento de uso não autorizado de um recurso ou serviço ou a permissão destes.

2.5.2 Criptografia

Criptografia é a técnica de utilizar algoritmos matemáticos para transformar um texto claro em algo ininteligível, William [53]. Sua compreensão está ligada ao detentor de uma chave e ao mesmo algoritmo matemático utilizado na criptografia para recuperar os dados em texto claro. Atualmente as criptografias simétricas e assimétricas são as mais utilizadas, Da [11], além da função *hash* que unida com as duas anteriores formalizam as três classes criptográficas básicas aprovadas, Barker [6].

De acordo com William [53], até a década de 1970 a criptografia simétrica era o único tipo de criptografia conhecida, em que se utilizava uma única chave secreta e um algoritmo de criptografia para transformar um texto claro em um texto cifrado, e que a partir do algoritmo de descriptografia e a mesma chave de criptografia, conseguiria recuperar do texto cifrado, o texto claro, conforme mostrado na Figura 2.6.

Os algoritmos de criptografia ou descriptografia não precisam ser secretos o que torna a criptografia simétrica praticável, a dificuldade está em manter a chave secreta, pois ela é a única informação sigilosa que impede um intruso de descriptografar os dados transmitidos, Kurose [27]. Um dos maiores questionamentos é como realizar a troca da chave secreta entre o remetente e destinatário de forma segura e garantir a proteção da mesma contra terceiros, Sousa [51]. Exemplos

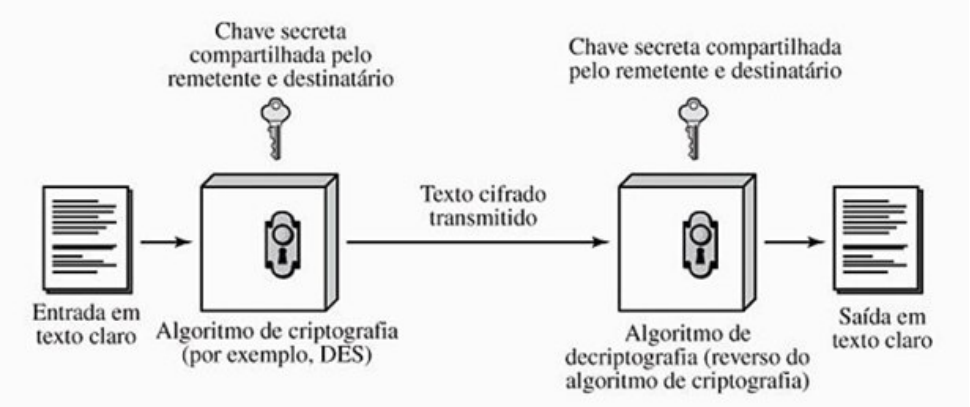

Figura 2.6: Modelo simplificado de Criptografia Simétrica, William [53]

de algoritmos de criptografia simétrica mais conhecidos são o *Advanced Encryption Standard* (AES) e o *Data Encryption Standard* (DES).

O algoritmo de criptografia simétrica requer que a distribuição da chave seja feita por um canal seguro. Por tanto, a dificuldade está em assegurar esse compartilhamento com sucesso. A solução proposta é utilizar a criptografia assimétrica, pois ela consiste em utilizar um par de chaves, uma para criptografar (chave pública) e outra para descriptografar (chave privada). A criptografia assimétrica é também conhecida como criptografia de chave pública, Sousa [51].

Em um exemplo de utilização de chaves públicas, Kurose [27], Alice e Bob querem se comunicar pelos computadores utilizando a Internet Bob possui uma chave pública K_b^+ que é de conhecimento de todos e uma privada K_b^- que somente ele a conhece.

Para se comunicar com Bob, Alice primeiramente busca a chave pública dele e utiliza um algoritmo de criptografia, padronizado e conhecido, para desta forma codificar uma mensagem m, obtendo $K_b^+(m)$. Após a mensagem ser enviada a Bob, este utiliza um algoritmo de descriptografia (padronizado) e sua chave privada K_b^-, $K_b^-(K_b^+(m))$, para assim recuperar a mensagem m de Alice.

Este método permite que Alice e Bob se comuniquem sem que haja troca de uma chave secreta antecipadamente, conforme a Figura 2.7. Exemplos de algoritmos de criptografia assimétrica são o Diffie-Hellman e o *Rivest, Shamir e Adleman* (RSA).

Criptografar e descriptografar mensagens são processos muito dispendiosos, portanto utilizar um resumo da mensagem original para ser criptografado é mais viável. Existem vários algoritmos para calcular o resumo

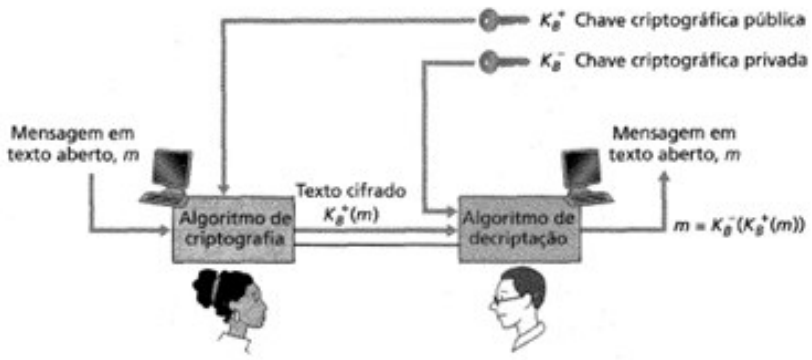

Figura 2.7: Modelo de Criptografia de chave pública, Kurose [27]

de uma mensagem, conhecidos como funções de *hash* (H), mas devem ter as seguintes propriedades: (1) duas mensagens diferentes (x e y) devem ser inviáveis computacionalmente de produzirem, H(x) = H(y), além de (2) ser simples do ponto de vista do cálculo, Kurose [27].

Para proteger a mensagem original, seu resumo de mensagem é assinado digitalmente. Utiliza-se a chave privada de Bob para criptografar a mensagem, não podendo ser alterado por ninguém, $K_b^-(H(m))$, para verificar se a mensagem m recebida pelo destinatário não fora modificada, (m, $K_b^-(H(m))$), por um intruso ou por acidente é utilizado a função de Hash na mensagem m e comparada com o conteúdo da mensagem descriptografada com a chave pública do remetente, caso sejam iguais a mensagem enviada é íntegra e não repudiável. Alguns algoritmos de *hash* são o *Message-Digest algorithm 5*(MD5) desenvolvido pelo Rivest em 1992 [44] e o *Secure Hash Algorithm 1*(SHA-1), Eastlake [15].

2.5.3 TLS

A TLS foi desenvolvida pela Netscape em 1994, mas somente lançada em 1996 na sua versão 3.0, Freier [18]. Este protocolo foi projetado para utilizar TCP (*Transmission Control Protocol*) para fornecer um serviço de segurança confiável fim a fim, William [53]. A camada TLS situa-se entre a camada de aplicação e a de transporte, Kurose [27], do modelo *Open System Interconnection*(OSI). A TLS é independente do protocolo utilizado seja HTTP, FTP, etc. Esta não é limitada à *Web* por meio de navegadores, podendo ser implementada em servidores e aplicativos para comunicação na Internet.

Os mecanismos ofertados pela TLS são:

- **Autenticação:** as partes comunicantes, servidor e cliente, podem se autenticar mutuamente ou opcionalmente por meio de algoritmos de chaves públicas (RSA).

- **Confidencialidade:** é criada uma sessão TLS que criptografa todos os dados transferidos pelo remetente por meio de chaves secretas e algoritmos simétricos (AES), estes definidos durante a troca de chaves na primeira fase do estabelecimento da sessão.

- **Integridade:** para assegurar a integridade das mensagens transmitidas são utilizados Códigos de Autenticação de Mensagem (MAC) calculados por algoritmos de funções de *hash* (MD5 ou SHA), Weber [52].

Segundo William [53], TLS tem dois conceitos fundamentais:

- **Sessão TLS:** é a associação entre o cliente e servidor, criado pelo protocolo de estabelecimento de conexão. Serve para definir os parâmetros criptográficos compartilhados entre várias conexões.

- **Conexão TLS:** é um transporte apropriado a um tipo de serviço. As conexões são ponto a ponto e estão associadas a uma sessão.

2.5.3.1 Arquitetura e Estabelecimento da Comunicação

A arquitetura TLS é composta por duas camadas de protocolos diferentes, a primeira tem-se o TLS *Record Protocol* e a segunda o *Change Cipher*, *Alert Protocol* e *Handshake Protocol*. A Figura 2.8 ilustra a pilha de protocolos TLS.

De acordo com William [53], significa que o:

- ***TLS Record Protocol:*** fragmenta cada mensagem da camada superior em blocos de 2^{14} *bytes* ou menos transformando-os em registros *TLSPlaintext*, realiza compressão de *TLSPlaintext* para registros *TLSCompressed* sem que haja perda de dados e aumento de no máximo 1024 *bytes*. O algoritmo

Camadas TCP/IP com SSL	
	Camada de Aplicação HTTP, FTP, TELNET...
SSL	Change cipher, Alert Protocol, Handshake Protocol
SSL	Camada SSL Record Protocol
	Camada TCP
	Camada IP

Figura 2.8: Pilha de protocolos do TLS/SSL

utilizado para compressão foi negociado durante o *handshake* (estabelecimento da comunicação), seja o valor padrão é nulo para o TLS na versão 3.0. A etapa seguinte utiliza a chave secreta compartilhada durante o *handshake* para calcular o MAC sobre os dados compactados. A autenticação da mensagem compactada é acrescida ao MAC para posteriormente realizar a criptografia de chaves simétricas (as funções criptográficas foram definidas durante o *handshake*), transformando *TLSCompressed* em *TLSCiphertext*. A última etapa é definir o cabeçalho: o tipo de conteúdo (protocolo da camada mais alta que processa o fragmento transportado), versão principal (versão utilizada caso seja TLSv3, valor 3), versão secundária (caso TLSv3, valor 0) e o tamanho do conteúdo (tamanho em *bytes* do fragmento do texto claro).

- ***TLS Change Cipher:*** é composto por uma única mensagem de um 1 *byte* sua funcionalidade é atualizar o conjunto de cifras a ser utilizado na conexão.

- ***TLS Alert Protocol:*** utilizado para enviar alertas sobre o TLS para as partes envolvidas. As mensagens de alerta são compactadas e criptografadas, é composta por dois *bytes*. O primeiro *byte* identifica o grau de "gravidade" da mensagem podendo ser do tipo *warning* ou fatal, caso seja fatal o TLS encerra a conexão. O segundo *byte* contém o código de um tipo de alerta específico (erros de certificados, de criptografia ou sequências de mensagem).

- ***TLS Handshake Protocol:*** realiza a autenticação entre cliente e servidor, negocia o MAC, algoritmos criptográficos e chaves que utilizarão para proteger os dados transmitidos. Utiliza criptografia de chaves públicas para realizar a negociação

inicial, em que a chave simétrica de sessão gerada aleatoriamente é transferida por um meio seguro. Para garantir uma maior segurança é utilizada a função *hash* (MD5 ou SHA) conjuntamente com o MAC. O protocolo *Handshake* é utilizado antes que qualquer transferência de dados seja realizada.

Uma das formas mais genéricas para o *handshake* é descrita em capacidades de segurança, autenticação do servidor, autenticação do cliente e término, William [53]. As quatro fases são apresentadas na Figura 2.9.

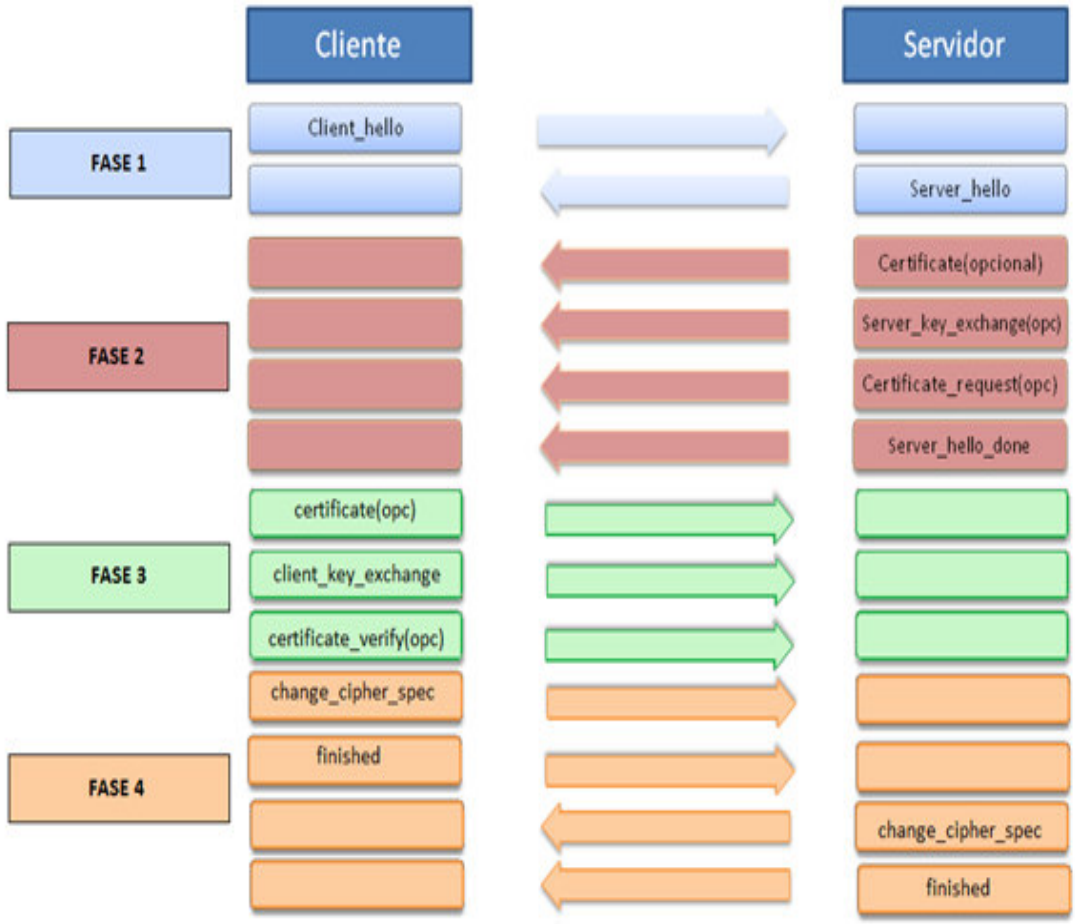

Figura 2.9: Protocolo do *handshake*

Na primeira fase são estabelecidas as capacidades de segurança pelos seguintes parâmetros: versão do TLS, valor aleatório para impedir ataques por repetição, ID de sessão para atualizar ou criar uma nova conexão na sessão, o conjunto de cifras suportado e uma lista de métodos de compactação admitida. Na apresentação mútua, uma mensagem *client hello* define os parâmetros da forma que o cliente admite e a mensagem *server hello* contém os mesmo parâmetros da mensagem *client hello*, desta forma negociando quais parâmetros irão utilizar.

Na segunda fase o servidor pode enviar o seu certificado (caso seja necessário autenticá-lo), a troca de chaves e pode solicitar o certificado do cliente. Uma mensagem é enviada sinalizando o término da fase de apresentação mútua.

Na terceira fase o cliente envia seu certificado caso tenha sido solicitado, uma mensagem com a troca de mensagens (caso seja o tipo RSA este gera um pré-segredo-mestre aleatório e criptografa-o com a chave pública fornecida pelo servidor) e uma mensagem para oferecer explicitamente uma verificação do certificado do cliente.

Na quarta fase tanto o cliente e o servidor enviam uma mensagem *change cipher spec* que faz parte do protocolo *Change Cipher* e não do protocolo de estabelecimento de sessão para determinar o serviço de criptografia, e uma mensagem de conclusão (*finished*), por meio do qual esta verifica se o processo de autenticação e troca de chaves foram bem sucedidos. Para garantir a entrega confiável das mensagens é utilizado o TCP.

2.6 Trabalhos Relacionados

Os dispositivos móveis possuem recursos limitados, e ainda é um grande desafio otimizá-los e fornecer segurança simultaneamente. Na literatura existem várias formas de prover a segurança às aplicações e dados do dispositivo móvel como Pirmez [37], An [3], Cirqueira [10], Lee [28] e Moraes [14]. Enumeram-se a seguir os trabalhos relacionados.

2.6.1 Prometheus: Um Serviço de Segurança Adaptativa

O trabalho proposto por Pirmez [37] provê duas funcionalidades adaptação dinâmica dos controles de segurança e das próprias políticas de segurança. O diferencial deste trabalho é a adaptação em tempo de execução destas duas funcionalidades durante a execução de aplicações ubíquas em dispositivos móveis garantindo a disponibilidade, a confidencialidade e a integridade dessas aplicações. Figura 2.10 apresenta o serviço de segurança adaptativo que possui como os principais componentes do Prometheus o gestor de segurança adaptativa, gestor de conexões e gestor de recursos.

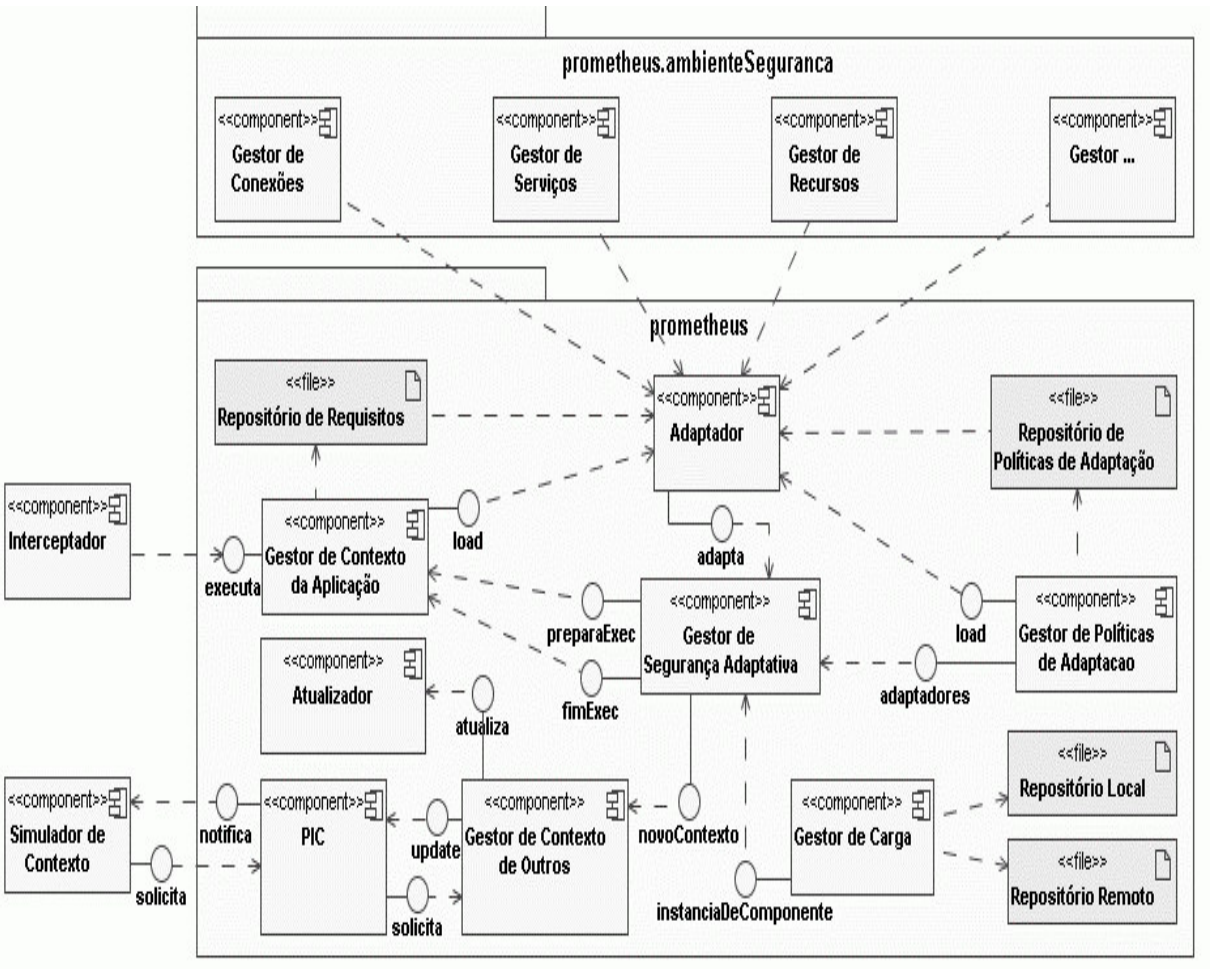

Figura 2.10: Diagrama de componente do Prometheus, Pirmez [37]

- Gestor de segurança adaptativa: é o principal componente responsável por determinar a política e os controles de segurança dinamicamente. Utilizando 3 níveis de segurança para a adaptação nível fraco (DES), médio (3DES) e alto (AES).

- Gestor de conexões: garante a confidencialidade e integridades das informações trafegadas. Além de criar canais de segurança utilizando algoritmos de criptografia mais adequados ao ambiente em que o dispositivo está localizado.

- Gestor de recursos: responsável por gerenciar os recursos do dispositivo. Por exemplo, quando um dispositivo possuir pouco nível de bateria podendo diminuir a potência de transmissão para economizar o recurso da bateria.

2.6.2 *Context-aware Dynamic Security Configuration for Mobile Communication Device*

O esquema proposto por An [3], possui como finalidade maximizar o desempenho dos dispositivos móveis em termos de recursos computacionais sem degradar seu nível de segurança, baseando-se em sensibilidade ao contexto. Fornece ao usuário alta conveniência e economia de recursos do dispositivo. Esse esquema não lida com adaptação à vários algoritmos criptográficos e automação da segurança na fase de avaliação sendo proposto para trabalhos futuros. A avaliação deste esquema fora realizado na plataforma Linux Ubuntu computador pessoal DELL XPX M1710. Para manter a proteção as funções de segurança *firewall*, sistema de detecção de intrusão, *scanner* de vírus, controle de acesso, monitor de bateria e proteção contra vazamento de dados. Na Figura 2.11, a arquitetura do esquema é dividida em 3 modulos e 2 tabelas:

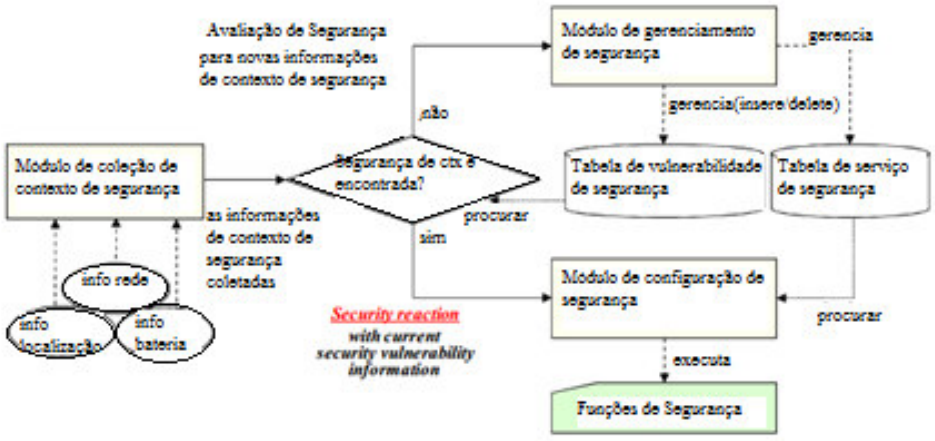

Figura 2.11: *Arquitetura do Context-aware Dynamic Security Configuration*, An [3]

- **Módulo de coleção de contexto de segurança:** para coletar informações de contexto do dispositivo móvel;

- **Módulo de gerenciamento de segurança:** gerencia as tabelas de vulnerabilidades de segurança e serviço de segurança;

- **Módulo de configuração de segurança:** é fornecido o melhor serviço de segurança ao contexto que se encontra o dispositivo móvel a ser empregado;

- **Tabela de vulnerabilidades de segurança:** indica a qual ataque o dispositivo móvel está vulnerável;

- **Tabela de serviço de segurança:** armazena as capacidades e funções de segurança.

2.6.3 Um Mecanismo de Segurança com Adaptação Dinâmica em Tempo de Execução para Dispositivos Móveis

O mecanismo apresentado por Cirqueira [10], propõe um esquema de segurança adaptativa com confidencialidade aos dados trafegados pelo dispositivo móvel, capaz de adaptar o nível de segurança baseado no contexto e emprego eficiente dos recursos. Alocando um algoritmo criptográfico melhor indicado ao contexto que se encontra o dispositivo móvel. Todas as interações do usuário com a aplicação são informações de contexto relevantes. Por exemplo, um dispositivo móvel encontra-se conectado a um ambiente de rede seguro, nesse caso não há a necessidade de utilizar um esquema de criptografia que demande muitos recursos computacionais. Dessa forma, utiliza-se um algoritmo que consome menos recursos para que aumente a vida útil do dispositivo.

2.6.4 *Context-Aware Security model for Social Network Service*

O trabalho proposto por Lee [28] é um modelo de segurança para serviços de redes sociais para *smartphones*. Este artigo sugere um sistema de controle de acesso e autenticação ciente do contexto via cenários que se encontra o *smartphone*. Utiliza-se o algoritmo *Fuzzy* para analisar as informações do *smartphone* adquiridos pela autenticação do usuário. As informações coletadas são ambíguas para que possam ser definidas quantitativamente, somente sendo estimadas. Caso não fosse utilizada a lógica *Fuzzy* muitas regras poderiam ser geradas para definir se o desempenho é bom ou não, dificultando a aplicação do nível de segurança que deveria ser utilizado.

2.6.5 SSACC

O serviço proposto por Moraes [14] tem como objetivo fornecer um canal seguro de transferência de arquivos entre o DM e o servidor, utilizando o TLS com processos criptográficos diferentes. A autenticação é realizada por meio de criptografia de chave pública. A integridade e confidencialidade são fornecidas por criptografia de chave privada e assinaturas digitais.

Visa fornecer segurança na autenticação, em relação ao gasto da criptografia selecionada, nível de segurança que a rede oferece, quantidade de bateria disponível e poder de processamento do DM. A contribuição deste trabalho é criar um serviço de segurança de adaptação dinâmica, que tome a decisão de qual algoritmo de criptografia é suficientemente seguro para a autenticação dos DMs, a partir do ambiente que estão conectados. A Tabela 2.1 apresenta uma análise comparativa entre os mecanismos descritos anteriormente.

Tabela 2.1: Comparação dos mecanismos de segurança adaptativos para computação Móvel.

A.P	**A.D**	**N.S**	**C.I**	**Fuzzy**	**A.A.C**	**TLS**	**CN**	**D.M Reais**
Pirmez [37]	✓	✓	✓	-	-	✓	-	✓
An [3]	✓	-	✓	-	-	-	-	✓
Cirqueira [10]	✓	✓	✓	-	✓	-	-	✓
Lee [28]	✓	✓	✓	✓	-	-	-	✓
Moraes [14]	✓	✓	✓	-	✓	✓	✓	✓

- **A.P** - Abordagens Propostas **A.D** - Adaptação Dinâmica **N.S** - Nível de Segurança
- **C.I** - Coleta de Informações **Fuzzy** - Análise de informações baseado Fuzzy **A.A.C** - Avaliação de Algoritmos Criptográficos
- **TLS** - Transport Layer Security **CN** - Computação em Nuvem **D.M Reais** - Dispositivos Móveis Reais.

2.7 Síntese

Neste capítulo, os conceitos relacionados com a computação verde, computação em nuvem, computação móvel em nuvem, computação ciente de contexto e segurança em redes foram apresentados. Em computação verde, tem-se como principal objetivo a economia de recursos.

A computação em nuvem é um modelo de acesso conveniente aos seu recursos compartilhados em rede e dos recursos computacionais os serem provisionados dinamicamente, a partir da necessidade do dispositivo, tornando-se bastante atraente a computação móvel que possui recursos limitados.

Na seção de computação ciente do contexto entende-se como a capacidade dos dispositivos ou sistemas adaptarem-se as suas necessidades e aos ambientes que o cercam. A seção de segurança em redes mostra as vulnerabilidades que os dados de usuários de aplicações estão sujeitos e algumas soluções utilizadas.

Na seção de trabalhos relacionados, alguns projetos aplicados para segurança adaptativa e suas principais características foram apresentados. E por fim uma comparação entre as características dos trabalhos foi apresentada. Com essa comparação concluiu-se que há necessidade de implementar mecanismos que forneçam segurança sem degradar os recursos do dispositivo móvel da melhor forma possível, além de que qualquer economia realizada a mais realizada pode vir a ser relevante.

3 Mecanismo de Segurança Ciente do Contexto para Computação em Nuvem Móvel

Este capítulo tem como objetivo apresentar o mecanismo de segurança ciente do contexto para dispositivo móvel para Computação em Nuvem e um avaliador de algoritmos criptográficos para dispositivo móvel, para que forneça uma comunicação segura e adaptável ao servidor em nuvem. Serão discutidos objetivos e requisitos do mecanismo. Em seguida, a arquitetura do mecanismo cliente, servidor e seus componentes são detalhados, bem como o seu funcionamento.

3.1 Objetivos

As informações do usuário no dispositivo móvel tornam-se vulneráveis ao serem transmitidas na rede de computadores. Medidas de segurança devem ser realizadas para dificultar aos invasores o seu acesso. A segurança tem um nível de complexidade e uso de recursos computacionais muito alta para ser empregada. Um dispositivo móvel por ter recursos limitados, teria dificuldade para implementá-las. Tentando resolver este problema utiliza-se a segurança adaptativa. A partir da necessidade do dispositivo móvel são utilizados mecanismos de segurança que consomem mais ou menos recursos.

De forma a resolver o problema de armazenamento de dados no dispositivo móvel, utiliza-se a computação em nuvem que vêm a fornecer uma maior capacidade de armazenamento, além de tratar da computação móvel no paradigma da computação em nuvem. O objetivo do mecanismo de segurança ciente do contexto para computação em nuvem móvel aqui proposto é identificar o contexto do dispositivo móvel e adaptar o grau de segurança no momento de envio de dados ao servidor na nuvem, para prover uma economia dos recursos do dispositivo móvel. Com isso garantimos a segurança e ergonomia do dispositivo móvel. Neste trabalho, foi proposto:

- Avaliar o desempenho dos algoritmos criptográficos para o dispositivo móvel em uso;
- Representar o contexto do dispositivo móvel utilizando a lógica *Fuzzy*;
- Tomar decisões automaticamente para a adaptação dos níveis de segurança empregado.

3.2 Arquitetura

A arquitetura proposta nesta dissertação tem como objetivo fornecer segurança ciente do contexto para computação em nuvem móvel. Dessa forma, o mecanismo cliente avalia os algoritmos criptográficos quanto ao uso de recursos do dispositivo móvel, analisa as informações de contexto representando-as com a lógica *Fuzzy* e toma a decisão de utilizar o algoritmo de criptografia TLS mais adequado para a situação que o dispositivo móvel se encontrar. Na Figura 3.1, verificam-se os componentes da arquitetura do mecanismo cliente que são TestStress, Analisador e TomadaDecisão, estes componentes cooperam entre si para o seu bom funcionamento. A Figura 3.2, a arquitetura do Servidor é dividida em aplicação, initSSL e configClient.

Figura 3.1: Arquitetura do mecanismo proposto Cliente.

Os componentes do mecanismo cliente e servidor serão descritas com mais detalhes nas próximas seções.

Figura 3.2: Arquitetura do mecanismo proposto Servidor.

3.2.1 Componentes do Servidor

O Servidor foi concebido utilizando a *JAVA SECURE SOCKET EXTENSION* (JSSE). Ele é composto pelos componentes Aplicação, InitSLL e ConfigClient, executando sobre o ambiente de infra-estrutura EUCALYPTUS Liu [29]. A seguir serão apresentados os componentes que formam o Servidor, bem como seu funcionamento.

3.2.1.1 Componente Aplicação

Esse componente possui a interface do aplicativo servidor e rotinas para executar os demais componentes.

3.2.1.2 Componente InitSSL

Este componente é responsável por iniciar os parâmetros de configuração do TLS do lado Servidor, como carregar os materiais das classes *Keymanager* e *Trustmanager*, para realizar o *handshake*(estabelecimento de conexão), além de envio dos seus arquivos por um canal seguro. O *Keymanager* decide qual chave utilizar para as credenciais de autenticação. O *Trustmanager* é responsável por decidir se as credenciais apresentadas são confiáveis para permitir as conexões.

3.2.1.3 Componente ConfigClient

O componente responsável por negociar qual tipo de criptografia simétrica que fora escolhido pelo cliente para que possa iniciar a descriptografia ou criptografia de dados, conforme a necessidade do dispositivo móvel. Dessa forma inicializa o recebimento ou o envio dos dados solicitados pelo cliente móvel.

3.2.2 Ambientação e componentes do mecanismo de segurança ciente do contexto

O ambiente de construção do mecanismo cliente proposto foi realizado para a plataforma Android [4] Cirqueira [10]. Segundo Gartner [19], ela é uma das plataformas mais utilizadas no mundo e aberta, além de prover um ambiente rico para o desenvolvimento de aplicações. A seguir serão apresentados os componentes que formam o mecanismo de segurança ciente do contexto proposto, bem como seu funcionamento.

3.2.2.1 Componente Aplicação

Este componente possui a interface da aplicação. Como citado anteriormente, o mecanismo foi implementado para a plataforma Android que é *Activity*, ou atividade, representa uma tela onde usuário pode realizar algo. Segundo Android [4], a *Activity* é da ordem de uma a várias telas no mesmo programa. É necessário definir a primeira tela a ser exibida ao usuário chamada de "*main activity*" ou atividade principal. Toda *activity* pode iniciar outra que realize operações diferentes, sendo que a navegação é guiada por *Intents*, classe que especifica tanto a atividade exata que se deseja iniciar ou o tipo de ação que realiza, além de transportar pequenas quantidades de dados que você deseja utilizar na próxima atividade a ser iniciada.

3.2.2.2 Componente TestStress

O TestStress é responsável por gerar o perfil dos algoritmos criptográficos para o dispositivo móvel. Na Figura 3.3, apresenta-se o processo de estressar os algoritmos criptográficos ao selecionar o arquivo para ser enviado ao servidor, durante o envio são capturadas informações de contexto como o numero de *rounds* (rodadas ou testes), o uso de memória e CPU livre. Estas informações serão salvas no perfil daquele algoritmo até o nível de bateria do dispositivo decrescer 5% . Caso isso ocorra é verificado se todos os algoritmos criptográficos do repositório do TLS compatíveis com o Android foram testados. Em caso negativo, testa um novo algoritmo, enquanto que no caso positivo, o teste é encerrado.

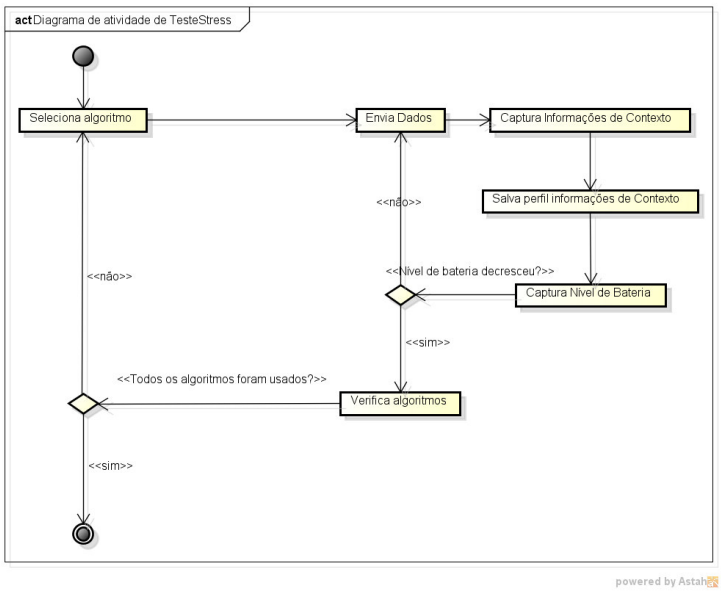

Figura 3.3: Diagrama de atividade de TestStress.

3.2.2.3 Componente Analisador

O Analisador atua nas informações de contexto do dispositivo móvel e representado-os em lógica *Fuzzy*. O processo do Analisador é apresentado na figura 3.4. As informações de contexto são inseridas para serem transformadas em termos linguísticos a partir dos intervalos estabelecidos, aplica a inferência dos termos linguísticos nas regras definidas do *Fuzzy*, para identificar a qual regra ele tem o maior grau de pertinência ou seja aquela que melhor representa o contexto do dispositivo.

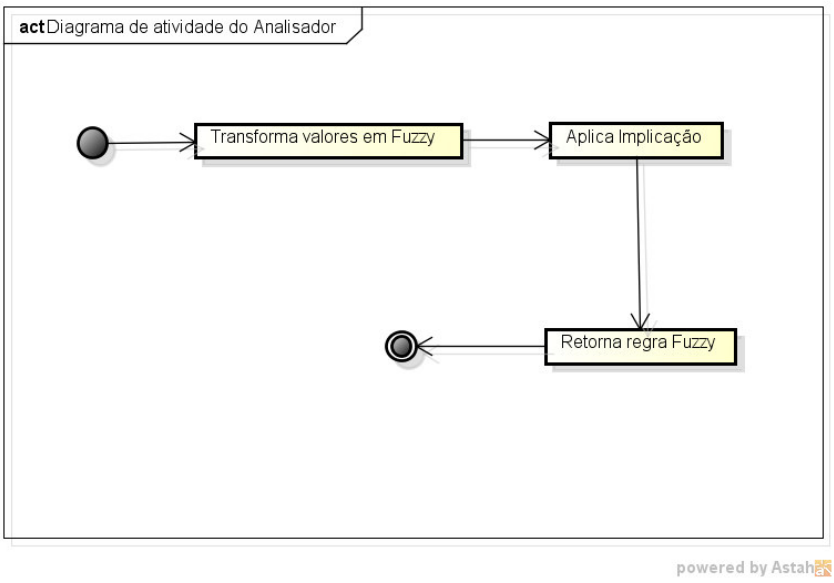

Figura 3.4: Diagrama de atividade do Analisador.

Nos gráficos das Figuras 3.5, 3.6 e 3.7, verificam-se que os intervalos de bateria, os termos linguísticos "*poor*" e "*good*", se interceptam nos valores de 22,2% e 33,3% , e "*good*" e "*excelent*" em 55,5% e 66,6% . Valores dos níveis de bateria que estão entre esses intervalos podem vir a dificultar a identificação do estado da bateria do dispositivo móvel, desta forma utiliza-se a inferência *Fuzzy* para determinar o grau de pertinência para cada termo. Esse grau pode ser identificado como a variação do "*membership*" ou grau de pertinência, que quanto mais próximo do valor 1 mais pertinente àquele estado a variável se encontra. A bateria é a única informação de contexto que possui o intervalo fechado para uso em 100% . As inferências para os intervalos que se interceptam nos gráficos de memória e CPU são realizadas da mesma forma como na da bateria.

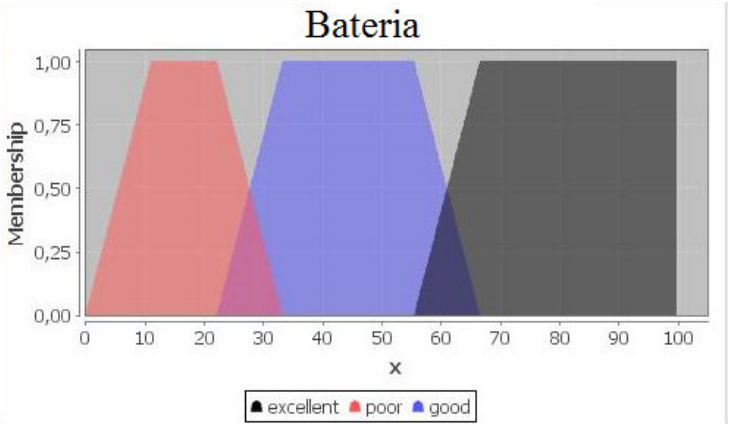

Figura 3.5: Gráfico de intervalos de níveis de Bateria.

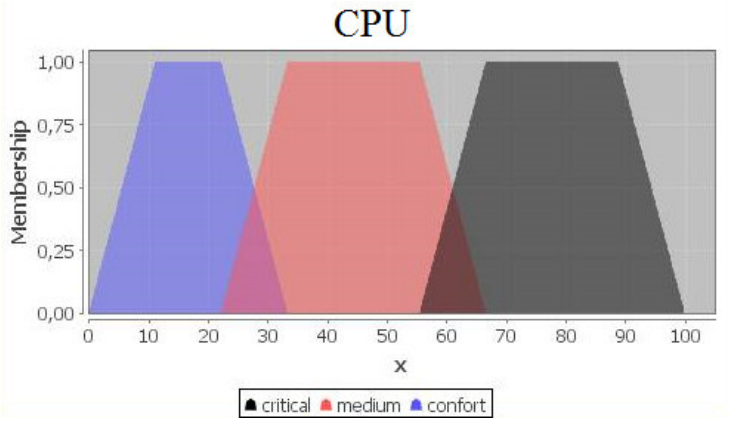

Figura 3.6: Gráfico de intervalos de níveis de CPU.

As regras *Fuzzy* são definidas conforme a figura 3.8. As características que determinam as regras são bateria(definidas pelos termos linguísticos *Poor*, *Good* e *Excellent*), CPU(*Critical*, *Medium* e *Confort*) e memória(*Low*, *Half* e *Full*), que

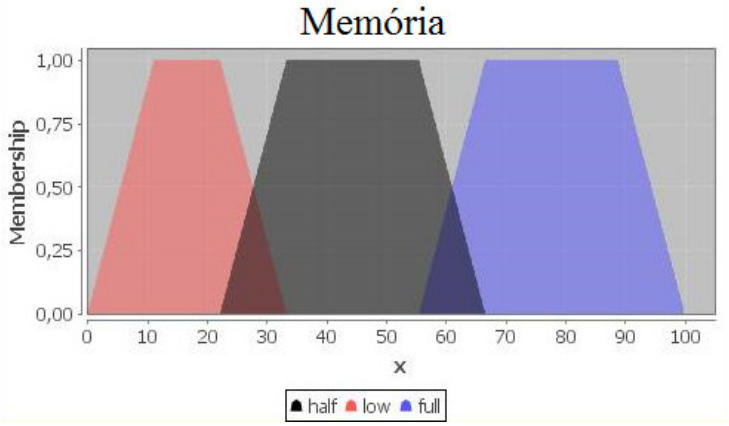

Figura 3.7: Gráfico de intervalos de níveis de Memória.

representam uma combinação entre as mesmas formando as 27 regras implementadas, na figura são mostradas regras do pior e melhor caso. Estas regras enumeradas em 27 descrevem os possíveis estados que o dispositivo móvel pode vir a apresentar.

RULE 1: IF BATTERY IS POOR AND CPU IS CRITICAL AND MEMORY IS LOW THEN status IS DOWN;
RULE 27: IF BATTERY IS EXCELLENT AND CPU IS CONFORT AND memory IS FULL THEN status IS UP;

Figura 3.8: Regras *Fuzzy*.

3.2.2.4 Componente TomadaDecisão

A TomadaDecisão responsável por pontuar os algoritmos criptográficos levando em consideração o perfil gerado pelo componente TestStress e a análise do contexto, para escolha do mais adequado. O processo de TomadaDecisão é apresentado na Figura 3.9, que lustra o perfil numérico dos testes de cada algoritmo transformando-os em termos linguísticos, salvando em um arquivo chamado perfil linguístico. As informações de contexto são capturadas para serem utilizadas como parâmetros de entradas para o componente Analisador, que este ao realizar o seu processo retorna a regra com o maior grau de pertinência. A partir desta regra as informações de contexto presentes nela são pontuadas, para cada perfil do algoritmo.

Por exemplo, no caso da CPU, se o perfil do contexto analisado para CPU for ”CONFORT” o algoritmo criptográfico que possuir o perfil correspondente ao contexto pontua 1, caso seja ”MEDIUM” pontua 0.5 e se for ”CRITICAL” pontua também 0.5. Caso o perfil do contexto analisado para CPU seja ”MEDIUM” o algoritmo

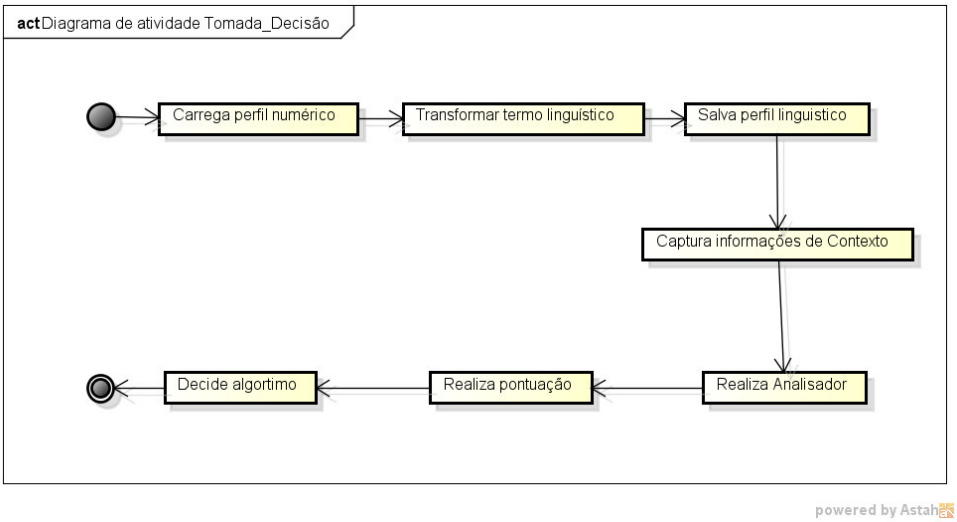

Figura 3.9: Diagrama de atividade TomadaDecisão.

criptográfico que possuir o perfil ”CONFORT” pontua 0, por estar acima do contexto analisado e este algoritmo realiza um consumo acima da capacidade que o dispositivo dispõe no momento, caso seja ”MEDIUM” pontua 1 e se for ”CRITICAL” pontua 0.5. O último estado de contexto que a CPU poderia tomar seria ”CRITICAL” caso o perfil do algoritmo criptográfico seja ”CONFORT” este não pontua, o ”MEDIUM” também não pontua por estar acima do contexto analisado e o que possuir ”CRITICAL” pontua 1.

A pontuação é também realizada nas características de bateria e memória, e o algoritmo que possuir a maior pontuação total será aquele que deve ser utilizado. Caso haja empate, o ambiente de rede (público, privado e doméstico) será utilizado para decidir o algoritmo para a criptografia dos dados e autenticação.

Os algoritmos de criptografia simétricos que serão utilizados são o RC4, a AES-128 e a AES-256, respectivamente, para os níveis de segurança são definidos como fraco, médio e forte, Moraes [14] e Cirqueira [10]. No caso de empate, como dito anteriormente, se o dispositivo móvel estiver no ambiente de rede ”PÚBLICO” aquele algoritmo criptográfico que possuir maior nível de segurança será o escolhido, visto que qualquer pessoa pode vir a realizar um ataque neste tipo de rede por ser aberto. Já no caso do ambiente de rede ser ”PRIVADO”, pressupõe-se que há um nível de segurança maior fornecido pelo proprietário da rede privada, por não haver necessidade do uso de uma criptografia mais segura que possui uma complexidade maior (consumo maior de recursos) é fornecido um nível segurança fraco que possui

um menor gasto de recursos. Por fim, no caso do ambiente de rede "DOMÉSTICO", que possui uma segurança parcial, visto que nem todos tem acesso a rede mas que não está inteiramente livre de um ataque, fornece-se um nível de segurança médio. A criptografia escolhida é enviada ao servidor pra que este possa inicializar o recebimento ou envio de dados. Após a confirmação do recebimento, o dispositivo móvel inicializa a configuração do componente Socket.

3.2.2.5 Componente Socket

O componente Socket é similar ao componente initSocket do servidor, no qual é inicializada a configuração do TLS para o estabelecimento do canal seguro entre o cliente e servidor com a informação de qual tipo de criptografia será utilizada.

3.2.3 Funcionamento do mecanismo de segurança adaptativo

O diagrama de sequência da Figura 3.10 demonstra o funcionamento onde o mecanismo em sua primeira execução utiliza as informações de contexto como entrada de dados para o componente TestStress que gera dessa forma o perfil de cada algoritmo criptográfico, ao enviar o arquivo do dispositivo móvel para o servidor em Nuvem as informações de contexto são coletadas e repassadas para o componente Analisador. Este retorna a regra que melhor representa o estado do dispositivo móvel em relação aos seus recursos.

A partir desta regra, o componente de tomada de decisão realiza a pontuação como descrito na seção anterior. A aquele que possuir a maior pontuação é escolhido. Caso haja empate é decidido a partir do ambiente de rede que o dispositivo estiver conectado, após ser decidido o cliente envia o nome do tipo de algoritmo criptográfico simétrico decidido ao servidor para este realizar a reconfiguração do *socket* e dessa forma realizar o envio dos dados de modo seguro.

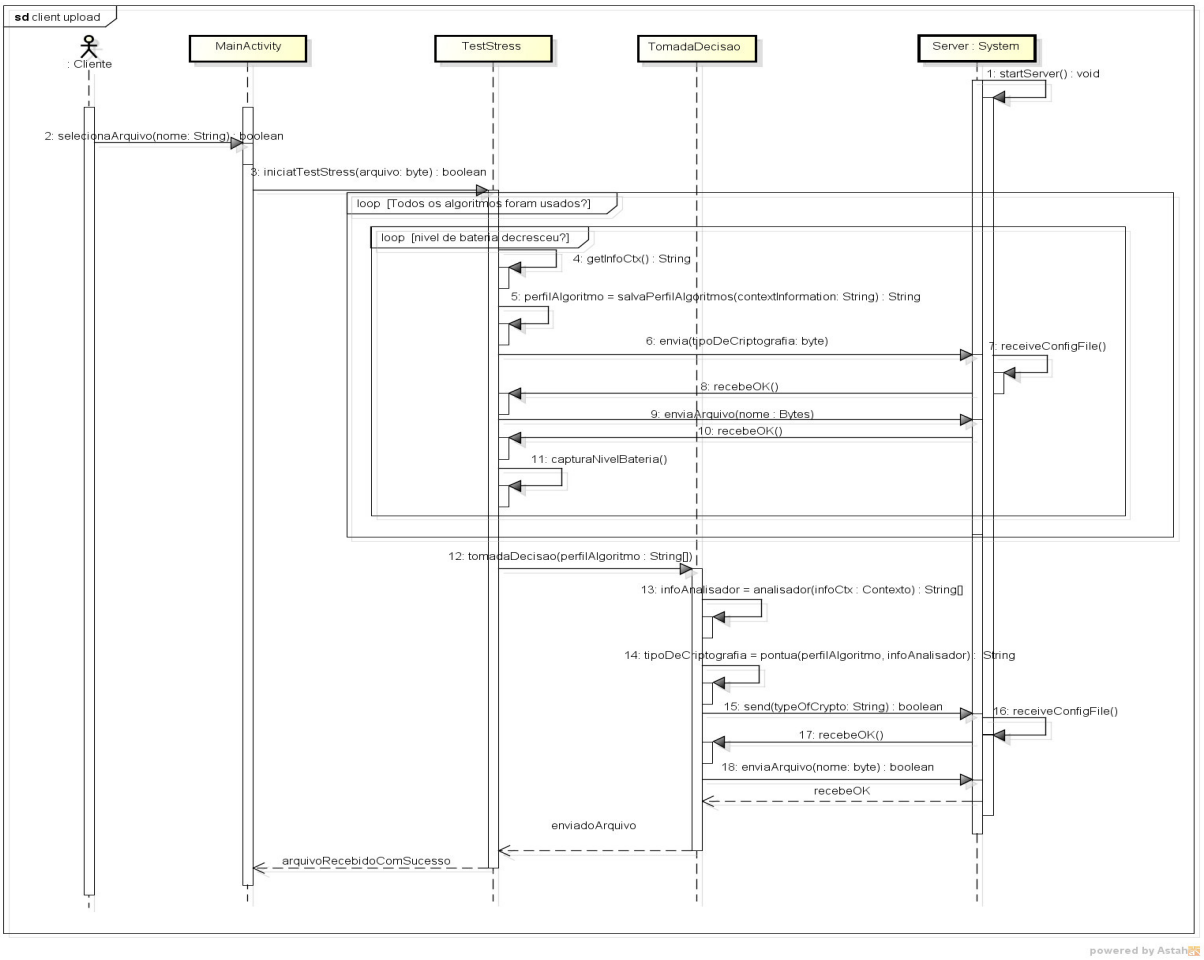

Figura 3.10: Diagrama de sequência sobre o funcionamento do mecanismo.

3.2.4 Análise comparativa

Conforme descrito no Capítulo 2, existem inúmeras técnicas e resultados de implementações para fornecer segurança ciente do contexto para dispositivos móveis, baseados na arquitetura cliente/servidor. A Tabela 3.1 apresenta as características que o mecanismo proposto possui em relação aos mecanismos de segurança adaptativos citados no Capítulo 2 em trabalhos relacionados.

As características do mecanismo proposto, como adaptação dinâmica por solicitação, nível de segurança baseado nos tipos de ambientes de redes (público, privado e doméstico) que o dispositivo móvel está conectado, a coleta de informações de contexto no momento que é processada a solicitação de envio ou recebimento de um arquivo. A análise de contexto baseado em lógica *Fuzzy* para apresentar o estado do dispositivo móvel no momento da solicitação, avaliação de desempenho (CPU,

memória e bateria) dos algoritmos criptográficos em uso no dispositivo móvel real para desta forma ser utilizado como critério na escolha do algoritmo.

O uso da tecnologia TLS para produzir um canal de comunicação seguro, o uso da arquitetura de computação em Nuvem para fornecer elasticidade de recursos e armazenamento ao dispositivo móvel, e o uso do mecanismo em dispositivos móveis reais, são atributos integrados a partir da contribuição de todos os autores da tabela, que visa a fornecer um mecanismo que abranja em sua totalidade estas características.

As contribuições de cada autor influenciaram o desenvolvimento dessa proposta (técnicas de segurança ciente do contexto propostas para dispositivos ao enviar dados ao ambiente de Computação em Nuvem), visto que em todos os trabalhos há uma nítida importância em economizar os recursos dos dispositivos móveis ao fornecer segurança.

Tabela 3.1: Relação dos trabalhos relacionados com o mecanismo de segurança ciente do contexto para computação em nuvem móvel.

A.P	**A.D**	**N.S**	**C.I**	**Fuzzy**	**A.A.C**	**TLS**	**CN**	**D.M Reais**
Pirmez [37]	✓	✓	✓	-	-	✓	-	✓
An [3]	✓	-	✓	-	-	-	-	✓
Cirqueira [10]	✓	✓	✓	-	✓	-	-	✓
Lee [28]	✓	✓	✓	✓	-	-	-	✓
Moraes [14]	✓	✓	✓	-	✓	✓	-	✓
MSCC	✓	✓	✓	✓	✓	✓	✓	✓

- **A.P** - Abordagens Propostas **A.D** - Adaptação Dinâmica **N.S** - Nível de Segurança
- **C.I** - Coleta de Informações **Fuzzy** - Análise de informações baseado *Fuzzy* **A.A.C** - Avaliação de Algoritmos Criptográficos
- **TLS** - Transport Layer Security **CN** - Computação em Nuvem **D.M Reais** - Dispositivos Móveis Reais.
- **MSCC** - Mecanismo de Segurança Ciente do Contexto para Computação em Nuvem Móvel

3.2.5 Síntese

O mecanismo de segurança ciente do contexto possui adaptação dinâmica, nível de segurança que o trabalho An [3] não possui, coleta de informações, análise das informações baseada em *Fuzzy* que os trabalhos Pirmez [37], An [3], Cirqueira [10] e Moraes [14] não implementam, um avaliador de algoritmos criptográficos desenvolvido algo que os trabalhos Pirmez [37], An [3] e Lee [28] não realizam, o uso de TLS não foi contemplado por An [3], Cirqueira [10] e Lee [28], o uso de computação em nuvem não fora utilizado por Pirmez [37], An [3], Cirqueira [10], Lee [28] e Moraes [14], e o uso de dispositivos móveis reais para experimentos.

4 Implementação e Resultados

Este capítulo aborda a implementação do protótipo para o modelo proposto na pesquisa. Com o objetivo de verificar o funcionamento do modelo proposto, foi implementado o mecanismo para a segurança ciente do contexto, utilizando a linguagem de programação Java para dispositivos móveis que executam a plataforma Android.

Primeiramente, foi necessário conceber, implementar e implantar os componentes do mecanismo de segurança ciente do contexto utilizando a metodologia proposta no trabalho. Com o objetivo de verificar o funcionamento do modelo proposto, foram implementadas técnicas para melhorar a ergonomia do dispositivo móvel fornecendo segurança adaptativa. A construção do ambiente de testes, os detalhes da implementação de cada um dos componentes do mecanismo de segurança ciente do contexto e os resultados obtidos são apresentados a seguir.

4.1 Ambiente de Testes

Para a realização do trabalho foi criado no LABSAC (Laboratório de Sistemas e Arquiteturas Computacionais) da UFMA (Universidade Federal do Maranhão) um ambiente de teste para que o dispositivo móvel envie os dados ao servidor em funcionamento na infra-estrutura em nuvem. O dispositivo móvel utilizado é um tablet Samsung com sistema operacional de fábrica e um servidor com o ambiente de Nuvem Eucalyptus executando a aplicação servidor na máquina virtual instanciada de sistema operacional Ubuntu 14.10. Na Tabela 4.1, apresentam-se as características de *hardwares* e *softwares* utilizados para o ambiente de testes.

A máquina virtual "VMSERVIDOR" possui processamento de 2 núcleos, cedidos pela NUVEM, que fora configurado no ato da sua criação. O dispositivo móvel possui somente o protótipo do mecanismo cliente instalado, além dos *softwares* instalados de fábrica.

Tabela 4.1: Características dos dispositivos do ambiente de testes.

Nome da Máquina	IP	Hardware	Software
NUVEM	192.168.27.80	Intel Core i7 com 3.4 Ghz, RAM 8GB, HD 500 GB	Linux CentOS 12.03 server, EUCALYPTUS
VMSERVIDOR	192.168.27.200	1GB RAM, HD 10GB	Ubuntu 14.10, Java 7, OpenSSH Server, protótipo *AppServidor*
TABLET	192.168.27.84	Dual Core Marvell PXA986 com 1.2Ghz, RAM 1GB, HD 8GB	Android 4.1.2, Softwares de padrão de Fábrica, protótipo fr segurança ciente do contexto

O ambiente de testes com os dispositivos inseridos é conforme a Figura 4.1. A rede utilizada é LABSAC, em que os dispositivos NUVEM e VMSERVIDOR, estão conectados por uma rede virtual interna, o VMSERVIDOR possui um endereço IP externo para que haja a comunicação da *AppServidor* com a aplicação do mecanismo de segurança ciente do contexto, que está sendo executado no dispositivo móvel TABLET, este conectado via um roteador sem fio.

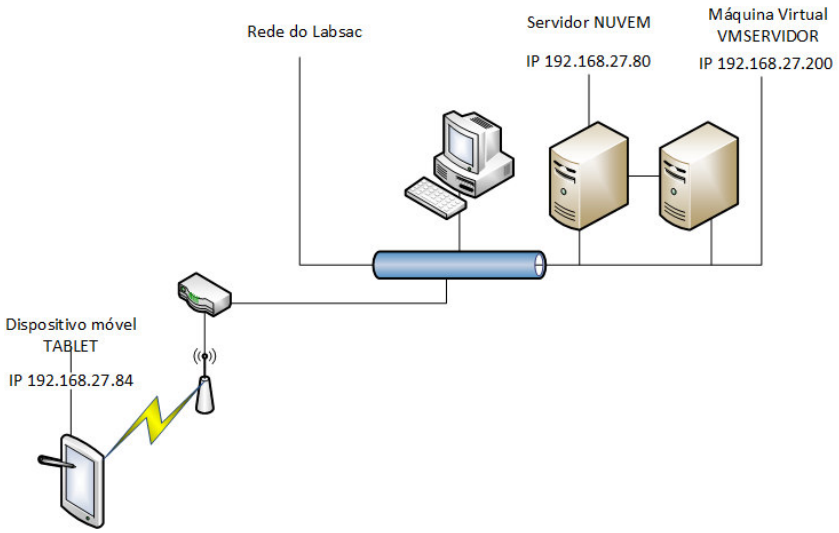

Figura 4.1: Ambiente de testes.

4.2 Implementação dos Protótipos

O principal objetivo da solução proposta é fornecer segurança ciente do contexto para dispositivos móveis ao utilizar o serviço de armazenamento da Nuvem, de forma a aumentar a ergonomia e a segurança dos dispositivos móveis. Os protótipos são divididos em Servidor e aplicação do mecanismo de segurança ciente do contexto para dispositivos móveis. Utilizando o Portecle [39], foram criados quatro

certificados (gerados pelo algoritmo RSA com chaves de 2048 bits de comprimento), dois .p12 (PKCS12), bob (servidor) e alice(cliente), um *truststore* no formato .bks (*Bouncy Castle KeyStore*), formato que o Android suporta, e um *truststorejks* no formato .jks (Java *KeyStore*) para o aplicativo Servidor.

4.2.1 Implementação do aplicativo Servidor

A máquina virtual VMSERVIDOR ao ser criada inicializa-se com um comando, euca-authorize - P tcp -p 8006 default, para permitir o tráfego de dados externos com aplicativo Servidor. Para o aplicativo Servidor poder operar com a criptografia AES chave 256 bits, foi realizado a instalação na máquina virtual VMSERVIDOR do pacote *Java Cryptography Extension* (JCE).

4.2.1.1 Diagrama de classes

As classes, métodos e atributos do aplicativo servidor podem ser visualizados no diagrama de classes da Figura 4.2. O aplicativo Servidor possui 2 classes, *Interface* e *Main*, em que a *Main* possui toda a lógica de negócios, para a troca de dados entre o servidor e o cliente. A classe *Main* possui 3 métodos que são sslInit, *receiveConfigToClient* e *archieveReceiver*.

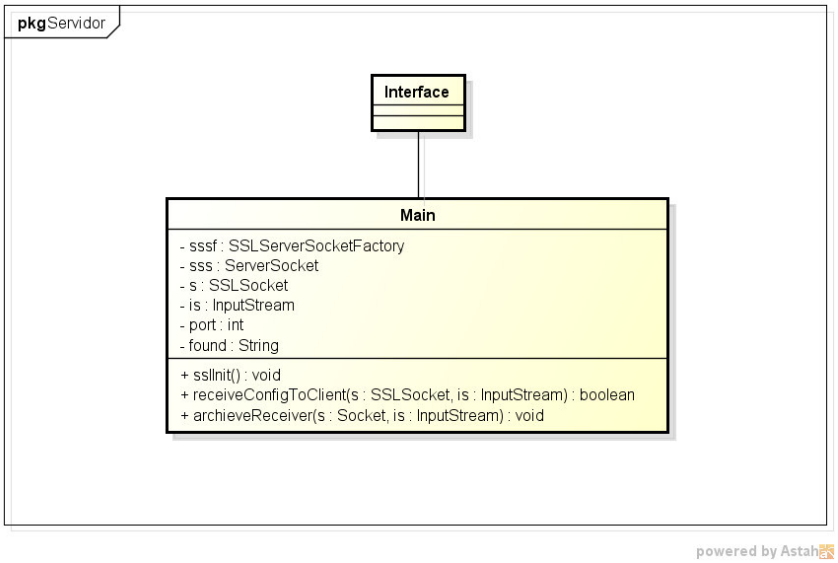

Figura 4.2: Diagrama de classe servidor.

O *sslInit* como descrito na seção anterior, tem como função configurar o *socket* com as credenciais do servidor e cliente, para que haja uma autenticação mútua para realizar uma comunicação segura. Foi utilizado um dos principais pacotes da API JSSE, a *javax.net.ssl*. Conforme pode ser visto na figura 4.3, suas principais classes estão dispostas em ordem lógica para a criação de um *SSLSocket*. Neste método é definida a versão do TLS 1.2, porta 8006, certificados Bob e *truststorejks*.

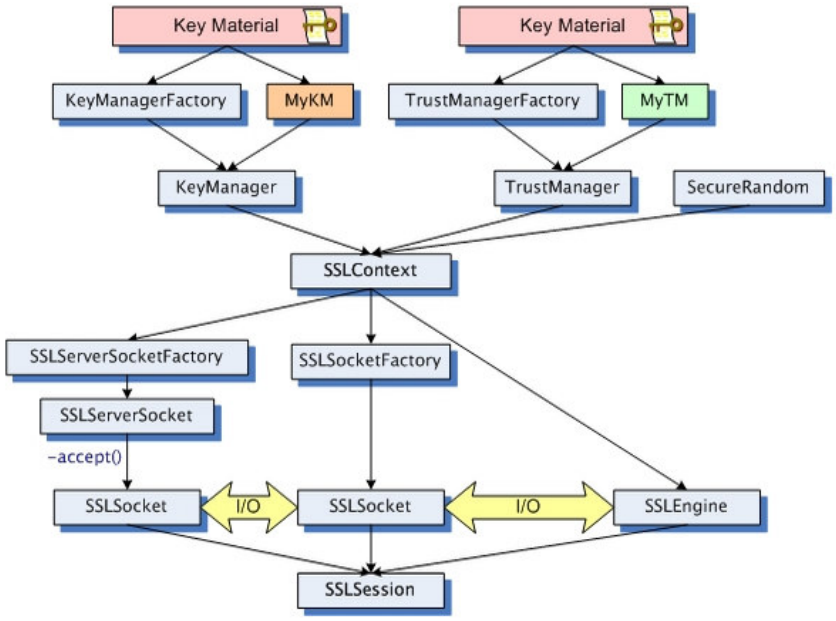

Figura 4.3: Diagrama de relacionamentos das classes da API TLS, Oracle [33].

O certificado Bob é carregado para a classe *KeyStore* e por conseguinte ao *KeyManagerFactory*, o *truststorejks* é carregado ao *TrustManagerFactory*,que é uma fábrica de gestores de confiança. Estas classes são necessárias para inicializar o motor *SSLContext*, com *KeyManager* e *TrustManager*, para a implementação do protocolo de *socket* seguro, Oracle [33]. O *SSLServerSocketFactory*, criado a partir do *SSLContext*, é responsável por encapsular a criação e configuração de *sockets* seguros do servidor, desta forma criando um *socket SSLServerSocket* para quando a conexão for aceita inicialize o *SSLSocket* para estabelecer a comunicação.

O *receiveConfigToClient* recebe o nome do algoritmo de criptografia simétrico que o cliente do dispositivo móvel tomou a decisão a partir do mecanismo de segurança ciente de contexto, além do *SSLSocket*. O algoritmo é repassado ao atributo "*found*" para ser cruzado com uma pré lista de possíveis algoritmos ordenados por nível de segurança de quem possui maior segurança ao menos, AES chave 256 bits,

AES chave 128 bits e RC4 chave 128 bits, respectivamente. A cifra encontrada é habilitada ao *SSLSocket* para inicializar a troca de dados por uma comunicação segura através da atribuição do algoritmo selecionado ao método *setEnabledCipherSuites* e em seguida inicializando o *handshake*. Por fim, o *archieveReceiver* é responsável por receber os *bytes* do arquivo e montá-lo, além de retornar ao final do recebimento à interface que o arquivo fora recebido com sucesso.

4.2.2 Implementação do aplicativo Cliente

Todo o mecanismo de segurança ciente do contexto encontra-se no aplicativo cliente no dispositivo móvel, por este deter recursos limitados que devem ser economizados. A economia destes recursos é direcionada ao nível de bateria, memória e CPU, além de fornecer uma segurança adaptável à este contexto, recursos citados anteriormente, e ao ambiente de rede (público, privado e doméstico) em que está conectado. O dispositivo móvel está conectado via uma rede *wifi* no LABSAC. A função do aplicativo é enviar dados do cliente para o servidor em nuvem provendo confidencialidade, integridade, autenticação e não repudio, com o máximo de ergonomia ao dispositivo móvel. Os detalhes da implementação dos componentes do mecanismo de segurança ciente do contexto são apresentados a seguir.

4.2.2.1 Diagrama de classes

As classes, métodos e os atributos implementados para o mecanismo proposto podem ser visualizados pelo diagrama de classe da Figura 4.4. O diagrama possui as classes *MainActivity*, *Fuzzy*, *StressDisp*, *BatteryInformation*, *Preferences*, *Profile e SocketSSL*. Os detalhes das classes implementadas são apresentados na próxima seção.

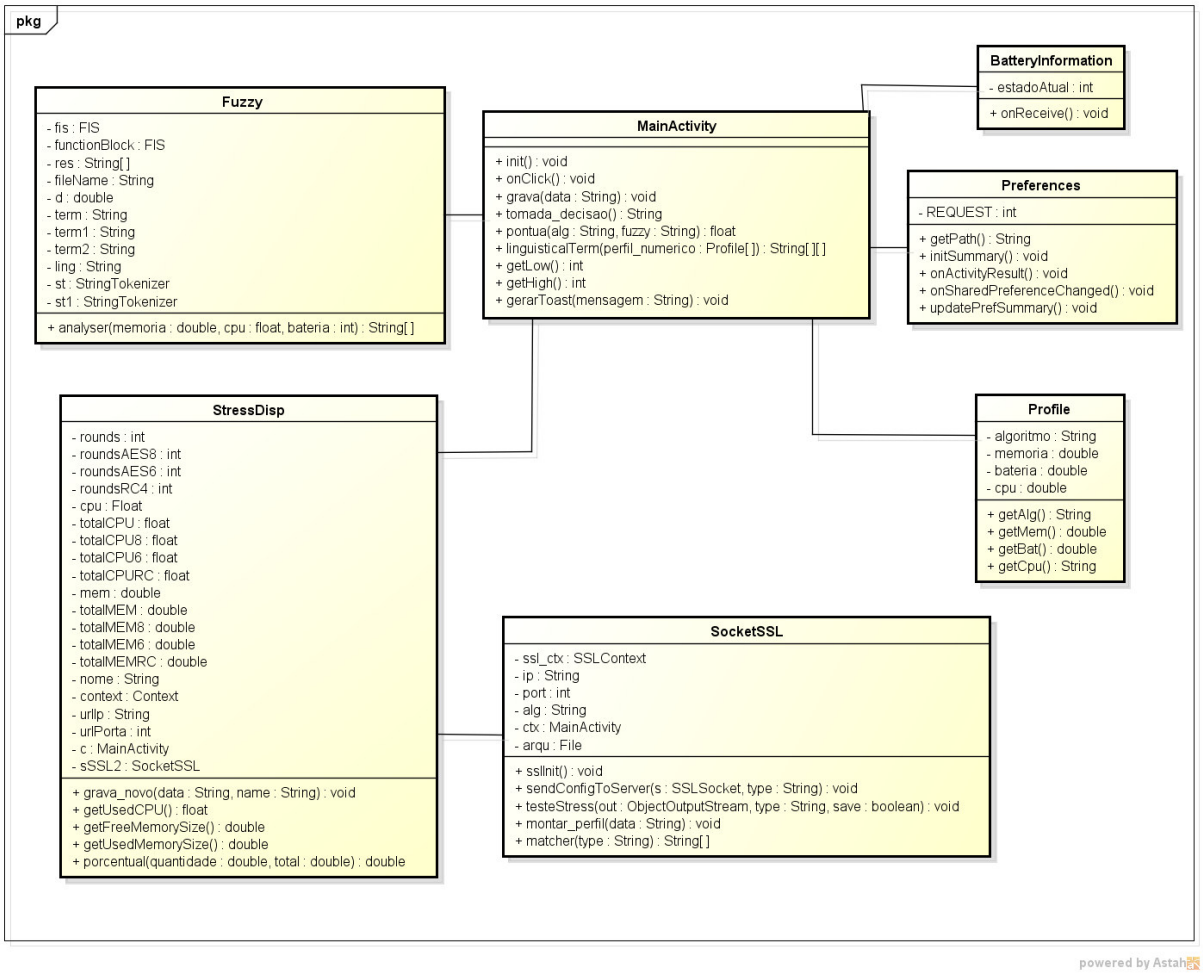

Figura 4.4: Diagrama de classe cliente.

4.2.2.2 Implementação da Classe MainActivity

A classe *MainActivity* possui 9 métodos que são:

- ***init*:** inicializa os botões e textos da interface da aplicação;
- ***onClick*:** é um método associado ao botão "enviar arquivo", que ao ser pressionado obtém as preferências de conexão ip e porta da classe *Preferences*, para serem repassadas à classe *SocketSSL* ou *Stress*Disp;
- **grava:** responsável por salvar os dados do perfil linguístico para análise posterior;
- **tomadaDecisao:** recebe os dados do perfil de cada algoritmo para ser cruzado com a regra retornada pela classe *Fuzzy*. Esta é a regra que melhor representa o contexto atual do dispositivo. Transformando a regra em termos linguísticos para o cruzamento se dar por um método chamado pontua. Aquele algoritmo

simétrico que possuir ao final da soma dos pontos de CPU, memória e bateria, a maior pontuação será o escolhido. Caso haja empate o ambiente de rede previamente fornecido pelo usuário será utilizado para o desempate;

- **pontua:** possui dois parâmetros o do perfil do algoritmo simétrico e o perfil do contexto do dispositivo no momento. Este relaciona da seguinte forma, por exemplo, se o algoritmo simétrico AES_256 possuir perfil ”MEDIUM” e o contexto atual for ”GOOD’ pontua-se 1 por serem correspondentes, caso o contexto fosse ”EXCELLENT” seria 0,5 pelo fato do perfil estar entre o intervalo do correspondente e já se fosse ”LOW” pontuaria 0 por estar fora do intervalo. Esta forma de pontuação seria realizada para os algoritmos simétricos AES_256, AES_128 e RC4;

- **linguisticaTerm:** recebe o perfil numérico da bateria, memória e CPU, de cada algoritmo e os converte em ”HIGH”, ”MEDIUM” e ”LOW”. Por exemplo, caso o valor de memória utilizada do AES_256 seja maior que os demais este possuirá o perfil de algoritmo para memória como ”HIGH”, caso seja menor que o AES_128 e maior que o RC4 será ”MEDIUM” ou menor que todos será ”LOW”.

- **getLow:** calcula qual dos algoritmos é o menor;

- **getHigh:** calcula qual dos algoritmos é maior;

- **gerarToast:** gera uma mensagem na tela do dispositivo móvel.

4.2.2.3 Implementação da Classe *Fuzzy*

A classe *Fuzzy* utiliza a API *jFuzzyLogic* carrega o arquivo android.fcl com as inferências do sistema *Fuzzy*, que possui uma base dados, por exemplo 4.5 que define os termos e os valores do seus conjuntos, e regras. A fuzzificação ocorre com o mapeamento da variável de entrada a um conjunto *Fuzzy* definido no universo da variável correspondente. O sistema de inferência verifica a regra que possui o maior grau de pertinência. Retornando a regra com os seus termos linguísticos correspondentes, para o uso na pontuação. A defuzzificação não foi utilizada, pois o método de pontuação poderia implicar no empate de 2 ou mais algoritmos simétricos.

```
FUZZIFY battery
 TERM poor := (0,0)(11.1,1) (22.2,1) (33.3,0);
 TERM good := (22.2,0) (33.3,1)(55.5,1)(66.6,0);
 TERM excellent := (55.5,0)(66.6,1)(88.8,1)(100.0,1);
END_FUZZIFY
```

Figura 4.5: Base de dados *Fuzzy*.

4.2.2.4 Implementação da Classe *BatteryInformation*

A classe tem como objetivo obter a informação do nível de bateria. Para isso foi criado um *BroadcastReceiver*, que é um receptor de qualquer alteração que fora estabelecido por um filtro, nesse caso em particular o nível de bateria.

4.2.2.5 Implementação da Classe *Preferences*

É responsável por manter as preferências de conexão, como IP e porta do aplicativo servidor, e o endereço do arquivo selecionado para envio. Para isso utiliza-se a classe *SharePreferences* do Android para armazenar as preferências tornando-as persistentes.

4.2.2.6 Implementação da Classe *Profile*

Armazena o perfil do algoritmo simétrico como o seu nome e os valores numéricos de CPU, memória e bateria.

4.2.2.7 Implementação da Classe *SocketSSL*

A classe *SocketSSL* possui os métodos *sslInit*, *sendConfigToServer*, *matcher*, montarPerfil e *testStress*.

- **sslInit:** similar ao método utilizado no aplicativo servidor, mas que neste caso configura o *socket* do aplicativo cliente com as suas credenciais e o do servidor. A credencial cliente é Alice e do servidor é *truststore*, a única diferença é que o Android trabalha com o formato .bks para o *truststore*;

- ***sendConfigToServer*:** envia o nome do algoritmo simétrico ao servidor por um *socket* com algoritmo simétrico *default*, para uma posterior mudança ao enviado;

- ***matcher*:** realiza o cruzamento do algoritmo simétrico informado com a pré lista de algoritmos que possui;

- **montarPerfil:** salva os valores numéricos do perfil de cada algoritmo simétrico no arquivo perfil.

- ***testStress*:** um método para estressar cada algoritmo simétrico, com o objetivo de obter o gasto de uso médio de CPU, memória e bateria. Realiza-se um *loop*, em que a condição de parada é o nível de bateria anterior ao entrar no *loop* é -5% ao atual realizando a troca do algoritmo por outro. Para cada iteração é habilitado o tipo de algoritmo, envia-se o nome do algoritmo pelo *sendConfigToServer* e o arquivo teste.rar de 2MB respectivamente, além de salvar o uso de CPU, memória, *rounds* (número de iterações) e a média destes. O perfil será criado na primeira execução da aplicação servindo como parâmetro de entrada para o método pontua.

4.2.2.8 Implementação da Classe *Stress*Disp

A classe *Stress*Disp analisa o mecanismo ciente do contexto como um todo, desde o teste de *stress* que cria um perfil do desempenho dos algoritmos criptográficos pela primeira vez até a tomada de decisão após a análise de contexto pelo *Fuzzy*. Esta classe tem como objetivo mostrar a economia proporcionada pelo mecanismo ao ser adotado, realizando um comparativo entre uma seleção de algoritmo criptográfico dinâmico (mecanismo de segurança ciente do contexto) e estático.

O estático utilizará o algoritmo simétrico que possui maior segurança que neste caso é o AES_256, visto que possui a maior chave criptográfica. Para este teste define-se a quantidade de bateria em porcentagem que será estressada em 80% segmentada em 8 partes de 10% cada, pois desta forma tem-se uma certeza que o mecanismo vai agir no dispositivo nos 3 níveis definidos no *Fuzzy*.

O teste foi realizado em um ambiente controlado em que somente a aplicação cliente estava em execução, para que não houvesse uma interferência de aplicações no gasto de bateria, memória e CPU, que fosse o aplicativo cliente, além de possuir uma rede exclusiva para a transmissão de dados entre a aplicação cliente

e o servidor sem interferência de outros comunicantes. A class StressDisp possui dois métodos stressaDisp e gravaNovo.

O método stressaDisp inicia a captura das informações do nome do algoritmo, quantidade de rounds, nível de bateria atual, memória e CPU utilizados, para cada algoritmo simétrico selecionado pela tomada de decisão, a partir do processo em execução, salvando-os pelo método gravarNovo em 4 arquivos diferentes, um arquivo com resultado de todas as iterações em ordem de processamento até a condição de parada ser satisfeita, e 3 arquivos correspondente ao número de algoritmos simétricos, por exemplo, o arquivo resultado_stress_dispAES_256 possuirá uma lista de somente informações do algoritmo.

4.3 Teste e Resultados

Para avaliar o mecanismo ciente do contexto com relação a economia de recursos, foi realizado um *stress* de bateria em 80% segmentada em 8 partes como dito anteriormente, executando em um ambiente controlado, em que somente o sistema operacional Android e suas primitivas estariam funcionando, além da aplicação cliente desenvolvida.

O mecanismo possui em sua implementação métodos para fornecer segurança e economia, da melhor forma possível para o contexto que se encontra, Lee [28]. Para a avaliação do comportamento do mecanismo proposto, permaneceu o mesmo utilizado por Pirmez [37], An [3], Cirqueira [10], que utiliza as informações de tempo de ocupação de CPU, memória e gasto de nível de bateria do dispositivo que servirá para montar o perfil dos algoritmos criptográficos como também para demonstrar o uso do mecanismo de segurança ciente do contexto e configuração estática de segurança.

A Figura 4.6 apresenta as configurações para serem preenchidas antes do envio ou recebimento de arquivos como: IP, porta, o arquivo a ser selecionado, seja para enviar ou receber, e o botão de testar, para verificar se o servidor está ativo.

A Figura 4.7 é a tela princial da aplicação cliente onde temos os botões enviar ou receber arquivos previamente selecionados na configuração, além dos ambientes de rede para o usuário informar a qual o dispositivo móvel está conectado.

Figura 4.6: Tela de configurações da aplicação Cliente.

Figura 4.7: Tela principal da aplicação Cliente.

4.3.1 Testes para montar o perfil de cada algoritmo simétrico

A Figura 4.8 mostra o gráfico com o número de rounds(iterações) de envio de arquivo tamanho 2MB por um canal seguro TLS até o consumo de 5% do nível de bateria para cada um dos 3 algoritmos simétricos, AES_256, AES_128 e RC4, sendo executados no dispositivo móvel real para a criação do perfil algorítmico. Verifica-se que o algoritmo simétrico RC4 é o mais econômico por realizar mais rounds que os demais ao custo de 5%. O estabelecimento de conexão é feito uma única vez para cada algoritmo, persistindo a conexão até o consumo de 5% de bateria, realizando um novo estabelecimento de conexão para os demais algoritmos criptográficos restantes.

Figura 4.8: Gráfico de rounds x algoritmos.

Na figura 4.9, demonstra-se a ocupação de memória média para realizar o envio de dados pelo canal seguro utilizado por cada algoritmo. Este gráfico apresenta o AES_256 como o que ocupa mais memória para realizar esta tarefa. Esse tipo de informação é importante para auxiliar na tomada de decisão do mecanismo de segurança ciente do contexto.

O gráfico 4.10, o AES_256 possui o maior tempo de alocação de CPU para realizar a tarefa de envio de dados por um canal seguro, visto que este trabalha com uma chave de 256 bits que demanda um maior esforço para os cálculos que o 128.

O objetivo destes testes é criar um perfil dos algoritmos simétricos para o dispositivo móvel na primeira execução da aplicação, identificando assim quais os algoritmos que consomem ou alocam mais recursos para a tarefa de envio de dados por um canal seguro. Este perfil somente será recriado caso seja excluído pelo usuário.

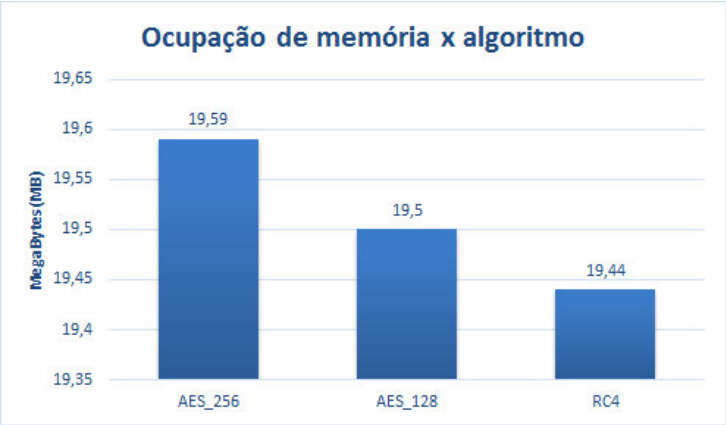

Figura 4.9: Gráfico de ocupação de memória por algoritmo.

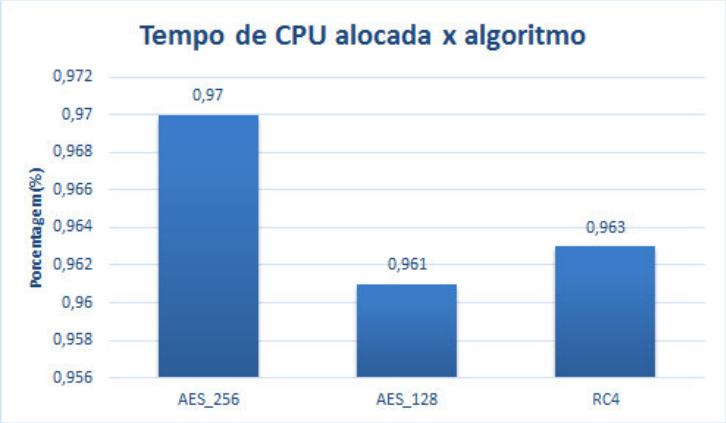

Figura 4.10: Gráfico de tempo de CPU alocada por algoritmo.

No teste realizado temos consumo de bateria, CPU e memória, respectivamente do AES_256 alto, alto e alto, AES_128 médio, baixo e médio, e RC4 baixo, médio e baixo.

4.3.2 Testes do mecanismo de segurança ciente do contexto x configuração estática de segurança

Os testes do mecanismo de segurança ciente do contexto os ambientes de rede (público, privado e doméstico) e a configuração estática de segurança são explicitados no gráfico 4.11. Este gráfico demonstra o número de rounds (iterações) para a tarefa de envio de dados do cliente móvel ao servidor, similar ao teste de montar o perfil algorítmico, a diferença que para ter uma maior precisão dos resultados fez-se um stress de 80% do nível de bateria do dispositivo móvel carregado a 100%. Isto para ter a certeza que a tomada de decisão passará por todos os estados da bateria e não

assumirá o estado crítico que o sistema operacional android firma em menos de 20%, fato constatado por testes.

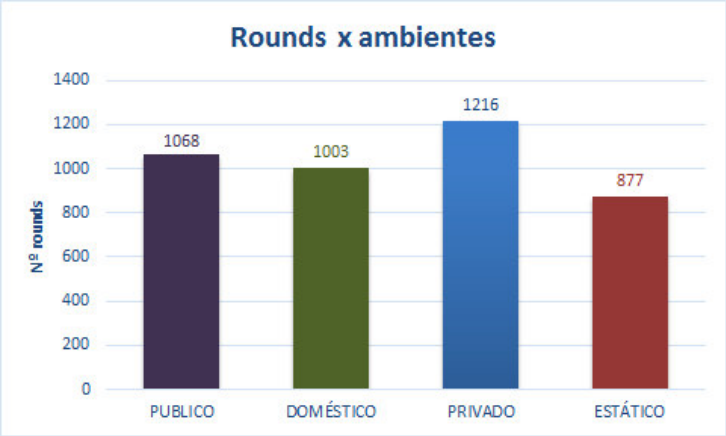

Figura 4.11: Gráfico de rounds x ambientes.

Foi realizado o teste para cada ambiente de rede e estes cruzados com o estático para demonstrar o quanto de economia que o mecanismo poderia prover em relação ao uso de um algoritmo simétrico fixo(estático). Neste gráfico verifica-se que utilizando um algoritmo simétrico fixo, o AES_256, possui um número de rounds inferior aos ambientes de rede público, doméstico e privado do mecanismo de segurança ciente do contexto, afirmando que o mecanismo trabalha mais ao mesmo custo de bateria.

O gráfico 4.12, apresenta o comportamento do mecanismo de segurança no ambiente de rede privado com a menor alocação média de memória para realizar a tarefa de envio de dados, é esperado pelo fato de utilizar mais os algoritmos AES_128 e RC4 que como visto anteriormente ocupam menos memória para suas tarefas.

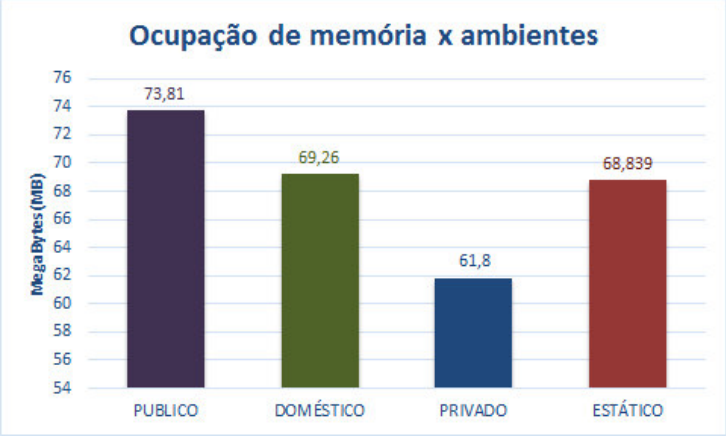

Figura 4.12: Gráfico de ocupação de memória x ambientes.

No gráfico 4.13, o tempo de CPU média alocada ao mecanismo no ambiente de rede público e ao estático são os mais altos, visto que entre todos os ambientes de redes o público é o mais vulnerável verifica-se que o AES_256 aloca mais o tempo de CPU refletindo em sua média para o mecanismo neste ambiente. Os outros são muito baixos constatando que o mecanismo mudando de ambientes de rede pode vir a ter uma melhora significativa na alocação de CPU.

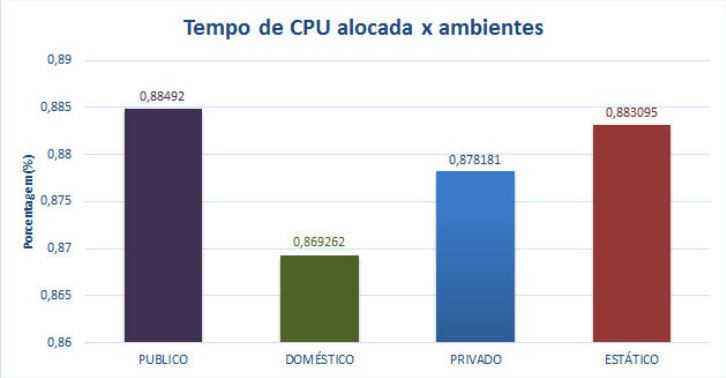

Figura 4.13: Gráfico de tempo de CPU alocada x ambientes.

A Tabela 4.2 apresenta os valores dos recursos consumidos baseado nos gráficos exibidos anteriormente.

Tabela 4.2: Tabela com os valores de cada recurso consumido para a configuração estática e dinâmica.

Medidas	Config. de Seg. Estática	Mecanismo,de Segurança Ciente do Contexto		
		privado	doméstico	público
rounds(n°)	877	1216	1003	1068
CPU(%)	0,883095	0,878181	0,869262	0,88492
memória(MB)	68,839	61,8	69,26	73,81

A Tabela 4.3 resume a economia e gastos que o mecanismo de segurança ciente do contexto possui em relação a configuração de segurança estática. A fórmula utilizada para obter os valores de economia é:

$$E_{recursorede} = (100 - \frac{valorDinamico_{recursorede}}{valorEstatico_{recurso}})$$

A variável "$E_{recursorede}$" é a economia do recurso computacional para o tipo de ambiente de rede que está sendo calculada, que pode ter como recurso CPU,

memória e *rounds* (bateria) e o ambiente de rede como público, privado e doméstico. O "$valorDinamico_{recursorede}$" representa o valor do mecanismo de segurança ciente do contexto para o recurso e ambiente de rede e o "$valorEstatico_{recurso}$" representa o valor da configuração estática para o recurso não é utilizado o ambiente de rede, pois o algoritmo de criptografia estático utilizado é único durante todo o processo seja na rede privada ou doméstica. A fórmula expressa a porcentagem ao subtrair por 100 o resultado obtendo a economia ou gasto do mecanismo de segurança ciente do contexto em relação a configuração estática.

Por exemplo, como apresentado na Tabela 4.2 o valor do recurso de CPU no ambiente privado do mecanismo de segurança ciente do contexto é 0.878181, já o valor do recurso de CPU da configuração estática é 0.883095, estes valores são submetidos a fórmula citada anteriormente para que se obtenha a quantidade de economia ou gasto do mecanismo de segurança ciente do contexto em relação ao estático. Neste exemplo houve uma economia de -0.66, no consumo de CPU; em contrapartida o mesmo recurso no ambiente doméstico teve um gasto de 0.21%, a mais que a configuração estática.

Os valores negativos demonstrados na tabela identificam a economia do recurso pelo uso de um mecanismo de segurança ciente do contexto em relação a configuração estática, os valores positivos representam os gastos que o mecanismo teve a mais em relação ao estático.Nos recursos computacionais de CPU e memória, para encontrar os valores de economia multiplica-se -1 pelo resultado, segue-se a lógica descrita anteriormente, para os números positivos e negativos, pois CPU e memória possuem economia com o menor uso enquanto *rounds* possui economia na lógica que se trabalha mais ao custo de 80% de bateria.

Tabela 4.3: Tabela de economia ou gasto de recursos pelo uso do mecanismo de segurança ciente do contexto para computação em nuvem móvel em relação a configuração estática.

	Consumo(%)		
	privado	doméstico	público
rounds	-38,65%	-14,37%	-21,78%
CPU	-0,66%	-1,57%	0,21%
memória	-10,33%	0,61%	7,22%

4.4 Síntese

Neste capítulo, foi apresentado a implementação do protótipo para o mecanismo de segurança ciente do contexto para dispositivos móveis para computação em nuvem proposto nesta dissertação. Realizaram-se testes nos ambientes montados em laboratório demostrando o consumo e a alocação média dos recursos de bateria, memória e CPU do dispositivo móvel real, a partir do uso do mecanismo para cada ambiente de rede conectado. Estes valores foram cruzados com os da configuração de segurança estática para evidenciar a economia dos recursos pelo mecanismo. No capítulo seguinte apresentaremos a conclusão do trabalho, bem como as sugestões para trabalhos futuros.

5 Conclusões e Trabalhos Futuros

Este capítulo apresenta as contribuições deste trabalho, conclusões sobre os resultados alcançados e sugestões para trabalhos futuros.

5.1 Contribuição do trabalho

Baseado nas pesquisas existentes na área de segurança ciente do contexto, foi proposta, nesta dissertação, como contribuição, um mecanismo de segurança para dispositivos móveis, baseado em TLS associado à análise de contexto em *Fuzzy* para a computação em Nuvem, de forma a melhor identificar os contextos e garantir a confidencialidade e autenticação dos dados de forma econômica. Um avaliador de algoritmos criptográficos para dispositivos móveis, para determinar o gasto médio de recursos, também foi desenvolvido.

O mecanismo proposto foi implementado em camadas, possibilitando uma maior interação entre o avaliador de algoritmos e o analisador de contexto. Além disso, é possível atualizar o sistema de pontuação sem interferir na montagem do perfil algorítmico.

Por fim, obteve-se um protótipo funcional do mecanismo proposto. A principal contribuição é adaptar o grau de segurança ao contexto, com capacidade de coletar dados em tempo real e tomar a decisão do melhor algoritmo simétrico para fornecer segurança sem impactar na ergonomia do dispositivo móvel. Os resultados obtidos nos testes implementados no Capítulo 4 são considerados satisfatórios, tendo em vista que o desempenho apresentado pelo mecanismo com adaptação dinâmica em relação ao estático foi superior.

5.2 Limitações

Nesta proposta foram identificadas as seguintes limitações:

- O mecanismo foi projetado para funcionar com o *download* e *upload* de arquivos ao servidor em nuvem a partir de uma conexão rede persistente. Caso haja uma mudança de conexão de rede durante o envio de dados, o processo deverá ser reiniciado;

- O mecanismo pode ser comprometido internamente, caso seja invadido pelo uso de outra aplicação executada no dispositivo móvel;

- A criptografia de dados é aplicada a cada solicitação de *download* e *upload* de arquivos, de forma persistente até o final da transferência. Caso haja uma mudança de contexto do dispositivo móvel este não será detectado, pois a tomada de decisão é realizada apenas no início da transferência e não durante.

5.3 Trabalhos Futuros

Com o desenvolvimento deste trabalho, foram identificados algumas possibilidades para trabalhos futuros, dentre as quais podemos sugerir:

- Adicionar variáveis de controle de fluxo (*uplink* e *downlink*) e ambiente de rede ao analisador de contexto *Fuzzy*;

- Renegociar em tempo de execução os parâmetros de segurança de acordo com a troca de contexto;

- Retroalimentar o sistema de tomada de decisão;

- Implementar um sistema de tolerância a falhas.

Referências Bibliográficas

[1] G. D. Abowd, A. K. Dey, P. J. Brown, N. Davies, M. Smith, and P. Steggles. Towards a better understanding of context and context-awareness. In *Handheld and ubiquitous computing*, pages 304–307. Springer, 1999.

[2] M. A. AlZain, B. Soh, and E. Pardede. A new model to ensure security in cloud computing services. *Journal of Service Science Research*, 4(1):49–70, 2012.

[3] G. An, G. Bae, K. Kim, and D. Seo. Context-aware dynamic security configuration for mobile communication device. In *New Technologies, Mobility and Security (NTMS), 2009 3rd International Conference on*, pages 1–5. IEEE, 2009.

[4] Android. *Welcome to the Android Open Source Project!* Disponível em: http://source.android.com/. Acessado em: 22 de janeiro de 2015.

[5] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, et al. A view of cloud computing. *Communications of the ACM*, 53(4):50–58, 2010.

[6] E. Barker, W. Barker, W. Burr, W. Polk, and M. Smid. Nist special publication 800-57. *NIST Special Publication*, 800(57):1–142, 2007.

[7] P. Brown, W. Burleson, M. Lamming, O.-W. Rahlff, G. Romano, J. Scholtz, and D. Snowdon. Context-awareness: some compelling applications. In *Proceedings the CH12000 Workshop on The What, Who, Where, When, Why and How of Context-Awareness*, 2000.

[8] R. Buyya, J. Broberg, and A. M. Goscinski. *Cloud computing: Principles and paradigms*, volume 87. John Wiley & Sons, 2010.

[9] G. Chen, D. Kotz, et al. A survey of context-aware mobile computing research. Technical report, Technical Report TR2000-381, Dept. of Computer Science, Dartmouth College, 2000.

[10] A. C. Cirqueira, R. M. Andrade, and M. F. de Castro. Um mecanismo de segurança com adaptação dinâmica em tempo de execução para dispositivos móveis. *Seminário Integrado de Software e Hardware*, pages 1337–1351, 2011.

[11] E. F. DA CRUZ. A criptografia e seu papel na segurança da informção e das comunicações (sic)–retrospectiva, atualidade e perspectiva. 2009.

[12] Q. Dai, H. Yang, Q. Yao, and Y. Chen. An improved security service scheme in mobile cloud environment. In *Cloud Computing and Intelligent Systems (CCIS), 2012 IEEE 2nd International Conference on*, volume 1, pages 407–412. IEEE, 2012.

[13] D. Dev and K. L. Baishnab. A review and research towards mobile cloud computing. In *Mobile Cloud Computing, Services, and Engineering (MobileCloud), 2014 2nd IEEE International Conference on*, pages 252–256. IEEE, 2014.

[14] R. U. M. e Moraes. Ssacc - serviço de segurança para autenticação ciente do contexto: para dispositivos móveis no paradigma da computação em nuvem. Master's thesis, UNIVERSIDADE FEDERAL DO MARANHÃO, 2014.

[15] D. Eastlake and P. Jones. Us secure hash algorithm 1 (sha1), 2001.

[16] N. Fernando, S. W. Loke, and W. Rahayu. Mobile cloud computing: A survey. *Future Generation Computer Systems*, 29(1):84–106, 2013.

[17] A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, and I. Stoica. Above the clouds: A berkeley view of cloud computing. *Dept. Electrical Eng. and Comput. Sciences, University of California, Berkeley, Rep. UCB/EECS*, 28:13, 2009.

[18] A. Freier. The ssl protocol version 3.0. *http://wp. netscape. com/eng/ssl3/draft302. txt*, 1996.

[19] Gartner. *Gartner Says Worldwide Traditional PC, Tablet, Ultramobile and Mobile Phone Shipments On Pace to Grow 7.6 Percent in 2014*. Disponível em: http://www.gartner.com/newsroom/id/2645115. Acessado em: 15 de janeiro de 2015.

[20] F. Gomide, R. R. Gudwin, and R. Tanscheit. Conceitos fundamentais da teoria de conjuntos fuzzy, lógica fuzzy e aplicações. In *Proc. 6 th IFSA Congress-Tutorials*, pages 1–38, 1995.

[21] F. A. C. Gomide and R. R. Gudwin. Modelagem, controle, sistemas e lógica fuzzy. *SBA Controle & Automação*, 4(3):97–115, 1994.

[22] T.-M. Grønli, G. Ghinea, and M. Younas. Context-aware and automatic configuration of mobile devices in cloud-enabled ubiquitous computing. *Personal and ubiquitous computing*, 18(4):883–894, 2014.

[23] L. Guan, X. Ke, M. Song, and J. Song. A survey of research on mobile cloud computing. In *Proceedings of the 2011 10th IEEE/ACIS International Conference on Computer and Information Science*, pages 387–392. IEEE Computer Society, 2011.

[24] A. Khan, M. Othman, S. Madani, and S. Khan. A survey of mobile cloud computing application models. 2013.

[25] N. Kshetri. Privacy and security issues in cloud computing: The role of institutions and institutional evolution. *Telecommunications Policy*, 37(4):372–386, 2013.

[26] T. N. Kudo. Computação ciente de contexto aplicada ao monitoramento de condições críticas em ambientes físicos. Master's thesis, Dissertação (Mestrado em Ciência da Computação)–Departamento de Computação. Universidade Federal de São Carlos, São Carlos, 2004.

[27] J. F. Kurose, K. W. Ross, and A. S. Marques. *Redes de Computadores ea Internet: Uma nova abordagem*, volume 1. Addison Wesley, 2003.

[28] H. Lee and M. Chung. Context-aware security model for social network service. In *Broadband and Wireless Computing, Communication and Applications (BWCCA), 2011 International Conference on*, pages 144–151. IEEE, 2011.

[29] S. Liu, Y. Liang, and M. Brooks. Eucalyptus: a web service-enabled e-infrastructure. 2007.

[30] P. M. Mell and T. Grance. Sp 800-145. the nist definition of cloud computing. Technical report, NIST, Gaithersburg, MD, United States, 2011.

[31] C. Modi, D. Patel, B. Borisaniya, A. Patel, and M. Rajarajan. A survey on security issues and solutions at different layers of cloud computing. *The Journal of Supercomputing*, 63(2):561–592, 2013.

[32] S. Murugesan. Harnessing green it: Principles and practices. *IT professional*, 10(1):24–33, 2008.

[33] Oracle. *Java^{TM} Secure Socket Extension (JSSE)*. Disponível em: http://docs.oracle.com/javase/7/docs/technotes/guides/security/jsse/JSSERefGuide.html. Acessado em: 03 de março de 2015.

[34] N. R. S. Ortega. *Aplicação da Teoria de Conjuntos Fuzzy a problemas da Biomedicina*. PhD thesis, Universidade de São Paulo, 2001.

[35] P. H. Pedrosa and T. Nogueira. Computação em nuvem. *artigo disponível em http://www. ic. unicamp. br/˜ ducatte/mo401/1s2011*, 2, 2010.

[36] F. L. S. Piragibe, R. Cymrot, and A. Sapiro. A importância dada pelos consumidores às questões de consumo eficiente de energia, 2010.

[37] M. Pirmez, L. Pirmez, L. F. R. da Costa Carmo, F. C. Delicato, P. F. Pires, and E. B. de Sousa. Prometheus: Um serviço de segurança adaptativa. *SBSEG2008, Gramado, RS*, 2008.

[38] D. Popa, M. Cremene, M. Borda, and K. Boudaoud. A security framework for mobile cloud applications. In *Roedunet International Conference (RoEduNet), 2013 11th*, pages 1–4. IEEE, 2013.

[39] Portecle. *Portecle*. Disponível em: http://portecle.sourceforge.net/. Acessado em: 30 de janeiro de 2015.

[40] H. Qi and A. Gani. Research on mobile cloud computing: Review, trend and perspectives. In *Digital Information and Communication Technology and it's Applications (DICTAP), 2012 Second International Conference on*, pages 195–202. IEEE, 2012.

[41] S. S. Qureshi, T. Ahmad, K. Rafique, et al. Mobile cloud computing as future for mobile applications-implementation methods and challenging issues. In *Cloud Computing and Intelligence Systems (CCIS), 2011 IEEE International Conference on*, pages 467–471. IEEE, 2011.

[42] M. R. Rahimi, J. Ren, C. H. Liu, A. V. Vasilakos, and N. Venkatasubramanian. Mobile cloud computing: A survey, state of art and future directions. *Mobile Networks and Applications*, 19(2):133–143, 2014.

[43] C. Reimsbach-Kounatze. Towards green ict strategies: Assessing policies and programmes on ict and the environment. Technical report, OECD Publishing, 2009.

[44] R. Rivest. The md5 message-digest algorithm. 1992.

[45] C. Rong, S. T. Nguyen, and M. G. Jaatun. Beyond lightning: A survey on security challenges in cloud computing. *Computers & Electrical Engineering*, 39(1):47–54, 2013.

[46] M. Satyanarayanan. Pervasive computing: Vision and challenges. *Personal Communications, IEEE*, 8(4):10–17, 2001.

[47] B. Schilit, N. Adams, and R. Want. Context-aware computing applications. In *Mobile Computing Systems and Applications, 1994. WMCSA 1994. First Workshop on*, pages 85–90. IEEE, 1994.

[48] B. N. Schilit and M. M. Theimer. Disseminating active map information to mobile hosts. *Network, IEEE*, 8(5):22–32, 1994.

[49] L. F. Soares, D. A. Fernandes, J. V. Gomes, M. M. Freire, and P. R. Inácio. Cloud security: State of the art. In *Security, Privacy and Trust in Cloud Systems*, pages 3–44. Springer, 2014.

[50] F. R. Sousa, L. O. Moreira, and J. C. Machado. Computação em nuvem: Conceitos, tecnologias, aplicações e desafios. *Ceará: Universidade Federal do Ceará*, 2009.

[51] L. B. SOUSA. Redes de computadores: guia total. *São Paulo: Editora Érica Ltda*, 2009.

[52] R. F. Weber. *Utilização da Tecnologia J2ME para Desenvolvimento de Sistemas de M-Banking*. PhD thesis, UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL, 2008.

[53] S. William and W. Stallings. *Cryptography and Network Security, 4/E*. Pearson Education India, 2006.

[54] Q. Zhang, L. Cheng, and R. Boutaba. Cloud computing: state-of-the-art and research challenges. *Journal of internet services and applications*, 1(1):7–18, 2010.

A Apêndice

A.1 Regras Fuzzy

Regras Fuzzy definidas no arquivo android.fcl.

RULE 1: IF battery IS poor AND cpu IS critical AND memory IS low THEN status IS down;

RULE 2: IF battery IS poor AND cpu IS critical AND memory IS half THEN status IS down;

RULE 3:IF battery IS poor AND cpu IS critical AND memory IS full THEN status IS floor;

RULE 4:IF battery IS poor AND cpu IS medium AND memory IS low THEN status IS down;

RULE 5:IF battery IS poor AND cpu IS medium AND memory IS half THEN status IS floor;

RULE 6:IF battery IS poor AND cpu IS medium AND memory IS full THEN status IS up;

RULE 7:IF battery IS poor AND cpu IS confort AND memory IS low THEN status IS down;

RULE 8:IF battery IS poor AND cpu IS confort AND memory IS half THEN status IS floor;

RULE 9:IF battery IS poor AND cpu IS confort AND memory IS full THEN status IS up;

RULE 10:IF battery IS good AND cpu IS critical AND memory IS low THEN status IS down;

RULE 11:IF battery IS good AND cpu IS critical AND memory IS half THEN status IS floor;

RULE 12:IF battery IS good AND cpu IS critical AND memory IS full THEN status IS floor;

RULE 13:IF battery IS good AND cpu IS medium AND memory IS low THEN status IS down;

RULE 14:IF battery IS good AND cpu IS medium AND memory IS half THEN status IS floor;

RULE 15:IF battery IS good AND cpu IS medium AND memory IS full THEN status IS up;

RULE 16:IF battery IS good AND cpu IS confort AND memory IS low THEN status IS floor;

RULE 17:IF battery IS good AND cpu IS confort AND memory IS half THEN status IS floor;

RULE 18:IF battery IS good AND cpu IS confort AND memory IS full THEN status IS up;

RULE 19:IF battery IS excellent AND cpu IS critical AND memory IS low THEN status IS down;

RULE 20:IF battery IS excellent AND cpu IS critical AND memory IS half THEN status IS floor;

RULE 21:IF battery IS excellent AND cpu IS critical AND memory IS full THEN status IS up;

RULE 22:IF battery IS excellent AND cpu IS medium AND memory IS low THEN status IS down;

RULE 23:IF battery IS excellent AND cpu IS medium AND memory IS half THEN status IS floor;

RULE 24:IF battery IS excellent AND cpu IS medium AND memory IS full THEN status IS up;

RULE 25:IF battery IS excellent AND cpu IS confort AND memory IS low THEN status IS down;

RULE 26:IF battery IS excellent AND cpu IS confort AND memory IS half THEN status IS floor;

RULE 27:IF battery IS excellent AND cpu IS confort AND memory IS full THEN status IS up;