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

RENATO UBALDO MOREIRA E MORAES

SSACC -SERVIÇO DE SEGURANÇA PARA AUTENTICAÇÃO CIENTE DO CONTEXTO: para Dispositivos Móveis no Paradigma da Computação em Nuvem

São Luís - MA
2014

RENATO UBALDO MOREIRA E MORAES

SSACC -SERVIÇO DE SEGURANÇA PARA AUTENTICAÇÃO CIENTE DO CONTEXTO: para Dispositivos Móveis no Paradigma da Computação em Nuvem

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

2014

Moreira e Moraes, Renato Ubaldo

SSACC -SERVIÇO DE SEGURANÇA PARA AUTENTICAÇÃO CIENTE DO CONTEXTO: para Dispositivos Móveis no Paradigma da Computação em Nuvem / Renato Ubaldo Moreira e Moraes. – São Luís - MA, 2014.

120 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, 2014.

1. Criptografia 2. Computação em Nuvem. 3. Ciência do Contexto. 4. SSL 5. Computação Verde I. Abdelouahab, Zair, orient. II. Título.

CDU 004.056.55

SSACC - SERVIÇO DE SEGURANÇA PARA AUTENTICAÇÃO CIENTE DO CONTEXTO: PARA DISPOSITIVOS MÓVEIS NO PARADIGMA DA COMPUTAÇÃO EM NUVEM

Renato Ubaldo Moreira e Moraes

Dissertação aprovada em 26 de setembro de 2014.

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

Profa. Christiane Ferreira Lemos, Dra.
(Membro da Banca Examinadora)

Prof. Sofiane Labidi, Dr.
(Membro da Banca Examinadora)

À minha mãe, ao meu pai, à minha noiva, e aos meu irmãos, aqueles pelo qual luto todos os dias da minha vida.

Resumo

Atualmente houve uma adesão em massa aos dispositivos móveis inteligentes, conhecidos como *smartphones*, e, com essa adesão, houve consequentemente um grande aumento no consumo da informação, principalmente proveniente da internet. Para atender a grande demanda de acesso à informação foi criado inúmeros artifícios para facilitar tanto o acesso, quanto a criação e o armazenamento dessas informações, dentre os mais conhecidos e difundidos atualmente está a computação em nuvem. A informação assume, hoje em dia, uma importância crescente e até vital para algumas entidades, e com tamanho valor acaba se tornando muito desejada, sendo muitas vezes alvo de tentativas de captura e espionagem. Para se obter dados de informações confidenciais *hackers* usam inúmeros artifícios, e o mais usado é o *scan* de redes, que em outras palavras pode ser descrito *scan* como "notificações de varreduras em redes de computadores, com o intuito de identificar quais computadores estão ativos e quais serviços estão sendo disponibilizados por eles. É amplamente utilizado por atacantes para identificar potenciais alvos, pois permite associar possíveis vulnerabilidades aos serviços habilitados em um computador" [10]. De acordo com o [10] o número de ataques só vem crescendo a cada ano como mostra a figura 1.1 e 1.2 que estão na seção 1.1. Com base nesse alto número de incidentes, o crescimento do consumo da informação por meio de dispositivos móveis e a necessidade de melhorar gastos de energia, a proposta de criação do Serviço de Segurança para Autenticação Ciente do Contexto (SSACC) é necessária para a atualidade. O SSACC tem como principal objetivo fornecer um canal seguro para transferência de arquivos para um servidor, fazendo uso de informações de contexto e diminuindo o desperdício de energia, consequentemente economizando recursos e se enquadrando à Computação Verde. Feito com base no *Secure Socket Layer*(SSL), que é um "protocolo amplamente utilizado que fornece comunicação segura através de uma rede. Ele usa vários processos criptográficos diferentes para garantir que os dados enviados por meio de rede são seguras. Ele fornece um aprimoramento de segurança para o protocolo *Transport Control Protocol* (TCP)/ *Internet Protocol* (IP) padrão, que é usado para comunicação com a *internet*.

SSL utiliza criptografia de chave pública para fornecer autenticação. O protocolo SSL também usa criptografia de chave privada e assinaturas digitais para garantir a privacidade e a integridade dos dados" [26].

Palavras-chaves: Cirptografia. Computação em Nuvem. Ciência do Contexto. SSL. Computação Verde.

Abstract

Nowadays, there was a massive inclusion of smart mobile devices, known as smartphones, and with this accession, there‘s consequently a large increase in the consumption of information, especially from the internet. To support the great demand for information access, it‘s created a numerous devices to facilitate both access, the creation and the storage of such information, among the best known and disseminated currently is cloud computing. The feedback takes currently, an increasingly important and even critical for some entities, size and value turns out to be very desirable. Being often target capture and espionage attempts. To obtain data confidential information hackers use numerous devices, and more is used to scan networks. In other words can be described as scan "Scans notifications in computer networks, in order to identify which computers are active and which services are available for them. It is widely used by attackers to identify potential targets because it allows associate potential vulnerabilities to services enabled on a computer " [10]. According to [10] the number of attacks has been widening each year as shown in Figure 1.1 and 1.2 which are in section 1.1. Based on this high number incidents, the growth of the information consumer by means of devices furniture and the need to improve energy costs, the proposed establishment of the Office Security for Context Aware of authentication (Serviço de Segurança para Autenticação Ciente do Contexto(SSACC)) is required for today. The ssacc focus to provide a secure channel for transfer files to a server, using context information and reducing energy waste, thus saving resources and framing the Green Computing. Made based on the Secure Socket Layer (SSL), which is a ’widely used protocol that provides secure communication through a network. It uses several different cryptographic processes to ensure that data sent through the network is secure. It provides a security enhancement for the Transport Control Protocol (TCP) / Internet Protocol (IP) standard, which is used for communication with the Internet. SSL uses public key cryptography to provide authentication. The SSL protocol also uses encryption of the private key and digital signatures to ensure privacy and the integrity of data " [26].

Keywords: Cryptography. Cloud Computing. Context Aware. SSL. Green Computing.

Agradecimentos

A Deus, por me proporcionar a sabedoria e a oportunidade de realizar um sonho como este, ter o título de Mestre.

Ao meu orientador, Prof. Ph.D. Zair Abdlouahab, por me confiar, incentivar e orientar nessa jornada que é o mestrado, foi de fundamental importância para a conclusão desta dissertação.

Ao meu pai, João Ubaldo, e minha mãe, Maria das Dores, por sempre me puxarem a orelha pro lado certo, sempre me botarem na linha, e aos apoios incondicionais concedidos e meus irmãos, Ricardo e Rafael, que também sempre estiveram ao meu lado e ajudaram no que fosse preciso, inclusive incentivando.

À minha noiva, Priscila Costa, que por muitas vezes me ajudou, me levantou quando cai, me incentivou, sempre acreditou em mim até mesmo quando eu não cheguei a acreditar, pelo carinho, compreensão e os apoios incondicionais concedidos.

Ao meu amigo Cláudio Aroucha, que me ajudou, clareou a caminhada e sempre incentivou e ao Mário que ajudou diretamente no desenvolvimento dos aplicativos.

Aos meus colegas de laboratório, Josenilson, Eduardo, Mauro, Luiz, Dhully, Gleison, Yuri, Higor, Willian, Steve, Bruno, Luan, Leonardo, Marcos pelo companheirismo e auxílio técnico.

Por fim e não menos importante, agradecer à CAPES pelo auxílio e ao CPGEE e todos os docentes que compõe o programa e que proporcionou toda essa experiência incrível e todos aqueles que de alguma forma contribuíram com meu trabalho e que certamente jamais serão esquecidos.

"O mundo não está preparado para isso. É algo muito além do nosso tempo, mas as leis vão prevalecer, e um dia farão um sucesso triunfante."

Nikola Tesla

Lista de Figuras

1.1 Incidentes reportados ao CERT.br - Janeiro a Dezembro de 2013 20

1.2 Gráfico linear de incidentes reportados ao CERT.br - Janeiro a Dezembro de 2013 21

1.3 Estatísticas de incidentes reportados ao CERT 21

2.1 Modelos de serviços da computação em nuvem. 26

2.2 Camadas da infraestrutura física da computação em nuvem. 28

2.3 Modelo visual de definição de computação em nuvem do *National Institute of Standards and Technology* (NIST). 29

2.4 Tipos de modelos de implementação de nuvem 30

2.5 Diagrama de exemplo de uso da Ciência do Contexto 37

3.1 Diagrama de camadas do SSACC disposto com SSL 49

3.2 Diagrama de sequência do App stress.SSACC 51

3.3 Diagrama de sequência de upload do App SSACC 52

3.4 Diagrama de sequência de download do App SSACC 52

3.5 Descrição do Handshake SSL 55

4.1 Diagrama do ambiente físico usado para testes dos aplicativos 57

4.2 Fluxograma do serviço da nuvem 58

4.3 Fluxograma do aplicativo *stress*.SSACC 59

4.4 Fluxograma do aplicativo SSACC 63

Lista de Tabelas

2.1 Tabela comparativa do exemplo da aplicação de mensagem ciente do contexto 35

4.1 Tabela de descrição dos dispositivos de teste 57

4.2 Média dos resultados de testes feito no DM 1 62

4.3 Média dos resultados de testes feito no DM 2 62

4.4 Algoritmo de decisão 62

Lista de Siglas

3DES	*Triple Data Encryption Standard*.
ADT	*Android Development Tools*.
AES	*Advanced Encryption Standard*.
API	*Application Programming Interface*.
APP	Aplicativos (*Application*).
CC	Ciência do Contexto.
CERT.br	Centro de Estudos, Resposta e Tratamento de Incidentes de Segurança no Brasil.
CN	Computação em Nuvem.
CPU	*Central Processing Unit*.
CV	Computação Verde.
DM	Dispositivo Móvel.
EPA	*Environmental Protection Agency*.
EUA	Estados Unidos da América.
IaaS	*Infrastructure as a Service*.
Java EE	*Java Platform Enterprise Edition*.
LABSAC	Laboratório de Sistemas em Arquiteturas Computacionais.
NIST	*National Institute of Standards and Technology*.

PaaS	*Platform as a Service*.
RC4	*Rivest Cipher* 4.
SaaS	*Software as a Servic*.
sdCard	*Security Digital Card*.
SSACC	Serviço de Segurança para Autenticação Ciente do Contexto.
SSL	*Secure Socket Layer*.
TI	Tecnologia da Informação.
TSL	*Transport Layer Security*.
UFMA	Universidade Federal do Maranhão.
XML	*eXtensible Markup Language*.

Sumário

Lista de Figuras **xii**

Lista de Tabelas **xiii**

Lista de Siglas **xiv**

1 Introdução **19**

1.1 Motivação e Justificativa 19

1.2 Objetivos 22

1.2.1 Objetivo Gerais 22

1.2.2 Objetivos Específicos 22

1.3 Estrutura da Dissertação 23

2 Fundamentação Teórica **24**

2.1 Computação em Nuvem 24

2.1.1 Modelos de serviços da computação em nuvem 25

2.1.2 Arquitetura da computação em nuvem 27

2.1.3 Características Essenciais da computação em nuvem 28

2.1.4 Modelos de implementação da computação em nuvem 29

2.2 Computação Ciente do Contexto para Adaptação de Software 31

2.3 Segurança 37

2.3.1 Visão Geral 37

2.3.2 Segurança em Nuvem 40

2.3.3 SSL 41

2.4 Computação Verde 43

2.5 Conclusões 46

3 Mecanismo **48**

3.1 Características Gerais 48

3.2 Arquiteturas dos Aplicativos 50

3.2.1 Arquitetura do Aplicativo de *Stress* 50

3.2.2 Arquitetura do Aplicativo SSACC 51

4 Implementações e Resultados **56**

4.1 Ambiente Usado 56

4.2 Implementação do Serviço da Nuvem 57

4.2.1 Implementação do Método *sslInit* 58

4.3 Implementação do Teste de Stress 58

4.3.1 Implementação do Método onClick 59

4.3.2 Implementação da Classe *SocketSSACC.java* 60

4.3.3 Implementação da Classe de Contexto de Bateria 61

4.3.4 Implementação do *AndroidManifest* 61

4.4 Resultados do Teste de *Stress* 61

4.5 Implementação do Aplicativo Final 63

4.5.1 Implementação da classe *SocketSSL*.java 64

4.5.2 Implementação da classe *ChooseThePatch*.java 64

5 Conclusões e Trabalhos Futuros **65**

5.1 Contribuição do trabalho 65

5.2 Limitação 66

5.3 Trabalhos Futuros 66

Referências Bibliográficas **68**

A Apêndice **73**

A.1 Códigos do Serviço da Nuvem 73

A.1.1 Código do Servidor.java 73

A.2 Códigos do Aplicativo *stress*.SSACC 80

A.2.1 Código da MainActivity.java 80

A.2.2 Código de SocketSSL.java 86

A.2.3 Código de BatteryInformation.java 97

A.2.4 Código de ArchiveTestsSaver.java 99

A.2.5 Código de AndroidManifest.xml 102

A.3 Código do Aplicativo Final SSACC 104

A.3.1 Código de MainActivity.java 104

A.3.2 Código de SocketSSL.java 110

A.3.3 Código de BatteryInformation.java 117

A.3.4 Código de ChooseThePath.java 119

A.3.5 Código de AndroidManifest.xml 120

1 Introdução

Esta dissertação apresenta um modelo de serviço de autenticação segura baseado na ciência do contexto para Dispositivo Móvel (DM). O foco principal do desenvolvimento do serviço visa fornecer segurança na autenticação baseada no contexto no qual o DM está inserido, levando em conta o consumo da criptografia selecionada, o tipo de segurança de conexão à *internet*, a quantidade de bateria disponível e o poder de processamento do DM.

Este capítulo apresenta uma visão geral do trabalho. A seção 1.1 apresenta a motivação e justificativa do trabalho. Os objetivos específicos e geral estão descritos na seção 1.2, e, por fim, a Seção 1.3 apresenta a organização da estrutura desta dissertação.

1.1 Motivação e Justificativa

"Podemos dizer que Computação em Nuvem (CN) é um termo para descrever um ambiente de computação baseado em uma imensa rede de servidores, seja estes virtuais ou físicos. Uma definição simples pode então ser um conjunto de recursos como capacidade de processamento, armazenamento, conectividade, plataformas, aplicações e serviços disponibilizados na internet. O resultado é que a nuvem pode ser vista como o estágio mais evoluído do conceito de virtualização, a virtualização do próprio data center" [41].

Com a facilidade que este conceito disponibiliza no mercado o acesso rápido, fácil e relativamente barato de serviços essenciais, principalmente às empresas, acabou tornando-se um grande aliado aos negócios. Com a CN as aplicações, os arquivos e qualquer outros dados relacionados não estarão na máquina local. Estes serviços estarão disponíveis nas nuvens, ou seja, estarão disponíveis na internet para acesso rápido e fácil do cliente a partir de qualquer máquina.

Além disso, com a disponibilidade deste serviço tem-se opções como escalabilidade, podendo aumentar o armazenamento ou poder de processamento mediante necessidade, implicando diretamente na redução de custos para o

contratante, o qual somente irá pagar pela quantidade usada e não terá que manter uma estrutura que por vezes é maior que a demanda, tornando maior os gastos com poder de hardware, consumo de energia, e espaço para armazenamento.

De acordo com [10], os maiores número de incidentes reportados a eles são classificados como *scan*, assim como mostra a figura 1.1 e 1.2 de [10]. "Varredura em redes, ou *scan*, é uma técnica que consiste em efetuar buscas minuciosas em redes, com o objetivo de identificar computadores ativos e coletar informações sobre eles como, por exemplo, serviços disponibilizados e programas instalados. Com base nas informações coletadas é possível associar possíveis vulnerabilidades aos serviços disponibilizados e aos programas instalados nos computadores ativos detectados"de acordo com [9].

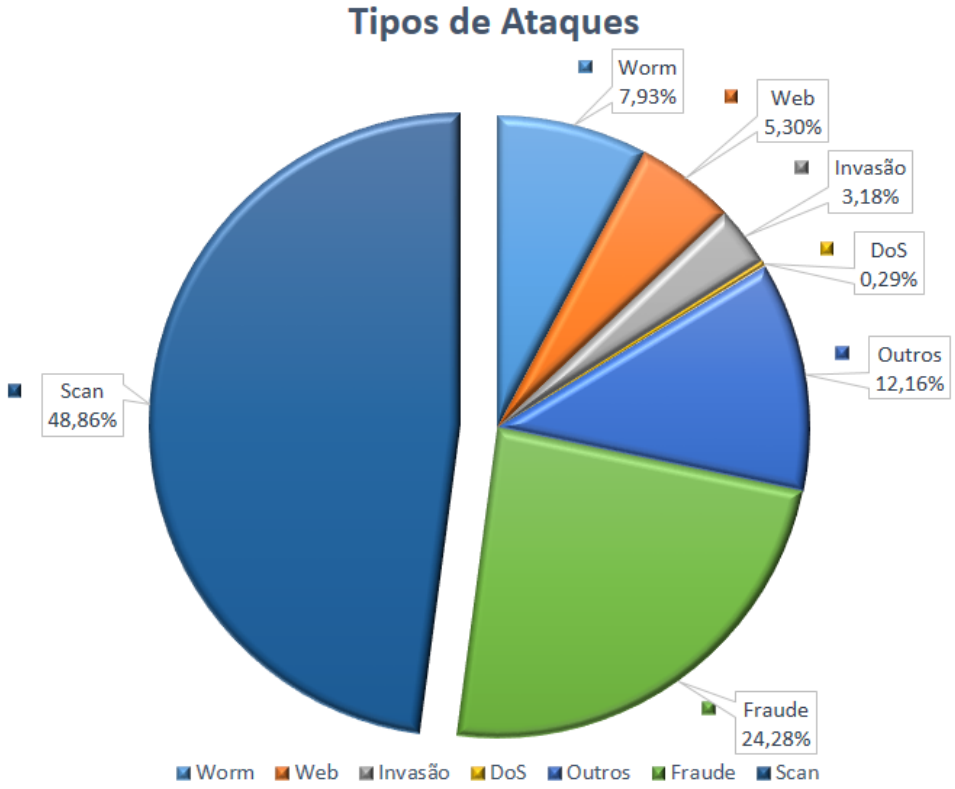

Figura 1.1: Incidentes reportados ao CERT.br - Janeiro a Dezembro de 2013

Foram reportados ao Centro de Estudos, Resposta e Tratamento de Incidentes de Segurança no Brasil (CERT.br) inúmeros casos de ataques, e no decorrer dos anos, os índices só aumentavam assim como mostra a figura 1.3 de [10].

"A ideia da Computação Verde (CV) começou em 1992, quando a *Environmental Protection Agency* (EPA) dos Estados Unidos da América (EUA) lançou

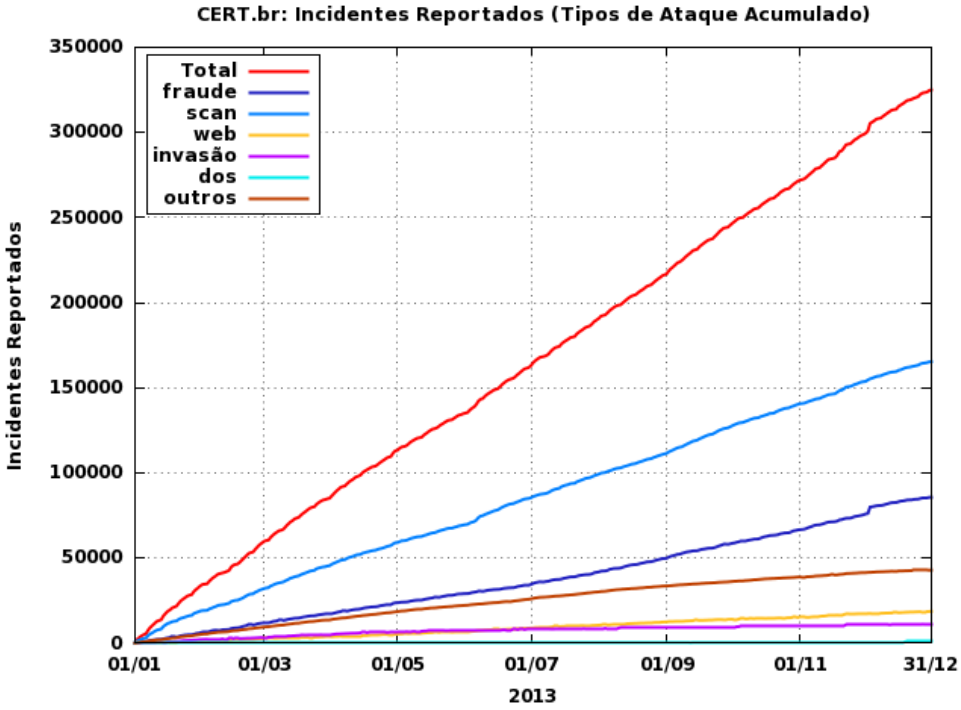

Figura 1.2: Gráfico linear de incidentes reportados ao CERT.br - Janeiro a Dezembro de 2013

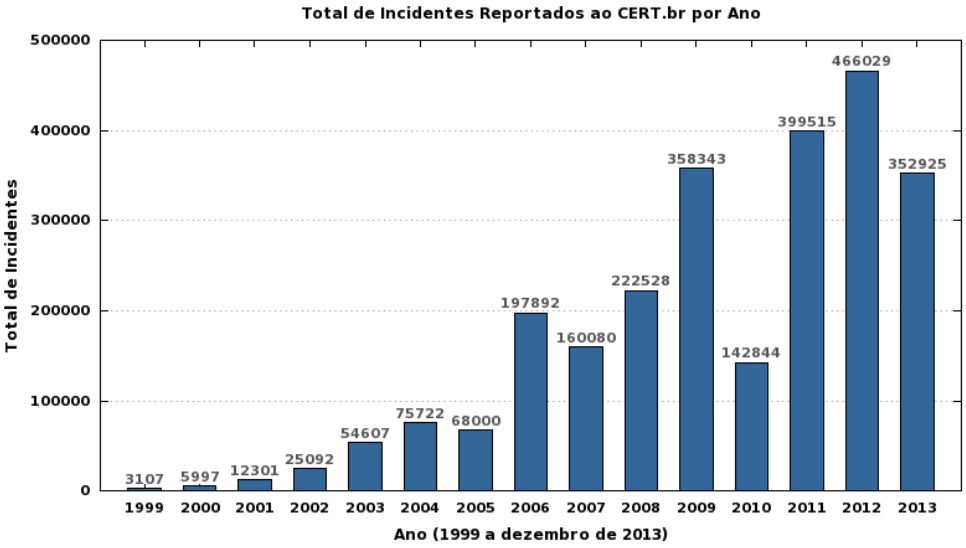

Figura 1.3: Estatísticas de incidentes reportados ao CERT

Energy Star, uma abordagem de rotulagem voluntária para reconhecer características de eficiência energética em equipamentos eletrônicos. *Energy Star* tornou-se uma importante certificação, com o reconhecimento do nome significativo nos EUA e em outros países. Hoje, servidores, laptops, sistemas de jogos, e muitas outras ofertas de equipamentos de TI incluem o selo *Energy Star* nas descrições dos seus produtos" [34].

A ideia do selo *Energy Star* se popularizou tanto que nos dias atuais ele esta presente em vários produtos, e em várias partes do mundo. No cenário atual é comum

noticiarem práticas de metodologias ecologicamente corretas e preocupações com as manufaturas de inúmeros produtos, e em diversas áreas, incluído principalmente os produtos voltados para as áreas da ciência. O desenvolvimento de *software* também integrou em sua manufatura a chamada CV, fazendo com que tenhamos uma evolução dos algoritmos para que sejam mais eficientes. Isso implica diretamente na rapidez com que o *software* é executado e evitando desperdício de energia com processamento desnecessário nos equipamentos. Portanto a CV tem impacto direto no desenvolvimento de *software* [17].

A justificativa para este trabalho é a criação de um serviço que visa fornecer uma adaptação dinâmica, em tempo de execução, para tomar a decisão de qual algoritmo de criptografia é seguro o suficiente para que haja a autenticação no ambiente em que os DM estão inseridos. Garantindo sigilo na autenticação do cliente à nuvem e menor gasto de energia em situações mais críticas em relação a disponibilidade de energia.

1.2 Objetivos

1.2.1 Objetivo Gerais

A principal finalidade do desenvolvimento desta dissertação é fornecer um serviço que garantirá ao cliente segurança ao fazer uma autenticação em um serviço na nuvem, sem que tenha desperdício desnecessário de energia e alocação exagerada da memória, com base na tomada de decisão da criptografia que será usada em relação à banda e energia disponíveis no DM.

1.2.2 Objetivos Específicos

Para alcançar o objetivo geral, foram definidos os seguintes objetivos específicos:

1. Pesquisa aprofundada sobre os seguintes temas:

   Computação em Nuvem.

   Segurança Adaptativa.

Computação verde.

Criptografia.

2. Definição do problema alvo.
3. Implementação do aplicativo de análise de eficiência dos algoritmos de criptografia.
4. Análise do relatório do aplicativo anterior.
5. Definição da tomada de decisão.
6. Implementação do aplicativo com o serviço de segurança.

1.3 Estrutura da Dissertação

Este trabalho está organizado na seguinte forma:

O capítulo 1 explicita os assuntos fundamentais da dissertação de forma resumida, mostrando também o problema a ser resolvido, os objetivos e a estrutura da dissertação.

O capítulo 2 mostra uma visão geral sobre computação em nuvem, ciência do contexto para adaptação de *software* e computação verde, destacando as principais características de cada assunto. Por fim, são mostrados trabalhos relacionados à dissertação e conclusões.

O capítulo 3 apresenta o mecanismo proposto para que possa atingir os objetivos de prover uma segurança adaptativa visada na computação verde, dividida em mais duas subseções, as características e as arquiteturas.

O capítulo 4 apresenta as implementações e os resultados dos testes.

O capítulo 5 apresenta a conclusão do trabalho e possíveis trabalhos futuros.

2 Fundamentação Teórica

Este capítulo visa apresentar os conceitos básicos das tecnologias que foram aplicadas ao desenvolvimento do projeto. Esses conceitos formam a base principal, e a integração dos mesmos foi o que possibilitou o surgimento do serviço criado no Laboratório de Sistemas em Arquiteturas Computacionais (LABSAC) da Universidade Federal do Maranhão (UFMA), portanto estes conceitos são a base de todo o projeto Serviço de Segurança para Autenticação Ciente do Contexto (SSACC), sendo assim extremamente importante conceituá-los.

2.1 Computação em Nuvem

A CN tornou-se uma das abordagens no mundo da tecnologia mais utilizada nos dias de hoje. Uma de suas características essenciais é a forma de oferecer seus serviços, os serviços fornecidos pelas nuvem tem uma grande vantagem quando comparados com os serviços tradicionais, a adesão dessa tecnologia proporciona ao usuário final uma redução dos investimentos iniciais, um maior desempenho na realização das tarefas, uma alta disponibilidade, escalabilidade, dentre outras vantagens. Em virtude das suas vantagens a computação em nuvem ganhou espaço no meio das empresas de Tecnologia da Informação (TI) [44] [2].

No entanto, a adesão das nuvem implica na transferência do controle dos dados do usuário, ou seja, toda a infraestrutura ficará sob controle do provedor da nuvem. A transferência de dados para o fornecedor da nuvem gera desconforto no cliente com base nos quesitos de privacidade e segurança [36] [39]. Tais inseguranças ao longo dos últimos anos vem sendo bastante discutidas e amenizadas com técnicas eficientes. Em [36], além de colocar a privacidade e a segurança como a principal preocupação dos clientes antes de aderir a nuvem, enfatizam que os riscos, levando em consideração a essas duas características podem ser menores que nos modelos tradicionais.

A compreensão de fato dos benefícios e riscos oferecidos pela computação em nuvem, deve ser tomada como processo fundamental para verificar a viabilidade da adesão dessa nova abordagem. Para auxiliar na compreensão é necessário que os usuários que têm interesses em adotar essa abordagem, façam um levantamento completo de tais benefícios e risco, e com isso passem a enxergar a CN com outros olhos e mais aceitação, visto que seus benefícios sobrepõem os problemas que possam surgir. A escalabilidade é atributo-chave da CN adaptado de [23], é através da virtualização de servidores que essa característica é obtida.

A CN é um ambiente baseado em redes e que fornece o compartilhamento de recursos computacionais. Na verdade, a CN é segundo [23] um modelo que possibilita acesso, de modo conveniente e sob demanda, a um conjunto de recursos computacionais configuráveis (por exemplo: redes, servidores, armazenamento, aplicações e serviços). Os serviços oferecidos pela nuvem podem ser liberados de maneira mais rápida e sem o mínimo de esforço por parte dos usuários, vale ressaltar que os usuários pagarão somente pelo que for usado e não precisam adquirir *hardwares* de ponta.

2.1.1 Modelos de serviços da computação em nuvem

A CN é identificada e conhecida pela sua forma de distribuir os recursos fornecidos para os usuários [19]. Tais formas são encontradas na literatura científica, como já mencionado na seção 2.1.3, como *Software as a Servic* (SaaS), *Platform as a Service* (PaaS) e *Infrastructure as a Service* (IaaS). Esses modelos de serviços, podem ser vistos na Figura 2.1 que foi adaptado de [19] e serão descritos a seguir.

A interação dessa nova abordagem se dá por meio dos clientes e do servidor (provedor da nuvem). No meio desses dois atores da computação em nuvem, encontram-se os serviços. Como já citados SaaS, PaaS e IaaS, são descritos segundo [23] [19] [22].

- *Software-as-a-Service* (SaaS) – Os serviços na nuvem, como software, são fornecidos através da *internet*, eliminando assim a necessidade de instalar e executar o aplicativo no sistema do usuário. Em [22] são apresentados algumas características importantes desse modelo:

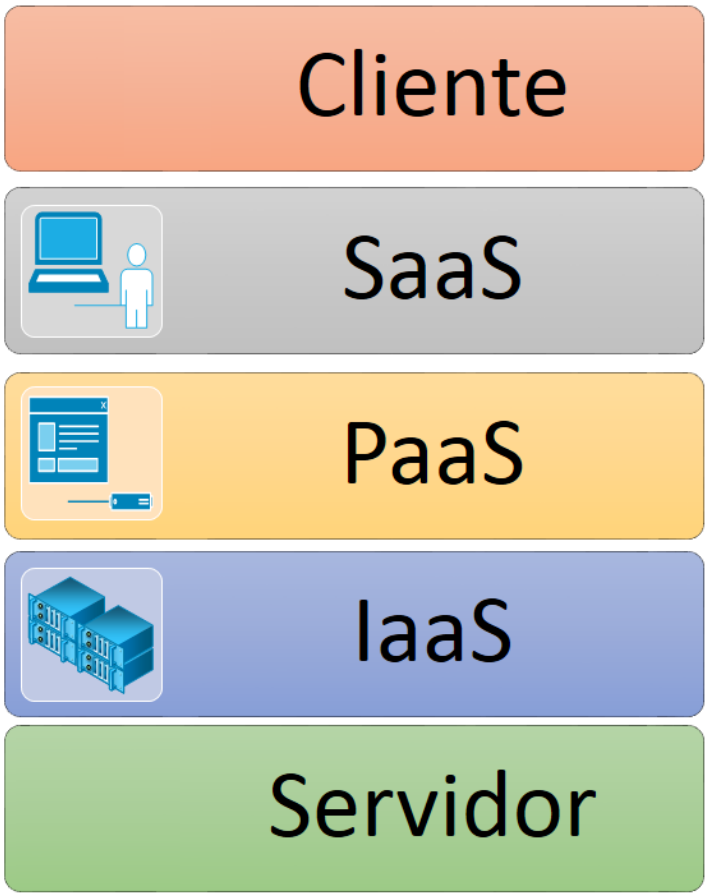

Figura 2.1: Modelos de serviços da computação em nuvem.

  - Acesso baseado em rede e gerenciamento de software disponível comercialmente que são gerenciados a partir de locais centralizados (provedores) e permitindo aos clientes acessar esses aplicativos remotamente através da internet. Como exemplo desse tipo de modelo temos:
    - Google Apps (que é o SaaS mais utilizado);
    - SalesForce (SFDC);
    - NetSuite;
    - Oracle;
    - Microsoft.

- *Platform-as-a-Service* (PaaS) – Um ambiente de desenvolvimento é fornecido, ou seja, uma plataforma de computação utilizando a infraestrutura da nuvem. Esse modelo de nuvem tem todos os aplicativos normalmente exigidos pelos clientes implantados. Assim, os clientes que solicitarem esse tipo de modelo de CN, não

precisa atravessar as dificuldades de comprar e instalar o software e hardware necessários para isso. Isso, porque os desenvolvedores através desses serviços podem obter todos os sistemas e ambientes necessários para o ciclo de vida de um *software*, seja ele desenvolvimento, teste, implantação e hospedagem de aplicações web [32]. Como exemplo tem-se:

  - *Microsoft Azure*.

- *Infrastructure-as-a-Service* (IaaS) – Esse modelo de CN fornece a infraestrutura necessária como um serviço. O cliente não precisa comprar os servidores necessários, *data center* ou recursos de redes, isso porque, toda essa infraestrutura será fornecida pelo provedor da nuvem. Além disso, a principal vantagem é que os clientes precisam pagar apenas pelo tempo de duração que utilizam o serviço. Como resultado, as organizações que utilizam a CN, passaram a ter mais tempo para dedicar aos seus serviços, tornando-os mais rápidos e com menos custos.

O servidor (provedor da nuvem) consiste nas características computacionais do hardware e/ou *software* necessário para a entrega dos serviços acima mencionados. Todos os serviços fornecidos pela CN são *pay-per-use*, ou seja, o cliente pagará somente pelo o que for usado, o que torna essa abordagem uma opção bastante atraente para as organizações que não desejam comprar, instalar e manter os gastos e serviços necessários para se ter um ambiente desses.

2.1.2 Arquitetura da computação em nuvem

A computação em nuvem possui uma arquitetura em camadas, onde cada camada possui seu nível de abstração e controle sobre as demais, e por trás encontram-se servidores, equipamentos de rede e sistemas operacionais. O nível de abstração torna-se maior quanto mais alto for a camada, já o controle da infraestrutura é inversamente proporcional quanto maior a camada menor será o controle. Esses níveis de abstração e controle foram apresentados por [2] como mostra a figura 2.2 adaptado de [2].

Para auxiliar melhor na compreensão da computação em nuvem é necessário definir as suas classificações tanto dos tipos, como dos serviços oferecidos e suas características essenciais. Segundo [23] [2] a CN é composta por cinco

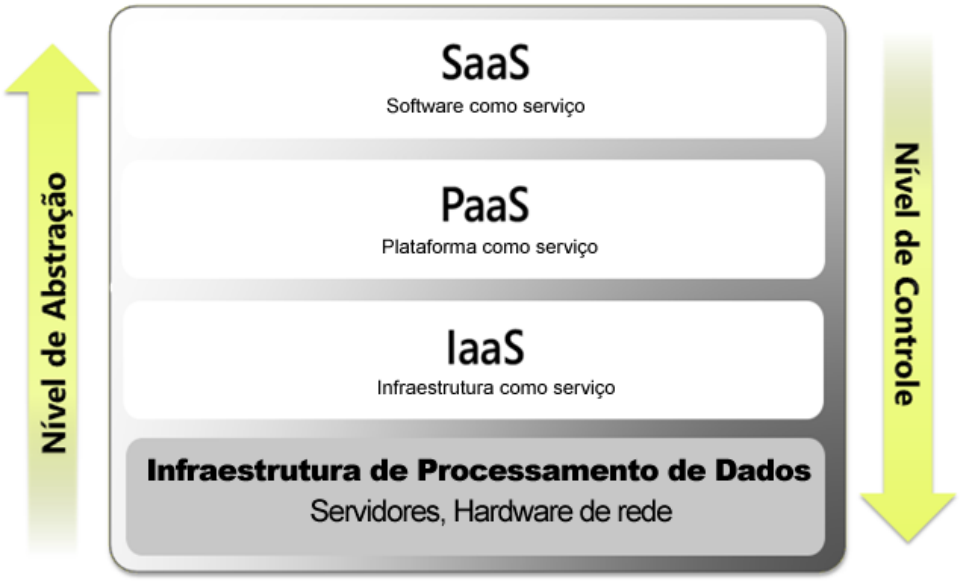

Figura 2.2: Camadas da infraestrutura física da computação em nuvem.

características essenciais (Serviço sob demanda, amplo acesso a rede, *pooling* de recursos, elasticidade rápida e serviços mensurados), três modelos de serviço (SaaS, PaaS, IaaS) e quatro modelos de implementação (pública, privada, híbrida e comunitária) [23]. Na seção 2.1.3 serão apresentados as características, os modelos de serviço e de implementação.

2.1.3 Características Essenciais da computação em nuvem

Segundo o NIST em uma de suas publicações [23], o que torna a CN uma opção de TI são suas características essências. Tais características serão descristas, e podem ser observadas na figura 2.3 adaptado de [23].

- **Amplo acesso a rede** – Os recursos serão disponibilizados através da rede e com isso o acesso do usuário é amplo, visto que o usuário só terá que conter um simples mecanismo que possa acessar os recursos. Logo, poderá acessar de qualquer lugar, a qualquer hora, de qualquer dispositivo, desde que tenham conexão (*internet*);

- **Serviço sob demanda** – O consumidor poderá solicitar os recursos da nuvem conforme a necessidade. Com o mínimo de interação humana com o provedor de serviço;

Figura 2.3: Modelo visual de definição de computação em nuvem do NIST.

- **Elasticidade rápida** – Ao solicitar os recursos da nuvem o usuário passará a ter a ilusão de recursos ilimitados, pois poderá adquirir em qualquer quantidade e a qualquer momento. Com essa característica a nuvem provêm com maior facilidade e poupa os recursos da nuvem, a medida que se faz necessário;

- **Serviço mensurado** – O provedor da nuvem monitora, controla e reporta. Com isso, todas as atividades e serviços são transparentes tanto para o provedor como também para o consumidor do serviço utilizado;

- ***Pooling* de recursos** – o consumidor só terá que se preocupar com um dispositivo que seja capaz de acessar a *internet*, pois todos os recursos serão fornecidos (exemplos de recursos incluem o armazenamento, processamento, memória, largura de banda, de rede e máquinas virtuais). Os recursos oferecidos pela nuvem serão liberados quando o usuário fizer uma solicitação, podendo ser em pequena ou larga escala de acordo com a necessidade.

2.1.4 Modelos de implementação da computação em nuvem

Quando uma organização decidir migrar para a CN, a primeira tarefa é escolher o tipo de nuvem que será implementado. Atualmente existem três tipos de implementações de nuvem, que são elas:

- nuvem pública;

- nuvem privada;
- nuvem híbrida.

A Figura 2.4 ilustra os tipos de modelos supracitados de implementações de nuvens e logo a seguir será feita uma descrição de cada modelo baseado nos estudos feitos por [23] [32] [29].

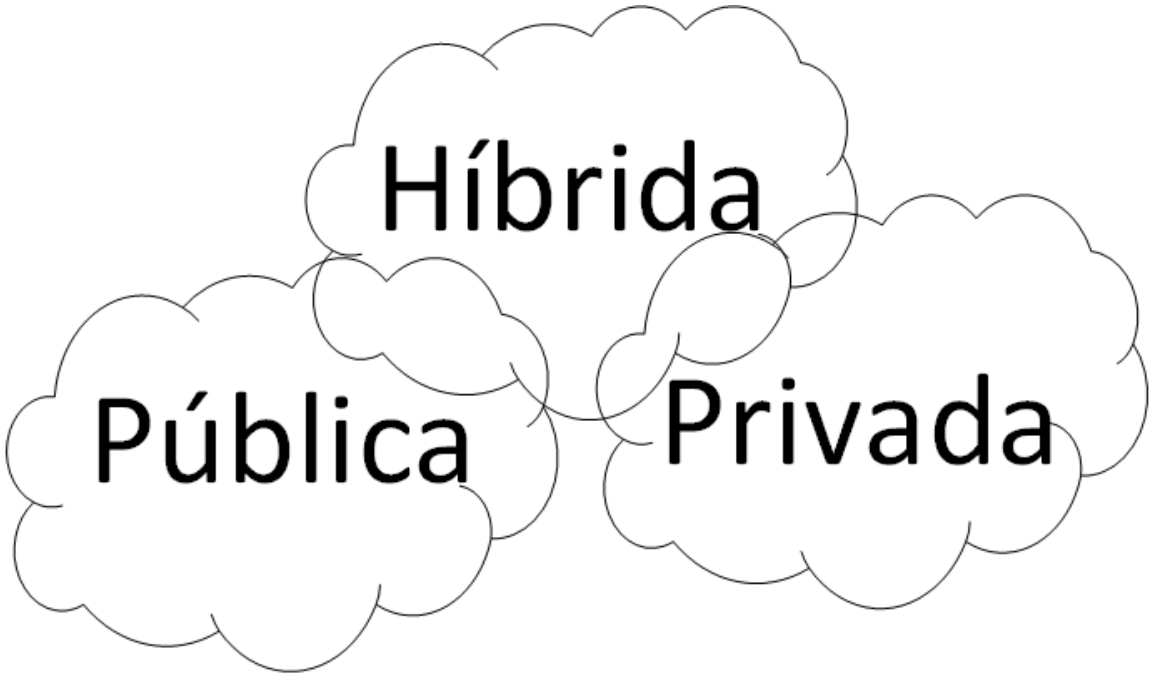

Figura 2.4: Tipos de modelos de implementação de nuvem

- **Nuvem Pública** – A utilização desse tipo de implementação de nuvem é por meio da utilização dos navegadores web. Neste modelo de implementação de CN os recursos (aplicativos, armazenamento, sistemas de rede etc.) são fornecidos como serviço por provedores (serviço terceirizado), e os usuários pagarão somente pelo tempo que utilizar os serviços, ou seja, *pay-per-use*.

  Em [19] é feita uma comparação da computação em nuvem com o sistema elétrico que é fornecido em diversas casas. É pago apenas uma quantia pelo que é usado, e esse serviço pode ser poupado, evitando assim desperdícios. Isso ajuda a diminuir os custos das despesas. Ainda em [19] é relado que os modelos de nuvens públicas são menos seguros em comparação com outros modelos de nuvem, isso porque, todos os aplicativos e dados ficarão disponíveis

a todos que solicitarem esse tipo de nuvem, tornando-se então, um modelo de implementação, segundo [19], propenso a ataques maliciosos.

Para solucionar esses problemas, ambas as partes (cliente e provedor) devem identificar as suas responsabilidades dentro de seus limites de operação, e sempre fazer verificações de segurança.

- **Nuvem Privada** – As organizações possuem suas própria nuvem, ou seja, os centros de dados e toda a infraestrutura está sobre o domínio da organização. Segundo [19], a principal vantagem em adotar esse tipo de modelo de implementação é a facilidade para gerenciar a segurança, manutenção e atualização e também o controle sobre as aplicações utilizadas. Os recursos e aplicações são gerenciados pela própria organização.

- **Nuvem Híbrida** – É uma combinação de nuvem pública e nuvem privada. Neste modelo pode-se ver a interação da nuvem privada com um ou mais serviços de nuvem externas. [19] lança esse modelo como o mais seguro, pois a organização terá controle nos dados, aplicações e toda infraestrutura, permitindo acesso à informação através da internet.

- **Nuvem comunitária** – Esse modelo se caracteriza pela junção de várias organizações, onde haverá um compartilhamento da infraestrutura da nuvem, as suas necessidades e as políticas. A infraestrutura pode ser hospedada por um terceiro (provedor) ou dentro da própria organização da comunidade.

2.2 Computação Ciente do Contexto para Adaptação de Software

Ciência do Contexto (CC) foi citado pela primeira vez na área de sistemas de informação e aplicações por [38], em 1994. "Um tema importante do projeto é "consciência de contexto", um termo que descreve a capacidade do computador para sentir e agir de acordo com informações sobre o seu ambiente, tais como localização, tempo, temperatura, identidade do usuário, entre outros. Esta informação pode ser usada não só para marcar informações que são coletadas no ambiente, mas também

para permitir respostas seletivas, como acionar alarmes ou recuperar informações relevantes para as tarefas", esta foi a definição de [35].

De acordo com [4], o conceito de contexto usa seis componentes essenciais: natureza, comportamento, restrição, influência, estrutura, sistema. Percebe-se na literatura científica que não há consenso em alguns da caracterização do contexto, a seguir alguns deles: se é conjunto de fenômenos ou uma rede organizada, se é conjunto de informação ou de processos, se estático ou dinâmico, se interno ou externo. Na tentativa de explicar o que vem a ser CC dentro da ciência da computação [37] mostram os três principais aspectos para que haja a compreensão da mesma com apenas algumas frases intuitivas:

1. "Onde você está";
2. "Quem está com você";
3. "Quais os recursos próximos".

De acordo com [37] há vários tipos de classificação, e dentre eles os mais básicos são:

- Contexto computacional: aquele que é provido das informações de configuração do contexto computacional, por exemplo:
  - Quantidade de processamento;
  - Custos de comunicação;
  - Conectividade;
  - Memória disponível;
  - Largura de banda;
  - Quantidade de bateria disponível;
  - Recursos ou serviços disponíveis.
- Contexto do usuário: são informações colhidas dos usuários, tais como:
  - Dados que definem perfil;
  - Identificação de contatos;

  - Identificação de indivíduos ao redor;
  - As tarefas que o indivíduo está executando;
  - A localização do usuário;
  - Características espaciais.

- Contexto físico: é aquele que é caracterizado pelo ambiente físico ao qual o dispositivo, o usuário ou ambos estão localizados.

"No serviço da ciência do contexto, as situações dos clientes são verificados por valores de contexto ou suas combinações, e os serviços adequados são fornecidos, se estes contextos conseguirem restringir as condições" [38].

"Um sistema é sensível ao contexto se ele usa contexto para fornecer informações e/ou serviços relevantes para o usuário, onde a relevância depende da tarefa do usuário" [14].

Com as definições acima, descreve a identificação de um sistema ser ciente do contexto ou não, [18] apresentou três abordagens principais que pode-se seguir para o desenvolvimento de aplicações cientes do contexto:

- Modelo de contexto explícito: utiliza uma infraestrutura de gerenciamento de contexto. Ações como a aquisição de contexto, pré-processamento, armazenamento e processamento não coincidem com os limites de aplicação. Gestão de contexto e aplicação estão separados e podem ser desenvolvidos de forma independente.
- Modelo de contexto implícita: usa bibliotecas, *frameworks* e ferramentas para executar aquisição contexto, pré-processamento, armazenamento e processamento.
- Modelo de contexto ausente ao nível do aplicativo: executa todas as ações, como aquisição contexto, pré-processamento, armazenamento e processamento dentro dos limites de aplicação.

A CC é muito utilizada na área da computação para criar ou adaptar sistemas de informação voltados diretamente para o domínio da segurança, com base nas necessidades do usuário ou do dispositivo, ou de ambos [14]. Estas adaptações

tem maior nível de acerto quando são coletadas e usadas as informações contextuais relevantes aos sistemas, especialmente em sistemas para DM. Dentre as informações contextuais pode-se citar o uso do tempo, características do DM, localização do usuário, recursos disponíveis de rede, descoberta de outros dispositivos no local, entre outros.

Em uma situação abstrata onde existe uma aplicação que faça comunicação por mensagem instantânea por meio da *internet*. Em determinado instante o usuário está em um local que tem acesso a uma rede que ofereça segurança e alta velocidade de transmissão de dados. Neste caso a aplicação pode detectar o contexto em que esse DM está inserido e decidir que por conta do ambiente não há necessidade de uma criptografia alta e nem uma compressão de mensagem. Com isso a aplicação economiza na quantidade de bateria, por ter decidido que o ambiente oferecia um risco menor e maior velocidade de conexão.

Em outro instante o usuário que utiliza a aplicação se desloca para um *shopping*, e perde a conexão com aquela rede segura e rápida que estava utilizando. Agora esse usuário passa a utilizar a rede 3G, disponível no trajeto dele até o *shopping*, por meio da operadora. A aplicação de mensagem, que ele está utilizando, identifica a rede em que ele está conectado e verifica que a rede é considerada segura em relação a um possível roubo de dados, mas também verifica que a velocidade de conexão diminuiu. Então a aplicação decidirá que para poupar bateria, ele irá tirar a criptografia da mensagem, mas como a velocidade de conexão é baixa ele terá de comprimir a mensagem para que ela fique menor e seja transferida mais rapidamente.

Em um terceiro momento chega-se ao *shopping* e se conecta à rede disponível para clientes. A aplicação identifica que ele se conectou a uma rede sem segurança e lenta, com isso o algoritmo de decisão da aplicação resolve o problema da rede insegura inserido uma criptografia para transmitir a mensagem, e também resolve comprimir esta mensagem para que seja transmitido mais rápido na rede lenta ao qual o DM está conectado. O resultado desta decisão terá o efeito de maior consumo na energia do DM. Na tabela 2.1 se visualiza com mais facilidade a situação que foi mostrada agora.

É compreendido até aqui que há uma variedade de definições de contexto, dependendo apenas das necessidades de uso e informações categorizadas em

Rede	Criptografia	Compressão	Gasto de Energia
Segura e rápida	Não	Não	Baixo
Segura e lenta	Não	Sim	Médio
Insegura e lenta	Sim	Sim	Alto

Tabela 2.1: Tabela comparativa do exemplo da aplicação de mensagem ciente do contexto

domínios específicos, entre elas aquela que será utilizada nesse trabalho e é definido por [14], a computação móvel. Esta definição engloba os dispositivos móveis, as aplicações, os usuários e as relações entre eles, e também os ambiente acessíveis.

De acordo com [33] o contexto pode ser categorizado de duas maneiras com base no tipo de informação: estática ou dinâmica.

- Estático: "o engenheiro de software pode combinar diversos componentes, em tempo de compilação, e gerar uma aplicação" [33].
- Dinâmico: "o engenheiro de software pode adicionar, remover ou reconfigurar componentes dentro de uma aplicação em tempo de execução" [33].

As informações de contexto são de extrema importância para a realização de uma tarefa essencial e desafiadora dos sistemas que buscam permanecer mais tempo no mercado: a potencial capacidade destes se adaptarem às diferentes situações que lhe são impostas. A dinamicidade das informações que podem ser obtidas através da exploração dos contextos envolvidos no domínio da computação móvel evidencia a possibilidade de se trabalhar diferentes tipos de adaptações em nível de demanda de aplicações. Exemplificando, pode-se desativar serviços complexos em execução a fim de aumentar a disponibilidade de recursos, tornando possível a execução em dispositivos que apresentam limitação de recursos [27].

Foi identificado por [20] e discutido seis princípios de design relacionados ao framework de ciência de contexto. Além disso, [30] e [5] também identificaram vários requisitos de design, como por exemplo:

- Arquitetura de camadas e componentes: as funcionalidades têm de ser divididos em camadas e componentes. Cada componente deverá executar uma quantidade limitada da tarefa e deve ser capaz de realizar de maneira independente, em grande medida.

- Escalabilidade e extensibilidade: o componente deve ser capaz de adicionar ou remover funcionalidade ou recurso de forma dinâmica. Por exemplo: novas funcionalidades devem ser capazes de ser adicionadas sem alterar os componentes existentes. A funcionalidade deve ser desenvolvida de acordo com padrões, o que melhora a escalabilidade e capacidade de expansão.

- Gerenciamento do ciclo de contexto automático: *frameworks* cientes do contexto devem ser capazes de ser entendidos pelas fontes de contexto disponíveis, a sua estrutura de dados e construir modelos de dados internos. Além disso, as informações brutas de contexto precisam ser recuperadas e transformadas em modelos de representação de contexto apropriado corretamente com mínima intervenção humana.

A aquisição do contexto pode ser realizado principalmente através de dois métodos [28]: Puxar: A aplicação de software que é responsável pela aquisição de dados de sensores faz um pedido (por exemplo, consulta) a partir do hardware do sensor periodicamente (isto é, depois de certos intervalos) ou constantemente para a aquisição de dados. Empurrar: O sensor físico ou virtual envia dados para a aplicação de software que é responsável para a aquisição de dados do sensor periodicamente ou constantemente.

Além disso, o contexto pode ser gerada com base em dois tipos de eventos diferentes [27]: Instantâneo: Estes eventos ocorrem instantaneamente. Abrir a porta, ligar a luz, ou animal entra no campo de visão são alguns tipos de eventos instantâneos. Para detectar este tipo de evento, dados do sensor precisam ser adquiridos quando ocorre o evento. Ambos métodos empurrar e puxar podem ser empregados. Intervalo: Estes eventos abrangem um certo período de tempo. Chovendo, animal comer uma planta, ou no inverno são exemplos de eventos de intervalo. Para detectar este tipo de evento, dados do sensor precisam ser adquiridos periodicamente. Ambos métodos empurrar e puxar podem ser empregados.

Métodos de aquisição de contexto podem ser classificados em três categorias [11] com base no contexto originado: diretamente do sensor, por meio da infra-estrutura da aplicação e por meio dos servidores de contexto [27].

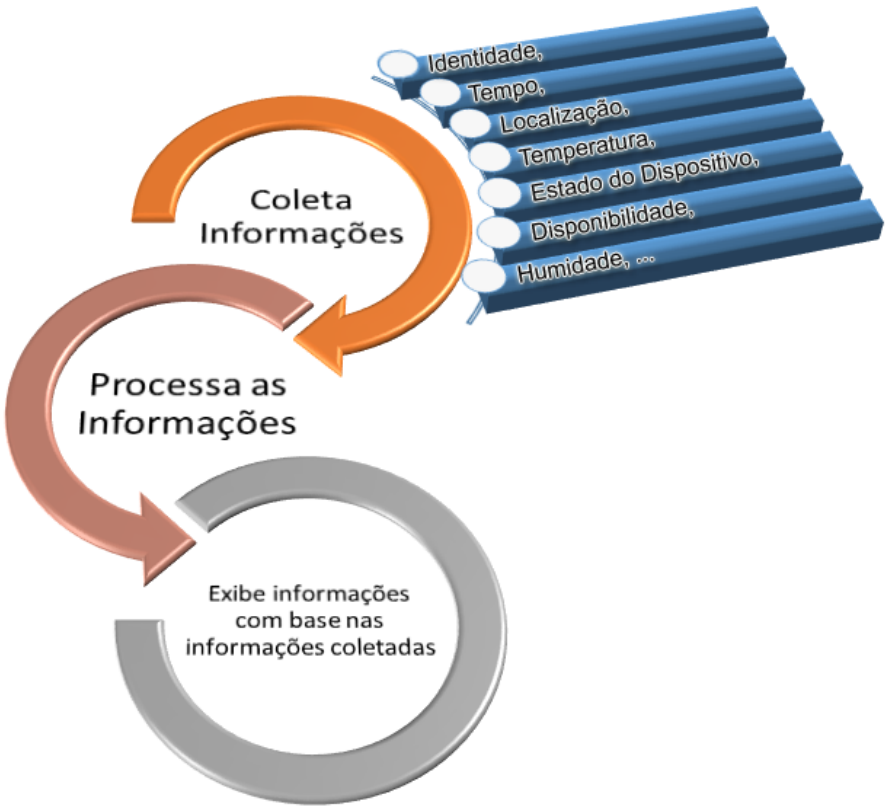

Figura 2.5: Diagrama de exemplo de uso da Ciência do Contexto

2.3 Segurança

2.3.1 Visão Geral

A segurança busca definir metodologias e técnicas para manter as informações protegidas de possíveis ações danosas praticadas por entidades não autorizadas [16]. A segurança visa a proteção contra a indisponibilidade, vazamento da informação e a leitura ou modificação não-autorizada das informações [6].

Os mecanismos de segurança são concebidos para detectar e prevenir um ataque à segurança bem como recuperar-se do mesmo [40]. Por exemplo, a criptografia é um destes mecanismos que é utilizado para atender os requisitos de segurança.

Entretanto, a evolução tecnológica tem mostrado que novos obstáculos em segurança são criados à medida que as barreiras são vencidas. A segurança de um algoritmo criptográfico deve resistir ao segredo da chave, e não ao sigilo do algoritmo. No caso de um ambiente móvel sem fio, outro desafio de segurança é que o desempenho pode atuar como fator limitante da utilização de alguns algoritmos e

protocolos, uma vez que um determinado dispositivo móvel pode apresentar limitação de recursos na execução dos mesmos [8]. Os requisitos de segurança baseados na criptografia são confidencialidade, integridade, autenticação e não-repúdio [16] [40] [24]. Além destes, existem requisitos baseados em outras propriedades, tais como: disponibilidade e controle de acesso.

A confidencialidade (também conhecida por sigilo e privacidade) é a propriedade de segurança utilizada para garantir que os dados não sejam expostos a entidades não autorizadas. Este requisito consiste em proteger a informação contra leitura de todos que não tenham autorização [8]. O foco desta dissertação é o requisito de confidencialidade baseado na criptografia.

A integridade é utilizada para garantir que entidades não-autorizadas modifiquem os dados. Assim, a informação transmitida deve ser idêntica a recebida, consequentemente, está protegida contra modificações. A modificação inclui ações como inserção, retirada e substituição do conteúdo das informações [8].

A autenticação assegura que as identidades de ambas as partes envolvidas na transação foram verificadas e confirmadas. As informações entregues também devem ser autenticadas. Este requisito normalmente é subdividido nas duas classes principais citadas anteriormente: autenticação de entidade e autenticação de dados. Vale ressaltar que autenticação garante a integridade dos dados implicitamente, porque identifica quando uma mensagem é modificada [8].

O não-repúdio é utilizado para assegurar que as entidades envolvidas em uma transmissão não possam negar sua participação nessa comunicação. Assim, quando uma mensagem é enviada, o receptor pode provar que a mensagem foi enviada pelo remetente. Semelhantemente, quando uma mensagem é recebida, o remetente pode provar que a mensagem foi recebida pelo receptor [8].

A disponibilidade consiste na proteção dos serviços prestados pela aplicação, de forma que eles não sejam degradados ou tornem-se indisponíveis sem autorização, isto é, a informação ou aplicação de computador deve estar disponível no momento em que as mesmas sejam requisitadas. Já o controle de acesso consiste na capacidade de permitir ou negar o uso dos serviços e recursos oferecidos pela aplicação [8].

Criptografia é a ciência que estuda as forma de codificar a mensagem, e criptografar é o processo de codificar uma mensagem de tal forma que oculte o seu conteúdo [13]. A criptografia utiliza técnicas para transformar uma informação fácil de ser compreendida em um código difícil de ser lido [12].

O intuito desta transformação é a busca por garantias de confidencialidade de uma informação que precisa ser enviada para um destinatário, o qual possuirá o conhecimento de como decifrá-la [12].

Entre as maneiras de criptografar uma informação existe aquela realizada por algoritmos criptográficos que utilizam chaves para esta tarefa. Eles podem ser classificados de acordo com a utilização destas chaves, podendo ser simétricos ou assimétricos [12].

Quando uma mensagem é criptografada aplicando alguma regra onde a transformação da mensagem para um código é feita com base em uma função e uma chave, e esta mesma chave é utilizada na função de transformação inversa deste código, que resulta na mensagem original, tem-se a identificação de uma criptografia por simetria. Nesta estão classificados todos os algoritmos com estas características, conhecidos como algoritmos simétricos [12].

Outra forma de criptografia é aquela que utiliza os algoritmos assimétricos, que são aqueles algoritmos que utilizam uma função e uma chave pública para cifragem e decifragem da mensagem em um dos lados da comunicação e outra função e outra chave privada, para cifragem e decifragem da mensagem no outro lado da comunicação (lado do fornecedor). Assim, existem duas chaves, uma pública, para ser usada por todos os clientes que precisam se comunicar com o fornecedor, e outra privada, conhecida apenas pelo fornecedor da chave pública [12].

A mensagem codificada utilizando a chave pública produz uma mensagem que só pode ser decifrada utilizando a chave privada. Mesmo que a chave pública seja descoberta, não será possível usá-la para decifrar uma mensagem. Dessa forma, o fornecedor das chaves garante que quem possuir a chave de domínio público conseguirá ler sua mensagem e qualquer mensagem retornada para ele será decifrada apenas por ele. Este tipo de criptografia é bastante robusta e incrementa o nível de segurança das mensagens trocadas, porém requer um custo de processamento muito maior do que o necessário para os algoritmos simétricos [12].

2.3.2 Segurança em Nuvem

No cenário atual, o rápido crescimento no campo da computação em nuvem também tem contribuído para aumentar a preocupação com a segurança. A computação em nuvem é rodeada por questões de segurança como a privacidade dos dados armazenados na nuvem, a utilização indevida do recursos da nuvem por usuários maliciosos além da administração dos dados dos usuários ser feita por terceiros. Assim, para uma implantação bem sucedida da computação em nuvem, é preciso um gerenciamento da segurança [15].

Na norma ISO 7498-2, produzido pela Organização Internacional de Normalização (ISO), Segurança da Informação deverá abranger um número de temas sugeridos [29]. Segurança em Computação em Nuvem deve também seguir estas recomendações, a fim de tornar-se uma solução de tecnologia eficaz e segura, Estes itens são:

- Identificação e autenticação: dependendo do tipo de nuvem adotada e do modelo de troca de dados, os usuários em primeiro lugar devem ser especificados e estabelecidos às prioridades de acesso e permissões. Isto é feito com o proposito de verificar e validar os usuários da nuvem através de nome de usuário e senhas [29].

- Autorização: exigência importante de segurança da informação em computação em nuvem para garantir que a integridade referencial seja mantida. Decorre sobre exercer controle e privilégios sobre os fluxos de processo dentro de computação em nuvem [29].

- Confidencialidade: é uma obrigação quando se emprega uma nuvem pública, devido à natureza das nuvens públicas acessíveis por várias organizações e usuários diferentes. Afirmando confidencialidade nos perfis dos usuários e protegendo seus dados, isso deve impedir que os dados sejam acessados por outros usuários. São usados protocolos de segurança da informação em várias camadas diferentes de aplicações na nuvem [29].

- Integridade: é uma implicação principalmente do acesso aos dados na computação em nuvem. Portanto, as propriedades ACID (atomicidade,

consistência, isolamento e durabilidade) de dados da nuvem devem sem dúvida ser robustamente impostas em todos os modelos de computação em nuvem [29].

- Não-repúdio: em computação em nuvem pode ser obtido através da aplicação dos tradicionais protocolos de segurança de e-commerce e provisionamento token para transmissão de dados dentro de aplicações em nuvem, tais como assinaturas digitais, *timestamps* e serviços de confirmação de recebimentos (de recepção digital de mensagens de dados confirmando enviadas/recebidas) [29].
- Disponibilidade: é um dos requisitos de segurança da informação mais crítico na computação em nuvem, porque é um fator de decisão importante ao decidir entre os fornecedores de nuvem privados, pública ou híbrida [29].

2.3.3 SSL

Transport Layer Security (TSL) e o seu predecessor, *Secure Socket Layer* (SSL), são protocolos criptográficos que conferem segurança de comunicação na internet para serviços como email, navegação por páginas e outros tipos de transferência de dados. Há algumas pequenas diferenças entre o SSL 3.0 e o TSL 1.0, mas o protocolo permanece substancialmente o mesmo. O termo "SSL" usado aqui aplica-se a ambos os protocolos [42].

Sua proposta é permitir a autenticação de servidores, encriptação de dados, integridade de mensagens e, como opção, a autenticação do cliente, operando nas comunicações entre aplicativos de forma interoperável [3].

O SSL também permite a montagem de uma framework onde outras chaves públicas e métodos de encriptação podem ser utilizados, evitando a necessidade de implementação de toda uma pilha de protocolos (com os riscos da introdução de fraquezas) [3].

Como uma vantagem adicional, a questão do desempenho foi levada em consideração no projeto, para reduzir o número de conexões e minimizar o tráfego na rede pode ser usado opcionalmente um esquema de cache em memória durante o

estabelecimento da sessão, com a finalidade de reduzir o número de conexões e reduzir a atividade no acesso à rede [3].

O SSL é um protocolo que se utiliza de dois tipos de criptografia, assimétrica e simétrica. Inicialmente estabelece a conexão com uma criptografia assimétrica e através desta conexão segura realiza a troca de chave da criptografia simétrica e por fim continua a comunicação dos dados com uma criptografia simétrica [1].

Uma descrição do SSL foi dada por Franck Martin em "SSL Certificates HOWTO" [21]:

- O navegador solicita uma pagina de segurança;
- O servidor web envia sua chave pública com o seu certificado;
- O navegador verifica o certificado que foi emitido de uma fonte confiável, onde seu certificado ainda é validado e se o certificado está relacionado ao local contactado;
- O navegador então usa a chave pública, para criptografar uma segunda a chave de criptografia simétrica aleatória e envia para o servidor com a URL criptografada necessária, bem como outros dados HTTP criptografado;
- O servidor web decifra a chave de criptografia simétrica usando a sua chave privada e usa a chave simétrica para descriptografar os dados de URL e HTTP;
- O servidor web envia de volta os documentos HTML e dados HTTP criptografados com a chave simétrica solicitados;
- O navegador decifra os dados HTTP e documento HTML usando a chave simétrica e exibe as informações.

Este fluxo resolve diversos problemas que se tem utilizando os modelos de criptografia simétrica/assimétrica separadamente [42]:

- Criptografia simétrica

  Rápido;

  Uma só chave para cifrar e decifrar;

Problema para a troca de chaves (pois o cliente e servidor precisam conhecer a chave).

- Criptografia assimétrica

  Lento;

  Usa um par de chaves. Onde uma chave cifra e outra decifra;

  O cliente precisa apenas conhecer a chave pública para cifrar e enviar os dados ao servidor, que por sua vez consegue decifrar a informação com a chave privada.

Ao estabelecer a conexão, o *Handshake Protocol* estabeleceu o identificador de sessão, o conjunto criptográfico a ser adotado e o método de compressão a ser utilizado. O conjunto criptográfico negociado define três algoritmos [3]:

- Um algoritmo para troca de chaves;
- Um algoritmo para cifragem de dados;
- Um algoritmo para inserção de redundância nas mensagens.

2.4 Computação Verde

O uso eficiente de recursos computacionais visando minimizar o impacto ambiental tendo como objetivo maximizar a economia destes recursos é chamada de computação verde. Com a CV é possível que as ações tecnológicas futuras irão prejudicar minimamente ao meio ambiente por serem mais ecológicas e eficientes, principalmente no consumo do recurso de energia que é primordial para o contexto computacional [43].

Os objetivos da computação verde se baseiam em reduzir o uso excessivo ou abusivo de materiais perigosos e maximizar o uso consciente de energia durante a vida do equipamento, além de promover a reciclagem ou biodegradabilidade de produtos e resíduos da fabricação [43].

De acordo com pesquisa realizada por [3], a tecnologia em nuvem pode contribuir significativamente para a redução do consumo de energia na computação

empresarial e consequente emissão de gases do efeito estufa. O estudo que analisou a quantidade de carbono emitido por servidores, rede e infraestrutura de armazenamento concluiu que as empresas menores são mais beneficiadas pela utilização da Computação em Nuvem [31].

Empresa que disponibilizam nuvens computacionais, administram e operam a capacidade e o armazenamento computacional de dados, proporcionando de forma escalar e disponibilizando o processamento, evitando a dependência de um único recurso físico, ainda garante a privacidade dos clientes. Esta arquitetura acompanha a redução do consumo de energia de Data Centers e contribui para a preservação do meio ambiente [31].

A Computação em Nuvem vem se mostrando uma tecnologia muito promissora para esse paradigma sustentável. Com o modelo explicado e com o objetivo de apenas disponibilizar recursos mediante a necessidade do usuário, permite-se um melhor gerenciamento de energia e processamento, resultando na busca por estratégias de computação verde em conjunto com a computação em nuvem, gerando a busca do conceito de Computação em Nuvem Verde [43].

A Nuvem Verde é muito semelhante a Computação em Nuvem, porém ela difere na preocupação a mais sobre a estrutura e consumo menor de energia [1] sem interferir na performance [7]. Suas características são:

- Flexibilidade: Oferece as mesmas características de reconfiguração da computação em nuvem, além de poder gerenciar o status das máquinas físicas (ligando/desligando) quando necessário, assim como possibilita o agrupamento dos recursos, e possibilita a mobilidade da estrutura [43];

- Disponibilidade: Oferece as mesmas características da computação em nuvem, e poderia gerenciar as máquinas físicas (hibernando) e remover máquinas virtuais ociosas [43];

- Custo: Com a funcionalidade de movimentação de máquinas virtuais extra nuvem, centros de dados adotam uma configuração minimalista, impactando também no aumento da vida útil dos equipamentos. Já a estrutura diminui os seus custos mensais, reduzindo o consumo de energia derivada das políticas de "desligamento e hibernação" [43];

- Sustentabilidade: Com a funcionalidade de movimentação de máquinas virtuais, tem-se a possibilidade de, em períodos de baixa demanda, concentrar o processamento das máquinas virtuais em poucos servidores físicos permitindo o desligamento dos equipamentos ociosos. O sistema trabalha em conjunto ao ambiente, inferindo a estratégia de trabalho sob demanda também aos equipamentos externos (por exemplo, refrigeração e rede), assim como leva em consideração os fatores externos, como agir pro-ativamente em caso de desastre (por exemplo, Incêndio) [43].

O crescimento tecnológico, juntamente com o maior consumo de energia é inevitável, porém é imprescindível que o meio ambiente seja preservado e recursos sejam melhores empregados.

A valorização da Computação Verde é necessária para suportar as necessidades da sociedade e a Computação em Nuvem pode referenciar investimentos baseados na sustentabilidade, pois os princípios e as práticas de Computação Verde preza pela conscientização e mudança de hábitos para melhorar o desempenho e reduzir o consumo de energia e emissão de gases poluentes [31].

Dos principais benefícios da Computação Verde a questão financeira é bastante relevante. Os investidores e os consumidores estão começando a exigir mais divulgações de empresas em relação a suas emissões de carbono, bem como as suas iniciativas ambientais. Por meio de iniciativas de TI verde obtém-se melhor eficiência energética e consequentemente é financeiramente mais viável [25].

Computação verde não se caracteriza apenas como algoritmos eficientes de computação em nuvem. Ela envolve todo o contexto de uso de equipamentos, design e descarte de recursos utilizados.

- Uso Verde: Reduz o consumo de energia dos computadores e outros sistemas de informação e usá-los de uma forma ambientalmente correta [25].
- Descarte Verde: Reformar e reutilizar computadores antigos e reciclar corretamente computadores indesejados e outros equipamentos eletrônicos [25].
- Design Verde: Eficiência do design de computadores, servidores e equipamentos de refrigeração energeticamente eficientes e ambientalmente saudáveis [25].

- Fabricação Verde: Fabricação de componentes eletrônicos, computadores e outros subsistemas associados com o mínimo ou nenhum impacto sobre o meio ambiente [25].

Pode-se reduzir significativamente o consumo de energia, fazendo pequenas alterações nas formas de uso de computadores. Além disso, computador geram calor e necessitam de refrigeração adicional, o que aumenta o consumo de energia total e o custo para a empresa. Enquanto as economias em custos de energia por PC pode não parecer muito, as economias combinadas para centenas de computadores em uma empresa é considerável.

O desligamento de sistemas ociosos que não estão em uso é outro exemplo de medida de TI verde, bem como a adoção de data centers verdes, com maior eficiência energética, e que fazem uso da virtualização de sistemas de maneira eficiente. A virtualização permite que data centers consolidar sua infraestrutura de servidores físicos por hospedar vários servidores virtuais em um número menor de servidores mais poderosos, usando menos eletricidade e simplificar o centro de dados [25].

2.5 Conclusões

Foram apresentados neste capítulo alguns conceitos relacionados com a computação em nuvem, ciência do contexto, segurança e computação verde.

Na seção de 2.1 foi apresentado as definições, características e classificações da computação em nuvem.

Na seção 2.2 foi apresentado os principais conceitos de computação ciente do contexto, com foco em adaptação de software, as principais classificações, os requisitos de design, os métodos de aquisição do contexto. Com essa fundamentação conclui-se porque a CC é um pilar para esse trabalho, se tornando uma das principais características do mecanismo proposto na dissertação.

Na seção 2.3 mostrou-se as principais características da segurança, os objetivos principais dos métodos de segurança, os requisitos básicos de segurança, foi dado ênfase na criptografia que um dos requisitos fundamentais desta dissertação, e apresentado a segurança voltada para CN e os requisitos básicos de segurança para

CN. Na mesma seção de segurança foi apresentado o SSL outro conceito fundamental da dissertação.

Por último, foi abordado na fundamentação teórica a computação verde, e esta foi colocada por existir uma relação muito próxima da ciência do contexto, visto que a CC implica diretamente no consumo e melhoria de uso dos recursos computacionais existentes. Nesta seção foi apresentada as características principais da CV

3 Mecanismo

Este capítulo tem como objetivo apresentar o serviço que será capaz de fornecer uma adaptação dinâmica ciente do contexto ao qual o DM está inserindo, para que haja uma autenticação e um canal seguro para comunicação com uma nuvem. Será apresentado a seguir os objetivos e requisitos do serviço. Em seguida, a arquitetura do serviço e seus componentes são detalhados.

3.1 Características Gerais

A CN tem muitas vantagens, apesar disso, ainda preocupa muitos usuários com relação à segurança e também à privacidade de suas informações. A nuvem pode ser resumida como uma coleção de recursos alocados para o usuário com disponibilidade de serviço e elasticidade. Ela tem como principal atividade, prover sistemas de controle e alocação de arquivos dos usuários, por muitas vezes usados para informações confidenciais. Por ter essas características acaba se tornando um alvo propício a ataques, em um ataque bem sucedido o atacante pode ter acesso a informações sigilosas do cliente.

A CC tem como principal objetivo tomar conhecimento do meio em que o DM está inserido. Com a CC obtém-se, por exemplo, quantidade de bateria, memória, processador disponíveis, também em qual tipo de rede o usuário está conectado, entre outros. De posse dessas informações pode-se definir qual melhor método a ser utilizado no dado momento para fornecer um serviço de qualidade e que não desperdice de forma desnecessária os recursos disponíveis. Este método de aproveitar da melhor forma possível os recursos de um dispositivo é a definição da CV.

A segurança se faz de técnicas e metodologias para garantir proteção às informações como confidencialidade e sigilo das informações [16]. A criptografia é um mecanismo utilizado para atender a prevenção do acesso ao conteúdo em caso de ataque à segurança [40]. O SSL tem como objetivo permitir a autenticação de

servidores, integridade de mensagens, encriptação de dados e opcionalmente poderá ser usado como autenticação do cliente de forma interoperável.

O SSACC foi implementado com base na arquitetura SSL. Na figura 3.1 apresenta o diagrama de camadas do SSACC juntamente com o SSL.

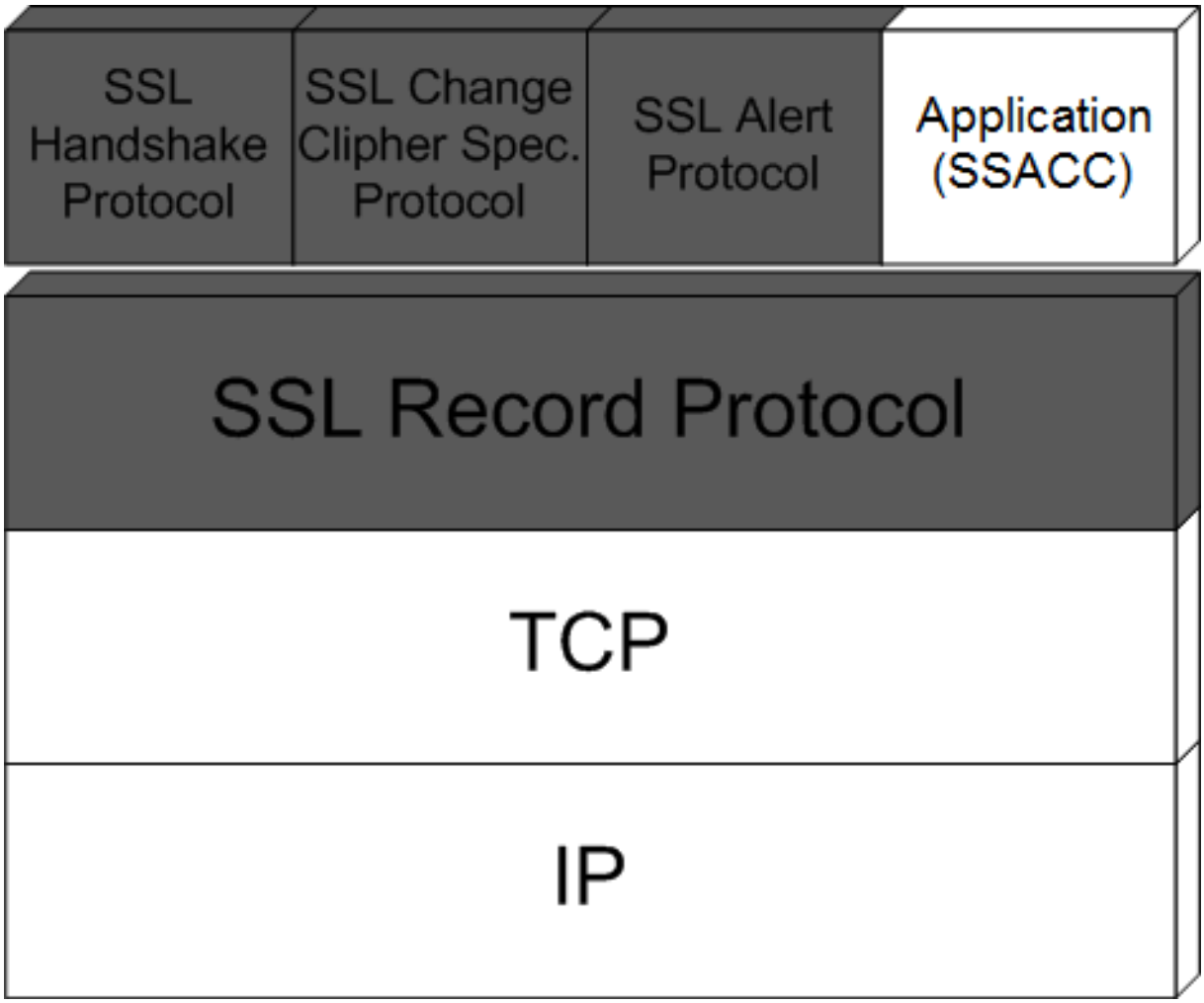

Figura 3.1: Diagrama de camadas do SSACC disposto com SSL

O serviço deste trabalho é descrito como um serviço que fornecerá ao usuário um canal seguro de comunicação com uma nuvem computacional e adaptado ao contexto em tempo de execução, denominado SSACC. O SSACC terá como características principais ter consciência do seu estado físico atual, levando em conta principalmente a quantidade de bateria disponível no DM e que tipo de rede está conectado, ser capaz de decidir qual algoritmo criptográfico ideal para uso no momento da requisição do cliente para que não haja desperdício computacional e consequentemente desperdício de energia do dispositivo.

O serviço do SSACC, para ser implementado, é necessário obtenção de algumas métricas importantes para que seja definida a tomada de decisão do algoritmo criptográfico a ser usado. Diante da necessidade de dados para a tomada de decisão, foi elaborado um aplicativo de teste denominado *stress*.SSACC o qual será descrito no decorrer do capítulo.

3.2 Arquiteturas dos Aplicativos

A seguir será apresentado a arquitetura dos Aplicativos (*Application*) (APP)'s. O APP *stress*.SSACC que será usado para obtenção de métricas necessárias para formulação do controle de decisão de criptografia do APP principal, o SSACC.

3.2.1 Arquitetura do Aplicativo de *Stress*

A figura 3.2 apresenta o diagrama de sequência do aplicativo que servirá para teste de *stress* dos algoritmos de criptografia selecionados, com base no que há disponível no pacote do SSL 3.1. Os algorítimos são o *Advanced Encryption Standard* (AES), *Triple Data Encryption Standard* (3DES) e *Rivest Cipher* 4 (RC4) [26].

Cada execução do aplicativo de teste conterá a seguinte configuração:

- Arquivo de tamanho padrão;
- O algorítimo criptográfico que será testado;
- O teste somente será realizado se houver entre 30% e 80% da bateria.

Na figura 3.2 é mostra a sequência dos processos, onde é coletado antes do início do *loop* a quantidade de bateria disponível no DM e armazenada para comparação a cada *loop*. A cada *loop* será coletado o contexto do DM (bateria) e gravado em um arquivo o *log* do evento. O ciclo só terminará ao ser constatado um consumo de 1% da bateria do DM. Serão feitos 6 testes ao total, cada teste será configurado com um algorítimo criptográfico e a função que será avaliada (criptografia ou descriptografia).

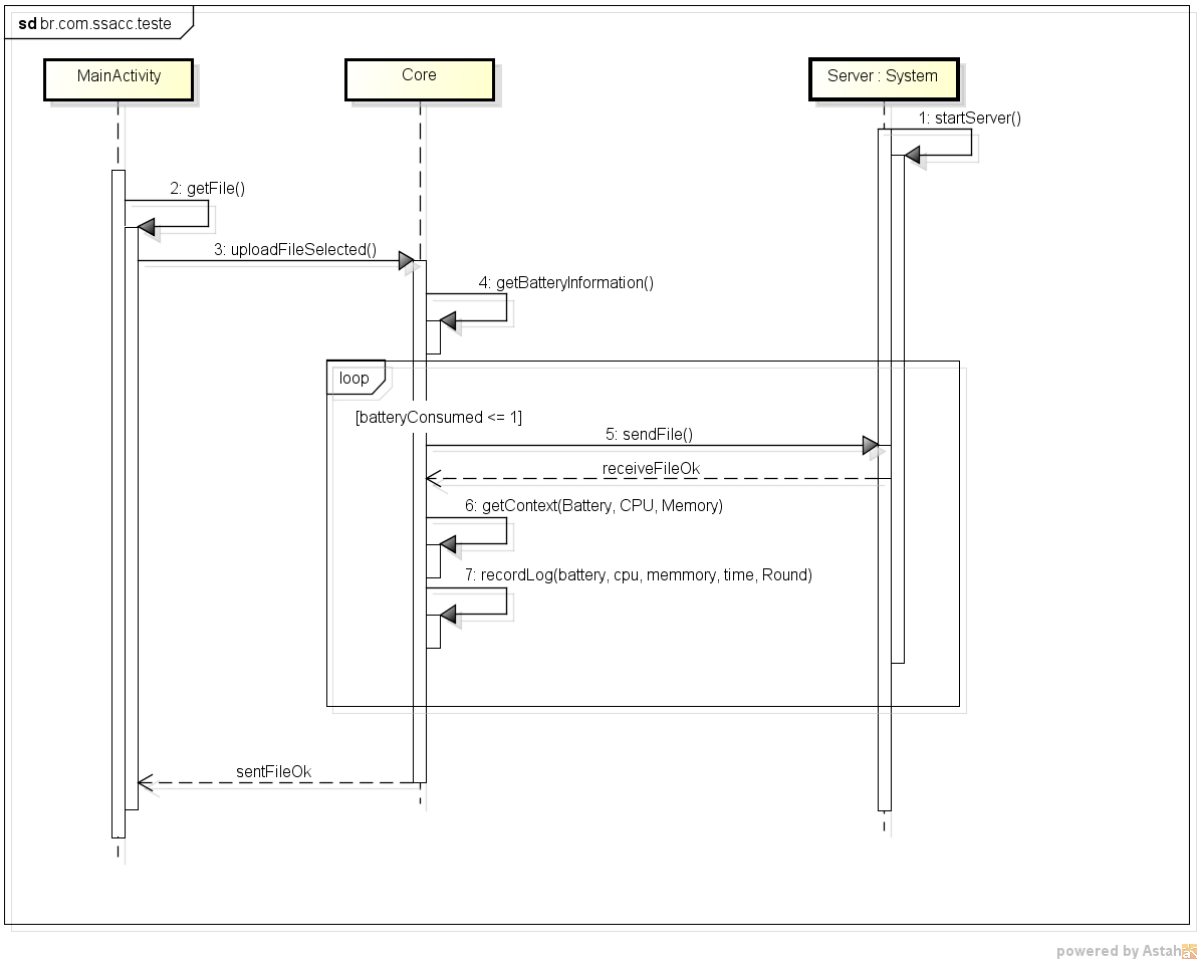

Figura 3.2: Diagrama de sequência do App stress.SSACC

3.2.2 Arquitetura do Aplicativo SSACC

O serviço do SSACC, como foi dito anteriormente na seção 3.2.1, precisará de métricas que serão formadas pelo aplicativo *stress*.SSACC. Com base nessas métricas coletadas dará forma ao módulo mais importante do aplicativo SSACC, a tomada de decisão. A figura 3.3 e 3.4 ilustra os diagramas que servirão para o aplicativo SSACC. A diferença entre os dois diagramas exibidos é que um demonstra o método de *upload* e o outro de *download*.

Na figura 3.3 e 3.4 descrevem a sequência de execução do aplicativo SSACC, a figura 3.3 demontra o processo de upload e a figura 3.4 demonstra o processo de download. A sequência inicia pelo cliente escolhendo o arquivo ou mensagem que deseja enviar a nuvem, o DM selecionará o arquivo, irá colher a informação de bateria, logo após o algoritmo de decisão previamente implementado irá escolher o algoritmo de criptografia a ser usado. O DM então criptografa a mensagem e envia

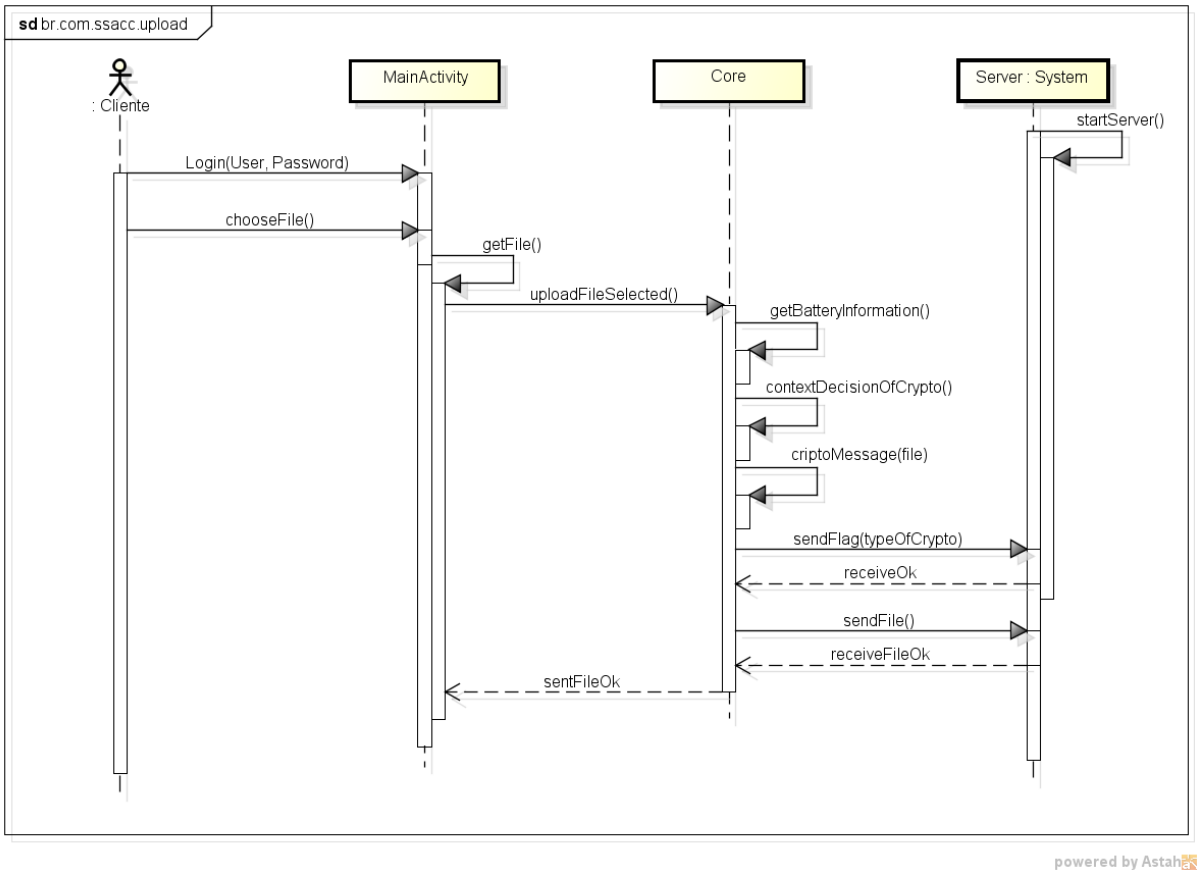

Figura 3.3: Diagrama de sequência de upload do App SSACC

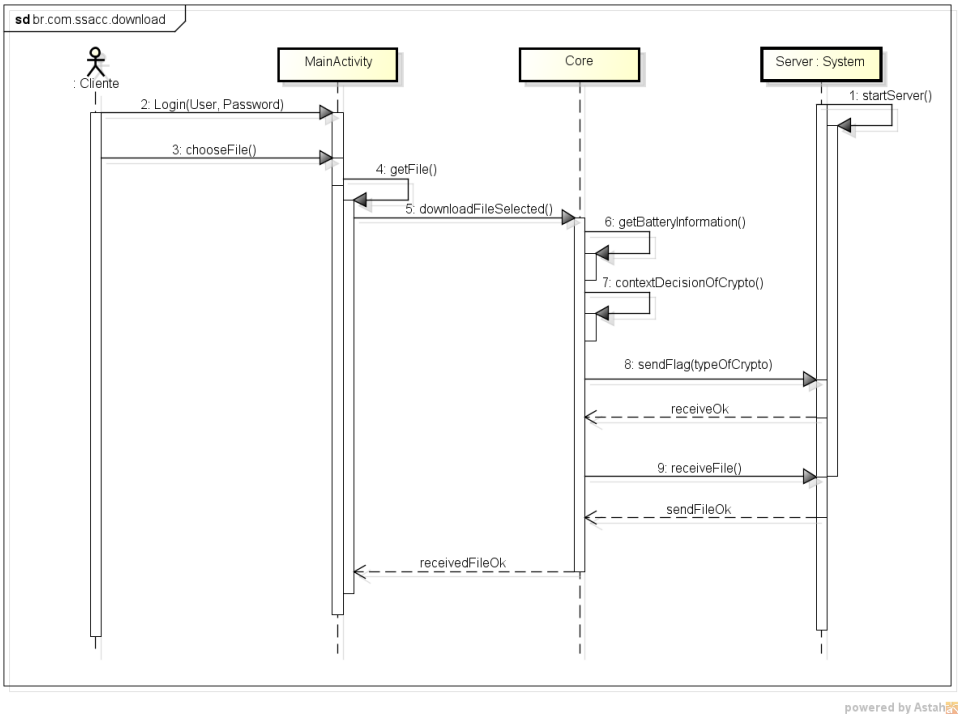

Figura 3.4: Diagrama de sequência de download do App SSACC

uma notificação ao servidor indicando que a única criptografia disponível é aquela selecionada pelo algoritmo de decisão. Com isso o servidor sinaliza que está pronto para a transferência e esta ocorre, no final o DM indica ao usuário que foi enviado ou recebido a mensagem.

O texto a seguir foi traduzido e adaptado de [26] e representa a tradução da figura 3.5 de [26].

1. O cliente envia uma mensagem *Hello* para o servidor.
2. A mensagem inclui uma lista de algoritmos suportados pelo cliente (No caso do aplicativo SSACC o algoritmo predeterminado) e um número aleatório que irá ser utilizado para gerar as chaves.
3. O servidor responde enviando uma mensagem *Hello* para o cliente. Esta mensagem inclui:
   - O algorítimo que será usado (No caso do aplicativo SSACC o algoritmo predeterminado). O servidor seleciona a partir da lista que foi enviada pelo cliente.
   - Um número aleatório, que irá ser utilizado para gerar as chaves.
4. O servidor envia seu certificado para o cliente.
5. O cliente autentica o servidor usando o certificado do servidor.
6. O cliente gera um valor aleatório ("*pre-master secret*"), criptografa usando a chave pública do servidor e envia para o servidor.
7. O servidor usa sua chave privada para descriptografar a mensagem para recuperar o *pre-master secret*.
8. O cliente e servidor calculam separadamente as chaves que irão ser utilizadas na sessão SSL.

   Estas chaves não são trocadas entre cliente e servidor, porque as chaves são calculados com base no *pre-master secret* e os números aleatórios, que são conhecidos para cada lado. As chaves incluem:

- Chave de criptografia que o cliente usa para criptografar os dados antes de enviá-lo para o servidor.
- Chave de criptografia que o servidor usa para criptografar os dados antes de enviá-lo ao cliente.
- Chave que o cliente usa para criar uma mensagem de dados compilados.
- Chave que o servidor usa para criar uma mensagem de dados compilados.

As chaves de criptografia são simétricas, ou seja, a mesma chave é usada para criptografar e descriptografar os dados.

9. O cliente e o servidor enviam uma mensagem "finalizado"para o outro. Estas são as primeiras mensagens que são enviadas usando as chaves geradas na etapa anterior (as primeiras mensagens de "seguro").

   A mensagem de finalização inclui todas as mensagens de "aperto de mão"anteriores que foram enviados do cliente e do servidor. Em cada lado, verifica se as mensagens anteriores que receberam combinam com as mensagens de finalização. Este verifica se as mensagens de "aperto de mão"não foram adulterados.

10. O cliente e o servidor agora transferem os dados usando a criptografia, as chaves *hash* e os algoritmos.

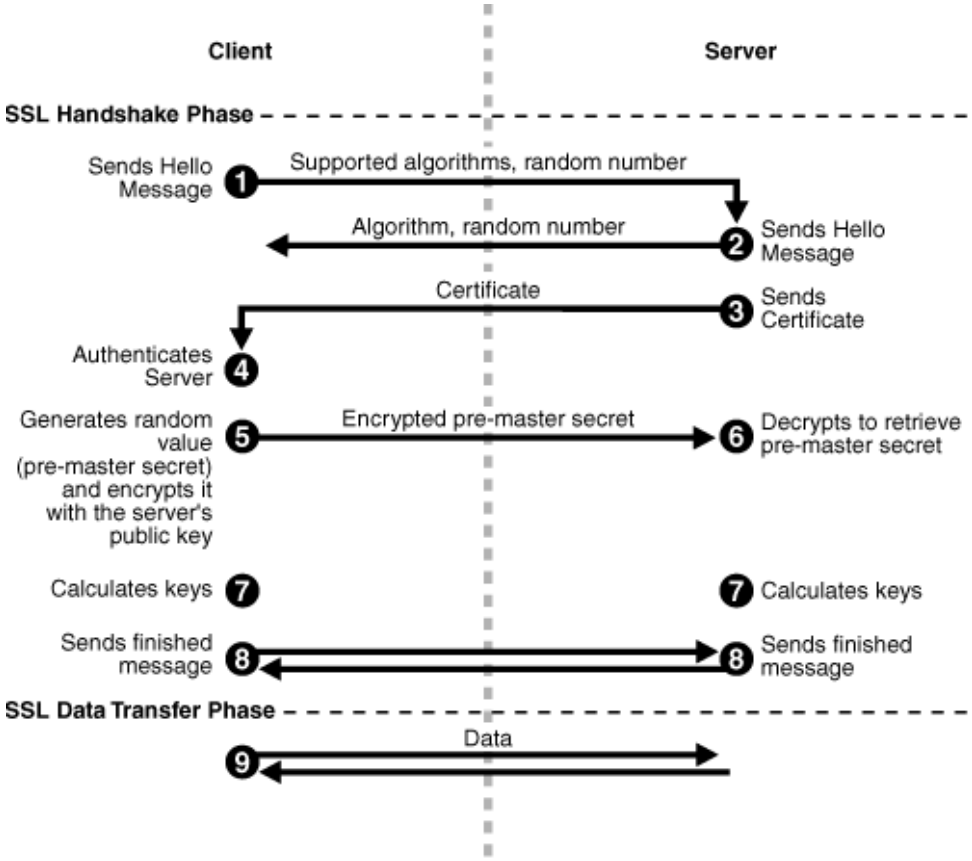

Figura 3.5: Descrição do Handshake SSL

4 Implementações e Resultados

Este capítulo aborda a implementação do protótipo de teste e o aplicativo final do modelo proposto na pesquisa. Para chegar ao objetivo final, foi usado linguagem de programação Java e *eXtensible Markup Language* (XML) para programação na plataforma Android, e linguagem de programação Java *Enterprise Edition* para implementação do serviço da nuvem.

Primeiramente será apresentado o ambiente usado, descrito na seção 4.1, na seção **??** deste capítulo será apresentado o serviço implementado para a nuvem computacional com o objetivo de avaliar os algoritmos criptográficos. O aplicativo de teste de *stress* dos algoritmos de criptografia usados será apresentado na seção 4.3, em seguida será apresentado os resultados dos testes. Com base nesses dados obtidos nos testes foi implementada a tomada de decisão e usado para a implementação do aplicativo final que será apresentado na seção 4.5. Na seção 4.4 será apresentado os resultados dos aplicativo *stress*.SSACC.

O projeto foi todo implementado no ambiente de desenvolvimento *Eclipse Luna* com *Java Platform Enterprise Edition* (Java EE) e *Android Development Tools* (ADT).

4.1 Ambiente Usado

Para a realização do trabalho, foi configurado no LABSAC da UFMA uma infraestrutura de nuvem IaaS para os testes. Também foi utilizado para testes o DM da marca Motorola com duas versões do sistema operacional, com todos os processos de segundo plano encerrados e os testes feitos em sequência para que não haja variações indesejáveis no hardware e software, com exceção da instalação dos protótipos criados para proposta. Na figura 4.1 é apresentado o ambiente físico de teste, e a tabela 4.1 apresenta as configurações dos dispositivos usados no ambiente de teste.

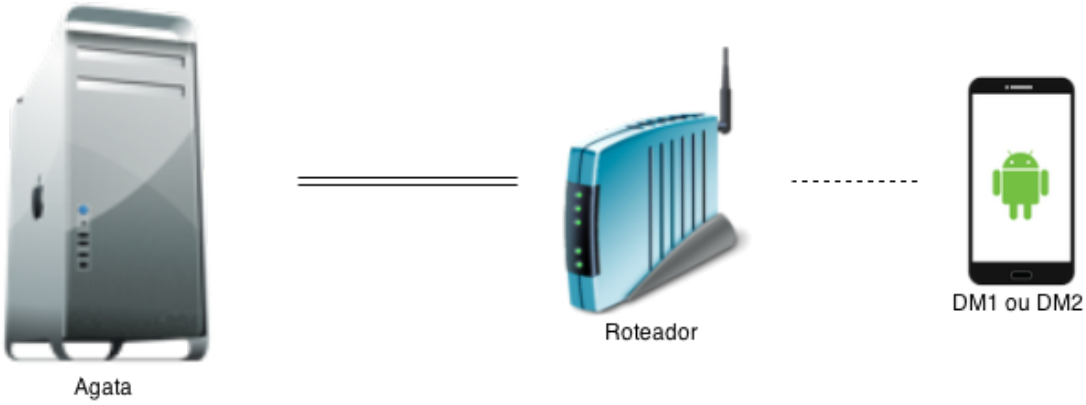

Figura 4.1: Diagrama do ambiente físico usado para testes dos aplicativos

Nome	Configuração de *hardware*	Configuração de *software*
Ágata	Core i7; 8 Gb Ram; 1 Tb HD	*Linux CentOS* 12.03 *server*, *EUCALYPTUS*
Moto G (DM1)	Processador *SnapDragon QuadCore* 1.2Ghz ; 1Gb Ram; 16Gb Memória de armazenamento	Android 4.4.4
Moto G (DM2)	Processador *SnapDragon QuadCore* 1.2Ghz ; 1Gb Ram; 16Gb Memória de armazenamento	Android 5.0.2

Tabela 4.1: Tabela de descrição dos dispositivos de teste

4.2 Implementação do Serviço da Nuvem

A implementação do serviço da nuvem foi feito no *Eclipse Luna* e usado o Java EE. O serviço teve como base o fluxograma mostrado na figura 4.2, que demonstra que no início do serviço haverá o inicio de todos os processos, em seguida ele fará um *loop* verificando se há alguma requisição, quando tiver a requisição o serviço iniciará o processo padrão de *handshake* do SSL e fará a sincronização dos dados requisitados.

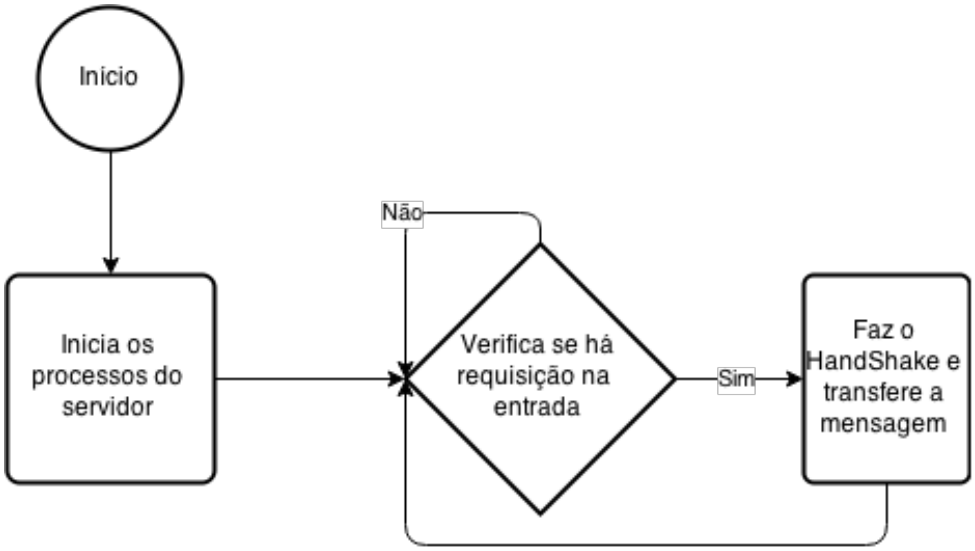

Figura 4.2: Fluxograma do serviço da nuvem

4.2.1 Implementação do Método *sslInit*

No apêndice A.1.1 é apresentado o código da classe Servidor.java, o método *sslInit* é onde o servidor configura e inicializa o *TrustManagerFactory*, Esta classe funciona como uma fábrica para os gestores de confiança com base em uma fonte de material confiança. Cada gerenciador de confiança administra um tipo específico de material de confiança para ser usado por *secure sockets*. O material de confiança é baseada em um armazenamento de chave e/ou provedores de fontes específicas.

A classe *KeyManagerFactory* funciona como uma fábrica para os principais gestores de chaves com base em uma fonte de *key material*. Cada gerente de chave administra um tipo específico de *key material* para uso por *secure sockets*. O material de chave é baseada em um armazenamento de chave e/ou fontes específicas do provedor.

Instâncias da classe *SSLContext* representam uma implementação do protocolo *Secure Socket* que atua como uma fábrica de 'fábricas de *Secure Socket*' ou *SSLEngines*. Esta classe é inicializada com um conjunto opcional de chave e gerentes de confiança e fonte de bytes aleatórios seguras.

4.3 Implementação do Teste de Stress

A figura 4.3 apresenta o fluxograma da aplicação *stress*.SSACC, onde o primeiro processo é verificar o estado da bateria, após isso faz uma verificação para que

a carga da bateria esteja entre os limites de 30 e 80 % da bateria. Estes limites foram empregados para que possamos ter um teste mais seguro de que não haja variação entre a medição e a carga real da bateria. Se a bateria não estiver nesse limite, o APP envia imprime na tela do DM que não é possível prosseguir com o teste por causa dos limites de energia da bateria.

Caso seja averiguado estar nos limites da bateria então o APP prosseguirá para o próximo processo, a seleção da criptografia que será testada, depois verificará se já foram selecionadas todas as criptografias disponíveis. Se ainda tiver criptografia para ser testada ele prosseguirá com o teste, criptografando e enviando a mensagem ao servidor, logo depois verificará se a bateria teve gasto de energia equivalente a 1%, se não teve irá repetir o processo de criptografia e envio ao servidor, se teve o gasto então voltará para o processo de verificação dos limites da bateria. Se o APP já tiver selecionado todas as criptografias então ele gravará o *log* dos testes e finalizará o APP.

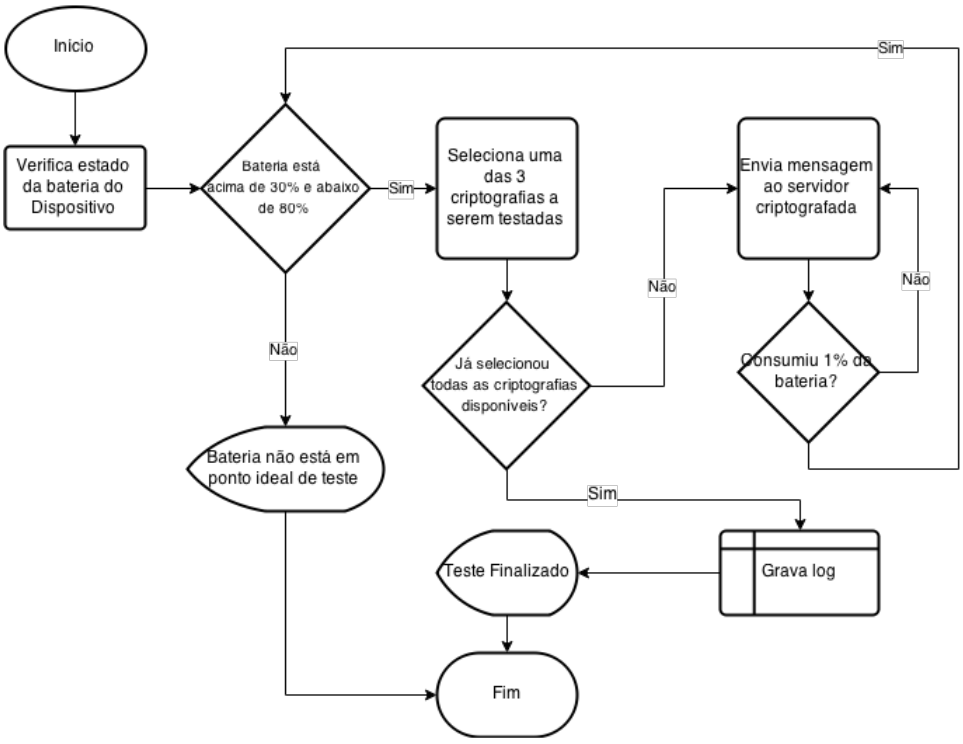

Figura 4.3: Fluxograma do aplicativo *stress*.SSACC

4.3.1 Implementação do Método onClick

Em um trecho de código no apêndice A.2.1 apresenta a ação do botão de "Enviar arquivo", realizado pela classe *MainActivity* do projeto. Neste código obtém-

se os parâmetros de conexão (IP, Porta) obtidos pela classe nativa *SharedPreferences* do *Android* a qual realiza a persistência dos dados inseridos pelo usuário. Após realiza-se as validações de preenchimento desses campos e enviá-los para a classe que realiza a conexão SSL com o servidor.

4.3.2 Implementação da Classe *SocketSSACC.java*

Em um trecho de código no apêndice A.2.2 apresenta o código da classe *SSLInit* onde o cliente configura e inicializa o *TrustManagerFactory*. Esta classe funciona como uma fábrica para os gestores de confiança com base em uma fonte de material confiança. Cada gerenciador de confiança administra um tipo específico de material de confiança para ser usado por *secure sockets*. O material de confiança é baseada em um armazenamento de chave e/ou provedores de fontes específicas.

A classe *KeyManagerFactory* funciona como uma fábrica para os principais gestores de chaves com base em uma fonte de *key material*. Cada gerente de chave administra um tipo específico de *key material* para uso por *secure sockets*. O material de chave é baseada em um armazenamento de chave e/ou fontes específicas de provedor.

Instâncias da classe *SSLContext* representam uma implementação do protocolo *Secure Socket* que atua como uma fábrica de 'fábricas de *Secure Socket*' ou *SSLEngines*. Esta classe é inicializada com um conjunto opcional de chave e gerentes de confiança e fonte de bytes aleatórios seguras. Este é inicializado com o protocolo TSL e uma lista de gestores de confiança.

Em um trecho de código no apêndice A.2.2 apresenta o *SSLSocketFactory* que atua como uma fábrica para a criação de bases seguras, esta classe é uma subclasse abstrata de *javax.net.SocketFactory* e encapsula os detalhes da criação e configura inicialmente *secure sockets*. Isso inclui chaves de autenticação, a validação do certificado de pares, conjuntos de cifras habilitados e afins. Foi criado a partir do *SSLContext*, logo após especificando o ip e porta e criando em seguida o o *socket* para iniciar o *handshake* com o servidor assim estabelecendo a conexão.

O *startHandshake* é chamado iniciando o processo de *handshake* e quando o *handshake* é concluído, este será assinalado como um evento. Este método é síncrono

para o *handshake* inicial em uma conexão, e retorna quando o *handshake* negociado está completo.

4.3.3 Implementação da Classe de Contexto de Bateria

O código no apêndice A.2.3 apresenta a classe *BatteryInformation* que obtém o contexto de bateria, a qual é uma extensão da Classe *BroadCastReceiver*. É necessário registrar um *Intent* para que a aplicação possa receber a informação do nível da bateria quando for alterada, faz-se isso adicionando no método *onCreate* da *MainActivity*. A instância da Classe *BatteryManager* obtém os detalhes da bateria. O resultado do contexto da bateria é enviado pelo método dadosBateria() o qual envia o nível atual da bateria para a classe *TesteStress* que utilizará para manter o log atualizado.

4.3.4 Implementação do *AndroidManifest*

O código no apêndice A.2.5 apresenta o *AndroidManifest.XML* que contém informações essenciais sobre nosso aplicativo tais como, nome do pacote *Java*, as permissões que o aplicativo possui, qual a versão mínima da *Application Programming Interface* (API) do *Android*. Dentro da tag <*manifest*> declaramos o pacote principal do projeto, utilizando a tag <*package*>. As *activity's* do projeto estão declaradas na *tag activity*, para que possa ser utilizada. Para declarar a *activity* é utilizada a tag <*activity*>, que recebe o nome da classe, e é sempre relativa ao pacote principal. As autorizações para que o aplicativo possa acessar o *Security Digital Card* (sdCard), uso de rede, *internet* foi realizada pela *tag* <*uses-permission android:name*>.

4.4 Resultados do Teste de *Stress*

A média dos resultados colhidos pelo aplicativo *stress*.SSACC nos dois dispositivos são apresentados nas tabelas 4.2 e 4.3. Os testes foram realizados conforme descrito na seção 4.1, e usados arquivos para teste de tamanhos que variaram em 256 Mb, 512 Mb e 1024 Mb. Foram feitos vários testes para cada arquivo, criptografia e DM.

As tabelas de resultados mostram a média dos testes realizados em cada DM, foi constatado uma consistência nos dados obtidos assim como esperado. Em todos os testes foi claramente visto que o algoritmo criptográfico RC4 teve mais rounds com a mesma quantidade de energia utilizada nas outras criptografias, e o algoritmo criptográfico AES obteve maior quantidade de rounds que o algoritmo 3DES.

A conclusão dos testes indicou que por ter uma quantidade maior de rounds em uma mesma quantidade de energia utilizada, o algoritmo RC4 é aquele que consome menos recursos do DM, seguido do AES e o 3DES se mostrou o maior consumidor de recursos dentre os algoritmos testados. Portanto o algoritmo de decisão que será implementado no serviço SSACC ficou definido como mostrado na tabela 4.4, quando o DM estiver com mais de 2/3 de energia na bateria disponível será usado a criptografia 3DES por ser a que consome mais recursos, quando o DM estiver entre 1/3 e 2/3 de energia na bateria disponível, será usado a criptografia AES, e quando tiver menos de /3 de energia na bateria disponível, será usado a criptografia RC4 por consumir menos recursos.

Teste	RC4	AES	3DES
256 Mb (1º teste)	875 Rounds	792 Rounds	571 Rounds
512 Mb (1º teste)	606 Rounds	477 Rounds	404 Rounds
1024 Mb (1º teste)	418 Rounds	342 Rounds	261 Rounds

Tabela 4.2: Média dos resultados de testes feito no DM 1

Teste	RC4	AES	3DES
256 Mb (2º teste)	870 Rounds	790 Rounds	575 Rounds
512 Mb (2º teste)	603 Rounds	519 Rounds	425 Rounds
1024 Mb (2º teste)	370 Rounds	336 Rounds	241 Rounds

Tabela 4.3: Média dos resultados de testes feito no DM 2

RC4	AES	3DES
abaixo de 1/3 de energia	entre 1/3 e 2/3 de energia	2/3 ou mais de energia

Tabela 4.4: Algoritmo de decisão

4.5 Implementação do Aplicativo Final

A figura 4.4 apresenta o fluxograma do aplicativo SSACC, onde o primeiro processo é verificar o estado da bateria, após isso ele executará o algoritmo de decisão. Se a bateria tiver acima de 2/3 da quantidade de energia total disponível, o algoritmo decidirá pelo algoritmo criptográfico RC4, se a energia disponível estiver entre 1/3 e 2/3 ele decidirá pelo algoritmo criptográfico AES, caso a energia esteja abaixo de 1/3 será decidido o uso do algoritmo criptográfico 3DES. Após essa execução a mensagem será criptografada e enviada ao servidor.

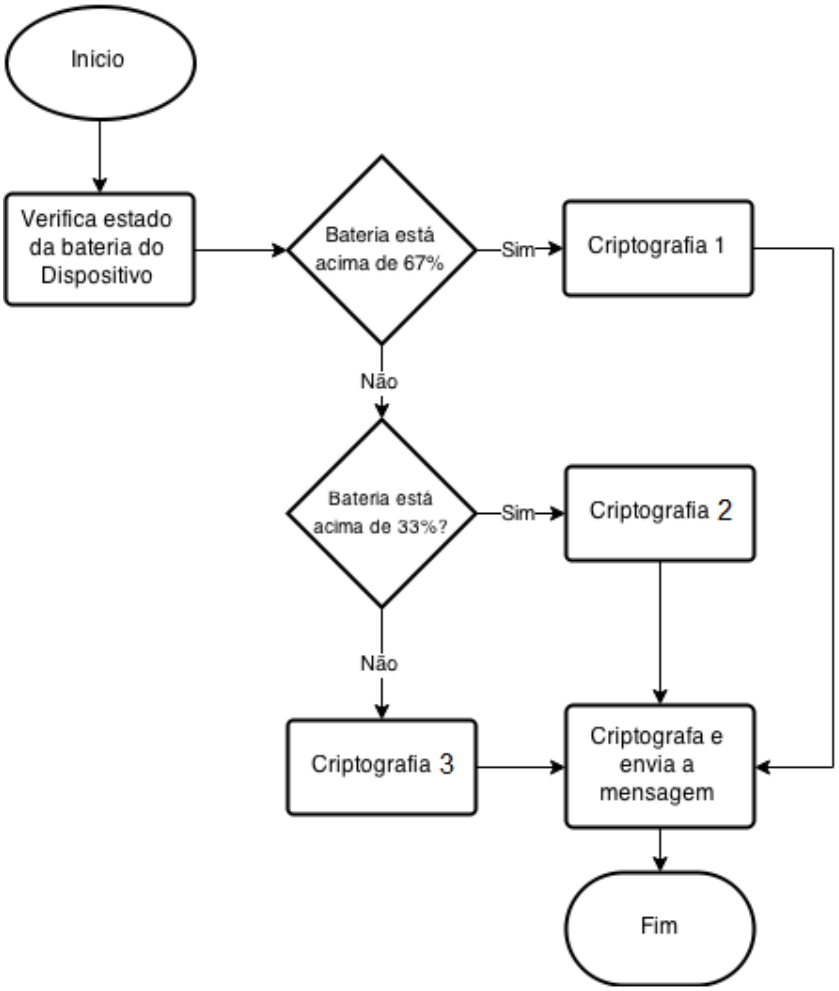

Figura 4.4: Fluxograma do aplicativo SSACC

Este programa seguiu como modelo o aplicativo de teste, onde houve mudanças na classe *SocketSSL*.java, foi retirado a classe *ArchiveTestSaver*.java e implementado uma nova classe que fará o papel do algoritmo de decisão denominada de *ChooseThepath*.java. Portanto será comentado somente sobre os diferenciais entre o aplicativo *stress*.SSACC e o aplicativo final SSACC

4.5.1 Implementação da classe *SocketSSL*.java

Nesta classe que pode ser vista no apêndice A.3.2 podemos verificar que houve mudanças em relação a forma de execução, retirando o *loop* de teste e apenas aplicando a criptografia que será utilizada na sessão, e a retirada do *log* de teste.

4.5.2 Implementação da classe *ChooseThePatch*.java

Esta classe foi criada para fazer o papel de algoritmo de decisão, decidindo qual será o algoritmo criptográfico usado na sessão atual com base na quantidade de energia disponível no DM. Onde acima de 2/3 de energia disponível será escolhido o algoritmo criptográfico RC4, se a quantidade de energia disponível estiver entre 1/3 e 2/3 será decidido o uso do algoritmo criptográfico AES, e abaixo de 1/3 da quantidade de energia disponível no DM, será usado o algoritmo criptográfico 3DES.

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

Na atualidade tem-se um número crescente de uso de DM para comunicação via *internet*, e também um número crescente nos ataques de qualquer natureza, principalmente os do tipo *scan*.

Umas das grandes dificuldades hoje nos DM's é a quantidade de energia disponível limitada, portanto qualquer forma de economia de energia é consideravelmente válido para os DM's. Atualmente se tem melhorado bastante o hardware para que este se torne cada vez mais poderoso e econômico, mas também estão sendo criados novos métodos de programação para haja uma economia do consumo de bateria por parte das aplicações.

O SSL é uma tecnologia já bem difundida e muito usada para conexões cliente x servidor, mas é uma tecnologia que não visa diretamente diminuir o consumo de energia, somente fornecer um canal seguro a qualquer custo.

Diante disto, o trabalho realizado nessa dissertação foi adicionar uma forma de analisar o nível de bateria do dispositivo e decidir qual a melhor criptografia a ser usada para aquele estado em que se encontra.

A principal contribuição deste é trabalho foi criar um canal seguro com as seguinte características:

- Utilização de criptografias de chave pública;
- Utilização de criptografias de chave privada;
- Utilização de certificados digitais;
- Utilização da função *hash* criptografada;

- Utilização de ciência do contexto para definir qual criptografia pode ser usada naquele estado;
- Diminuir o consumo de bateria.

O trabalho garante os seguintes serviços básicos:

- Integridade;
- Autenticidade da origem;
- Não-repúdio;
- Privacidade.

5.2 Limitação

Nesta foi proposta foram detectados as seguintes limitações:

- Levar em considerações outras ciências do contexto do DM;
- Adicionar mais algoritmos para a tomada de decisão, tais como quantidade de memória e *Central Processing Unit* (CPU) disponíveis, entre outros.

5.3 Trabalhos Futuros

No decorrer do desenvolvimento deste trabalho identificou-se alguns pontos relevantes que podem ser considerados para trabalhos futuros, tais como:

- Utilizar outros tipos de contexto para uma tomada de decisão mais efetiva, como:
  - Quantidade disponível de processador,
  - Quantidade disponível de memória,
  - Tipo de rede e segurança da mesma,
  - Qualidade de conexão com a *internet*.

- Criar um algoritmo para que faça uma análise de relevância de cada contexto, para que a tomada de decisão seja mais eficaz;

- Adicionar mais algoritmos de criptografia.

Referências Bibliográficas

[1] *ICAC-INDST '09: Proceedings of the 6th International Conference Industry Session on Autonomic Computing and Communications Industry Session*, New York, NY, USA, 2009. ACM. 102091.

[2] A. Abuhussein, H. Bedi, and S. Shiva. Evaluating security and privacy in cloud computing services: A stakeholder's perspective. In *Internet Technology And Secured Transactions, 2012 International Conference for*, pages 388–395, Dec 2012.

[3] W. Accenture. *Cloud Computing and Sustainability: The Environmental Benefits of Moving to the Cloud*, 2010. Disponível em: http://www.accenture.com/SiteCollectionDocuments/PDF/Accenture_Sustainability_Cloud_Computing_TheEnvironmentalBenefitsofMovingtotheCloud.pdf . Acessado em: 01-09-2014.

[4] M. Bazire and P. Brézillon. Understanding context before using it. In *Modeling and using context*, pages 29–40. Springer, 2005.

[5] A. M. Bernardos, P. Tarrío, and J. R. Casar. A data fusion framework for context-aware mobile services. In *Multisensor Fusion and Integration for Intelligent Systems, 2008. MFI 2008. IEEE International Conference on*, pages 606–613. IEEE, 2008.

[6] M. A. Bishop. The art and science of computer security. 2002.

[7] R. Buyya, A. Beloglazov, and J. Abawajy. Energy-efficient management of data center resources for cloud computing: A vision, architectural elements, and open challenges. *arXiv preprint arXiv:1006.0308*, 2010.

[8] A. F. M. Carvalho. M-code: Um modelo para medição de confidencialidade e desempenho para aplicações móveis seguras. 2008.

[9] CERT. *Cartilha de Segurança para Internet*. CERT, Centro de Estudos, Resposta e Tratamento de Incidentes de Segurança no Brasil, Jun 2012. Disponível em: http://cartilha.cert.br. Acessado em: 11-08-2014.

[10] CERT. *Incidentes Reportados ao CERT.br*. CERT, Centro de Estudos, Resposta e Tratamento de Incidentes de Segurança no Brasil, Fev 2014. Disponível em: www.cert.br/stats/incidentes. Acessado em: 11-08-2014.

[11] H. Chen, T. Finin, A. Joshi, L. Kagal, F. Perich, and D. Chakraborty. Intelligent agents meet the semantic web in smart spaces. *Internet Computing, IEEE*, 8(6):69–79, 2004.

[12] 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.

[13] G. F. Coulouris, J. Dollimore, and T. Kindberg. *Distributed systems: concepts and design*. pearson education, 2005.

[14] A. K. Dey. Understanding and using context. *Personal and ubiquitous computing*, 5(1):4–7, 2001.

[15] D. ELETRICIDADE and J. D. ARAÚJO. Um modelo de detecção de intrusão para ambientes de computação em nuvem. 2013.

[16] M. Fiorio, C. O. Emmanoel, P. F. Pires, and F. C. Delicato. Soluções para o desenvolvimento de sistemas seguros, 2009.

[17] D. R. Herrick and M. R. Ritschard. Greening your computing technology, the near and far perspectives. In *Proceedings of the 37th annual ACM SIGUCCS fall conference*, pages 297–304. ACM, 2009.

[18] P. Hu, J. Indulska, and R. Robinson. An autonomic context management system for pervasive computing. In *Pervasive Computing and Communications, 2008. PerCom 2008. Sixth Annual IEEE International Conference on*, pages 213–223. IEEE, 2008.

[19] Y. Jadeja and K. Modi. Cloud computing - concepts, architecture and challenges. In *Computing, Electronics and Electrical Technologies (ICCEET), 2012 International Conference on*, pages 877–880, March 2012.

[20] D. Martin, C. Lamsfus, and A. Alzua. Automatic context data life cycle management framework. In *Pervasive Computing and Applications (ICPCA), 2010 5th International Conference on*, pages 330–335. IEEE, 2010.

[21] F. Martin. *SSL Certificates HOWTO*, 10 2002. Disponível em: http://www.tldp.org/HOWTO/SSL-Certificates-HOWTO. Acessado em: 01092014.

[22] P. Mathur and N. Nishchal. Cloud computing: New challenge to the entire computer industry. In *Parallel Distributed and Grid Computing (PDGC), 2010 1st International Conference on*, pages 223–228. IEEE, 2010.

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

[24] V. Menezes. Oorschot:"handbook of applied cryptography", 1997. *CRC Press*, 200:6–10, 1997.

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

[26] Oracle. first edition, aug 2014. Disponível em: http://docs.oracle.com. Acessado em: 23-08-2014.

[27] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos. Context aware computing for the internet of things: A survey. *Communications Surveys & Tutorials, IEEE*, 16(1):414–454, 2014.

[28] S. Pietschmann, A. Mitschick, R. Winkler, and K. Meißner. Croco: Ontology-based, cross-application context management. In *Semantic Media Adaptation and Personalization, 2008. SMAP'08. Third International Workshop on*, pages 88–93. IEEE, 2008.

[29] S. Ramgovind, M. M. Eloff, and E. Smith. The management of security in cloud computing. In *Information Security for South Africa (ISSA), 2010*, pages 1–7. IEEE, 2010.

[30] F. Ramparany, R. Poortinga, M. Stikic, J. Schmalenstroer, and T. Prante. An open context information management infrastructure-the ist-amigo project. 2007.

[31] R. M. RICHTER. *TI Verde: Sustentabilidade por meio da Computação em Nuvem*. Disponível em: http://www.centropaulasouza.sp.gov.br/pos-graduacao/workshop-de-pos-graduacao-e-pesquisa/007-workshop-

2012/workshop/trabalhos/desenvgestti/ti-verde-sustentabilidade.pdf. Acessado em: 01-09-2014.

[32] B. P. Rimal, E. Choi, and I. Lumb. A taxonomy and survey of cloud computing systems. In *INC, IMS and IDC, 2009. NCM'09. Fifth International Joint Conference on*, pages 44–51. Ieee, 2009.

[33] L. S. Rocha, C. Castro, J. C. Machado, and R. M. Andrade. Utilizando reconfiguração dinámica e notificação de contextos para o desenvolvimento de software ubíquo. *XXI Simpósio Brasileiro de Engenharia de Software*, pages 219–235, 2007.

[34] S. Ruth. Green it more than a three percent solution? *Internet Computing, IEEE*, 13(4):74–78, 2009.

[35] N. S. Ryan, J. Pascoe, and D. R. Morse. Enhanced reality fieldwork: the context-aware archaeological assistant. In *Computer applications in archaeology*. Tempus Reparatum, 1998.

[36] F. Sabahi. Cloud computing security threats and responses. In *Communication Software and Networks (ICCSN), 2011 IEEE 3rd International Conference on*, pages 245–249, May 2011.

[37] 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.

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

[39] F. Shaikh and S. Haider. Security threats in cloud computing. In *Internet Technology and Secured Transactions (ICITST), 2011 International Conference for*, pages 214–219, Dec 2011.

[40] W. Stallings. Network security essentials: Applications and standards, 2000.

[41] C. Taurion. *Cloud Computing-Computação em Nuvem*. Brasport, 2009.

[42] S. Thomas. Ssl and tls essentials. *New York*, 2000.

[43] J. Werner et al. Uma abordagem para alocação de máquinas virtuais em ambientes de computação em nuvem verde. 2011.

[44] M. Zhou, R. Zhang, D. Zeng, and W. Qian. Services in the cloud computing era: A survey. In *Universal Communication Symposium (IUCS), 2010 4th International*, pages 40–46, Oct 2010.

A Apêndice

A.1 Códigos do Serviço da Nuvem

A.1.1 Código do Servidor.java

```
1  import java.io.FileInputStream;
2  import java.io.FileOutputStream;
3  import java.io.IOException;
4  import java.io.InputStream;
5  import java.io.ObjectInputStream;
6  import java.net.InetAddress;
7  import java.net.ServerSocket;
8  import java.net.Socket;
9  import java.security.KeyStore;
10 import java.util.Calendar;
11 import java.util.regex.Matcher;
12 import java.util.regex.Pattern;
13
14 import javax.net.ssl.KeyManagerFactory;
15 import javax.net.ssl.SSLContext;
16 import javax.net.ssl.SSLServerSocketFactory;
17 import javax.net.ssl.SSLSocket;
18 import javax.net.ssl.TrustManagerFactory;
19 import javax.swing.JFrame;
20 import javax.swing.JOptionPane;
21
22
23
24 public class Servidor {
25     private static SSLServerSocketFactory sssf;
26     // private static SSLServerSocket sss;
27     private static ServerSocket sss;
```

```
28
29      public static void main(String[] args) {
30          // porta do server
31          int port = 8006;
32          sslInit();
33
34          try {
35              // Create the secure server socket
36              sss = sssf.createServerSocket(port);
37              // sss = new ServerSocket(8006);
38              System.out.println("IP do servidor: "+ InetAddress.
                    getLocalHost().getHostAddress());
39              System.out.println("setando porta: "+ port);
40              System.out.println(">> Thread iniciada, esperando por
                    conexao... ---\n");
41
42              while (true) {
43                  // espera conexão de algum cliente
44                  SSLSocket s = (SSLSocket) sss.accept();
45
46                  receiveConfigToClient(s);
47
48                  new ArchiveReceiver(s).start();
49              }
50
51          } catch (IOException e) {
52              // Imprime uma notificação na saída padrão caso haja algo
                    errado.
53              // System.out
54              // .println("Algum problema ocorreu para criar ou receber
                    o socket.");
55          }
56      }
57
```

```
58      private static void receiveConfigToClient(SSLSocket s) throws
            IOException {
59          InputStream inputStream = s.getInputStream();
60
61          byte buf[] = new byte[4];
62          inputStream.read(buf);
63          // inputStream.close();
64
65          // String CRIP = new String(buf, "UTF-8");
66          // CRIP = CRIP.trim();
67
68          // Pick all AES algorithms of 256 bits key size
69          String patternString = new String(buf, "UTF-8");
70          patternString = patternString.trim();
71          System.out.println("
                ------------------------------------------------------------
                ");
72          System.out.println(">" + patternString + "|");
73
74          // if (CRIP.equals("DES"))
75          // patternString = "DES.*64";
76          // else
77          // patternString = CRIP + ".*128";
78          //
79          // System.out.println("> Pattern: " + patternString);
80
81          Pattern pattern = Pattern.compile(patternString);
82          Matcher matcher;
83          boolean matchFound;
84
85          String str[] = { "SSL_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA", "
                SSL_RSA_WITH_3DES_EDE_CBC_SHA",
86                  "TLS_RSA_WITH_AES_128_CBC_SHA", "
                        SSL_RSA_WITH_RC4_128_MD5" };
87          // for (int i = 0; i < str.length; i++)
```

```
88          // System.out.println(str[i]);
89
90          int len = str.length;
91          String set[] = new String[1];
92
93          int j = 0, k = len - 1;
94          for (int i = 0; i < len; i++) {
95              // Determine if pattern exists in input
96              matcher = pattern.matcher(str[i]);
97              matchFound = matcher.find();
98
99              if (matchFound) {
100                 set[0] = str[i];
101                 System.out.println("Match: " + str[i]);
102                 break;
103             }
104             // else
105             // set[k--] = str[i];
106         }
107
108         s.setEnabledCipherSuites(set);
109
110         str = s.getEnabledCipherSuites();
111
112         System.out.println("\nAvailable Suites after Set:");
113         for (int i = 0; i < str.length; i++)
114             System.out.println(str[i]);
115         System.out.println("Using cipher suite: " + (s.getSession()).
               getCipherSuite());
116     }
117
118     private static class ArchiveReceiver extends Thread {
119         private final Socket s;
120
121         public ArchiveReceiver(final SSLSocket socket) {
```

```
122             s = socket;
123         }
124 
125         @Override
126         public void run() {
127             String dateTime = String.format("%1$tF %1$tT", Calendar.
                    getInstance());
128 
129             System.out.println("Começando a transferência (iniciada
                    em: " + dateTime + ")");
130             String nomeArqTeste = "TESTE_RENATO.txt";
131 
132             try {
133                 ObjectInputStream in = new ObjectInputStream(s.
                        getInputStream());
134 
135                 // String fileName = in.readUTF();
136                 // long size = in.readLong();
137 
138                 // System.out.println("Recebendo arquivo: " +
                        fileName + " - " + size + " bytes.");
139 
140                 FileOutputStream fos = new FileOutputStream(
                        nomeArqTeste);
141                 byte buffer[] = new byte[4096];
142                 while (true) {
143                     int len = in.read(buffer);
144                     if (len == -1)
145                         break;
146 
147                     fos.write(buffer, 0, len);
148                 }
149                 fos.flush();
150                 fos.close();
151 
```

```
152
153                 System.out.println("Pronto em : " + dateTime);
154                 System.out.println("------------------------------
                       ---------------------------\n");
155
156                 s.close();
157                 JOptionPane.showMessageDialog(null, "A Transferência
                       Terminou");
158             } catch (Exception excp) {
159                 excp.printStackTrace();
160             }
161
162         }
163     }
164
165     private static void sslInit() {
166         try {
167             // Setup truststore
168             KeyStore trustStore = null;
169             // Servidor entende jks nao bks
170             trustStore = KeyStore.getInstance("JKS");
171
172             TrustManagerFactory trustManagerFactory = null;
173             trustManagerFactory = TrustManagerFactory.getInstance(
                   TrustManagerFactory
174                         .getDefaultAlgorithm());
175             InputStream trustStoreStream = new FileInputStream("certs
                   /truststorejks");
176
177             trustStore.load(trustStoreStream, "MyPassword".
                   toCharArray());
178             trustManagerFactory.init(trustStore);
179
180             // Setup keystore
181             // Get an SSL context using the TLS protocol
```

```
182             SSLContext sslContext = SSLContext.getInstance("TLS");
183
184             // Get a key manager factory using the default algorithm
185             KeyManagerFactory kmf = KeyManagerFactory
186                     .getInstance(KeyManagerFactory.
                        getDefaultAlgorithm());
187
188             // Load the PKCS12 key chain
189             KeyStore ks = KeyStore.getInstance("PKCS12");
190             InputStream keyStoreStream = new FileInputStream("certs/
                    bob.p12");
191             ks.load(keyStoreStream, "bob".toCharArray());
192             kmf.init(ks, "bob".toCharArray());
193
194             // Initialize the SSL context
195             sslContext.init(kmf.getKeyManagers(), trustManagerFactory
                    .getTrustManagers(), null);
196
197             // Create the SSL server socket factory
198             sssf = sslContext.getServerSocketFactory();
199
200         } catch (Exception e) {
201             e.printStackTrace();
202         }
203     }
204 }
```

A.2 Códigos do Aplicativo *stress*.SSACC

A.2.1 Código da MainActivity.java

```
1  package br.com.ssacc.activity;
2
3  import android.app.Activity;
4  import android.app.AlertDialog;
5  import android.app.AlertDialog.Builder;
6  import android.content.Context;
7  import android.content.Intent;
8  import android.content.IntentFilter;
9  import android.content.SharedPreferences;
10 import android.media.AudioManager;
11 import android.media.ToneGenerator;
12 import android.os.Bundle;
13 import android.os.Handler;
14 import android.preference.PreferenceManager;
15 import android.util.Log;
16 import android.view.Menu;
17 import android.view.MenuInflater;
18 import android.view.MenuItem;
19 import android.view.View;
20 import android.view.View.OnClickListener;
21 import android.widget.Button;
22 import android.widget.RadioGroup;
23 import android.widget.TextView;
24 import android.widget.Toast;
25 import br.com.ssacc.R;
26 import br.com.ssacc.context.BatteryInformation;
27 import br.com.ssacc.preference.Preferences;
28 import br.com.ssacc.sockets.SocketSSL;
29
30 public class MainActivity extends Activity implements OnClickListener
       {
31     public static int batteryLevel = -1;
```

```
32
33      private final String typeAES = "AES";
34      private final String typeRC4 = "RC4";
35      private final String type3DES = "3DES";
36      public static String type = "AES"; // inicia com AES
37
38      private Button but;
39      private Context context;
40      private TextView batteryInfo;
41      private RadioGroup group;
42      private final Handler handler = new Handler();
43      private BatteryInformation bateria;
44
45      @Override
46      protected void onCreate(Bundle savedInstanceState) {
47          super.onCreate(savedInstanceState);
48          setContentView(R.layout.activity_main);
49
50          context = getApplicationContext();
51
52          init();
53
54          bateria = new BatteryInformation(batteryInfo);
55          this.registerReceiver(bateria, new IntentFilter(Intent.
                ACTION_BATTERY_CHANGED));
56
57      }
58
59      @Override
60      protected void onDestroy() {
61          this.unregisterReceiver(bateria);
62          super.onDestroy();
63      }
64
65      private void init() {
```

```
66          this.group = (RadioGroup) findViewById(R.id.rg);
67
68          this.but = (Button) findViewById(R.id.button1);
69          this.but.setOnClickListener(this);
70
71          batteryInfo = (TextView) findViewById(R.id.
                textViewBatteryInfo);
72      }
73
74      /** Ação do botão Enviar */
75      @Override
76      public void onClick(View arg0) {
77          try {
78              /* Obtem as constantes IP/PORTA da classe Preferences */
79              SharedPreferences preferenciasConexao = PreferenceManager
80                      .getDefaultSharedPreferences(this);
81              String urlIp = preferenciasConexao.getString("ip", "");
82              String porta = preferenciasConexao.getString("porta", "")
                    ;
83              // String file = preferenciasConexao.getString("
                    filePicker", "uhull");
84              // Log.i("PREF-RECUP", file);
85
86              int urlPorta = Integer.parseInt(porta);
87
88              /* Realiza as validações se preenchimento de IP/Porta/
                    Mensagem de Envio */
89              if ((urlIp.equals("")) || porta.equals("")) {
90                  showDialong("Por favor, insira IP/Porta");
91                  return;
92              }
93
94              switch (group.getCheckedRadioButtonId()) {
95                  case R.id.rgAES:
96                      type = typeAES;
```

```
97                      break;
98                  case R.id.rgRC4:
99                      type = typeRC4;
100                     break;
101                 case R.id.rg3DES:
102                     type = type3DES;
103             }
104
105             /* Envia parâmetros para Classe SocketSSL -
                   sendMessageToServer(msg,urlIp,urlPorta); */
106             new Thread(new SocketSSL(this, urlIp, urlPorta)).start();
107
108             // "desliga" o layout para evitar várias execuções
109             but.setText("Executando " + type);
110             but.setEnabled(false);
111             group.setEnabled(false);
112
113             Log.d("URL-MONTADA", urlIp + ":" + urlPorta);
114
115         } catch (NumberFormatException e) {
116             e.printStackTrace();
117         } catch (Exception e) {
118             e.printStackTrace();
119             gerarToast(e.getMessage());
120         }
121     }
122
123     /** Adiciona o menu a Tela da aplicação. */
124     @Override
125     public boolean onCreateOptionsMenu(Menu menu) {
126         MenuInflater menuInflater = getMenuInflater();
127         menuInflater.inflate(R.menu.lista_menu, menu);
128
129         return (true);
130     }
```

```
131
132     /** Adicão do menu da aplicação. */
133     @Override
134     public boolean onOptionsItemSelected(MenuItem item) {
135         switch (item.getItemId()) {
136             case R.id.menu_configuracao:
137                 Intent itentPreferencias = new Intent(MainActivity.
                        this, Preferences.class);
138                 startActivity(itentPreferencias);
139
140                 return (true);
141             case R.id.menu_sobre:
142                 Intent itentSobre = new Intent(MainActivity.this,
                        Sobre.class);
143                 startActivity(itentSobre);
144
145                 return (true);
146             default:
147                 break;
148         }
149
150         return (super.onOptionsItemSelected(item));
151     }
152
153     /**
154      * Mostra uma caixa de dialogo para ao usuário.
155      *
156      * @param mensagem
157      */
158     private void showDialong(CharSequence mensagem) {
159         Builder builder = new AlertDialog.Builder(MainActivity.this);
160         builder.setTitle("SSACC");
161         builder.setMessage(mensagem);
162         builder.setNeutralButton("OK", null);
163         builder.show();
```

```
164     }
165
166     private void gerarToast(CharSequence mensagem) {
167         Toast.makeText(context, mensagem, Toast.LENGTH_SHORT).show();
168         ToneGenerator toneG = new ToneGenerator(AudioManager.
                STREAM_ALARM, 100);
169         toneG.startTone(TRIM_MEMORY_RUNNING_CRITICAL, 2000);
170     }
171
172     public void executionFinished() {
173         // reability layout
174
175         handler.post(new Runnable() {
176             @Override
177             public void run() {
178                 but.setText("Enviar");
179                 but.setEnabled(true);
180                 group.setEnabled(true);
181
182                 gerarToast("Envio concluído");
183             }
184         });
185     }
186 }
```

A.2.2 Código de SocketSSL.java

```
1  package br.com.ssacc.sockets;
2
3  import java.io.File;
4  import java.io.FileInputStream;
5  import java.io.IOException;
6  import java.io.InputStream;
7  import java.io.ObjectOutputStream;
8  import java.io.OutputStream;
9  import java.io.RandomAccessFile;
10 import java.net.Socket;
11 import java.net.UnknownHostException;
12 import java.security.AccessControlException;
13 import java.security.KeyManagementException;
14 import java.security.KeyStore;
15 import java.security.KeyStoreException;
16 import java.security.NoSuchAlgorithmException;
17 import java.security.UnrecoverableKeyException;
18 import java.security.cert.CertificateException;
19 import java.util.List;
20 import java.util.regex.Matcher;
21 import java.util.regex.Pattern;
22
23 import javax.net.ssl.KeyManagerFactory;
24 import javax.net.ssl.SSLContext;
25 import javax.net.ssl.SSLSocket;
26 import javax.net.ssl.SSLSocketFactory;
27 import javax.net.ssl.TrustManagerFactory;
28
29 import android.app.ActivityManager;
30 import android.app.ActivityManager.RunningAppProcessInfo;
31 import android.app.AlertDialog;
32 import android.content.Context;
33 import android.content.SharedPreferences;
34 import android.os.Debug.MemoryInfo;
```

```
35 import android.preference.PreferenceManager;
36 import android.util.Log;
37 import android.widget.Toast;
38 import br.com.ssacc.R;
39 import br.com.ssacc.activity.MainActivity;
40 import br.com.ssacc.util.ArchiveTestsSaver;
41
42 public class SocketSSL implements Runnable {
43     private SSLContext ssl_ctx;
44     private final String ip;
45     private final int port;
46
47     private final MainActivity ctx;
48
49     public SocketSSL(MainActivity c, String _ip, int _port) {
50         ctx = c;
51         ip = _ip;
52         port = _port;
53     }
54
55     public SocketSSL(MainActivity context, String urlIp, int urlPorta
           , AlertDialog alertDialog) {
56         ctx = context;
57         ip = urlIp;
58         port = urlPorta;
59     }
60
61     @Override
62     public void run() {
63         sslInit();
64
65         ObjectOutputStream out = null;
66
67         try {
68             // sslsocket
```

```
69              SSLSocketFactory socketFactory = ssl_ctx.getSocketFactory
                    ();
70              Socket client = socketFactory.createSocket(ip, port);
71
72              ((SSLSocket) client).setEnabledCipherSuites(matcher(
                    MainActivity.type));
73              ((SSLSocket) client).startHandshake();
74
75              sendConfigToServer((SSLSocket) client);
76
77              out = new ObjectOutputStream(client.getOutputStream());
78
79              testeStress(out);
80
81              out.close(); // Fecha o stream de envio para o servidor
82              client.close();
83
84              ctx.executionFinished();
85
86          } catch (UnknownHostException e) {
87              e.printStackTrace();
88          } catch (IOException e) {
89              e.printStackTrace();
90          } catch (Exception e) {
91              e.printStackTrace();
92              // gerarToast("Erro na conexão com o servidor");
93          }
94      }
95
96      private String[] matcher(String type) {
97          Pattern pattern = Pattern.compile(type);
98          Matcher matcher;
99          boolean matchFound;
100
```

```
101         String str[] = { "SSL_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA", "
                SSL_RSA_WITH_3DES_EDE_CBC_SHA",
102                 "TLS_RSA_WITH_AES_128_CBC_SHA", "
                       SSL_RSA_WITH_RC4_128_MD5" };
103
104         int len = str.length;
105         String set[] = new String[len];
106
107         int j = 0, k = len - 1;
108         for (int i = 0; i < len; i++) {
109             // System.out.println(str[i]);
110
111             // Determine if pattern exists in input
112             matcher = pattern.matcher(str[i]);
113             matchFound = matcher.find();
114
115             if (matchFound)
116                 set[j++] = str[i];
117             else
118                 set[k--] = str[i];
119         }
120
121         return (set);
122     }
123
124     private void sendConfigToServer(SSLSocket s) throws IOException {
125         OutputStream outputStream = s.getOutputStream();
126
127         outputStream.write(MainActivity.type.getBytes());
128         outputStream.close();
129     }
130
131     private void gerarToast(String mensagem) {
132         Toast.makeText(ctx, mensagem, Toast.LENGTH_LONG).show();
133     }
```

```
134
135     private synchronized void testeStress(ObjectOutputStream out)
            throws IOException {
136         int round = 0;
137         int nivel = MainActivity.batteryLevel, emNivel = nivel;
138         long timeStart, timeFinished = 0;
139         long sumElapsedTime = 0;
140         float cpu, sumCpuUsage = 0f;
141         float memo, sumMemoUsage = 0f;
142
143         ArchiveTestsSaver saver = new ArchiveTestsSaver(MainActivity.
                type);
144
145         // StringBuffer dadosContextoLog = new StringBuffer();
146         SharedPreferences SPref = PreferenceManager.
                getDefaultSharedPreferences(ctx);
147         String uriFile = SPref.getString("filePicker", "storage/
                emulated/legacy/test2.txt");
148
149         Log.i("ARCH", uriFile);
150         File f = new File(uriFile);
151
152         Log.d("dado-batterry0", Integer.toString(nivel));
153         while (nivel == emNivel) {
154             try {
155                 FileInputStream in = new FileInputStream(f);
156                 byte[] buffer = new byte[4096];
157
158                 while (true) {
159                     int len = in.read(buffer);
160                     if (len == -1)
161                         break;
162                 }
163                 in.close();
164
```

```
165                 emNivel = MainActivity.batteryLevel;
166                 Log.d("dado-batterry0", Integer.toString(emNivel));
167 
168             } catch (IOException e) {
169                 e.printStackTrace();
170                 break;
171             }
172         }
173 
174         nivel = emNivel; // para fazer o novo loop
175 
176         Log.d("dado-batterry1", Integer.toString(nivel));
177         Log.i("dado-CPU", "" + getUsedCPU());
178         while (nivel == emNivel) {
179             timeStart = System.currentTimeMillis();
180             cpu = 0f;
181 
182             try {
183                 FileInputStream in = new FileInputStream(f);
184                 byte[] buffer = new byte[4096];
185 
186                 while (true) {
187                     int len = in.read(buffer);
188                     if (len == -1)
189                         break;
190                     out.write(buffer, 0, len);
191                 }
192                 cpu = getUsedCPU();
193                 memo = getMemo();
194                 out.flush();
195                 in.close();
196 
197                 // ATUALIZA VALORES
198                 round++;
199                 sumCpuUsage += cpu;
```

```
200                 sumMemoUsage += memo;
201                 emNivel = MainActivity.batteryLevel;
202                 sumElapsedTime += (timeFinished = System.
                        currentTimeMillis()) - timeStart;
203
204                 // Salva os dados
205                 saver.saveLine(round, timeStart, timeFinished, cpu,
                        memo);
206
207             } catch (IOException e) {
208                 e.printStackTrace();
209                 break;
210             }
211         }
212         Log.d("dado-batterry2", Integer.toString(emNivel));
213
214         saver.averageLine(--round, sumElapsedTime, sumCpuUsage);
215         saver.close(--round, sumElapsedTime);
216
217         Log.d("TRANSF", "END");
218     }
219
220     private void sslInit() {
221         try {
222             /****************** inicio ssl *********************/
223
224             // Setup truststore
225             KeyStore trustStore = null;
226             trustStore = KeyStore.getInstance("BKS"); // MainActivity
                    .type
227
228             TrustManagerFactory trustManagerFactory = null;
229             trustManagerFactory = TrustManagerFactory.getInstance(
                    TrustManagerFactory
230                         .getDefaultAlgorithm());
```

```
231             InputStream trustStoreStream = ctx.getResources().
                    openRawResource(R.raw.truststore);
232
233             trustStore.load(trustStoreStream, "MyPassword".
                    toCharArray());
234             trustManagerFactory.init(trustStore);
235
236             // Setup keystore
237             KeyStore keyStore = null;
238             keyStore = KeyStore.getInstance("BKS");
239
240             KeyManagerFactory keyManagerFactory = null;
241             keyManagerFactory = KeyManagerFactory.getInstance(
                    KeyManagerFactory.getDefaultAlgorithm());
242
243             InputStream keyStoreStream = ctx.getResources().
                    openRawResource(R.raw.client);
244             keyStore.load(keyStoreStream, "MyPassword".toCharArray())
                    ;
245             keyManagerFactory.init(keyStore, "MyPassword".toCharArray
                    ());
246
247             ssl_ctx = SSLContext.getInstance("TLS");
248             ssl_ctx.init(keyManagerFactory.getKeyManagers(),
                    trustManagerFactory.getTrustManagers(),
249                     null);
250
251         } catch (NoSuchAlgorithmException nsae) {
252             Log.d("SSL", nsae.getMessage());
253             gerarToast("Erro ao iniciar SSL");
254         } catch (KeyStoreException kse) {
255             Log.d("SSL", kse.getMessage());
256             gerarToast("Erro ao iniciar SSL");
257         } catch (IOException ioe) {
258             Log.d("SSL", ioe.getMessage());
```

```
259             gerarToast("Erro ao iniciar SSL");
260         } catch (CertificateException ce) {
261             Log.d("SSL", ce.getMessage());
262             gerarToast("Erro ao iniciar SSL");
263         } catch (KeyManagementException kme) {
264             Log.d("SSL", kme.getMessage());
265             gerarToast("Erro ao iniciar SSL");
266         } catch (AccessControlException ace) {
267             Log.d("SSL", ace.getMessage());
268             gerarToast("Erro ao iniciar SSL");
269         } catch (UnrecoverableKeyException uke) {
270             Log.d("SSL", uke.getMessage());
271             gerarToast("Erro ao iniciar SSL");
272         } catch (NullPointerException nexc) {
273             Log.d("SSL", nexc.getMessage());
274             gerarToast("Erro ao iniciar SSL");
275         } catch (Exception e) {
276             gerarToast("Erro ao iniciar SSL");
277         }
278     }
279
280     public void toast(String msg) {
281         Toast.makeText(ctx, msg, Toast.LENGTH_SHORT).show();
282     }
283
284     /* UTILITIES */
285
286     private float getUsedCPU() {
287         try {
288             RandomAccessFile reader = new RandomAccessFile("/proc/
                    stat", "r");
289             String load = reader.readLine();
290
291             String[] toks = load.split(" ");
292
```

```
293             long idle1 = Long.parseLong(toks[4]);
294             long cpu1 = Long.parseLong(toks[2]) + Long.parseLong(toks
                    [3]) + Long.parseLong(toks[5])
295                     + Long.parseLong(toks[6]) + Long.parseLong(toks
                        [7]) + Long.parseLong(toks[8]);
296
297             try {
298                 Thread.sleep(360);
299
300             } catch (Exception e) {
301             }
302
303             reader.seek(0);
304             load = reader.readLine();
305             reader.close();
306
307             toks = load.split(" ");
308
309             long idle2 = Long.parseLong(toks[4]);
310             long cpu2 = Long.parseLong(toks[2]) + Long.parseLong(toks
                    [3]) + Long.parseLong(toks[5])
311                     + Long.parseLong(toks[6]) + Long.parseLong(toks
                        [7]) + Long.parseLong(toks[8]);
312
313             return ((float) (cpu2 - cpu1) / ((cpu2 + idle2) - (cpu1 +
                    idle1)));
314
315         } catch (IOException ex) {
316             ex.printStackTrace();
317         }
318
319         return (0);
320     }
321
322     private long getUsedMemorySize() {
```

```
323         ActivityManager activityManager = (ActivityManager) ctx
324                 .getSystemService(Context.ACTIVITY_SERVICE);
325
326         List<RunningAppProcessInfo> appProcesses = activityManager.
               getRunningAppProcesses();
327
328         int[] pids = null;
329
330         MemoryInfo[] processMemoryInfo = activityManager.
               getProcessMemoryInfo(pids);
331
332         return (processMemoryInfo[0].getTotalPss());
333     }
334
335     private float getMemo() {
336         int mb = 1024 * 1024;
337
338         // Getting the runtime reference from system
339         Runtime runtime = Runtime.getRuntime();
340
341         System.out.println("##### Heap utilization statistics [MB]
               #####");
342
343         float memo = ((float) runtime.totalMemory() / mb);
344
345         return (memo);
346     }
347 }
```

A.2.3 Código de BatteryInformation.java

```
1  package br.com.ssacc.context;
2
3  import android.content.BroadcastReceiver;
4  import android.content.Context;
5  import android.content.Intent;
6  import android.content.IntentFilter;
7  import android.os.BatteryManager;
8  import android.util.Log;
9  import android.widget.TextView;
10 import br.com.ssacc.activity.MainActivity;
11
12 public class BatteryInformation extends BroadcastReceiver {
13     private final TextView batteryInfo;
14     public int estadoAtual;
15
16     public BatteryInformation(TextView _infoBatrry) {
17         batteryInfo = _infoBatrry;
18     }
19
20     @Override
21     public void onReceive(Context context, Intent intent) {
22         IntentFilter ifilter = new IntentFilter(Intent.
               ACTION_BATTERY_CHANGED);
23         Intent batteryStatus = context.registerReceiver(null, ifilter
               );
24
25         MainActivity.batteryLevel = batteryStatus.getIntExtra(
               BatteryManager.EXTRA_LEVEL, -1);
26         Log.i("BATTERY", "" + MainActivity.batteryLevel);
27
28         estadoAtual = batteryStatus.getIntExtra(BatteryManager.
               EXTRA_STATUS, -1);
29
30         batteryInfo.setText("Bateria : " + MainActivity.batteryLevel
```

```
31                      + " Estado:" + estadoAtual + "\n");
32          }
33  }
34  }
```

A.2.4 Código de ArchiveTestsSaver.java

```
1  package br.com.ssacc.util;
2
3  import java.io.File;
4  import java.io.FileNotFoundException;
5  import java.io.FileOutputStream;
6  import java.io.IOException;
7
8  import android.os.Environment;
9
10 public class ArchiveTestsSaver {
11     private final String initLine = "| %s |    %s    |   %s  | %s |
           %s  | %s |\r\n";
12     private final String modelLine = "| %5d | %d | %d | %-15d | %2.3f
           | %3.1f |\r\n";
13     private final String separator = "|------- ---------------
           --------------- ----------------  ------- -------|\r\n";
14
15     private FileOutputStream fileOS;
16
17     public ArchiveTestsSaver(String criptType) {
18         File f = new File(Environment.getExternalStorageDirectory(),
               "testeStress" + criptType
19                     + ".txt");
20
21         try {
22             fileOS = new FileOutputStream(f, true);
23
24             initLine();
25
26         } catch (FileNotFoundException e) {
27             e.printStackTrace();
28         } catch (IOException e) {
29             e.printStackTrace();
30         }
```

```
31      }
32
33      private void initLine() throws IOException {
34          String line = String.format(initLine, "ROUND", "T-START", "T-
               FINISHED", "T-ELAPSED (mls)",
35                  "CPU", "MEMO");
36
37          fileOS.write(separator.getBytes());
38          fileOS.write(line.getBytes());
39          fileOS.write(separator.getBytes());
40
41          fileOS.flush();
42      }
43
44      public void saveLine(int round, long tmStart, long tmEnd, float
           cpu, float memo)
45              throws IOException {
46          long tmElapsed = tmEnd - tmStart;
47
48          fileOS.write(String.format(modelLine, round, tmStart, tmEnd,
               tmElapsed, cpu, memo).getBytes());
49          fileOS.flush();
50      }
51
52      public void averageLine(int round, long sum, float sumCpu) throws
           IOException {
53          final String separator = "|------- ---------------
               --------------- ---------------  ------- -------|\r\n";
54          final String mediaLine = "|               TT: %-7d
                        |   MTR: %-6.3f  | MCPU: %-3.3f  |\r\n";
55
56          fileOS.write(separator.getBytes());
57          fileOS.write(String.format(mediaLine, sum, ((float) sum /
               round), (sumCpu / round))
58                  .getBytes());
```

```
59          fileOS.flush();
60          fileOS.write(separator.getBytes());
61      }
62
63      public void close(int round, long sum) throws IOException {
64          fileOS.write("\r\n".getBytes());
65
66          fileOS.close();
67      }
68  }
```

A.2.5 Código de AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="br.com.ssacc"
    android:versionCode="1"
    android:versionName="1.0"
    >

   <uses-sdk
    android:minSdkVersion="8"
        android:targetSdkVersion="18"
    />

    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission android:name="android.permission.
       ACCESS_NETWORK_STATE"/>
    <uses-permission android:name="android.permission.
       ACCESS_WIFI_STATE"/>
    <uses-permission android:name="android.permission.
       WRITE_EXTERNAL_STORAGE"/>


    <application
        android:allowBackup="true"
      android:icon="@drawable/ic_launcher"
      android:label="@string/app_name"
      android:theme="@style/AppTheme"
      >
      <activity
        android:name="br.com.ssacc.activity.MainActivity"
         android:label="@string/app_name"
         >
         <intent-filter>
            <action android:name="android.intent.action.MAIN" />
```

```
31                  <category android:name="android.intent.category.
                        LAUNCHER" />
32              </intent-filter>
33          </activity>
34
35        <activity android:name=".preference.Preferences"></activity>
36        <activity android:name=".activity.Sobre"></activity>
37      </application>
38
39  </manifest>
```

A.3 Código do Aplicativo Final SSACC

A.3.1 Código de MainActivity.java

```
1  package br.com.ssacc.activity;
2
3  import android.app.Activity;
4  import android.app.AlertDialog;
5  import android.app.AlertDialog.Builder;
6  import android.content.Context;
7  import android.content.Intent;
8  import android.content.IntentFilter;
9  import android.content.SharedPreferences;
10 import android.os.Bundle;
11 import android.os.Handler;
12 import android.preference.PreferenceManager;
13 import android.util.Log;
14 import android.view.Menu;
15 import android.view.MenuInflater;
16 import android.view.MenuItem;
17 import android.view.View;
18 import android.view.View.OnClickListener;
19 import android.widget.Button;
20 import android.widget.RadioGroup;
21 import android.widget.TextView;
22 import android.widget.Toast;
23 import br.com.ssacc.R;
24 import br.com.ssacc.context.BatteryInformation;
25 import br.com.ssacc.preference.Preferences;
26 import br.com.ssacc.sockets.SocketSSL;
27
28 public class MainActivity extends Activity implements OnClickListener
       {
29     public static int batteryLevel = -1;
30
31     public static String type = "AES"; // inicia com AES
```

```
32
33      private Button but;
34      private Context context;
35      private TextView batteryInfo;
36      private RadioGroup group;
37      private final Handler handler = new Handler();
38      private BatteryInformation bateria;
39
40      @Override
41      protected void onCreate(Bundle savedInstanceState) {
42          super.onCreate(savedInstanceState);
43          setContentView(R.layout.activity_main);
44
45          context = getApplicationContext();
46
47          init();
48
49          bateria = new BatteryInformation(batteryInfo);
50          this.registerReceiver(bateria, new IntentFilter(Intent.
                ACTION_BATTERY_CHANGED));
51
52      }
53
54      @Override
55      protected void onDestroy() {
56          this.unregisterReceiver(bateria);
57          super.onDestroy();
58      }
59
60      private void init() {
61          this.but = (Button) findViewById(R.id.button1);
62          this.but.setOnClickListener(this);
63
64          batteryInfo = (TextView) findViewById(R.id.
                textViewBatteryInfo);
```

```
65      }
66
67      /** Ação do botão Enviar */
68      @Override
69      public void onClick(View arg0) {
70          try {
71              /* Obtem as constantes IP/PORTA da classe Preferences */
72              SharedPreferences preferenciasConexao = PreferenceManager
73                      .getDefaultSharedPreferences(this);
74              String urlIp = preferenciasConexao.getString("ip", "");
75              String porta = preferenciasConexao.getString("porta", "")
                    ;
76
77
78              int urlPorta = Integer.parseInt(porta);
79
80              /* Realiza as validações se preenchimento de IP/Porta/
                    Mensagem de Envio */
81              if ((urlIp.equals("")) || porta.equals("")) {
82                  showDialong("Por favor, insira IP/Porta");
83                  return;
84              }
85
86              /* Envia parâmetros para Classe SocketSSL -
                    sendMessageToServer(msg,urlIp,urlPorta); */
87              new Thread(new SocketSSL(this, urlIp, urlPorta)).start();
88
89              // "desliga" o layout para evitar várias execuções
90              but.setText("Executando " + type);
91              but.setEnabled(false);
92              group.setEnabled(false);
93
94              Log.d("URL-MONTADA", urlIp + ":" + urlPorta);
95
96          } catch (NumberFormatException e) {
```

```
97              e.printStackTrace();
98          } catch (Exception e) {
99              e.printStackTrace();
100             gerarToast(e.getMessage());
101         }
102     }
103
104     /** Adiciona o menu a Tela da aplicação. */
105     @Override
106     public boolean onCreateOptionsMenu(Menu menu) {
107         MenuInflater menuInflater = getMenuInflater();
108         menuInflater.inflate(R.menu.lista_menu, menu);
109
110         return (true);
111     }
112
113     /** Adição do menu da aplicação. */
114     @Override
115     public boolean onOptionsItemSelected(MenuItem item) {
116         switch (item.getItemId()) {
117             case R.id.menu_configuracao:
118                 Intent itentPreferencias = new Intent(MainActivity.
                        this, Preferences.class);
119                 startActivity(itentPreferencias);
120
121                 return (true);
122             case R.id.menu_sobre:
123                 Intent itentSobre = new Intent(MainActivity.this,
                        Sobre.class);
124                 startActivity(itentSobre);
125
126                 return (true);
127             default:
128                 break;
129         }
```

```
130
131         return (super.onOptionsItemSelected(item));
132     }
133
134     /**
135      * Mostra uma caixa de diálogo para ao usuário.
136      *
137      * @param mensagem
138      */
139     private void showDialong(CharSequence mensagem) {
140         Builder builder = new AlertDialog.Builder(MainActivity.this);
141         builder.setTitle("ssacc");
142         builder.setMessage(mensagem);
143         builder.setNeutralButton("OK", null);
144         builder.show();
145     }
146
147     private void gerarToast(CharSequence mensagem) {
148         Toast.makeText(context, mensagem, Toast.LENGTH_SHORT).show();
149     }
150
151     public void executionFinished() {
152         // reability layout
153
154         handler.post(new Runnable() {
155             @Override
156             public void run() {
157                 but.setText("Enviar");
158                 but.setEnabled(true);
159                 group.setEnabled(true);
160
161                 gerarToast("Envio concluído");
162             }
163         });
164     }
```

```
165 }
```

A.3.2 Código de SocketSSL.java

```
1  package br.com.ssacc.sockets;
2
3  import android.content.SharedPreferences;
4  import android.preference.PreferenceManager;
5  import android.util.Log;
6  import android.widget.Toast;
7  import java.io.File;
8  import java.io.FileInputStream;
9  import java.io.IOException;
10 import java.io.InputStream;
11 import java.io.ObjectOutputStream;
12 import java.io.OutputStream;
13 import java.io.RandomAccessFile;
14 import java.net.Socket;
15 import java.net.UnknownHostException;
16 import java.security.AccessControlException;
17 import java.security.KeyManagementException;
18 import java.security.KeyStore;
19 import java.security.KeyStoreException;
20 import java.security.NoSuchAlgorithmException;
21 import java.security.UnrecoverableKeyException;
22 import java.security.cert.CertificateException;
23 import java.util.regex.Matcher;
24 import java.util.regex.Pattern;
25 import javax.net.ssl.KeyManagerFactory;
26 import javax.net.ssl.SSLContext;
27 import javax.net.ssl.SSLSocket;
28 import javax.net.ssl.SSLSocketFactory;
29 import javax.net.ssl.TrustManagerFactory;
30 import br.com.ssacc.R;
31 import br.com.ssacc.activity.MainActivity;
32 import br.com.ssacc.util.ChooseThePath;
33
34 public class SocketSSL implements Runnable {
```

```
35      private final String ip;
36      private final int port;
37      private final MainActivity ctx;
38      private SSLContext ssl_ctx;
39
40      public SocketSSL(MainActivity c, String _ip, int _port) {
41          ctx = c;
42          ip = _ip;
43          port = _port;
44      }
45
46      @Override
47      public void run() {
48          sslInit();
49
50          ObjectOutputStream out;
51
52          try {
53              // sslsocket
54              SSLSocketFactory socketFactory = ssl_ctx.getSocketFactory
                    ();
55              Socket client = socketFactory.createSocket(ip, port);
56
57              MainActivity.type = ChooseThePath.getEncryption(
                    MainActivity.batteryLevel);
58
59              ((SSLSocket) client).setEnabledCipherSuites(matcher(
                    MainActivity.type));
60              ((SSLSocket) client).startHandshake();
61
62              sendConfigToServer((SSLSocket) client);
63
64              out = new ObjectOutputStream(client.getOutputStream());
65
66              testeStress(out);
```

```
67
68              out.close(); // Fecha o stream de envio para o servidor
69              client.close();
70
71              ctx.executionFinished();
72
73          } catch (UnknownHostException e) {
74              e.printStackTrace();
75          } catch (IOException e) {
76              e.printStackTrace();
77          } catch (Exception e) {
78              e.printStackTrace();
79          }
80      }
81
82      private String[] matcher(String type) {
83          Pattern pattern = Pattern.compile(type);
84          Matcher matcher;
85          boolean matchFound;
86
87          String str[] = {"SSL_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA", "
                SSL_RSA_WITH_3DES_EDE_CBC_SHA", "
                TLS_RSA_WITH_AES_128_CBC_SHA", "SSL_RSA_WITH_RC4_128_MD5"
                };
88
89          int len = str.length;
90          String set[] = new String[len];
91
92          int j = 0, k = len - 1;
93          for (String aStr : str) {
94
95              // Determine if pattern exists in input
96              matcher = pattern.matcher(aStr);
97              matchFound = matcher.find();
98
```

```
99              if (matchFound) set[j++] = aStr;
100             else set[k--] = aStr;
101         }
102
103         return (set);
104     }
105
106     private void sendConfigToServer(SSLSocket s) throws IOException {
107         OutputStream outputStream = s.getOutputStream();
108
109         outputStream.write(MainActivity.type.getBytes());
110         outputStream.close();
111     }
112
113     private void gerarToast(String mensagem) {
114         Toast.makeText(ctx, mensagem, Toast.LENGTH_LONG).show();
115     }
116
117     private synchronized void testeStress(ObjectOutputStream out)
            throws IOException {
118         int nivel = MainActivity.batteryLevel, emNivel = nivel;
119
120         // StringBuffer dadosContextoLog = new StringBuffer();
121         SharedPreferences SPref = PreferenceManager.
                getDefaultSharedPreferences(ctx);
122         String uriFile = SPref.getString("filePicker", "storage/
                emulated/legacy/test2.txt");
123
124         Log.i("ARCH", uriFile);
125         File f = new File(uriFile);
126
127         Log.d("dado-battery0", Integer.toString(nivel));
128         Log.i("dado-CPU", "" + getUsedCPU());
129         while (nivel == emNivel) {
130
```

```
131             try {
132                 FileInputStream in = new FileInputStream(f);
133                 byte[] buffer = new byte[4096];
134 
135                 while (true) {
136                     int len = in.read(buffer);
137                     if (len == -1) break;
138                     out.write(buffer, 0, len);
139                 }
140                 out.flush();
141                 in.close();
142 
143             } catch (IOException e) {
144                 e.printStackTrace();
145                 break;
146             }
147         }
148         Log.d("dado-battery1", Integer.toString(emNivel));
149 
150         Log.d("TRANSFER", "END");
151     }
152 
153     private void sslInit() {
154         try {
155             /****************** inicio ssl *********************/
156 
157             // Setup truststore
158             KeyStore trustStore = KeyStore.getInstance("BKS");
159 
160             TrustManagerFactory trustManagerFactory;
161             trustManagerFactory = TrustManagerFactory.getInstance(
                    TrustManagerFactory.getDefaultAlgorithm());
162             InputStream trustStoreStream = ctx.getResources().
                    openRawResource(R.raw.truststore);
163 
```

```
164             trustStore.load(trustStoreStream, "MyPassword".
                    toCharArray());
165             trustManagerFactory.init(trustStore);
166
167             // Setup keystore
168             KeyStore keyStore = KeyStore.getInstance("BKS");
169
170             KeyManagerFactory keyManagerFactory;
171             keyManagerFactory = KeyManagerFactory.getInstance(
                    KeyManagerFactory.getDefaultAlgorithm());
172
173             InputStream keyStoreStream = ctx.getResources().
                    openRawResource(R.raw.client);
174             keyStore.load(keyStoreStream, "MyPassword".toCharArray())
                    ;
175             keyManagerFactory.init(keyStore, "MyPassword".toCharArray
                    ());
176
177             ssl_ctx = SSLContext.getInstance("TLS");
178             ssl_ctx.init(keyManagerFactory.getKeyManagers(),
                    trustManagerFactory.getTrustManagers(), null);
179
180         } catch (NoSuchAlgorithmException nsae) {
181             Log.d("SSL", nsae.getMessage());
182             gerarToast("Erro ao iniciar SSL");
183         } catch (KeyStoreException kse) {
184             Log.d("SSL", kse.getMessage());
185             gerarToast("Erro ao iniciar SSL");
186         } catch (IOException ioe) {
187             Log.d("SSL", ioe.getMessage());
188             gerarToast("Erro ao iniciar SSL");
189         } catch (CertificateException ce) {
190             Log.d("SSL", ce.getMessage());
191             gerarToast("Erro ao iniciar SSL");
192         } catch (KeyManagementException kme) {
```

```
193                 Log.d("SSL", kme.getMessage());
194                 gerarToast("Erro ao iniciar SSL");
195             } catch (AccessControlException ace) {
196                 Log.d("SSL", ace.getMessage());
197                 gerarToast("Erro ao iniciar SSL");
198             } catch (UnrecoverableKeyException uke) {
199                 Log.d("SSL", uke.getMessage());
200                 gerarToast("Erro ao iniciar SSL");
201             } catch (NullPointerException nexc) {
202                 Log.d("SSL", nexc.getMessage());
203                 gerarToast("Erro ao iniciar SSL");
204             } catch (Exception e) {
205                 gerarToast("Erro ao iniciar SSL");
206             }
207         }
208
209 }
```

A.3.3 Código de BatteryInformation.java

```
1  package br.com.ssacc.context;
2
3  import android.content.BroadcastReceiver;
4  import android.content.Context;
5  import android.content.Intent;
6  import android.content.IntentFilter;
7  import android.os.BatteryManager;
8  import android.util.Log;
9  import android.widget.TextView;
10 import br.com.ssacc.activity.MainActivity;
11
12 public class BatteryInformation extends BroadcastReceiver {
13     private final TextView batteryInfo;
14     public int estadoAtual;
15
16     public BatteryInformation(TextView _infoBatrry) {
17         batteryInfo = _infoBatrry;
18     }
19
20     @Override
21     public void onReceive(Context context, Intent intent) {
22         IntentFilter ifilter = new IntentFilter(Intent.
               ACTION_BATTERY_CHANGED);
23         Intent batteryStatus = context.registerReceiver(null, ifilter
               );
24
25         MainActivity.batteryLevel = batteryStatus.getIntExtra(
               BatteryManager.EXTRA_LEVEL, -1);
26         Log.i("BATTERY", "" + MainActivity.batteryLevel);
27
28         estadoAtual = batteryStatus.getIntExtra(BatteryManager.
               EXTRA_STATUS, -1);
29
30         batteryInfo.setText("Bateria : " + MainActivity.batteryLevel
```

```
31                      + " Estado:" + estadoAtual + "\n");
32          }
33  }
```

A.3.4 Código de ChooseThePath.java

```
1  package br.com.ssacc.util;
2
3  /**
4   *
5   */
6  public class ChooseThePath {
7      private static final String typeAES = "AES";
8      private static final String typeRC4 = "RC4";
9      private static final String type3DES = "3DES";
10
11     public static String getEncryption(int lvl) {
12         if (lvl < 33)
13             return (typeRC4);
14         else if (lvl >= 33 && lvl <= 66)
15             return (typeAES);
16         return (type3DES);
17     }
18 }
```

A.3.5 Código de AndroidManifest.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3      package="br.com.ssacc"
4      android:versionCode="1"
5      android:versionName="1.0"
6      >
7
8     <uses-sdk
9      android:minSdkVersion="8"
10         android:targetSdkVersion="18"
11     />
12
13     <uses-permission android:name="android.permission.INTERNET"/>
14     <uses-permission android:name="android.permission.
          ACCESS_NETWORK_STATE"/>
15     <uses-permission android:name="android.permission.
          ACCESS_WIFI_STATE"/>
16
17
18     <application
19         android:allowBackup="true"
20       android:icon="@drawable/ic_launcher"
21       android:label="@string/app_name"
22       android:theme="@style/AppTheme"
23       >
24       <activity
25         android:name="br.com.ssacc.activity.MainActivity"
26          android:label="@string/app_name"
27          >
28          <intent-filter>
29             <action android:name="android.intent.action.MAIN" />
30                 <category android:name="android.intent.category.
                      LAUNCHER" />
31             </intent-filter>
```

```
32          </activity>
33
34        <activity android:name=".preference.Preferences"></activity>
35        <activity android:name=".activity.Sobre"></activity>
36      </application>
37
38  </manifest>
```