UNIVERSIDADE FEDERAL DO CEARÁ
CENTRO DE CIÊNCIAS
DEPARTAMENTO DE COMPUTAÇÃO
MESTRADO E DOUTORADO EM CIÊNCIA DA COMPUTAÇÃO

ALEXANDRE CORREIA CIRQUEIRA

UM MECANISMO DE SEGURANÇA COM ADAPTAÇÃO DINÂMICA PARA DISPOSITIVOS MÓVEIS

FORTALEZA
2011

ALEXANDRE CORREIA CIRQUEIRA

UM MECANISMO DE SEGURANÇA COM ADAPTAÇÃO DINÂMICA PARA DISPOSITIVOS MÓVEIS

Dissertação submetida à Coordenação do Curso de Pós-Graduação em Ciência da Computação, da Universidade Federal do Ceará, como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

Orientadora: Profa. Dra. Rossana Maria de Castro Andrade.
Co-orientador: Prof. Dr. Miguel Franklin de Castro

FORTALEZA
2011

ALEXANDRE CORREIA CIRQUEIRA

UM MECANISMO DE SEGURANÇA COM ADAPTAÇÃO DINÂMICA PARA DISPOSITIVOS MÓVEIS

Dissertação submetida à Coordenação do Curso de Pós-Graduação em Ciência da Computação, da Universidade Federal do Ceará, como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

Aprovada em 07 / 10 / 2011.

BANCA EXAMINADORA

Profa. Dra. Rossana Maria de Castro Andrade (Orientadora)
Universidade Federal do Ceará (UFC)

Prof. Dr. Miguel Franklin de Castro (Co-orientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. José Neuman de Souza
Universidade Federal do Ceará (UFC)

Profa. Dra. Judith Kelner
Universidade Federal de Pernambuco (UFPE)

Prof. Dr. José de Ribamar Martins Bringel Filho
Université de Grenoble I

Dedico esta dissertação à minha família e à minha querida esposa, fontes de inspiração e de apoio em minha jornada em busca do saber.

AGRADECIMENTOS

Agradeço a Deus, por me proporcionar a oportunidade, o entusiasmo, a força, a renovação e a paciência, tão necessárias para esta jornada.

Aos professores, pelo tempo concedido e pelas valiosas contribuições ao trabalho.

Aos colegas da turma de mestrado e do GREat (Grupo de Redes, Engenharia de Software e Sistemas), pelas reflexões, críticas e sugestões recebidas.

A todos aqueles que de alguma forma contribuíram com o meu processo de aprendizado.

“Tempo virá em que uma palavra que cair do bico da pena, daí à uma hora, correrá o universo por uma rede imensa... falando por milhões de bocas, reproduzindo-se infinitamente como as folhas de uma grande árvore”.

José de Alencar, 1854

RESUMO

A crescente utilização de dispositivos móveis, redes sem fio e aplicações móveis evidencia a importância da garantia de segurança da informação. Esta preocupação surge devido aos riscos envolvidos no tráfego de informações sensíveis por meio sem fio, uma vez que o meio não limita os riscos de ataques, tal como nas redes convencionais. Adicionalmente, a tendência no uso de práticas sustentáveis defendidas pela Computação Verde impõe a necessidade de concepção de aplicações flexíveis que busquem a redução do consumo de recursos, como o de energia. Assim, mecanismos para o provimento de confidencialidade de informações que trafegam por meio sem fio devem considerar a alocação eficiente de recursos computacionais. Esta é uma questão chave a ser considerada no momento da concepção de aplicações móveis seguras. Portanto, os mecanismos de proteção devem balancear o nível de segurança requerido de acordo com o consumo de recursos alocados para provê-lo. O emprego de informações que caracterizam a situação corrente (contexto) pode auxiliar nessa tarefa. Assim, a utilização de proteção adequada aos requisitos de segurança das aplicações e combinada com o contexto pode identificar situações nas quais será necessário aumentar ou diminuir o nível de segurança, de forma a diminuir o consumo de recursos do dispositivo. Esse trabalho propõe, portanto, um Mecanismo de Segurança com Adaptação Dinâmica (MeSAD), com foco na confidencialidade, capaz de adaptar o nível de segurança de acordo com o contexto e reduzir o consumo de recursos dos dispositivos móveis. O objetivo principal consiste em encontrar o ponto de equilíbrio no *tradeoff* entre nível de segurança e consumo de recursos. A fim de atingir este objetivo, este trabalho apresenta também uma ferramenta de suporte à utilização do MeSAD durante o desenvolvimento de aplicações móveis, além de possibilitar a realização de avaliações sobre o desempenho dos algoritmos criptográficos que são utilizados nos diferentes dispositivos.

Palavras chaves – Confidencialidade, Adaptação Dinâmica, Dispositivos Móveis, Computação Verde.

ABSTRACT

The increasing use of mobile devices, wireless networks and mobile applications highlights the importance of ensuring information security. This concern arises because of the risks involved in traffic sensitive information via wireless, since it does not limit the risk of attacks, as in conventional networks. Additionally, the trend in the use of sustainable practices advocated by the Green Computing imposes the need for designing flexible applications that seek to reduce consumption of resources such as energy. Thus, mechanisms for providing confidentiality of information passing over the wireless medium should consider the efficient allocation of computing resources. This is a key issue to be considered when designing secure mobile applications. Therefore, the protection mechanisms should balance the security level required in accordance with the consumption of resources allocated to provide it. The use of information that characterizes the current situation (context) can assist in this task. Thus, the use of appropriate protective security requirements of applications and combined with the context can identify situations where you need to raise or lower the security level in order to reduce the resource consumption of the device. This work proposes a Security Mechanism Dynamic Adaptation (MeSAD), focusing on confidentiality, able to adapt the level of security according to the context and reduce the resource consumption of mobile devices. The main objective is to find the balance point in the *tradeoff* between the level of security and resource consumption. In order to achieve this goal, this paper presents a tool to support the use of MeSAD during the development of mobile applications, and enable the assessments on the performance of cryptographic algorithms that are used in different devices.

Key words – Confidentiality, Dynamic Adaptation, Mobile Devices, Green Computing.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Esta dissertação apresenta o MeSAD, que é um mecanismo de segurança com adaptação dinâmica para dispositivos móveis, e a 4MeSAD, uma ferramenta de suporte à utilização do MeSAD na criação de aplicações. Esta última também faz análises de consumo de recursos computacionais necessários para o uso de algoritmos criptográficos em dispositivos móveis.

Este capítulo apresenta uma visão geral do trabalho. A Seção 1.1 contextualiza e caracteriza o problema. Na Seção 1.2 está exposta a motivação para o desenvolvimento desta dissertação. O objetivo e as contribuições deste trabalho estão descritos na Seção 1.3, e, por fim, a Seção 1.4 apresenta a estrutura na qual está organizada este documento.

1.1 Contextualização e Caracterização do Problema

Os dispositivos que operam a tecnologia móvel e sem fio estão cada vez menores e capazes de executar diversas tarefas por meio de aplicações, que são disponibilizadas para uso neste tipo de equipamento, aqui chamado de dispositivo móvel (DM). A mobilidade proporcionada pela computação móvel é a grande incentivadora da popularização do uso dos DMs e de suas aplicações que, cada vez mais, trafegam informações nos mais variados ambientes. Contudo, desafios também foram introduzidos pela computação móvel no cotidiano dos usuários deste tipo de tecnologia. Por exemplo, a segurança adequada dos dados que estão sendo transmitidos no meio sem fio é um destes desafios, principalmente por ter que considerar a limitação do poder computacional dos DMs para sua implementação.

Por um lado, como o DM utiliza o espaço aberto como meio de propagação da informação, esses dados trafegados podem ser captados por outro dispositivo receptor preparado para este fim, o que justifica a preocupação quanto à confidencialidade desta informação [BRINGEL, 2004]. Portanto, é necessária a adoção de um tratamento eficaz de segurança na criação das aplicações para DMs na tentativa de combater possíveis interceptações ou ataques a esses dados.

Por outro lado, a utilização de mecanismos de segurança em aplicações demanda a alocação de recursos computacionais, e quanto maior o nível de segurança requerido por um determinado serviço ou aplicação, maior será o nível de consumo de recursos do dispositivo para desempenhar tal tarefa [HAMAD et al., 2009]. Tendo em vista que DMs possuem limitações de recursos, é primordial que este fator seja considerado durante o desenvolvimento de suas aplicações.

Sendo assim, a utilização de mecanismos de segurança que garantam a confidencialidade das informações transmitidas, assim como a alocação eficiente dos recursos computacionais, são questões que merecem atenção e que precisam ser consideradas no momento da criação de aplicações para DM.

No que diz respeito ao uso eficiente de recursos, é importante mencionar o termo computação verde (*Green Computing*), que visa diminuir a saturação no uso de recursos, priorizando a sustentabilidade. A *Green Computing* é o estudo e a prática da utilização eficiente dos recursos computacionais, primando pela redução do uso de materiais maléficos à natureza, maximizando a eficiência energética durante a vida útil do produto, promovendo a reciclagem e a biodegradabilidade do descarte desses produtos e evitando o aumento dos resíduos de sua fabricação [WILBANKS, 2008]. No que se refere ao software, a Computação Verde inspira muitas possibilidades, desde a substituição de algoritmos pouco eficientes, até a utilização de sistemas operacionais que agregam a utilização eficiente dos recursos computacionais com esta visão verde [STEPHEN, 2009].

Diante deste cenário, um grande desafio a ser solucionado pela computação móvel consiste na oferta de mecanismos de segurança para as aplicações, sem comprometer a qualidade na execução e utilizando os recursos de maneira eficiente.

Pesquisas sobre novas técnicas para lidar com o *tradeoff* entre uso de mecanismos de segurança e alto consumo de recursos computacionais proporcionado pela implementação da segurança são incentivadas e destacadas como necessárias e de interesse da computação [MALIKI, 2010] [TADDEO, 2010].

Considerando a construção de aplicações com a adoção de segurança da maneira tradicional (em que um único algoritmo criptográfico é usado para oferta de confidencialidade durante o tempo de vida da aplicação), um fator geralmente utilizado para a escolha deste algoritmo é o desempenho. Em geral, o desempenho é desproporcional ao nível de confidencialidade proporcionada à aplicação, uma vez que o algoritmo utilizado pode não

oferecer o nível de segurança adequado ao contexto de uso, o qual pode mudar constantemente.

Para exemplificar, pode-se imaginar a seguinte situação: uma aplicação hospedada em um servidor que está sendo acessada por um DM através de um ponto de acesso (*Access Point* – AP). O AP pertence a uma rede pública, sem controle de senhas, localizada em um espaço onde transitam vários usuários com perfis diferentes e que está sujeita a qualquer tipo de acesso ou ataque (como uma rede aberta em um aeroporto). Neste cenário, mostrado na Figura 1, o usuário 1 necessita utilizar a aplicação. Entretanto, por se tratar de uma rede pública, o usuário tem receio de que seus dados sejam capturados por alguma entidade maliciosa, devido ao baixo grau de confiança que ele deposita no ambiente. Neste caso, a aplicação deve oferecer um nível de segurança adequado, reforçando a confidencialidade das informações a serem transmitidas. Caso contrário, uma entidade maliciosa, com os recursos computacionais apropriados, pode capturar a informação e decodificá-la através de técnicas como força bruta e exploração de falhas típicas de cada algoritmo de segurança.

Figura 1 – Cenários de transmissão de dados entre uma aplicação cliente-servidor e um DM

A Figura 1 mostra ainda o usuário 2 que acessa a aplicação através de um DM conectado a um AP localizado em sua fazenda, de uma rede privada que oferece autenticação, gerenciamento de chaves de criptografia e segurança conhecida por este usuário. Este último ambiente proporciona ao usuário 2 uma sensação de segurança mais elevada do que a do usuário 1. Portanto, não haveria necessidade em utilizar um algoritmo de criptografia de

complexidade elevada para garantir a confidencialidade da informação. Como normalmente as aplicações utilizam um algoritmo estático para o provimento de confidencialidade, não considerando os diferentes requisitos de cada ambiente (i.e. rede de acesso), a confidencialidade provida pode ser inadequada (insuficiente na proteção ou alocando mais recursos do que o necessário).

Note que a Figura 1 apresenta a entidade maliciosa B, que não se localiza no ambiente de cobertura da rede sem fio. A ameaça pode se encontrar além do tráfego dos dados pela rede sem fio. Apesar dos riscos de interceptação das informações durante a propagação através de uma rede aberta serem maiores que a probabilidade de um ataque realizado pela entidade B, em muitos casos esse último precisa ser considerado a fim de aumentar o nível de segurança. Por esse motivo, esse trabalho defende a adoção do mecanismo de segurança na camada de aplicação, estendendo a proteção das informações para as camadas inferiores. Além disso, quando se considera a utilização da confidencialidade na camada de aplicação é possível oferecer um tratamento personalizado para a aplicação e de acordo com seus requisitos de segurança.

Estes cenários ilustram situações diferentes com níveis de risco distintos. Devido ao baixo nível de segurança da primeira situação, o tratamento de confidencialidade deveria ser mais forte. Porém no segundo cenário, o usuário deposita mais confiança na rede de acesso, visto que a fazenda é considerada mais segura do que o ambiente do aeroporto. Neste caso, não haveria a necessidade de aumentar o nível de confidencialidade. Nesse segundo cenário, a aplicação poderia utilizar um algoritmo de criptografia menos robusto e de menor índice de consumo de recursos do aparelho, enquanto que no primeiro caso, um algoritmo mais robusto deveria ser alocado para a garantia de confidencialidade das informações.

1.2 Motivação

Diante do que foi apresentado na Seção 1.1, percebe-se que os mecanismos de segurança utilizados em aplicações para DMs não se preocupam em reduzir o consumo de recursos além de não atenderem dinamicamente e pró-ativamente aos requisitos de segurança face aos riscos que são dependentes de contexto [CHEN et al., 2010]. Mecanismos dinâmicos e evolutivos devem acompanhar a evolução constante dos DMs, a mudança de contextos e o tráfego de informações proporcionadas pela popularização das aplicações móveis [ELKODARY, 2007].

Ao avaliar o consumo de recursos computacionais percebe-se que quanto mais alto for o nível de segurança requerido pela aplicação maior será o nível de consumo de recursos [HAMAD, 2009]. Portanto, iniciativas que considerem os pilares da *Green Computing* para o uso eficiente dos recursos são necessárias, embora essa iniciativa não seja colocada em prática durante a criação de aplicações móveis que adotam segurança [CHEN et al., 2010].

Nesse contexto, um importante aspecto relacionado à segurança em computação móvel consiste no *tradeoff* entre o provimento de confidencialidade através de algoritmos criptográficos e a redução do consumo de recursos computacionais [MALIKI, 2010] [TADDEO, 2010].

Por exemplo, ao se verificar que em algumas situações uma aplicação com requisitos de segurança não exige o nível máximo de confidencialidade, percebe-se a importância em adaptar o mecanismo de criptografia utilizado, de acordo com os requisitos da aplicação. A adaptação dinâmica proporcionará o emprego adequado de algoritmo para provimento da confidencialidade e também a gestão do consumo de recursos, uma vez que a mobilidade dos DMs impõe requisitos de segurança diferentes, de acordo com as mudanças de contextos. As informações contextuais podem ser exploradas a fim de permitir a adaptação dinâmica dos mecanismos de confidencialidade embutidos em uma aplicação móvel. Assim, durante a execução desta aplicação em um ambiente considerado inseguro o mecanismo de adaptação poderia utilizar um algoritmo de criptografia mais complexo aumentando a segurança. Da mesma forma, ao selecionar um algoritmo com menor complexidade o mecanismo com adaptação dinâmica proporcionará a redução do consumo de recursos, poupando o hardware da execução de processos mais complexos. Esse uso eficiente do hardware coloca o trabalho em sintonia com as práticas da computação verde.

Em síntese, para o provimento de segurança em aplicações para DMs convencionalmente é utilizado um único algoritmo de criptografia durante o tempo de vida da aplicação. Quanto maior for a complexidade do algoritmo maior será o consumo de recursos do DM. Por outro lado, quanto menor for a complexidade do algoritmo, maior será a vulnerabilidade de segurança em determinados ambientes. Assim, a principal motivação deste trabalho consiste na necessidade de prover mecanismos de confidencialidade de forma adaptada ao ambiente atual em que o DM se encontra. Às vezes será necessário aplicar algoritmos de criptografia mais robustos, enquanto que em outras ocasiões será possível a redução do consumo de recursos através do uso de criptografia de complexidade menor, em ambientes que apresentam menores riscos. O desenvolvimento de uma ferramenta para os

engenheiros de software e programadores de aplicações de DMs que facilite a adoção desta proposta é outra motivação.

1.3 Objetivos e Contribuições

Com a adoção da adaptação dinâmica de algoritmos criptográficos de acordo com o contexto de uso da aplicação, aplicações que originalmente necessitariam de um nível elevado de segurança, mas que acabam sendo construídas com um nível inferior, e vice-versa, poderiam oferecer segurança de maneira mais adequada.

Nesse contexto, o principal objetivo do trabalho é propor um mecanismo de adaptação dinâmico em tempo de execução para o tratamento da confidencialidade na comunicação entre aplicações móveis, através da utilização de informações contextuais.

Visando atingir este objetivo, identifica-se a necessidade de avaliar os algoritmos criptográficos quanto ao desempenho e consumo de recursos em DMs, além de configurar seus níveis de proteção de acordo com suas características. Com este intuito, é proposta uma ferramenta que possibilite a coleta e a análise de desempenho e consumo de recursos. O intuito da avaliação dos algoritmos é a geração de informações que caracterizem cada um deles de acordo com seu comportamento e que sirvam de parâmetro para identificar aqueles que atendem as necessidades da aplicação.

Portanto, a principal contribuição deste trabalho de dissertação consiste na adaptação do nível adequado de confidencialidade de aplicações de DMs de forma adaptada ao contexto. As informações contextuais podem caracterizar o ambiente, o DM quanto ao nível de consumo de bateria ou quanto ao desempenho. A ferramenta proposta permite auxiliar engenheiros de *software* no desenvolvimento de aplicações seguras. Esta ferramenta será capaz de fornecer, aos engenheiros de software, condições para eles estabelecerem configurações de segurança de suas aplicações, possibilitando a criação das entradas necessárias para a adaptação da confidencialidade em tempo de execução.

Além disso, este trabalho oferece uma opção robusta para que os engenheiros de software leigos em segurança ou que ignorem este requisito na criação de suas aplicações, seja por razões de diminuição de desempenho ou pela complexidade na sua implementação,

passem a utilizar mecanismos de proteção. Adicionalmente, o trabalho oferece flexibilidade para sua adoção por parte de engenheiros que possuem conhecimentos aprofundados em segurança, pois eles podem personalizar as necessidades de criptografia de suas aplicações.

1.4 Estrutura do documento

Além deste capítulo introdutório, esta dissertação está dividida em seis capítulos descritos a seguir:

- **Capítulo 2** – apresenta definições e abordagens em adaptação de software baseada em contexto, apresentando conceitos gerais e, em seguida, focando no domínio de segurança considerado nesta dissertação.
- **Capítulo 3** – contextualiza segurança adaptativa com considerações sobre confidencialidade e uso eficiente de recursos. Este capítulo também apresenta os trabalhos relacionados a esta dissertação.
- **Capítulo 4** – apresenta a principal contribuição desta dissertação, que é o mecanismo proposto, descrevendo sua arquitetura, seu funcionamento e sua construção.
- **Capítulo 5** – apresenta a segunda contribuição do trabalho, que é a ferramenta de apoio para utilização do mecanismo proposto, descrevendo seu funcionamento e os serviços disponíveis para auxiliar o engenheiro de software na criação das aplicações seguras.
- **Capítulo 6** – descreve a avaliação do mecanismo proposto e os resultados obtidos na análise de seu funcionamento.
- **Capítulo 7** – apresenta as considerações finais e os trabalhos futuros.

2 ADAPTAÇÃO DE SOFTWARE BASEADA EM CONTEXTO

Este capítulo apresenta uma discussão sobre adaptação de *software* baseada em contexto. A Seção 2.1 traz considerações sobre conceitos de contexto. A Seção 2.2 introduz a adaptação de *software* e esclarece como este conceito é tratado, com o foco voltado para aplicações que estão preocupadas com o uso de segurança. Na Seção 2.3 são tratados conceitos sobre componentes e seus benefícios, assim como sua relação com a adaptação de *software*. Por fim, a Seção 2.4 apresenta uma conclusão sobre o assunto tratado neste capítulo com algumas considerações.

2.1 Contexto

Bazire et al. (2005), em seu trabalho sobre o entendimento do que é contexto, coletou 150 definições deste termo em diferentes domínios da ciência. Estas definições foram contextualizadas de acordo com cada domínio e disciplina, e um detalhamento foi realizado a fim de apontar algumas compreensões problemáticas para o entendimento do que é o contexto. A autora [Bazire, 2005] ressalta que o contexto pode ser avaliado através de seis componentes essenciais: restrição, influência, comportamento, natureza, estrutura e sistema. E conclui que não há consenso sobre algumas questões de caracterização do contexto, tais como: se ele é interno ou externo, se é um conjunto de informações ou de processos, se é estático ou é dinâmico, se é um conjunto de fenômenos ou uma rede organizada. Para a autora, o significado mais aceito é aquele que define contexto como um conjunto de circunstâncias que moldam um evento ou um objeto. Todavia, ela declara que a definição de contexto depende do campo de conhecimento ao qual ele pertence, e que alguns fatores são determinantes para esta definição, tais como a entidade interessada no contexto, o seu foco de atenção, a sua situação, o seu ambiente e, eventualmente, um observador.

Na área da computação, especificamente no domínio de segurança que este trabalho se situa, o contexto é explorado para criar ou adaptar sistemas de informação de acordo com as necessidades do usuário ou do dispositivo, ou ainda de ambos [DEY, 2001]. Esta adaptação pode ser mais acertada quando são extraídas e utilizadas as informações contextuais relevantes aos sistemas, especialmente quando se tratar de aplicações de DMs,

que são o foco deste trabalho. Essas informações contextuais podem ser a localização do usuário, o tempo, os recursos da rede em utilização, as características do dispositivo em uso, entre outras possibilidades.

No âmbito da computação e do desenvolvimento de *software*, o conceito de contexto largamente aceito pela comunidade científica, o define como qualquer informação que pode ser utilizada para caracterizar uma pessoa, lugar ou objeto, e que seja relevante na interação do usuário com a aplicação [DEY, 2001].

Para facilitar a compreensão do que vem a ser contexto, pode-se imaginar uma aplicação que criptografa informações antes de enviá-las por uma rede. Em determinado instante a aplicação é utilizada em um local de acesso restrito com uma rede de acesso controlado e que oferece alta segurança no tráfego das informações, como a rede de uma fazenda citada no Capítulo 1. Esta aplicação pode identificar que neste ambiente a proteção por meio da criptografia é desnecessária e, então, desativá-la, a fim de reduzir consumo de bateria decorrente da execução do mecanismo de proteção. Em outro instante, o usuário que utiliza a aplicação se locomove para um aeroporto e perde a conexão com a rede que estava utilizando. A aplicação, então, identifica a disponibilidade apenas da rede da operadora de telefonia móvel (e.g. rede 3G), a qual passa a ser utilizada para o tráfego das informações. Neste caso, a aplicação ativa o mecanismo de proteção de informações. Neste momento, a aplicação pode identificar que a rede da operadora possui segurança e utiliza um algoritmo criptográfico de menor nível de complexidade para garantir a confidencialidade fim-a-fim. Em um terceiro momento o usuário está no aeroporto onde existem redes sem fio de acesso público (sem controle de usuários ou senhas de acesso). Neste instante a aplicação identifica a rede disponível e passa a utilizá-la, para evitar o custo monetário proporcionado pelo uso da rede 3G da operadora móvel. Entretanto, devido ao maior risco de exposição das informações na rede detectada, a aplicação adapta novamente o mecanismo de segurança, visando oferecer uma proteção compatível, passando a utilizar um algoritmo de criptografia com alta complexidade. Nota-se no exemplo, que levou-se em conta a relação entre a aplicação e as situações nas quais ela foi utilizada para adaptar o mecanismo de segurança de acordo com as informações de cada local percorrido. Esta relação é que caracteriza o contexto.

Entre as tentativas de pesquisadores em ciência da computação para esclarecer o que vem a ser contexto, pode-se citar o trabalho de Schilit [SCHILIT et al., 1994]. Os autores elencam os três principais aspectos para esta compreensão, conduzindo o entendimento do significado de contexto por estas frases intuitivas: “onde você está”, “quem está com você” e “quais recursos estão próximos”. Segundo Schilit et al. (1994), existem vários tipos de

classificação de contexto, entre os quais três são básicos: o contexto computacional, o contexto do usuário e o contexto físico.

O contexto computacional é caracterizado pelas informações de configuração do dispositivo computacional, tais como: o poder de processamento, a capacidade de memória, os custos da comunicação, a conectividade do dispositivo à rede, a largura de banda e os recursos ou serviços disponíveis para uso. Já a caracterização do contexto do usuário constitui-se de informações próprias deste, como dados que definem seu perfil, identificação de outras pessoas no seu entorno, as tarefas que ele está executando e suas características espaciais ou sua localização. Finalmente, o contexto físico é caracterizado pelo ambiente físico no qual o dispositivo, o usuário ou ambos estão localizados. Exemplificando o tipo de informação que caracteriza este tipo de contexto, têm-se a luminosidade do ambiente e seus níveis, os sons, a temperatura, entre outras.

O trabalho de Chen [CHEN et al, 2000] adiciona outras classificações de contexto, essenciais ao domínio da computação móvel. Os autores discutem sobre a importância de levar em consideração mais dois tipos de contexto: o contexto temporal e o contexto histórico. O contexto temporal se compõe pelas informações de tempo como data, hora, mês ou ano, as quais são importantes para aplicações que necessitam realizar adaptação, como aquelas em que o tipo de apresentação diurna difere da noturna. Por fim, o contexto histórico que se caracteriza pelo gerenciamento de registros ao longo de um período de tempo definido. Estas informações coletadas possibilitam, por exemplo, a realização de inferências sobre as necessidades do usuário, podendo ser fonte de tomadas de decisões baseadas no histórico de decisões precedentes.

Percebe-se uma variedade de definições de contexto, levando-se em conta necessidades de uso e informações categorizadas em domínios específicos, tais como computação móvel. Portanto, a definição de contexto utilizada por este trabalho será a definida por Dey (2001), uma vez que engloba os usuários, as aplicações, os DMs e as relações entre eles, além dos ambientes possíveis de serem acessados.

O contexto pode ser ainda categorizado de duas maneiras, de acordo com o tipo de suas informações: uma maneira estática e outra dinâmica [ROCHA, 2007 apud Henricksen 2002]. O modo estático é composto por informações que não variam no decorrer do tempo. Informações como as que são utilizadas em tempo de construção e definidas pelos engenheiros de *software,* de acordo com as necessidades da aplicação em utilizar determinados contextos. Além disso, incluem-se aquelas utilizadas durante a instalação da aplicação no dispositivo e as diretamente especificadas pelo próprio usuário. Quanto à

maneira dinâmica do contexto, esta é caracterizada pelas informações em tempo real que podem ser obtidas por maneiras distintas, como através da utilização de sensores incorporados ou adaptados aos DMs (e. g., GPS – *Global Positioning System*, termômetro e acelerômetro), da análise da rede (utilizando análise do tráfego, diagnosticando a velocidade ou conectividade, utilização de determinados tipos de servidores), da obtenção da data e hora, da análise dos objetos que englobam o usuário, e também do histórico de situações registradas em situações anteriores.

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.

Levando-se em conta a obtenção de informações de contexto computacional, no que diz respeito às características de segurança da rede em uso, verifica-se que estas informações podem ser utilizadas como parâmetro de entrada para o processo de adaptação do nível de segurança em tempo de execução. A informação sobre o nível residual de bateria do DM, também pode ser utilizada pelo processo de adaptação, selecionando um algoritmo eficiente e de baixo consumo de energia para garantir a confidencialidade de dados. Este é um dos objetivos deste trabalho, uma vez que sistemas sensíveis ao contexto devem ser capazes de adaptar o seu comportamento às situações em constante mudança.

2.2 Adaptação de *Software*

Mudanças são inevitáveis na maioria dos sistemas, pois mudam também as gerações de usuários e suas necessidades, de forma que se torna necessária a utilização de novas abordagens de construção de *software* que reduzam os custos e os riscos de evolução dos sistemas existentes, e que ainda reduzam o tempo de inatividade do sistema durante as atualizações [OREIZY, 2008]. As crescentes ameaças no meio sem fio também impõem necessidades de mudanças na concepção de *software* com o intuito de torná-los menos

susceptíveis aos ataques. Conforme mencionado no Capítulo 1, a adaptação de *software* em tempo de execução é uma abordagem apropriada para lidar com essa questão, provendo segurança de acordo com o contexto de uso das aplicações.

Diversos sistemas de segurança têm sido desenvolvidos utilizando-se dos métodos existentes de desenvolvimento de software adaptativo. Tais soluções adaptativas têm sido implementadas nos níveis de hardware, do sistema operacional, de rede e no nível de aplicação [ELKODARY, 2007].

Adaptação de software diz respeito aos sistemas que mudam de acordo com as necessidades impostas a ele, e esta evolução deste tipo de sistemas é motivada pelas necessidades da sociedade [OREIZY, 2008].

Assim como existem diversas formas de tratamento de contextos, também existem maneiras diferentes de realizar adaptação de acordo com as necessidades das aplicações e dos usuários, sendo essa adaptação diretamente dependente do contexto de utilização.

A adaptação de software pode ser realizada de maneira estática ou dinâmica [ROCHA, 2007]. A estática está relacionada com a facilidade de mudança de código de acordo com o surgimento de novos requisitos, enquanto que a dinâmica está relacionada com mudanças de comportamento das aplicações de acordo com o contexto de execução.

Levando-se em consideração o ambiente de computação móvel, no qual ocorrem mudanças de contexto e, consequentemente, as necessidades de adaptação podem ser constantes, a adaptação dinâmica é a maneira que mais se relaciona com essas necessidades.

McKinley [MCKINLEY, 2004] categoriza a adaptação dinâmica de duas maneiras:

a) por **composição** ou **paradigma composicional** – o sistema é composto de acordo com as necessidades identificadas dinamicamente.

b) por **parametrização** – parâmetros pré-configurados do sistema influenciam em seu comportamento.

Abordagens para sistemas adaptativos de segurança, focado em mecanismos de segurança no nível de aplicação, são apresentadas no trabalho de Elkodary (2007), um *survey* que aponta a adaptação dinâmica por **composição** como ponto central de atuação de sistemas de segurança adaptativa [ELKHODARY, 2007 apud MCKINLEY, 2004].

McKinley (2004) classifica a adaptação dinâmica por composição em três dimensões: paradigma computacional, tempo de adaptação e camada de adaptação. Contudo, a dimensão do paradigma computacional é a dimensão mais abrangente para o domínio de segurança, e por este motivo Elkhodary (2007) a adota em seu estudo. As outras duas

dimensões estão relacionadas com a maneira como a composição do sistema acontece em relação ao tempo e à camada de atuação e não são consideradas no *survey*.

Elkhodary (2007) considera ainda a importância de 5 novas dimensões relevantes ao considerar a adaptação dinâmica no domínio de segurança, são elas: autenticação, autorização, tolerância, escala de reconfiguração, e manipulação de conflito.

A dimensão do paradigma computacional de McKinley (2004) somada com essas 5 novas dimensões são agrupadas em duas perspectivas: serviços de segurança adaptativa (com as dimensões autenticação, autorização e tolerância) e método de adaptação (com as dimensões paradigma computacional, escala de reconfiguração e manipulação de conflitos).

Os **serviços de segurança adaptativa** identificam um conjunto de serviços de segurança que devem ser implementados pelos sistemas a fim de satisfazer os objetivos de segurança do software. Não basta a presença destes serviços, é necessário que quando determinados eventos alterarem o nível de ameaça do sistema, os serviços de segurança se adaptem em conformidade [ELKODARY, 2007]. A autenticação se refere ao processo de verificação da identidade de um usuário, que é fornecida para o acesso a um sistema. Já a autorização é um processo de controle de acesso de um usuário aos recursos do sistema, e a tolerância diz respeito ao processo de mascarar as falhas de execução de um sistema através da eliminação dos componentes defeituosos, limitando os efeitos das falhas e alocando substitutos válidos. Cada um desses serviços de segurança deve oferecer uma capacidade de adaptação que garanta uma melhor segurança de acordo com as mudanças percebidas no sistema.

Quanto à perspectiva dos **métodos de adaptação**, é realizada uma análise do mecanismo de configuração e reconfiguração utilizado pelo sistema em três dimensões já citadas anteriormente: o paradigma computacional (i), a escala de reconfiguração (ii) e a manipulação de conflitos (iii).

i) O paradigma computacional é o mecanismo de transformação pelo qual unidades de reconfiguração são iniciadas ou compostas em tempo de execução para a transição de um estado válido de configuração de um sistema para outro. Elkhodary (2007 apud MCKINLEY, 2004) destaca e comenta quatro paradigmas principais de reconfiguração:

> a) Parametrização – a habilidade de mudar o comportamento de uma unidade de configuração em tempo de execução baseada no valor de um contexto ou de um parâmetro selecionado por um usuário. Esta abordagem é adequada quando o número de unidades de configuração é pequeno e todas as possíveis combinações são conhecidas em tempo de projeto. As principais desvantagens desta abordagem são: o

seu suporte limitado para composições complexas e o grande número de configurações que precisam ser codificadas em tempo de projeto. Os trabalhos [ELKHODARY, 2007] e [MCKINLEY, 2004] apresentam mais detalhes sobre parametrização;

b) Composição baseada em componentes – a capacidade de religar diferentes implementações de uma interface de componentes bem definidos em tempo de execução e sem recompilação. Esta abordagem suporta qualquer configuração do sistema, desde que este satisfaça os contratos especificados pelas interfaces. No entanto, a alteração da implementação interna de componentes não é tratada pelo paradigma. Mais detalhes podem ser encontrados em [SZIPERSKI, 1998], [LIU, 2004] e [CAMPBELL, 1999];

c) Reflexão – a capacidade de um programa de observar e modificar seu próprio comportamento através da revelação de alguns detalhes de sua implementação em um meta nível. Modificar, em tempo de execução, a implementação interna de um componente é possível neste paradigma. Além disso, esta mudança não é limitada por um contrato como na composição baseada em componentes. Assim, uma gama muito ampla de reconfigurações inesperadas pode ser executada. Os trabalhos de [SMITH, 1982] e [MAES, 1987] possuem mais detalhes sobre reflexão;

d) Orientação a aspectos – permite a separação de recursos funcionais dos transversais. Em tempo de execução, todos os recursos podem ser compostos com base em um modelo de confecção "*joinpoint*" (ponto de execução no código, que são referências a um conjunto de atributos que caracterizam um comportamento). Orientação a aspectos, assim como reflexão, também suporta uma ampla gama de reconfigurações, além de simplificar o desenvolvimento e a manutenção do sistema de forma significativa. Mais detalhes sobre orientação a aspectos pode ser visto em [MOHAMED, 2000] e [TARR, 2001].

ii) A escala de reconfiguração refere-se a uma escala na qual a reconfiguração pode ser alcançada, seja uma simples unidade, unidades inter-relacionadas ou sistemas amplos. O limite inferior desta dimensão é uma única unidade de reconfiguração que permite adaptação do comportamento ou das invariantes dentro de um único componente ou serviço. Já o limite superior é a reconfiguração de uma arquitetura ampla que envolve reestruturação geral dos componentes do sistema, assim como a alteração das propriedades externas da arquitetura do sistema.

iii) Quanto à manipulação de conflitos, esta se refere à capacidade do sistema para detectar e resolver os conflitos que surgem devido às incompatibilidades entre as unidades de

configuração (interações entre os recursos) ou devido à natureza conflitante dos objetivos almejados. A detecção de conflitos é um requisito essencial para reconfiguração dinâmica, ou seja, o restabelecimento dinâmico da disponibilidade de um sistema. É muito provável, especialmente nas configurações baseadas em tempo de execução, que ocorram conflitos estruturais ou comportamentais. Tais conflitos podem ter consequências graves sobre o sistema e, por isso, devem ser detectados antes da realização da reconfiguração.

Elkodary (2007), em seu trabalho sobre abordagens de adaptação em segurança no nível de aplicação, ao concluí-lo recomenda mais pesquisa sobre projetos de serviços de segurança adaptativos, reuso e manutenção dos paradigmas existentes, formalismo de reconfiguração e sistemas de solução de conflitos. Do ponto de vista dos autores, a segurança adaptativa, em geral, é demandada a partir da fase de projeto, uma vez que existem inúmeras formas de configuração possíveis, além dos ataques diversos que evoluem significativamente com o tempo. Eles destacam ainda a importância de agregar as boas práticas da engenharia de software na definição e construção de sistemas adaptativos sensíveis à segurança.

Entre os métodos de adaptação apresentados, o paradigma computacional é o que está diretamente relacionado com as necessidades de composição da segurança adequada em tempo de execução, proposta nesta dissertação. A manipulação de conflitos e a escala de reconfiguração estão fora do foco desta dissertação.

As dimensões do paradigma computacional podem ser utilizadas e combinadas de acordo com as necessidades dos engenheiros de *software*. Entre essas dimensões, o desenvolvimento baseado em componentes possui características importantes que respondem aos requisitos deste trabalho, tais como a capacidade de adição, remoção e reconfiguração de implementações em tempo de execução [MCKINLEY, 2004], [SZIPERSKI, 1998], [ROCHA, 2007] e [VIANA, 2005]. Tais características possibilitam que implementações diferentes dos algoritmos criptográficos possam ser inseridas na aplicação sem causar impacto funcional. Por esse motivo, este trabalho se concentra na adaptação baseado em componentes, considerando a parametrização e a reflexão. Já a orientação a aspectos não é tratada neste trabalho pela complexidade envolvida na manutenção e implementação do código [MCKINLEY, 2004], o que descaracterizaria a necessidades deste trabalho, no que diz respeito à facilidade de compreensão de seu código e abertura à evolução.

2.3 Adaptação e Desenvolvimento Baseado em Componentes

A necessidade de sintonia com os avanços tecnológicos coloca o software em permanente estado de evolução. Tanto a implantação de *software* cresceu quanto as técnicas utilizadas na sua construção foram aprimoradas, impondo grandes mudanças ao ambiente no qual foram inseridas. A componentização adicionou qualidade ao processo de desenvolvimento, devido à facilidade de modificações e à alta dinamicidade no conhecimento e na utilização [SZYPERSKI, 1998].

A definição de componentes apresentada por [SZYPERSKI, 1998] é "unidades executáveis e independentes de produção, aquisição e implantação que podem formar um sistema funcional". A independência e a capacidade dessas unidades em serem executáveis, mencionadas por SZYPERSKI (1998), são essenciais para utilização e aproveitamento destas, pois servem para a composição de produtos novos buscando os benefícios da reutilização de software [SZYPERSKI, 1998].

Diante das definições existentes de componentes percebe-se uma relação de convergência entre elas: um componente é uma unidade independente de software de propósitos claros, com interfaces de configuração e atributos de qualidade [BOSCH, 2000].

Componentes são criados para serem independentes e para interagirem com ambientes e com outros componentes, todavia, para garantir essas características é necessário o uso de interfaces, as quais definem um contrato entre a funcionalidade requerida de um componente e a disponibilidade desta funcionalidade por outro [BOSCH, 2000]. A interface pode ser independente ou implementada pelo componente, e serve para diminuir o acoplamento, para definir claramente suas interações com este componente e para encapsular suas fronteiras.

Três tipos de interfaces são geralmente necessárias em um componente: i) aquelas fornecidas por ele, ii) as que são necessárias para seu uso e iii) as de configuração [BOSCH, 2000]. As duas primeiras são interfaces de interação com outros componentes, sendo que a última é para a configuração das variações deste componente, necessárias para a sua utilização em um determinado sistema.

Pelo fato de serem normalmente bem projetados e testados, além de possuírem ampla utilização, os componentes inspiram confiança. A sua utilização na arquitetura de um sistema pode incrementar a confiança no sistema como um todo [BOSCH, 2000].

Componentes podem ser criados especificamente para atender determinado requisito ou para criar vantagens estratégicas. Soluções tradicionais totalmente integradas requerem atualizações periódicas, entretanto, estas resultam em um processo custoso que pode implicar a migração de banco de dados, assegurar compatibilidade, treinar equipe, além da necessidade de compra de equipamentos mais robustos.

No desenvolvimento baseado em componentes, verifica-se que as evoluções necessárias dos componentes permitem a simplificação de operações [SZYPERSKI, 1998], contudo, um novo modo de gerenciamento é requerido, que implicará em ganhos potenciais.

Conforme mencionado na Seção 2.2, a propriedade de uma aplicação de adequar-se às mudanças no contexto em que é executada impõe a adoção da adaptação dinâmica. Este tipo de adaptação exige um alto nível de dinamicidade, pois as mudanças no contexto acontecem com muita frequência, e isto é comum quando se trata de aplicações para dispositivos móveis. Estas mudanças podem ser decorrentes, em certos momentos, das alterações da localização do dispositivo, do interesse do usuário e da largura de banda de comunicação [ROCHA, 2007].

As maneiras de realizar adaptação podem acontecer tanto em tempo de construção como em tempo de execução. Quanto à adaptação em tempo de execução, com o objetivo de atender às mudanças que ocorrem no ambiente em que esta aplicação está sendo executada, a aplicação modifica seu comportamento sem a intervenção do engenheiro de software [VIANA, 2005]. Um exemplo é uma aplicação capaz de modificar as propriedades de segurança para se adequar às vulnerabilidades existentes nas diversas redes que ela costuma utilizar para transmitir seus dados. Assim, esta aplicação pode adotar um elevado nível de segurança ao identificar que a rede em uso em determinado momento é uma rede que oferece risco de segurança para o tráfego de seus dados.

Portanto, este trabalho pretende utilizar a adaptação dinâmica, pois pretende oferecer uma maneira de atenuar as ameaças existentes no meio sem fio e que atue dinamicamente de acordo com o contexto no momento da execução das aplicações. O desenvolvimento baseado em componente é uma tecnologia que permite a adaptação dinâmica [McKINLEY, 2004].

2.4 Conclusões

Neste capítulo foram apresentadas considerações relacionadas com adaptação de software através da exploração de informações de contexto. Além disso, foram apresentados estudos na área de segurança adaptativa, conceitos e abordagens em adaptação dinâmica.

A complexidade no tratamento de questões relacionadas à sensibilidade ao contexto impõe desafios aos engenheiros de software que trabalham com aplicações para DMs, principalmente quando se referem à segurança. Isso acontece devido à dinamicidade dos ambientes móveis em que estas aplicações são utilizadas. Esses desafios estimulam a descoberta de alternativas que tornem estas aplicações capazes de se adaptarem ao contexto corrente, oferecendo segurança de modo eficiente. Estes foram os fatores motivadores para o desenvolvimento desta dissertação.

O próximo capítulo apresenta os conceitos sobre segurança e seus princípios básicos, a sua abordagem em sistemas sem fio e em aplicações para DM, assim como soluções de medição em segurança e a prática da sustentabilidade na computação móvel. Além disso, são apresentados trabalhos existentes na área de segurança adaptativa.

3 SEGURANÇA ADAPTATIVA: CONFIDENCIALIDADE E EFICIÊNCIA

Neste capítulo são detalhados os aspectos de segurança adaptativa, com foco na confidencialidade e na eficiência proporcionada pelos algoritmos criptográficos que fornecem tal confidencialidade. A Seção 3.1 apresenta conceitos sobre segurança e seus requisitos básicos, define criptografia e faz uma análise sobre a utilização da segurança em sistemas de comunicação sem fio. Na Seção 3.2 é abordada a mensuração e sua relação com segurança para aplicações de DM. A Seção 3.3 traz considerações sobre eficiência e sua forma de adoção na computação. Por fim, a Seção 3.4 apresenta os trabalhos relacionados com esta dissertação, seus pontos fortes e fracos e as principais diferenças com este trabalho.

3.1 Segurança

Em sistemas de informação a segurança está relacionada com o controle de acesso, de alteração ou de exclusão da informação, porém é importante destacar que nenhum sistema pode ser considerado totalmente seguro. Portanto, de acordo com o valor da informação, mecanismos podem prover um nível de segurança de maneira tal que o esforço requerido para a violação deste controle não seja compensável [BISHOP, 2002] [TADDEO apud PFLEEGER, 2006]. Uma definição clássica é que segurança é a proteção contra a indisponibilidade, vazamento e a leitura ou modificação não-autorizada de informações [BISHOP, 2002].

3.1.1 Requisitos Básicos de Segurança

Segundo Bishop (2002), a segurança depende diretamente de seus componentes básicos, confidencialidade, integridade e disponibilidade. Contudo, as interpretações desses componentes variam, assim como os contextos em que eles ocorrem, e são influenciadas pelas necessidades dos indivíduos, seus costumes e leis [BISHOP, 2002].

A confidencialidade pode ser compreendida como a propriedade de segurança que é caracterizada pela ocultação de informações ou de recursos computacionais, restringindo o seu acesso para oferecer uma garantia de que ele esteja disponível apenas quando permitido pelo proprietário da informação [BISHOP, 2002]. Exemplificando pode-se considerar aplicações de DM que enviam informações do usuário no meio sem fio. Essas informações podem ser de conhecimento público, como a divulgação da localização de um restaurante de comida chinesa que o usuário costuma frequentar para ser disponibilizada em sua rede social. Porém, outras informações podem ser de conhecimento restrito e necessitam de controle, como as informações de senha de acesso a determinados serviços. Dentro deste domínio de informações sigilosas existem aquelas que possuem menores e maiores graus de sigilo. Quanto mais secreta a informação for, mais técnicas para a garantia do controle de seu conhecimento deverão ser postas em prática. Esta ocultação de informação para garantia de seu segredo é necessária, por exemplo, em domínios como aqueles onde acontece tráfego de informações militares, informações sigilosas de governo ou informações que contenham segredos comerciais entre outros casos onde se identifica a necessidade de confidencialidade da informação.

Em sistemas de informação, integridade é qualidade que garante que o dado não seja arbitrariamente modificado [BRAGG et al, 2004]. Ela caracteriza a manutenção do estado íntegro da informação, tentando garantir a sua exatidão original.

A disponibilidade, no âmbito da computação, é definida como sendo a capacidade do uso de informações ou recursos desejados [BISHOP, 2002].

O controle do conhecimento da informação, a garantia de que as mudanças nesta informação aconteçam apenas quando autorizadas e a disponibilidade dessas informações são os pilares da segurança [BISHOP, 2003].

Segundo Bishop (2002), os métodos, ferramentas e técnicas usadas para incrementar as políticas de segurança são conhecidos como mecanismos de segurança. Um mecanismo de segurança que funciona como um controle de acesso à informação para preservar sua confidencialidade é a criptografia, a qual atua de forma a tornar os dados incompreensíveis, podendo ser compreendido apenas após o seu acesso através do uso de uma chave de criptografia [BISHOP, 2003]. Este trabalho se aprofundará no estudo do componente confidencialidade e o mecanismo de segurança de criptografia de dados, pois o objetivo da proposta é proteger as informações que são trafegadas no meio sem fio.

3.1.2 Criptografia

O conceito de criptografia é que ela é a ciência da manutenção de mensagens seguras, e criptografar é o processo de modificar uma mensagem de tal forma que oculte o seu conteúdo [COULOURIS et al, 2005]. A criptografia utiliza técnicas para transformar uma informação fácil de ser compreendida em um código difícil de ser percebido.

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.

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.

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.

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. 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.

3.1.3 Segurança em sistemas de comunicação sem fio

A comunicação sem fio apresenta uma preocupação quanto à segurança, pois utiliza o espaço aberto como meio de propagação das informações, que podem ser captadas por um dispositivo receptor. As vulnerabilidades deste tipo de comunicação vão além das ameaças encontradas em redes cabeadas [BRINGEL, 2004].

Devido à exposição dos dados trafegados na comunicação sem fio, alguns protocolos incorporam mecanismos de segurança na camada de enlace dos dados, contudo, algumas aplicações necessitam de segurança na camada de aplicação (segurança fim a fim).

A implementação de segurança fim a fim aumenta a garantia de que os critérios de segurança requeridos pela aplicação estarão sendo atendidos durante o tempo em que houver comunicação. Esta agregação de segurança fim a fim é necessária, pois a segurança existente em protocolos de enlace não é suficiente para determinados tipos de aplicações que requerem um maior nível de segurança. Para implementar a segurança em aplicações para DMs os algoritmos simétricos são utilizados, devido ao fato da execução destes algoritmos ser mais indicada para estes tipos de dispositivos, uma vez que eles requerem menos recursos computacionais do que os assimétricos [BRINGEL, 2004] [CARVALHO, 2008].

A segurança é uma das principais preocupações em aplicações de DMs devido à necessidade de garantia de identidades e do estabelecimento do segredo na comunicação. A garantia de autenticação é difícil, pois a informação pode ser alterada nos trajetos de envio e recebimento. A garantia de recebimento de uma mensagem apenas por parte do destinatário também é algo muito difícil de ser proporcionado, pois a mensagem é enviada no meio sem fio, um ambiente aberto onde um dispositivo preparado para este fim, pode capturar a mensagem.

Nesse domínio da computação móvel existem muitas dificuldades para o tratamento seguro das informações. Devido a estas dificuldades, ao alto custo de implementação e à baixa eficiência proporcionada pelo uso de segurança, normalmente ela é deixada de lado [CHEN et al., 2000], [LI, 2006], [CHEN, 2010] e [TADDEO, 2010]. Contudo, é importante oferecer uma solução satisfatória de privacidade dessas informações ao invés de ignorá-la.

A escolha correta dos mecanismos de segurança para as aplicações criadas para dispositivos móveis são fatores chaves, pois as características específicas de cada dispositivo podem atuar como fatores limitantes aos tipos de serviços oferecidos [CARVALHO, 2008]

[BRINGEL, 2004]. Por isso, é necessário medir as necessidades destas aplicações e adequá-las aos dispositivos nos quais serão executadas.

3.2 Medição em segurança de aplicações móveis

Mensuração ou medição é o processo pelo qual números ou símbolos são associados aos atributos de entidades do mundo real de forma que sua determinação esteja de acordo com regras claramente definidas [FENTON, 1994]. Fenton (1994) explica que estas entidades podem ser a representadas por um objeto, uma fase de um processo de software ou uma pessoa. Já os atributos são propriedades ou uma característica de uma dessas entidades, como a duração da fase de processo ou a altura de uma pessoa. Se este atributo pode ser avaliado isoladamente ele pode ser chamado de métrica. Para exemplificar, o esforço envolvido em um projeto é uma avaliação relacionada com o tamanho do projeto. Esse esforço é uma métrica, e uma forma de calcular essa métrica seria o somatório de cada registro de hora trabalhada (que é uma medida) em todo o cronograma e por todos os envolvidos.

A medição é uma atividade chave para a compreensão da complexidade ocasionada pela quantidade de parâmetros que determinado sistema possui. Um exemplo de tal complexidade é citado por Fenton (1994) ao se referir à alta complexidade de se obter uma medida geral de controle de fluxo representada por um único valor.

Uma definição sobre medição com foco no quesito de segurança destacado em [CARVALHO, 2008] é que a medição pode ser vista como o processo de definir metas, identificar objetivos, implementar, analisar a eficiência e eficácia e verificar os impactos sobre medidas que possam potencialmente melhorar a segurança da informação de uma organização. Em Engenharia de Software, medida indica quantificação (extensão, quantidade, dimensão, capacidade ou tamanho) de certo atributo de um produto ou de um processo. Assim, medição é o ato de determinar a medida, enquanto que uma métrica de software relaciona as medidas individuais de alguma forma (como o número médio de erros encontrados por teste de unidade).

A medição oferece mais objetividade na condução e análise de atividades e decisões, isso pelo fato de que as medições e suas representações, seja em números, gráficos ou qualquer outro modo, oferecem uma maior compreensão.

A confidencialidade pode indicar graus que definem a complexidade da proteção proporcionada, então em certos domínios ela precisa ser medida. Contudo, a confidencialidade é provida com o uso da criptografia que, por sua vez, utiliza algoritmos criptográficos para esta tarefa. Estes algoritmos possuem diferentes complexidades em suas implementações e de acordo com esta complexidade o algoritmo oferece níveis diferentes de oferta de segurança. Essa graduação de oferta de segurança dos algoritmos pode ser influenciada por alguns fatores como o conhecimento de fraquezas ou vulnerabilidades do algoritmo, tamanho de chave, notícias de quebra de segredo e baixa complexidade na ocultação da informação. Estas fraquezas podem compor um grau que define a graduação de confidencialidade proporcionada por determinado algoritmo. Normalmente este grau de confidencialidade é definido de acordo com o tamanho da chave do algoritmo, com base na tentativa de quebra do segredo por força bruta. Isto acontece devido à alta complexidade para medição, de maneira eficaz, dos outros fatores mencionados. Assim, a confidencialidade pode ser classificada de acordo com o algoritmo de criptografia que está sendo utilizado e esta classificação é indicada pelo grau de confidencialidade.

Uma maneira de mitigar os riscos de uma aplicação de acordo com o grau de confidencialidade que esta aplicação necessita possuir para a execução de suas tarefas é utilizando medições. O que é necessário neste caso é identificar e deixar claro qual a necessidade de segurança requerida pela aplicação e depois disto saber o quanto segura esta aplicação precisa ser nas diversas situações em que se espera que ela esteja em execução. Após esta definição é necessário colocar em prova este nível de segurança, que avaliará, por exemplo, se o algoritmo criptográfico selecionado para uso atende às necessidades de confidencialidade da aplicação para o contexto no qual ela está sendo executada.

No caso de testes da confidencialidade proporcionada para a aplicação, isto é uma tarefa complexa e muito dispendiosa, tanto em relação ao tempo quanto à criação dos ambientes de verificação. Adicionando o fato de ser monetariamente muito custoso e com possibilidades de aplicação de diversos modelos contraditórios [GENTRY, 2004]. Desta forma, ela pode ser feita baseando-se em reflexão de dados históricos para simulação de ambientes de teste destes cenários, como explicado a seguir.

Um modelo adversativo ou contraditório utilizado para testar o nível de segurança do algoritmo deve possuir limitações das capacidades de esperança desta contradição de confidencialidade, por exemplo, se existem notícias de que um determinado algoritmo já foi quebrado n vezes e isto aconteceu em x segundos, utilizando os z recursos computacionais e intelectuais. Esta informação pode servir para configurar o modelo adversativo. Assim como a

notícia de que outro algoritmo é superado facilmente em x segundos, utilizando recursos fáceis de serem encontrados, e com um software simples de ser montado, é uma informação que certamente excluiria este algoritmo da lista de uso em uma aplicação que necessita de um maior grau de confidencialidade.

Como não é fácil montar um modelo desses para ser seguido na prática, o histórico e o bom senso podem ser usados para a definição dessas limitações de verificação de confidencialidade do algoritmo.

Um modelo de avaliação de nível de segurança proporcionada em dispositivos com restrições de recursos computacionais é apresentado por Jianyoung Chen [CHEN et al. 2010]. Ele utiliza variáveis com definição de peso para avaliação dos algoritmos de criptografia, o que requer o conhecimento em algoritmos de segurança por parte do engenheiro de software, e destaca que seus pesos são baseados de acordo com a complexidade computacional e o tamanho de chave dos algoritmos. Tal modelo foi considerado para medições necessárias para a implementação desta dissertação.

3.3 Computação Verde na criação de *software*

Em 1992 surgiu um selo para identificação de equipamentos que possuíam características de eficiência energética [STEPHEN, 2009]. Este selo foi criado pela Agência de Proteção Ambiental dos Estados Unidos quando lançou o *Energy Stars*, uma abordagem de uso voluntário deste símbolo que acabou se tornando uma importante certificação com reconhecimento significativo nos Estados Unidos e em outros países do mundo. A idéia se popularizou e 18 anos depois está presente em vários produtos utilizados na tecnologia da informação.

Conforme o tempo avança mais se tem notícia sobre a prática de metodologias de produção ecologicamente corretas ou preocupações verdes na manufatura de diversos produtos em diversas áreas da ciência. Na engenharia de *software* não é diferente. Apesar de ser mais notada na construção ou adaptação de infra-estrutura e na preparação dos ambientes computacionais, a computação verde também se manifesta na criação de *software* [HERRICK, 2009]. Um exemplo prático disto é a preocupação de criação de algoritmos eficientes com a intenção de agilizar sua execução e evitar processamento desnecessário em equipamentos.

A prática da *green computing* em empresas, universidades e em locais onde existe infra-estrutura de sistemas de informação já produzem publicações científicas como em [STEPHEN, 2009], [HERRICK, 2009] e [WILBANKS, 2008].

A *green computing* ou computação verde é o estudo e a prática da utilização eficiente dos recursos computacionais, primando pela redução do uso de materiais maléficos à natureza, maximizando a eficiência energética durante a vida útil do produto, promovendo a reciclagem e a biodegradabilidade do descarte desses produtos e evitando o aumento dos resíduos de sua fabricação [WILBANKS, 2008].

Dentre as áreas de preocupação e desenvolvimento de planos e atividades para redução de consumo e melhor aproveitamento do potencial tecnológico estão o *E-Wast* ou resíduo eletrônico, os *data centers* e servidores, os PCs, monitores e estações de trabalho, o *software*, as telecomunicações e as métricas. Nestas áreas de concentração existem diversas orientações de como proceder para aplicar a *green computing* e ter uma atitude ecologicamente mais responsável e também gerando economia de utilização de recursos e consequentemente reduzindo as despesas monetárias.

No que se refere ao software, a computação verde inspira muitas possibilidades, desde a substituição de algoritmos pouco eficientes em sistemas, até a utilização de sistemas operacionais que agregam a utilização eficiente dos recursos computacionais com esta visão verde [STEPHEN, 2009]. Grandes empresas de construção de sistemas operacionais já incorporam essa tendência na sua estratégia de negócios, percebendo a mudança de atitude do mercado consumidor. Elas constroem e oferecem produtos de software mais eficientes e preocupados com o consumo energético dos equipamentos onde esses softwares serão executados. Este trabalho considera o uso da computação verde no provimento de segurança para aplicações de DM, quando o contexto permitir a flexibilidade na provisão de criptografia, com o intuito de diminuir o consumo da energia da bateria. Essa diminuição do consumo permite o aumento da disponibilidade do hardware e conseqüentemente o aumento de sua vida útil.

3.4 Eficiência na Adaptação em Segurança

Conforme apresentado na Seção 2.2, a adaptação é considerada em sistemas com variadas preocupações, porém, no tocante à adaptação em segurança existe uma consciência geral de

que o assunto é bastante relevante e carece de pesquisa. Principalmente, para indicação de estratégias para resolver o *tradeoff* entre oferecer mais segurança e otimizar o uso dos recursos computacionais. As propostas encontradas e destacadas a seguir, com exceção de Pirmez (2009), tratam de adaptação de protocolos de segurança na camada de enlace com diferenças bem sutis. Entretanto, nenhuma delas mantém, no comportamento adaptativo, a independência dos requisitos de segurança de cada aplicação, uso das características dinâmicas de contexto na adaptação, e preocupação com utilização dos recursos.

Um mecanismo de adaptação dinâmica é encontrado no trabalho de Taddeo (2010) voltado para redes de sensores sem fio (RSSF), o qual adapta a segurança de forma gradual pela seleção de algoritmos criptográficos que são aplicados nas aplicações em uso pelos sensores. O autor afirma que normalmente neste tipo de rede se utiliza uma abordagem onde, ou se usa o melhor algoritmo de segurança ou não se usa nenhum. Esta abordagem se estende para os ambientes de computação móvel contrariando as necessidades deste domínio, que por incluir mobilidade, flexibilidade, configuração em tempo real e dinamicidade de ambientes, necessita de segurança adaptativa [TADDEO, 2010]. O trabalho destaca também que o problema principal e complexo neste tipo de adaptação é a manutenção do nível adequado de proteção, principalmente quando ela é feita em tempo de execução.

Taddeo (2010) propõe então, um mecanismo que lida com adaptação de segurança de acordo com os requisitos de segurança da aplicação e com as restrições de energia de forma dinâmica. Ele se baseia no princípio da proteção adequada [TADDEO apud PFLEEGER, 2006], que prega que uma informação deve ser protegida em escalas consistentes com este valor, sendo assim aplicada adequadamente de acordo com o contexto. Seu foco principal é satisfazer os requisitos de consumo de energia, atuando na adaptação do nível do sinal propagado, encerrando algumas aplicações e com base nas políticas definidas que atuam sobre todas as aplicações.

O trabalho de Taddeo (2010) tenta proporcionar garantias de segurança para as aplicações em vários níveis no maior tempo possível de acordo com as restrições de energia. A diferença de sua estratégia para a adotada nesta dissertação é que ele provê apenas um algoritmo para uso em todas as aplicações, enquanto o trabalho desta dissertação mantém um algoritmo para cada aplicação e ainda provê uma avaliação mais complexa e baseada no contexto de uso da aplicação para melhor prover esta adaptação.

Outro trabalho relacionado utiliza a adaptação com base em estudos de possíveis cenários para decisões em tempo de projeto, considerando sua adoção em dispositivos com limitações de poder de processamento, banda de comunicação, tempo de vida da bateria e

memória [HAMAD et al., 2009]. Os autores investigam alguns algoritmos considerados pelos autores como os mais populares quanto às suas complexidades e utilização dos recursos. Sua adaptação está centrada em oferecer aos usuários opções para que eles escolham o melhor esquema de segurança para ser adotado. Apesar dos autores defenderem a utilização de algoritmos de criptografia e maneiras de adaptação que também proporcionem diminuição do consumo de bateria, eles consideram isto como uma abordagem a ser tratada em trabalhos futuros. A proposta de Hamad (2009) deixa todas as decisões de adaptações para serem realizadas pelo usuário e atua na camada de enlace e esta adaptação não é dinâmica, diferente da proposta desta dissertação que limita estas políticas de decisões a partir do usuário e atua em diferentes camadas.

Um serviço de criptografia adaptativo é proposto por Izquierdo [IZQUIERDO et al. 2007], levando em conta as limitações de capacidade de processamento e requisitos de segurança. É uma arquitetura de criptografia de dados que usa mais de um algoritmo para criptografar vários blocos de informação, de acordo com os recursos dos dispositivos e com a rede. O serviço usa a criptografia de apenas alguns blocos da informação, ou seja, dentre o total de blocos trafegados alguns trafegam sem proteção [IZQUIERDO, 2007]. Outros algoritmos podem ser utilizados em paralelo para criptografar diferentes blocos de informação. Os autores propõem o uso de criptografia de trechos da informação de acordo com áreas de localização do DM no momento do uso. Sua deficiência está na identificação dos trechos de informação que necessitam de melhor criptografia, o que não é tratado no trabalho, e que pode comprometer o sigilo de uma informação. O serviço também repete a criptografia em determinados momentos, aumentando o consumo de recursos, pois blocos já criptografados podem receber nova criptografia ou serem decifrados e criptografados novamente com outro algoritmo.

Com foco em desempenho, o trabalho de Bruno Rocha [ROCHA et al. 2010] é basicamente um *middleware* adaptativo de seleção dinâmica de protocolos de segurança, na camada de enlace. O trabalho usa variações de parâmetros da rede sem fio, recursos disponíveis do sistema em uso, níveis de segurança, e trata diferentes tipos de dados com configurações específicas de segurança. Nesta abordagem os autores deixam claro que não será definido qual algoritmo ou protocolo de segurança é mais forte do que o outro, deixando esta tarefa para ser definida pelo engenheiro de software para então ser adotada no middleware. Apesar dessa afirmação eles usam alguns protocolos e algoritmos, classificando-os como de criptografia de nível forte, e não justificam a adoção ou a classificação.

O Prometheus [PIRMEZ, 2009] é definido como um serviço de segurança ciente de contexto capaz de adicionar controles de segurança dinamicamente. Apesar de propor a solução de uma porção de problemas não o faz, ou faz com restrições, deixando o restante como trabalhos futuros. O trabalho não oferece algumas características destacadas como importantes pelo próprio autor, como considerações sobre medições de consumo de energia e o devido tratamento para redução de consumo. Sua adoção está focada em nichos de aplicações, onde as políticas de utilização destas aplicações precisam ser bem definidas, com regras e definições de prioridades de execução entre si. As relações entre as aplicações existentes no dispositivo precisam estar definidas no Prometheus para serem executadas, estando a ele atreladas, sem interdependência de uso. Assim, o uso de uma aplicação depende da sua prioridade configurada no Prometheus, necessitando que o usuário tenha que encerrar uma aplicação de prioridade menor para executar outra de prioridade maior.

O autor considera que as entradas de informações de contexto e outras necessárias para a adaptação serão fornecidas quando necessário, e não se atém ao tratamento de captura de informação de contexto necessária para a adaptação ou qualquer outra entrada. É diferente da proposta desta dissertação que indica quais informações de contexto são necessárias, as captura em tempo de execução e as utiliza na adaptação da segurança, primando pela independência das aplicações em execução.

O SARM (*Security Adaptation Reference Monitor*) foi proposto por Maliki (2010) para lidar com ambientes dinâmicos da rede sem fio e também defende a abordagem de formas flexíveis para lidar com a segurança de maneira dinâmica, e com uso de informações de contexto. O autor propõe um monitor genérico de adaptação de segurança considerando o desempenho em relação ao consumo de energia. O trabalho também defende o uso de adaptação de segurança em tempo de execução na rede sem fio para atenuar as consequências do número substancial de ameaças quando não se pode eliminá-las por completo. O autor justifica que não há consciência de qual mecanismo de segurança se deve utilizar nas aplicações e que sejam tão dinâmicos na proteção quanto os ataques são nas ameaças.

Maliki (2010) destaca que um sistema de segurança adequado deve ser provido de maneira a conseguir o máximo de segurança mantendo o desempenho dos outros serviços de maneira aceitável. Sua proposta centraliza o tratamento de segurança de várias aplicações no monitor que é integrado ao *kernel*, assim todos os programas de comunicação passam pelo SARM. O mecanismo é focado em avaliações realizadas pelo usuário que necessita ter certo conhecimento de segurança para fazer com que o SARM seja mais efetivo. Para cada aplicação o usuário deve preencher uma avaliação técnica sobre a aplicação em questão

detalhando também suas preferências de uso e é baseado nestas avaliações que o SARM realiza as adaptações. Os riscos, o desempenho e o consumo de energia são definidos de acordo com as preferências do usuário e com um *log* de informações já existentes. Estas informações são utilizadas em conjunto com informações de contexto para decidir se deve aplicar segurança ou não em determinada rede ou para decidir que rede usar entre as disponíveis no momento.

O autor informa que a verificação do SARM seria difícil devido à existência de grande variedade de redes sem fio na atualidade e por isso restringiu-se a validação de uma versão *light,* sem a implementação de todos os módulos e usando apenas redes sem fio baseadas em proximidade. Para identificar a necessidade de segurança de uma rede nova, utiliza-se do envio de um formulário de avaliação aos usuários da rede, onde define se usa segurança ou não, de acordo com as respostas, ou usa uma decisão aleatória para esta escolha.

O trabalho se restringe a avaliar se usa ou não segurança e com esta decisão considera que está economizando energia quando não for necessário usar segurança. O autor encoraja a pesquisa de técnicas que melhorem o *tradeoff* entre segurança e consumo de energia e indica que o ajuste adequado do nível de segurança de acordo com a análise do ambiente é a forma mais adequada de melhorar a segurança para as redes sem fio e ainda proporcionar economia de recursos.

Na Tabela 1 pode ser verificado que os trabalhos relacionados deixam de fora pelo menos um dentre os aspectos que são considerados nesta dissertação de mestrado, a qual respeita os requisitos de segurança e de execução de cada aplicação, usa a sensibilidade ao contexto na adaptação dinâmica e prima pela eficiência na alocação dos recursos do DM, além de prover segurança na camada de aplicação mantendo sempre um nível de segurança.

Tabela 1 – Principais características dos trabalhos relacionados e da proposta

Abordagens propostas	Usa graus ou escalas de segurança	Sensível ao Contexto	Trata economia de recursos	Mantém sempre um nível de segurança	Adaptação dinâmica em tempo de execução	Atua na camada de aplicação
Taddeo	✓	✓	✓		✓	
Hamad	✓		✓	✓		
Izquierdo		✓	✓		✓	
Rocha	✓	✓	✓		✓	
Pirmez		✓	✓		✓	✓
Maliki		✓	✓		✓	

3.5 Conclusões

Este capítulo apresentou conceitos sobre segurança e segurança adaptativa, além de considerações sobre a sua adoção em sistemas de comunicação sem fio, focando sempre no tratamento de confidencialidade promovida pela criptografia. Medições em segurança e as complexidades envolvidas nesse contexto também foram assuntos tratados no capítulo. A questão do uso eficiente dos recursos computacionais através da *Green Computing* foi um dos temas apresentados como importantes na criação de *software*. Por fim, foram apresentados os trabalhos relacionados com esta dissertação e ao final foi apresentada uma tabela com as principais características de cada um, assim como suas principais diferenças com a proposta desta dissertação.

As grandes dificuldades envolvidas em quantificar a segurança proporcionada dão uma idéia da complexidade em oferecer mecanismos de segurança, principalmente quando envolvem adaptação dinâmica. Estas dificuldades desafiam trabalhos que oferecem soluções para este domínio, porém também servem de estímulo para a criação de novas propostas.

O próximo capítulo apresenta o mecanismo proposto nesta dissertação, como ele foi construído, seu funcionamento, sua preocupação com a eficiência e suas contribuições para o incremento de segurança em aplicações de DMs.

4 O MECANISMO PROPOSTO

Este capítulo apresenta um mecanismo de segurança com adaptação dinâmica, tendo como foco confidencialidade para aplicações de DMs. A visão geral e os algoritmos do mecanismo são apresentados na Seção 4.1. A arquitetura do mecanismo e o seu funcionamento são detalhados na Seção 4.2. Na Seção 4.3, estão descritas as atividades chave que possibilitam a definição do algoritmo a ser utilizado pelas aplicações e, por fim, na Seção 4.4, o capítulo é concluído.

4.1 Visão Geral

Para lidar com a confidencialidade de informações existem duas maneiras básicas destacadas por Hongwei Li (2006) e descritas a seguir. Na primeira, a confidencialidade é baseada em mecanismos previstos pela infraestrutura computacional subjacente (composta pelas camadas que estão mais abaixo da camada de aplicação), com a vantagem de ser completamente transparente ao usuário ou à aplicação. Esta forma, apesar de tornar transparente o processo de confidencialidade, tem a desvantagem de impor esse tratamento para todas as aplicações, impossibilitando a exploração das suas propriedades específicas e adicionando uma alta carga no sistema como um todo [LI , 2006]. Quando se considera o domínio dos DMs, esta carga é ainda mais crítica devido à sua limitação de poder computacional, e à dificuldade de agregação de *hardware* específico para desempenhar esta tarefa, diferente da infraestrutura das redes convencionais.

A segunda maneira é focando na camada de aplicação, onde se utiliza a confidencialidade apenas quando necessário, com a clara vantagem da exploração das propriedades específicas de certas aplicações para criação de esquemas de criptografia mais eficientes.

O Mecanismo de Adaptação Dinâmica de Segurança (MeSAD), aqui proposto, visa possibilitar a adoção desta última solução, explorando as características individuais de cada aplicação para melhor adaptar a utilização de confidencialidade, diminuindo assim a carga imposta pelo custo computacional desta operação e preservando os recursos do DM.

A visão geral do MeSAD é apresentada na Figura 2, a qual mostra uma aplicação que precisa garantir confidencialidade da informação a ser trafegada. A aplicação implanta aspectos de segurança para realizar a transmissão de seus dados. O requisito de segurança considerado é o de confidencialidade, que utiliza a criptografia (blocos tracejados) dos dados através de um algoritmo criptográfico, todos representados na Figura 2. Este último está sendo definido pelo MeSAD, que utiliza informações de contexto do DM, do usuário, da aplicação, da rede e dos algoritmos criptográficos, no momento da execução da aplicação, para avaliar qual é o algoritmo criptográfico mais adequado.

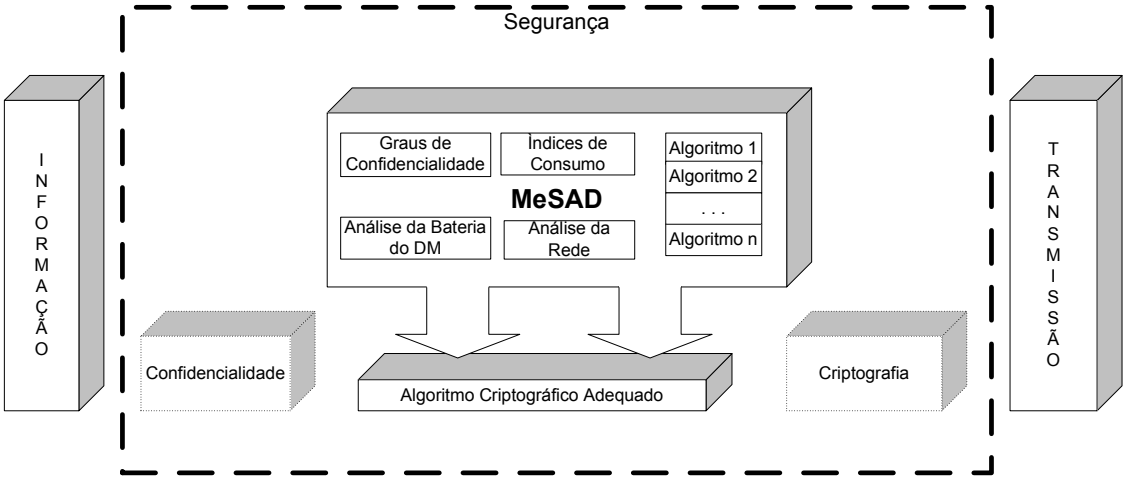


Figura 2 – Visão Geral do MeSAD

O ambiente no qual são utilizados os DMs apresenta uma série de particularidades. Dentre elas, a descrita no Capítulo 1 desta dissertação: a necessidade da adequação dos diferentes níveis de confidencialidade devido aos diferentes riscos existentes nos vários ambientes de utilização dos DMs. Esta adequação do nível de confidencialidade de acordo com as mudanças do ambiente de uso do DM, geralmente não é considerada devido à complexidade envolvida neste tratamento e, por conta disso, a segurança é simplesmente deixada de lado ou utilizada de forma inadequada [ROCHA, 2010], [LI, 2006], [IZQUIERDO et al., 2007], [CHEN et al., 2010] e [TADDEO et al., 2010].

O MeSAD foi projetado para resolver esse problema da adequação do grau de confidencialidade de acordo com os ambientes de utilização do DM. Ele atua para manter o grau de confidencialidade mais adequado, de maneira dinâmica, à medida que o DM passeia pelos diversos ambientes.

Neste tipo de cenário, no qual a mobilidade do DM faz variar as necessidades da aplicação constantemente, um aspecto crítico relacionado com sistemas sensíveis ao contexto

é a capacidade de adaptar o comportamento do sistema de acordo com as mudanças identificadas nesses contextos. Para tratar este problema de forma adequada, informações oriundas dos contextos devem ser adquiridas, por vezes, de fontes externas. Isso pode ser feito por sensoriamento, tanto de hardware como de software. O MeSAD utiliza informações de contexto capturadas em tempo de execução para definir a criptografia mais adequada, a fim de garantir o grau adequado de confidencialidade da aplicação.

O MeSAD preserva a autonomia das aplicações quanto ao tratamento da confidencialidade, dando suporte à escolha correta e ao uso eficiente dos algoritmos de criptografia, considerando diversas informações de contexto detalhadas neste capítulo. Desta forma, ele possibilita que diferentes aplicações possam utilizar algoritmos de criptografia distintos, de acordo com seu contexto e com seu grau de confidencialidade. Atuando assim, o MeSAD proporciona o emprego mais eficiente dos recursos do dispositivo e evita a carga imposta pelas soluções de segurança generalizadas (que atuam nas camadas inferiores, mencionadas no início da Seção 4.1).

Os algoritmos de funcionamento do mecanismo proposto estão evidenciados na Figura 3, a qual mostra um diagrama de atividades UML (*Unified Modeling Language*) que utiliza partição de atividades com duas raias (*swimlanes*) [OMG, 2011]. A raia Aplicação detalha o algoritmo do módulo criptográfico, que é um subsistema do mecanismo que estará embutido na aplicação. Já a raia MeSAD detalha o algoritmo do mecanismo, o qual deve estar instalado no DM.

A Figura 3 mostra ainda a atividade “Tratar confidencialidade da mensagem”**,** iniciada pela aplicação (raia Aplicação). A partir dela, as atividades do MeSAD interagem com o conjunto de atividades que controlam a confidencialidade da aplicação. Este conjunto de atividades está representado por “Controlar Confidencialidade da Aplicação”, com suas sub-atividades. Estas sub-atividades fazem o controle das mensagens a serem transmitidas ou recebidas com o uso de criptografia, são elas: o uso de um protocolo para o tráfego dessas mensagens, e a divulgação das chaves e do algoritmo simétrico definido. Elas são responsáveis também por identificar e atualizar, em tempo de execução, os parâmetros que caracterizam os contextos dos DMs que participam do canal de comunicação. Essas informações servem para o estabelecimento de valores que facilitarão a escolha de um algoritmo que atenda aos requisitos de segurança nos contextos identificados. Por fim, as sub-atividades utilizarão o algoritmo indicado para garantia de confidencialidade, seja na transmissão (cifrando a mensagem) ou na recepção (decifrando a mensagem).

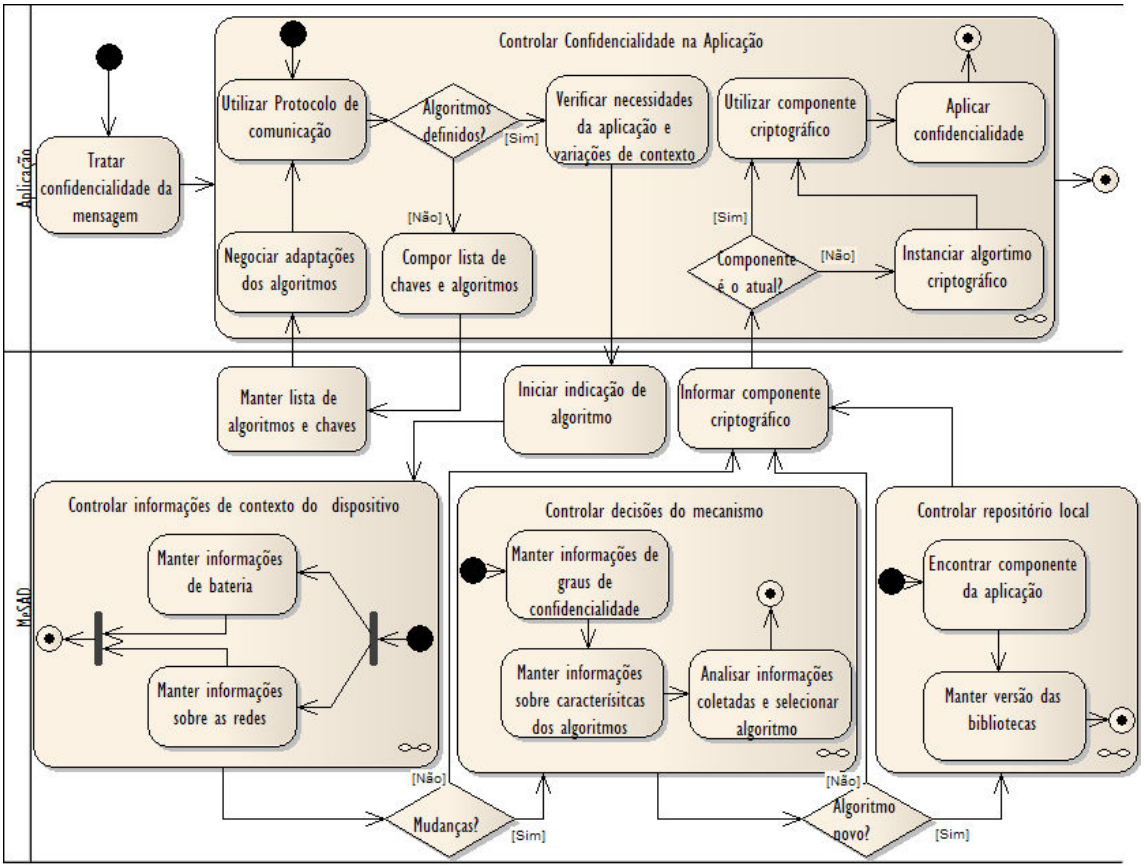


Figura 3 – Os algoritmos de funcionamento do MeSAD

A raia MeSAD apresentada na Figura 3 detalha o funcionamento do mecanismo, que inicialmente realiza a configuração das chaves dos algoritmos de criptografia que a aplicação utiliza e as divulga para a aplicação. Em seguida, quando a aplicação solicita a indicação do algoritmo adequado para uso, as atividades "Controlar informações de contexto do dispositivo" e "Controlar decisões do mecanismo", verificam informações atuais sobre o contexto, analisam as necessidades da aplicação e as características da lista de algoritmos, e seleciona o algoritmo adequado. A atividade "Controlar repositório local" recebe a informação do algoritmo escolhido e o identifica nas bibliotecas de segurança disponíveis para que a aplicação possa utilizá-lo.

Para proporcionar segurança adaptativa e individualizada por aplicação, o MeSAD faz uso de informações de configuração da aplicação criadas em tempo de projeto e informações de contexto capturadas em tempo de execução. Estas informações (detalhadas na Seção 4.3) são identificadas basicamente como:

- Graus de confidencialidade, que podem ser da aplicação ou dos algoritmos criptográficos, fornecidos na fase de projeto;

- Níveis de confiança das redes, os quais refletem o nível de confiança que o usuário deposita nas redes que ele utiliza e que são armazenados em tempo de execução para geração de um histórico para consultas posteriores;
- Níveis de consumo de energia proporcionados pelos algoritmos criptográficos, os quais serão fornecidos também na fase de projeto da aplicação; e
- Nível de carga da bateria, que será verificado em tempo de execução.

As informações de contexto obtidas em tempo de execução, juntamente com as informações geradas em tempo de projeto, irão influenciar na seleção do algoritmo de criptografia a ser utilizado pela aplicação.

O MeSAD trabalha com fraco acoplamento e seu funcionamento é apresentado na Figura 4, que mostra um repositório local contendo os componentes criptográficos da biblioteca BC que serão utilizados pelas aplicações instaladas no DM. Esses componentes implementam a interface IBC o que impõe às aplicações que também utilizem esta interface.

A aplicação, representada na figura pelo pacote à esquerda, utiliza o módulo UseCrypto, existente em seu Módulo Criptográfico local e responsável por tratar a criptografia e decifragem das informações. Este módulo possui um componente que implementa a interface da biblioteca de segurança adotada, representada por IBC. Ele consulta o módulo principal MeSAD, quando necessita utilizar criptografia, e envia parâmetros específicos da aplicação. O mecanismo analisa as informações de contexto e de configuração, aplica suas regras de decisão e acessa o seu Repositório Local para identificar a localização da biblioteca específica do componente identificado. Em seguida, fornece este componente para que UseCrypto o utilize.

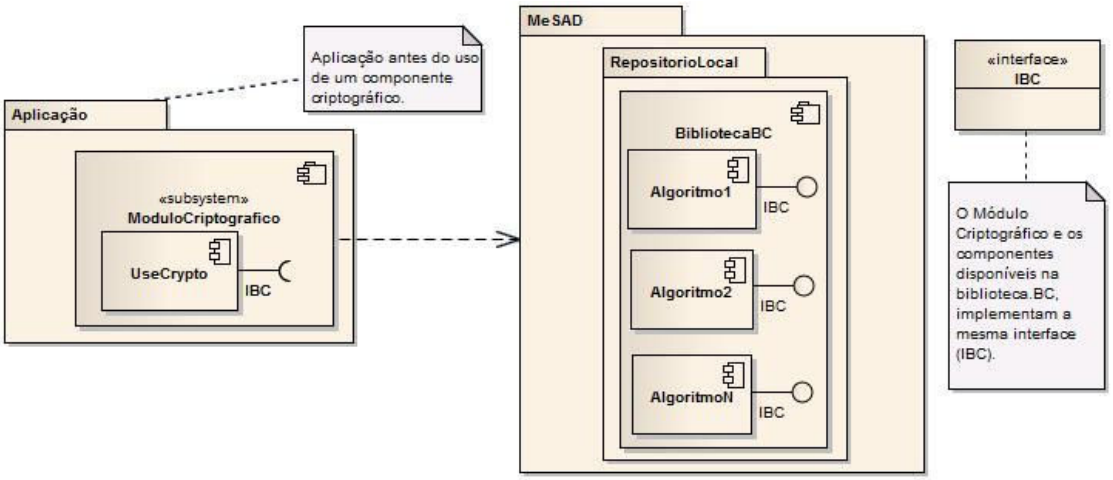


Figura 4 – Uma aplicação antes de usar um algoritmo criptográfico selecionado pelo MeSAD

A Figura 5 mostra a aplicação após interagir com o MeSAD, a qual apresenta seu componente UseCrypto utilizando o algoritmo 2. Pode-se notar que o componente UseCripto implementa a interface IBC (interface da biblioteca BC vista na Figura 4). O MeSAD escolhe um componente da biblioteca BC que implementa a interface IBC, provendo compatibilidade com UseCrypto. Este componente pode, em tempo de execução, ser substituído de acordo com a necessidade da aplicação e com o contexto identificado no momento do uso.

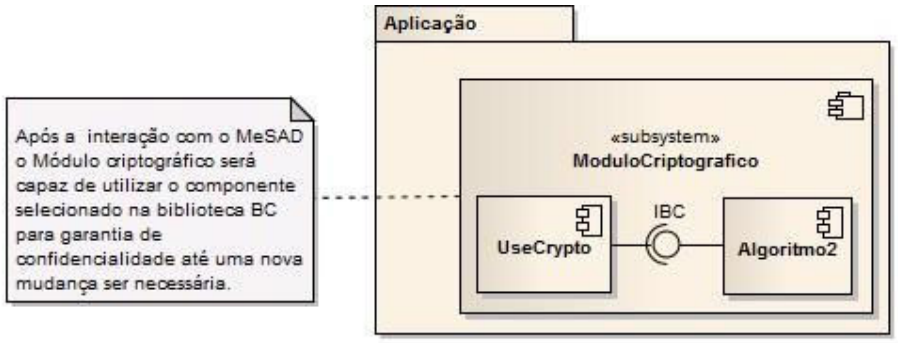


Figura 5 – Uma aplicação após receber um algoritmo selecionado pelo MeSAD

4.2 A Arquitetura do MeSAD

A arquitetura do MeSAD é focada no uso de componentes para prover adaptação dinâmica, pois eles podem ser adicionados, substituídos e reconfigurados em tempo de execução, permitindo adaptação às novas necessidades [ROCHA 2007 apud McKinley 2004].

O MeSAD é instalado no DM e interage com os módulos criptográficos (subsistemas) das aplicações que o adotam, sendo assim o mecanismo é dividido em duas partes: a solicitante e a fornecedora (destacadas por "Módulo Criptográfico" e "MeSAD" nas figuras 4, 5 e 6, e detalhadas na Figura 3 através das raias "Aplicação" e "MeSAD", respectivamente).

A Figura 6 mostra o diagrama de pacotes do mecanismo proposto. Pode-se observar nesta figura a existência de um Módulo Criptográfico na representação da Aplicação. Este módulo é adicionado, em tempo de construção, às aplicações que irão utilizar o mecanismo. Ele também é o responsável por acionar o MeSAD, provendo informações sobre sua versão e sobre a configuração da aplicação. A partir desta comunicação, torna-se possível o estabelecimento do componente certo para tratar a criptografia.

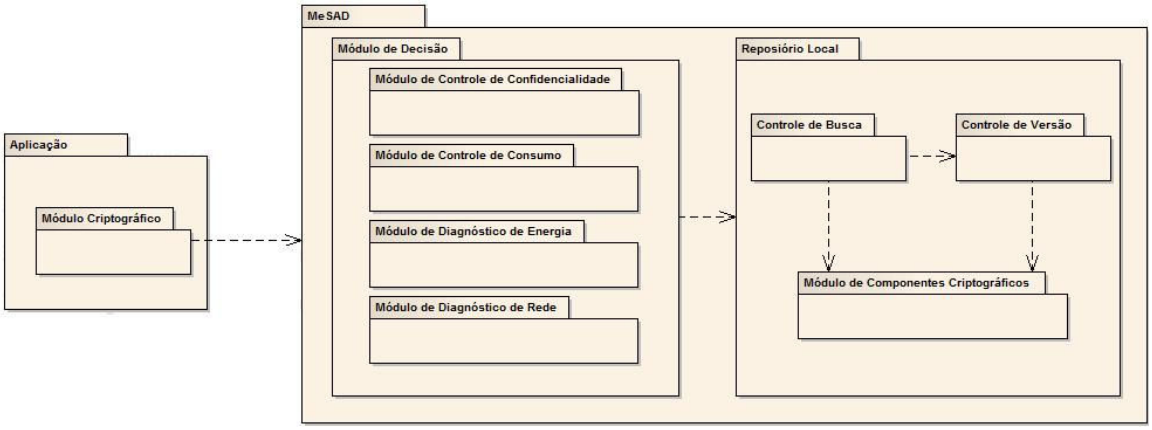


Figura 6 – Diagrama de pacotes do MeSAD

O pacote MeSAD, visto na Figura 6, representa a maior porção do mecanismo instalado no DM e é responsável por atender às requisições de decisão e pelo fornecimento de componentes criptográficos para as aplicações. Ele aciona o Módulo de Decisão, ao receber as informações sobre a aplicação que requer criptografia, e aguarda sua resposta para suprir o componente adequado para prover confidencialidade para a aplicação.

O Módulo de Decisão, visto na Figura 6, utiliza seus mecanismos (representados por seus módulos internos) para capturar as informações necessárias para a tomada de decisão sobre qual algoritmo de criptografia é o mais adequado para a situação verificada e, em seguida, consulta o Repositório Local para encontrar o componente que o represente. Este último módulo analisa a informação recebida sobre a versão do componente MeSAD da aplicação e verifica, no Controle de versão, o local de busca da versão do componente para, finalmente, suprir a aplicação.

A decisão do algoritmo a ser fornecido para garantia de confidencialidade acontece com a utilização de informações geradas em tempo de projeto e em tempo de execução. Essas informações estão detalhadas na Seção 4.3.

O funcionamento do MeSAD está modelado no diagrama de sequência da Figura 7. Nesta figura, estão representados CtrlCripto e UseCripto que implementam as interfaces do MeSAD e são partes do Módulo Criptográfico presente em cada aplicação. O CtrlCripto controla o acesso ao MeSAD para conseguir o componente de criptografia mais adequado para a aplicação, de acordo com o contexto. Ele fornece este componente ao UseCripto que coloca em prática a criptografia e decifragem das informações a serem trafegadas. Ele também verifica, em intervalos regulares, se o componente precisa ser modificado e, em caso positivo, fornece o novo componente para UseCripto. Esse processo está representado na Figura 7 pelo laço e pela condição de mudança do componente. Na figura pode-se notar toda a

interação do Módulo de Decisão com seus módulos internos, que representam as atividades do mecanismo (graus de confidencialidade, índices de consumo, análise de bateria e análise de rede) e também o acesso ao Repositório Local onde será identificado o componente de acordo com as necessidades e configurações da aplicação.

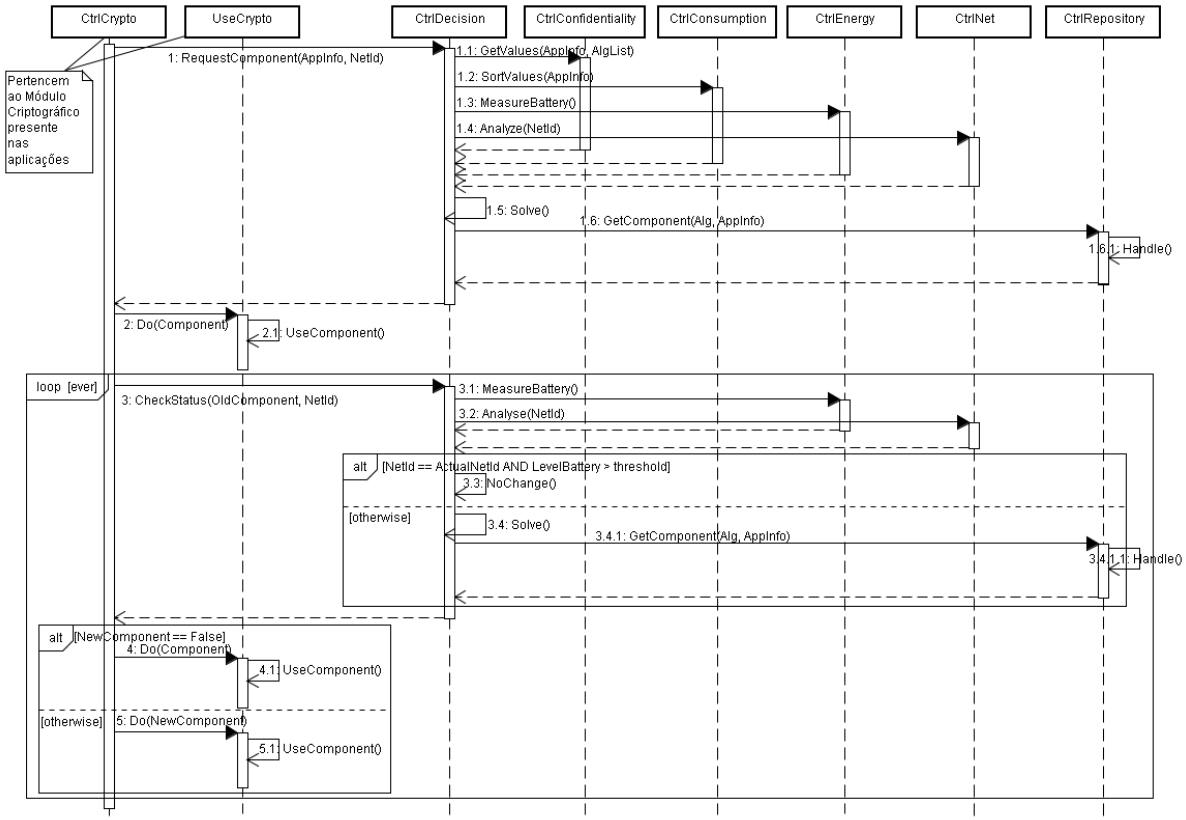


Figura 7 – Diagrama de Sequência Simplificado do MeSAD

Quando uma aplicação necessita criptografar os dados, ela repassa ao mecanismo informações da aplicação e a identificação da rede que é responsável pelo tráfego. Neste momento, o mecanismo acessa a classificação de confiança desta rede para, então, associá-la ao(s) algoritmo(s) que atende(m) ao grau de confidencialidade deste tipo de rede. Com o algoritmo criptográfico definido, o mecanismo o utiliza para realizar a criptografia dos dados. Este algoritmo é utilizado pela aplicação até que o mecanismo identifique a necessidade de uma nova adaptação. Esta mudança de algoritmo ocorre caso o nível de bateria entre em estado crítico, ou a rede utilizada pelo usuário mude, ou ainda, caso a aplicação solicite a mudança. As novas configurações são divulgadas pelo protocolo da aplicação.

A parte do MeSAD que permanece na aplicação e trata a confidencialidade das mensagens está descrita no diagrama de sequência da Figura 8, o qual mostra a interação do Módulo Criptográfico da aplicação com os outros módulos. A aplicação interage com

CtrlCrypto passando o texto para que ele promova o tratamento (cifragem/decifragem). Informações de configuração desta aplicação são colhidas e enviadas ao MeSAD do DM. Em seguida, informações como chaves, algoritmo inicial e parâmetros para mudanças posteriores são protegidos com criptografia assimétrica, e um protocolo (CtrlProtocolo) é utilizado na comunicação entre as partes envolvidas.

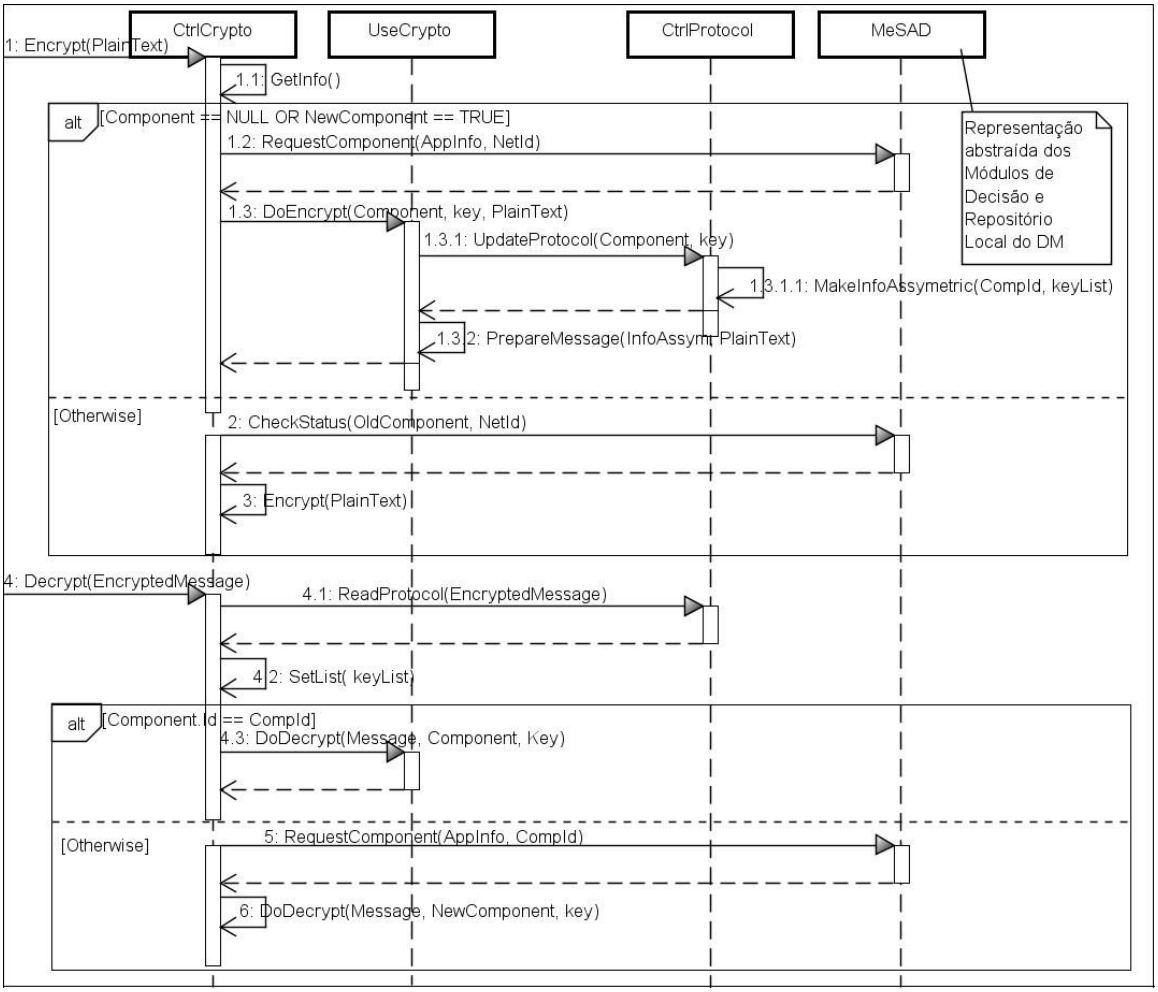


Figura 8 – Diagrama de sequência da parte do MeSAD presente na aplicação

Após a definição dos parâmetros iniciais, as mensagens posteriores utilizam o algoritmo simétrico estabelecido para implementar a confidencialidade no tráfego das informações. Quando é necessário realizar uma nova adaptação, esta informação é adicionada na última mensagem criptografada com o algoritmo em uso, com a intenção de divulgar o próximo algoritmo a ser utilizado nas próximas mensagens trocadas.

Quanto à decifragem de uma mensagem recebida, a Figura 8 mostra a interação do protocolo para verificação da lista de chaves e do componente em uso na criptografia da

mensagem recebida. Caso o algoritmo já esteja em uso, não é necessário novo fornecimento. Caso contrário, o MeSAD passa a utilizar o novo algoritmo.

Para o funcionamento do mecanismo é necessária a inserção do subsistema na aplicação a ser criada. Essa porção de código a ser inserida na aplicação facilitará o uso do MeSAD e será realizada com a utilização de uma ferramenta de apoio (4MeSAD), a qual será descrita no próximo capítulo.

A adoção do MeSAD para a criação de aplicações para DMs adiciona a preocupação com segurança nos processos de criação de *software*, tão necessária, porém tão pouco utilizada [ROCHA, 2010] [CHEN, 2010] [TADDEO, 2010].

A arquitetura do MeSAD facilita o processo de evolução da confidencialidade necessária para a aplicação, pois, para isso, basta que sejam definidos os novos graus de confidencialidade e que se gere um arquivo de configuração para ser substituído. Essa configuração dos graus de confidencialidade dos algoritmos, assim como a identificação dos índices de consumo de bateria de cada algoritmo criptográfico, são informações que podem ser reutilizadas em todas as aplicações que o engenheiro de software vier criar. As atividades envolvidas na captura destas informações estão detalhadas na próxima seção.

4.3 As Atividades Colaborativas do Mecanismo

O funcionamento do mecanismo está atrelado às atividades que colaboram entre si, adquirindo os valores de entradas específicas para atingir o propósito do MeSAD. As informações fornecidas pelas atividades servem de base para as tomadas de decisão quanto ao algoritmo criptográfico a ser utilizado na provisão de criptografia das mensagens provenientes das aplicações.

As atividades estão representadas na Figura 9. Uma delas é vista na figura com a descrição “Graus de Confidencialidade” (detalhada na Seção 4.3.1), a qual define os níveis de confidencialidade fornecidos pelos algoritmos criptográficos e as necessidades de segurança da aplicação. A próxima atividade ilustrada na Figura 9 é “Índices de Consumo”, que identifica e disponibiliza os índices de consumo de energia proporcionado pelos algoritmos criptográficos (detalhada na Seção 4.3.2). A Figura 9 também mostra atividades de controle de informações de contexto como a rede em uso pela aplicação (“Análise da Rede” que é

detalhada na Seção 4.3.1) e o controle do nível residual da energia da bateria do DM ("Análise da Bateria", detalhada na Seção 4.3.2).

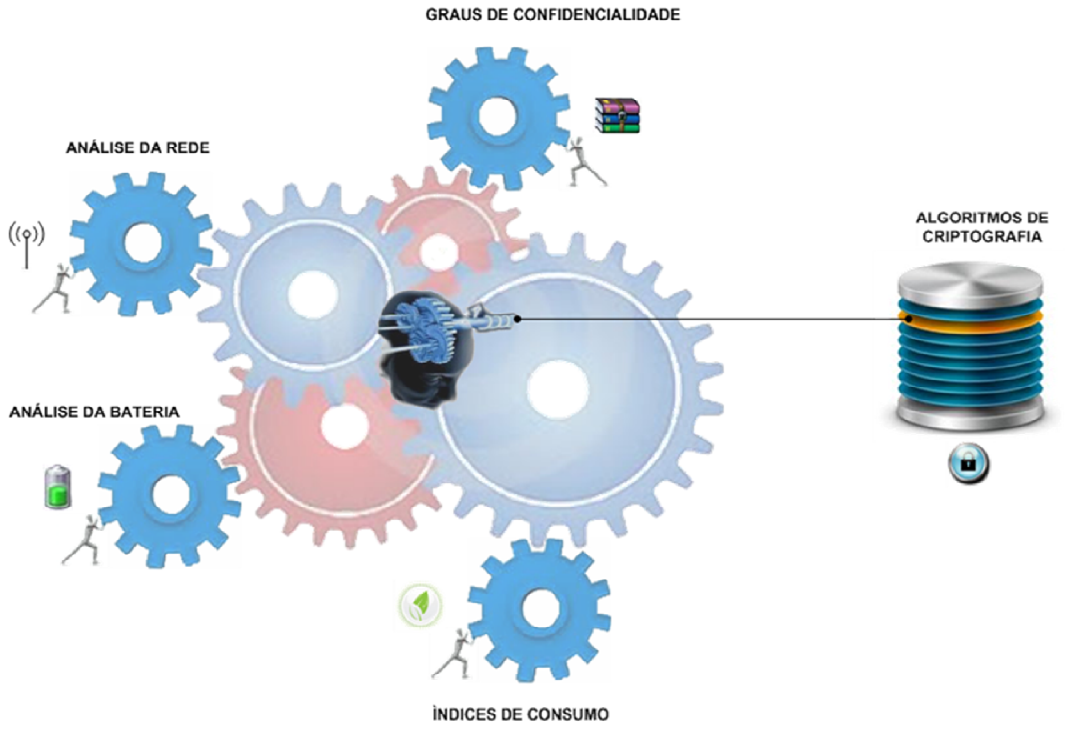


Figura 9 – O MeSAD e suas atividades colaborativas

A atuação do mecanismo proposto está ilustrada na Figura 9, a qual expõe as atividades chaves que atuam em conjunto e de forma a proporcionar o resultado esperado. Esse resultado é a decisão sobre o algoritmo criptográfico adequado para a provisão de criptografia no contexto identificado. As atividades de identificação e controle dos graus de confidencialidade, criadas em tempo de projeto e consultadas em tempo de execução, auxiliam a tomada de decisão central indicando um conjunto de algoritmos que atendem a necessidade de segurança da aplicação, também criada em tempo de projeto. A análise da rede em uso refina a lista dos algoritmos selecionados, separando aqueles que oferecem a proteção mais adequada para atender às expectativas do usuário. A análise do nível residual da energia da bateria e os índices de consumo proporcionados pelos algoritmos criptográficos, formam um conjunto de informações que influenciam na escolha do algoritmo criptográfico. A imagem central da figura que representa o módulo de decisão do MeSAD, o qual está selecionando um algoritmo de criptografia, entre vários existentes. A seleção é realizada com base nas informações fornecidas pelas atividades colaborativas.

Cada uma dessas atividades colaborativas ilustradas na Figura 9 é detalhada nas próximas subseções.

4.3.1 Graus de Confidencialidade

Como informado nas seções 4.1 e 4.2, existem informações que deverão ser geradas em tempo de projeto pelo engenheiro de *software* utilizando a ferramenta de apoio (mais detalhes no Capítulo 5). Entre essas informações, estão a definição do grau de confidencialidade da aplicação que será criada, de acordo com seus requisitos de segurança, e a definição dos graus de confidencialidade dos algoritmos de criptografia, contidos nas bibliotecas de segurança. Esses dois conceitos estão descritos nesta subseção, logo após a apresentação da metodologia utilizada para a definição dos valores dos graus de confidencialidade.

A seção 3.2 introduziu o termo Grau de Confidencialidade, apontando as variáveis que influenciam a composição deste valor e as complexidades envolvidas na utilização de um modelo que possa ser utilizado na prática.

O modo de classificação da confidencialidade considerado nesta dissertação foi definido com base em outros trabalhos que consideram níveis de confidencialidade proporcionada por algoritmos de criptografia, principalmente pelo modelo de avaliação proposto por Jianyoung Chen [CHEN, 2010]. O trabalho de Chen (2010), que trata de segurança em rede sem fio, considera como fator de influência central nos seus cálculos, o tamanho da chave, que é atrelado a uma definição de peso que representa a complexidade computacional oferecida pelo algoritmo, pois o autor afirma que avaliar a oferta de segurança proporcionada pelos algoritmos é algo altamente complexo. Adicionalmente, este nível de segurança ofertado pelo algoritmo pode mudar em curto tempo (e.g. a descoberta de uma fraqueza no algoritmo), inviabilizando a existência ou utilização de um modelo mais confiável. Ademais, geralmente a escolha de algoritmos criptográficos, com base em sua robustez, considera apenas o tamanho da chave, o que reforça o modelo de Chen (2010). Isso ocorre porque se leva em conta a dificuldade oferecida pelo algoritmo para que seja superado por um ataque de força bruta.

O nível de garantia de segurança proporcionada pela criptografia pode ser determinado pelo algoritmo criptográfico e pelo tamanho de chave utilizada (CHEN, 2010). Assim, neste trabalho, a classificação dos níveis de confidencialidade de cada algoritmo chamada aqui de graus de confidencialidade, foi definida com base no trabalho de Chen (2010).

Analisando a possibilidade de tentativas de quebra de segurança por força bruta, e considerando que um comprimento de chave com 8 bits ($2^8 = 256$) possui 256 possibilidades de definição, e agora substituindo o expoente por uma variável k de tamanho igual à chave utilizada (k_{tam}), pode-se generalizar esta relação como:

$$2^{k_{tam}} \qquad \textbf{(1)}$$

Com isso, a expressão **(1)** representa o número de tentativas para obtenção da chave de criptografia e, assim definida, pode se concluir que, quanto maior for o k_{tam} maior também será o nível de segurança proporcionado por um algoritmo.

Assim, para classificar o grau de proteção contra ataques de força bruta, assume-se que este valor seja igual a 1 quando se utiliza o menor tamanho de chave conhecido (k_{min}). Então, o grau de confidencialidade da criptografia (l_k), considerando a proteção contra ataque de força bruta, pode ser definido como:

$$\mathbf{l_k} = 2^{k_{tam}/k_{min}} - 1 \qquad \textbf{(2)}$$

Para definir a equação final do grau de confidencialidade (l_k), deve-se agora considerar a complexidade da criptografia oferecida pelos algoritmos criptográficos.

Na literatura não foi encontrada uma definição sobre uma classificação de algoritmos criptográficos de acordo com sua complexidade e importância no processo de adequação da criptografia e da garantia de confidencialidade. Adicionalmente, é relativamente fácil encontrar trabalhos com a clara preocupação, por vezes explicitada, em não fazer uso desta classificação. Diante desta dificuldade em classificar tais algoritmos, suas diferenças são consideradas, neste trabalho de dissertação, pela atribuição de um peso (p), com valores de 0 até 1, na equação **(2).**

Desta forma, tem-se agora:

$$\mathbf{l_k} = (2^{k_{tam}/k_{min}} - 1) * p \qquad \textbf{(3)}$$

A atribuição do peso (p) facilita a adoção do mecanismo por especialistas em segurança ou engenheiros que queiram modificar as configurações originais de acordo com seus conhecimentos e suas necessidades, além de oferecer a facilidade de novas classificações em futuras versões do mecanismo. Exemplificando esta maleabilidade com o uso de um peso, se for definido um peso de maior valor para o algoritmo AES de uma biblioteca que o tem como seu algoritmo mais significativo e, posteriormente, for disponibilizado um algoritmo mais seguro nesta biblioteca, os pesos podem ser redefinidos para acomodar esta mudança.

Após a definição dos graus de confidencialidade para cada algoritmo criptográfico de determinada biblioteca de segurança, é disponibilizada uma lista de algoritmos com seus

valores de confidencialidade respectivos, a qual pode ser reutilizada em todas as próximas aplicações que vierem a ser construídas pelo engenheiro de software.

Os graus de confidencialidade considerados neste trabalho são:

a) Grau de confidencialidade dos algoritmos. O engenheiro de software ou especialista em segurança deve ser capaz de classificar uma lista de algoritmos de criptografia que é utilizada na criação de suas aplicações. Caso ele prefira, pode utilizar a classificação utilizada neste trabalho, a qual é a classificação padrão, configurada e disponibilizada para reutilização. Esta classificação deve ser realizada com a utilização da 4MeSAD, ferramenta de suporte ao uso do MeSAD descrita no próximo capítulo desta dissertação. A lista dos algoritmos pode ser atualizada a qualquer tempo, de acordo com as necessidades do engenheiro de software, e serve para qualquer projeto de software que vier a utilizar o MeSAD. Os algoritmos normalmente são encontrados em bibliotecas de segurança e, para esta classificação, o ideal é que seja utilizada uma metodologia científica ou uma ferramenta construída com essa consideração. Conhecer o comportamento dos algoritmos para facilita a identificação de parâmetros que influenciam na execução da aplicação e no fornecimento de segurança. Neste trabalho a classificação dos algoritmos foi realizada com base em conhecimentos científicos na área de segurança [CHEN, 2010], e através do uso da ferramenta 4MeSAD. Ela oferece pontos de maleabilidade para permitir que aqueles usuários que detiverem maiores conhecimentos e preferirem criar as suas próprias configurações, possam fazê-lo.

b) Grau de confidencialidade da aplicação. Com relação à definição do grau de confidencialidade da aplicação que está sendo criada, o engenheiro de *software* deve identificar a necessidade de segurança que sua aplicação requer e classificá-la nas opções disponíveis na ferramenta de suporte ao MeSAD.

Seguindo o modelo de classificação comumente adotado em aplicações que utilizam classificações de níveis de segurança requerido e o trabalho adotado para estas definições de classificação de segurança já mencionado anteriormente (CHEN, 2010), três níveis estão disponíveis na ferramenta, são eles: baixo, médio e alto. Estes níveis definem o grau de confidencialidade da aplicação e estão relacionados com os algoritmos classificados pelo engenheiro de *software.* O engenheiro ao classificar cada algoritmo na 4MeSAD (apresentada no Capítulo 5) visualiza, dinamicamente, qual dos 3 níveis está atrelada a classificação do algoritmo e ao final é apresentada a lista de todos os algoritmos que compõem cada nível. Conhecendo os algoritmos que formam o conjunto de cada um dos 3 níveis, o engenheiro pode inferir qual deles será atribuído à aplicação que está sendo criada.

Ele pode definir uma política de uso para facilitar esta seleção na criação das próximas aplicações (por exemplo, sabendo que o nível “alto” só utiliza algoritmos altamente complexos, que diminuem o desempenho da aplicação, mas garantem a confidencialidade, o engenheiro pode determinar seu uso apenas em aplicações que utilizem transações monetárias). Assim o requisito de segurança da aplicação, conhecidos pelo engenheiro, define a seleção do grau de confidencialidade desta aplicação.

Esta classificação de grau de confidencialidade da aplicação é necessária a cada novo cadastro de aplicação que o engenheiro realizar, pois cada aplicação possui seus próprios requisitos de segurança. Isso permite a realização de uma filtragem dos algoritmos criptográficos no momento da escolha da criptografia a ser utilizada pela aplicação, em tempo de execução.

Estas definições e classificações utilizando a ferramenta 4MeSAD formam a primeira entrada para o mecanismo proposto e essencial para sua adoção.

É importante deixar claro que estes não são fatores absolutos para definição dos graus de confidencialidade, são índices relativos e que podem ser personalizados.

Neste trabalho, para efeito de experimentação definiu-se estas classificações com valores de limiares baseados no trabalho de Chen (2010).

4.3.2 Índice de Consumo do Algoritmo de Criptografia

Para gerar o índice de consumo de cada algoritmo de criptografia o engenheiro deve realizar o *download* de uma aplicação a partir da 4MeSAD e executá-la nos DMs (mais detalhes no Capítulo 5). A aplicação realiza testes de *benchmarking* que possibilitam a criação da segunda lista de informações que servirá de entrada para o mecanismo de adaptação da segurança.

A 4MeSAD foi criada para ser uma ferramenta de coleta e análise de informações relacionadas com a execução de diversos algoritmos criptográficos em DMs. Trata-se de um avaliador de desempenho de algoritmos criptográficos quanto ao consumo de energia da bateria, tempo de execução e uso de memória. Para extrair esta informação de índices de consumo um aplicativo é instalado nos DMs a serem testados. Os procedimentos para o início dos testes estão descritos na ferramenta e devem ser seguidos rigorosamente, para se obter precisão nas informações geradas. Alguns parâmetros devem ser informados pelo usuário e a aplicação deve, então, ser executada. Os resultados dos testes vão sendo produzidos e

guardados localmente no DM e, ao final, são enviados para a ferramenta localizada em um servidor *web*. Tais dados são mantidos no servidor da ferramenta e utilizados para a criação de entradas de configuração do mecanismo a ser utilizado.

O engenheiro de *software* deve produzir esses testes de *benchmarking* para avaliar o impacto do uso de cada algoritmo criptográfico nos DMs que ele espera que a aplicação esteja em uso. Assim, uma lista de índices de consumo oferecido por cada algoritmo é produzida, possibilitando a criação de um valor médio de consumo para determinada categoria de DMs.

Esta é a segunda entrada para o mecanismo proposto e é importante para refinar a seleção de um algoritmo de criptografia considerando preocupações provenientes da *Green Computing*, quanto ao uso eficiente dos recursos computacionais com o intuito de aumentar seu tempo de vida.

4.3.3 Análise de Confiança das Redes

Considerando que confiança é uma sensação própria do indivíduo, e que é o usuário do DM que interage com os elementos de contexto que propiciam ou não essa sensação, este trabalho considera a participação do usuário para a definição de um nível de confiança que indique o quanto ele confia na rede que o DM faz uso para o tráfego de suas informações. Assim sendo, o MeSAD dá suporte para o usuário analisar suas redes e atribuir um nível de confiança para elas, pois este tipo de informação é diretamente dependente das percepções do usuário.

Esta informação possui duas formas de ser identificada, e estas duas formas se autocomplementam para estabelecer qual rede é mais ou menos confiável. Uma delas é através de uma análise que é feita em tempo de execução, no momento do uso da aplicação, na qual a rede em atividade é avaliada para identificar os protocolos que incrementam a segurança no tráfego aéreo das informações, caso existam. Esta análise pode informar se a rede utiliza um protocolo seguro de enlace de dados ou não e, de acordo com esta informação, um grau de confiança desta rede pode ser sugerido ao usuário para que ela seja classificada.

Como a identificação das características das redes não é suficiente para a definição de uma escala que provém de uma sensação própria do indivíduo, por mais leigo que ele possa ser quanto ao assunto, o envolvimento do usuário nesta etapa é fundamental.

Exemplificando, o usuário pode estar utilizando uma rede que possui os mais avançados protocolos de segurança, no entanto a senha de acesso a esta rede é de conhecimento público. O MeSAD é capaz de identificar que a rede utiliza um protocolo forte, porém a informação de que a senha é pública é o tipo de informação que apenas o usuário pode informar e que faz total diferença na avaliação de confiança desta rede. Assim, outra forma de identificar o grau de confiança da rede é a análise direta do usuário quanto à sua confiança na rede que ele está utilizando. Esta avaliação é importante porque apenas o usuário pode informar o quanto ele confia ou não em determinadas redes e, de acordo com as características desta rede, este usuário pode avaliá-la e definir o seu nível de confiança. Um exemplo de como esta informação é restrita ao conhecimento do usuário seria uma rede existente num ambiente remoto ou isolado (sua chácara com acesso à internet fornecido por um *access point* de fraca segurança em seu enlace aéreo), no qual o usuário sabe que não existiria um agente malicioso nas proximidades capaz de capturar a sua transmissão. Outro exemplo típico seria o usuário acessando uma rede a partir de um aeroporto em um *access point* com tecnologia mais moderna de proteção no seu enlace, porém com chave de acesso compartilhada por várias pessoas. Este usuário certamente não sentiria confiança em utilizar esta rede. Nos dois cenários, a aplicação requer níveis diferentes de confidencialidade.

Essa interação do usuário com o MeSAD para a avaliação de suas redes produz um histórico que é guardado e sempre que ele for utilizar uma rede já avaliada, essa informação será reutilizada. Por outro lado, quando uma nova rede for detectada, o usuário fará a sua avaliação. Esta classificação das redes em uso é pessoal e é realizada de acordo com a percepção do usuário quanto à confiança que ele deposita nesta rede.

Todas as avaliações podem ser refeitas ou apagadas pelo usuário do DM, bastando que ele acesse as configurações de redes do MeSAD.

Para acessar as configurações do MeSAD, o usuário do DM deve acessar o menu do dispositivo e selecionar “MeSAD”. Em seguida o usuário deve escolher os dados de configuração que pretende acessar e realizar suas alterações.

Esta é a terceira entrada para o mecanismo proposto e é essencial para sua adoção.

4.3.4 Análise do Nível da Bateria

Como o MeSAD atua em tempo de execução de acordo com informações provenientes do contexto e possui a preocupação com o consumo eficaz dos recursos computacionais, como a bateria, nada mais natural que o nível de energia disponível no DM seja monitorado e utilizado para refinar a escolha do algoritmo criptográfico para a garantia de confidencialidade.

Sendo assim, a captura do nível de bateria do dispositivo móvel torna possível o acompanhamento do consumo e pode incrementar a capacidade de adaptação de uma aplicação mais alinhada com a preocupação em preservar este tipo de recurso.

Esta informação é capturada em determinados instantes, quando a adaptação estiver sendo considerada, e pode influenciar na escolha de um algoritmo que consome menos de acordo com o nível da bateria.

Exemplificando, podemos considerar que determinada aplicação está sendo executada em um contexto que foi avaliado e o grau de confidencialidade da aplicação selecionou mais de um algoritmo que oferecem o grau de confidencialidade pertinente a este contexto. Neste caso, o algoritmo que consome menos recurso de bateria poderá ser selecionado para uso sem prejudicar o grau de confidencialidade esperado pela aplicação. Uma situação mais complexa seria uma aplicação sendo executada em determinado momento onde o nível de bateria disponível para uso esteja crítico ou muito baixo, de modo a tornar duvidosa a sua execução. Neste caso a aplicação poderia solicitar a interferência do usuário para oferecer a possibilidade dele autorizar momentaneamente a utilização de um algoritmo menos complexo para garantir a utilização da aplicação, porém com um grau de confidencialidade mais baixo.

Esta é a quarta entrada para o mecanismo proposto adquirida apenas em tempo de execução e utiliza as informações sobre o índice de consumo de bateria dos algoritmos, adquiridas com o uso da ferramenta de suporte ao MeSAD detalhada no próximo capítulo.

4.3.5 Determinação do algoritmo

As informações geradas pelas atividades colaborativas descritas neste capítulo servem para a determinação do algoritmo mais apropriado para utilização por parte da aplicação. Elas servem também para a divulgação de parâmetros de negociação no estabelecimento do algoritmo quando mais de um DM participar da comunicação. Algumas informações sobre a forma utilizada para a determinação do algoritmo são apresentadas a seguir.

A influência de determinada atividade colaborativa na escolha do algoritmo pode alterar de acordo com o contexto. Exemplificando, se o DM tiver atingido o nível crítico de bateria, a atividade de índices de consumo dos algoritmos passa a ter prioridade maior, pois o DM necessita poupar o consumo de bateria. Contudo, o grau de confidencialidade da aplicação continuará sendo respeitado, pois o algoritmo econômico a ser escolhido deverá possuir um grau de confidencialidade maior ou igual ao valor mínimo de confidencialidade da aplicação.

Conforme já mencionado neste capítulo, quando a aplicação estiver utilizando uma rede considerada de baixa confiança, o MeSAD irá priorizar o uso de algoritmos com os maiores graus de confidencialidade. Nesta situação, acontecerão os seguintes casos:

- Caso o nível residual de bateria do DM possua um valor acima de um determinado *threshold* (valor limiar do nível de bateria, a partir do qual os algoritmos são substituídos dinamicamente) configurado no DM, o algoritmo selecionado será sempre o de maior grau de confidencialidade.
- Quando o *threshold* é atingido o MeSAD passa a considerar a mudança do algoritmo em uso. O novo algoritmo deverá possuir um grau de confidencialidade com valor acima da média do valor entre os algoritmos disponíveis para uso. A cada intervalo de consumo pré-definido (pode ser ajustado pelo usuário) de bateria, o MeSAD realiza uma nova escolha.
- Quando o nível crítico de bateria é atingido o algoritmo escolhido é o de menor consumo entre os algoritmos que estão acima da média do valor de grau de confidencialidade dos algoritmos disponíveis.

Se a rede que a aplicação utiliza possuir uma classificação de confiança média, o MeSAD irá atuar da seguinte forma:

- Caso o nível residual de bateria não tenha atingido o *threshold,* o MeSAD escolhe um entre os algoritmos com grau de confidencialidade acima do

valor médio de grau de confidencialidade de forma aleatória, evitando o de maior grau e irá modificar esta escolha a cada intervalo de adaptação, até atingir o valor médio do grau de confidencialidade.

- Caso o *threshold* já tenha sido atingido, o MeSAD irá escolher o algoritmo de grau de confidencialidade igual ou imediatamente inferior ao valor médio de grau de confidencialidade. O mecanismo irá realizar uma nova escolha a cada intervalo de adaptação considerando as menores taxa de consumo, e entre os valores abaixo do valor médio de grau de confidencialidade, evitando o de menor valor.
- Quando o nível crítico for atingido o algoritmo escolhido será o de menor taxa de consumo.

Por fim, quando a rede possuir a confiança do usuário, ou seja, for considerada segura, o algoritmo escolhido é o de menor taxa de consumo para qualquer nível de bateria.

O conteúdo apresentado nesta dissertação evidenciou a necessidade de adaptação da confidencialidade de acordo com o contexto. A compreensão dessa necessidade juntamente com estudos científicos em segurança, possibilitou a definição das regras apresentadas. Assim, parâmetros que caracterizam os algoritmos são utilizados em conjunto com informações sobre o contexto para que o MeSAD possa atender as necessidades identificadas (e.g. uma rede avaliada como insegura deve utilizar um algoritmo com grau de confidencialidade forte, mesmo que a bateria esteja com nível baixo, pois a segurança precisa ser garantida). Com o conhecimento das necessidades de segurança nos contextos de uso do DM e com as determinações de oferta de segurança definidas pelos algoritmos, o MeSAD pode atuar de maneira transparente na definição do algoritmo mais adequado. As regras foram estabelecidas com base no objetivo deste trabalho, apresentado no Capítulo 1, e considerando as características dos algoritmos criptográficos, quanto ao seu grau de confidencialidade, o qual foi determinado utilizando a metodologia apresentada neste capítulo.

Com estas regras, o MeSAD preserva a confidencialidade da aplicação de acordo com a confiança do usuário quanto à rede em uso e de acordo com os requisitos de segurança da aplicação, definidos pelo engenheiro de software. Adicionalmente, diminui o consumo de bateria nos ambientes de menores riscos, preservando este recurso e aumentando sua disponibilidade no DM.

4.4 Conclusões

Como o objetivo do MeSAD é proporcionar maior segurança para a aplicação quando estiverem em uso em ambientes inseguros e diminuir a utilização dos recursos computacionais em ambientes mais seguros, o uso da adaptação dinâmica utilizando contexto mostra-se como a maneira ideal para atingir este objetivo. Assim, o MeSAD trata este problema extraindo informações do contexto e as usando para realizar a adaptação no provimento de confidencialidade para cada aplicação.

Para facilitar a utilização do MeSAD na criação de aplicações seguras, e com o intuito de popularizar o uso de segurança em DM, foi desenvolvida uma ferramenta de apoio, a qual é de fácil utilização. Esta ferramenta está descrita no próximo capítulo e será responsável por gerar os arquivos necessários para uso nas aplicações.

5 A FERRAMENTA 4MeSAD

Este capítulo apresenta a ferramenta de suporte ao uso do MeSAD. A visão geral e os algoritmos do mecanismo são apresentados na Seção 5.1. A arquitetura do mecanismo e o seu funcionamento são detalhados na Seção 5.2. Na Seção 5.3 estão descritas as atividades chaves que possibilitam a definição do algoritmo a ser utilizados pelas aplicações e, por fim, na Seção 5.4, este capítulo é concluído.

5.1 Visão Geral

A Ferramenta de suporte ao MeSAD, denominada de 4MeSAD[1], foi desenvolvida na plataforma Java EE[2] e oferece apoio para que os engenheiros de *software* criem aplicações para DMs adotando o MeSAD. Eles podem definir parâmetros quanto ao tratamento da confidencialidade das informações, cadastrar suas aplicações e configurar suas necessidades de segurança. A 4MeSAD permite ainda que se realizem testes para diagnosticar o impacto de consumo de recursos que cada algoritmo proporciona nos DMs nos quais se espera que as aplicações estejam sendo executadas. Assim, esta ferramenta possui 3 pontos principais que precisam ser esclarecidos: o primeiro diz respeito à coleta de dados dos algoritmos criptográficos ao serem executados nos DMs; o segundo é a avaliação desses dados coletados através de gráficos disponibilizados na 4MeSAD; e o terceiro é a configuração de parâmetros sobre esses algoritmos levando em consideração as informações coletadas e analisadas. Este último serve para a criação dos arquivos de configuração para as aplicações que utilizarão o MeSAD. Enquanto que os dois primeiros tratam-se de um *benchmarking* sobre os algoritmos criptográficos em uso nos DMs testados.

Como já mencionado, a 4MeSAD surgiu da necessidade de identificação de parâmetros que representassem o comportamento dos algoritmos criptográficos nos DMS. Um trabalho que foi utilizado como base para a construção da 4MeSAD foi a ferramenta

[1] 2011 4MeSAD - GREat – (http://www.great.ufc.br:8080/4mesad)

[2] Java Enterprise Edition. Copyright © 2010, Oracle Corporation and/or its affiliates (http://www.java.com)

PEARL (Performance Evaluator of CriptogRaphic aLgorithms for mobile device) [BRINGEL, 2004]. O trabalho de BRINGEL (2004) disponibilizou uma ferramenta que permite avaliar algoritmos de criptografia em DMs que utilizam a plataforma J2ME[3].

A PEARL [BRINGEL, 2004] não pôde ser utilizada nesta dissertação pelo fato da linguagem de programação adotada ser incompatível, porém suas primitivas de funcionamento serviram de inspiração para a construção da 4MeSAD. Essas primitivas são apresentadas na Figura 10 e são basicamente: a aplicação que realiza os diagnósticos dos algoritmos nos DMs, o armazenamento dos dados local e o posterior envio para o banco de dados da ferramenta, além da apresentação desses dados na ferramenta através de gráficos.

A Figura 10 mostra uma visão geral da 4MeSAD. Nesta figura estão sendo ilustradas as formas de interação do engenheiro de *software* (ou especialista em segurança) com a ferramenta. Para o engenheiro verificar o consumo de recursos dos DMs ele precisa realizar o *download* de um aplicativo de diagnósticos e executá-lo nos DMs. Este aplicativo é responsável por realizar testes de consumo de memória, de bateria e tempo de execução dos algoritmos. Ele armazena os dados localmente até a finalização das análises de todos os algoritmos criptográficos disponíveis, em seguida os envias para a 4MeSAD utilizando uma rede sem fio. Esses resultados são mantidos em um banco de dados e utilizados posteriormente para gerar gráficos de análises e também para gerar as configurações de confidencialidade das aplicações do engenheiro.

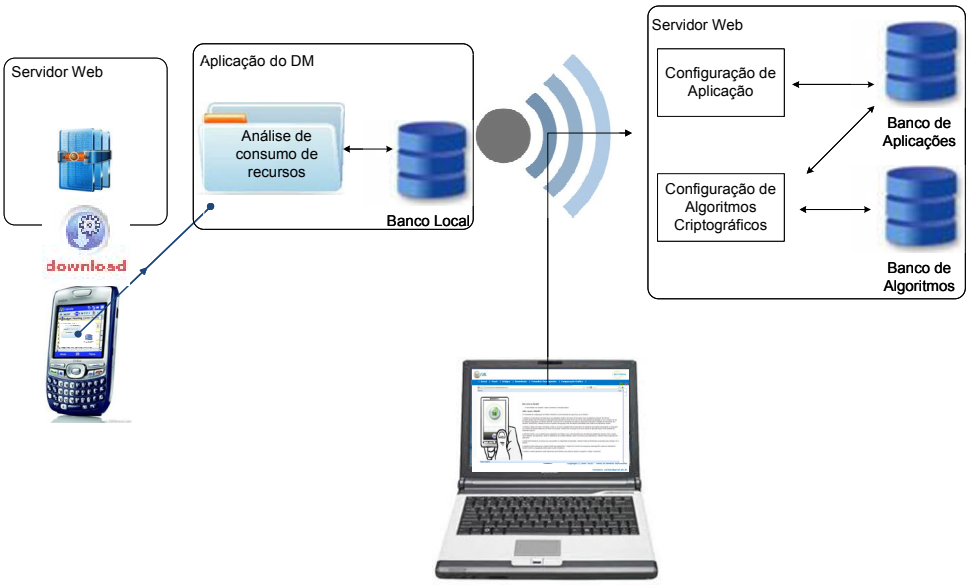


Figura 10 – Visão geral da 4MeSAD

[3] Java 2 Micro Edition. Copyright © 2010, Oracle Corporation and/or its affiliates (http://www.java.com)

Na Figura 10 também é mostrado o acesso à 4MeSAD que é realizado a partir de um equipamento utilizando um navegador de *internet*, no qual o engenheiro realiza um *login* para acessar os recursos da ferramenta. Esses recursos serão apresentados nas próximas seções.

5.2 Utilizando a ferramenta

Conforme mencionado anteriormente, a 4MeSAD oferece apoio ao uso do MeSAD e essa é apresentada nesta seção passo a passo. Este detalhamento foi criado com a intenção de orientar a utilização da 4MeSAD.

A 4MeSAD disponibiliza opções pré-configuradas para uso. Contudo, também é possível personalizar uma configuração de parâmetros própria do engenheiro ou especialista em segurança, de acordo com sua necessidade e seus conhecimentos, para adoção nas suas aplicações, ao invés de utilizar as pré-definidas. Para facilitar a compreensão sobre o uso da 4MeSAD será considerada a situação em que todo o processo de uso da ferramenta tem que ser seguido, desde a população com as coletas até a criação da aplicação.

a) Passo 1: Cadastros iniciais na 4MeSaD

Inicialmente o engenheiro de *software* precisa se cadastrar na ferramenta, informando seus dados pessoais e definindo um *login* e senha de acesso. Em seguida, ele precisa saber a arquitetura dos DMs que suas aplicações farão uso, e cadastrar os DMs indicando suas configurações técnicas (e.g. frequência do processador, modelo). Após cadastrar cada um dos DMs o engenheiro recebe um identificador desses DMs que precisa ser guardado para uso posterior.

b) Passo 2: Coleta dos dados no DM

Com os cadastros prontos e com os identificadores dos DMs é necessária a geração de informações sobre o comportamento dos algoritmos criptográficos ao serem executados nos DMs. Para isso, uma aplicação deve ser instalada nos DMs. Ela é disponibilizada na 4MeSAD para *download*. Após a instalação da aplicação no DM orientações sobre o manuseio do dispositivo durante a realização dos testes devem ser seguidas, sob a pena de não produzir informações confiáveis. Seguidas as instruções a aplicação é executada e o DM deve permanecer sem intervenções até o fim da coleta dos dados, que será indicada pela aplicação com a apresentação de um relatório.

Finalizada a coleta, os dados que permanecem armazenados no DM devem ser enviados para a 4MeSAD e isso é feito utilizando uma rede sem fio e preenchendo a identificação do usuário e do DM (identificadores fornecidos quando do cadastro na ferramenta) e clicando no botão de envio.

Esse passo deve ser repetido para cada DM cadastrado na 4MeSAD e serve para popular a ferramenta para usos futuros.

c) Passo 3: Análise dos dados gerados

Após os envios dos dados coletados nos DMs, o engenheiro deve acessar a 4MeSAD e verificar como os algoritmos se comportaram nos DMs. Para isso, são disponibilizados gráficos que apresentam as informações sobre tempo de execução dos algoritmos, quantidade de memória consumida, nível de consumo de bateria proporcionada e quantidade de repetições possíveis até o consumo de 1% da bateria. Esses gráficos devem ser analisados pelo engenheiro para auxiliarem na decisão sobre a classificação dos algoritmos que serão utilizados por suas aplicações.

d) Passo 4: Classificando os Graus de Confidencialidade

Após a análise dos dados gerados sobre os algoritmos nos DMs o engenheiro já deve saber como cada algoritmo se comportará quando em uso nas suas aplicações. Com base nesta informação e com base nas necessidades de segurança de suas aplicações o engenheiro deve criar uma classificação de grau de confidencialidade dos algoritmos definindo um peso para cada um deles. Conforme os pesos vão sendo atribuídos aos algoritmos seus graus de confidencialidade vão sendo atualizados e vai aparecendo, em tempo real, os subgrupos de algoritmos que compõem cada uma das classificações de nível de confidencialidade das aplicações descritas no Capítulo 4. Isso facilita a atribuição dos pesos, pois possibilita que o engenheiro verifique qual categoria de confidencialidade o algoritmo ocupou e permite a correção de maneira rápida.

Ao final, será possível visualizar todos os algoritmos separados por seus valores de grau de confidencialidade e organizados em 3 categorias: baixa, média e alta (conforme definição apresentada no Capítulo 4).

e) Passo 5: Criando um perfil de consumo

O engenheiro de *software* agora precisa criar um perfil de consumo dos algoritmos que será utilizado pelo MeSAD. Ele faz isso escolhendo quais dos DMs testados ele quer que este perfil seja considerado. Após essa escolha ele gera o perfil clicando em um botão. O perfil de consumo tem o intuito de representar o consumo proporcionado pelos algoritmos e serve para auxiliar a escolha feita pelo MeSAD. Ele é gerado calculando a média

de cada parâmetro avaliado nos testes nos DMs selecionados (a média de consumo de bateria, de ocupação de memória e de tempo de execução).

d) Passo 6: Criando os arquivos para a aplicação

Com as configurações de confidencialidade e os dados de desempenho dos algoritmos, o engenheiro precisa agora cadastrar suas aplicações e indicar o nível de confidencialidade que cada uma delas requer. Para isso, ele indica a versão, seleciona uma biblioteca de algoritmos de criptografia que a aplicação utilizará entre as disponíveis na 4MeSAD, e em seguida seleciona o nível de segurança que a aplicação requer. Ao salvar esse cadastro a 4MeSAD apresenta os campos de escolha do perfil de consumo e de classificação de confidencialidade criados. O engenheiro seleciona então a classificação de confidencialidade que ele criou e o perfil também criado por ele e manda gerar o arquivo XML. Esse arquivo é gerado e disponibilizado para o engenheiro, que precisa salvá-lo na pasta do código fonte de sua aplicação.

e) Passo 7: Instalando o subsistema MeSAD

O engenheiro precisa instalar o subsistema do MeSAD nas suas aplicações e para isso ele faz seu *download* a partir da 4MeSAD. Esse *download* é realizado uma única vez e reutilizado em todas as aplicações. Os arquivos compõem uma biblioteca Android e para ser utilizada basta criar um novo projeto no Eclipse indicando a pasta onde foi realizado este *download*. Assim, o Eclipse disponibilizará um projeto que deve ser referenciado em todas as aplicações que utilizarão o MeSAD. Dessa forma, a partir das próximas configurações de aplicação na 4MeSAD, apenas o arquivo XML será necessário, pois a aplicação referência o projeto da biblioteca Android já criado anteriormente.

Esses passos estão evidenciados na Figura 11 pelo diagrama de atividades para a configuração do MeSAD na aplicação. Como descrito no passo a passo, caso o engenheiro opte por utilizar valores personalizados dos parâmetros dos algoritmos criptográficos ele deve seguir os passos indicados pelas atividades apresentadas na Figura 10: “Criar um Perfil de Consumo da biblioteca de segurança” e “Classificar Confidencialidade dos Algoritmos”.

A Figura 11 mostra que para “Indicar os parâmetros dos algoritmos criptográficos” da biblioteca de segurança selecionada, o engenheiro também pode optar por escolher valores pré-definidos como o perfil de consumo e a classificação dos graus de confidencialidade dos algoritmos. Caso ele opte por criar seus próprios parâmetros de consumo e confidencialidade ele altera os valores de pesos, que possibilita a reclassificação dos algoritmos, deixando, por exemplo, um algoritmo muito complexo fora da consideração para uso em sua aplicação.

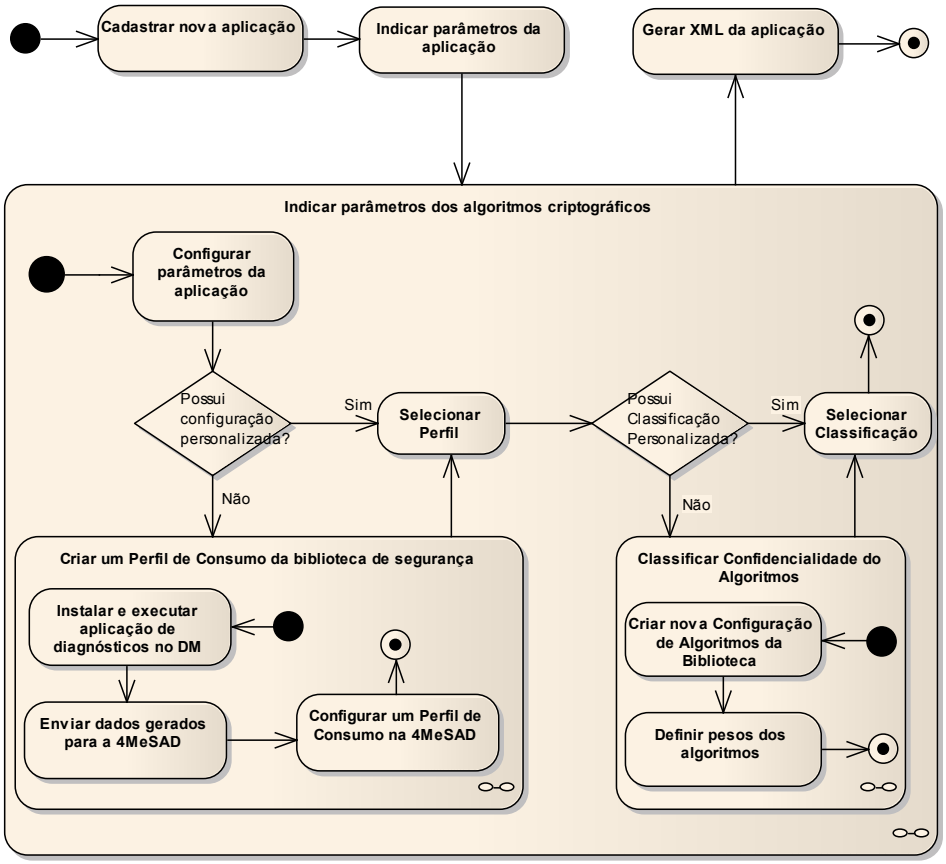


Figura 11 – Cadastrando uma aplicação na 4MeSAD

A atividade "Criar um perfil de consumo da biblioteca de segurança", vista na Figura 11 serve para indicar o consumo de recursos proporcionados pelos algoritmos criptográficos (caso o engenheiro não queira utilizar os valores pré-existentes). Este perfil de consumo é interessante para proporcionar o uso dos algoritmos de acordo com sua complexidade e alocação dos recursos.

Para exemplificar, o engenheiro pode definir valores de configuração de hardware mínimos para o uso de sua aplicação, como quantidade de memória, poder de processamento entre outros recursos computacionais. Com isso ele pode executar a aplicação de diagnósticos nos DMs que atendem aos critérios de arquitetura que ele definiu. Após os dados serem coletados e enviados para a 4MeSAD, ele gera um perfil de consumo para aqueles DMs testados. Essa flexibilidade da 4MeSAD permite que os engenheiros de *software* avaliem o quanto suas aplicações utilizam dos recursos dos DMs e possibilita que essas informações sejam consideradas no momento do fornecimento de confidencialidade pelo MeSAD.

A criação de uma classificação de confidencialidade dos algoritmos requer que o engenheiro possua conhecimento em segurança, pois a atribuição de pesos aos algoritmos permite que o engenheiro adicione-o ou retire-o da lista a ser considerada.

A Figura 12 também apresenta uma visão das entradas que a 4MeSAD necessita e a saída proporcionada.

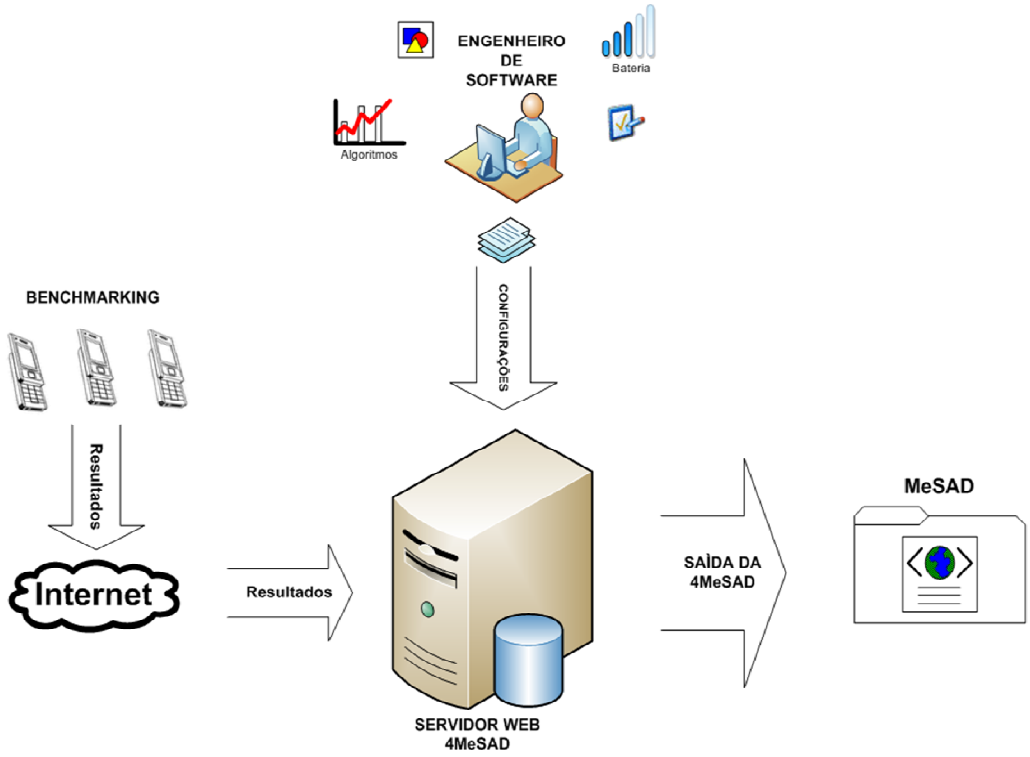


Figura 12 – Entradas e saídas da 4MeSAD

Na Figura 12, pode-se verificar a geração do benchmarking nos DMs e o engenheiro de software analisando os dados produzidos e definindo suas configurações. Após esses dois procedimentos, a saída gerada pela 4MesAD é um arquivo XML que indica todos os parâmetros da aplicação, suas necessidades de confidencialidade, a lista de algoritmos com várias informações que foram identificadas em suas análises.

A Figura 13 mostra um exemplo do arquivo XML gerado a partir das informações da aplicação cadastrada e dos algoritmos criptográficos da biblioteca de segurança. A Figura 13 exibe as *tags: application*, *library* e *algorithm*, com as configurações da aplicação, da biblioteca de segurança e dos algoritmos desta biblioteca, respectivamente.

Com o XML adicionado na aplicação e com a adição da referência da biblioteca Android (subsistema), mencionada no passo 7 (e), o trabalho do desenvolvedor da aplicação está resumido a chamada de dois métodos: o método de preparo da mensagem para envio e o método de preparo para uso da mensagem recebida. O subsistema é responsável por todos os

passos de tratamento de regras, negociação de parâmetros do MeSAD, controle de chaves de segurança e processos de confidencialidade.

```
−<application confidentialitydegree="MEDIO" confidentialitydegreemaximum="2.0" confidentialitydegreeminimum="0.43" name="App_teste"
 overallaveragealgconfdegree="1.18482523608291" overallaveragealgconsrate="4.8411864877885417E-4" version="1">
 −<library name="BouncyCastle v145" platform="java 1.4" programminglanguage="Java" version="1">
    <algorithm confidentialitydegreeofmyalgorithm="1.8284271247461898" decrypttime="34" encrypttime="34" keylengthdefault="192" memoryrequired="9445"
    myconsumptionrate="5.1796914E-4" name="AES" operationmode="CBC" typeofalgorithm="Symmetric"/>
    <algorithm confidentialitydegreeofmyalgorithm="0.54" decrypttime="19" encrypttime="19" keylengthdefault="128" memoryrequired="9453"
    myconsumptionrate="2.9738215E-4" name="AES FAST" operationmode="CBC" typeofalgorithm="Symmetric"/>
    <algorithm confidentialitydegreeofmyalgorithm="0.46499999999999997" decrypttime="20" encrypttime="20" keylengthdefault="256" memoryrequired="13166"
    myconsumptionrate="2.7606907E-4" name="BLOWFISH" operationmode="CBC" typeofalgorithm="Symmetric"/>
    <algorithm confidentialitydegreeofmyalgorithm="1.6500000000000001" decrypttime="31" encrypttime="32" keylengthdefault="256" memoryrequired="31804"
    myconsumptionrate="5.1177054E-4" name="CAST6" operationmode="CBC" typeofalgorithm="Symmetric"/>
    <algorithm confidentialitydegreeofmyalgorithm="0.48" decrypttime="19" encrypttime="19" keylengthdefault="128" memoryrequired="9283"
    myconsumptionrate="2.86728E-4" name="RC5 32 BITS" operationmode="CBC" typeofalgorithm="Symmetric"/>
    <algorithm confidentialitydegreeofmyalgorithm="0.44999999999999996" decrypttime="14" encrypttime="15" keylengthdefault="256" memoryrequired="9421"
    myconsumptionrate="2.2654884E-4" name="RC5 64 BITS" operationmode="CBC" typeofalgorithm="Symmetric"/>
    <algorithm confidentialitydegreeofmyalgorithm="1.5" decrypttime="24" encrypttime="25" keylengthdefault="128" memoryrequired="9132"
    myconsumptionrate="3.5126213E-4" name="RC6" operationmode="CBC" typeofalgorithm="Symmetric"/>
    <algorithm confidentialitydegreeofmyalgorithm="1.7999999999999998" decrypttime="70" encrypttime="64" keylengthdefault="256" memoryrequired="9554"
    myconsumptionrate="0.0013312956" name="SERPENT" operationmode="CBC" typeofalgorithm="Symmetric"/>
    <algorithm confidentialitydegreeofmyalgorithm="1.9500000000000002" decrypttime="28" encrypttime="28" keylengthdefault="256" memoryrequired="22064"
    myconsumptionrate="5.580424E-4" name="TWOFISH" operationmode="CBC" typeofalgorithm="Symmetric"/>
  </library>
</application>
```

Figura 13 – Exemplo de um arquivo de configuração da aplicação

O cadastro inicial na 4MeSAD, mencionado no passo 1 desta seção, pode ser visto na Figura 14, a qual apresenta a página de acesso à 4MeSAD. O acesso é possível após a digitação do nome do usuário da 4MeSAD e sua senha de acesso, na opção “Login”. A opção de novo cadastro está disponibilizada na opção “Efetuar Cadastro”, vista nesta mesma figura.

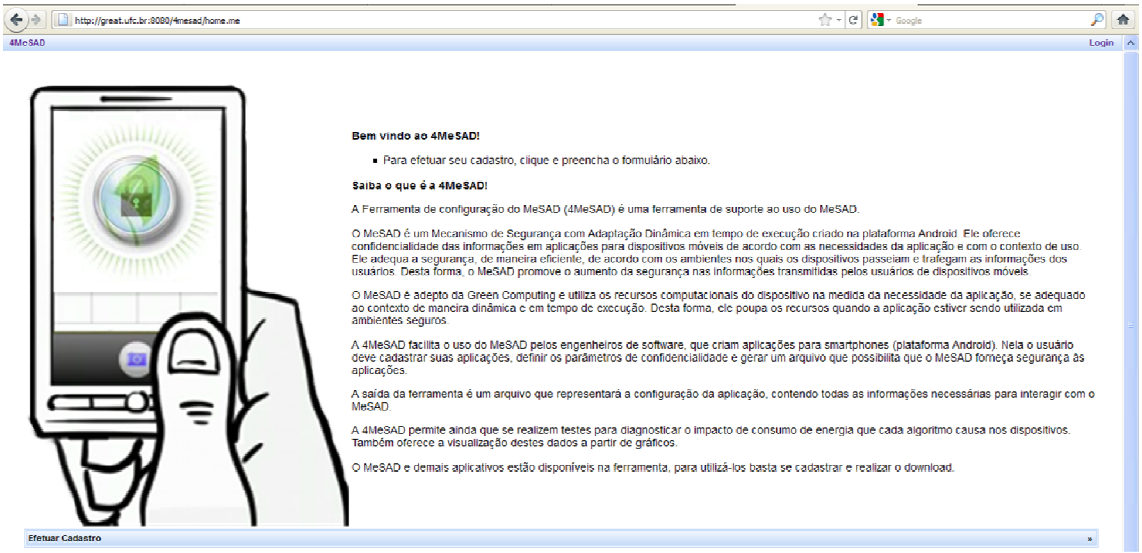
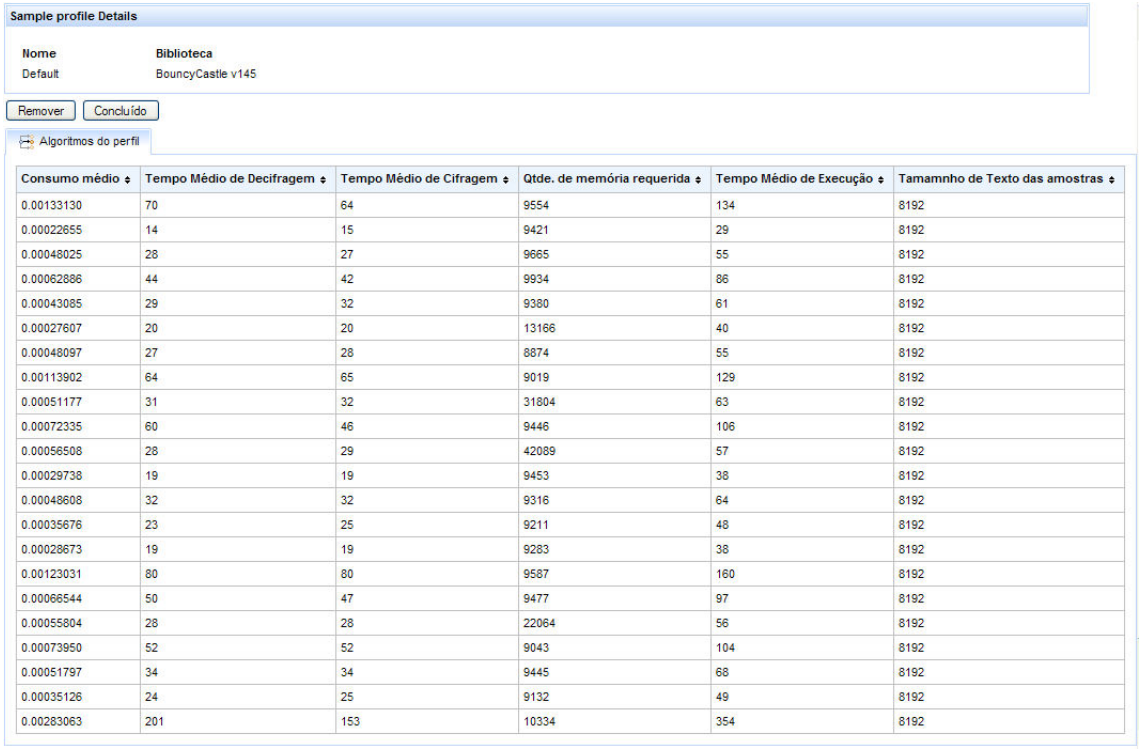


Figura 14 – Tela inicial da 4MeSAD, para acesso e cadastro de novo usuário da ferramenta

A Figura 15 mostra um perfil de consumo criado por um engenheiro de *software* ou um especialista em segurança, neste perfil nota-se a existência de valores de consumo de recursos identificados na execução dos testes nos DMs, para cada algoritmo criptográfico.



Sample profile Details

Nome	Biblioteca
Default	BouncyCastle v145

Remover | Concluído

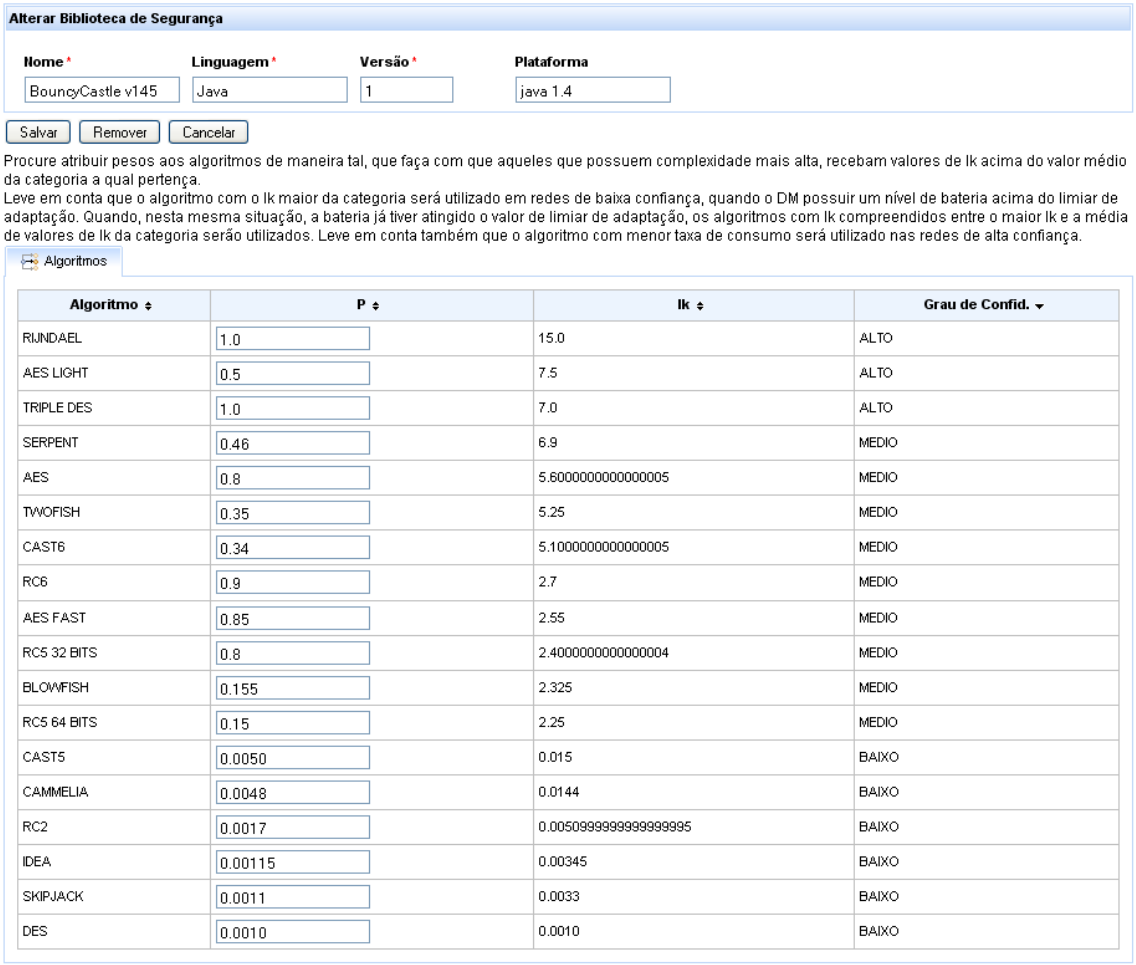
Algoritmos do perfil

Consumo médio	Tempo Médio de Decifragem	Tempo Médio de Cifragem	Qtde. de memória requerida	Tempo Médio de Execução	Tamamnho de Texto das amostras
0.00133130	70	64	9554	134	8192
0.00022655	14	15	9421	29	8192
0.00048025	28	27	9665	55	8192
0.00062886	44	42	9934	86	8192
0.00043085	29	32	9380	61	8192
0.00027607	20	20	13166	40	8192
0.00048097	27	28	8874	55	8192
0.00113902	64	65	9019	129	8192
0.00051177	31	32	31804	63	8192
0.00072335	60	46	9446	106	8192
0.00056508	28	29	42089	57	8192
0.00029738	19	19	9453	38	8192
0.00048608	32	32	9316	64	8192
0.00035676	23	25	9211	48	8192
0.00028673	19	19	9283	38	8192
0.00123031	80	80	9587	160	8192
0.00066544	50	47	9477	97	8192
0.00055804	28	28	22064	56	8192
0.00073950	52	52	9043	104	8192
0.00051797	34	34	9445	68	8192
0.00035126	24	25	9132	49	8192
0.00283063	201	153	10334	354	8192

Figura 15 – Configuração de um perfil de consumo

A 4MeSAD oferece uma configuração de uso padrão e permite que o engenheiro de *software* que possuir maiores conhecimentos em criptografia, crie e adote uma configuração própria. Contudo, a reclassificação dos algoritmos criptográficos por parte do engenheiro se restringe à definição de pesos para cada algoritmo. Tais pesos indicam a importância que cada algoritmo possui e são estabelecidos de acordo com o conhecimento e as necessidades do engenheiro de *software* ou especialista em segurança. Assim, o engenheiro poderá indicar (mudando os pesos dos algoritmos) as suas preferências de algoritmos para cada nível de segurança e permitir que suas aplicações sejam protegidas de acordo com esta nova classificação.

A Figura 16 mostra uma das telas da ferramenta, onde o engenheiro atribui pesos para os algoritmos. A cada atribuição de peso o algoritmo assume um valor de grau de confidencialidade (lk) diferente e pode mudar de classificação quanto ao nível de segurança atendida na aplicação, o que pode ser visto na última coluna do *grid* na Figura 16.



Alterar Biblioteca de Segurança

Nome * BouncyCastle v145 | Linguagem * Java | Versão * 1 | Plataforma java 1.4

Salvar Remover Cancelar

Procure atribuir pesos aos algoritmos de maneira tal, que faça com que aqueles que possuem complexidade mais alta, recebam valores de Ik acima do valor médio da categoria a qual pertença.
Leve em conta que o algoritmo com o Ik maior da categoria será utilizado em redes de baixa confiança, quando o DM possuir um nível de bateria acima do limiar de adaptação. Quando, nesta mesma situação, a bateria já tiver atingido o valor de limiar de adaptação, os algoritmos com Ik compreendidos entre o maior Ik e a média de valores de Ik da categoria serão utilizados. Leve em conta também que o algoritmo com menor taxa de consumo será utilizado nas redes de alta confiança.

Algoritmos

Algoritmo	P	Ik	Grau de Confid.
RIJNDAEL	1.0	15.0	ALTO
AES LIGHT	0.5	7.5	ALTO
TRIPLE DES	1.0	7.0	ALTO
SERPENT	0.46	6.9	MEDIO
AES	0.8	5.6000000000000005	MEDIO
TWOFISH	0.35	5.25	MEDIO
CAST6	0.34	5.1000000000000005	MEDIO
RC6	0.9	2.7	MEDIO
AES FAST	0.85	2.55	MEDIO
RC5 32 BITS	0.8	2.4000000000000004	MEDIO
BLOWFISH	0.155	2.325	MEDIO
RC5 64 BITS	0.15	2.25	MEDIO
CAST5	0.0050	0.015	BAIXO
CAMMELIA	0.0048	0.0144	BAIXO
RC2	0.0017	0.0050999999999999995	BAIXO
IDEA	0.00115	0.00345	BAIXO
SKIPJACK	0.0011	0.0033	BAIXO
DES	0.0010	0.0010	BAIXO

Figura 16 – Definições de pesos para os algoritmos

Um exemplo da utilidade desta flexibilização da 4MeSAD na classificação dos algoritmos: um engenheiro de *software* pode considerar que determinado algoritmo, que faz parte da lista dos algoritmos do nível de segurança alto, não é interessante para uso em suas aplicações. Ele pode então atribuir um valor 0 (zero) ao peso deste algoritmo, o que fará com que o algoritmo passe a possuir um grau de confidencialidade igual a zero, saindo da lista dos algoritmos considerados para serem usados no nível de segurança alto. A atribuição dos valores do peso igual a 0 é destaca por Carvalho (2008) como uma forma de desconsiderar a sua importância na análise que está sendo realizada.

5.2 Gráficos

A 4MeSAD também disponibiliza a geração de gráficos para que o engenheiro de *software* possa verificar o desempenho dos algoritmos criptográficos testados nos DMs. Com isto, ele pode tomar suas decisões quanto à modificação ou não nas classificações de grau de confidencialidade ou nos perfis de consumo.

Os gráficos disponíveis mostram os valores identificados na aplicação de diagnósticos executada nos DMs. Eles apresentam visualizações quanto aos valores de memória ocupada, o tempo no processamento da tarefa de criptografia, o consumo de bateria e a quantidade de rodadas para consumir 1% da bateria (menor medida de consumo).

Todos os gráficos apresentam os valores de cifragem e de decifragem de cada algoritmo. Isso permite que o engenheiro verifique o desempenho dos algoritmos quando realizam essas duas operações, pois esses valores são bem diferentes em alguns casos.

A 4MeSAD também disponibiliza um gráfico que apresenta os comportamentos dos algoritmos quanto ao consumo de bateria e quantidade de vezes que podem ser reutilizados nas adaptações.

Exemplificando, o engenheiro conhece o grupo de algoritmos que fazem parte do nível de segurança de sua aplicação, ele pode então, verificar como esses algoritmos se comportariam se fossem utilizados em conjunto durante um período. Ele escolhe então, na 4MeSAD, o tipo de gráfico "*Rounds* x Consumo", preenche os campos solicitados, seleciona o DM que quer verificar, seleciona esse grupo de algoritmos e seleciona um valor de consumo de bateria, a partir do qual será modificado o algoritmo em uso. O gráfico apresentado será a sequência dos algoritmos selecionados mostrando o desempenho de cada um, em número de *rounds* executados durante o consumo da bateria.

A Figura 17 mostra um desses gráficos com o uso de nove algoritmos sendo modificados a cada 1% de consumo de bateria. No eixo x pode-se verificar a quantidade de rodadas que cada um executa durante o consumo de 1% da energia da bateria. Nota-se, por exemplo, que o algoritmo *Serpent* executa bem menos rodadas do que os outros durante o consumo da mesma energia da bateria (1%).

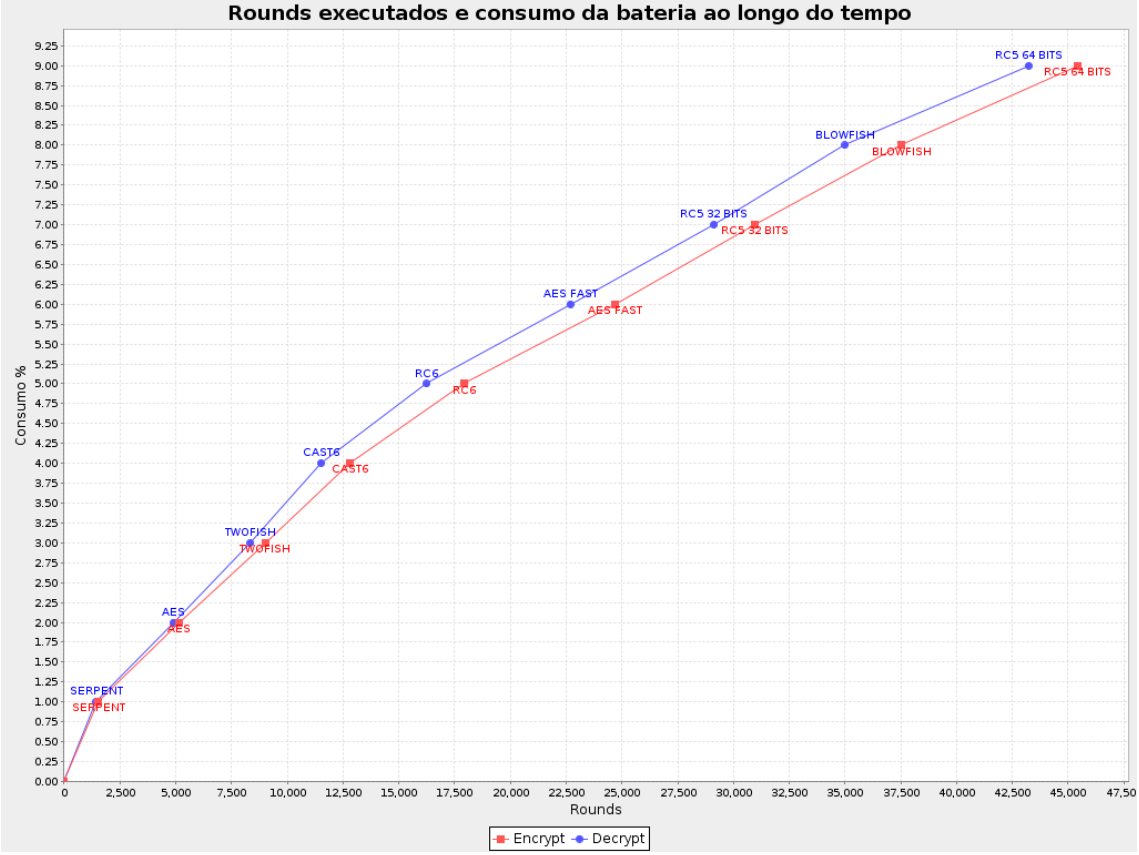


Figura 17 – Gráfico "*Rounds* x Consumo", disponível na 4MeSAD

Este gráfico da Figura 17 possibilita, ao engenheiro, ter uma idéia do comportamento do MeSAD quando estiver em funcionamento, pois o engenheiro pode selecionar os algoritmos que sua aplicação utilizará e verificar o comportamento desses algoritmos na adaptação.

A Figura 18 mostra um gráfico disponível na 4MeSAD que apresenta o tempo de execução dos algoritmos testados em um DM específico (selecionado pelo usuário da 4MeSAD). Este gráfico é importante para o engenheiro verificar o desempenho dos algoritmos de determinada biblioteca de segurança, em relação ao seu tempo de execução nos DMs testados (quando da execução da aplicação dos DMs que gera dados para popular a 4MeSAD). Pode-se verificar no gráfico da Figura 18, que os algoritmos *Serpent* e RC5 64 Bits possuem tempos de execução muito diferentes.

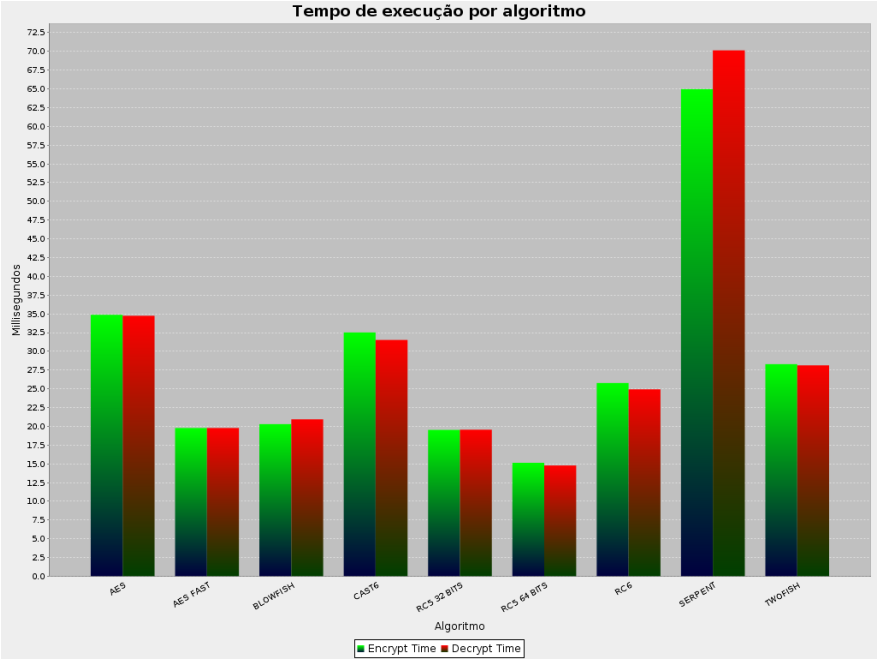


Figura 18 – Gráfico de tempo de execução dos algoritmos

Na Figura 19 está sendo apresentado o gráfico de ocupação de memória de alguns algoritmos em um DM específico, no qual a aplicação de diagnósticos foi executada. Neste caso trata-se do DM Xperia X10a. A ocupação de memória dos algoritmos CAST6 e *Twofish*, por exemplo, é bem superior a dos outros algoritmos, assim como o uso de memória da maioria dos algoritmos visualizados possuem um valor muito parecido.

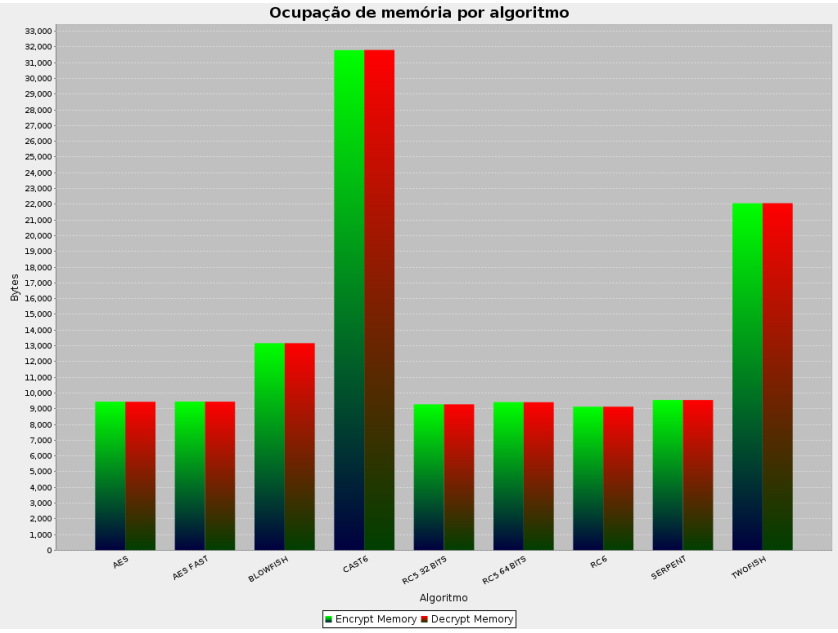


Figura 19 – Gráfico de ocupação de memória dos algoritmos

Esse tipo de informação pode auxiliar o engenheiro a excluir algoritmos que ocupam mais memória do que ele espera que sua aplicação utilize nos DMs.

A Figura 20 mostra um gráfico que apresenta dados sobre o consumo proporcionado pelos algoritmos avaliados em determinado DM (selecionado pelo usuário da 4MeSAD). Esta informação é útil para o engenheiro selecionar os algoritmos que atendam aos requisitos de baixo consumo que suas aplicações requerem.

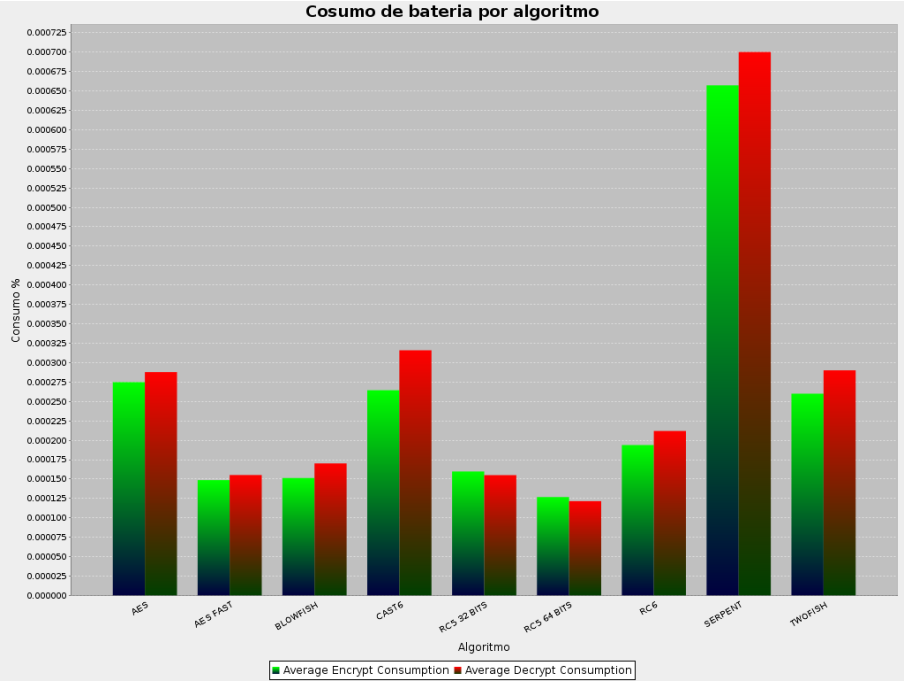


Figura 20 – Gráfico de indicação do consumo de energia da bateria do DM

A 4MeSAD oferece orientações para a produção dos experimentos e a aplicação para a produção e coleta desses dados, além de gráficos para a visualização do comportamento dos algoritmos nos DMs testados pelo engenheiro de *software* ao utilizar a ferramenta. Os gráficos auxiliam o engenheiro, nas suas decisões sobre o uso dos algoritmos nas suas aplicações. Assim, para que o MeSAD atue de forma eficaz, assume-se que a classificação dos algoritmos feita pelo engenheiro de software ou especialista em segurança tenha sido produzida de maneira adequada. Caso o engenheiro não possua nenhum conhecimento sobre criptografia e confidencialidade de informações um especialista em segurança pode auxiliar essa etapa de configuração para que seja reutilizada em todas as aplicações desse engenheiro.

5.3 Conclusões

A 4MeSAD foi construída com o intuito de facilitar a adoção do MeSAD nas aplicações para DMs e de oferecer praticidade no seu manuseio, sendo ainda flexível para acomodar personalizações dos engenheiros de *software* com maior conhecimento em segurança.

O principal objetivo da ferramenta é alcançado, pois é possível realizar as análises pelo engenheiro de *software*, a configuração e a geração dos arquivos necessários para uso pelo MeSAD na adaptação dinâmica.

A 4MeSAD é útil não só para quem vai utilizar o MeSAD mas também para quem quer apenas avaliar algoritmos de criptografia quanto à alocação dos recursos de memória, consumo de bateria e tempo de execução nos DMs,.

Os atributos da 4MeSAD possibilitam que ela seja uma ferramenta facilitadora na tarefa de tornar menos complexo o processo de implantação de confidencialidade em aplicações criadas para DMs, um dos objetivos deste trabalho.

O próximo capítulo desta dissertação trata sobre a avaliação deste trabalho.

6 AVALIAÇÃO DO MeSAD

Este capítulo apresenta a avaliação do mecanismo de segurança adaptativa proposto nesta dissertação. A Seção 6.1 introduz o assunto tratado no capítulo. Na Seção 6.2 é apresentado o ambiente utilizado na avaliação. As medições e análises produzidas na ferramenta de apoio são mostradas na Seção 6.3. Os critérios adotados e a avaliação do MeSAD são apresentados na Seção 6.4. A Seção 6.5 apresenta a avaliação realizada no mecanismo de segurança adaptativa. Na Seção 6.6 considerações sobre o uso estático de um algoritmo para provisão de confidencialidade em comparação com o uso do MeSAD são apresentadas. Por fim, a Seção 6.7 conclui o capítulo.

6.1 Introdução

Soluções em segurança envolvem parâmetros complexos de serem medidos, sendo um problema difícil de ser resolvido (conforme mencionado nas Seções 3.2 e 3.4.1). A verificação de confidencialidade se enquadra neste cenário, no qual o tempo envolvido na execução dos testes, a criação dos ambientes de verificação, a criação dos modelos contraditórios, o esforço e o custo envolvido na verificação dificultam, ainda mais, o procedimento de medição. Esta complexidade aumenta significativamente quando se envolvem redes sem fio e o uso de segurança adaptativa em tempo de execução. Fenton (1994) e Gentry (2004) citam esta problemática de representação de parâmetros complexos, custos envolvidos, limitações de esperança na verificação, além de tempo para a montagem de modelos adversativos, para utilização na prática de verificação em domínios de segurança (já citados na Seção 3.2.1).

Esta complexidade envolvida na medição de segurança proporcionada pelo trabalho, adicionadas à ausência de outros trabalhos equivalentes que pudessem ser comparados para a validação da contribuição, não permite a verificação de todas variáveis de segurança oferecidas pelo MeSAD. Contudo, foi possível verificar o atendimento do que o trabalho se propõe a resolver quanto ao aumento da confidencialidade nos contextos de baixa confiança e quanto à diminuição do consumo de energia em contextos de alta confiança.

6.2 Ambiente dos experimentos

Os recursos utilizados para avaliar o mecanismo de segurança adaptativa e considerados suficientes para a verificação do MeSAD, são apresentados nesta seção. Eles foram utilizados, em conjunto com as plataformas livres ofertadas no mercado, para execução dos experimentos em dispositivos reais e para a construção dos códigos utilizados na proposta.

O desenvolvimento deste trabalho enfrentou grandes dificuldades com as plataformas disponíveis, quanto à captura de valores de contexto do dispositivo. Um deles foi a ausência de uma maneira fiel de medição do consumo de bateria em tempo de execução. Diante disto, para se obter este valor, o trabalho de medição aumentou consideravelmente, pois esse valor é um critério chave da dissertação e precisava ser garantido.

Para se ter uma idéia da dificuldade, a plataforma JME oferece um modo de acessar o valor do nível de bateria em apenas alguns dispositivos que implementam a JSR 256 (*Java Specification Requests*). Ainda assim, os valores disponíveis para verificação se restringem a três: 33%, 66% e 99%; além destes valores só serem acessíveis às aplicações que possuem certificados de validação fornecidos por empresas especializadas. Cada fabricante também personaliza o modo de acesso a esta informação, necessitando de um tratamento personalizado, por DM e por fabricante, para acessá-la.

Estas dificuldades tornam a verificação do consumo de energia por determinada aplicação, ou pior, por determinado trecho de código, uma tarefa muito difícil de ser realizada por uma aplicação de análise de consumo que tem o objetivo de ser genérica o suficiente para ser executada em vários modelos de DMs.

Como informado, para o desenvolvimento deste trabalho, inicialmente, foi necessário conhecer as plataformas que oferecessem maior viabilidade para implementação do mecanismo. Inicialmente considerou-se o uso de JME, com o intuito de reutilizar os trabalhos científicos produzidos pelo GREat[4] [BRINGEL, 2004]. Porém, essa plataforma não ofereceu viabilidade para aquisição das informações de contexto necessárias ao MeSAD. Este também foi um dos motivos para a criação da 4MeSAD, apresentada no Capítulo 5.

[4] Copyright @ GREat (http://great.ufc.br)

Foram analisadas paralelamente as plataformas *Symbiam*[5], *Windows Mobile*[6] *iPhone*[7] e *Android*[8] para a escolha de uma que não oferecesse as mesmas dificuldades que JME ofereceu no início do trabalho. Entre as plataformas avaliadas, a *Android* foi identificada como a mais viável de se obter as informações necessárias sobre o contexto dos DMs e foi a escolhida para a implementação do projeto. Um dos principais motivos da escolha foi a menor curva de aprendizado estimada, devido à familiaridade com a linguagem Java, a qual é a linguagem de desenvolvimento adotada pelo *Android*.

Após lidar com as dificuldades acima expostas e após encontrar uma plataforma que oferecesse maior acesso às informações de contexto do DM em tempo de execução, encontrou-se um modo de se medir o consumo proporcionado pelos algoritmos em um DM. Isto tornou possível a disponibilização da aplicação de diagnósticos para uso pelos engenheiros de software na 4MeSAD, para que eles avaliassem o consumo de energia proporcionado pelo uso dos algoritmos criptográficos nos DMs.

A verificação do MeSAD foi realizada em dispositivos reais, pois não foi encontrado simulador capaz de reproduzir algumas situações necessárias, como a aquisição, em tempo de execução, do valor de bateria e parâmetros das redes sem fio.

Os DMs utilizados nas verificações deste trabalho e nos experimentos produzidos na ferramenta foram dois: o modelo Xperia X10a e o modelo GT540. As características destes dois modelos estão apresentadas na Tabela 2.

Tabela 2 – Características dos dispositivos utilizados no trabalho

Modelo	Bateria	Fabricante	Memória (MB)	Freq. do Processador (MHz)	Máquina virtual
Xperia X10a	Bateria de LI-Polymer 1500 mAh	Sony Ericson	384	1000	Android Dalvik
GT 540	Bateria de LI-Polymer 1500 mAh	LG	200	600	Android Dalvik

O modelo Xperia X10a, visto na Tabela 2, possibilitou a realização dos experimentos provenientes da ferramenta de apoio, fornecendo dados para avaliação dos algoritmos que serão considerados no MeSAD. Este modelo fornece informações sobre o contexto do dispositivo em tempo de execução em escalas mais precisas do que o outro modelo disponível para uso (GT 540). Já o modelo GT 540 possibilitou o estabelecimento do

[5] Copyright © 2011Nokia.All rights reserved (http://symbian.nokia.com/)

[6] Windows Mobile ® (http://www.microsoft.com/)

[7] iPhone Copyright © 2011 Apple Inc. Todos os direitos reservados (http://www.apple.com/br/iphone/)

[8] Android ® (http://www.android.com/)

tráfego das informações tratadas na aplicação criada para utilizar o MeSAD, recebendo e transmitindo dados provenientes do primeiro modelo (Xperia X10a).

O ambiente de criação da ferramenta 4MeSAD foi basicamente a plataforma de desenvolvimento Eclipse[9], com o uso da linguagem JAVA em conjunto com o *framework* SEAM[10] e um banco de dados MySQL[11].

A 4MeSAD forneceu apoio para a verificação do comportamento dos algoritmos e a partir disso, realizou a seleção e classificação de seus graus de confidencialidade para utilização no MeSAD. Os DMs possibilitaram a verificação do funcionamento real do mecanismo de segurança adaptativa, e os resultados estão apresentados nas seções seguintes.

6.3 Medições realizadas

Para a criação de uma lista de algoritmos para uso pelo mecanismo de segurança adaptativa foi utilizada a 4MeSAD, com o intuito de verificar o comportamento de alguns algoritmos. Essa verificação possibilitou classificá-los em uma lista, de acordo com seu grau de confidencialidade e com seu grau de consumo de bateria. As medições criadas para a verificação e classificação dos algoritmos criptográficos na 4MeSAD foram realizadas com o intuito de fornecer uma análise do comportamento destes algoritmos nos DMs avaliados. A intenção foi produzir dados para facilitar a verificação do comportamento dos algoritmos quando executados nos dispositivos.

Os algoritmos avaliados fazem parte da biblioteca de segurança *Bounce Castle Crypto*[12]. A partir dos dados obtidos na análise, foi criada uma lista com suas classificações seguindo os critérios mencionados nos Capítulos 4 e 5. Assim, os algoritmos foram classificados de acordo com o tamanho das chaves utilizadas. Após isto a definição dos pesos com base na complexidade dos algoritmos foi utilizada para diferenciá-los. Os valores atribuídos aos pesos foram definidos com base na complexidade dos algoritmos considerada em alguns trabalhos encontrados [LI, 2006], [NIRAJAN, 2004], [SON, 2000], [LAMPRECHT, 2008], [FIPS 197, 2001], [ROCHA et al., 2010], [CHEN, 2010],

[9] Copyright © 2011 The Eclipse Foundation. All Rights Reserved (http://www.eclipse.org/org/)

[10] Copyright © 2009, Red Hat Middleware, LLC. All rights reserved. JBoss and Seam are registered trademarks and servicemarks of Red Hat, Inc. {Privacy Policy} (http://seamframework.org)

[11] Copyright © 2010, Oracle Corporation and/or its affiliates (http://www.mysql.com)

[12] Copyright © 2000 - 2011 The Legion Of The Bouncy Castle (http://www.bouncycastle.org)

[HAMAD,2009], [IZQUIERDO, 2005], [TADDEO, 2010], [DOOMUN, 2007] e [SCHNEIER, 2000]. Contudo, não foram encontrados na literatura trabalhos que considerassem alguns dos algoritmos da biblioteca, nestes casos o critério adotado foi a verificação do comportamento quanto ao desempenho verificado em relação ao tempo de execução, uso de memória e consumo de bateria desses algoritmos ao executar os testes da 4MeSAD. Entretanto, esse último critério é insuficiente para definir a complexidade de um algoritmo, portanto é necessário o estudo para definir um valor que represente a complexidade dos algoritmos. Esse estudo é uma atividade direcionada para trabalhos futuros.

É importante deixar claro que os resultados dos testes evidenciam o comportamento de uma determinada implementação dos algoritmos de uma biblioteca específica, sendo executados em um determinado DM. Alguns parâmetros, como o tamanho do texto, que influenciam nos resultados possuem uma flexibilidade para serem alterados na aplicação do DM, contudo esses parâmetros não influenciam no objetivo das medições e não alteram a forma como os dados são coletados [RAJ JIAN, 1991]

Os dados produzidos pela aplicação de verificação do comportamento dos algoritmos no DM XPeria XA podem ser vistos nos gráficos a seguir. Tais dados auxiliaram a escolha e a configuração dos parâmetros de segurança e de consumo dos algoritmos criptográficos.

O gráfico da Figura 18 apresenta a média amostral dos experimentos quanto ao tempo para cifragem e para decifragem de um texto de tamanho 8kBytes no DM Xperia XA. O nível de confiança considerado foi de 95%, com um *alpha* de 0,05 e número de amostras coletadas igual a 31. Estes valores também foram utilizados nos outros gráficos que serão apresentados nesta seção. Com base nestes dados, foi possível verificar quais algoritmos demoram mais para cifrar e decifrar o texto, e quais são os mais rápidos. Os valores de tempo estão representados em milissegundos.

Os experimentos contemplaram separadamente os dados de cifragem e decifragem, pois eles possuem comportamentos diferentes de acordo com cada algoritmo. Os dados dos tempos de cifragem e de decifragem de cada algoritmo podem ser vistos no gráfico da Figura 21 com a indicação dos intervalos de confiança.

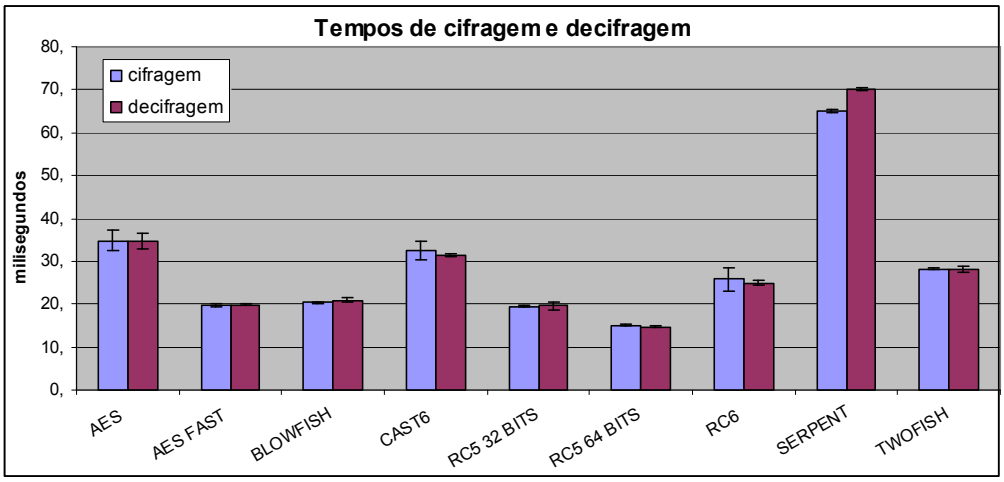


Figura 21 – Tempos de cifragem e de decifragem dos algoritmos utilizados

Analisando o gráfico pode-se verificar, por exemplo, que o algoritmo *Serpent* é o algoritmo que demora mais para realizar tanto a cifragem como a decifragem do texto. Isto acontece devido à sua alta complexidade, confirmando o que já se esperava, pois ele é o algoritmo de mais alto nível de segurança proporcionada, entre os algoritmos listados.

O gráfico da Figura 22 apresenta a quantidade de memória requerida pelos algoritmos para realizarem a tarefa de cifragem e de decifragem do mesmo texto utilizado para a geração dos dados de tempo. Os valores de intervalo de confiança também foram inseridos no gráfico, mas não são visualizados devido ao fato de serem muito baixos.

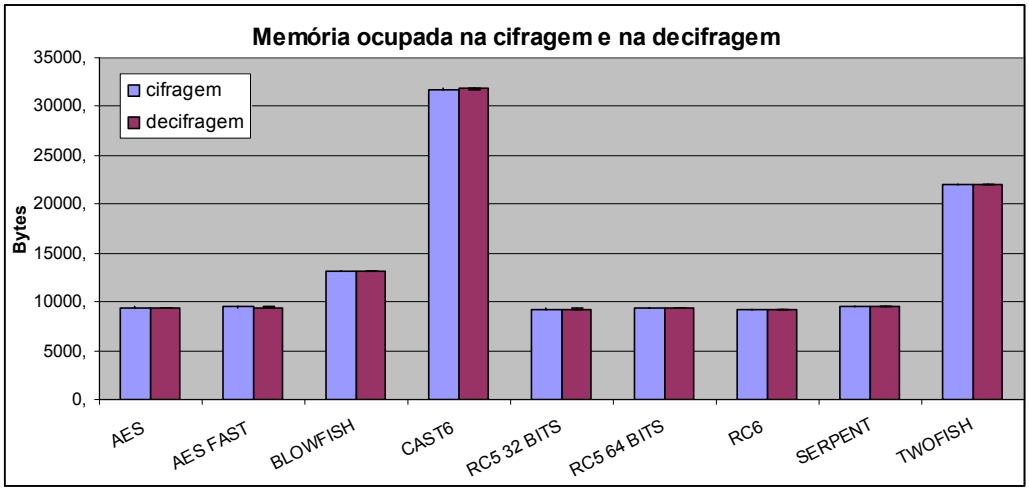


Figura 22 – Uso de memória pelos algoritmos selecionados

No gráfico da Figura 22, pode-se verificar, por exemplo, que o algoritmo CAST6 requer a maior quantidade de memória entre os algoritmos analisados, tanto para cifragem como para decifragem. Esta informação serve para indicar quanto de recursos estes algoritmos

exigirão dos DMs, serve também para prover uma expectativa quanto ao consumo de bateria, pois se sabe que quanto mais memória for utilizada em um dispositivo mais energia será gasta para acessar estes dados, reduzindo o nível residual da bateria.

As informações de consumo dos algoritmos estão apresentadas no gráfico da Figura 23, o qual mostra o número de rodadas executados até que o DM consumisse 1% de bateria. Esta medida foi necessária devido à dificuldade, citada no início desse capítulo, para a aquisição de um valor de consumo proporcionado por um processo isolado de cifragem ou decifragem. A única informação sobre consumo de bateria, acessível pela plataforma em tempo de execução, é a porcentagem do nível residual da bateria no momento de consulta. A escala desta informação, no DM avaliado, apresenta valores diferentes em uma unidade, assim, é possível identificar quando a bateria decai sua energia em 1%. Com base neste fator limitante, foi decidido que o parâmetro a ser avaliado seria o número de rodadas executados até que o consumo da bateria decaísse em 1%. Com esta informação pode-se chegar a uma estimativa de média de consumo por rodada, dividindo o valor mínimo da escala disponível (1%) pelo número de repetições do processo de cifragem até o consumo deste valor mínimo (**1/rodadas**). O mesmo critério foi utilizado para a decifragem. Assim, obtém-se uma média do consumo proporcionado pelo algoritmo em uma rodada.

O gráfico da Figura 23 apresenta a média amostral de rodadas executadas por cada algoritmo até o consumo de 1% da energia da bateria. Pode-se notar pelo gráfico que o algoritmo *Serpent*, por exemplo, executou o menor número de rodadas entre os algoritmos. Isto demonstra que ele consumiu mais energia para executar a cifragem e a decifragem dos dados, devido a sua complexidade e confirmando o que se esperava sobre o consumo proporcionado pelos algoritmos de maiores complexidade.

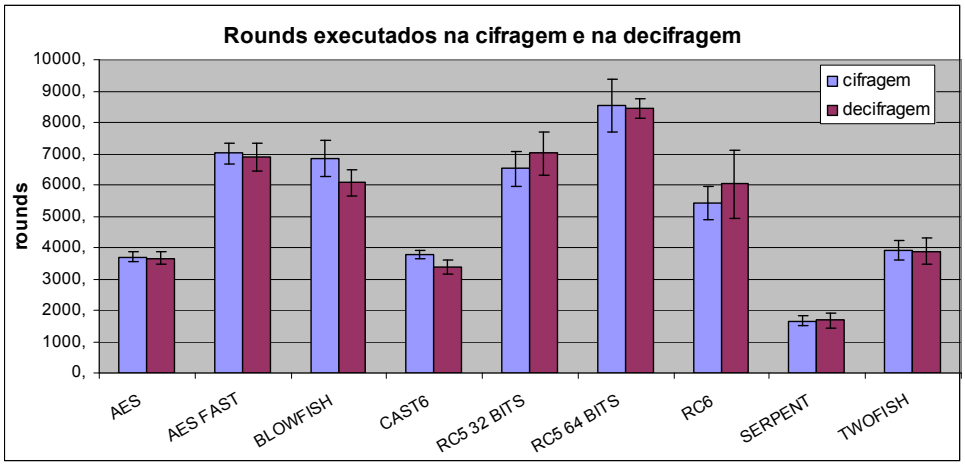


Figura 23 – Repetição de rodadas até o consumo de 1% da energia da bateria

Na Figura 23, pode-se verificar que os intervalos de confiança são os maiores entre os gráficos apresentados. Isto acontece devido às características do processo de medição de energia na plataforma *Android*, pois os valores que são retornados não possuem alta precisão. Contudo, verifica-se que as médias possuem valores bem diferentes, o que proporciona uma constatação das diferentes quantidades de rodadas que cada algoritmo necessita executar até consumir a mesma quantidade de energia (1%). Assim, nota-se que o algoritmo com o menor número de rodadas executadas é aquele que consome mais energia por rodada.

A análise dos gráficos apresentados tornou possível a identificação dos algoritmos que mais consomem recursos no DM avaliado, e aqueles que são mais econômicos. Os resultados mostraram o que já se esperava, pois os algoritmos mais complexos e mais fortes na provisão de confidencialidade foram os que mais consumiram recursos.

Os tamanhos de chaves indicados pela biblioteca de segurança *Bounce Castle Crypto*, em conjunto com a análise dos dados produzidos na 4MeSAD, viabilizaram a classificação dos algoritmos avaliados na ferramenta, conforme o modo de definição de grau de confidencialidade apresentado no Capítulo 4.

Para a avaliação do MeSAD, foram classificados os nove algoritmos aqui explorados utilizando a metodologia definida no início dessa seção. Eles foram configurados para atender as necessidades de aplicações cadastradas na ferramenta, com a indicação de nível médio de confidencialidade.

Uma aplicação foi criada com o único intuito de testar o funcionamento real do MeSAD. Ela utiliza um ponto de acesso para transmitir suas informações para o seu destinatário e, antes de enviar estas informações, utiliza o MeSAD para aplicar a confidencialidade. Ao receber informações da rede esta aplicação utiliza novamente o MeSAD para tratar a confidencialidade e obter a informação decifrada.

Para se comunicar com o MeSAD, a aplicação precisa adicionar ao seu projeto de implementação o subsistema mencionado nos Capítulos 4 e 5. Esse subsistema é adquirido após o cadastro desta aplicação na ferramenta 4MeSAD. Assim, a aplicação foi cadastrada na 4MeSAD com a indicação de necessidade de confidencialidade de nível MÉDIO. Em seguida foram gerados os arquivos de configuração, e adicionados ao código da aplicação antes de ser compilada. A aplicação foi então compilada e instalada nos DMs, que foram utilizados para se comunicarem transmitindo textos de tamanho 8Kbytes através de uma rede *WiFi*.

Os tamanhos de textos disponíveis na aplicação criada podiam variar de 1kBytes a 256kBytes. Contudo, para concentrar a atenção no funcionamento do MeSAD e em suas

adaptações à medida que a bateria fosse sendo consumida e para a identificação mais precisa do momento no qual a bateria mudasse de valor, foi escolhido um texto de tamanho pequeno. Assim, foi utilizado um tamanho de texto de 8kBytes para a realização da cifragem e transmissão.

A avaliação do MeSAD contemplou casos reais e os DMs foram utilizados em diferentes redes, com diferentes classificações de confiança do usuário e em diferentes momentos de uso.

Além das diferentes redes, diferentes níveis de bateria, em diferentes momentos, compuseram os cenários de análises. Essas informações estão apresentadas na seção 6.4, 6.5 e 6.6.

6.4 Critérios de Avaliação

A finalidade desta etapa é descrever e analisar os critérios da avaliação, para determinar se os requisitos arquiteturais de segurança adaptativa, focada na confidencialidade, são viáveis e podem ser atendidos pelo MeSAD, em tempo de execução.

Conforme a motivação e a construção deste trabalho apresentadas nos Capítulos 1 e 4, os critérios, segundo os quais a avaliação deste trabalho foi realizada, foram obtidos a partir dos requisitos considerados importantes para a arquitetura do MeSAD, e que impulsionaram sua construção.

O MeSAD foi criado com a intenção de mostrar como o tratamento da confidencialidade, com o uso adequado dos algoritmos de criptografia e de acordo com cada contexto, pode funcionar de maneira satisfatória. Ele busca atender aos requisitos de alta confidencialidade e de economia no consumo de bateria. Para isto ele se baseia em algumas regras de adaptação, as quais foram criadas com base nas premissas deste trabalho e que devem ser seguidas para que o objetivo deste mecanismo de segurança seja alcançado. O atendimento destas regras, evidenciadas na Tabela 3, forma o conjunto de critérios de avaliação utilizados e aplicados ao MeSAD.

As regras devem ser obedecidas quando o MeSAD necessitar selecionar um algoritmo de criptografia. Para isto ele deverá identificar o contexto no qual se encontra e selecionar, dentre os algoritmos disponíveis para utilização, aquele que atende às regras do contexto identificado.

Tabela 3 – Regras de adaptação do MeSAD

GRAU DE CONFIANÇA NA REDE	ESTADO DA BATERIA			REGRA
	Acima do Threshold (A)	Entre o Threshold e o Valor Crítico (B)	Abaixo do Valor Crítico (C)	
ALTO	X			1.a
		X		1.b
			X	1.c
MÉDIO	X			2.a
		X		2.b
			X	2.c
BAIXO	X			3.a
		X		3.b
			X	3.c
1.a: O primeiro algoritmo deve ser escolhido, de maneira aleatória, entre os de grau de confidencialidade abaixo da média e diferente daquele de menor valor. Os próximos serão os de valor imediatamente inferior ao último em uso, mudando até o de menor valor.				
1.b: Deve sempre retornar o algoritmo de menor consumo de bateria.				
1.c: Deve sempre retornar o algoritmo de menor consumo de bateria.				
2.a: Deve retornar o algoritmo com o segundo maior grau de confidencialidade.				
2.b: O primeiro algoritmo deve ser escolhido, de maneira aleatória, entre os de grau de confidencialidade acima da média e diferente daquele de maior valor. Os próximos serão os de valor imediatamente inferior ao último em uso e diferente daquele de menor valor.				
2.c: Deve retornar sempre o de menor consumo de bateria.				
3.a: Deve sempre retornar o algoritmo de maior grau de confidencialidade.				
3.b: O primeiro algoritmo deve ser o que possui o segundo maior grau de confidencialidade. Os próximos serão os de valor imediatamente inferior ao último selecionado, desde que não seja menor do que o valor médio de confidencialidade dos algoritmos.				
3.c: O primeiro algoritmo deve ser o que possui o segundo maior grau de confidencialidade. Os próximo serão os de valor imediatamente inferior ao último selecionado, desde que não seja menor do que o valor médio de confidencialidade dos algoritmos.				

Os algoritmos utilizados foram classificados por seus graus de confidencialidade, e quanto maior o grau de confidencialidade do algoritmo, mais seguro e mais complexo ele é. Na medida em que diminui o grau dos algoritmos utilizados, diminui também o consumo de bateria proporcionado pelo seu uso. Estas características de consumo foram verificadas na 4MeSAD após a execução da aplicação de diagnósticos nos DMs utilizados na avaliação, conforme detalhado na Seção 6.3. Estes algoritmos estão relacionados na Tabela 4, e a ordem dos códigos coincide com a ordem decrescente de seus graus de confidencialidade.

Tabela 4 – Algoritmos selecionados para uso na aplicação

Código	Algoritmo	Regras atendidas
1	SERPENT	3.a
2	AES	2.a, 2.b, 3.b, 3.c
3	TWOFISH	2.b, 3.b, 3.c
4	CAST6	2.b e parcialmente: 3.b, 3.c
5	RC6	2.b e parcialmente: 3.b, 3.c
6	AES FAST	1.a e parcialmente: 2.b
7	RC5 32	1.a e parcialmente: 2.b
8	BLOWFISH	1.a e parcialmente: 2.b
9	RC5 64	1.a, 1.b, 1.c

A Tabela 4 também mostra quais regras cada algoritmo passou a atender após a configuração realizada na 4MeSAD. Nos gráficos das próximas seções, os algoritmos foram referenciados pelos códigos expostos nesta tabela.

A palavra “parcialmente”, encontrada na Tabela 4, indica que este algoritmo atende parcialmente a regra quando considerada a ordem de seleção, pois em determinados contextos as regras atuam em conjunto. Assim, este algoritmo pode contribuir para o atendimento da regra se for usado em conjunto com outros, e desde que ele não seja a primeira escolha do MeSAD. Por exemplo, o DM está utilizando um determinado algoritmo em uma rede de confiança alta e passa a utilizar uma rede de média confiança, selecionando um algoritmo mais forte. O algoritmo que estava em uso inicialmente não pode continuar sendo utilizado na nova rede, mas pode ser utilizado em conjunto com outros mais fortes do que ele.

Pode-se notar que os algoritmos estão classificados de maneira a atender às necessidades expostas por este trabalho. Por exemplo, o algoritmo mais forte está reservado para atender a regra 3.a, a qual atende ao requisito de fornecimento de alta confidencialidade. Ele será selecionado quando a aplicação estiver sendo utilizada em uma rede de baixa confiança e possuir um nível alto de bateria. Assim como o algoritmo mais econômico está atendendo às regras 1.a, 1.b e 1.c, que atendem aos requisitos de menor consumo de bateria em ambientes mais seguros.

A validação do MeSAD será satisfatória se o mecanismo for capaz de identificar as informações de contexto que necessita, e se for capaz de selecionar o algoritmo correto para uso, em tempo de execução e de forma dinâmica. Esta seleção deverá ser de acordo com as regras acima estabelecidas para a adaptação, pois foram criadas para atender às necessidades de alto grau de confidencialidade nos ambientes mais inseguros e de economia de bateria nos ambientes identificados como de maior confiança.

6.5 Validação do MeSAD

A abordagem de validação adotada levou em consideração a complexidade envolvida na verificação de parâmetros que influenciam na análise do grau de confidencialidade proporcionada. Tal complexidade está na adoção de técnicas de identificação de fraquezas como a criptoanálise (linear e diferenciada), a complexidade de chaves em uso, as tentativas de quebra por força bruta, entre outras envolvidas em metodologias que tratam verificação de criptografia de dados. Existem ainda dificuldades que envolvem a medição precisa do consumo de energia proporcionado pelo uso do MeSAD, tais como: energia gasta para transmissão das informações, características de transmissão que influenciam no consumo, distância do DM até o ponto de acesso (que influencia na potência do sinal emitido, impactando em maior ou menor consumo da bateria), entre outros diversos fatores. Tais variantes tornariam impraticável a completa verificação e validação do MeSAD, com uma metodologia que avaliasse todos os parâmetros envolvidos no fornecimento dinâmico de confidencialidade em tempo de execução. Isso pelo fato da restrição de tempo, de recursos e de custos envolvidos com o desenvolvimento deste trabalho.

As verificações do MeSAD foram organizadas em tabelas e divididas da seguinte forma:

a) verificações realizadas em rede de alta confiança do usuário (ALTO), consumindo a bateria até atingir todos estados tratados: acima do *threshold*, abaixo do *threshold* e acima do valor crítico, e abaixo do valor crítico (representados por A, B e C, respectivamente);
b) verificações realizadas em redes de média e baixa confiança do usuário (MÉDIO E BAIXO), consumindo bateria até atingir todos estados (A, B e C);
c) verificações realizadas mudando as redes em uso, com confiança alta, média e baixa, e em estados diferentes de bateria.

Esses dados estão representados em tabelas indicando cada caso mencionado, e as seleções dos algoritmos estão representadas pelos seus códigos entre chaves. Para exemplificar, quando houver {2,3,4,5} significa que o primeiro algoritmo selecionado foi o de código 2, na adaptação seguinte foi o algoritmo 3, depois o 4 e por último o 5. Permanecendo o 5 até uma mudança de contexto ou encerramento da verificação na etapa selecionada.

Quando houver dois conjuntos, como {2,3,4,5} e {3,4,5}, significa que são dois resultados identificados, e no segundo caso, o primeiro algoritmo foi o de código 3.

As mudanças de estado de bateria podem indicar continuidade do passo de verificação, e os algoritmos selecionados farão parte da sequência de adaptação verificada anteriormente. Por exemplo, quando na coluna A houver {2,3,4,5} e {3,4,5}, e na coluna B for {9} e {8,9}, significa que no estado de bateria A estes dois conjuntos de algoritmos foram utilizados. Quando o estado de bateria mudou para B, o algoritmo de código 9 foi escolhido para continuar a primeira verificação e o algoritmo de código 8 para continuar a segunda. Note que, na segunda verificação, o MeSAD ainda mudou o algoritmo para o de código 9.

Como em determinados momentos o MeSAD faz escolhas aleatórias de um subgrupo de algoritmos, foi necessário a repetição de alguns cenários até que as verificações proporcionassem a cobertura de todos os casos de seleção dos algoritmos.

Duas técnicas de verificação da cobertura dos algoritmos utilizados foram basicamente utilizadas: uma com o intuito de produzir a mesma faixa de consumo em cada estado de bateria verificado, e outra com o intuito de verificar se os algoritmos corretos seriam selecionados, mesmo antes da utilização de todos os algoritmos disponíveis no estado de bateria anterior.

Na primeira, a cada consumo de 1% de bateria foi verificada a necessidade de uma nova seleção de algoritmo. Por esse motivo, como no caso em que o maior número de algoritmos selecionados será um conjunto de sete algoritmos, que é o caso do uso em uma rede com confiança média e bateria no estado B, foi definido que um processo de verificação consumirá 7% de bateria em cada situação na qual uma regra diferente deve ser aplicada. Para exemplificar, a primeira linha da Tabela 5 possui três amostras {6,7,8,9}, {7,8,9} e {8,9}, nas quais o algoritmo 9 teve que consumir 4% de bateria na primeira amostra, 5% na segunda e 6% na terceira, encerrando a etapa de verificação com 7% de consumo em cada regra atendida. Esta equiparação dos valores de consumo, para verificação do atendimento de cada regra, possibilitará a realização de uma comparação entre alguns algoritmos na Seção 6.6.

A segunda técnica também utilizou uma faixa de consumo de 1% para realizar a verificação de nova adaptação. Contudo, como o intuito era verificar a mudança do algoritmo antes da utilização de todos os algoritmos disponíveis para atendimento da regra, as mudanças de estado de bateria aconteceram mais cedo. Isto permitiu que fossem verificados todos os casos de mudança de algoritmos, porém aumentou também o número de verificações necessárias para esta verificação. Estas seleções estão representadas nas tabelas com um

asterisco (*) antes da chave que mostra os algoritmos selecionados, para diferenciá-las das seleções nas quais a primeira técnica foi utilizada.

A Tabela 5 mostra os resultados das etapas de verificação do MeSAD, realizadas para atender as regras de adaptação quando o DM esteve em uso em uma rede classificada pelo usuário como de confiança alta. Os estados de bateria A, B e C foram avaliados isoladamente até o consumo de 7% da energia da bateria, e em conjunto, para verificar se no momento de mudança dos estados da bateria o MeSAD atua de maneira correta na adaptação da confidencialidade de acordo com o contexto identificado.

A Tabela 5 mostra que o MeSAD atendeu satisfatoriamente a todas as regras, adaptando os algoritmos corretos em cada situação. As escolhas aleatórias dos algoritmos, na coluna do estado de bateria A, mostram a cobertura de todas as possibilidades de seleção, as quais foram possíveis de serem verificadas após a repetição da verificação até que todos os casos fossem identificados.

Tabela 5 – Resultados da verificação do MeSAD em uma rede de alta confiança (ALTO*)*

CONFIANÇA NA REDE	ESTADO DA BATERIA			Regra a seguir	Algoritmos selecionados pelo MeSAD		
	A	B	C		em (A)	em (B)	em (C)
ALTO	X			1.a	{6,7,8,9}, {7,8,9}, {8,9}		
		X		1.b		{9}	
			X	1.c			{9}
	X	X		1.a, 1.b	{6,7,8,9}, {7,8,9}, {8,9}	{9}, {9}, {9}	
	X	X	X	1.a, 1.b, 1.c	{6,7,8,9}, {7,8,9}, {8,9}, *{6,7,8}, *{7,8}, *{8}, *{6,7}, *{7}, *{6}	{9}, {9}, {9}, *{9}, *{9}, *{9}, *{9}, *{9}, *{9}	{9}, {9}, {9}, *{9}, *{9}, *{9}, *{9}, *{9}, *{9}
		X	X	1.b, 1.c		{9}	{9}

Os casos, nos quais se verifica um asterisco (*) na sequência selecionada, indicam que estas escolhas estão representando uma situação onde a verificação que está sendo feita é o atendimento da segunda técnica mencionada anteriormente. Então, se no momento de uso do último algoritmo da sequência houver uma mudança de estado da bateria, o MeSAD vai selecionar o algoritmo correto no novo contexto, mesmo que os algoritmos restantes do contexto anterior ainda não tiverem sido utilizados. Também neste cenário o MeSAD atuou de maneira satisfatória, mudando sempre para o algoritmo adequado à regra do contexto identificado.

As verificações realizadas na rede com confiança média estão representadas na Tabela 6, a qual também representa o DM sendo verificado nos estados de bateria A, B e C isoladamente e em conjunto, após o consumo de 7% da energia da bateria.

Pode-se notar na Tabela 6 que o MeSAD também atendeu corretamente todas as regras, tanto quando o estado da bateria não se modificou para os casos B e C, como quando esta mudança aconteceu.

Nota-se que, na Tabela 6, não há todas as possibilidades de sequências de algoritmos selecionados representando a segunda técnica de verificação mencionada (marcadas com um asterisco). Isto acontece pelo fato de que algumas sequências presentes nesta verificação já estarem representadas em algum momento anterior, seja na própria tabela ou na Tabela 5, na qual a técnica é realizada, não havendo necessidade de repetição dessas verificações.

Tabela 6 – Resultados da verificação do MeSAD em uma rede de média confiança (MÉDIO*)*

CONFIANÇA NA REDE	ESTADO DA BATERIA			Regra a seguir	Algoritmos selecionados pelo MeSAD		
	A	B	C		em (A)	em (B)	em (C)
MÉDIO	X			2.a	{2}		
		X		2.b		{2,3,4,5,6,7,8}, {3,4,5,6,7,8}, {4,5,6,7,8}, {5,6,7,8}	
			X	2.c			{9}
	X	X		2.a, 2.b	{2}	{3,4,5,6,7,8}	
	X	X	X	2.a, 2.b, 2.c	{2}, {2}, {2}, {2}	{3,4,5,6,7,8}, *{3,4,5}, *{3,4}, *{3}	{9}, *{9}, *{9}, *{9}
		X	X	2.b, 2.c		{2,3,4,5,6,7,8}, {3,4,5,6,7,8}, {4,5,6,7,8}, {5,6,7,8}, *{2,3,4,5,6,7}, *{2,3,4,5,6}, *{2,3,4,5}, *{2,3,4}, *{2,3}	{9}, {9}, {9}, {9}, *{9} , *{9} , *{9} , *{9} , *{9}

A Tabela 7 apresenta as etapas de verificação do MeSAD atuando em uma rede de confiança baixa nos três (3) estados de bateria. A atuação acontece de forma isolada, ou seja, atuando apenas em um estado de bateria de cada vez, e em conjunto com outros estados, conforme as verificações realizadas nos contextos anteriores. Nota-se que aqui também foram atendidas as regras de adaptação, e os algoritmos selecionados correspondem aos algoritmos que deveriam ser utilizados no contexto identificado.

Na Tabela 7, nota-se que as seleções dos algoritmos não apresentam muitas mudanças em relação ao conjunto de algoritmos que estão sendo selecionados, pois todos correspondem aos algoritmos que possuem seu grau de confidencialidade acima do valor médio de confidencialidade da categoria da aplicação. Isto acontece pelo fato das seleções neste contexto privilegiarem os algoritmos que são mais seguros, pois a rede envolvida na verificação foi avaliada como uma rede da baixa confiança, necessitando do aumento do grau de confidencialidade fornecido para a aplicação.

Tabela 7 – Resultados da verificação do MeSAD em uma rede de baixa confiança (BAIXO*)*

CONFIANÇA NA REDE	ESTADO DA BATERIA			Regra a seguir	Algoritmos selecionados pelo MeSAD		
	A	B	C		em (A)	em (B)	em (C)
BAIXO	X			3.a	{1}		
		X		3.b		{2,3,4,5}	
			X	3.c			{2,3,4,5}
	X	X		3.a, 3.b	{1}	{2,3,4,5}	
	X	X	X	3.a, 3.b, 3.c	{1}, {1}, {1}, {1}	{2,3,4,5}, *{2,3,4}, *{2,3}, *{2}	{5}, *{5}, *{4,5}, *{3,4,5}
		X	X	3.b, 3.c		{2,3,4,5}	{5}

Com o resultado das verificações apresentados nas Tabelas 5, 6 e 7, pode-se constatar que o MeSAD selecionou os algoritmos conforme as regras de adaptação e de acordo com o contexto identificado.

As verificações de mudanças de estado de bateria também testadas e apresentadas nas tabelas, com a identificação através do asterisco (*), mostraram que o MeSAD selecionou o algoritmo correto quando a mudança ocorreu antes da utilização de todos os algoritmos do contexto identificado anteriormente. Isto mostra que quando uma aplicação estiver utilizando um algoritmo em um estado de bateria (A ou B) e esta entrar no seu valor crítico (estado C), por exemplo, o MeSAD irá considerar este novo estado para a nova seleção com base no contexto novo identificado.

As verificações do MeSAD atuando na seleção e uso dos algoritmos até o consumo de bateria de 7%, em cada estado de bateria, também mostraram que todos os algoritmos em todos os contextos foram selecionados de forma correta nas verificações. Estas verificações também possibilitaram a criação de alguns gráficos com a utilização dos experimentos produzidos na ferramenta 4MeSAD. Tais gráficos estão apresentados na próxima seção.

6.6 Comparação no uso de algoritmos criptográficos

A ferramenta 4MeSAD possibilita a análise e a verificação do comportamento dos algoritmos de determinada biblioteca de segurança ao serem executados em DMs. Ela foi utilizada para coletar dados destes algoritmos executados no DM XPeria X10a. Estes dados foram utilizados nesta seção, para fazer uma análise de algumas situações, nas quais os algoritmos foram utilizados pelo MeSAD, e apresentados na Seção 6.5. O intuito é deixar mais claro a diferença

do número de rodadas possíveis relacionadas diretamente com o consumo de energia proporcionado pelos algoritmos criptográficos utilizados no fornecimento de confidencialidade de maneira adaptativa e no modo estático.

A Figura 24 apresenta um gráfico que mostra o número de rodadas possíveis de serem executados pelos algoritmos após o consumo de 7% da bateria (valor de consumo definido na etapa de verificação). Na Figura 24, pode-se verificar o melhor rendimento proporcionado pelo uso da criptografia com o conjunto de algoritmos {6,7,8,9} nos contextos identificados (bateria acima do *threshold* e MeSAD atuando em três redes com grau de confianças diferentes), pois o contexto identificado permite o uso dos algoritmos menos complexos. Isto acontece porque como o MeSAD leva em consideração a prática da computação verde, em determinados momentos suas escolhas levarão em conta também a economia de energia. Assim, como os algoritmos selecionados consomem menos bateria, podem ser executados um número maior de vezes até consumir a mesma quantidade de energia que os outros algoritmos (2 e 1) nas outras redes.

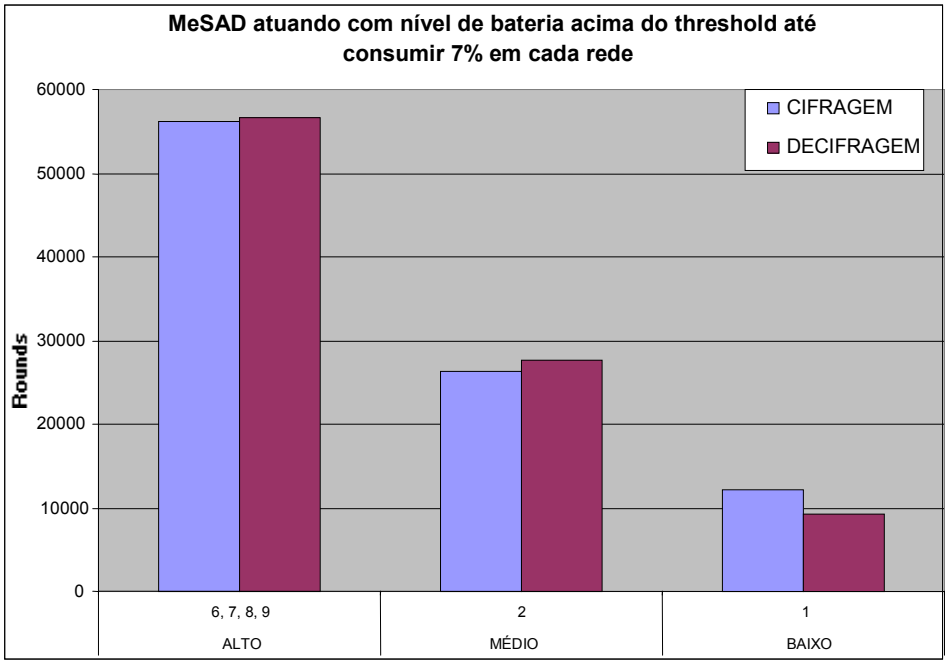


Figura 24 – Rodadas executadas até o consumo de 7% da energia da bateria

Na Figura 24, ainda pode ser verificado que o uso dos algoritmos de maiores complexidades provoca menor rendimento. Porém, isto é necessário devido à confiança do usuário ser menor no contexto identificado. O uso dos algoritmos mais complexos está restrito aos ambientes identificados como de maiores riscos, rede de confiança MÉDIO e BAIXO. Nota-se que o menor desempenho ficou restrito ao ambiente considerado como de alto risco, e

nos ambientes mais seguros os algoritmos puderam ser utilizados por mais vezes. O comportamento é justificado, pois o algoritmo de código 1 é o mais forte e a rede é de baixa confiança.

O rendimento apresentado neste e nos próximos gráficos dizem respeito ao número de rodadas que cada algoritmo é capaz de executar, cifrando ou decifrando um texto de 8kBytes durante o tempo necessário para consumir os 7% de bateria. Quando o número de rodadas nos gráficos é maior, significa que o MeSAD possibilitou que o consumo de energia para cada rodada fosse menor. Estes gráficos foram produzidos a partir dos dados coletados na 4MeSAD, a qual armazena informações sobre o número de rodadas que cada algoritmo é capaz de executar até consumir 1% da bateria. Assim para representar os 7% de consumo de um único algoritmo selecionado, uma estimativa foi produzida multiplicando o número de rodadas por sete. Quando a seleção contempla mais de um algoritmo, seguiu-se o mesmo critério utilizado na verificação do MeSAD, no qual quando há menos de 7 algoritmos o último selecionado se repete até consumir os 7% de bateria definidos. Assim, a Figura 24 possui o algoritmo 1 (na rede BAIXO) e o 2 (na rede MÉDIO) consumindo 7% de bateria cada, e os algoritmos {6,7,8,9} (na rede ALTO) consumindo da bateria 1%, 1%, 1% e 4%, respectivamente.

A atuação do MeSAD apresentada na Figura 25 representa a etapa de verificação realizada com o nível de bateria após atingir o valor do limiar de adaptação (*threshold*), mantendo-se, porém, acima do valor crítico. O DM passeia por redes com níveis de confiança diferentes (ALTO, MÉDIO E BAIXO), e o MeSAD seleciona os algoritmos conforme a Figura 25 apresenta ({9}, {5,6,7,8} e {2,3,4,5}). Mais uma vez, a figura mostra a estimativa do número máximo de rodadas que podem ser executados até o consumo de 7% da bateria. Nota-se na Figura 25 que, quando a bateria atinge o *threshold*, o MeSAD passa a considerar a economia na proporção da confiança na rede. Assim, ele passa a utilizar os algoritmos {5,6,7,8} e {9} nas redes onde o usuário identificou como de média e de alta confiança, respectivamente. Contudo, a seleção dos algoritmos mais complexos {2,3,4,5}, não contribui com a diminuição do consumo, porém isso é justificado pela necessidade de incremento do grau de confidencialidade da aplicação na rede de baixa confiança que foi identificada. Ainda assim, nota-se que o desempenho das rodadas na adaptação feita pelo MeSAD é superior ao proporcionado pelo algoritmo mais forte visto anteriormente na Figura 17 (12.173 rodadas de cifragem e 9.254 rodadas de decifragem). De forma análoga, é também superior ao proporcionado pelo algoritmo de código 2 (com 26.376 rodadas de cifragem e 27.713 rodadas

de decifragem). Observa-se que o algoritmo de código 2 também está sendo utilizado em conjunto com os outros mais fortes da lista (ele é o primeiro da seleção {2,3,4,5}).

Na rede de média confiança, pode ser notada (Figura 25) uma melhoria na quantidade de rodadas possíveis de execução, pois o MeSAD passa a utilizar, em conjunto, os algoritmos mais econômicos que atendem a configuração de confiança MÉDIO. Na rede mais confiável (ALTO), o MeSAD utiliza o algoritmo mais econômico, respeitando as regras de adaptação com o foco na economia de bateria para este contexto.

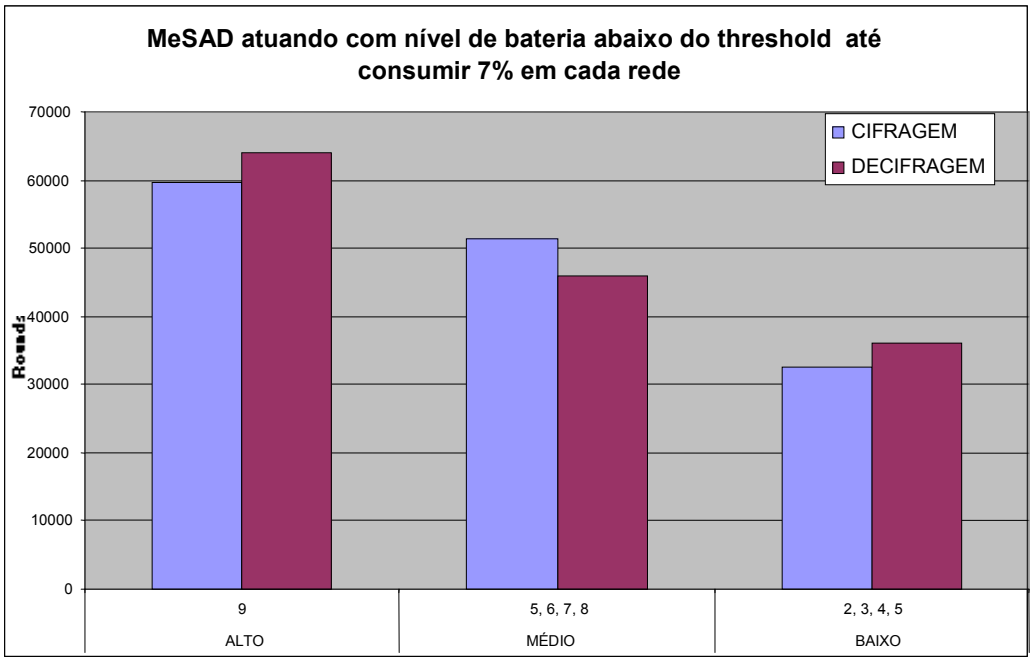


Figura 25 – Rodadas executadas pelos algoritmos até o consumo de 7% da energia da bateria

A Figura 26 apresenta os algoritmos selecionados na verificação do MeSAD realizada com o DM apresentando o nível de bateria abaixo do valor crítico. Neste cenário, a prioridade é a economia de recursos.

Nota-se, no gráfico da Figura 26, que o algoritmo em uso nas redes de média e alta confiança é o algoritmo mais econômico. Contudo, na rede de baixa confiança o MeSAD mantém o respeito à avaliação do usuário e seleciona os algoritmos acima da média de confidencialidade, diminuindo o número de rodadas mas aumentando a segurança no contexto identificado como de preocupação do usuário quanto ao tráfego de seus dados.

É possível perceber que ao aumentar a segurança utilizando os algoritmos 2, 3, 4 e 5, o desempenho caiu. No entanto, esta diminuição é esperada devido ao aumento da complexidade dos algoritmos necessários para aumentar o grau de confidencialidade na rede classificada como de menor confiança para uso (BAIXO).

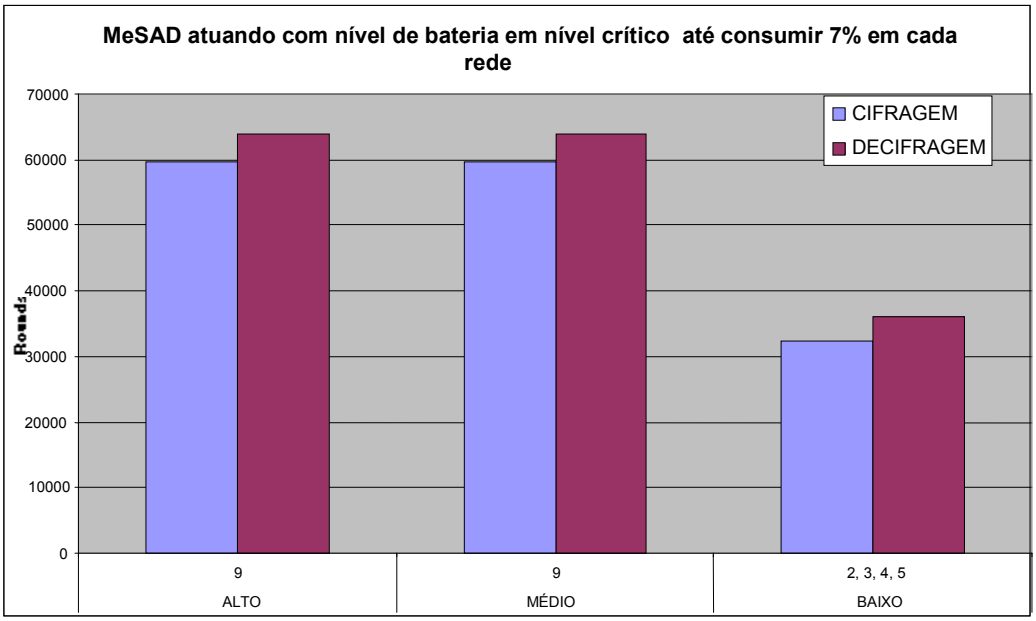


Figura 26 – Rodadas executadas até o consumo de 7% da energia da bateria, quando o MeSAD está atuando com nível de bateria abaixo do valor crítico

Os próximos gráficos foram produzidos para a realização da comparação anunciada no início desta seção. Assim, os gráficos irão comparar o uso da forma estática de provisão de confidencialidade com o uso da seleção dinâmica dos algoritmos realizada pelo MeSAD. A forma estática está representada pela utilização de apenas um algoritmo durante tempo necessário para consumir 7% da energia da bateria. Para a forma dinâmica, foram selecionados grupos de algoritmos já apresentados nas figuras 24, 25 e 26, além de outro grupo que corresponde ao cenário no qual acontece a seleção da maior quantidade de algoritmos (conjunto com os algoritmos {2,3,4,5,6,7,8}, que pode ser visto na verificação realizada na Tabela 6, já apresentada).

O comportamento representado na Figura 27, apresenta o quanto de bateria é consumida pelos algoritmos selecionados pelo MeSAD em alguns contextos já verificados (algoritmos {6,7,8,9}, {2,3,4,5,6,7,8}, {5,6,7,8} e {1}), ao executar o mesmo número de rodadas executados pelo algoritmo de maior grau de confidencialidade ({1}).

Nesta Figura 27, enquanto o algoritmo mais forte consome 7% da bateria para executar 12.173 rodadas de cifragem e 9.254 rodadas de decifragem, os outros algoritmos

consomem bem menos para executar a mesma quantidade de rodadas. Nota-se o impacto causado no consumo de bateria pelo uso do algoritmo mais complexo na provisão de confidencialidade. Por este motivo, a utilização deste algoritmo ficou restrita aos ambientes que o usuário classifica como inseguros. Desta forma, o MeSAD mantém o tratamento adequado da confidencialidade de acordo com esse contexto.

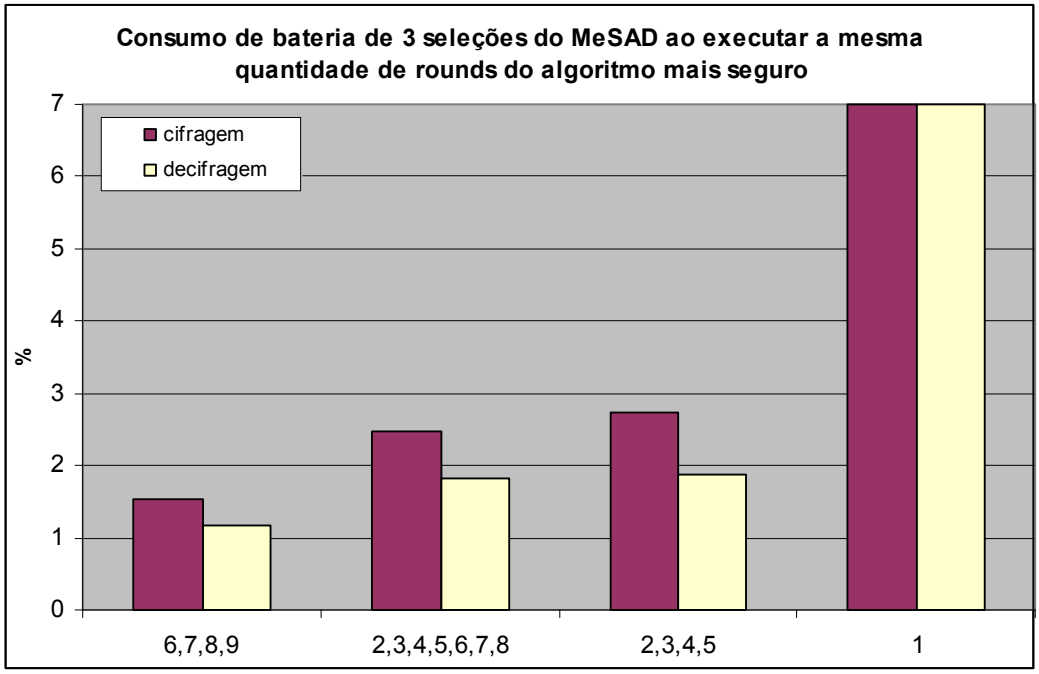


Figura 27 – Consumo de bateria proporcionado pelo MeSAD ao executar o mesmo número de rodadas do algoritmo *Serpent*

Para tornar mais clara a Computação Verde focada na economia de energia defendida pelo MeSAD, se uma aplicação utilizando o algoritmo 1 de maneira estática permanecesse cifrando o texto até consumir 7% da bateria, fosse comparada com outra aplicação idêntica, porém utilizando as seleções vistas na Figura 27, a economia de energia seria de 5,47% na seleção {6,7,8,9}, de 4,52% na seleção {2,3,4,5,6,7,8} e de 4,26% na seleção {2,3,4,5}, para realizar a mesma tarefa.

Para não restringir a comparação ao algoritmo mais complexo, outro algoritmo foi comparado com o mesmo critério de seleção dos grupos e pode ser vista no gráfico da Figura 28.

Desta vez, o algoritmo utilizado foi o vencedor do concurso promovido pelo NIST (*National Institute of Standards and Technology of the United States*) para seleção do algoritmo de criptografia indicado como padrão para uso nos EUA. Este algoritmo é o AES (*Advanced Encryption Standard*) e ele é amplamente adotado para prover criptografia.

A nova comparação apresentada na Figura 28 mostra o número de rodadas executados pelo algoritmo utilizado como padrão pelo NIST, o AES, em conjunto com os utilizados pelo MeSAD e apresentados anteriormente na Figura 27.

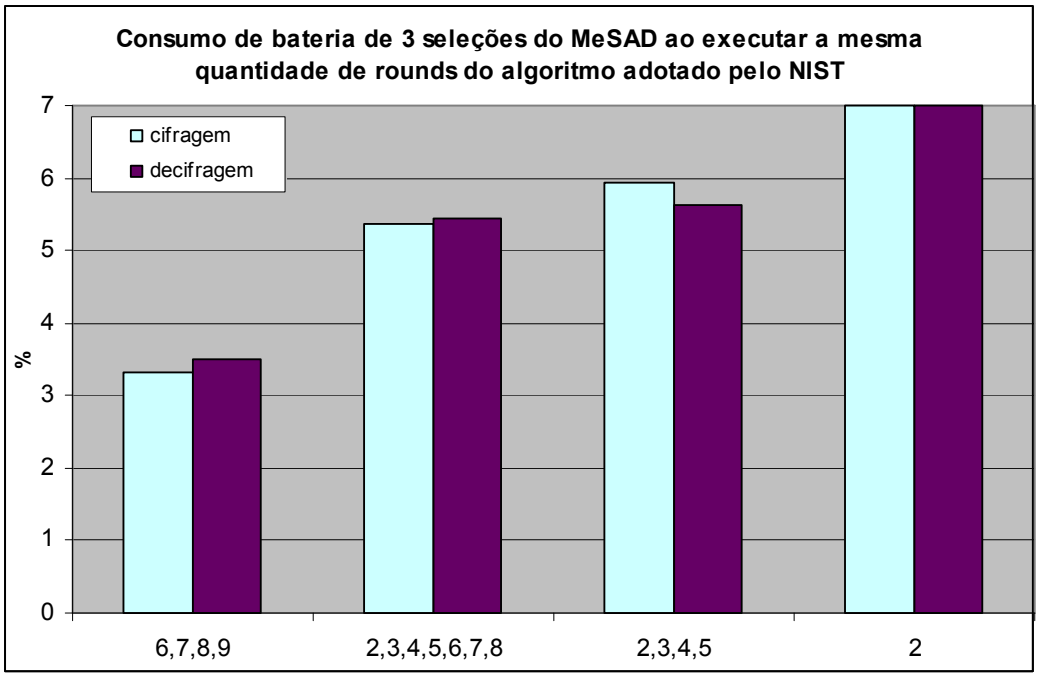


Figura 28 – Consumo de bateria proporcionado pelo MeSAD ao executar o mesmo número de rodadas do algoritmo AES

Conforme foi verificado nas Figuras 27 e 28, os algoritmos mais seguros consomem mais bateria para aplicar confidencialidade. Isso acontece devido ao aumento da complexidade no tratamento da informação proporcionada por estes algoritmos, o que causa um impacto direto no consumo. Por outro lado, algoritmos menos complexos consomem menos bateria, porém não oferecem o mesmo nível de segurança que os mais complexos. Por esse motivo é que o MeSAD considera o uso dos algoritmos mais complexos nos ambientes mais críticos, aumentando a segurança, e aplica os algoritmos de maior economia nos ambientes mais seguros para poupar o consumo de bateria.

A utilização dos algoritmos em cada contexto identificado proporciona a adequação no uso dos recursos do DM e a adequação do grau de confidencialidade a ser aplicado na informação. O uso estático de um algoritmo para a criptografia por vezes desperdiça recursos, ao utilizar criptografia complexa em ambientes que não necessita desta complexidade, e por outras vezes não protege adequadamente a informação em ambientes mais perigosos.

6.7 Conclusão

Este capítulo apresentou a avaliação do MeSAD quanto ao atendimento às necessidades de aplicações que estão preocupadas em utilizar segurança adaptativa para adequar sua segurança ao contexto. A forma como foi utilizada a 4MeSAD para a verificação, escolha e configuração dos algoritmos e da aplicação criada para utilização do MeSAD também foram tratadas no capítulo. A apresentação das regras de adaptação e as análises quanto ao atendimento destas regras foram verificadas através da criação de uma aplicação e da verificação do comportamento de adaptação do MeSAD ao utilizar esta aplicação em um ambiente real. Com isso, esta dissertação além de introduzir um mecanismo de segurança dinâmico que se adapta em tempo de execução para prover confidencialidade às aplicações de DMs, fornece uma ferramenta de apoio às decisões de configuração dos algoritmos que serão utilizados no mecanismo de segurança. Adicionalmente esta ferramenta oferece subsídios para que os seus usuários possam avaliar o comportamento dos algoritmos de determinadas bibliotecas em determinados DMs, e escolher aqueles que melhor atendem aos seus objetivos.

O próximo capítulo trata das conclusões sobre as principais contribuições deste trabalho e sobre os trabalhos futuros identificados.

7 CONCLUSÃO

Esta dissertação apresentou um mecanismo de segurança adaptativa focado na provisão de confidencialidade para aplicações de dispositivos móveis, o qual possui o objetivo de adequar a proteção de acordo com o contexto, considerando o consumo eficiente dos recursos. Adicionalmente, foi apresentada uma ferramenta que serve para dar apoio à utilização do mecanismo de segurança nas aplicações para dispositivos móveis e que verifica o comportamento de algoritmos criptográficos nesses dispositivos.

As principais contribuições deste trabalho e as indicações dos trabalhos futuros compõem o restante deste capítulo.

7.1 Contribuições e Resultados Alcançados

A complexidade envolvida na criação de aplicações para DMs, com a adoção de segurança e com a consideração de prolongamento da disponibilidade da bateria, é um problema difícil de ser resolvido (conforme apresentado nos Capítulos 1 e 4). Apesar de essas dificuldades influenciarem na decisão de uso de confidencialidade nessas aplicações, é importante também a existência de soluções que, além de fornecerem confidencialidade, abordem o uso eficiente dos recursos, assim como faz este trabalho.

A adaptação de segurança em tempo de execução na rede sem fio serve para atenuar as consequências do número substancial de ameaças quando não se pode eliminá-las por completo. Adicionalmente, o tratamento da confidencialidade na camada de aplicação explora as necessidades particulares de cada aplicação e possibilita o emprego da computação verde quanto à alocação eficiente dos recursos dos DMs para o fornecimento de confidencialidade de acordo com o contexto.

Diante deste contexto, o ajuste do nível de segurança de acordo com a análise do ambiente é uma forma de melhorar a segurança para as redes sem fio e ainda proporcionar economia de recursos, prolongando a disponibilidade dos outros serviços do dispositivo.

A principal contribuição desta dissertação, portanto, é a apresentação de um mecanismo de segurança capaz de adequar a confidencialidade de acordo com o contexto,

fornecendo maior segurança em ambientes inseguros e economizando a energia da bateria em ambientes mais seguros.

O MeSAD possibilita que a concepção de novas aplicações que consideram o uso da segurança adaptativa para adequar a confidencialidade ao contexto, possa acontecer de maneira mais simples do que o modo convencional, e ainda poupando os recursos do DM.

As análises quanto ao atendimento das regras de adaptação de segurança do MeSAD constataram a conformidade no atendimento, em todas as situações verificadas e apresentadas no capítulo anterior.

Ainda foi possível obter neste trabalho uma análise sobre a maneira tradicional de utilização de algoritmo de criptografia (maneira estática) realizando uma estimativa de desempenho em *rounds* executados até o consumo de determinada faixa de bateria. Isto permitiu verificar que o MeSAD proporciona um aproveitamento melhor da disponibilidade da bateria e, conseqüentemente, do uso mais prolongado do aparelho.

O uso eficiente dos recursos do DM, notado tanto na economia de bateria nas seleções do MeSAD quanto no desempenho de rodadas apresentados no capítulo 6, coloca o MeSAD em sintonia com as práticas e preocupações da Computação Verde. A medição dos ganhos em consumo proporcionados por cada possível seleção do MeSAD em comparação com cada algoritmo e todas as possibilidades de combinações entre si, seria uma tarefa complexa e custosa para ser realizada em tempo hábil e, por esse motivo não foi realizada, ficando para avaliações futuras. Contudo, é notória a economia de bateria proporcionada pela seleção dos algoritmos criptográficos realizados pelo MeSAD e apresentados no capítulo 6.

É importante destacar que a substituição temporal dos algoritmos contribui para o aumento da segurança da informação, pois a mudança das características de criptografia dificulta a realização de criptoanálise nos dados em tráfego no meio sem fio.

Além do mecanismo de segurança proposto, é importante mencionar que uma contribuição paralela deste trabalho foi a criação da ferramenta de apoio. A 4MeSAD não somente configura o MeSAD como também oferece meios de verificar o comportamento dos algoritmos nos DMs. Os recursos disponíveis na 4MeSAD facilitam o trabalho do engenheiro de *software* ou especialista em segurança, na escolha e na configuração dos algoritmos que melhor atendem suas necessidades. Como a 4MeSAD possibilita a configuração e personalização do MeSAD, por parte do engenheiro de *software* ou especialista em segurança, é importante deixar claro que este indivíduo necessita possuir conhecimento sobre as complexidades dos algoritmos criptográficos. Contudo, uma vez definida, a configuração

pode ser, e provavelmente será, reutilizada para a criação de todas as aplicações do engenheiro de *software*.

Por fim, vale ressaltar que este trabalho obteve publicação de artigo [CIRQUEIRA, 2011] no Seminário Integrado de Software e Hardware (XXXVIII SEMISH), principal fórum científico do XXXI Congresso da Sociedade Brasileira de Computação (CSBC 2011).

7.2 Trabalhos Futuros

A adoção da adaptação dinâmica da confidencialidade com base no contexto pode aumentar a segurança dos dados trafegados no meio sem fio, porém, por si só, não garante a segurança das informações, devido ao crescente número de ameaças existentes neste domínio. Assim esta dissertação aponta algumas fontes relevantes para pesquisa futura.

A agregação de outros serviços de segurança, além da confidencialidade é um trabalho futuro importante, pois tornaria o MeSAD mais robusto no combate ao número substancial de ameaças existentes no meio sem fio.

Um estudo mais aprofundado sobre as características de todos os algoritmos criptográficos disponíveis nas bibliotecas de segurança, quanto as suas complexidades no fornecimento da proteção dos dados, é um direcionamento de trabalho futuro que agregaria um valor substancial a este trabalho e serviria para a definição mais precisa dos pesos utilizados na 4MeSAD.

A necessidade de uma interação com o usuário das aplicações para fornecer uma recomendação sobre sua avaliação quanto à confiança que ele deposita nas suas redes é um ponto que pode ser aperfeiçoado neste trabalho. No trabalho, esta recomendação acontece de forma indireta, mostrando as características da rede em uso no momento da avaliação. Apesar disso, o usuário pode não levar em consideração estes aspectos. Assim, a recomendação refinada desta avaliação, teria o propósito de aumentar a precisão desta informação. Esta acurácia na captura da informação de confiança na rede ofereceria um melhor aproveitamento no uso dos recursos no momento da instanciação dos algoritmos criptográficos.

Por fim, a composição do repositório local do MeSAD com outras bibliotecas de segurança, também seria outro trabalho futuro importante.

8 REFERÊNCIAS BIBLIOGRÁFICAS

ANDROID. Google Copyright Android ®. Disponível em: http://www.android.com/. Acesso em: 01 out. 2010.

ATAY S., MASERA M. (2010). **Challenges for the security analysis of Next Generation Networks**. Information Security Technical Report (2010), doi:10.1016/j.istr.2010.10.010.

BAZIRE, M., BREZILLON, P. (2005). **Understanding Context Before Using It**. Lecture Notes in Computer Science, 2005, Volume 3554, Modeling and Using Context, Pages 29-40. 5th International and Interdisciplinary Conference CONTEXT 2005, Paris, France, July 5-8. Proceedings.

BAPATLA, S., CHANDRAMOULI, R. (2004). **Battery Power Optimized Encryption**. In: Proced. IEEE International Conference on Communications 7, 3802–3806.

BISHOP, M., **Computer Security: Art and Science**, Addison-Wesley, 2003.

BOSCH, Jan. **Design and Use of Software Architecture**, Addison-Wesley, 2000

BRAGG, R., RHODES-OUSLEY M., STRASSBERG K. et al. (2004). **Network Security: The Complete Reference**. McGraw-Hill.

BRINGEL, J. R. M. F., VIANA, W., ANDRADE, R. M. C. (2003). **Pearl: a performance evaluator of cryptographic algorithms for mobile devices**. In: The First International Workshop on Mobility Aware Technologies and Applications (MATA), 275-284.

BRINGEL, J. R. M. F. (2004). FRAMESEC: **Um Framework para a Provisão de Segurança Fim a fim para Aplicações no Ambiente de Computação Móvel**. 2004, 149p. Dissertação (Mestrado em Ciência da Computação) – Universidade Federal do Ceará (UFC), Fortaleza, Ceará.

BLANCHARD, C. W. (2001). **Wireless Security**. BT Technology Journal, vol. 19, No. 3, 67 – 75. Springer Netherlands. Julho. doi=10.1023/A:1011986115075.

BROWN, P. J., BOVEY, J. D., CHEN, X. (1997). **Context-Aware Applications: From the Laboratory to the Marketplace**. IEEE Personal Communications, 4(5), 58–64.

CAMPBELL, A. T., COULSON, G., KOUNAVIS, M. E., (1999). **Managing complexity: Middleware explained**. IT Professional, IEEE Computer Society, pp. 22–28, Setembro.

CARVALHO, A. F. M. (2008). M-CODE: **Um Modelo para Medição de Confidencialidade e Desempenho para Aplicações Móveis Seguras**. 2008. 96p. Dissertação (Mestrado em Ciência da Computação) – Universidade Federal do Ceará (UFC), Fortaleza, Ceará.

CHANDRAMOULI, R., BAPATLA, S., SUBBALAKSHMI, K. P., UMA, R. N (2006). **Battery Power-Aware Encryption**. ACM Transaction Inf. Systems Security, vol. 9, no. 2, pp. 162–180.

CHEN, G., KOTZ, D. (2000). A **Survey of Context-Aware Mobile Computing Research.** Technical Report. Dartmouth College. Hanover, NH, USA.

CHEN, J., HU, C., ZENG, H., (2010). **A Novel Model for Evaluating Optimal Parameters of Security and Quality of Service**. Journal of Computers, vol 5, no. 6.

CIRQUEIRA, A. C., ANDRADE, R. M. C., CASTRO, M. F. (2011). **Um Mecanismo de Segurança com Adaptação Dinâmica em Tempo de Execução para Dispositivos Móveis**. In: XXXI Congresso da Sociedade Brasileira de Computação (CSBC 2011), Natal, RN-Brasil. Anais do XXXVIII Seminário Integrado de Software e Hardware (SEMISH), Julho.

COULOURIS, G., DOLLIMORE J., KINFBERG T. (2005). **Distributed System – Concepts and Design**. Addison-Wesley.

DAEMEN J., RIJMEN V. **The Design of Rijndael. AES – The Advanced Encryption** Standard. Springer, 2001.

DARCO, P., DE SANTIS, A., FERRARA, A. L., MASSUCCI, B. (2010). **Variations on a theme by Akl and Taylor: Security and tradeoffs**. Theor. Comput. Sci. 411, 1 (Jan. 2010), 213-227. DOI= http://dx.doi.org/10.1016/j.tcs.2009.09.028.

DEY, A. K. (2001). **Understanding and Using Context**. Personal and Ubiquitous Computing Journal, v. 5, n. 1, pp. 4-7.

DOOMUN, M. R., SOYJAUDAH, K. M. S. (2007). **Adaptive IEEE 802.11i Security for Energy-Security Optimization**. Telecommunications, 2007. AICT 2007. The Third Advanced International Conference, pp.38. doi:10.1109/AICT.2007.3

ECLIPSE. Copyright © 2011 The Eclipse Foundation. All Rights Reserved. Disponível em: http://www.eclipse.org/org/. Acesso em: 01 fev. 2010.

EISENBARTH, T., KUMAR, S., PAAR, C., POSCHMANN, A., USHADEL, L. (2007). **A Survey of lightweight-cryptography implementations**. IEEE Design and Test of Computers, vol. 24, No 6, pp. 522-533. Nov.

ELKHODARY, A., WHITTLE, J. (2007). **Survey of Approaches to Adaptive Application Security**. ICSE Workshops SEAMS '07. pp. 16-16.

FENTON, N. (1994). **Software Measurement: A Necessary Scientific Basis**. IEEE Transactions on Software Engineering, v. 20, n. 3, pp. 199-206.

GALLEN, Christine. **Mobile Consumer Electronics Device Shipment to Grow 55-Fold in Six Years**. ABI Research News. Disponível em: http://www.abiresearch.com/press/1572. Acesso em: 27 jan. 2010.

GARLAN, D. (2000). **Software Architecture: A Roadmap. In The Future of Software Engineering**. 22nd International Conference on Software Engineering, ICSE 2000; University of Limerick, Ireland, 4-11 June.

GENTRY, C., RAMZAN, Z. (2004). **Provable cryptographic security and its applications to mobile wireless computing**. Wireless Personal Communications, vol. 29 (3-4 SPEC.ISS.), pp. 191-203. Jun.

GEORGAS, J.C., HOEK A.V.D., TAYLOR R.N. (2005). **Architectural Runtime Configuration Management in Support of Dependable Self-Adaptive Software**. WADS, vol. 30, pp. 1-6, July.

HAMAD, F., SMALOV, L., JAMES, A. (2009). **Energy-aware Security in M-Commerce and the Internet of Things**. IETE (Institution of Electronics and Telecommunication Engineers) Tech Rev. IETE Journals, vol. 26, Issue 5, pp. 357-62. Department of Computing and the Digital Environment, United Kingdom.

HERRICK, D. R., RITSCHARD, M. R. (2009). **Greening Your Computing Technology, the Near and Far Perspectives.** In Proceedings of the ACM SIGUCCS Fall Conference on User Services Conference (St. Louis, Missouri, USA, SIGUCCS '09. ACM, New York, NY, 297-304. http://doi.acm.org/10.1145/1629501.1629557.

HILTUNEN, M. A., SCHLICHTING R. D. (1996). **Adaptive Distributed and Fault-Tolerant Systems**. International Journal of Computer Systems Science and Engineering, vol. 11, pp. 125–133, Setembro.

IZQUIERDO, A., SIERRA, J. M., TORRES, J. (2006). **On the Implementation of Security Policies with Adaptative Encryption**. Journal Computing Communications vol. 29, Issue 15, pp. 2750–2758, Setembro. doi=http://dx.doi.org/10.1016/j.comcom.2005.10.026

IZQUIERDO, TORRES, J., A., SIERRA, J. M., CARBONELL, M. (2007). **Using Adaptative Encryption for Ubiquitous**. O. Gervasi and M. Gavrilova (Eds.): ICCSA 2007, LNCS 4706, Part II, pp. 540–548, 2007. Springer-Verlag Berlin Heidelberg.

J2ME, Java 2 Micro Edition. Copyright © 2010, Oracle Corporation and/or its affiliates Disponível em: http://www.java.com. Acesso em: 15 mar. 2010.

JEE, Java Enterprise Edition. Copyright © 2010, Oracle Corporation and/or its affiliates. Disponível em: http://www.java.com. Acesso em: 10 jan. 2011.

JIAN R, (2000). **Art of Computer Systems Performance Analysis Techniques For Experimental Design Measurements Simulation And Modeling**. Wiley Computer Publishing, John Wiley & Sons, Inc. ISBN: 0471503363.

KANDÉ, M. M., STROHMEIER, A. (2000). **On The Role of Multi-Dimensional Separation of Concerns in Software Architecture**. OOPSLA'2000 Workshop on Advanced Separation of Concerns. Swiss Federal Institute of Technology Lausanne.

KARRI, R., MISHRA, P. (2003). **Minimizing The Secure Wireless Session Energy**. Journal of Mobile Network and Applications (MONET) 8, 2 (Abril), 177–185.

KNAPSKOG, S.J. (2008). **New cryptographic primitives** (plenary lecture) Proceedings - 7th Computer Information Systems and Industrial Management Applications, CISIM 2008, art. No. 4557828, pp. 3-7. Jun.

LAMPRECHT, C. J., VAN MOORSEL, A. P. A. (2008). **Runtime Security Adaptation Using Adaptive SSL**. Dependable Computing, 2008. PRDC '08. 14th IEEE Pacific Rim International Symposium, pp. 305-312, Dezembro. doi: 10.1109/PRDC.

LASHKARI A. H., DANESH, M. M. S. (2009). **A Survey on Wireless Security protocols (WEP,WPA and WPA2/802.11i)**. Computer Science and Information Technology. ICCSIT 2009. 2nd IEEE International Conference, pp.48-52, 8-11 Agosto. doi: 10.1109/ICCSIT.2009.5234856.

LI, H., DHAWAN, A. (2004). **Agent Based Multiple Level Dynamic Multimedia Security System**. Information Assurance Workshop, 2004. Proceedings from the Fifth Annual IEEE SMC, pp. 291-297, 10-11 Junho.doi: 10.1109/IAW.2004.1437830

LI, H. (2006). **Multilevel Adaptive Security System**.92p. Thesis (Doctor of Philosophy in Computer Engineering). New Jersey Institute of Technology (NJIT), USA.

LIU H., PARASHAR M., HARIRI, S. (2004). **A Component-based Programming Model for Autonomic Applications**. In: Proceedings of the International Conference on Autonomic Computing (ICAC-04). (New York, NY), Maio.

MAES, P., (1987). **Concepts and experiments in computational reflection.** in Proceedings of the ACM Conference on Object-Oriented Languages (OOPSLA), pp. 147–155, ACM Press, Dezembro.

MALIKI, T. E., SEIGNEUR, J. (2010). **A Security Adaptation Reference Monitor (SARM) for Highly Dynamic Wireless Environments**. Emerging Security Information Systems and Technologies. In Proceedings of the 2010 Fourth International Conference on Emerging Security Information, Systems and Technologies (SECURWARE '10). IEEE Computer Society, Washington, DC, pp. 63-68, 18-25. DOI=10.1109/SECURWARE.2010.18 http://dx.doi.org/10.1109/SECURWARE.2010.18

MATSUI, M. (1993). **Linear Cryptanalysis Method for DES Cipher**. Advances in Cryptology - Eurocrypt. 386–397.

MCKINLEY, P. K., SADJADI, S.M.; KASTEN, E. P., CHENG, B. H. C. (2004). **Composing Adaptive Software**. Computer vol.37, no. 7, pp. 56- 64. doi: 10.1109/MC.2004.48. Julho.

MEDVIDOVIC N., EDWARDS G. (2010). **Software architecture and mobility: A roadmap**. J. Syst. Softw. 83, 6 (June 2010), 885-898. doi=10.1016/j.jss.2009.11.004.

MENDES, Antonio. (2002). **Arquitetura de Software: Desenvolvimento Orientado para Arquitetura**. Rio de Janeiro: Campus. 212 p.

MINNIS, B. J., MOORE, P. A., WHATMOUGH, P.N., BLANKEN, P.G., HEIJDEN M.P.V.D. (2009). **System-efficiency analysis of power amplifier supply-tracking regimes in mobile transmitters.** IEEE Transactions on Circuits and Systems. In: Regular Papers, vol. 56, No 1, pp. 268-279. Jan.

MySQL, Copyright © 2010, Oracle Corporation and/or its affiliates. Disponível em: http://www.mysql.com/. Acesso em: 01 fev. 2011.

NIRANJAN, G., A. (2004). **Adaptive link layer security for wireless networks (ALL-Sec)**. Military Communications Conference, 2004. MILCOM 2004. IEEE, pp. 153-159 Vol. 1, 31 Oct.-3 Novembro. doi: 10.1109/MILCOM.2004.1493261.

NIST - National Institute of Standards and Technology. FIPS 197. <http://www.nist.gov>. Acesso em: 02 jan. 2010.

NIST - National Institute of Standards and Technology. FIPS PUB 199. <http://www.nist.gov>. Acesso em: 10 jan. 2010.

NIST - National Institute of Standards and Technology. FIPS PUB 140-2. <http://www.nist.gov>. Acesso em: 10 jan. 2010.

OMG – Object Management Group (2011). **Unified Modeling Language, Superstructure Specification**, version 2.4, número do documento: ptc/2010-11-14. Disponível em: http://www.omg.org/spec/UML/2.4/. Acesso em: 19 Set. 2011.

OREIZY, P., MEDVIDOVIC N., TAYLOR, R. N. (2008). **Runtime Software Adaptation: Frameworks, Approach, and Styles**. In Companion of the 30th International Conference on Software Engineering (Leipzig, Germany, May 10 - 18, 2008). ICSE Companion '08. ACM, New York, NY, pp 899-910. DOI= http://doi.acm.org/10.1145/1370175.1370181.

OSSHER, H., TARR, P. (2000). **Multi-Dimensional Separation of Concerns and The Hyperspace Approach**. In: Proceedings of the Symposium on Software Architectures and Component Technology: The State of the Art in Software Development.

PIRMEZ, M. (2009). **Prometheus: Um Serviço de Segurança Adaptativa**. 2009. 115pp. Dissertação (Mestrado em Ciência da Computação) – Universidade Federal do Rio de Janeiro, Rio de Janeiro, RJ.

PONEMON, L. Ponemon **Security Megatrends Survey**. Ponemon Institute LLC. Disponível em: http://www.ponemon.org/local/upload/fckjail/generalcontent/18/file, acesso em: 27 Jan. 2010.

PORTMANN, M., SENEVIRATNE A. (2001). **Selective Security for TLS**. Networks, 2001. Proceedings. Ninth IEEE International Conference pp. 216- 221, 10-12. Outubro. doi: 10.1109/ICON.2001.962343

PRASITHSANGAREE, P., KRISHNAMURTHY, P. (2003). **Analysis of Energy Consumption of RC4 and AES algorithms in wireless lans**. In Proc. IEEE Global Telecommunications Conference 22, 1 (Dezembro). 1445–1449.

ROCHA, B. P. S., COSTA, D. N. O., MOREIRA, R. A. M., LOUREIRO, C. G. R., LOUREIRO, A. A. F., BOUKERCHE A. (2010). **Adaptive security protocol selection for mobile computing**. Journal of Network and Computer Applications, Volume 33, Issue 5, pp. 569-587, ISSN 1084-8045, 10.1016/j. London. Janeiro.

ROCHA, L. S., CASTRO, C. E. P. L., MACHADO, J. C., ANDRADE, R. M. C.(2007). **Utilizando Reconfiguração Dinâmica e Notificação de Contextos para o Desenvolvimento de Software Ubíquo**. In: XXI Simpósio Brasileiro de Engenharia de Software, João Pessoa.

ROCHA, L. S. (2007). **AdaptiveRME e AspectCompose: Um Middleware Adaptativo e um Processo de Composição Orientado a Aspectos para o Desenvolvimento de Software Móvel e Ubíquo**, 2007, 103p. Dissertação (Mestrado em Ciência da Computação) – Universidade Federal do Ceará, Fortaleza.

SALIDO, J., LAZOS, L., POOVENDRAN R. (2008). **Energy and bandwidth-efficient key distribution in wireless Ad Hoc networks: A cross-layer approach**. IEEE/ACM Transactions on Networking, vol. 15, No. 6, pp. 1527-1540. DEC.

SBC (2006). **Grandes desafios da pesquisa em computação 2006 – 2016**. Computação Brasil, Porto Alegre, Ano .7, n. 23, set./out./nov.

SCHILIT, B.; ADAMS, N.; WANT, R. (1994). **Context-aware computing applications**. In: IEEE Workshop on Mobile Computing Systems and Applications. 1., 1994, Santa Cruz, California. Anais do 1º IEEE Workshop on Mobile Computing Systems and Applications. IEEE Computer Society Press, 1994. p 85-90.

SCHNEIER, B., WHITING, D. (2000). **A Performance Comparison of the Five AES Finalists**. In Proceedings of AES Candidate Conference'2000. pp.123~135.

SEAM. Copyright © 2009, Red Hat Middleware, LLC. All rights reserved. JBoss and Seam are registered trademarks and servicemarks of Red Hat, Inc. {Privacy Policy}. Disponível em: http://seamframework.org. Acesso em: 01 mar. 2011.

SMITH, B. C., (1982). **Reflection and Semantics in a Procedural Language**. Technical Report 272, Massachusetts Institute of Technology, MIT. Laboratory for Computer Science, Janeiro.

SOLYMAN, S. H., OMARI M. (2004). **An Efficient Application of a Dynamic Crypto System in Mobile Wireless Security**. WCNC/IEEE Communication Society, vol.2, pp. 837-842, March.

SON, S. H., ZIMMERMAN, R., HANSSON, J. (2000). **An adaptable security manager for real-time transactions**. Real-Time Systems, 2000. Euromicro RTS 2000. 12th Euromicro Conference, pp.63-70. doi: 10.1109/EMRTS.2000.853993.

STEPHEN, Ruth. (2009). **Green IT – More Than a Three Percent Solution**. IEEE Internet Computing, vol. 13, no. 4, pp. 74-78, July/Aug., doi:10.1109/MIC.2009.82.

SUDHARSSUN, S., SAKTHI V. R., ANERUDHAN, G. (2010). **Intelligent Power Management and Improved Security in Bluetooth Devices Through Adaptive Range and Encrypted Hopping**. Wireless Communication and Sensor Computing. ICWCSC 2010. International Conference pp.1-5, 2-4 Jan. doi:10.1109/ICWCSC.2010.5415879

SYMBIAN. Copyright © 2011 Nokia. All rights reserved. Disponível em: http://symbian.nokia.com/. Acesso em: 01 out. 2010.

SZYPERSKI, C. A. (1998). **Component software: beyond object-oriented programming**. ACM Press/Addison-Wesley Publishing Co., New York, NY.

TADDEO, A. V., MARCON, P., FERRANTE, A. (2009). **Negotiation of security services: A multi-criteria decision approach**. Embedded Systems Week 2009, ESWEEK 2009 - 4th Workshop on Embedded Systems Security, WESS 2009 , art. no. 1631720. Outubro.

TADDEO, A. V., MICCONI, L., FERRANTE, A. (2010). **Gradual Adaptation of Security for Sensor Networks**. Em: IEEE International Symposium on a World of Wireless Mobile and Multimedia Networks (WoWMoM 2010), Montreal, Canadá, Junho. doi: 10.1109/WOWMOM.2010.5534903.

TARR, P., OSSHER, H., (2001). **Workshop on Advanced Separation of Concerns in Software Engineering**. Software Engineering ICSE 2001. Proceedings of the 23rd International Conference, vol., no., pp.778-779. doi: 10.1109/ICSE.2001.919175

TEKINERDOGAN, B. (1997) **Adaptability in Object-Oriented Software Development**. In: Special Issues in Object-Oriented Programming, Heidelberg, Germany, pp. 7-12.

THE LEGION OF THE BOUNCY CASTLE. Copyright © 2000 - 2011 The Legion Of The Bouncy Castle. Bounce Castle Crypto API. Disponível em: http://www.bouncycastle.org/. Acessado em: 01 mar. 2010.

VIANA, W., Cavalcante, P., Andrade, R. M. C. (2005). **Mobile Adapter: Uma abordagem para a construção de Mobile Application Servers adaptativos utilizando as especificações CC/PP e UAProf**. In: XXV Congresso da Sociedade Brasileira de Computação, 2005, São Leopoldo, RS-Brasil. Anais do XXXII SEMISH.

WEISER, M. (1999). **Some Computer Science Issues in Ubiquitous Computing**. Communications of the ACM, v. 36, n.7, p. 75-84, 1993.

WILBANKS, L. (2008). **Green, My Favorite Color**. IT Pro, vol. 10, no. 6. pp. 63-64. IEEE Computer Society.

WINDOWS MOBILE. Windows Mobile ®. Disponível em: http://www.microsoft.com/. Acesso em: 01 out. 2010.

WULLEMS, C., THAM, K., SMITH, J., LOOI, M. (2004). **A trivial denial of service attack on IEEE 802.11 direct sequence spread spectrum wireless LANs**. Wireless Telecommunications Symposium - WTS 2004, pp. 129-136. Maio.