FUNDAÇÃO DE ENSINO "EURÍPIDES SOARES DA ROCHA"
CENTRO UNIVERSITÁRIO EURÍPIDES DE MARÍLIA - UNIVEM
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

PAULO CEZAR NERI DE OLIVEIRA

ASSINADOR DIGITAL ICP-BRASIL: ESTUDO E IMPLEMENTAÇÃO DO VERIFICADOR DE CERTIFICADOS DIGITAIS PARA PLATAFORMA ANDROID

MARÍLIA
2015

PAULO CEZAR NERI DE OLIVEIRA

ASSINADOR DIGITAL ICP-BRASIL: ESTUDO E IMPLEMENTAÇÃO DO VERIFICADOR DE CERTIFICADOS DIGITAIS PARA PLATAFORMA ANDROID

Trabalho de Conclusão do Curso de Bacharelado em Ciência da Computação da Fundação de Ensino "Eurípides Soares da Rocha", mantenedora do Centro Universitário Eurípedes de Marília - UNIVEM, como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação.

MARÍLIA

2015

CENTRO UNIVERSITÁRIO EURÍPIDES DE MARÍLIA - UNIVEM
MANTIDO PELA FUNDAÇÃO DE ENSINO "EURÍPIDES SOARES DA ROCHA"

BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

Paulo Cezar Neri de Oliveira

TÍTULO: Assinador Digital ICP-Brasil: Estudo e Implementação do Verificador de Certificados Digitais para Plataforma Android.

Banca examinadora da monografia apresentada ao Curso de Bacharelado em Ciência da Computação do UNIVEM/F.E.E.S.R., para obtenção do Título de Bacharel em Ciência da Computação.

Nota: 8,0 (oito)

Orientador:Fábio Dacêncio Pereira

1º.Examinador:Rodolfo Barros Chiaramonte

2º.Examinador:Fabio Piola Navarro

Marília, 01 de dezembro de 2015.

DEDICATÓRIA

Um trabalho de conclusão de curso é sempre um grande desafio em qualquer graduação, para qualquer graduando, desafio este que se acentua quando o graduando em questão necessita acumular outras atividades além do estudo. No entanto, com força, garra e fé, não há desafios que não possam ser ultrapassados, assim dedicamos este trabalho a todas as pessoas que junto comigo o tornaram possível, não serão citados nomes, para evitar de algum ser esquecido, porém dedicamos este trabalho a todos os professores, amigos, todos que de alguma forma ajudaram na realização do mesmo. Além destes, também a família pela força e fé depositadas.

AGRADECIMENTOS

Agradeço inicialmente a Deus por ter nos permitido e lutado junto para chegarmos até aqui. Em seguida a família, por todo apoio força e fé depositados. E por último a todos os profissionais que junto comigo tornaram este trabalho possível.

Sumário

Lista de Figuras

Lista de Tabelas

Resumo

Com o aumento do uso das plataformas móveis, e com a facilitação da obtenção da informação através de redes sem fio, os dispositivos móveis se tornaram capazes de permitir que praticamente qualquer informação seja acessada de qualquer lugar. Atualmente, um dispositivo móvel "inteligente" pode ser definido de forma simplista como um dispositivo com diversas funções que podem ser estendidas por meio de aplicações que são executadas em seu sistema operacional. Considerando também o perceptível aumento da geração de informação digital, sendo boa parte desta gerada por dispositivos móveis, a necessidade de validação da mesma é uma das consequências desse aumento, e o método de assinatura digital é uma das alternativas a ser considerada para esta garantia de validação e autenticidade, quando necessário. Neste contexto este trabalho propõe o estudo e a implementação de técnicas de assinatura digital existentes na linguagem de programação Java em conformidade com as regras de Assinatura Digital ICP-Brasil, para a plataforma Android, para possibilitar uma melhora na segurança de algumas atividades executadas nesta e para verificar a viabilidade deste tipo de transição de códigos entre plataformas.

Palavras-chave: Dispositivos Móveis, segurança da informação, assinatura digital, Plataforma Android.

Abstract

With the increased use of mobile platforms, and with the facilitation of obtaining the information via wireless networks, mobile devices have become capable of allowing almost any information to be accessed from anywhere. Currently, a "smart" mobile can be defined simplistically as a device with multiple functions that can be extended by means of applications that run on your operating system. Considering also the noticeable increase in the generation of digital information, with much of this generated by mobile devices, the validation of these is a consequence of that increase, and the digital signature method is an alternative to be considered for this validation warranty and Authenticity when necessary. Therefore, this work proposes the study and implementation of digital signature techniques exist in the Java programming language in accordance with the rules of Digital Signature ICP-Brasil, for the Android platform to enable an improvement in the safety of some activities performed in this and to verify the feasibility of this type of transition codes between platforms.

Keywords: mobile devices, information security, digital signature, Android platform.

Introdução

Desde os primórdios da humanidade, na época em que a escrita ainda era jovem, o homem busca maneiras de se comunicar, ou mais precisamente, transmitir mensagens de forma segura.

Silva; Martins (2011. P. 20,21, Apud KAHN, 1967), comenta que inicialmente a criptografia era bastante primitiva, e não possuía um propósito definido. Ele afirma que datam de 2.000 a.c, o uso de hieróglifos criptografados, que tinham a intenção de deixar a mensagem mais ostensiva e luxuosa, não demonstrando, então, interesse em esconder seu conteúdo. "Algum escriba anônimo, no século XX a.C., em uma cidade chamada Menet Khufu, às margens do rio Nilo, na incumbência de contar a história da vida de seu senhor, deu início também a história registrada da criptologia", afirma ele.

Umas das ideias que o principio da segurança da informação se baseia é que apenas as pessoas autorizadas possam ter acesso a algum tipo de informação tida como sensível. Para isso foram criadas técnicas para proteção destas informações. A partir deste desejo de proteção criou-se o termo Criptologia.

Para Silva; Martins (2011. P. 16, Apud Couto 2008, p. 18) a Criptologia é a ciência que se ocupa dos dois ramos, ou seja, tanto os conhecimentos e técnicas necessárias a Criptoanálise, que tenta solucionar mensagens criptografadas, quanto da criptografia que é a codificação da escrita e mensagens.

Apesar da criptologia ser o estudo da criptografia e criptoanálise, o termo criptografia é o mais conhecido. Este método consiste em cifrar uma informação de modo que apenas pessoas autorizadas possam ter acesso a ela.

De acordo com (Silva; Martins. P. 16, Apud SINGH, 2001), a origem da palavra criptografia vem do grego antigo, sendo *kriptó* traduzido como oculto e *graphein* escrita. Neste caso a criptografia trata da criação de métodos para transmitir as mensagens ou dados de forma secreta, confidencial e autêntica ao seu receptor, podendo se utilizar de códigos ou cifras para este fim.

A criptoanálise por sua vez são as técnicas que buscam quebrar as técnicas de criptografia, conseguindo decifrar as informações, mesmo não sendo o sujeito origem ou destino das mesmas. "Já a criptoanálise é responsável por analisar e quebrar os mais variados tipos de cifras e códigos criados sob a ótica da criptografia". (Silva; Martins. P. 16).

Em termos gerais, a criptografia busca garantir, entre outras coisas, que alguns serviços de segurança sejam garantidos, resumidamente, os principais serviços de segurança são:

Confidencialidade: este serviço visa proteger a informação contra divulgação não autorizada.

Autenticação: este serviço visa manter a comunicação e garantir a identidade do autor ao conteúdo da mensagem.

Serviço de Integridade: este serviço visa garantir que a mensagem está íntegra, ou seja, não foi alterada por nenhum sujeito não autorizado.

Serviço de irretratabilidade: este serviço visa garantir que nenhum autor factual de uma mensagem negue a sua autoria ou produção, este serviço também é conhecido como serviço de não repúdio.

Serviço de comprovação temporal: este serviço visa garantir que determinada mensagem existe depois de determinado espaço temporal.

Mesmo que inicialmente a criptografia não possuísse um objetivo claro, hoje esta é imprescindível para o funcionamento de diversos serviços que a sociedade utiliza e que muitas vezes nem sabe que a mesma está envolvida.

Uma simples troca de dados em numa rede sem fio, uma mensagem de texto ou e-e-mail enviada, compras pela internet, transferências bancárias, enfim, todos estes serviços utilizam e dependem da criptografia para que seja garantida a segurança dos mesmos e para que estas transações sejam asseguradas, sendo, então, estes serviços e estudos indispensáveis para o funcionamento da sociedade humana tanto no passado, quanto no presente.

Problemática

Com o crescente aumento da produção de informação digital, e com o grande aumento do uso de dispositivos móveis para a produção desta, considerando também a crescente quantidade de funções adicionadas aos mesmos, se faz necessário o desenvolvimento de técnicas para aumentar a segurança em algumas operações realizadas nestes dispositivos.

Objetivo Geral

Estudar uma API Java de geração de assinaturas digitais de acordo com as regulamentações da ICP-Brasil, e adaptar as técnicas utilizadas na mesma para verificação de certificados à plataforma Android. Entendendo conceitos básicos de certificado digital, assinatura digital, criptografia e implementando parte destes conceitos, verificando se esta adaptação é viável, considerando as diferenças que existem relacionadas ao sistema operacional e plataformas como um todo.

Objetivos Específicos

Para se atingir os objetivos desse projeto, deverão ser atingidos os seguintes objetivos específicos:

- Estudo e aprofundamento de conhecimentos e técnicas de programação da linguagem de programação Java.
- Estudo e aprofundamento das técnicas de programação da linguagem de programação Java para plataforma Android.
- Estudo e compreensão de parte do algoritmo da API Java de geração de assinaturas digitais de acordo com as regulamentações da ICP-brasil, criada na linguagem de programação Java.
- Adaptação das técnicas de verificação de certificado digital existentes nesta API para a plataforma Android.

Trabalhos Correlatos

Atualmente existem diversos trabalhos que tratam sobre assinatura digital. De acordo com Tognoli (2012. p. 14) e Teotônio (2012. P.16) , a maioria destes trabalhos estão mais intimamente ligados às questões de direito, sendo que estes buscam validar estas aplicações e as questões que tangem sua validação dentro dos processos judiciais, ou seja, validade jurídica.

De fato, não foram encontrados trabalhos acadêmicos sobre assinadores digitais, assim como, verificadores de certificados digitais para a plataforma Android. No entanto, é possível se encontrar artigos tratando das validações dos certificados digitais utilizados para assinar as aplicações dentro da plataforma Android.

Por exemplo, Frohlich e Ignaczak, discorreram que boa parte dos certificados digitais utilizados na assinatura dos aplicativos da plataforma Android, possuem algum tipo de falha de segurança, como por exemplo, nome do requerente não relacionado ao nome da empresa divulgada na loja virtual, entre outros.

Foram encontrados, no entanto, produtos relacionados à certificação digital, e assinatura digital, mas nenhum destes estão adequados às regulamentações da ICP-Brasil, pelo menos não para a plataforma Android.

Foi encontrado um assinador digital oferecido em forma de produto por uma empresa que, de acordo com a mesma, respeita as regulamentações da ICP-Brasil. No entanto, este produto se destina aos dispositivos da empresa Apple.

Metodologia

A metodologia para a realização deste projeto foi dividida em cinco partes principais:

1. Estudo e revisão bibliográfica sistemática de técnicas de assinatura digital. Nessa etapa foram pesquisados e lidos trabalhos anteriores relacionados a este tema.

2. Estudo e aprofundamento de conhecimentos e técnicas de programação da linguagem de programação Java. Nesta etapa, foram revisadas técnicas de programação da linguagem Java.

3. Estudo e aprofundamento de conhecimentos e técnicas de programação da linguagem de programação Java utilizada para programação de aplicações Android. Nesta etapa, foram revisadas técnicas de programação da linguagem Java para desenvolvimento na plataforma Android.

4. Estudo e compreensão de parte do algoritmo da API Java de geração de assinaturas digitais de acordo com as regulamentações da ICP-brasil, criada na linguagem de programação Java. Nesta etapa, foi estudado parte do algoritmo implementado dentro da Univem, em parceria com a LSI-TEC.

5. Adaptação da verificação dos certificados digitais existentes nesta API para a plataforma Android. Nesta etapa, foram analisadas técnicas implementação da verificação de

certificados digitais, sendo analisado, inclusive, a viabilidade desta metodologia de migração de plataformas.

Materiais

- Linguagem de programação JAVA e diversas APIs relacionadas à segurança da informação e criptografia;
- Linguagem de programação JAVA para desenvolvimento Android, com SDK para desenvolvimento de aplicações móveis para esta plataforma;
- Ferramentas Eclipse e NetBeans, neste caso o NetBeans para verificação do fonte da API de Assinatura Digital já desenvolvida e o Eclipse para o desenvolvimento da aplicação, se utilizando do SDK de desenvolvimento Android;
- Dispositivo móvel com Sistema Operacional Android, SmartPhone LG Optimus G E977, com versão do Android 4.4.2;
- API Java de geração de assinaturas digitais de acordo com as regulamentações da ICP-Brasil (código fonte), criada na linguagem de programação Java;
- Certificados digitais fake utilizados para testes.

Capítulo 01 – Conceitos básicos de Criptografia

Neste capítulo serão apresentados alguns conceitos importantes sobre segurança da informação, mais especificamente nas áreas de criptografia, certificação digital e assinatura digital. Além disso, serão introduzidas certas informações sobre alguns dos materiais utilizados para estudo neste trabalho, como por exemplo, a linguagem de programação Java e a plataforma Android.

1.1 - Criptografia simétrica e assimétrica

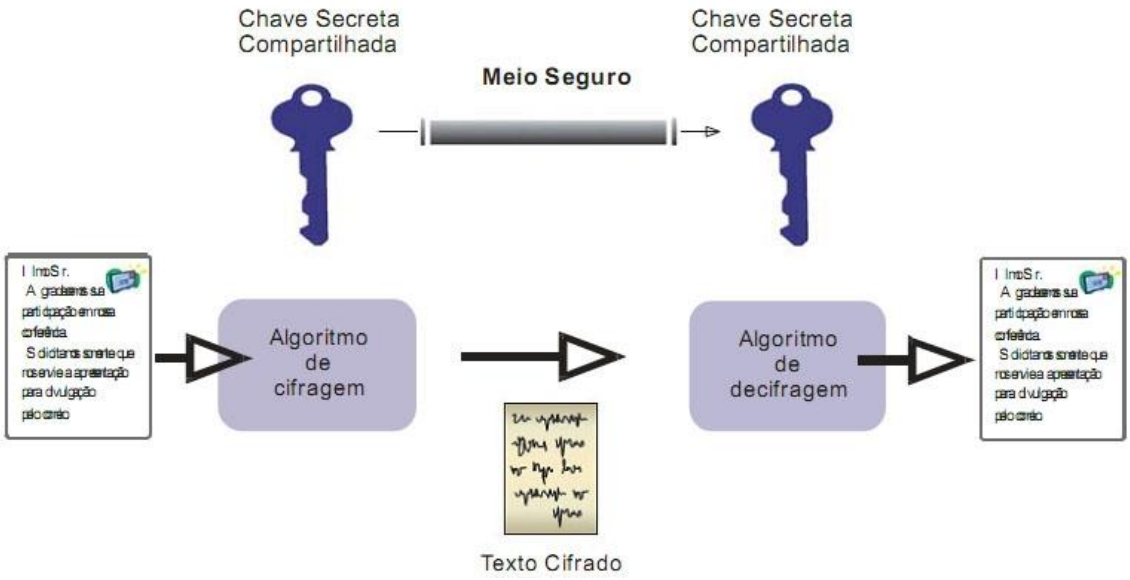
Existem basicamente dois tipos de algoritmos utilizados no âmbito criptográfico. São elas: a chave simétrica e a chave assimétrica.

1.1.1 - Criptografia de chave simétrica

Tognoli (2012. p. 16), afirma que a criptografia de chave simétrica, também conhecida como criptografia de chave única, se utiliza de apenas uma chave ou algoritmo para cifrar e decifrar as informações transmitidas. Ou seja, a mensagem é cifrada com uma chave secreta, transmitida e o receptor deve possuir a mesma chave para decifrá-la, caso ele não a possua, existe a necessidade do compartilhamento desta chave secreta.

A Figura 01, mostra de forma simplificada o processo de cifração/decifração utilizando-se chave simétrica. Nesta, uma mensagem qualquer é criptografada e enviada, sendo que também existe a necessidade de transmitir a chave privada por um canal seguro. Após isso a mensagem é decifrada com a chave privada transmitida através do canal seguro e retorna o seu conteúdo original.

Figura 01- Processo de cifração/decifração com chave simétrica



Fonte: Dias, 2012.

Este algoritmo de chave simétrica é utilizado ainda hoje, por ser mais simples e leve, todavia esta necessidade de compartilhamento da chave secreta pode ser encarada como um dos problemas correntes do uso deste método, pois qualquer pessoa que conseguir acesso à chave privada, conseguiria acesso ao conteúdo da mensagem cifrada.

Northcutt (2002, p. 618) discorre sobre isso e afirma que o algoritmo de chave simétrica é aquele método de criptografia que utiliza a mesma chave para codificação e decodificação. Ele explana que este uso considera a hipótese que já houve a troca desta chave secreta por um canal seguro que ninguém conhece, ou seja, que a troca desta chave não foi comprometida. Northcutt ainda afirma que uma das maiores dificuldades do algoritmo simétrico e aquela de executar trocas remotamente e autenticar o receptor como sendo a pessoa para quem deveria ser endereçada a mensagem. Por último ele comenta da dificuldade da transmissão desta chave caso o parceiro não a tenha, e da dificuldade de canais seguros para esta troca, pois caso os mesmos existissem, poder-se-ia inferir que não haveria necessidade de passar a chave e consequentemente de cifrar a mensagem.

> "Um exemplo simples de um algoritmo simétrico é a utilização do operador booleano XOR [...]; Por exemplo se utilizarmos um byte com o seguinte valor "00100110" como chave e aplicarmos ele no em byte com o valor "10101010" representando uma mensagem, obteremos o seguinte valor "10001100", isto é então a mensagem cifrada; Aplicando a mesma operação com a mesma chave teremos novamente o valor "10101010"." (Tognoli. 2012. p. 19).

Por exemplo, na Figura 02, um texto legível é submetido ao algoritmo de XOR com uma chave (byte) qualquer, o texto gerado, está criptografado e ao se aplicar neste texto gerado o mesmo algoritmo de XOR com a mesma chave utilizada para criptografá-lo, obtém-se o texto original.

Figura 02 - Processo de cifração/decifração com XOR

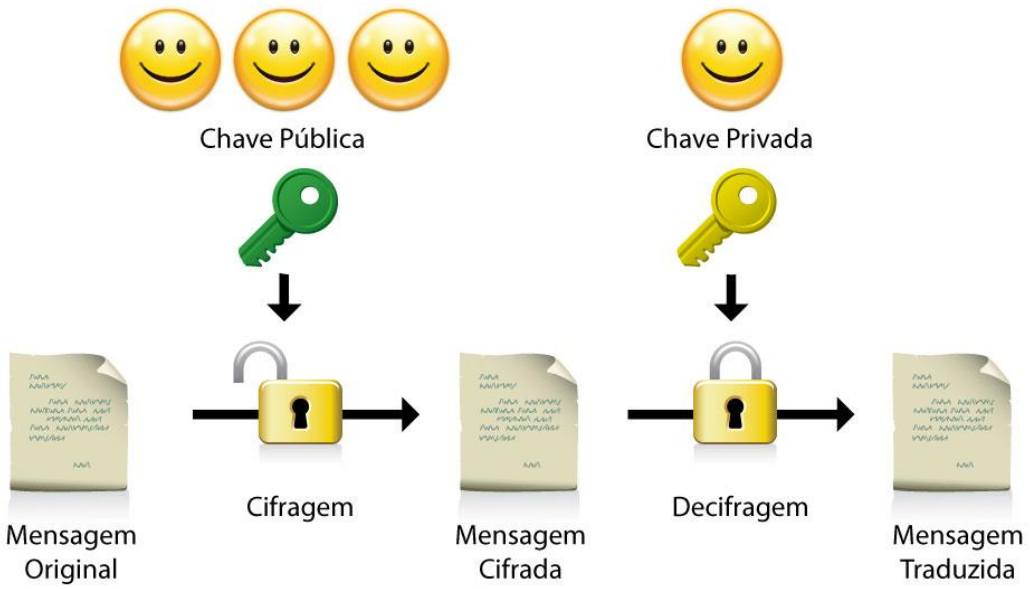
Fonte: Guelfi, 2012.

1.1.2 - Criptografia de chave assimétrica

Uma evolução do método de chave única foi quase inevitável. Desta evolução foi criado na década de 1970 o método de criptografia de chave assimétrica pelo matemático Clifford Cocks que trabalhava no serviço secreto inglês, o GCHQ. Este método faz uso de duas chaves, uma pública e uma privada. A chave pública é de conhecimento geral e pode ser compartilhada tranquilamente, a chave privada nunca deve ser compartilhada ficando apenas no poder de cada titular.

Na Figura 03 é demonstrado o processo de cifragem com a criptografia assimétrica, nesta uma mensagem é cifrada com uma chave pública e decifrada com a chave privada do destinatário.

Figura 03 - Processo de cifração/decifração com chave assimétrica



Fonte: Sobral, 2012

Oliveira, Ronielton (2012. P. 03,04), explana que no caso da criptografia de chave assimétrica, as chaves não são apenas senhas, mas sim arquivos digitais com uma maior complexidade técnica. Ele afirma que a chave pública pode ficar disponível para qualquer um que deseje se comunicar com o outro de forma segura, todavia, a chave privada deve estar sempre em poder do titular e em segredo, pois é com esta que o receptor da mensagem poderá decodificar alguma mensagem que foi criptografada para ele através de sua chave pública. Oliveira, ainda afirma que a grande serventia deste método é a possibilidade de qualquer pessoa poder enviar uma mensagem criptografada para outra, utilizando apenas sua respectiva chave pública. E como esta está disponível ao público, não existe a necessidade de enviar chaves privadas através de canais de comunicação, como é o caso do modelo simétrico, garantindo, desta forma, a confidencialidade.

Apesar do método de chave assimétrica ser mais seguro, ele utiliza algoritmos bastante complexos e que demandam uma maior carga computacional. Teotônio (2012. p.26), informa que a utilização deste par de chaves possibilita a realização de coisas, tidas anteriormente como impossíveis, no entanto, afirma ele, que os algoritmos de criptografia assimétrica possuem uma carga computacional muito maior se comparados aos algoritmos de chave única.

Mendes (2007, p. 23), afirma que estes algoritmos de chave assimétrica podem ser de 10 a 100 vezes mais lentos que os algoritmos de chave simétrica equivalentes.

Por fim Teotônio (2012. p.26), comenta que esta alta carga computacional e esta lentidão do algoritmo de chave assimétrica são os motivos do algoritmo do método de chave única ainda continuar em uso, normalmente em conjunto com a criptografia assimétrica, dependendo da situação.

Neste contexto pode-se concluir que dependendo da necessidade de segurança e da disponibilidade computacional, um ou outro algoritmo pode ser escolhido.

1.2 – Assinatura digital

O ato de assinar um documento é utilizado pela humanidade há séculos para garantir que aquele documento, de fato, foi criado pela pessoa/organização que o assinou. Com o advento dos dispositivos digitais, houve um aumento massivo da geração de informação digital. Assim, um método semelhante foi concebido no âmbito digital para tentar assegurar, por exemplo, que o serviço de autenticação seja garantido.

Para o entendimento do funcionamento de um algoritmo de assinatura digital, se faz necessário o entendimento do conceito de função hash.

A função hash, ou função resumo, pode ser definida de forma mais simples como uma função que cria uma espécie de resumo de um objeto maior. Ou seja, a partir de um objeto maior, o algoritmo cria um objeto menor, "resumido", do objeto original. Porém, se houver qualquer mudança no objeto original o objeto proveniente da função hash do mesmo será diferente.

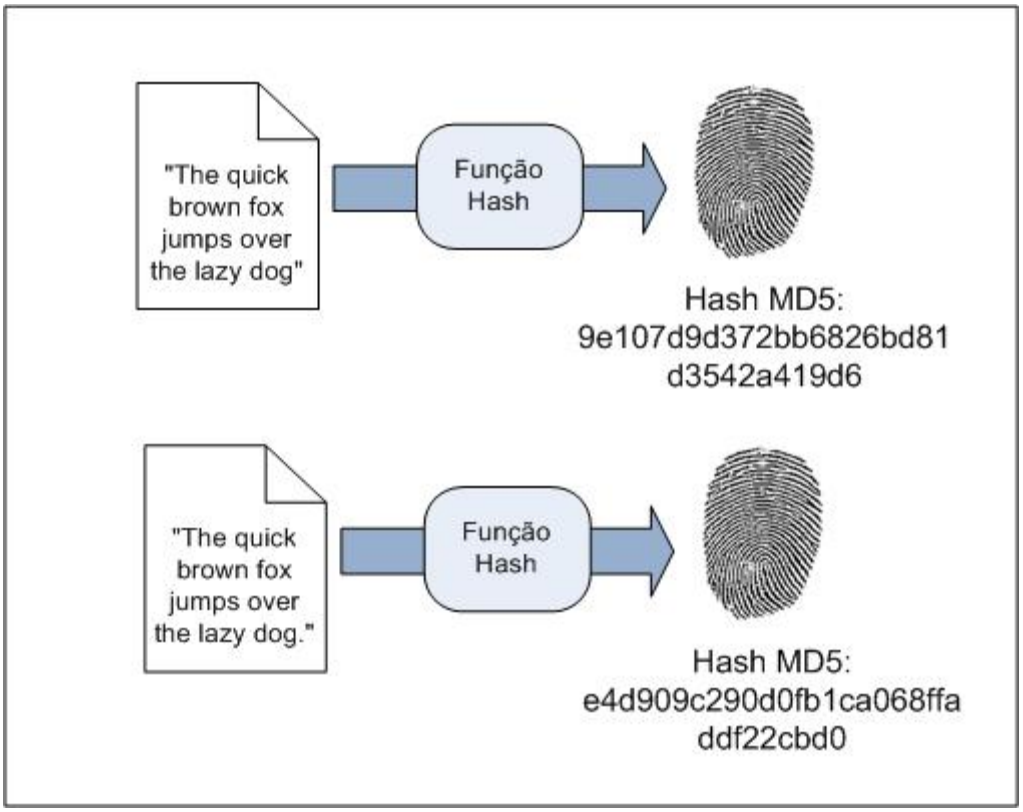
"A função hash é uma função unidirecional que utiliza todos os bits do valor de entrada oferecendo a capacidade de detecção de erros, pois qualquer mudança no valor de entrada altera o resultado da função hash". (Stallings,2008).

> "Essas funções são utilizadas para garantir a integridade da mensagem, já que o objetivo delas é gerar um valor y diferente para cada mensagem. Com isso, caso a mensagem x seja modificada para uma x', quando o destinatário receber a mensagem x', terá de recalcular y para a mensagem que recebeu (x'). Como cada mensagem gera um y diferente, ele irá detectar que y para a mensagem que ele recebeu é diferente daquilo que esperava e, com isso, detectar que a mensagem foi alterada." (Moreno;Chiaramonte;Pereira,2005).

A Figura 04 mostra um esquema simples da função *hash*, nesta, duas mensagens são submetidas a esta função e é criado um objeto de resumo das mesmas, ou seja, tem-se um valor de *hash*, nesta figura estes valores são representados inclusive com uma impressão digital, para mostrar seu valor único. Note que a diferença de um ponto final na primeira e

segunda mensagem gera uma função *hash* diferente, demonstrando a característica e funcionalidade desta função.

Figura 04 - Função de Hash

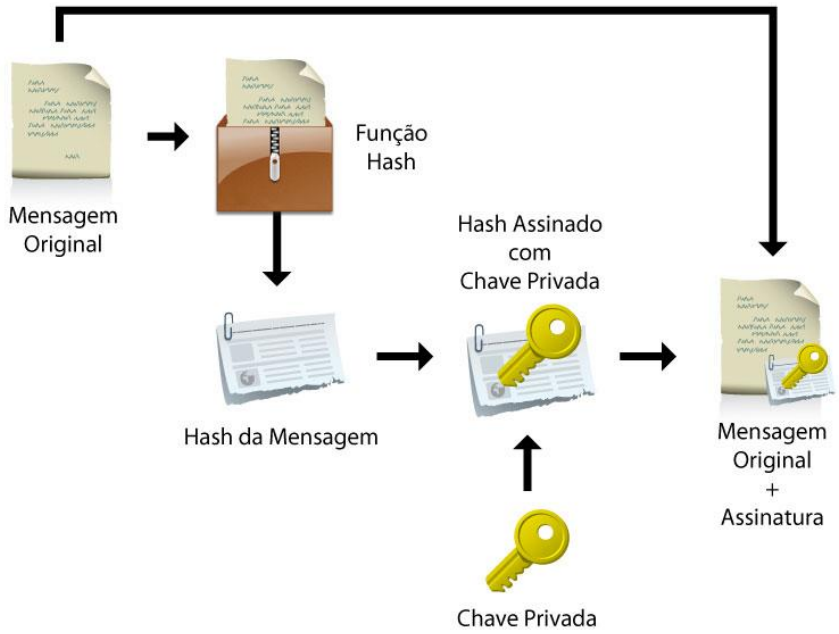


Fonte: Carvalho, 2008.

A assinatura digital é um método utilizado em âmbito computacional para tentar garantir a autenticidade de um objeto, ou seja, ter um valor semelhante ao que a assinatura em papel tem no mundo real. “Apesar da analogia com a assinatura manuscrita, a assinatura digital é elaborada e validada por sistemas computacionais, utilizando técnicas matemáticas e algoritmos criptográficos” (Teotônio, 2012 p. 14).

A Figura 05 representa uma comunicação hipotética entre duas entidades, com um arquivo assinado digitalmente. Nesta é criado um *hash* de uma mensagem, o mesmo é criptografado e enviado junto com a mensagem como um objeto de assinatura:

Figura 04 - Assinatura digital



Fonte: Sobral, 2012.

De acordo com Menezes et al (1996, apud Araújo, 2013, p. 05), as assinaturas digitais provem um meio de anexar a identidade de uma pessoa a algum tipo de informação criada e/ou enviada por ela. Assim, este processo de assinar consiste em transformar esta mensagem, e criar nela uma espécie de "marca" única, e esta marca é denominada assinatura.

Tognoli. (2012. p. 19), define que para que o processo de assinatura seja realizado, os seguintes passos devem ser seguidos: A geração de um hash do documento que se deseja assinar. Após isso este hash é cifrado com a chave privada do assinante. O resultado deste processo é o que se chama de assinatura. Esta assinatura deve ser vinculada ao documento original de alguma forma. No caso de validade temporal, a assinatura de um terceiro é utilizada para fornecer o tempo de forma segura. Que de acordo com ele são as chamadas autoridades de carimbo de tempo (ACT).

Em outras palavras, os algoritmos de assinatura digital criam uma espécie de objeto de assinatura, proveniente da função hash e este é criptografado e transmitido junto com o objeto que foi assinado, tentando garantir que pelo menos o serviço de autenticação seja respeitado.

Nas palavras de Moreno, Pereira e Chiaramonte (2005, p. 157) a assinatura digital é definida como:

> "[...] uma assinatura digital é o criptograma resultante da cifração de um determinado bloco de dados (documento) pela utilização da chave-privada de quem assina em um algoritmo assimétrico. A verificação da assinatura é feita decifrando-se o criptograma (assinatura) com a suposta chave-pública correspondente, Se o resultado for válido, a assinatura é considerada válida, ou seja, autêntica, uma vez que apenas o detentor da chave privada, par da chave pública utilizada, poderia ter gerado esse criptograma."

1.3 – Assinatura digital para dispositivos móveis

Com o massivo aumento percebido no uso das plataformas digitais para criação e obtenção de informações, além disso, com o também incremento do uso de dispositivos móveis para o mesmo fim, a informação passou a ser transmitida comumente tanto entre redes sem fio, quanto cabeadas. "[...] a maior parte desses equipamentos deverá ter capacidade de se comunicar com a parte fixa da rede [...] A esse ambiente de computação se dá o nome de computação móvel [...]" (Matheus; Ferreira, 1998, p. 01).

Tonin (2012, p.03), explica que a chamada "computação móvel" representa um novo padrão computacional, surgindo como a quarta revolução da computação, antecedida por toda a história conhecida da ciência da computação. Sendo que, de acordo com ele, a palavra que pode definir este novo modelo é mobilidade.

Com a crescente evolução e melhoramento destes dispositivos, tivemos a criação dos *smartphones, tablets, etc.*. "Os dispositivos móveis estão cada vez mais eficientes, especialmente com a chegada dos *Smartphones*, que possuem recursos inteligentes integrados." (Rovadosky et al, 2012, p.01).

Os dispositivos móveis atuais, chamados comumente de dispositivos "inteligentes", possuem diversas funções antes exclusivas dos computadores de mesa, no entanto estes possuem a vantagem da mobilidade completa. De acordo com Rodrigues (2009, p. 19) "Pelas funcionalidades disponíveis e diversas definições, podemos classificar os *smartphones* como dispositivos programáveis que convergem mobilidade e conectividade".

Atualmente existem diversos aplicativos que executam os serviços de assinatura digital para computadores de uso geral. Porém, para o âmbito dos dispositivos móveis, estes aplicativos ainda não são comuns. Sendo assim, é natural que se comece a pensar em soluções para sanar esta dificuldade.

1.4 – A linguagem de programação Java

Este trabalho buscará adequar técnicas de Assinatura Digital existentes em uma API desenvolvida na linguagem de programação Java. Faz-se necessário então uma pequena introdução sobre esta linguagem de programação e suas principais características e vantagens, para um melhor entendimento do objeto de estudo e implementação.

A linguagem de programação Java é uma linguagem de programação orientada a objetos, que tem como principal característica rodar suas aplicações sobre uma máquina virtual, como uma camada a mais na execução de um aplicativo, esta característica é útil quando se fala de aplicações multiplataforma. Esta linguagem foi concebida inicialmente para se tornar a linguagem base para o desenvolvimento dos softwares utilizados em produtos eletrônicos. A história da linguagem Java se inicia no ano de 1991, quando três engenheiros (Patrick Naugthon, Sun Fellow e James Gosling) se juntaram a fim de criar tecnologias de software para empresas eletrônicas.

A portabilidade é uma das principais características presentes nesta linguagem, como suas aplicações rodam sobre uma máquina virtual, as mesmas podem ser escritas apenas uma vez e rodarem em qualquer sistema operacional ou plataforma, contanto que o mesmo possua a máquina virtual instalada. A linguagem de programação Java é considerada multiplataforma, ou seja, ela pode funcionar em diversos sistemas operacionais, sem grandes alterações de código fonte. A compilação de um código fonte em Java, gera um código intermediário chamado bytecode. Este código intermediário é interpretado pelas máquina virtuais.

A característica multiplataforma da linguagem Java depende intrinsecamente da existência das máquinas virtuais da mesma, pois caso haja a um novo sistema operacional, uma máquina virtual para este deve ser criada e esta traduzirá os bytecodes para o código daquele sistema operacional, mantendo a característica multiplataforma da lingugaem Java.

De acordo com Oliveira, Igor (2011, p. 36) as aplicações em Java seguem o seguinte ciclo: “De um código Java, que está em um arquivo .Java, o compilador javac gera o bytecode: um arquivo .class. Após isso uma máquina virtual Java executa o bytecode e roda o programa.”. Demonstrando assim a característica multiplataforma desta linguagem de programação e o trabalho realizado pelas máquinas virtuais.

Dentro deste contexto a máquina virtual é a responsável pelo carregamento de todas as classes do programa, assim como a verificação dos *bytecodes,* no que se relaciona a aderência a JVM e a verificação da integridade do sistema, afirma Oliveira.

Esta independência evita o transtorno sofrido por outras linguagens quando o sistema tem que migrar para outra plataforma, não existindo então a necessidade de alteração do código fonte, sendo necessário apenas que a outra plataforma possua uma máquina virtual desenvolvida para ela, e a mesma esteja instalada. É importante lembrar que as máquinas virtuais são específicas para cada sistema operacional ou plataforma.

> "A independência de plataforma já fala por si, pois possibilita o programa ser executado em diferentes plataformas e sistemas operacionais, através de um emulador conhecido como a Máquina Virtual Java ou JVM (Java Virtual Machine) que ajuda rodar os sistemas baseados em Java. Pode-se também se denominar como uma máquina virtual baseada em software que é executada dentro dos aparelhos eletrônicos onde irá ler e executar os *bytecodes* do Java." (Palmeira).

Apesar de todas essas facilidades causadas pela utilização de uma máquina virtual, ao se comparar a linguagem Java com outras linguagens que geram o código de máquina diretamente para o sistema operacional, poder-se-ia inferir que existe uma perda de desempenho e velocidade de execução. No entanto com a constante evolução das máquinas virtuais Java estas diferenças tem se tornando cada vez menores.

Oliveira, Igor (2011, p. 36) explana sobre este assunto, ele explica que apesar de existir um programa traduzindo um código em todas as execuções do sistema, esta situação hoje em dia já não é tão crítica, pois houve no ano de 1996 uma melhoria nesta rotina, sendo que a Sun criou um compilador conhecido como Just-in-time (JIT), que tem por finalidade analisar e retirar códigos desnecessários do sistema, aumentando, desta forma, a velocidade da execução do código.

Além de todas estas características expostas, a linguagem Java possui diversas bibliotecas específicas para a área de segurança da informação e assinatura digital, por exemplo na API de assinatura digital, objeto de estudo deste trabalho, foi utilizada a coleção de APIs Bouncy Castle, que possui diversas bibliotecas, métodos e soluções específicas na parte de criptografia e segurança da informação.

Do site "*The Legion of the Bouncy Castle*", este pacote é definido como "Uma implementação Java de algoritmos criptográficos, que foi desenvolvida por esta legião e com ajuda de terceiros". Tradução nossa.

Por todas estas características e pela existência destas bibliotecas e API´s nesta linguagem de programação, esta foi a linguagem escolhida para ser utilizada na API de assinatura digital original que é o objeto de estudo deste trabalho e terá parte do seu código migrado para a plataforma Android.

1.5 – Escolha da plataforma Android

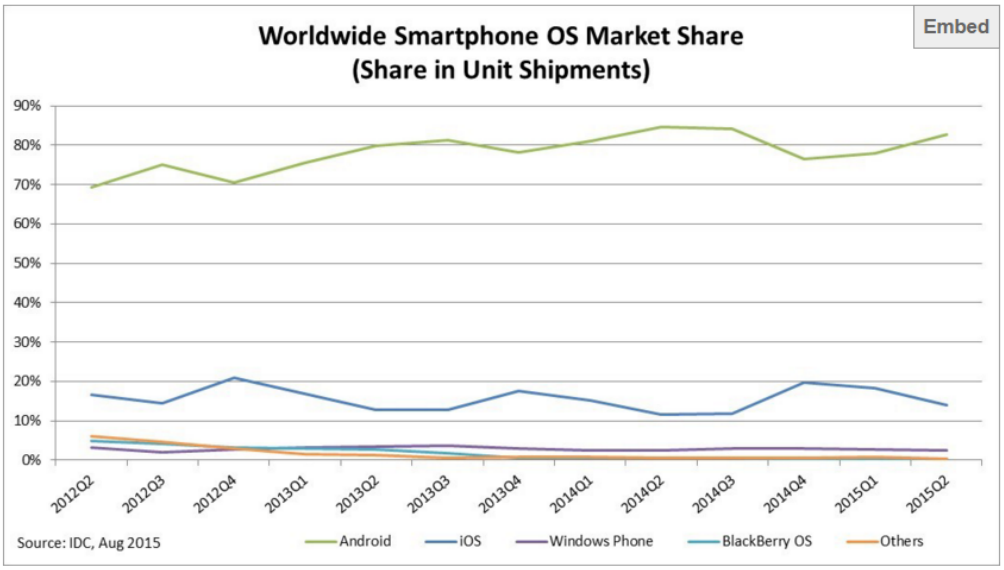
O Android é um sistema operacional desenvolvido pela Open Handset Alliance, que foi adquirido pela empresa de tecnologia Google no ano de 2005. “A Open Handset Alliance consiste em todas as estruturas envolvidas no processo de telefonia móvel.” (Pereira; Silva, 2009, p.2).

Atualmente é um sistema de código aberto baseado em Linux e se utiliza principalmente da linguagem de programação Java para desenvolvimento de suas aplicações.

Pereira e Silva (2009, p.3), define o Android como uma plataforma móvel completa. Que foi desenvolvido com o intuito de dar aos desenvolvedores a oportunidade de criar aplicações com tenham a capacidade de obter o maior proveito possível dos aparelhos portáteis. E como esta plataforma é Opensource, o mesmo pode ser adaptado para incorporar novas tecnologias quando as mesmas surgirem.

No âmbito do uso de dispositivos móveis, atualmente o sistema operacional Android predomina no mercado, de acordo com pesquisa realizada pelo IDC, este sistema é o mais utilizado em todo o mundo, estando presente em 82,8% dos dispositivos móveis no planeta. Na Figura 06 é mostrado um gráfico com os sistemas operacionais para dispositivos móveis e suas porcentagens de utilização no mundo.

Figura 06 – Gráfico do uso de smartphones no mundo.



Period	Android	iOS	Windows Phone	BlackBerry OS	Others
2015Q2	82.8%	13.9%	2.6%	0.3%	0.4%
2014Q2	84.8%	11.6%	2.5%	0.5%	0.7%
2013Q2	79.8%	12.9%	3.4%	2.8%	1.2%
2012Q2	69.3%	16.6%	3.1%	4.9%	6.1%

Source: IDC, Aug 2015

Fonte: IDC 2015

Sendo este o sistema operacional para dispositivos móveis mais utilizado no mundo, e como as aplicações desenvolvidas para ele tem como base a linguagem de programação Java. Considerando que existe uma aplicação desenvolvida nessa linguagem de acordo com as regras da ICP-Brasil (Infraestrutura de Chaves Públicas Brasileira) e ainda o aumento do uso de dispositivos móveis de forma mais intensa e abrangente, tem-se, então, a oportunidade e a necessidade da migração das técnicas desenvolvidas na aplicação Java, para plataforma Android, seguindo as regras da ICP-Brasil.

1.6 - Considerações finais do capítulo

Após este estudo inicial e entendimento dos principais conceitos relacionados à segurança da informação e criptografia, percebe-se a necessidade da implementação e/ou migração de técnicas relacionadas à segurança da informação para estas novas plataformas digitais.

Como muitas destas técnicas já estão implementadas para os computadores de uso geral, nada mais natural que a migração das mesmas para as plataformas móveis. Pois dessa

forma tem-se a oportunidade de melhorar a segurança em algumas das atividades executadas por estes dispositivos.

Este trabalho visa adequar técnicas de verificação de certificados digitais já existentes para computadores comuns, de âmbito geral, para os dispositivos *mobile* atuais, implementando esta verificação de certificados digitais para esta plataforma.

Além desta adequação e implementação, este trabalho também visa verificar se esta "conversão" das técnicas para o âmbito móvel, é viável. Pois sabe-se que apesar da evolução, tanto de *software* quanto de *hardware* destes dispositivos móveis, a maioria deles, pelo menos os mais populares, ainda não supera, ou se iguala, a um computador de uso geral.

Capítulo 02 – Assinatura digital e certificado digital (ICP-Brasil).

Neste capítulo será feito um breve estudo sobre as definições dos principais conceitos relacionados à ICP-Brasil, seus certificados digitais, assinaturas digitais e hierarquia.

2.1 – ICP-Brasil – Definições e principais métodos

Dentro do âmbito de regulação, fiscalização e controle da geração de certificados existem no mundo diversas organizações, no Brasil tem-se a ICP-Brasil fazendo parte do ITI.

De acordo com Tognoli (2012 p. 21), o objetivo de uma Infraestrutura de Chaves Públicas é a definição de uma estrutura para a emissão de chaves públicas e o estabelecimento de normas e técnicas para a utilização destas chaves. Estas normas devem garantir, de acordo com ele, uma relação entre a entidade e seu par de chaves criptográficas inclusive a validade das mesmas.

Já no site da ICP-Brasil temos a seguinte definição dela mesma:

> "A Infraestrutura de Chaves Públicas Brasileira (ICP-Brasil) é uma cadeia hierárquica e de confiança que viabiliza a emissão de certificados digitais para identificação virtual do cidadão. Observa-se que o modelo adotado pelo Brasil foi o de certificação com raiz única, sendo que o ITI, além de desempenhar o papel de Autoridade Certificadora Raiz (AC-Raiz), também tem o papel de credenciar e descredenciar os demais participantes da cadeia, supervisionar e fazer auditoria dos processos." (Site ICP-Brasil).

Dentro do site do ITI (Instituto Nacional de Tecnologia da Informação) Brasil (Site ITI - Quem Somos), o mesmo se define como uma autarquia federal que está vinculada à casa civil da Presidência da República. Tendo o objetivo de manter a ICP-Brasil, sendo a primeira autoridade da cadeia de certificação, conhecida como AC Raiz. Neste site também são listadas as principais competências do ITI, estre elas está a estimulação e articulação de projetos de pesquisa científica e de desenvolvimento tecnológico voltados a ampliação da cidadania digital. Para isto o ITI utiliza como linha de ação a popularização da certificação digital ICP-Brasil e a inclusão digital, atuando sobre as questões relacionadas com sistemas criptográficos, entre outros.

A Figura 07 representa os principais componentes e siglas utilizadas para definir os processos dentro da ICP-Brasil:

Figura 07- Identificação dos principais componentes

Fonte : Guelfi, 2012.

A Tabela 01 mostra separadamente qual o papel de cada um destes componentes dentro do âmbito da ICP-Brasil, com suas siglas e definições:

Tabela 01 - Componentes do ciclo de vida da ICP-Brasil
Fonte: Teotônio, 2012. p.44,45

Entidades	Entidade Final (EF)		Uma entidade final pode ser: um usuário humano, um servidor, uma aplicação ou uma organização. Ela é quem utiliza os recursos oferecidos pela ICP-Brasil.
	Autoridade Certificadora (AC)		Uma autoridade certificadora é uma entidade confiável para gestão do ciclo de vida dos certificados digitais, isso envolve: emitir certificados, renovar certificados, revogar certificados. Uma AC pode emitir dois tipos principais de certificados: certificado para EF ou certificado para uma AC subordinada.
	Autoridade Registradora (AR)		Uma autoridade registradora é uma entidade, opcional, subordinada à AC. Sua função é apenas manter repositórios de certificados emitidos e de LCRs. Sua justificativa é separar os processos mais críticos dos mais simples em uma AC.
Certificados Digitais			Seu conceito foi explicado na seção 1.9 **Obs.: Referência ao trabalho original, neste trabalho definição encontra-se no ítem 2.2 – Certificado digital ICP-Brasil.**
Infraestrutura	Repositório de Certificados (RC)		Sua função é armazenar e tornar disponível os certificados digitais emitidos por uma AC.
	Lista de Certificados Revogados (LCRs)		Sua função é possibilitar a uma entidade qualquer verificar se um dado certificado digital foi revogado ou não, elas devem ser mantidas disponíveis pela AC e devem ser periodicamente atualizadas e publicadas pela mesma.

2.2 – Certificado digital ICP-Brasil.

Dentro do ITI e da ICP-Brasil, um certificado digital é definido como uma "Identidade Virtual" que tem por objetivo a segura e inequívoca identificação do autor de uma mensagem ou transação. Sendo que este documento eletrônico também é assinado por uma terceira parte confiável, conhecida como Autoridade Certificadora (AC), que seguindo as regras da ICP-Brasil, associa uma entidade a um par de chaves criptográficas.

Assim sendo, um certificado digital pode ser definido como uma assinatura, inclusive com validade jurídica, que tem por objetivo garantir proteção a algumas transações efetuadas de forma eletrônica.

Ainda no site da ICP-Brasil temos uma introdução sobre as possibilidades do certificado digital ICP-Brasil, de acordo com este, o certificado além de garantir a identidade de um indivíduo, garante validade jurídica aos atos praticados com o seu uso.

Neste site é explicado que a certificação digital tem a finalidade de possibilitar a aplicações eletrônicas de diversas naturezas sejam realizadas e assinadas virtualmente, mas que demanda clara identificação da pessoa que a está realizando pela internet.

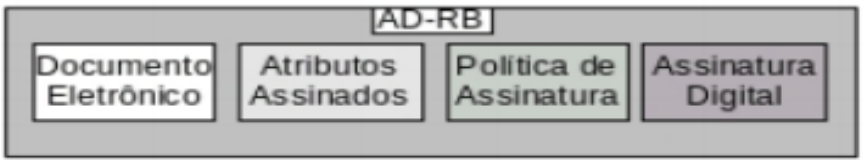
2.3 – Formatos de assinatura digital admitidos na ICP-Brasil

De acordo com o DOC ICP-15.01Versão 2.1, No âmbito da ICP-Brasil, a utilização de formatos padronizados é imprescindível para a confiabilidade e credibilidade do processo de criação e validação de assinatura.

Ainda de acordo com este mesmo DOC ICP-15.01Versão 2.1, uma assinatura digital ICP-Brasil deve ter um dos seguintes formatos:

a) Assinatura digital com Referência Básica (AD-RB): Formato mais básico aceito na ICP-Brasil, ou seja, no Brasil uma assinatura digital deve ter no mínimo os seguintes atributos demonstrados na Figura 08:

Figura 08 - Assinatura digital com Referência Básica

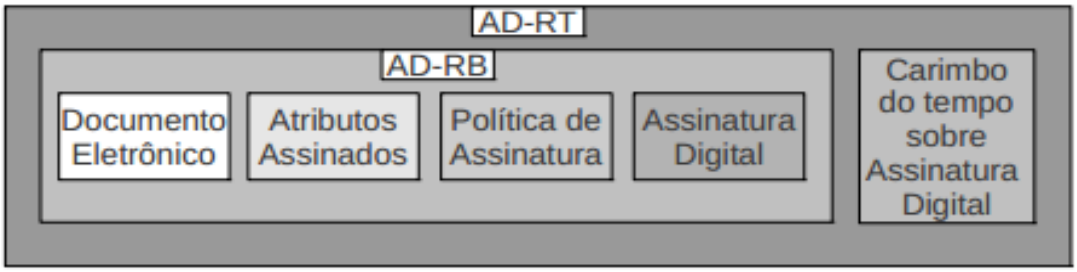


Fonte: DOC ICP-15.01 Versão 2.1 p. 04

b) Assinatura digital com Referência de Tempo (AD-RT): Contempla todos os atributos do tipo básico e adiciona um atributo chamado de carimbo de tempo, que garante

que assinatura foi gerada numa determinada data, a representação da mesma está representada na figura 09.

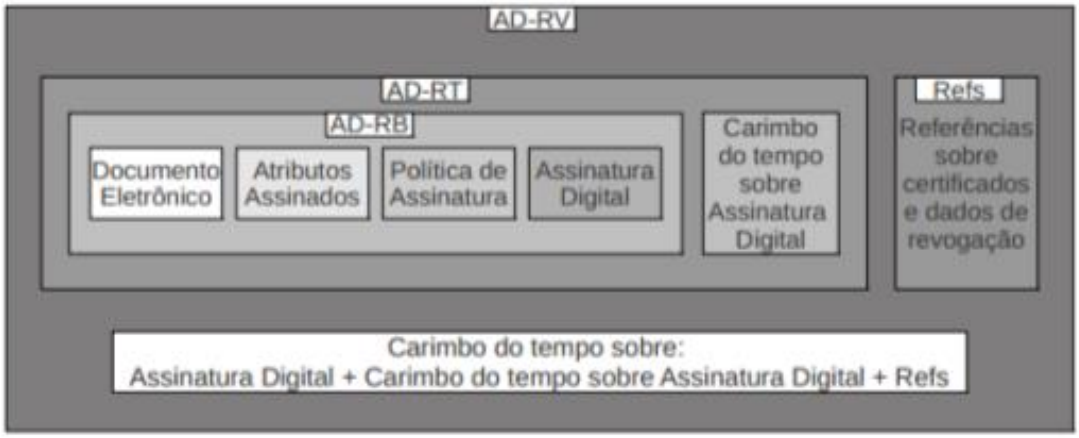
Figura 09 - Assinatura digital com Referência de Tempo (AD-RT)



Fonte: DOC ICP-15.01 Versão 2.1 p. 04

c) Assinatura digital com Referências para Validação (AD-RV). Além de todos os atributos anteriores, este tipo de assinatura digital contém os resumos criptográficos dos certificados e das CLR's, esta está representada na figura 10.

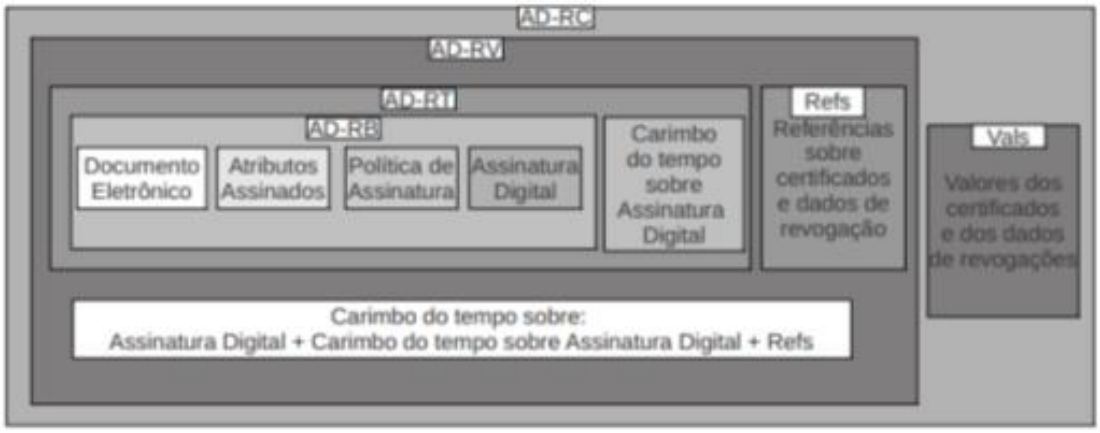
Figura 10 - Assinatura digital com Referências para Validação



Fonte: DOC ICP-15.01 Versão 2.1 p. 04

d) Assinatura digital com Referências Completas (AD-RC). Contém todos os atributos dos certificados anteriores e adiciona os valores dos certificados e as CLRs em sua lista de atributos não assinados, pode-se verificar uma representação da mesma na figura 11.

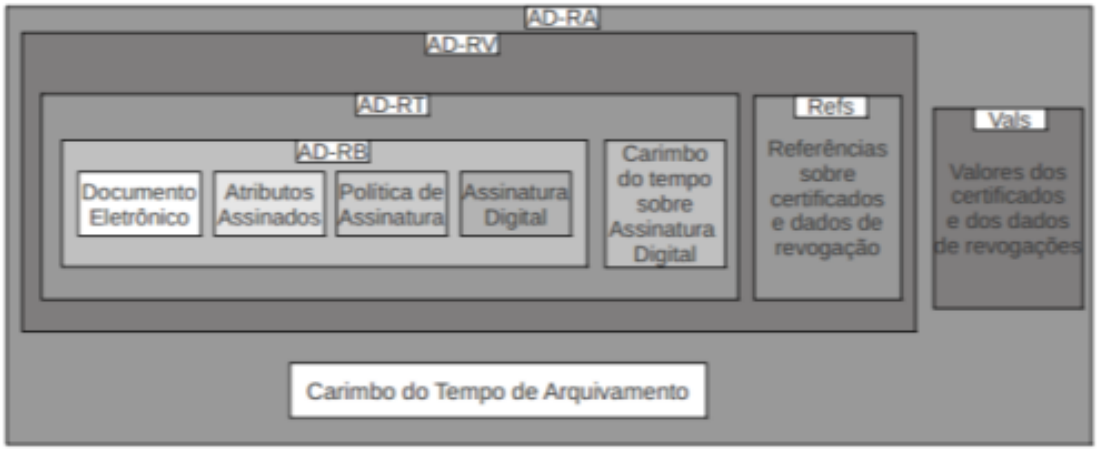
Figura 11 - Assinatura digital com Referências Completas



Fonte: DOC ICP-15.01 Versão 2.1 p. 05

e) Assinatura digital com Referências para Arquivamento (AD-RA). Utilizadas como meio de permitir que assinaturas digitais criadas com algoritmos ultrapassados e considerados inseguros possam ser arquivados de forma segura, a mesma está representada na figura 12.

Figura 12 - Assinatura digital com Referências para Arquivamento



Fonte: DOC ICP-15.01 Versão 2.1 p. 05

Capítulo 03 – Estudo sobre a API de assinatura digital desenvolvida de acordo com as normas da ICP-Brasil.

Este capitulo trará um breve estudo sobre a API de assinaturas digitais completa criada em conjunto com a Univem/Compsi/LSI-TEC e que é um dos materiais utilizados para a realização deste trabalho.

3.1 – Assinador Digital ICP-Brasil.

Este trabalho se baseia numa API de assinaturas Digitais existente, que foi desenvolvida de acordo com as normas da ICP-Brasil. A partir deste momento será feito um pequeno estudo sobre a mesma para possibilitar a compreensão de parte do funcionamento desta.

Para o desenvolvimento do Assinador foi utilizada como base as API's Bouncy Castle Cryptography, que possuem diversas bibliotecas com métodos e soluções relacionadas aos processos criptográficos. Mais precisamente a API BC, que é implementada na linguagem de programação Java, possuindo pacotes para desenvolvimento de assinaturas digitais e outros.

Teotônio (2012, p. 51), informa que a API Boucy Castle Criptografhy Library (BC) é uma coleção de bibliotecas que possuem métodos diversos para processos criptográficos, e uma de suas vantagens é que é periodicamente atualizada e revisada pelos seus idealizadores.

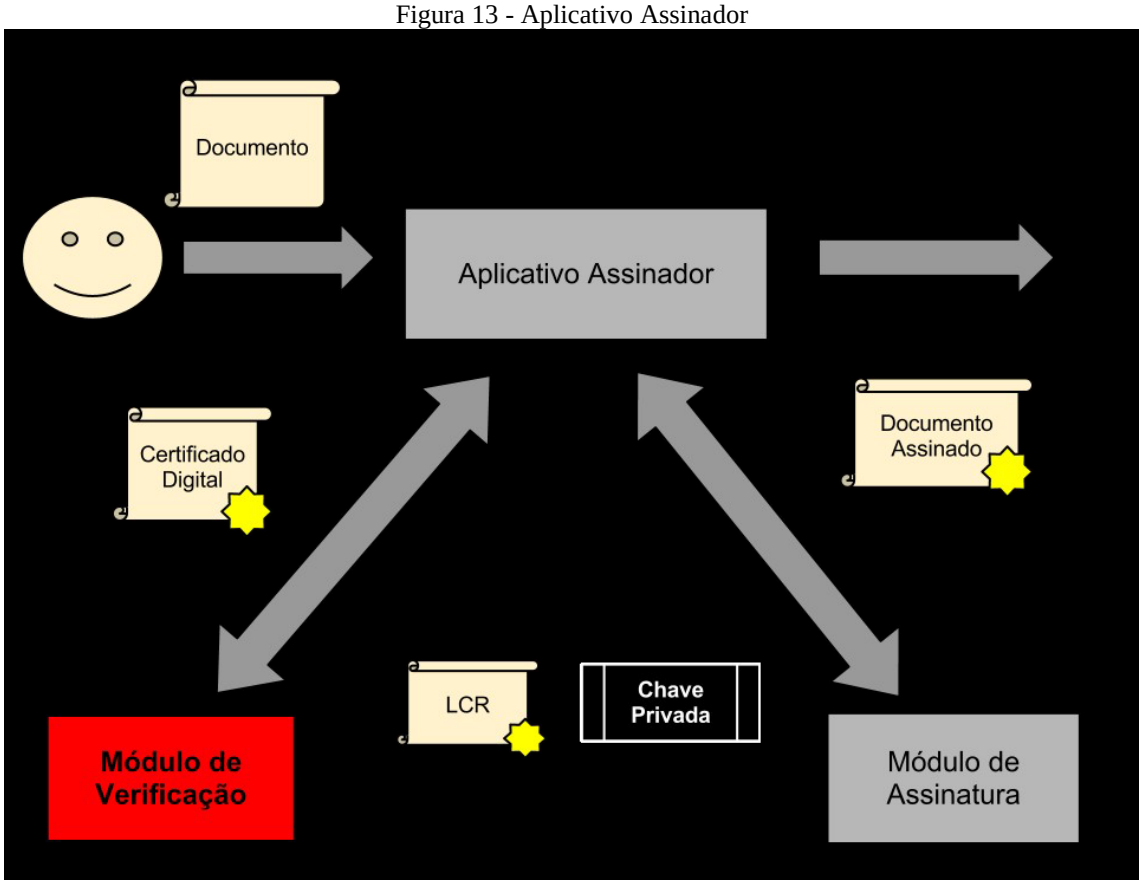
Já Tognoli (2012, p. 35), comenta que a API Bouncy Castle, tem a finalidade de complementar os recursos nativos da linguagem de programação Java. Pois esta, possui classes que facilitam o manuseio de certificados digitais, além de oferecer algoritmos criptográficos, assim como, suporte a documentos assinados. Esta API, foi utilizada com bastante frequência para extração dos dados dos certificados e assinaturas, possibilitando, então, as validações e testes necessárias.

Nas palavras de Tognoli (2012, p. 36), a estrutura do aplicativo assinador foi dividida em módulos, sendo principalmente o módulo de assinatura e o modo de verificação de certificados e assinaturas.

A figura 13 procura demonstrar de modo sucinto à forma como os módulos serão utilizados na aplicação completa. Tognoli (2012, p. 36) explicou que o módulo de assinatura faz uso do módulo de verificação para que possa obter a chave privada do assinante, a qual

deve ser instalada na keystore do sistema operacional. Sendo que o módulo de assinatura também se utiliza do módulo de verificação para obter documentos úteis para a geração da assinatura digital, tais como o certificado e a lista de certificados revogados.

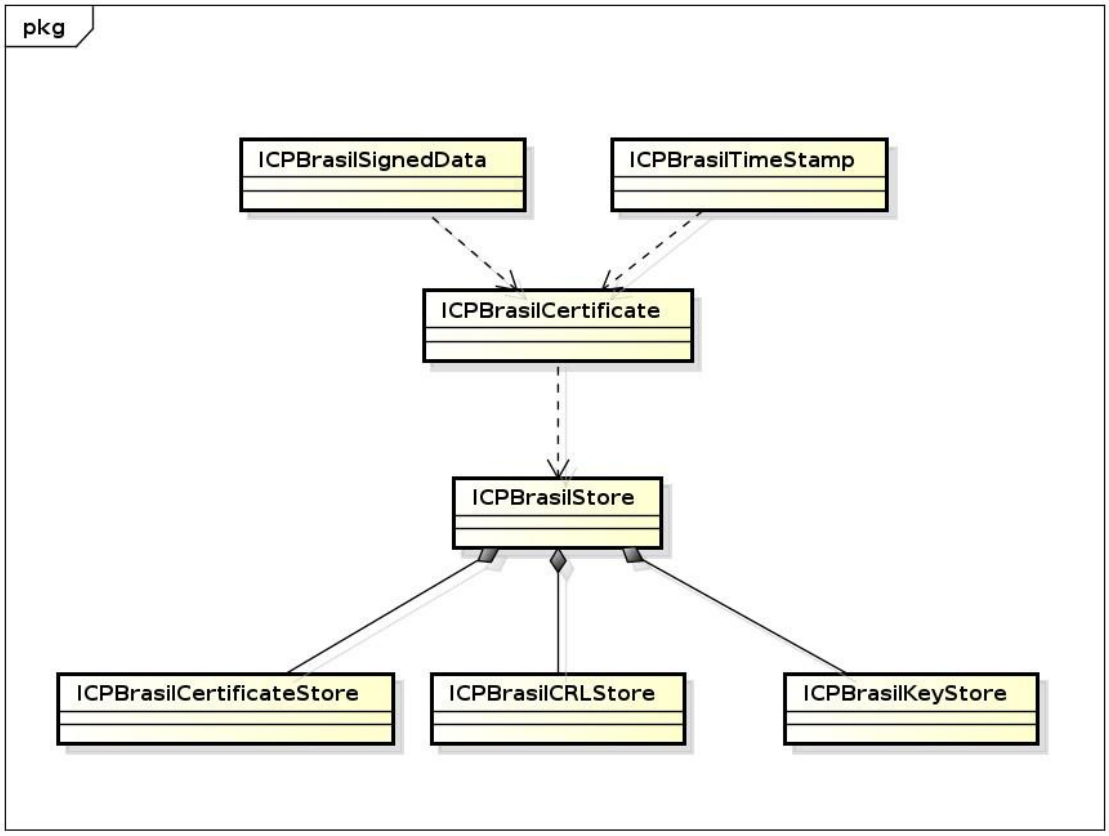
Figura 13 - Aplicativo Assinador



Fonte: Tognoli. 2012. p. 36

Tognoli (2012, p. 37) também construiu alguns diagramas de classes superficiais a fim de demonstrar o funcionamento do modo de verificação. No diagrama na figura 14 é mostrado a dependência que existe entre as classes de validação de assinatura (*ICPBrasilSignedData*; *ICPBrasilTimeStamp*) e de certificados (*ICPBrasilCertificate*). Ele informou que a classe de validação de certificados possui uma dependência do repositório de certificados (*ICPBrasilStore*), ele concluiu que as classes de validação de assinatura possuem uma dependência indireta das classes responsáveis pelo repositório.

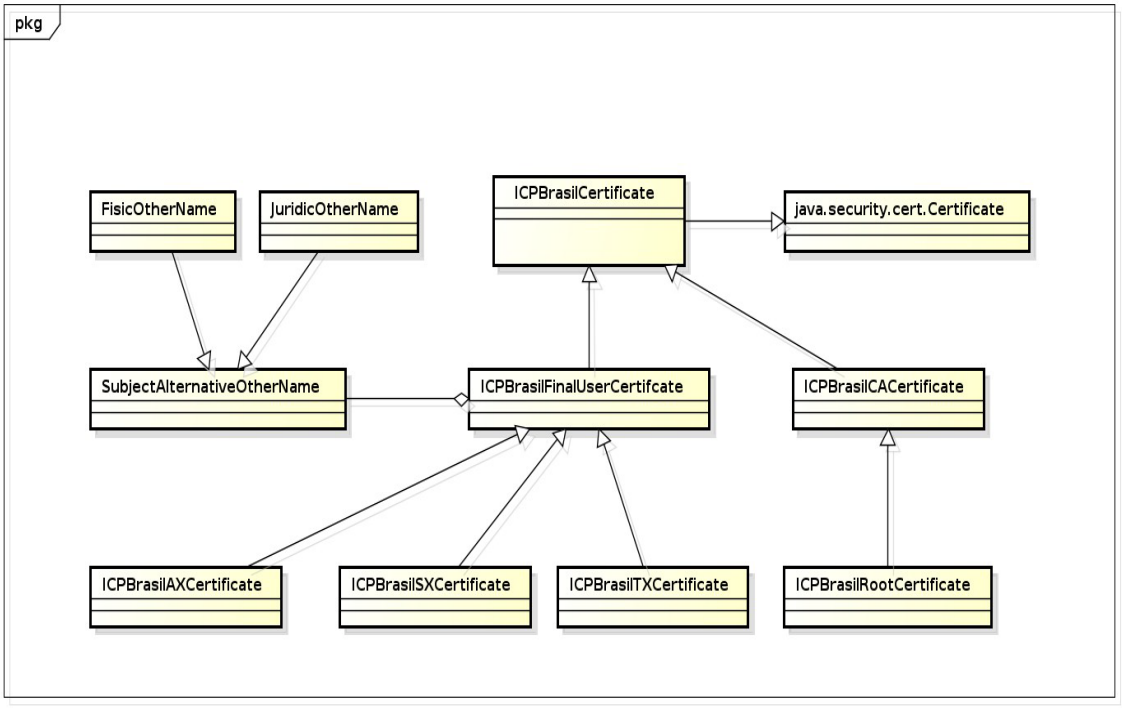
Figura 14 - Diagrama de classes Repositório



Fonte: Tognoli. 2012. p. 37

Ainda Tognoli (2012, p. 38), definiu o diagrama de classes representando a função de validação de certificado. Ele discorreu que foi implementada uma estrutura de classes que separa bem os tipos de certificados definidos pelas políticas ICPBrasil, tornando independentes os processos de validações. O diagrama de classes desenvolvido por ele encontra-se disponível através da figura 15:

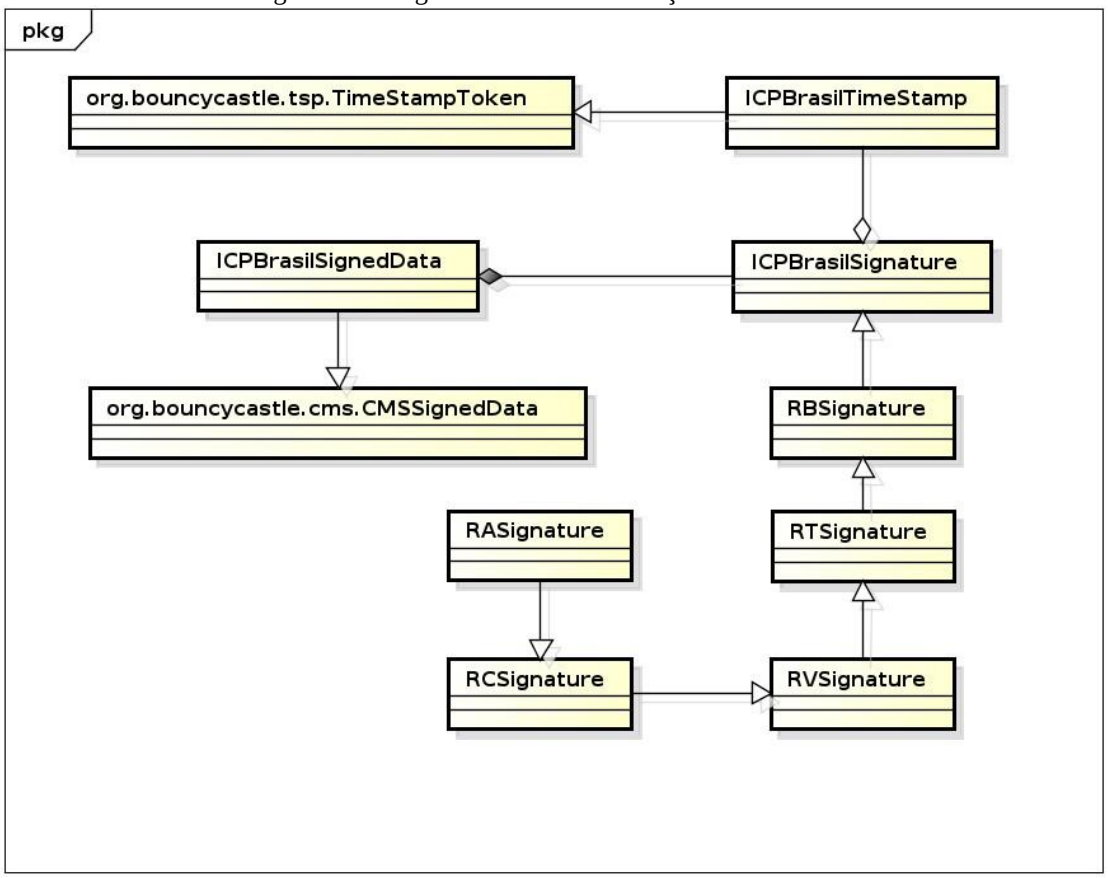
Figura 15 - Diagrama de Classes validação de certificado



Fonte: Tognoli. 2012. p. 38

Por fim, Tognoli (2012, p. 38) definiu também um diagrama de classes de validação de assinaturas. No diagrama da figura 16 está representada a estrutura das classes para a validação de CMSs e carimbos de tempo. Neste também se encontra a estrutura de classes utilizada na verificação e validação de certificados. Ele informou que os padrões de assinatura se relacionam com herança, reaproveitando os métodos de validação genéricos. A figura 16 possui o diagrama dessas estruturas:

Figura 16 - Diagrama de classes validação de assinaturas



Fonte: Tognoli. 2012. p. 39

Teotônio (2012, p. 52) discorreu sobre a organização da API de assinatura digital da seguinte maneira:

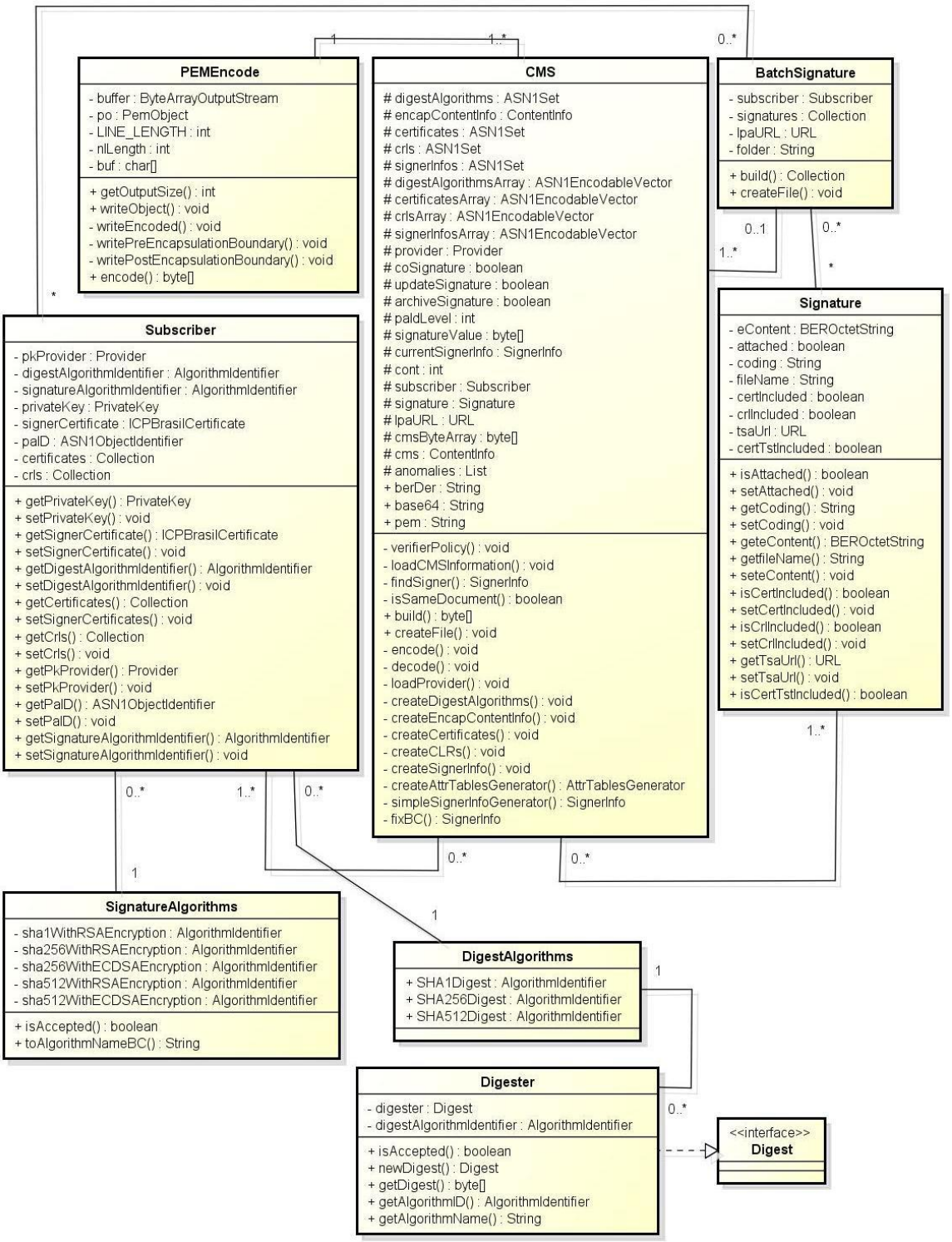
> "A API foi estruturada da seguinte forma, existe uma classe CMS que é a responsável por gerar todos os elementos do CMS: o CMSVersion, o DigestAlgorithmIdentifiers, o Encapsulated ContentInfo, o CertificateSet, o CertificateRevocationLists e o SignerInfos. Essa classe recebe as informações de entrada a partir de objetos das classes:
> a) Subscriber: classe que armazena as informações pertinentes ao assinante;
> b) Signature: classe que armazena as informações pertinentes à informação a ser assinada.
>
> Essa separação é feita para facilitar a assinatura em lote, que é realizada através da classe BatchSignature, dessa maneira basta N objetos Subscriber e um Signature para efetuar esse tipo de assinatura."

Foi também verificado que a API possui suporte para assinatura simples,co-assinatura, assinatura em lote e redefinição de assinatura. Todas geradas pela classe CMS.

Teotônio construiu ainda alguns diagramas de alguns pacotes com suas descrições:

"... O pacote signatures contém as principais classes da API, elas são utilizadas por praticamente todas as funções disponibilizadas pela ferramenta, de maneira quase idêntica.". (Teotônio, 2012 p. 54). A figura 17 representa o diagrama de classes deste pacote.

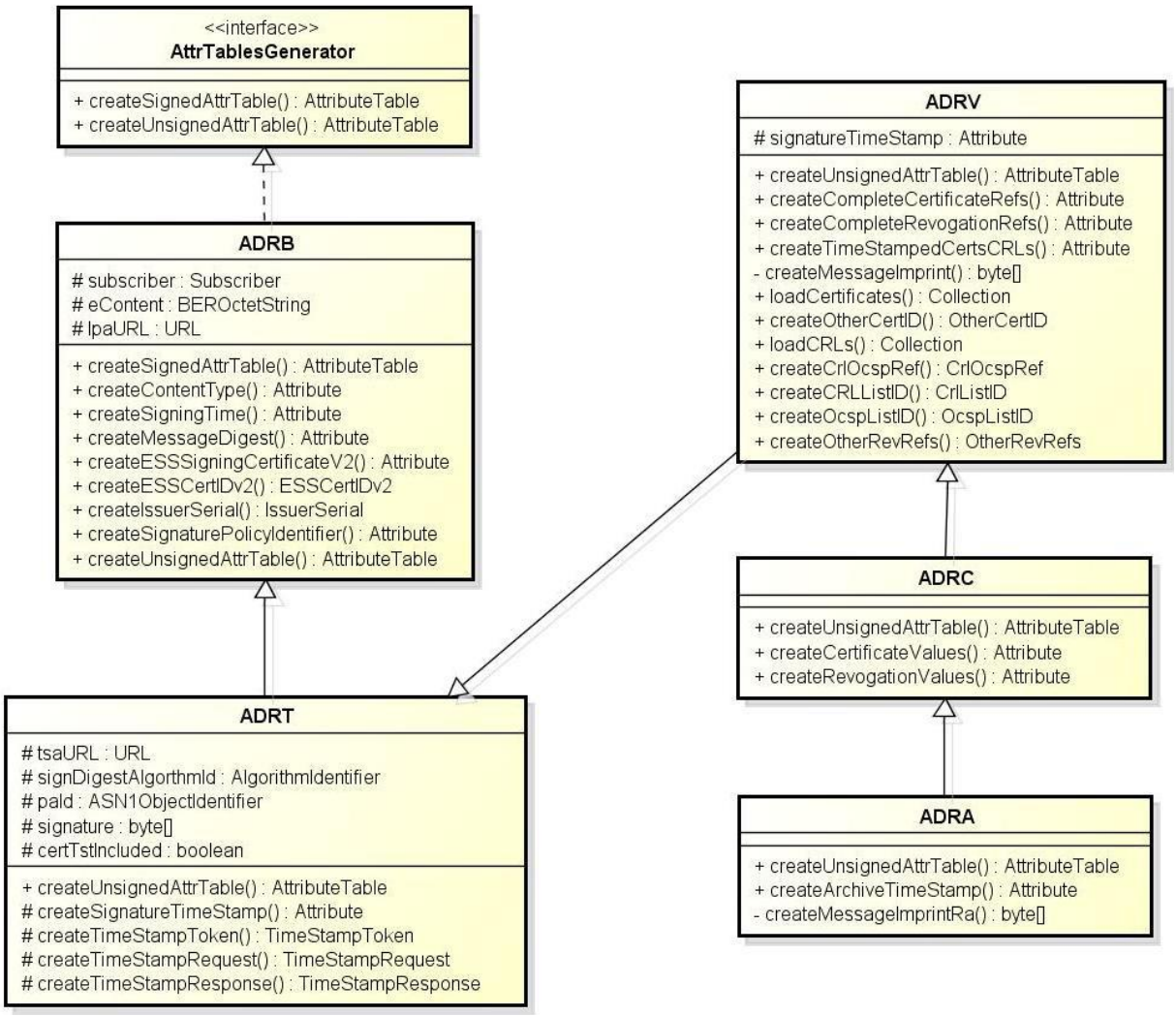
Figura 17 - Diagrama de classe do pacote signatures



Fonte: Teotônio. 2012. p. 55

"O pacote *formats* possui as classes responsáveis pela criação da tabelas de atributos assinados e não assinados, elas herdam umas as outras em ordem incremental de acordo com os formatos ICP-Brasil;" (Teotônio, 2012 p. 56). A figura 18 representa o diagrama de classes deste pacote.

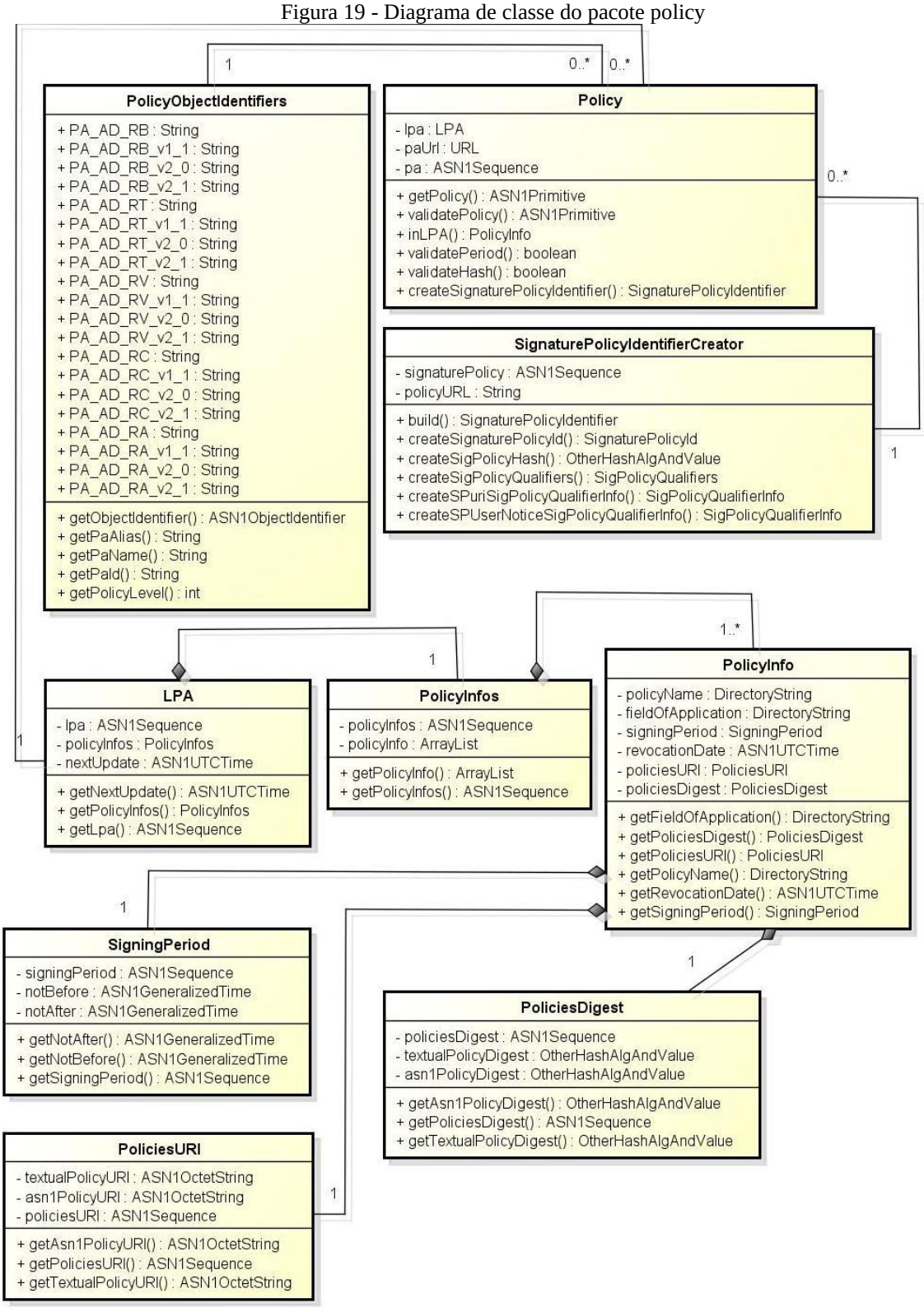
Figura 18 - Diagrama de classe do pacote formats



Fonte: Teotônio. 2012. p. 56

"O pacote *policy* possui as classes pertinentes as politicas da ICP-Brasil, implementa a estrutura ASN.1 da LPA, que é exclusiva da ICP-Brasil, além de implementar o SignaturePolicyIdentifier;". (Teotônio, 2012 p. 56). A figura 19 representa o diagrama de classes deste pacote.

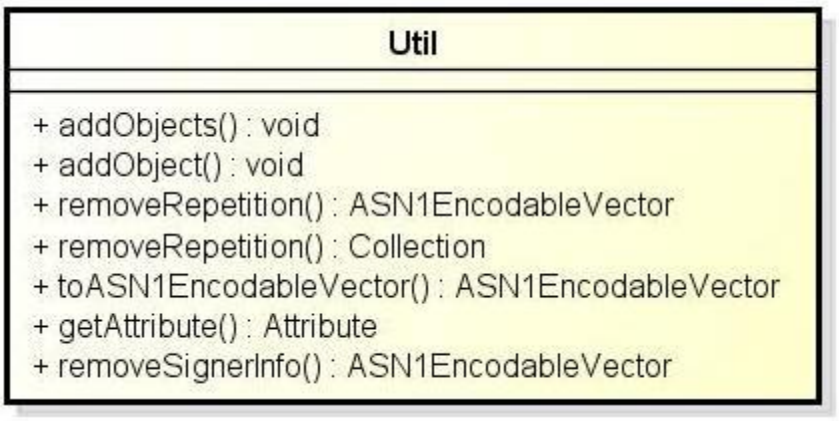
Figura 19 - Diagrama de classe do pacote policy



Fonte: Teotônio. 2012. p. 57

"O pacote *util* possui apenas uma classe, esse por sua vez possui atributos estáticos que fornecem funções genéricas úteis a API como um todo." (Teotônio, 2012 p. 58). A figura 20 representa o diagrama de classes deste pacote.

Figura 20 - Diagrama de classe do pacote útil



Fonte: Teotônio. 2012. p. 58

Foi verificado que o projeto original do Assinador Digital Completo se encontra, bastante estruturado, porém é um código bastante extenso e cheio de dependências de classes e API', s. Após este breve estudo feito este trabalho estará focado na adaptação das técnicas relacionadas ao reconhecimento e verificação de certificados digitais presentes neste Assinador Completo para a plataforma Android.

Capítulo 04 – Resultados Obtidos e desenvolvimento.

Neste capítulo, serão discutidos os resultados obtidos. Mostrando os passos do desenvolvimento e adequação dos códigos fontes para a plataforma Android.

4.1 – Objetivos alcançados

O objetivo principal deste trabalho é executar uma migração de parte de um código fonte desenvolvido para a plataforma Windows através da linguagem de programação Java, para uma aplicação desenvolvida para a plataforma Android, que se utiliza da mesma linguagem de programação base (linguagem Java). Considerando que as aplicações em Android são desenvolvidas na linguagem Java, esta migração pode ser considerada com uma alternativa, a fim de evitar o desenvolvimento do início de uma nova aplicação para a plataforma Android.

De forma mais específica, desejava-se que a parte de verificação de certificados digitais estivesse implementada para a plataforma Android, esta parte dos objetivos se baseia num desejo futuro da migração completa do Assinador Digital para esta plataforma móvel.

Assim o primeiro passo executado foi a extração do código responsável por esta funcionalidade para um novo projeto desenvolvido na ferramenta Eclipse, se utilizando do SDK para desenvolvimento Android.

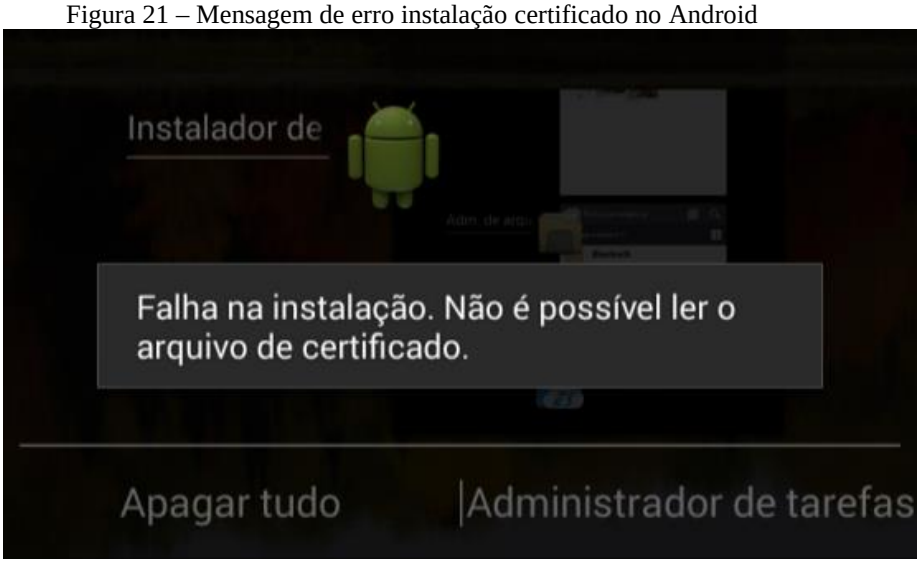
Esta migração foi executada com sucesso, inclusive vários outros pacotes que não estavam diretamente envolvidos com a verificação de certificados foram transferidos, a fim de adiantar o trabalho futuro de transferir a aplicação inteira.

Durante a migração percebeu-se que a maioria das bibliotecas e APIs utilizadas no programa original foram reconhecidas normalmente no projeto Android, inclusive a biblioteca *Boucy Castle Criptografhy Library*, que é uma das responsáveis pela implementação dos métodos de criptografia na aplicação.

No projeto Android, além de todos os pacotes e bibliotecas importadas, foi criada apenas uma tela, com a finalidade de exibir o *Status* do certificado, de instalado ou não instalado, a informação do nome do assinante, ou seja, do “dono” do certificado, assim como as informações referentes as anomalias encontradas, se for o caso.

Percebeu-se uma dificuldade na parte de instalação e reconhecimento do certificado na plataforma Android, pois o certificado não era instalado aparecendo a mensagem presente na figura 21:

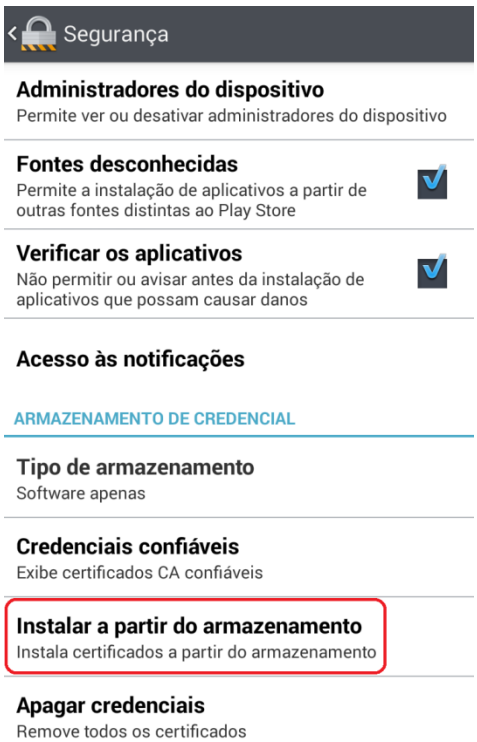
Figura 21 – Mensagem de erro instalação certificado no Android



Fonte: Própria

Após pesquisas, foi verificado que eram necessários dar permissões especiais de segurança nas configurações do próprio dispositivo para que os certificados fossem instalados, como observado e marcado na figura 22:

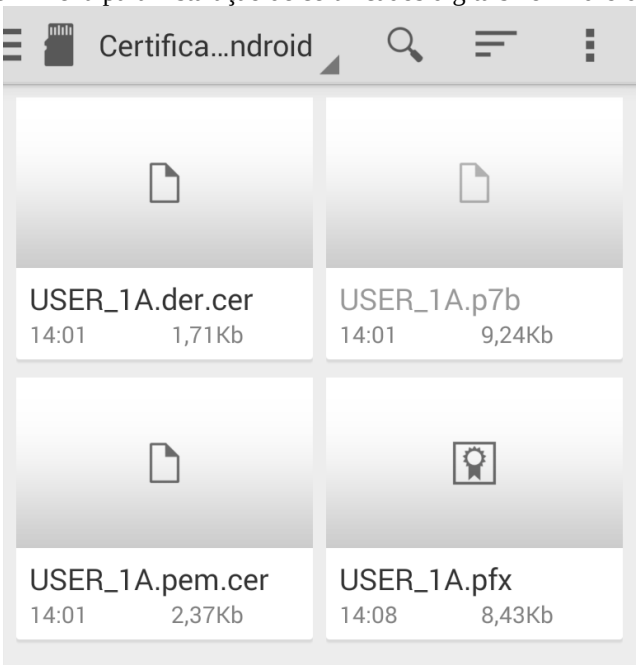
Figura 22 – Configurações de segurança necessárias para instalação do certificado em dispositivos Android



Fonte: Própria

Ao clicar nesta opção destacada em vermelho na figura 22, constatou-se também que os arquivos de certificados deveriam ser abertos e instalados através deste *menu* próprio aberto a partir deste clique como observado na figura 23:

Figura 23 – Menu para instalação de certificados digitais no Android



Fonte: Própria

Na figura 24 é mostrado os certificados instalados no dispositivo.

Figura 24 – Certificados digitais instalados no Android

Fonte: Própria

Após a instalação dos mesmos partiu-se para a parte da programação do certificado que tinha por objetivo capturar as informações deste, para por fim poder analisá-lo, porém a rotina padrão de captura de certificados específica para o sistema operacional Android não encontrava o certificado instalado neste dispositivo. Por exemplo o código abaixo retornava com erro de *Null Pointer Exception*:

```
KeyStore ks = KeyStore.getInstance("AndroidKeyStore");
ks.load(null);
ks.getCertificate(apelido_Certificado);
obs.: apelido_Certificado seria uma string com o alias do certificado.
```

Assim, foram pesquisadas, encontradas e implementadas algumas rotinas com a intenção de garantir a correta instalação do certificado digital na plataforma Android, em sua keystore própria. Abaixo esta rotina encontrada:

```
public boolean getCertsFromP12(String pathToFile, String passphrase) {
    try {
        KeyStore p12 = KeyStore.getInstance("pkcs12");
        p12.load(new FileInputStream(pathToFile), passphrase.toCharArray());
        Enumeration e = p12.aliases();
        while (e.hasMoreElements()) {
            String alias = (String) e.nextElement();
            X509Certificate c = (X509Certificate) p12.getCertificate(alias);
            addCertificateToKeyStore(c);
        }
        return true;
    } catch (Exception e) {
        System.out.println(e.getMessage());
    }
    return false;
}

private boolean addCertificateToKeyStore(X509Certificate c) {
    try {
        KeyStore ks = KeyStore.getInstance("AndroidKeyStore");
        ks.load(null);
        ks.setCertificateEntry("certificado_teste", c);
        return true;
    } catch (Exception e) {
    }
    return false;
}
```

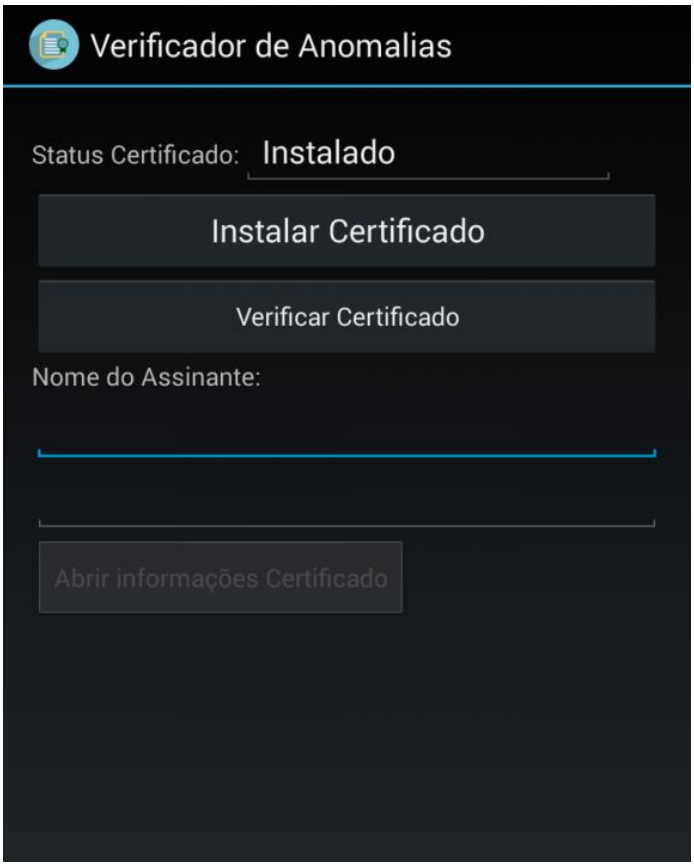
Estas rotinas funcionaram e foi possível capturar as informações do certificado para futura análise do mesmo, utilizando a mesma função de capturas de certificado supracitada.

Foi verificada também a necessidade de criação de uma rotina para criar uma pasta padrão dentro do disco do dispositivo, para que nesta fossem deixados os arquivos de instalação do certificado digital, caso se deseje instalar os mesmos, esta pasta fica localizada dentro do disco principal do dispositivo (memória interna) com o seguinte título “CertificadoAndroid”, nesta pasta ficará salvo também um arquivo de log com diversas

informações disponíveis do certificado verificado, esta parte será explicada com mais detalhes abaixo, ao se apresentar o botão referente ao comportamento desta rotina de geração de log.

A tela responsável pela verificação do certificado foi desenvolvida de forma bastante simples e funcional, como pode ser observado na figura 25:

Figura 25 – Tela da aplicação responsável por exibir as informações do certificado



Fonte: Própria

As funcionalidades desta tela são as seguintes:

- Caixa de texto com o título de “Status do Certificado”. Nesta, será verificado se existe um certificado instalado no dispositivo.
- Botão “Instalar Certificado”. Ao clicar neste botão a aplicação fará uma busca na pasta padrão do sistema e se encontrar algum arquivo de certificado nesta pasta instalará o mesmo.
- Botão verificar certificado. Ao clicar neste botão todas as verificações serão executadas neste certificado encontrado, caso o mesmo exista. Caso o mesmo não exista será exibida na caixa de texto de resultado da verificação o texto “Certificado não instalado”.

- Caixa de texto “Nome do Assinante.”. Após o certificado ser verificado, esta caixa mostrará o nome do assinante do certificado.
- Existe outra caixa de texto que não possui título, nesta serão depositadas as informações referentes as anomalias verificadas no certificado.
- O botão “Abrir Informações Certificado”. Este botão será habilitado após ser executada a verificação em algum certificado. Ao clicar neste botão, será aberto um arquivo de log com diversas informações referentes ao próprio certificado analisado. O arquivo de log será aberto no navegador de arquivo “.txt” padrão do dispositivo, caso não haja nenhum selecionado será dada a opção de escolher algum aplicação do dispositivo para abrir este arquivo de log.

Após ser clicado no botão “verificar Certificado” as informações referentes àquele certificado, assim como as anomalias encontradas no mesmo serão exibidas na tela, como pode ser observado na figura 26:

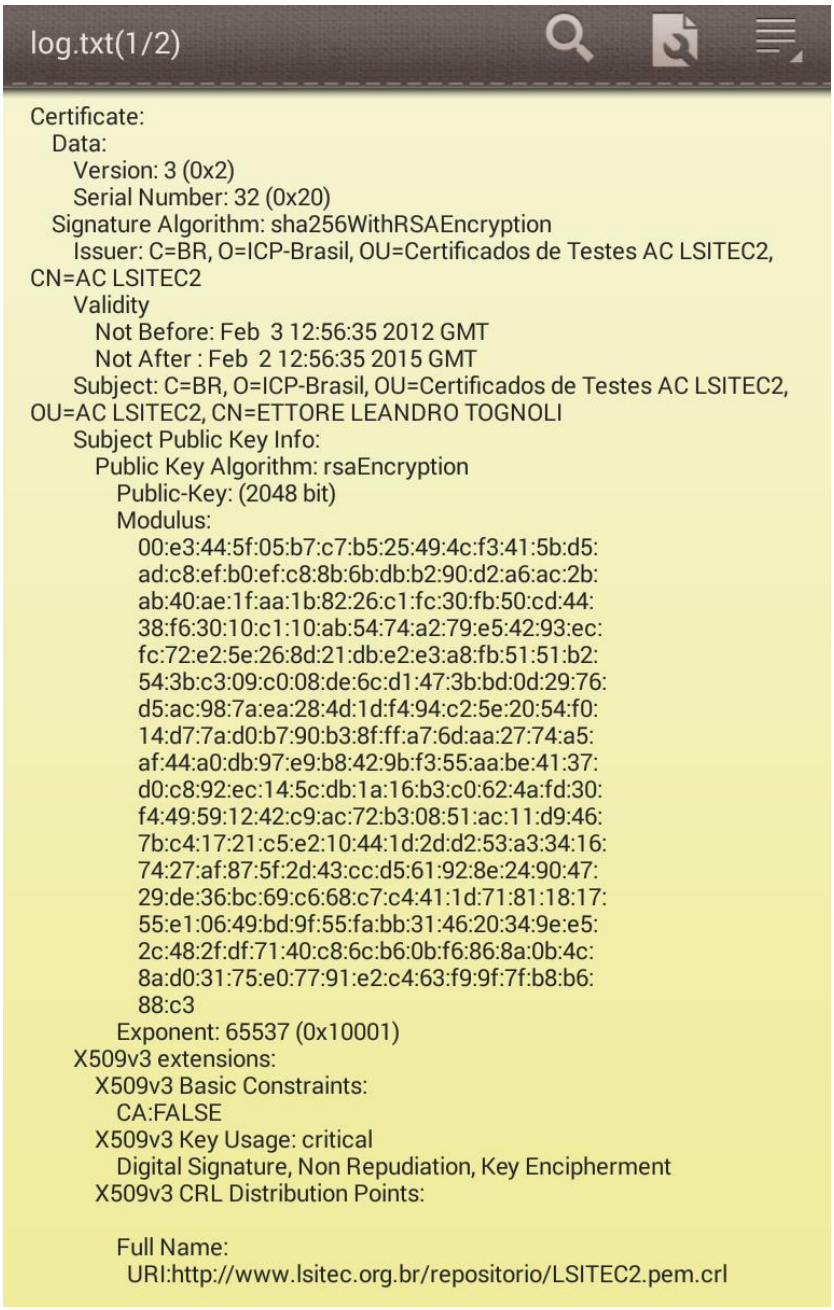
Figura 26 – Tela da aplicação com as anomalias encontradas

Fonte: Própria

Para a verificação das anomalias foram utilizadas as regras já existentes no projeto original, tendo que ser feitas algumas adaptações para que estas verificações funcionassem de forma correta na plataforma Android.

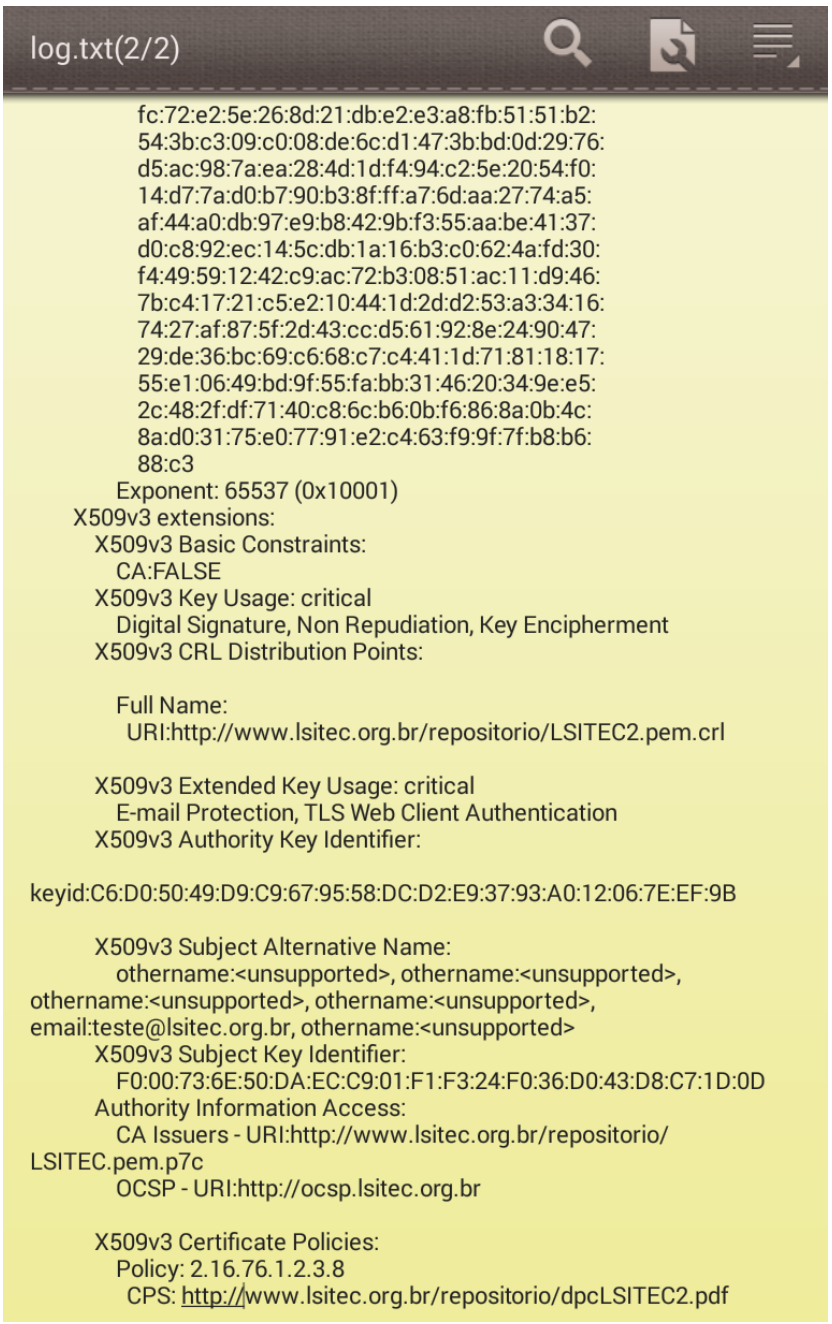
Após clicar no botão “Verificar Certificado”, o botão “Abrir Informações Certificado” é habilitado e ao se clicar nele o arquivo “log.txt” gerado ao fazer a análise do certificado é aberto, com algum leitor de arquivos”.txt” do próprio dispositivo, como por exemplo nas figuras 27 e 28:

Figura 27 – Informações do log, parte 01



Fonte: Própria

Figura 28 – Informações do log, parte 02



Fonte: Própria

A tabela 02 lista as principais informações, e o significado das mesmas, que foram extraídas através do log gerado (Figura 27 e 28).

Tabela 02 – Principais informações extraídas do certificado
Fonte: Open SSL/Própria (Tradução Nossa)

Principais informações Certificado		
Campo	**Significado**	**Informação**
Signature Algorithm	Algoritmo de Assinatura	sha256WithRSAEncryption
Issuer	Emissor do certificado	C=BR, O=ICP-Brasil, OU=Certificados de Testes AC LSITEC2, CN=AC LSITEC2
Validity	Validade do Certificado, este certificado é válido somente no dia 02 de fevereiro de 2012	Not Before: Feb 3 12:59:26 2012 GMT Not After : Feb 2 12:59:26 2015 GMT,
Subject	Assuntos do certificado, inclusive nome do assinante	C=BR, O=ICP-Brasil, OU=Certificados de Testes AC LSITEC2, OU=AC LSITEC2, CN=Ettore Leandro Tognoli
Public Key Algorithm:	Algoritmo de criptografia de chave pública	RsaEncryption
Public-Key: (2048 bit)	Tamanho da chave pública	2048 bits
X509v3 Basic Constraints: CA:FALSE	Restrições do Certificado, um certificado de usuário final normalmente define o campo CA para FALSE	CA:FALSE
X509v3 Key Usage:	Utilizações do certificado, consiste em uma lista de nomes dos principais usos permitidos	Critical Digital Signature, Non Repudiation, Key Encipherment
X509v3 CRL Distribution Points	"Pontos de distribuição" dos certificados, repositório	URI:http://www.lsitec.org.br/repositorio/LSITEC2.pem.crl
X509v3 Extended Key Usage	Utilizações do certificado extendidas, consiste em extensões de uma lista de utilizações indicando que o certificado de chave pública pode ser usado para outros fins.	E-mail Protection, TLS Web Client Authentication
Authority Key Identifier:	Seção que lista opções para certificados de usuário final, diferente da seção CA	keyid:C6:D0:50:49:D9:C9:67:95:58:DC:D2:E9:37:93:A0:12:06:7E:EF:9B
X509v3 Subject Alternative Name	A extensão de nome alternativo do sujeito permite que vários valores literais sejam incluídos no arquivo de configuração . Estas incluem e-mail (um endereço de e-mail) URI um indicador de recursos uniforme, DNS (nome de domínio DNS), entre outros	othername:<unsupported>, othername:<unsupported>, othername:<unsupported>, othername:<unsupported>, email:teste@lsitec.org.br, othername:<unsupported>
Authority Information Access:	A extensão de acesso a informações da autoridade dá detalhes sobre como acessar certas informações relativas à CA.	CA Issuers - URI:http://www.lsitec.org.br/repositorio/LSITEC.pem.p7c OCSP - URI:http://ocsp.lsitec.org.br
Certificate Policies:	Políticas do certificado, cada certificado pode possuir sua própria política e sintaxe apropriada	Policy: 2.16.76.1.2.3.8

Todas estas consultas e implementações foram executadas tomando como base os códigos e classes já existentes no projeto original do Assinador Digital, estes códigos foram apenas tratados para conseguirem ser acessados e retornar informações solicitadas no projeto da plataforma Android.

Na parte de anexos deste trabalho, mais especificamente na página 61, existe o anexo A. Neste anexo estão disponíveis os principais códigos criados e modificados para o correto funcionamento do verificador de certificados digitais na plataforma Android, assim, caso se deseje obter mais detalhes sobre as implementações executadas, deve-se verificar este anexo. Lembrando que apenas os códigos desenvolvidos para este trabalho estão neste anexo, os códigos e verificações referentes ao Assinador Digital completo para o sistema operacional Windows não estão presentes nele, pois não foram desenvolvidos neste trabalho.

Capítulo 05 - Conclusão

Como mencionado anteriormente, um dos objetivos principais do projeto era verificar se a adaptação das técnicas existentes na aplicação de assinatura digital era possível e/ou viável para a plataforma Android.

Conclui-se que sim, esta adaptação é possível de ser executada. Como o código do assinador está num nível bom de normatização e organização de métodos, classes e pacotes, esta adaptação é completamente possível. É óbvio que existem problemas quando se faz um trabalho deste tipo, existem chamadas que são executadas aos componentes de tela da aplicação do desktop, componentes como por exemplo JFrame, JPanel, etc. No caso destas chamadas as mesmas devem ser desconsideradas e reescritas, o que se mostrou ser bastante trabalhoso e em alguns casos bem difícil.

Além destas situações, as chamadas a diretórios, funções, ou qualquer outra chamado específica do sistema operacional devem ser desconsideradas e refeitas. Foi verificado também que todas as bibliotecas e APIs relacionadas a criptografia e assinatura digital, funcionaram normalmente para a plataforma Android, tanto a API Boucy Castle Criptografhy Library, quanto as demais APIs criadas e/ou implementadas/adequadas para o funcionamento do assinador original.

É possível afirmar também que esta implementação é viável, visto que as regras e classes já estão prontas, sendo necessário a importação, tratamento e reutilização das mesmas. Provavelmente o tempo para executar estes trabalhos deverá ser menor que o desenvolvimento das regras desde o principio.

Assim, conclui-se que o objetivo do trabalho foi alcançado, tanto na parte de implementar parte do código para a plataforma Android, quanto na verificação da possibilidade e viabilidade.

Em relação a trabalhos futuros que possam ser executados tomando este como base, têm-se algumas considerações e sugestões.

A implementação do projeto do verificador de certificados dentro da plataforma Android se mostrou completamente possível, e satisfatoriamente viável, assim a principal sugestão é que seja implementado o assinador completo, utilizando este trabalho como base.

Dentro do projeto desenvolvido e adequado já foram importados a maioria dos pacotes do projeto original, porém os mesmos não estão sendo utilizados para verificação dos certificados, sendo que esta parte foi executada apenas com a intenção de adiantar este trabalho futuro da implementação do assinador completo e verificar a compatibilidade das bibliotecas e pacotes.

Ao importar todo o restante do código do assinador, o ideal é que haja uma revisão da organização dos pacotes, a fim de organizar melhor o código dentro do Android.

Por fim sugere-se também que haja uma migração do projeto para a *IDE* Android Studio, visto que o suporte ao *plugin* do Android dentro da IDE Eclipse está sendo finalizado no final do ano de 2015.

Bibliografia

SILVA, Alexandre Ferreira da. MARTINS, Renato Marinho. **Criptografia: aspectos históricos e matemáticos.** Trabalho de Conclusão de Curso apresentado como requisito parcial para a obtenção do grau de Licenciatura em Matemática, Universidade do Estado do Pará. Belém, 2011.

GUELFI, Adilson Eduardo. **Apresentações de slides utilizadas na ministração de treinamento em processos e conceitos pertinentes a assinatura digital no Brasil.** Marília, 2012.

TOGNOLI, Éttore Leandro. **VERIFICADOR DE CERTIFICADOS E ASSINATU-RAS ICP-BRASIL EM JAVA.** Trabalho de conclusão de curso apresentado ao Curso de Bacharelado em Ciência da Computação, do Centro Universitário Eurípides de Marília, como parte dos requisitos necessários para a obtenção do grau de Bacharel em Ciência da Computação. Marília, 2012.

NORTHCUTT, Stephen et al. **Desvendando segurança em redes: O guia definitivo para fortificação de perímetros de rede usando Firewalls, VPNs, roteadores e sistemas de detecção de invasores.** 1. ed. Rio de Janeiro: Campus, 2002.

OLIVEIRA, Ronielton Rezende. **Criptografia simétrica e assimétrica - os principais algoritmos de cifragem.** Revista Segurança Digital, Brasília, DF, v. 6, p. 21 - 24, 31 maio 2012.

TEOTÔNIO, Cleverson Abreu. **Implementação de uma API Java de geração de assinaturas digitais de acordo com as regulamentações da ICP-Brasil.** Trabalho de conclusão de curso apresentado ao Curso de Bacharelado em Ciência da Computação, do Centro Universitário Eurípides de Marília, como parte dos requisitos necessários para a obtenção do grau de Bacharel em Ciência da Computação. Marília, 2012.

STALLINGS, William. **Criptografia e segurança de redes: princípios e práticas**. 4. ed. São Paulo: Pearson Prentice Hall, 2008.

MORENO, Edward David; PEREIRA, Fábio Dacêncio; CHIARAMONTE, Rodolfo Barros. **Criptografia em Software e Hardware.** 1.ed. São Paulo: Novatec Editora Ltda., 2005.

MATEUS, Geraldo Robson; LOUREIRO, Antônio Alfredo Ferreira. **Introdução A Computação Móvel.** Rio de Janeiro, RJ: 11a. Escola de Computação, 1998. v. 1. 200p.

TONIN, Graziela Simone. **Tendências em computação móvel**. Monografia realizada na disciplina de Computação Móvel na Universidade de São Paulo, como parte do programa de Pós-graduação em Ciência da Computação. São Paulo, 2012.

ROVADOSKY, Douglas Samuel. Et al. **Uma ferramenta de realidade aumentada usando dispositivo móvel com sistema operacional Android**. Revista Brasileira de Computação Aplicada (ISSN 2176-6649), Passo Fundo, v. 4, n. 1, p. 25-37, mar. 2012 25.

PEREIRA, Lúcio Camilo Oliva; SILVA, Michel Lourenço da. **Android para desenvolvedores.** Rio de Janeiro 2009.

RODRIGUES, Guilherme Rodrigues E. **Smartphones e suas tecnologias.** Trabalho de conclusão de curso apresentado a Escola de Engenharia de São Carlos, da Universidade de São Paulo. São Carlos, 2009.

IDC home Page**" Smartphone OS Market Share",** disponível em < http://www.idc.com/prodserv/smartphone-os-market-share.jsp>, acessado em Novembro de 2015.

Site ICP-Brasil, disponível em < http://www.iti.gov.br/icp-brasil>, acessado em Maio de 2015.

ICP-BRASIL. **DOC-ICP-15.01: REQUISITOS PARA GERAÇÃO E VERIFICAÇÃO DE ASSINATURAS DIGITAIS NA ICP-BRASIL**. Versão 2.1, julho de 2012.

ARAÚJO, Rafael Will Macedo de. **Autenticação e comunicação segura em dispositivos móveis de poder computacional restrito.** Dissertação apresentada ao instituto de

Matemática e Estatística da Universidade de São Paulo para obtenção do Título de Mestre em Ciências. São Paulo, 2013.

MAGRI, Bernardo Caraponale. **Assinatura digital Rabin-Williams sem randomização e com prova eficiente de segurança.** Dissertação apresentada ao instituto de Matemática e Estatística da Universidade de São Paulo para obtenção do Título de Mestre em Ciências. São Paulo, 2012.

OLIVEIRA, Igor Carvalho. **SISTEMA WEB DE REGISTRO DE PRESENÇA VIA LEITURA BIOMÉTRICA DE IMPRESSÃO DIGITAL PARA INSTITUIÇÕES DE ENSINO.** Trabalho apresentado ao Centro Universitário de Brasília (UniCEUB) como pré-requisito para a obtenção de Certificado de Conclusão de Curso de Engenharia de Computação. Brasília, 2011.

Frohlich ,Jonata; Ignaczak ,Luciano. **Uma análise dos certificados digitais para assinatura de código de aplicações de diferentes segmentos na plataforma Android**. Artigo da Universidade do Vale do Rio dos Sinos(UNISINOS). . disponível em http://sbseg2015.univali.br/content/uploads/files/wticg/completos/147999.pdf>.

SILVEIRA, I.F. **Java, das torradeiras à Internet**. Disponível em <http://www.infowester.com/lingjava.php>, publicado em *30/06/2003, acessado em* 30-09-2015;

Dias, Luiz Gustavo. **Fundamentos de Criptografia.** disponível em < http://www.fundacaoaprender.org.br/fundamentos-de-criptografia>. Acessado em 30-09-2015;

Sobral, Fábio. **Certificação digital.** Disponível em < http://biblioo.info/certificacao-digital/>. Acessado em 14-11-2015.

Reis, Fabio dos. **Introdução à criptografia - Bóson Treinamentos**. disponível em < http://pt.slideshare.net/bosontreinamentos/introduo-criptografia-35524633>. Acessado em 14-11-2015.

Palmeira, Thiago Vinícius Varallo. **Java: história e principais conceitos.** disponível em <http://www.devmedia.com.br/java-historia-e-principais-conceitos/25178>. Acessado em 14-11-2015.

The Legion of the Bouncy Castle. Disponível em : <https://www.bouncycastle.org/documentation.html> Acessado em 15-09-2015.

Open SSL. **x509v3_config**. Disponível em:<https://www.openssl.org/docs/manmaster/apps/x509v3_config.html>. Acessado em 18-11-2015.

Carvalho, Hugo Eiji Tibana. **PKI - Infra-estrutura de Chaves Públicas**. Trabalho desenvolvido para a disciplina Redes de Computadores II, da UFRJ, no período 2008.2. Universidade Federal do Rio de Janeiro. Disponível em: <http://www.gta.ufrj.br/ensino/eel879/trabalhos_vf_2008_2/hugo/Hashing.html.> Acessado em 15-11-2015.

MENDES, Aliane Veloso. **Estudo de Criptografia com Chave Pública Baseada em Curvas Elípticas. 2007**. 50 f. Monografia (Bacharel em Sistemas de Informação) – Universidade Estadual de Montes Claros, Montes Claros, 2007.

Anexo A – Principais códigos fontes pesquisados e utilizados para garantir o funcionamento do reconhecimento de certificados na Plataforma Android

01 - Função responsável pela criação da pasta padrão do sistema para armazenamento de certificados e arquivo de log.

```
private void criarPastaPadrao() {

    File folder = new File(Environment.getExternalStorageDirectory() + "/CertificadoAndroid");
    if (!folder.exists()) {
        folder.mkdir();
    }

}
```

02 – Função de evento do click no Android do botão “Instalar Certificado”, verificando a existência de certificado na pasta padrão do sistema para instalação e tentando instalar o mesmo.

```
btInstalarCertificado.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(final View v) {

        if (Certificate.nomeArquivo.toString().contains(".pfx"))
            Toast.makeText(getApplicationContext(),
                    "Não existe nenhum certificado para ser instalado",
                    1).show();
        else {
            if (instalarCertificado()) {
                Toast.makeText(getApplicationContext(),
                        "Certificado Instalado com sucesso!!", 1)
                        .show();
            } else
                Toast.makeText(
                        getApplicationContext(),
                        "Houve Problemas na instalação do certificado!!",
                        1).show();
        }

    }
});
```

03 – Funções responsáveis pela instalação do certificado:

```
public boolean instalarCertificado() {

    try {

        if (Certificate.nomeArquivo.length > 0)
            return InstallCert.getCertsFromP12(Certificate.endPasta + Certificate.nomeArquivo, "1234");

    } catch (Exception e) {
        // TODO: handle exception
    }

    return false;

}
```

```
public boolean getCertsFromP12(String pathToFile, String passphrase) {
    try {
        KeyStore p12 = KeyStore.getInstance("pkcs12");
        p12.load(new FileInputStream(pathToFile), passphrase.toCharArray());
        Enumeration e = p12.aliases();
        while (e.hasMoreElements()) {
            String alias = (String) e.nextElement();
            X509Certificate c = (X509Certificate) p12.getCertificate(alias);
            addCertificateToKeyStore(c);
        }
        return true;
    } catch (Exception e) {
        System.out.println(e.getMessage());
    }
    return false;
}

private boolean addCertificateToKeyStore(X509Certificate c) {
    try {
        KeyStore ks = KeyStore.getInstance("AndroidKeyStore");
        ks.load(null);
        ks.setCertificateEntry("certificado_teste", c);
        return true;
    } catch (Exception e) {
    }
    return false;
}
```

04 – Função de evento do click no Android do botão “Verificar Certificado”, verificando as anomalias presentes no certificado instalado na KeyStore do Android:

```
btVerifica.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(final View v) {

        try {
            verificaAnomaliasCertificado();
            btinfocert.setEnabled(true);
        } catch (KeyStoreException | NoSuchAlgorithmException
                | CertificateException | IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
});
```

05 – Funções que verificam as anomalias no certificado, caso o mesmo esteja instalado e pegam o nome do assinante:

```
    public void verificaAnomaliasCertificado() throws KeyStoreException,
            NoSuchAlgorithmException, CertificateException, IOException {
        Certificate nomeAssinante = new Certificate() {
        };
        if (verifica_certificado) {

            String anomalias = null;
            anomalias = listarAnomalias();
            edtResultado.setText(anomalias);
            edtNomeAssinante.setText(nomeAssinante.getAssinante());

        } else {
            edtResultado.setText("Certificado não instalado!");
        }
    }
}
```

```
public String listarAnomalias() throws KeyStoreException,
        NoSuchAlgorithmException, CertificateException, IOException {

    int numNomalia = 1;
    String anomalias = "Anomalias Encontradas: \n";
    String montaranomalia = null;

    Certificate certificado = new Certificate() {
    };

    LinkedList<CertificateAnomaly> anomaliasCert = new LinkedList<>();
    anomaliasCert = certificado.getValidations();

    if (anomaliasCert.size() > 0) {
        while (anomaliasCert.size() > 0) {
            montaranomalia = anomaliasCert.getFirst().toString();
            anomalias += numNomalia + " - " + (String) montaranomalia.subSequence(
                        montaranomalia.indexOf("message = ") + 10,
                        montaranomalia.indexOf("}")) + ";" + "\n";
            anomaliasCert.removeFirst();
            numNomalia++;
        }
        numNomalia--;
        anomalias += "Total: " + numNomalia + " Anomalias.";
    } else
        anomalias = "Nenhuma Anomalia Encontrada!";

    return anomalias;
}
```

```
    public LinkedList<CertificateAnomaly> getValidations()
            throws KeyStoreException, NoSuchAlgorithmException,
            CertificateException, IOException {

        ICPBrasilCertificate signCert = initKeystore();

        LinkedList<CertificateAnomaly> lmodel = new LinkedList<>();

        lmodel = (LinkedList<CertificateAnomaly>)signCert.getAnomalys();

        return lmodel;
    }

    public String getAssinante() throws KeyStoreException,
            NoSuchAlgorithmException, CertificateException, IOException {
        String nomeAssinante = null;
        nomeAssinante = initKeystore().getSubjectSimpleName().toString();
        return nomeAssinante;

    }

}
```

04 – Função de evento do click no Android do botão “Abrir Informações Certificado”, que abre o arquivo de log se o mesmo estiver salvo na pasta padrão do sistema:

```
    btinfocert.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(final View v) {
            Toast.makeText(getApplicationContext(),
                    "Abrindo arquivo de log", 1).show();
            try {
                abrirlog();

            } catch (Exception e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }
    });
}
```

```
public void abrirlog() {

    try {

        if (Certificate.folder.exists()) {

            File file = new File(Certificate.folder + "/log.txt");
            if(file.length()>0){
            Intent intent = new Intent();
            intent.setAction(android.content.Intent.ACTION_VIEW);
            Uri data = Uri.fromFile(file);
            String type = "*/*";
            intent.setDataAndType(data, type);
            startActivity(intent);
            }
            else
                Toast.makeText(getApplicationContext(),
                        "Não existe log para ser exibido!!", 1).show();}

    } catch (Exception e) {
        Toast.makeText(getApplicationContext(),
                "Problema ao abrir o LOG de informações!!", 1).show();
    }

}
```

Obs.:

01 - Todas as demais funções utilizadas para verificação das anomalias pertencem ao projeto original, sendo que este é o caminho percorrido para instalação e verificação de certificados no Android que a aplicação está utilizando.

02 - O código de instalação de certificado não é de completa autoria do autor deste trabalho, ele foi pesquisado originalmente no seguinte site:

Site: http://stackoverflow.com/questions/31713011/import-p12-file-into-androidkeystore, acessado em 10/10/2015;

Anexo B – Plataforma Xamarin - Uma das alternativas ao desenvolvimento Mobile em linguagem C# para IOS e Android

Durante o desenvolvimento deste trabalho, nos foi sugerido a utilização da plataforma Xamarin, para o desenvolvimento da aplicação objeto do mesmo. Na época desta sugestão a plataforma escolhida era a plataforma Android e esta se manteve até a finalização do projeto, pois a mesma utiliza a linguagem Java para programar suas aplicações e como a aplicação objeto inicial do estudo estava escrita em Java, nada mais natural que a escolha da plataforma Android.

Apesar da Xamarin não ter sido escolhida para implementação deste, foi executada uma pequena pesquisa mostrando as principais características e vantagens do uso desta plataforma.

Abaixo estão algumas informações sobre esta plataforma extraídas do site: http://comocriaraplicativos.com.br/entendendo-a-xamarin/, acessado em 15/09/2015.

> "A plataforma Xamarin consiste em uma série de elementos que lhe permitem desenvolver aplicativos para iOS e Android:
>
> - Linguagem C# – Permite que você use uma sintaxe familiar e recursos sofisticados como Generics, Linq e da Biblioteca Task Parallel.
> - .NET Framework Mono – Fornece uma implementação de plataforma cruzada dos extensos recursos do framework .NET da Microsoft.
> - Compilador- Dependendo da plataforma, produz um aplicativo nativo (por exemplo iOS) Ou um aplicativo .NET integrado e em tempo de execução (por exemplo, o Android).
> - Ferramentas de IDE – O Xamarin Studio IDE e o Xamarin plug-in para o Visual Studio permitem criar, construir e implantar projetos Xamarin.
>
> Além disso, porque a linguagem subjacente é C# com o framework .NET, os projetos podem ser estruturados para compartilhar código que também pode ser implantado para o Windows Phone."
>
> **"Compilação**
>
> A fonte C# faz o seu caminho em um aplicativo nativo de maneiras muito diferentes em cada plataforma:
>
> - iOS – C# é AOT compilado para a linguagem do grupo ARM. O framework .NET está incluído, com classes não utilizadas sendo retiradas durante a ligação para reduzir o tamanho do aplicativo. A Apple não permite a geração de código de tempo de execução em iOS, portanto, algumas características da linguagem não estão disponíveis (ver Xamarin.iOS limitações).
> - Android – C# é compilado para IL e embalado com MonoVM + JIT'ing. As classes não utilizadas são retiradas durante a ligação. O aplicativo é executado lado-a-lado

com Java/Dalvik e interage com os tipos nativos via JNI (ver Xamarin.Android limitações).

- Windows Phone – C# é compilado para IL e executado pelo tempo de execução built-in, e não requer ferramentas Xamarin. Projetando aplicativos do Windows Phone seguindo a orientação de Xamarin torna-se mais simples para reutilizar o código em iOS e Android."

Abaixo estão os requisitos, inclusive de software e hardware, necessários para utilização desta *IDE* nas diferentes plataformas:

"iOS

Desenvolver aplicativos iOS requer um computador Mac, executando o Mac OS X. Você também pode usar o Visual Studio para escrever e implementar aplicativos iOS com iOS plugin do Visual Studio de Xamarin. No entanto, um Mac ainda é necessário para fins de build, publicação e licenciamento.

O Xcode IDE da Apple deve ser instalado para fornecer o compilador e simulador para teste. Para testar em um dispositivo real e apresentar candidaturas para a distribuição, você deve se juntar Developer Program da Apple (US$ 99 USD por ano). Cada vez que você enviar ou atualizar um aplicativo, ele deve ser analisado e aprovado pela Apple antes e ser disponibilizado para os clientes baixarem.

O código é escrito com a Xamarin Studio ou Visual Studio IDE e layouts de tela podem ser construídos de forma programática ou editado com iOS Designer de Xamarin em qualquer IDE."

"Android

O desenvolvimento de aplicativos Android exige que os SDKs Java e Android sejam instalados. Eles fornecem o compilador, emulador e outras ferramentas necessárias para a construção, implantação e testes. Java, do Google Android SDK e ferramentas de Xamarin podem ser instalados e executados nas seguintes configurações:

- Mac OS X com o Xamarin Studio IDE
- Windows 7 ou 8 com a Xamarin Studio IDE
- Windows 7 ou 8 com o Visual Studio 2010, 2012 ou 2013

Xamarin fornece um instalador unificado que vai configurar o seu sistema com o pré-requisito de Java, Android e ferramentas Xamarin (incluindo um designer visual para layouts de tela).

Você pode construir e testar aplicativos em um dispositivo real, sem qualquer licença da Google, no entanto, para distribuir seu aplicativo por meio de uma loja (como o Google Play, Amazon ou Barnes & Noble) uma taxa de inscrição deverá ser paga ao operador. Então, Google Play vai publicar seu aplicativo

> instantaneamente, enquanto as outras lojas têm um processo de aprovação semelhante ao da Apple.”
> “Windows Phone
> Aplicativos do Windows Phone são construídos com o conjunto de ferramentas Microsoft Visual Studio 2010, 2012 ou 2013. Eles não usam Xamarin diretamente. No entanto, o código C# pode ser compartilhado entre o Windows Phone, iOS e Android usando as ferramentas da Xamarin.“

Como visto, esta IDE é muito interessante para desenvolvimento, e a mesma é uma excelente alternativa para este fim, dependendo das necessidades do utilizador. Se tratando de aplicativos híbridos, ou algum projeto deste tipo esta plataforma é bastante indicada.