UNIVERSIDADE DO OESTE PAULISTA
UNOESTE
PRESIDENTE PRUDENTE - SP

PRÓ-REITORIA DE PESQUISA E PÓS-GRADUAÇÃO
ESPECIALIZAÇÃO EM SEGURANÇA DA INFORMAÇÃO EM REDES DE COMPUTADORES E SISTEMAS

ANÁLISE E IMPLEMENTAÇÃO DE SEGURANÇA APLICADA À COMUNICAÇÃO DE MENSAGENS DE TEXTO EM TELEFONIA MÓVEL

Marcelo Giusti Tiziano

Presidente Prudente – SP
2006

UNIVERSIDADE DO OESTE PAULISTA
UNOESTE
PRESIDENTE PRUDENTE - SP

PRÓ-REITORIA DE PESQUISA E PÓS-GRADUAÇÃO
ESPECIALIZAÇÃO EM SEGURANÇA DA INFORMAÇÃO EM REDES DE COMPUTADORES E SISTEMAS

ANÁLISE E IMPLEMENTAÇÃO DE SEGURANÇA APLICADA À COMUNICAÇÃO DE MENSAGENS DE TEXTO EM TELEFONIA MÓVEL

Marcelo Giusti Tiziano

Trabalho monográfico apresentado no curso de pós-graduação, especialização em Segurança da Informação em Redes de Computadores e Sistemas, como requisito parcial para sua conclusão.
Área de concentração: Ciência da Computação.

Orientador: Prof. Dr. Adilson Eduardo Guelfi

Presidente Prudente – SP
2006

621.385 Tiziano, Marcelo
ANÁLISE E IMPLEMENTAÇÃO DE SEGURANÇA APLICADA À COMUNICAÇÃO DE MENSAGENS DE TEXTO EM TELEFONIA MÓVEL / Marcelo Giusti Tiziano - Presidente Prudente: Unoeste, 2006.
81p.:il.

Monografia (Pós Graduação)
Bibliografia

1- Telefone - Medidas de segurança
2- Telefonia celular.

MARCELO GIUSTI TIZIANO

Análise e Implementação de Segurança Aplicada à Comunicação de Mensagens de Texto em Telefonia Móvel

Dissertação apresentada a Pró-Reitoria de Pesquisa e Pós-Graduação, Universidade do Oeste Paulista, como parte dos requisitos obtenção do título de Especialista em.Segurança da Informação em Redes de Computadores e Sistemas.

PRESIDENTE PRUDENTE, __ DE _____2006.

BANCA EXAMINADORA

_____
Prof. Dr. Adilson Eduardo Guelfi.
UNOESTE.

_____
Prof. MSC. Emerson Silas Doria.
UNOESTE.

_____
Prof. MSC. Kleber Manrique Trevizani.
UNOESTE.

DEDICATÓRIA

Dedico este trabalho primeiramente a DEUS que sempre esteve ao meu lado, aos meus pais Mercedes Giusti Tiziano e Mauro José Tiziano, meus avós maternos Luigi Giusti e Rosa Giusti e meus avós paternos Virginio Tiziano Neto e Izabel Pavão Tiziano que nunca mediram esforços para que eu pudesse vencer.

A Camila Pirondi Krasucki, pelo seu amor, carinho, paciência, compreensão e apoio.

AGRADECIMENTOS

Agradeço a meu orientador, sendo um verdadeiro mestre, que não abriu uma porta e sim várias, que foi de essencial importância para realização deste trabalho. Aos meus colegas de trabalho, Ricardo Koji Ushizaki, Osmar Tonon, Gustavo Tadao Okida, e Rafael Shoji que sempre me incentivaram dando sugestões e criticas construtivas para que o trabalho se realizasse.

Agradeço a todos que contribuíram de alguma forma para a realização deste trabalho.

EPÍGRAFE

“Não cruze os braços para os obstáculos do mundo, pois o maior homem do mundo morreu de braços abertos”.

(Autor Desconhecido)

RESUMO

A crescente utilização da telefonia móvel entre as pessoas para a comunicação de informações, desde texto, imagem, som e até informações sensíveis, torna necessária a utilização de mecanismos que possibilitem que o tráfego de informações nessa rede esteja protegido do ponto de vista da segurança. Para prover a segurança na troca de dados, uma das opções é o uso de métodos criptográficos simétricos. Este trabalho tem como objetivo realizar uma análise sobre a utilização de métodos criptográficos simétricos na comunicação segura de mensagens de texto trocadas entre telefones móveis, mais especificamente os celulares. Os dados trocados são cifrados antes de serem enviados e decifrados quando recebidos. Foram realizados testes onde foi verificado o desempenho do programa desenvolvido, e a confidencialidade das informações cifradas.

ABSTRACT

The increasing use of mobile telephone systems among people to communication, including text, image, sound and secret information, makes necessary the use of mechanisms that can guarantee the security of information, and make sure that it will not be violated. To provide security in data exchanging, the option would be the use of symmetrical cryptographic methods. The goal of this work is carry out an analysis of the use of symmetrical cryptographic methods applied to provide a secure text message communication among mobile telephones. The exchanged data are encrypted before sending and decrypting when received. Some tests have been made showing the performance of the implemented program and the confidentiality of the encrypted information.

SUMÁRIO

Lista de Figuras e Tabelas

1 INTRODUÇÃO

Com o uso cada vez mais freqüente de celulares, torna-se necessária a utilização de mecanismos que forneçam a troca de informações de maneira segura.

O sistema telefônico tradicional não será capaz de atender a demanda do novo mercado, que são os usuários de telefonia móvel. Hoje já são feitas ligações de aviões, piscinas, do parque, sendo utilizado a telefonia móvel. Mas este mercado está crescendo constantemente onde no inicio da telefonia celular utilizava-se o celular apenas para efetuar chamadas telefônicas, e hoje, além da utilização do celular para chamadas telefônicas utiliza-se também para jogos, como agenda telefônica, despertador, tocador de MP3, enviar mensagens instantâneas (SMS), mensagens de voz, navegar na internet e até transações bancárias, o que irá exigir cada vez mais um sistema telefônico muito rápido, robusto e seguro. (Tanenbaum, 2003).

Este crescente mercado foi umas das principais motivações para a escolha deste tema, bem como, a segurança foi algo que me motivou o desenvolvimento do trabalho.

1.1 Motivações e Justificativas

A motivação deste trabalho deve-se ao fato do crescimento da utilização de dispositivos móveis no Brasil e no mundo. Segundo a *GSM Association* em novembro de 2006 2,1 bilhões o número de dispositivos móveis da tencologia GSM no mundo. Com a grande utilização de dispositivos moveis, surge também a necessidade que se utilizem mecanismos de segurança onde este trabalho focou o estudo de mecanismos de segurança em dispositivos móveis, mais especificamente a troca de mensagens curtas mais conhecidas como SMS (Short Message Service) entre dispositivos móveis, que é um dos diversos serviços oferecidos pela telefonia celular.

Com a maior utilização da telefonia móvel as aplicações que envolvem o tráfego, manipulação e acesso a informações confidenciais, como por exemplo, o *mobile banking*, motivou-me ainda mais a realização deste trabalho onde foi feita

uma analise da tecnologia de telefonia celular e uma implementação de um sistema de troca de mensagens SMS cifradas entre duas entidades adicionando assim um nível maior de segurança na telefonia móvel.

1.2 Objetivo do Trabalho

Esse projeto tem como objetivo realizar uma análise da tecnologia de telefonia celular e a implementação de um sistema com troca de mensagens SMS (*Short Message Service*) protegidas por algoritmos de criptografia simétrica e a utilização de algoritmos *hash* para validar a integridade da mensagem, visando proporcionar a segurança na comunicação entre duas entidades, para isto foi necessário o desenvolvimento de um protocolo para a troca de informações entre as duas entidades. Para o desenvolvimento foi necessário adotar a arquitetura Java, mais especificamente o J2ME.

1.3 Metodologia

A metodologia de desenvolvimento foi realizada em fases:

- Para o desenvolvimento inicial deste trabalho foi necessário o estudo e a utilização de ferramentas que forneça mecanismos para que fosse desenvolvido um sistema de troca de mensagens SMS cifradas, como a linguagem Java e algumas de suas arquiteturas como J2SE, J2ME, também foi necessário o estudo e utilização da biblioteca de criptografia Bouncy Castle, além de emuladores e o programa de transferência de aplicativos Java para celulares da motorola conhecido como MIDWay, para que fossem realizados testes na aplicação.
- Foi desenvolvido um programa de troca de mensagem SMS com criptografia simétrica e funções hash, como uma implementação desde estudo.
- Foram realizados testes de desempenho do programa desenvolvido em uma ambiente real, que no caso foi utilizado um celular motorola v185.

1.4 Estruturação do Texto

Este trabalho está estruturado da seguinte maneira:

- Criptografia: São apresentados os conceitos básicos de criptografia simétrica, criptografia assimétrica, algoritmos *hash*, assinatura digital e algoritmos simétricos utilizados na telefonia móvel.
- Telefonia Móvel: este capítulo será discutido uma introdução à telefonia móvel, sua evolução e tecnologias existentes.
- Especificação da Arquitetura para Comunicação Segura entre Dispositivos Móveis
- Descrição do Projeto: Neste capitulo é abordado assuntos sobre as tecnologias de telefonia móvel, como por exemplo, Sistemas Operacionais utilizados na telefonia móvel, linguagens de programação e bibliotecas de criptografia.
- Conclusão: Por fim neste capitulo foi descrito os resultados obtidos com as tecnologias utilizadas no projeto, desempenho da implementação realizada e futuras implementações.

2 CRIPTOGRAFIA

Um dos métodos que ajudam a garantir a segurança dos dados é a criptografia, que é a ciência que estuda manerias de escrever informações em forma de código secreto.

A criptografia já estava presente no sistema de escrita hieroglífica dos egípcios. Desde então vem sendo muito utilizada, principalmente para fins militares e diplomáticos. No âmbito da computação é importante para que se possa garantir a segurança em todo o ambiente computacional que necessite de sigilo em relação às informações que manipula. Pode ser usada para codificar dados e mensagens antes que esses sejam enviados por vias de comunicação, para que mesmo que sejam interceptados, dificilmente poderão ser decodificados, garantindo a privacidade.

Os conceitos discutidos no capítulo 2 foram sintetizados a partir da referência GUELFI (2005).

Sistemas criptográficos consistem em um grupo de tecnologias que auxiliam na implementação dos seguintes serviços de segurança:

- Confidencialidade;
- Integridade;
- Autenticação (usuário e parceiro);
- Autoria;
- Irretratabilidade ou não repudio;

O esquema de funcionamento da criptografia é bastante simples, e está ilustrado na Figura 1. A criptografia representa um processo de transformação, por meio de um algoritmo e uma chave criptográfica, tornando uma informação legível (*plaintext*), em uma informação ilegível, conhecida também como criptograma (*ciphertext*), ou seja, um criptograma. A chave criptográfica deve ter um caráter secreto, portanto, somente os indivíduos que conhecem tal chave podem ter a capacidade de decifrar o criptograma e recriar a mensagem original. O processo de decifração consiste exatamente no inverso da criptografia, ou seja, por meio de um algoritmo e chave criptográfica, consegue-se transformar o criptograma em uma mensagem legível.

A maior dificuldade em decifrar um criptograma para obter a mensagem original reside em descobrir a chave criptográfica secreta, e não em manter secretas as técnicas usadas (algoritmo de criptografia).

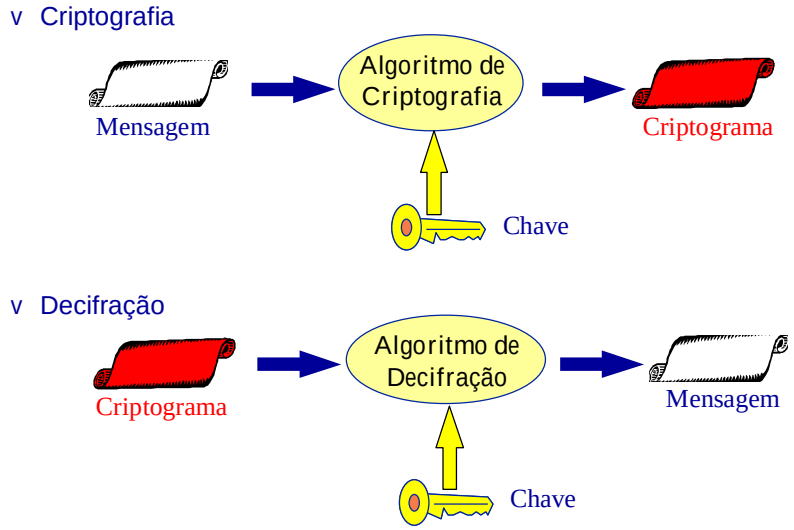


Figura 1. Esquema de Funcionamento da Criptografia. Fonte GUELFI (2005).

A mensagem corresponde à informação em um formato legível, sendo, como por exemplo, textos, programas fontes, programas executáveis, imagens, dados etc. E o criptograma corresponde à informação em um formato ilegível, sendo representada também por uma seqüência de bits nos sistemas de computação.

Abaixo será discutida a classificação da criptografia, como criptografia simétrica, criptografia assimétrica, funções *hash* e assinatura digital.

2.1 Classificação da Criptografia

A criptografia pode ser classificada segundo dois critérios:

- Número de chaves utilizadas.
- Forma de processamento.

Quanto ao critério do número de chaves utilizadas, a criptografia pode ser classificada em:

- Criptografia Simétrica ou Convencional: neste caso, a mesma chave criptográfica é usada para cifrar e decifrar a mensagem;
- Criptografia Assimétrica ou de Chave Pública: neste caso, duas chaves (um par) são usadas, sendo uma para cifrar e a respectiva chave parceira obrigatoriamente para decifrar a mensagem.

Conforme ilustrado na Figura 2, a criptografia simétrica utiliza uma única chave secreta que deve ser compartilhada e de conhecimento somente pelos parceiros de comunicação, ou seja, ninguém mais deve conhecer a chave criptográfica K, pois caso isso ocorra está informação poderá estar comprometida.

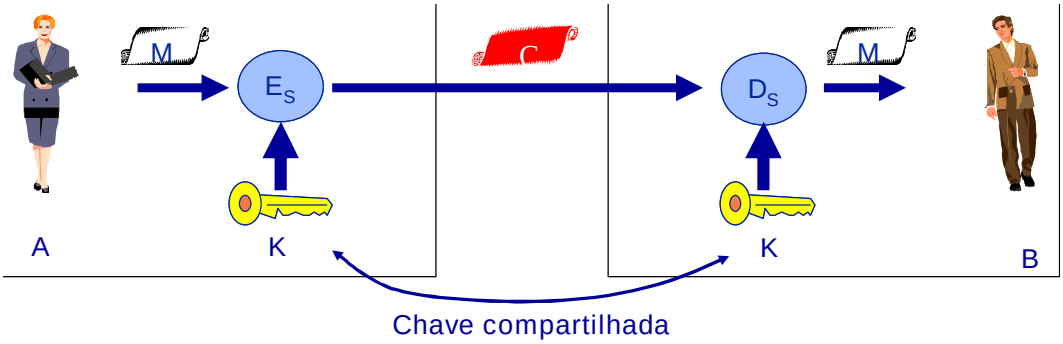


Figura 2. Esquema de Criptografia Simétrica. Fonte GUELFI (2005).

Conforme ilustrado na Figura 3 , a criptografia de chave pública ou assimétrica utiliza duas chaves, cifrando uma mensagem M com a chave1 e decifrando com a chave2, e vice-versa. Com relação às chaves assimétricas, normalmente adota-se as seguintes convenções:

- Uma das chaves é denominada de “Chave Privada”, a qual deve ter um caráter secreto, estar associada a somente uma dada entidade, e ser de conhecimento apenas pelo dono da chave;
- A outra chave é denominada de “Chave Pública”, a qual deve ter um caráter público, estar também associada a somente uma dada entidade por meio da respectiva chave privada, e ser de conhecimento de todas as outras entidades.

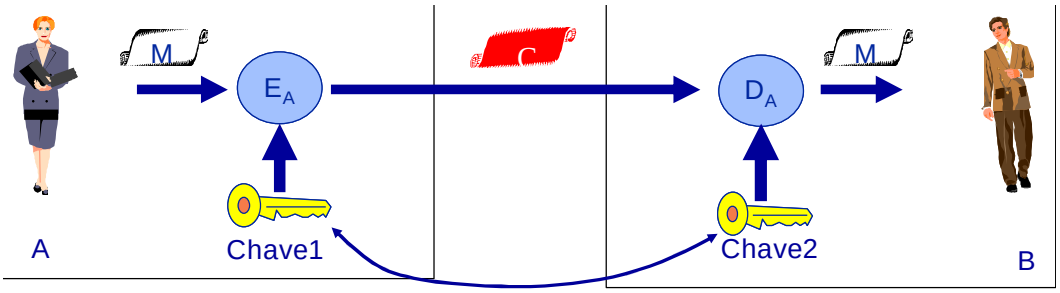


Figura 3 . Esquema de Criptografia Assimétrica. Fonte GUELFI (2005).

O critério da forma de processamento, a criptografia pode ser classificada em dois tipos:

- Por bloco: neste caso, um bloco de bits é processado por vez pelo algoritmo criptográfico, produzindo assim um bloco de criptograma correspondente;
- Stream (fluxo contínuo de informação): neste caso, o algoritmo criptográfico processa os elementos da mensagem de forma contínua (bit a bit, ou byte a byte).

Os algoritmos criptográficos que processam a mensagem por stream são mais eficientes em termos de desempenho (velocidade), e costumam ser mais indicados para a comunicação entre entidades.

2.1.1 Criptografia Simétrica ou Convencional

Mesmo os computadores terem mudado o campo da criptografia, seus princípios fundamentais permaneceram os mesmos; as mensagens eram codificadas com uma chave secreta ou compartilhadas e eram decodificadas com a mesma chave. Esse método, conhecido como criptografia tradicional, ou criptografia com chave simétrica, funciona bem em aplicações limitadas, como as militares, em que o emissor e o receptor podem se preparar antecipadamente para trocar a chave.

Conforme ilustrado na Figura 4, o modelo de criptografia convencional envolve o compartilhamento da chave K entre as entidades comunicantes (“A” e “B”). Assim, quando a entidade “A” deseja enviar uma mensagem M confidencial para “B”, “A” deve primeiro gerar a chave K e depois passá-la de forma segura para “B” (canal seguro). Somente depois que ambas as entidades comunicantes “A” e “B” possuam a mesma chave criptográfica K é que a mensagem M pode ser enviada com sigilo.

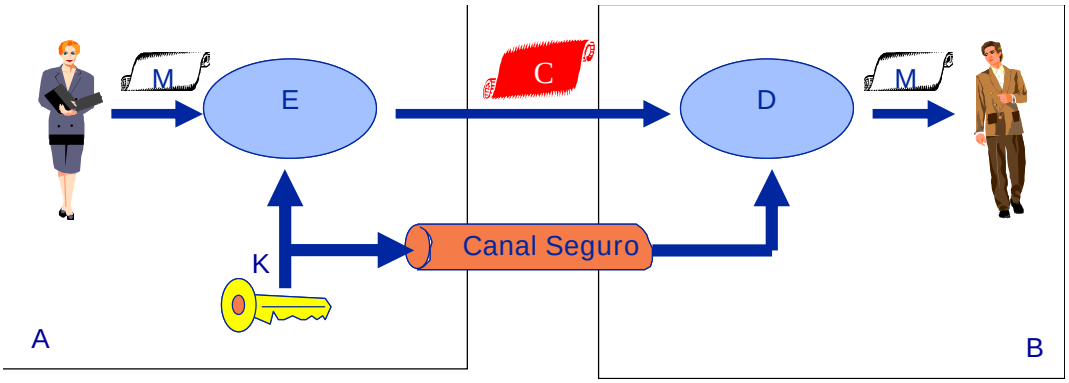


Figura 4. Modelo de Criptografia Convencional. Fonte GUELFI (2005).

O principal problema da criptografia convencional esta relacionado à troca de chave entre as entidades comunicantes, suponha que uma entidade “A” esteja geograficamente localizada na cidade de São Paulo e uma entidade “B” esteja em Presidente Prudente interior do estado de São Paulo e estas duas entidade “A” e “B” necessitam se comunicar utilizando mecanismos de sigilo, e o mecanismo escolhido entre as entidades é a criptografia convencional. A entidade “A” poderia enviar a chave criptográfica K para a entidade “B” por e-mail, telefone ou correio normal, entretanto, nenhuma destas sugestões é suficientemente segura. Uma maneira mais segura de se trocar à chave criptográfica K seria a entidade “A” entregar pessoalmente para a entidade “B”. No entanto, esta maneira não é tão atraente, pois geraria um custo financeiro ainda maior.

Deste modo, pode-se constatar que o principal problema da criptografia convencional é denominado de “Problema da Distribuição de Chaves”, pois não existe um meio seguro entre as entidades que se comunicam, de tal forma que a chave secreta K possa ser transmitida com proteção da origem para o destinatário.

Conforme ilustrado na Figura 5, um oponente poderia observar, ou mesmo interceptar, a chave K durante a transmissão de uma entidade “A” para uma entidade “B”, caso isso ocorra o oponente decifraria o criptograma C, tendo assim a mensagem M trocada entre a entidade “A” e “B”.

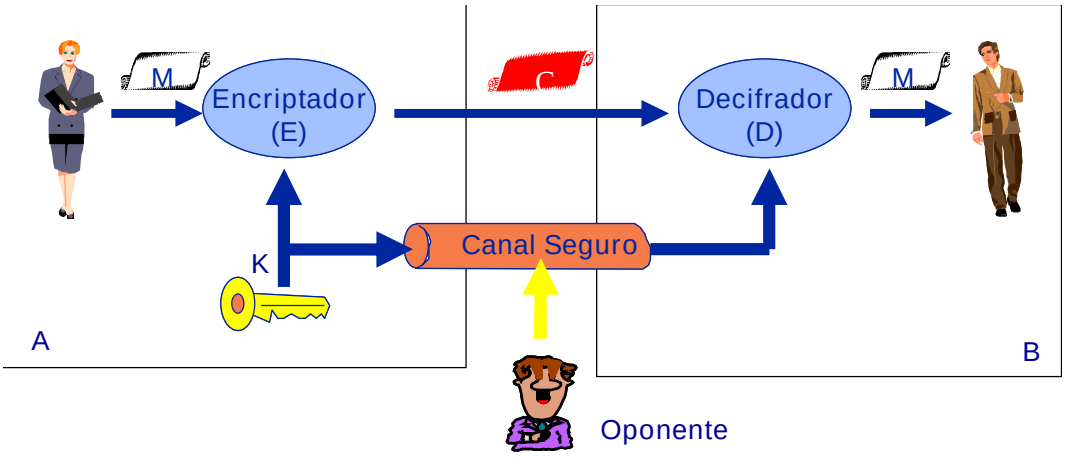


Figura 5. Problema da Distribuição de Chaves em Criptografia Convencional. Fonte GUELFI (2005).

2.1.1.1 Os algoritmos de chave simétrica

Abaixo será descrito algoritmos de criptografia simétrica utilizados na implementaçõ deste projeto e os utilizados na tecnologia GSM.

2.1.1.1.1 DES

O algoritmo DES (*Data Encryption Standard*) representou o primeiro padrão de criptografia para entidades do governo federal dos EUA. Tal algoritmo foi definido pelo NIST (*National Institute of Standards and Technologies*) e pode ser referenciado nos seguintes documentos:

- FIPS 46-3 – DES.
- FIPS 74 – Guia de implementação e uso do DES.
- FIPS 81 – DES modes of operation.

Atualmente, o algoritmo DES é considerado extremamente inseguro, não sendo recomendado principalmente para transações financeiras.

O algoritmo TripleDES (TDES) suporta dois tipos de implementação:

- Tamanho de chave criptográfica igual a 168 bits, correspondendo a três chaves DES de 56 bits;
- Tamanho de chave criptográfica igual a 112 bits, correspondendo a duas chaves DES de 56 bits.

O algoritmo TDES com chave criptográfica de 112 bits permite manter a compatibilidade com o DES simples, bastando para isto igualar as duas chaves DES simples (K1 = K2).

2.1.1.1.2 AES

Segundo GUELFI (2005), o AES (*Advanced Encryption Standard*), é o padrão atual de criptografia utilizado pelas entidades federais dos EUA, substituindo o padrão anterior que era o DES.

Para a escolha de um novo padrão foi feito um concurso para escolha de um novo algoritmo de criptografia simétrica que substitui-se o DES, onde o algoritmo de criptografia simétrica Rijandael foi o vencedor, que é um algoritmo de criptografia simétrica de bloco, onde este algoritmo processa blocos de dados de 128 bits, usando chaves com tamanho de 128, 192, 256 bits. Este padrão encontra se referenciado no documento FIPS 197 (NIST, 2001).

2.1.1.1.3 IDEA (International Data Encryption Algorithm)

Criado por X. Lai e J. Massey e um algoritmo com chave simétrica de 128 bits e processa blocos de dados de 64 bits. Foi projetado para ser eficiente em implementações por software. IDEA possui uma estrutura semelhante ao DES, possuindo um número fixo de iterações (*rounds*) de uma mesma função que utiliza subchaves distintas, e o mesmo algoritmo serve para criptografar e decriptografar alterando-se apenas a forma de geração das subchaves. (Tereda, 1999).

De acordo com SCHNEIER, o IDEA é significantemente mais seguro que o DES e como no caso do DES existe uma variante do IDEA que é o triplo – IDEA.

2.1.1.1.4 RC6

O RC6 foi um dos cinco candidatos finalista do padrão AES. Criado por R. Rivest, M.J.B. Robshaw, R. Sidney, e Y.L. Yin. O RC6 é uma variante do RC5, que foi alterado para concorrer ao padrão AES. Na criação do RC6 os autores quiseram torná-lo mais seguro contra criptoanálise e mais veloz que o RC5, e o RC6 possuem uma diferença com relação ao esquema de chaves, que é gerada mais derivações do que no RC5, estas derivações são chamadas de subchaves. Como o RC5, RC6 foi projetado para qualquer computador de 16 ou 32 ou 64 bits. Possui também uma descrição compacta e é adequado para implementações em software ou hardware. (*RSA Laboratorics*, 1998).

2.1.1.1.5 Algoritmo A5

Segundo SCHNEIER, A5 é um algoritmo de criptografia usada para cifrar e decifrar voz e dados na tecnologia GSM (*Global System Mobile*). É usado cifrar a ligação do telefone à estação base. Originalmente pensou-se que a criptografia da tecnologia GSM proibiria a exportação dos telefones a alguns países. Agora estão discutindo se A5 pôde prejudicar vendas de exportação.

A5 consiste em três LFSRs (*Linear Feedback Shift Register*), os comprimentos do registro são 19, 22, e 23. O resultado de saída é um XOR dos três registros LFSRs. Conforme mostra a Figura 6 é utilizado o algoritmo A5 para o trafego de voz e dados entre o dispositivo móvel e a estação base.

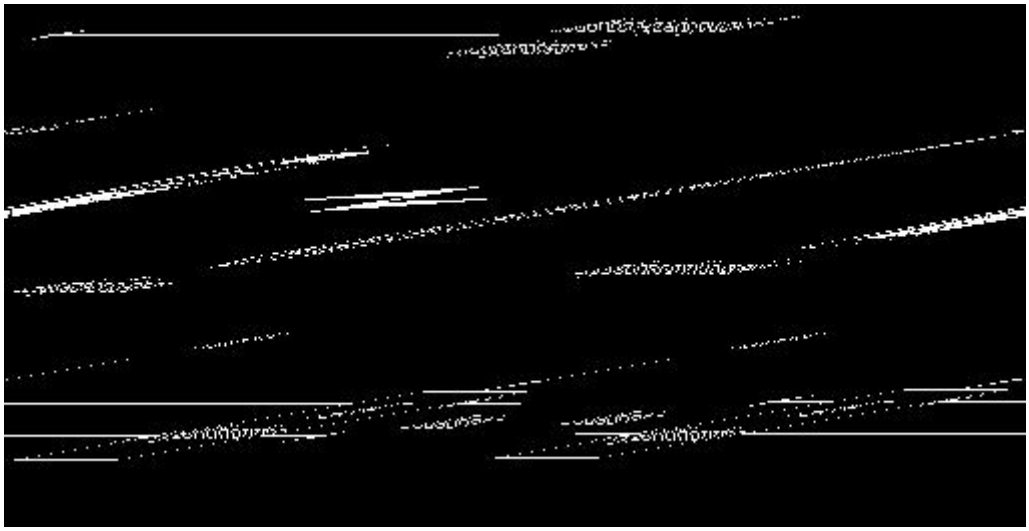


Figura 6, Mecanismo de Cifragem. Fonte MARGRAVE

2.1.1.1.6 Algoritmo A3

O algoritmo A3 é utilizado para autenticar o dispositivo móvel na rede de telefonia celular. A segurança desse algoritmo é baseada numa chave privada que fica armazenada dentro do dispositivo móvel que geralmente tem o comprimento de 128 bits (BARKAN; BIHAM; KELLER).

De acordo com Alencar (2004), o algoritmo A3 usa dois parâmetros de entrada, sendo o primeiro a chave privada do dispositivo móvel (Ki) de 128 bits e o segundo um número aleatório (RAND) de 128 bits, gerando como saída uma resposta sinalizada, mais conhecida como SRES, está resposta calculada pela tupla SRES = (Ki, RAND), onde SRES tem o tamanho de 32 bits. Após o SRES ser calculado o dispositivo móvel envia o SRES para a rede, que calcula sua integridade comparando o SRES gerado no dispositivo móvel com o SRES gerado pelo centro

de autenticação. A Figura 7 ilustra o processo de autenticação entre o dispositivo móvel e a rede de telefonia celular.

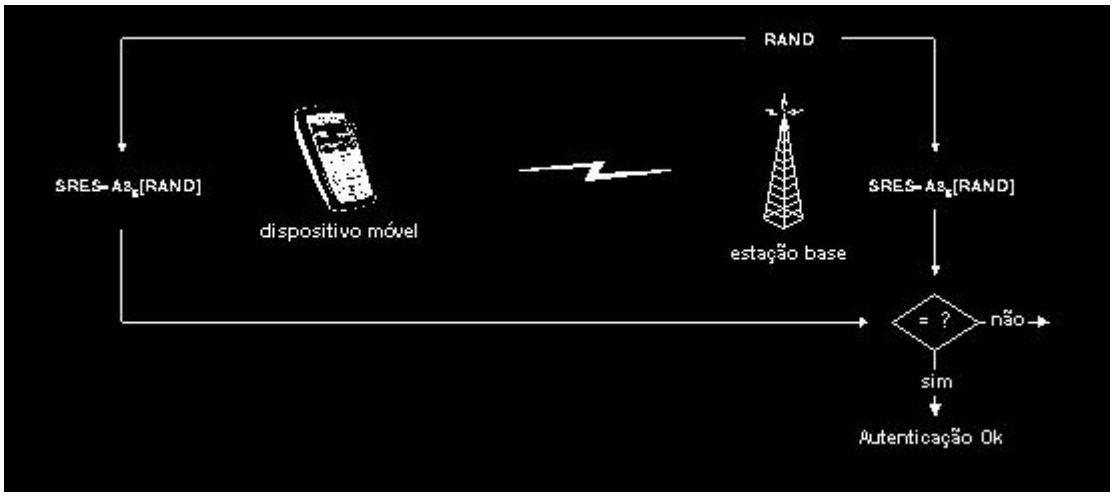


Figura 7, Mecanismo de Autenticação. Fonte MARGRAVE

2.1.1.1.7 Algoritmo A8

O Algoritmo A8 é utilizado para gerar chaves de sessão, como mostra a Figura 8, onde a chave de sessão é calculada pela seguinte tupla Kc = (Ki, RAND), sendo Ki a chave privada do dispositivo móvel de 128 bits e RAND um número aleatório de 128 bits, onde Kc tem 64 bits. Note que o algoritmo A8 é calculado de maneira similar ao A3, sendo que no algoritmo A8 é gerada uma chave (Kc) de 64 bits e no algoritmo A3 é gerado um SRES de 32 bits. (BARKAN; BIHAM; KELLER).

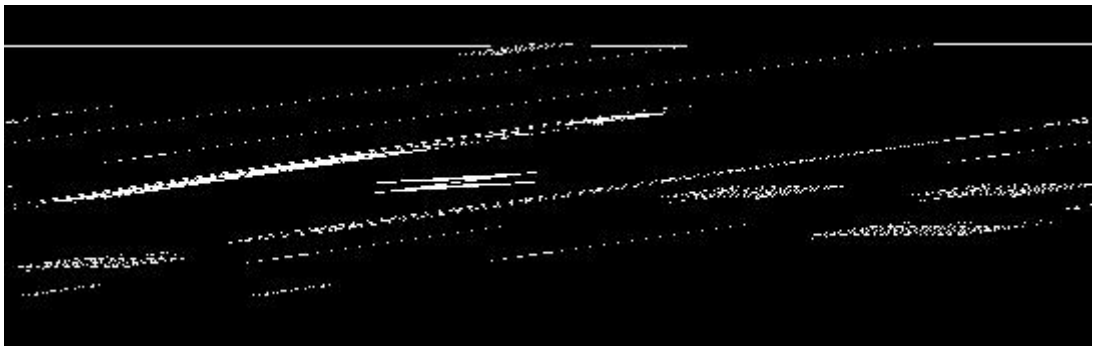


Figura 8, Mecanismo de geração de chave. Fonte MARGRAVE

2.1.2 Criptografia Assimétrica ou de Chave Pública

Segundo GUELFI (2005), criptografia assimétrica conhecida também por criptografia de chave publica utiliza duas chaves criptográficas por cada entidade,

sendo uma para cifrar e outra para decifrar como ilustrado na Figura 9. Neste caso, a regra geral do modelo de criptografia de chave pública estabelece que, se uma mensagem M for cifrada com uma chave, a única chave que permite restaurar a mensagem original é a chave parceira, ou seja, caso tenha cifrado uma mensagem M com a chave privada de uma entidade "A", será necessário utilizar a chave publica da entidade "A" para restaurar a mensagem M, ou vice-versa.

Assim, conforme mostra a Figura 9, cada entidade possui um par de chaves sendo uma a chave pública e a outra a chave privada. A chave pública pode ser de conhecimento de todos, enquanto que a chave privada deve ser mantida de forma secreta pela entidade. (GUELFI, 2005).

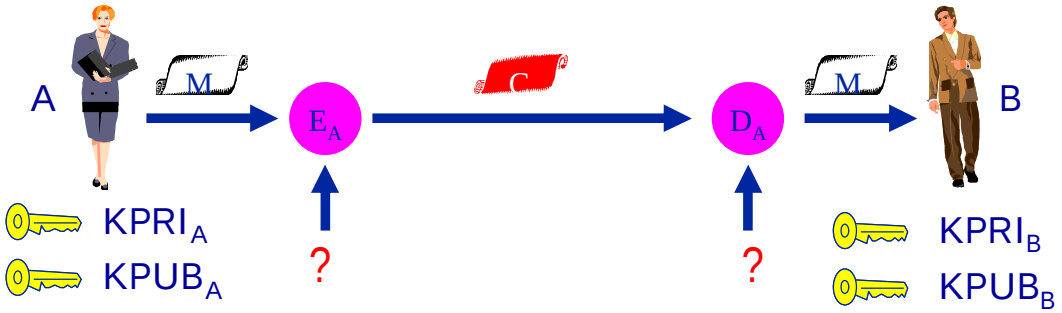


Figura 9. Modelo de Criptografia de Chave Pública. Fonte GUELFI (2005).

2.1.2.1 Uso da Criptografia de Chave Pública

A criptografia de chave pública pode ser utilizada para três casos:

- Confidencialidade
- Autenticação
- Confidencialidade e Autenticação

2.1.2.2 Confidencialidade

De acordo com GUELFI (2005), ao usar a criptografia de chave pública para obter confidencialidade, uma dada entidade tem como objetivo manter o sigilo da informação em trânsito. Conforme ilustrado na Figura 10, usando criptografia de chave pública, uma mensagem M pode ser transmitida de forma confidencial por meio do seguinte esquema de funcionamento:

- O remetente A criptografa a mensagem M utilizando a chave pública do destinatário B;

- O destinatário B decifra o criptograma C utilizando sua chave privada.

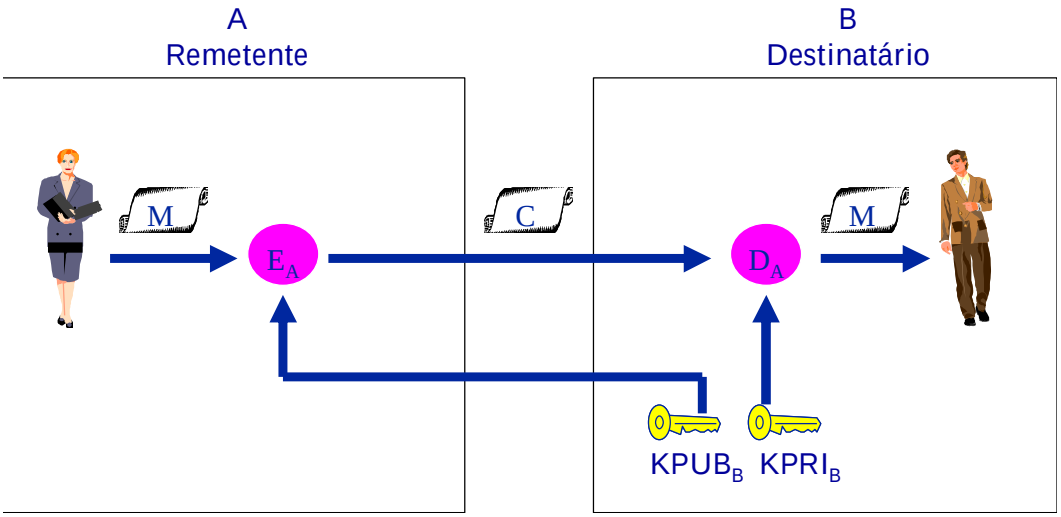


Figura 10. Uso da Criptografia de Chave Pública para Confidencialidade. Fonte GUELFI (2005).

Conforme ilustrado na Figura 10 acima se pode observar que para obter confidencialidade em uma comunicação usando criptografia de chave pública, basta o remetente cifrar a mensagem M com a chave pública do destinatário. Assim, como somente o destinatário conhece sua chave privada, apenas o destinatário conseguirá restaurar a Mensagem M utilizando sua chave privada, conseguindo assim obter o nível de sigilo desejado. (GUELFI, 2005).

2.1.2.3 Autenticação

Segundo GUELFI (2005), ao usar a criptografia de chave pública para obter autenticação, uma dada entidade tem como objetivo saber ou identificar o autor de uma determinada informação recebida.

Conforme ilustrado na Figura 11, usando criptografia de chave pública, uma mensagem M pode ser transmitida com autenticação por meio do seguinte esquema de funcionamento:

- O remetente A criptografa a mensagem M com sua chave privada;
- A decifração do criptograma C pode ser realizada por qualquer outra entidade utilizando a chave pública do remetente A.

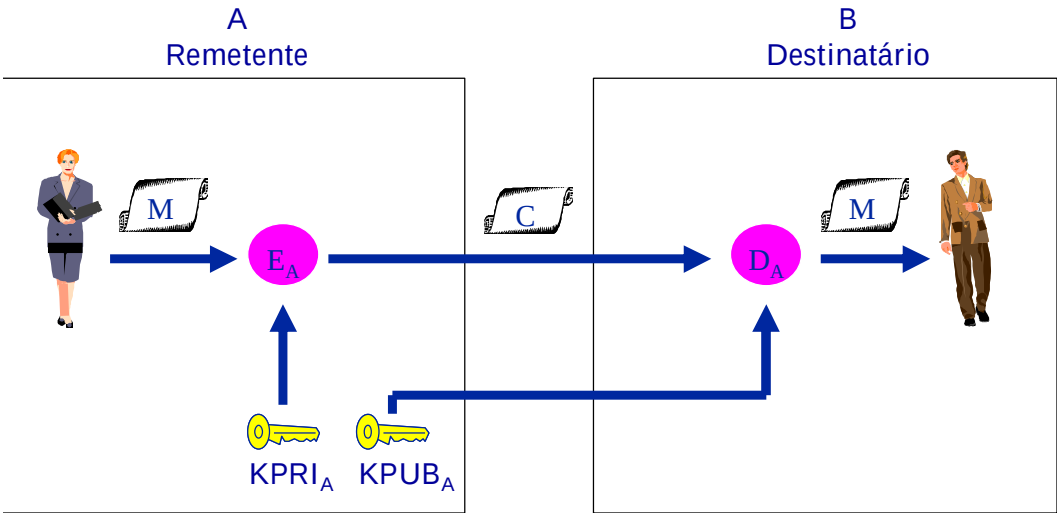


Figura 11. Uso da Criptografia de Chave Pública para Autenticação. Fonte GUELFI (2005).

De acordo com GUELFI (2005) “Com relação ao nível de autenticação obtido no esquema da Figura 11, a autoria da mensagem pode ser admitida se considerarmos que somente o remetente possui sua chave privada de forma secreta, ou seja, a chave privada do remetente está associada somente a ele, e não é compartilhada com qualquer outra entidade. Além disso, o criptograma C não pode ter sido gerado por outra chave (entidade), pois somente o remetente possui a chave privada associada à chave pública de decifração”.

O esquema de autenticação da Figura 11 não provê confidencialidade, uma vez que qualquer entidade pode decifrar a mensagem M original com a chave pública do remetente. Este esquema pode ser utilizado para a autenticação de uma entidade, onde este esquema iria garantir a autenticidade da entidade, como por exemplo, um sistema de *internet baking* que é enviado a chave publica para o usuário, sendo assim o usuário poderá garantir que está no sistema de *internet banking* desejado.

Em resumo, para obter autenticação de uma mensagem transmitida usando criptografia de chave pública, basta o remetente cifrar a mensagem M com sua própria chave privada.

2.1.2.4 Confidencialidade e Autenticação

Segundo GUELFI (2005), ao usar a criptografia de chave pública para obter confidencialidade e autenticação, uma dada entidade tem como objetivo manter

secreta uma informação em trânsito, e também permitir ao destinatário saber ou identificar a autoria da informação recebida. Conforme ilustrado na Figura 12, usando criptografia de chave pública, uma mensagem M pode ser transmitida com confidencialidade e autenticação por meio do seguinte esquema de funcionamento:

- O remetente A criptografa a mensagem M inicialmente com sua chave privada e, em seguida, com a chave pública do destinatário B;
- O destinatário B decifra o criptograma C2 primeiro com sua chave privada e, em seguida, o criptograma C1 com a chave pública do remetente A;

Este esquema representa uma junção dos dois anteriores.

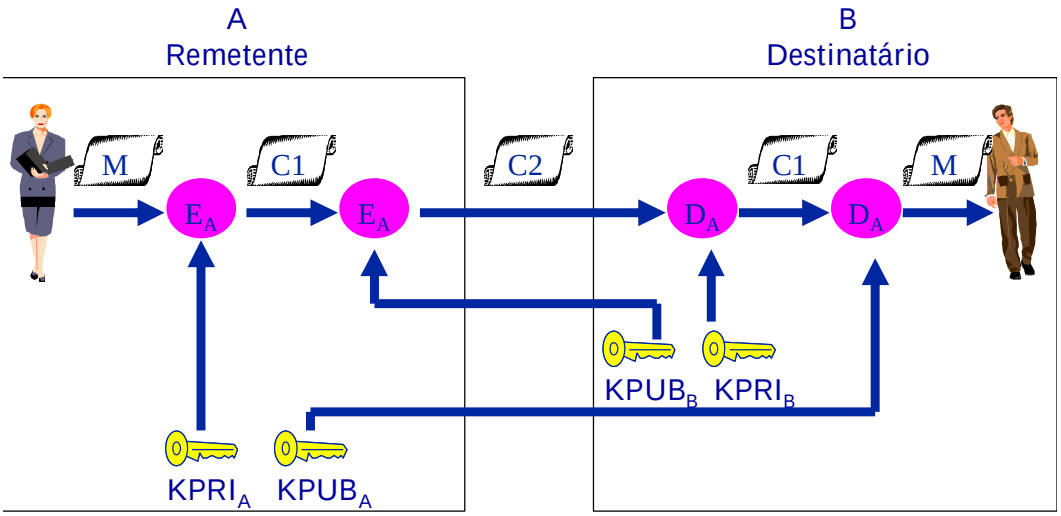


Figura 12. Uso da Criptografia de Chave Pública para Confidencialidade e Autenticação. Fonte GUELFI (2005).

A confidencialidade é obtida na primeira decifração, pois somente o destinatário B pode decifrar o criptograma C2 com sua chave privada, obtendo como resultado o criptograma intermediário C1. Por sua vez, a autenticação é obtida na segunda decifração (C1), pois o destinatário B pode admitir a autoria da mensagem por parte entidade A, visto que somente o remetente A poderia ter criptografado a mensagem M com sua chave privada. (GUELFI, 2005).

2.1.2.5 Criptografia de Chave Pública - Características

De acordo com GUELFI (2005), com relação à flexibilidade, a criptografia de chave pública torna-se bastante interessante devido à existência de duas chaves criptográficas, sendo que uma delas pode se tornar pública, enquanto que a outra se

mantém secreta, de posse de sua entidade proprietária, sem necessitar de qualquer aspecto de compartilhamento. Tal característica é fundamental para alguns serviços, como por exemplo, a autenticação.

Quanto ao custo computacional, a criptografia de chave pública é razoavelmente mais lenta do que a criptografia convencional. Em software, a criptografia de chave pública é cerca de 100 vezes mais lenta do que a criptografia convencional. Esta diferença aumenta ainda mais quando consideramos criptografia de chave pública em hardware, sendo cerca de 1000 vezes mais lenta do que a criptografia convencional nas mesmas condições. (GUELFI, 2005).

A criptografia de chave pública é radicalmente distinta com relação à criptografia simétrica ou convencional. Entretanto, apesar de todas as vantagens e serviços de segurança que podem ser implementados com a criptografia de chave pública, não é viável considerar a criptografia convencional obsoleta por causa de seu custo computacional mais eficiente.

Quanto ao “Problema da Distribuição de Chaves Criptográficas” apresentado pela criptografia convencional, a criptografia de chave pública não sofre desta deficiência em seu esquema.

Portanto, ao utilizar os dois esquemas de criptografia até então conhecidos (simétrica e assimétrica), pode-se concluir que a criptografia de chave pública consegue resolver o “Problema da Distribuição de Chaves Criptográficas Convencionais”, pois a confidencialidade (canal seguro) na distribuição da chave secreta K de criptografia simétrica poderia ser obtida por meio da criptografia de K usando a chave assimétrica pública do parceiro de comunicação (destinatário). Este caso será visto com maiores detalhes na seção que trata dos algoritmos de troca de chaves.

Apesar de todas as características até então apontadas, os esquemas de criptografia simétrica e assimétrica ainda não conseguem resolver o problema da autenticação de entidades. A autenticação de entidades está diretamente vinculada à autenticação da chave pública, necessitando para isto criar uma ligação legal que une ou atribui à chave pública a sua respectiva entidade.

2.1.2.6 RSA

O algoritmo de criptografia assimétrica RSA foi inventado por Ron Rivest, Adi Shamir e Len Adleman em 1978 onde na época eram professores do Instituto MIT (*Massachusetts Institute of Technology*), fundadores da atual empresa RSA Data Security, Inc., onde o RSA é a implementação de algoritmo considerada mais bem sucedida de sistemas de criptografia assimétrica. É considerado também um dos mais seguros, já que foram realizadas várias tentativas mal sucedidas que demonstrassem uma falha em seu algoritmo desde sua criação em 1978. Foi também o primeiro algoritmo a possibilitar a assinatura digital, e uma das grandes inovações em criptografia de chave pública. (*RSA Laboratory*, 2006).

Em traços gerais, são gerados dois pares de números – as chaves – de tal forma que uma mensagem cifrada com o primeiro par possa ser apenas decifrada com o segundo par; mais, o segundo número não pode ser derivado do primeiro. Esta propriedade assegura que o primeiro número possa ser divulgado a alguém que pretenda enviar uma mensagem cifrada ao detentor do segundo número, já que apenas essa pessoa pode decifrada a mensagem. O primeiro par é designado como chave pública, e o segundo como chave secreta. (*RSA Laboratory*, 2006).

O algoritmo RSA baseia-se no fato de que, se bem que encontrar um número primo aleatório de grandes dimensões é computacionalmente fácil, conseguir fatorar o produto de tais dois números é considerado computacionalmente difícil. De fato, este algoritmo mostra-se computacionalmente inquebrável com números de tais dimensões, e a sua força é geralmente quantificada com o número de bits utilizados para descrever tais números. Implementações atuais superam os 1024 bits e mesmo os 2048 bits que são padrões adotados pelo ITI (Instituto Nacional de Tecnologia Da Informação). (ITI, 2006)

2.1.2.7 IBE

STANFORD, em 1984 A. Samir criou um novo esquema de criptografia de chaves publicas onde é pode ser resumido em quatro etapas:

- Gerar a chave pública com os parâmetros, essa chave publica pode ser uma combinação de dados da entidade como o seu e-mail, ou o e-mail mais uma data.
- Gerar a chave privada correspondente extraindo as informações da chave publica.
- Cifrar as mensagens com a chave pública.
- Decifrar a mensagem com a chave privada correspondente.

Pode-se observar neste momento que a maior diferença é a chave publica onde ela pode ser construída por parâmetros como um e-mail, e é essa a maior vantagem do esquema IBE, pois não é necessário gerar uma chave publica como uma seqüência de bits e sim algum dado conhecido, com isso simplificando o gerenciamento de chaves públicas. Pode se utilizar o esquema de IBE inclusive para dar validade às chaves, por exemplo, gerar a chave publica com o e-mail mais uma data de expiração da mesma. O esquema IBE utiliza curvas elípticas para garantir a segurança, onde G1 X G1 à G2 onde G1, G2 são números grandes e considerados fortes. A utilização de curvas elípticas deve-se ao caso de que há um mapeamento bi linear computável, em um determinado ponto da curva há apenas outro correspondente.

Vamos denotar que o par de chaves publica/privada seja (R, s), onde R ∈ G1, s ∈ Fq e P é um ponto fixo em G1 de conhecimento público, também são utilizados *Hashing* sendo H1: {0,1} * à G1; H2: {0,1} * à Fq; H3: G2 à {0,1} *. Funções *Hash* serão vista no próximo tópico.

Na IBE também é utilizado uma Autoridade Certificadora de confiança, com sua respectiva chave publica/privada (RTA, sTA).

Agora vamos ver um exemplo de cifrar um texto utilizando IBE. Iremos utilizar a Alice e Bob como entidades que querem trocar uma mensagem.

Para Alice enviar uma mensagem para Bob ela tem algumas informações publicas e necessita fazer alguns cálculos, como descritos abaixo:

- Informações Publicas:

  - P, que é um ponto fixo em G1;
  - RTA, que é a chave publica da CA;
  - QBob, que é a chave pública de Bob, que pode ser Bob@unoeste.br.
- Alice calcularia:
  - U = rP, sendo r um elemento aleatório de Fq;
  - V = m ⊕ H3(t(RTA, QBob)).
  - *ciphertext* seria então o (U, V).
- Para decifrar a mensagem Bob faria o seguinte calculo:
  - M = V ⊕ H3(t(U, sBob)).

2.1.3 Funções Hash

Uma função *hash* é uma equação matemática que utiliza texto (tal como uma mensagem de e-mail) para criar um código chamado *message digest* (resumo de mensagem). Alguns exemplos conhecidos de funções *hash*: MD4 (MD significa *message digest*), MD4, MD5, SHA e SHA1. Uma função *hash* utilizada para autenticação digital deve ter certas propriedades que a tornem segura para uso criptográfico. Especificamente, deve ser impraticável encontrar texto conhecido seu *hash,* ou seja, mesmo que você conheça o *message digest*, não conseguirá decifrar a mensagem. Duas mensagens distintas que dão um *hash* ao mesmo valor. A capacidade de descobrir uma mensagem que dê um *hash* a um dado valor possibilita a um agressor substituir uma mensagem falsa por uma mensagem real que foi assinada. Permite ainda que alguém rejeite de forma desleal uma mensagem, alegando que, na realidade, ele ou ela assinou uma mensagem diferente, dando um *hash* ao mesmo valor e violando assim a propriedade de não-repúdio ou não rejeição das assinaturas digitais.

A capacidade de descobrir duas mensagens distintas que dêem um *hash* ao mesmo valor possibilita um tipo de ataque no qual alguém é induzido a assinar uma mensagem que dá um *hash* ao mesmo valor como sendo outra mensagem com um conteúdo totalmente diferente (OIKAWA, 2003).

2.2 Assinatura Digital

Ter duas chaves separadas proporciona outro benefício a assinatura digital. Imagine como seria usar o sistema de forma invertida. Em vez de Bob cifrar a mensagem com a chave pública de Alice, ele utiliza sua própria chave privada. Mas espere, você deve estar pensando agora todo mundo pode ler a mensagem, ela deixou de ser secreta. Isso é verdade, mas também é verdade que apenas Bob poderia ter escrito a mensagem. Ele é a única pessoa capaz de criar mensagens que possam ser lidas com sua chave pública, pressupondo-se, obviamente, que Bob não tenha compartilhado sua chave privada com ninguém mais e que a chave seja realmente secreta (BERNSTEIN, 1997).

Por exemplo, Bob quer enviar uma mensagem para todos os seus contatos. Bob não se importa em quem irá ler a mensagem, Bob apenas quer garantir aos seus contatos que a mensagem é realmente sua.

A seqüência a seguir atinge esse objetivo.

1. Bob escreve a mensagem e a cifra utilizando sua chave privada.
2. Bob envia a mensagem a seus contatos através da Internet.
3. Os contatos recebem a mensagem e a decifra utilizando a chave pública de Bob.

O fato de a chave pública de Bob ter decifrado a mensagem garante aos contatos que a mensagem realmente é dele. Qualquer mensagem decifrada com a chave pública de Bob só poderia ter sido criada com a chave privada. Isso é muito importante. Na criptografia com chave pública, cada par de chaves é único. Só existe apenas uma chave pública para cada chave privada e vice-versa. Se isso não fosse verdade, a assinatura digital não seria possível; um impostor poderia utilizar outra chave privada para criar uma mensagem que pudesse ser lida pela chave pública fornecida.

A assinatura digital implementa os objetivos de segurança da integridade e do não-repúdio. Como foi visto, a assinatura digital assegura aos contatos que a mensagem não foi alterada (integridade) e que ela veio de Bob (autenticidade). Além disso, Bob não pode negar que tenha enviado a mensagem (não-repúdio), pois é o único com acesso a sua chave privada.

Por exemplo, Bob quer enviar uma mensagem para Alice e quer que apenas ela leia a mensagem, com isso Bob irá precisar fazer uma assinatura digital para garantir que a mensagem é dele e que só Alice pode ler a mensagem. Por exemplo:

1. Bob escreve a mensagem e a cifra utilizando sua chave privada (ele assina a mensagem).
2. Em seguida, ele decifra a mensagem com a chave pública de Alice (tornando-a privada).
3. Bob envia a mensagem duplamente cifrada para Alice através da Internet.
4. Alice Recebe a mensagem.
5. Ela decifra a mensagem duas vezes. Primeiro, ela utiliza sua chave privada e, depois, a chave pública de Bob. Observe que ela está invertendo os passos que Bob executou para criar a mensagem.
6. Alice agora pode ler a mensagem e tem certeza de que ela é secreta e veio de Bob. Ela também tem certeza de que a mensagem não foi modificada; para alterá-la, o invasor teria de acessar a chave privada de Bob.

3 Telefonia Móvel

Basicamente há duas variedades básicas de telefones sem fio: os telefones sem fio propriamente ditos, que utilizam uma linha telefônica comum e os telefones móveis que são os celulares. Os telefones sem fio de linha comum não serão analisados no escopo deste trabalho, pois não estão relacionados com a transmissão de redes de telefonia celular.

Existem três gerações de telefones celulares, sendo:

1. ª Geração: Voz analógica.
2. ª Geração: Voz digital.
3. ª Geração: Voz digital e dados (Internet, correio eletrônico, etc.).

3.1 Telefones Móveis de Primeira Geração:

Por volta de 1960, o Sistema IMTS (*Improved Móbile Telefone System* – sistema de telefonia móvel aperfeiçoado) foi utilizado. Onde ele utilizava duas freqüências sendo uma de transmissão e outra para recepção.

Em 1982, surgiu o Sistema AMPS (*Advanced Móbile Phone System*) elaborado pelo Bell Labs, que trabalhava de forma diferente ao IMTS e utilizava células, como se fosse uma colméia, onde cada célula atingia uma área de 10 km com várias camadas de freqüência. A Figura 13 ilustra o sistema AMPS em forma de células.

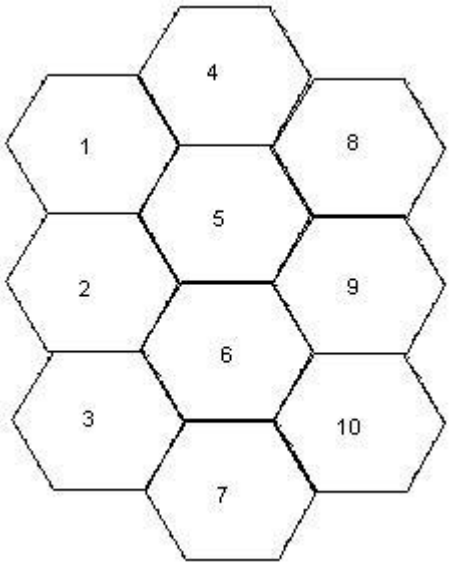


Figura 13 , Sistema AMPS. Fonte ALENCAR (2004).

No centro de cada célula há uma estação base que recebe as informações. A estação base nada mais é do que um computador e um receptor/transmissor ligados a uma antena. Além disso, as estações são conectadas a uma estação de comutação de telefonia móvel MTSO (*Móbile Telephone Switching*), para conectar todas as células a uma rede central, além de conectar as células o MTSO também as monitoram.

3.2 Telefones Móveis de Segunda Geração:

Nos telefones móveis de 2ª geração, existem tecnologias diferentes entre os celulares como o CDMA e o GSM que são os dois tipos de tecnologias utilizadas no Brasil.

O sistema global para comunicações móveis GSM (*Global System for Móbile Communications*) é empregado a mutiplexação por divisão de freqüência, transmitindo e recebendo numa freqüência de 55MHz.

O sistema CDMA de comutação de telefonia móvel (*Code Division Multiple Access*) funciona de modo completamente diferente do GSM, pois ao em vez de dividir a faixa de freqüência permitida em algumas centenas de canais estreitos, os CDMA permitem que cada estação transmita sobre todo o espectro de freqüência durante todo o tempo.

Os padrões CDMA e GSM são discutidos mais profundamente nas seções 3.5 e 3.6 respectivamente.

3.3 Telefones Móveis de Terceira Geração:

Está geração é utilizada no Brasil há pouco tempo e há uma expectativa muito grande por parte dos especialistas, contribuindo com o trafego de dados em conjunto com o trafego de voz.

Em 1992 a Organização ITU (*International Móbile Telecommunications*) onde o mesmo apresentou um projeto a ser alcançado. Pode - se resumir os serviços básicos da proposta em:

- Transmissão de voz de alta qualidade;

- Serviço de mensagens (substituindo correio eletrônico, fax, SMS, bate-papo etc.).
- Multimídia (reprodução de música, exibição de vídeos, filmes televisão, etc.).
- Acesso à internet (navegação web, incluindo páginas com áudio e vídeo).

O W-CDMA (*Wideband* CDMA), que é uma nova proposta para telefonia GSM, funciona em uma largura de banda de 5MHz e interage com redes GSM.

Foi proposto CDMA2000, que também trabalha com banda de 5 MHz, mas não interage com redes GSM.

Abaixo será descrito os três padrões utilizados no mercado atual que é o TDMA, o CDMA e o padrão GSM, como também o serviço de mensagens curtas, mais conhecidas como SMS (*Short Message Service*).

3.4 Padrão TDMA

Este padrão corresponde ao antigo sistema AMPS, como citado na seção 3.1, que como o sistema AMPS estava saturado por volta da década de oitenta, foi proposto um novo padrão do sistema AMPS chamado de IS-54 (*Ínterim Standard*), que visava as seguintes evoluções:

- Aumento da capacidade de 10 vezes em relação ao sistema analógico;
- Compatibilidade com o sistema analógico;
- Evolução suave entre os sistemas analógico e digital;

Mas apenas o primeiro item proposto inicialmente não foi concretizado, pois sua capacidade aumentou três vezes, e não dez vezes como na proposta inicial.

O padrão TDMA utiliza mecanismos para fornecer certo nível de segurança, fornecendo mecanismos de integridade e autenticação de dispositivos móveis na rede de telefonia celular.

3.4.1 Integridade e autenticação

O padrão TDMA utiliza alguns mecanismos para fornecer integridade da transmissão e autenticação entre dispositivos móveis e a rede de telefonia celular TDMA.

3.4.1.1 Integridade

O padrão IS-54/136 utiliza alguns mecanismos para garantir a integridade dos dados durante a transmissão que são:

- Número de identificação do móvel: Número de identificação do dispositivo móvel;
- Numero de série eletrônico (*Electronic Serial number*): Número serial que identifica uma dispositivo móvel, o mesmo pode ser trocado o que normalmente não é disponível para usuários;
- Classe da estação: Número que identifica a potência, banda e classe de transmissão da dispositivo móvel;
- Memória de localização de área: Utilizado para identificar mudanças de localização na dispositivo móvel;
- Primeiro canal de paging: Utilizado para identificar o canal de paging;
- Identificação de sistema local: Utilizado para identificar o sistema local do dispositivo móvel;
- Opção de controle: Utilizado para habilitar e desabilitar a opção de controle local;
- Seleção de sistema preferencial: Utilizado para indicar o sistema preferencial entre bandas;

3.4.1.2 Autenticação

O padrão IS-54/136 utiliza alguns mecanismos para fornecer autenticação como descrito abaixo:

- Número de identificação pessoal: Cada dispositivo móvel contém uma identificação única, administrada pelas operadoras do usuário, sendo o mesmo verificado a cada conexão com o sistema.
- Mensagem RANDS (RANDS, *Random Challenge Global Action Message*): um valor que é armazenado no dispositivo móvel que é adicionado a uma seqüência de overhead na mensagem;
- Parâmetro de Histórico de Chamada (COUNTS): É atualizada pelo dispositivo móvel toda vez que recebe uma Ordem de Atualização de parâmetros;

3.5 Padrão CDMA

Com a crescente utilização de dispositivos móveis a Associação *Cellular Telecommunications Industry Association* (CTIA), dos Estados Unidos, publicou um documento chamado de *User's Performance Requirements* (UPR), especificando requisitos básicos da tecnologia digital, que são:

- Aumento da capacidade de 10 vezes em relação ao sistema analógico;
- Longa vida e um crescimento adequado da tecnologia de segunda geração;
- Capacidade de introdução de novas funcionalidades;
- Melhoria na qualidade;
- Privacidade;
- Facilidade de transição e compatibilidade com o sistema analógico existente;
- Disponibilidade, a baixos custos, de rádios e células que operam nos dois sistemas;
- Arquitetura de rede aberta;

3.6 Padrão GSM

No inicio da década de 1980 a organização *Conference of European Postal and Telecommunications* (CEPT) criou o *Groupe Spéciale Móbile* com o objetivo de desenvolver um sistema pan-europeu, mas após alguns anos verificou que seu alcance seria global.

Em 1991 foi proposta uma padronização do sistema GSM, que sua evolução seria dividida em fases para que fossem adicionados novos serviço gradativamente. Segundo Alencar (2004), apud Yacoub o Sistema GSM tem um conjunto de objetivos ambiciosos que são:

- Roaming internacional;
- Arquitetura aberta;
- Alto grau de flexibilidade;
- Fácil instalação;

- Operação integrada com RDSI (Rede digital de serviços integrados), CSPDN (Rede de dados pública com comutação de circuitos), PSPDN (Rede de dados com comutação de pacotes) e PSTN (Rede telefonia pública comutada);
- Oferecimento de sinais de alta qualidade e garantia de integridade do enlace;
- Eficiência espectral de baixo custo;
- Infra-estrutura de baixo custo;
- Terminais pequenos, de baixo custo;
- Características de segurança;

O padrão GSM contém alguns mecanismos que fornecem um nível de segurança, como criptografia, autenticação de dispositivos móveis, um centro de autenticação, onde o dispositivo necessita se autenticar para integrar-se a rede de telefonia móvel, e o SIM card. que é uma característica encontrada apenas na tecnologia GSM.

3.6.1 SIM – módulo de identificação do cliente

Segundo Alencar (2004), O Módulo de Identificação do Cliente (*Subscribe Identity Modulo* - SIM) fornece uma identificação do dispositivo móvel, que sem o SIM o dispositivo fica inoperante (exceto para chamadas de emergência dependendo do modelo do dispositivo móvel). O SIM é um cartão inteligente que contém um processador e memória, onde ficam armazenados dados do usuário. A Figura 14 ilustra um cartão inteligente SIM, também conhecido como SIM Card.

Figura 14, SIM Card.

Para se acessar as informações contidas no cartão SIM existe um recurso de segurança para proteger as informações contidas do SIM, que é conhecida como PIN (Número de Identificação Pessoal), o número do PIN tem o comprimento mínimo de 4 dígitos e máximo de 8 dígitos, caso o usuário forneça errado o número do PIN e três vezes seguidas o cartão fica bloqueado, e para desbloquear o mesmo é necessário informar o PUK (Chave de Desbloqueamento Pessoal) que tem comprimento mínimo de 4 dígitos e máximo de 8 dígitos.

3.6.2 Centro de autenticação (AuC)

Segundo Alencar (2004), o centro de autenticação (AuC) é utilizado por razões de segurança. O AuC tem como funcionalidade fornecer três parâmetros para a autenticação sendo eles:

- Resposta Sinalizada (SRES): Mensagem gerada atraves da tupla (Kc,RAND);
- Número aleatório (RAND): Número aleatório gerada pelo AuC e enviado ao dispositivo móvel para que ele possa gerar um SRES;
- parâmetro Kc: Chave utilizada para autenticar o usuário, onde a chave está armazanada no SIMCard e no AuC;

3.6.3 Autenticação

O procedimento de autenticação verifica a identidade dos assinantes e se eles têm permissão para usar uma rede em particular. A autenticação é baseada no algoritmo A3, que é armazenado no cartão SIM e no centro de autenticação (AuC) (Alencar, 2004, p.365).

O Algoritmo A3 utiliza dois parâmetros de entrada. A Figura 15 ilustra este processo onde o primeiro parâmetro uma chave que é armazenada no SIM e na rede, o segundo parâmetro é um número aleatório gerado no AuC e transmitido para o dispositivo móvel. Ao receber o número aleatório o dispositivo utiliza este número como parâmetro de entrada para o algoritmo A3, o algoritmo A3 gera o SRES e envia de volta para a rede onde a mesma compara se o SRES recebido do dispositivo móvel é o mesmo que foi valor gerado no centro de autenticação (Alencar, 2004, p.366).

Figura 15. Autenticação GSM.

3.6.4 Criptografia

Segundo Alencar (2004), a criptografia é usada para proteger dados de sinalização e informação do usuário. Ela é realizada após a codificação dos diferentes canais lógicos, independente de o canal ser de sinalização ou de tráfego.

De acordo com Alencar (2004, p.384), “A proteção da mensagem e realizada em duas etapas. Primeiro, uma chave criptografada é gerada usando o algoritmo A8 junto com a chave do usuário e o RAND enviado pela rede. Segundo um número de 114 bits é produzido com a chave criptografada, um algoritmo chamado A5 e o número do quadro TDMA. É feita uma operação do tipo XOR entre essa seqüência de bits e dois blocos de dados de 57 bits incluídos em um burst normal.”.

3.7 SMS (*Short Message Service*)

SMS (Sort Message Service) é um serviço utilizado para envio de mensagens curtas de no máximo 160 caracteres. Inicialmente o serviço foi criado para a finalidade de enviar mensagens de alerta aos usuários de dispositivos móveis, como

por exemplo, mensagem de alerta de mensagem sobre a existência de mensagens na caixa de correio de voz. A primeira mensagem SMS enviada foi em meados de 1992 no Reino Unido.

As mensagens são enviadas através do serviço SMSC (Sort Message Service Central) que é uma central de envio de mensagens SMS, também é possível enviar pela internet utilizando o protocolo SMPP (short message peer-to-peer protocol) que é o protocolo utilizado para enviar mensagem SMS à rede de telefonia móvel. A Figura 16 ilustra o envio de uma mensagem SMS utilizando SMSC, (WIKIPEDIA, 2006).

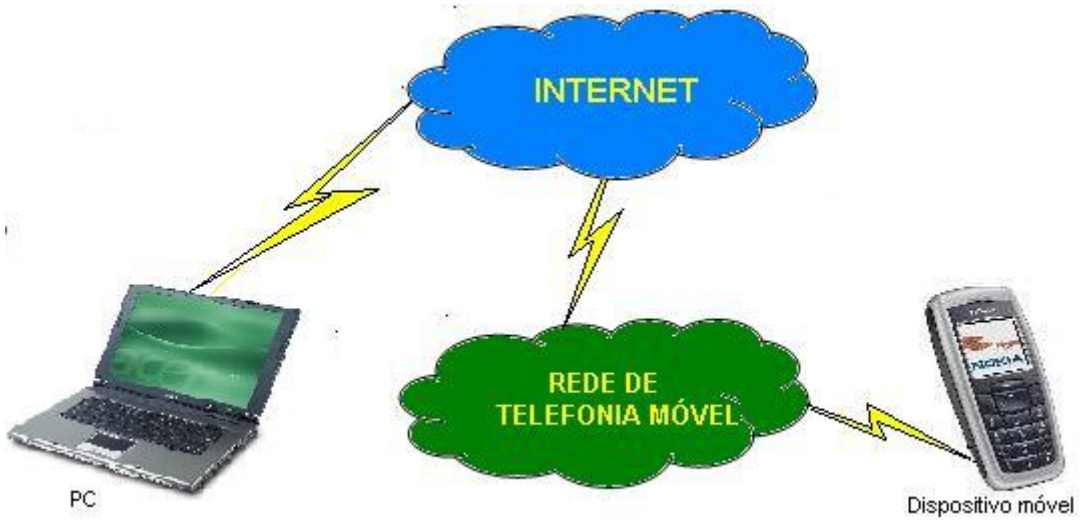


Figura 16. SMSC.

3.8 Sistemas operacionais para dispositivos portáteis:

Assim como para desktop e Mainframes existe também sistemas operacionais específicos para dispositivos portáteis tais como o Windows CE, Symbian, PalmOS, PocketOS, entre outros. Neste trabalho será descrito resumidamente os três primeiros.

3.8.1 Windows Mobile / CE:

O Windows CE é um novo sistema operacional da Microsoft que tem a mesma aparência e características do Windows 95, e é utilizado pelos Handhelds e Palmtops.

O Windows Mobile / CE suporta diversas linguagens como C, C++, C# ,Visual Basic, entre outras.

Esse Sistema Operacional utiliza um conjunto de bibliotecas criptográficas CryptoAPI, a mesma utilizadas em outras versões do Windows.

A CryptoAPI é um conjunto de bibliotecas que permitem ao desenvolvedor invocar funções que cifrem e decifrem informações e ainda utilizem de certificados digitais. A CriptoAPI funciona basicamente da seguinte maneira. A Aplicação comunica com o Sistema Operacional utilizando a CryptoAPI e o Sistema Operacional comunica com o CSP (*cryptographic service providers)* utilizando a interface de serviços criptográficos CSPI como se pode observar na Figura 17.

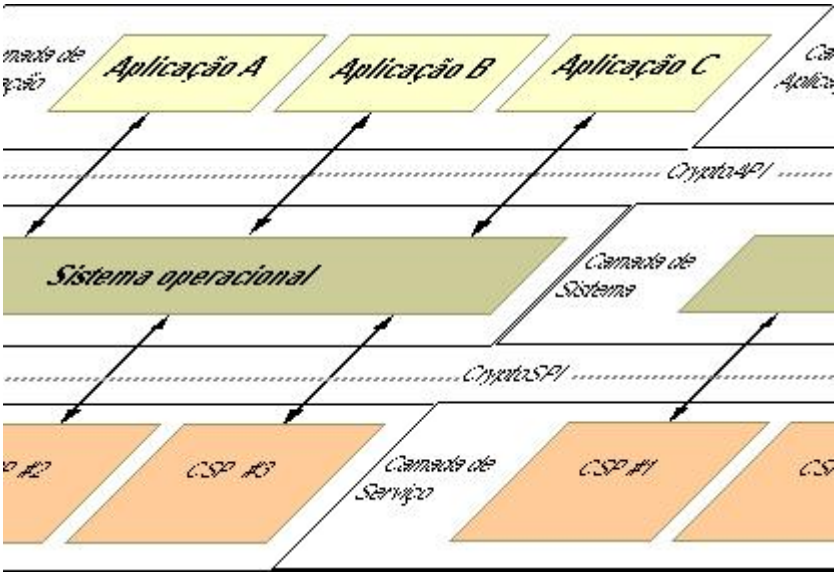


Figura 17. Arquitetura de serviços criptográficos do Windows Mobile. Fonte Microsoft.

Os CSPs da do Windows CE/Mobile suportam:

- RSA Base Provider: Suporte assinatura digital e sigilo de dados;
- RSA Enhanced Provider: Suporta a chaves de 128-bits;
- Diffie-Hellman CSP: Suporte ao Diffie-Hellman, SHA Hashing, assinatura de dados e verificação de assinatura;
- Smart Card CSP: Suporta smart cards para Windows;

A Tabela 1 exibe alguns algoritmos criptográficos simétricos suportados pelas CSPs:

Algoritmos Criptográficos Simétricos
3DES
3DES COM DUAS CHAVES
DES
RC2
RC4
RC5

Tabela 1: Algoritmos Simétricos suportados pela CryptoAPI.

A Tabela 2 exibe alguns algoritmos de *hash* suportados pelas CSPs:

Algoritmos de *hash*
MD2
MD4
MD5
SHA
SHA1

Tabela 2: Algoritmos *hash* suportados pela CryptoAPI.

3.8.2 Symbian:

Symbian é um produto que foi desenvolvido por algumas empresas na área de telefonia móvel, como Nokia, Motorola, Panasonic, Sony Ericsson, Psion e Siemens.

O Symbian suporta algumas linguagens de programação como C, C++, Java, Assembler, Java Script WMLScript, OPL.

O Symbian também suporta serviços de segurança como autenticação, integridade, confidencialidade e irretratabilidade. Para isso as API's fornecem algoritmos criptográficos, *hash*, geração de chaves, geração de números randômicos e geração de certificados.

O symbian suporta o gerenciamento de certificados que inclui as seguintes funcionalidades.

- Armazenamento e recuperação dos certificados;

- Construção e validação do certificado;
- Verificação do status do certificado;

As API's do gerenciamento de certificados têm uma hierarquia como mostra a

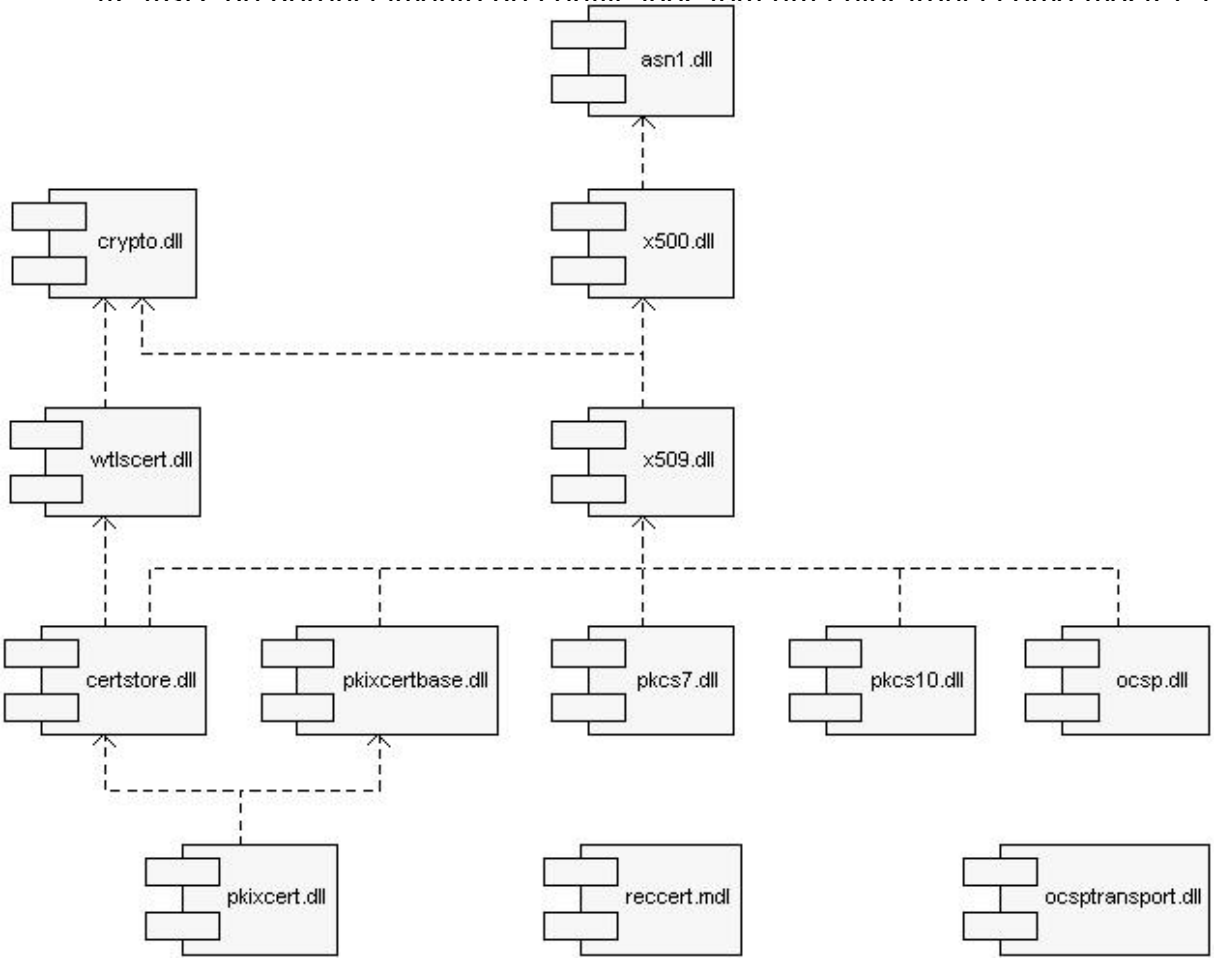


Figura 18. Hierarquia de gerenciamento de certificados no symbian 8.0.

As API's dão suporte a infra-estrutura de chaves públicas, Certificados Digitais e assinaturas Digitais. A Figura 19 ilustra a arquitetura de segurança do sistema Symbian v8.0;

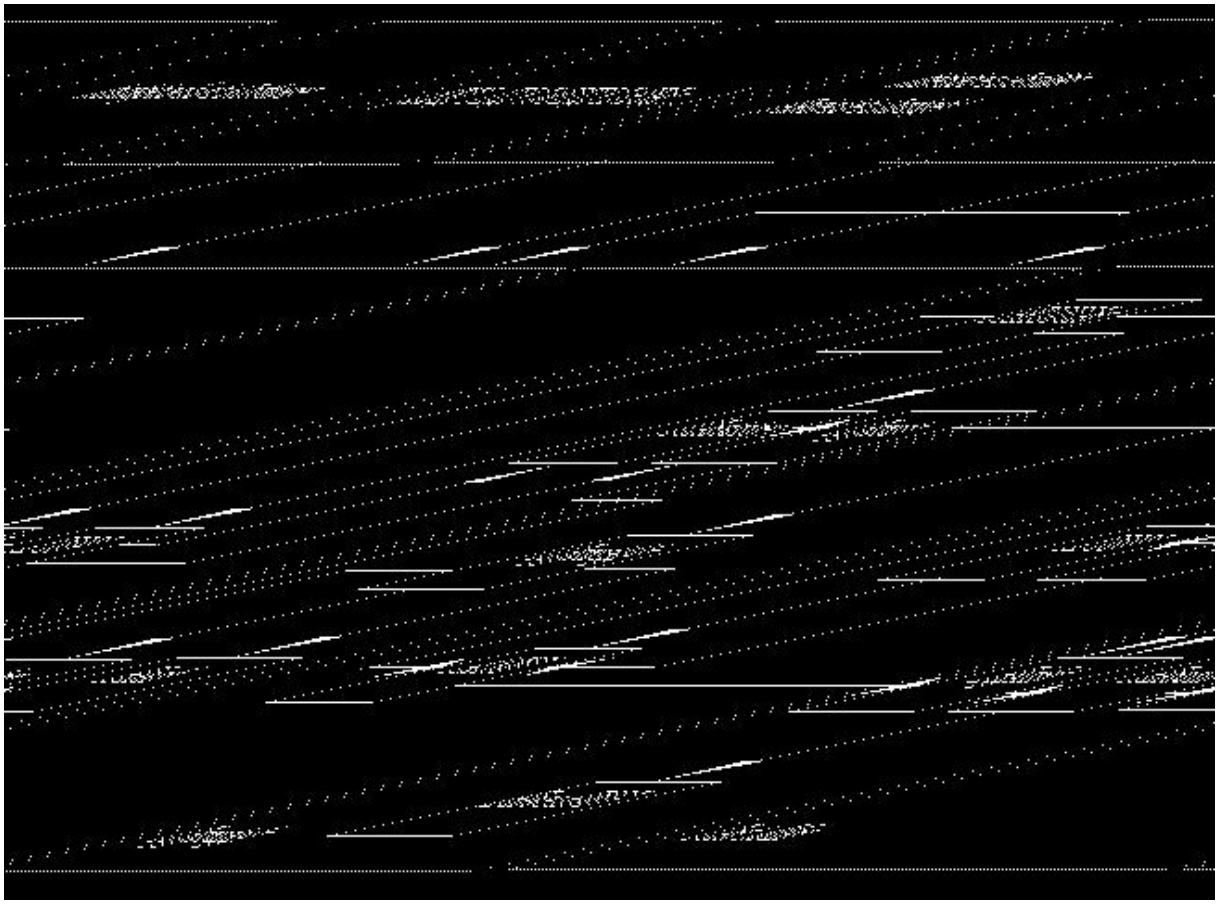


Figura 19. Arquitetura de segurança do symbian 8.0.

O symbian suporta o protocolo SSL (*Security Socket Layer*) vamos descrever algumas funções deste protocolo implementado no Symbian. Há a classe principal chamada de CSecEngine, onde outras classes são derivadas dela. Inicialmente deve-se efetuar a conexão através do da função Connect(), especificando servidor, porta, e o destino HTTP caso seja uma conexão HTTPS. Após a chamada a função Connect() a CSecEngine inicia um estado de espera no dispositivo onde fica aguardando uma requisição de resposta. Após a resposta a CSecEngine executa a função RunL() agora é necessário executar mais alguns passos para que a conexão seja estabelecida com sucesso, esse processo é chamado de *handshake.*

MakeSecureConnectionL(): criar um objeto CSecureSocket, e inicializar com a propriedade SecureSocket::StartClientHandshake();

MakePageRequestL() enviar o um pedido ao servidor utilizando a função CSecureSocket::Send();

GetServerResponseL(): nesta função é negociado o SSL nas versões 1.0, 2.0 3.0 ou TSL, através da função CSecureSocket::Protocol(), após isso é necessário estabelecer o algoritmo criptográfico que será utilizado na comunicação

CSecureSocket::CurrentCipherSuite() e requisitar o certificado do servidor utilizando a função CSecureSocket::ServerCert(), agora é necessário ficar esperando a resposta do servidor através da função CSecureSocket::Recv()

ReadServerResponseL(): examina a resposta do servidor e já inicia a conexão onde agora cliente/servidor podem trocar informações através de um túnel seguro criado pela conexão SSL.

ConnectionClosed(): utilizado para fechar a conexão com o servidor.

Para enviar uma mensagem SMS ou SMPT no caso de um e-mail há uma API especifica onde é implementada a classe CsendAs, onde é necessário criar um objeto no menu da aplicação e informar o assunto, os recipientes da mensagem, corpo da mensagem e anexos.

CSender::SenderCascadeMenuL(): adiciona ao menu da aplicação um objeto para que possa enviar a mensagem.

CSender::CreateMessageL(): Cria mensagem especificando o tipo da mensagem (SMTP, SMS), corpo da mensagem, assunto, anexo, e recipientes da mensagem.

3.8.3 PalmOS:

O PalmOS é um sistema operacional para dispositivos moveis mais utilizados nos dias atuais com cerca de 38 milhões de dispositivos sendo utilizados no mundo, ele é um produto da Palm Powered.

O PalmOS também dá suporte a diversas linguagens como C, C++, Java, Basic entre outras.

O Palm OS também suporta o protocolo SSL/TSL, onde são utilizando algumas chamadas as funções primeiramente as negociações (Handshake) e posteriormente as trocas de informações.

Inicialmente é necessário criar um objeto do tipo SSL lib, depois atribuirmos a ele o contexto através da chamada de função SslContextSet_Socket(). Agora é necessário escolher o SSL que irá trabalhar versões 1.0, 2.0 3.0 ou TSL utilizando a função SslContextSet_Mode().

Utilizando a função SslContextSet_InfoCallback() para verificar informações sobre o socket indicando que o socket ficará em aguardo enquanto a conexão não é efetuada. A função SslContextGet_HsState() retorna o estado do *handshake*, e

posteriormente é utilizada a função SslContextSet_VerifyCallback() para verificação do *handshake*. Após a negociação efetuada com êxito é utilizado a função SSLReceive para receber informações e SSLSend para enviar informações.

4 DESCRIÇÃO DO PROJETO

Para realização deste projeto foi implementado um sistema de troca de mensagens SMS protegidas. O sistema utilizou tecnologia J2ME para cifrar e decifrar as mensagens SMS.

4.1 Metodologia do projeto

A metodologia usada neste projeto consiste no estudo e utilização de ferramentas que depois foram utilizadas na implementação e análise de um sistema de troca de mensagem SMS cifradas com chave simétrica.

Para comunicação e implementação no sistema operacional do dispositivo móvel foi realizado um estudo e implementação utilizando a linguagem de programação Java sendo utilizado a arquitetura J2ME e para os serviços criptográficos foi estudado a biblioteca Bouncy Castle.

A seguir serão descritas a arquitetura J2ME, principais características de segurança da biblioteca Bouncy Castle e depois a especificação do sistema de troca de mensagens SMS.

4.2 Tecnologias Utilizadas

Abaixo será discutido sobre as tecnologias utilizadas neste projeto que são a tecnologia J2ME com suas características e limitações, e a biblioteca Bouncy Castle com suas funções criptografias utilizadas no projeto.

4.2.1 Java:

A tecnologia Java é uma linguagem orientada a objeto de plataforma independente e com ambiente de programação *multithreading*. Ele pode ser utilizado para web, serviços de rede, aplicações, plataformas independentes, robos e outros dispositivos, como os dispositivos de pequeno porte que é o foco desta pesquisa.

Conforme ilustra a Figura 20, atualmente existem três edições da tecnologia Java, que são o J2SE(Java 2 Platform, Standard Edition) que provê um desenvolvimento para desktop e servidores. J2EE (Java Platform, Enterprise Edition) que é um padrão para desenvolvimento de aplicativos robustos e escaláveis, sendo o J2EE uma extensão do J2SE, implementando algumas funcionalidades adicionais, como web services, gerenciamento de aplicações, serviços orientados a arquitetura (SOA) e aplicações web. O J2ME é uma edição do J2SE reduzido por causa das limitações de dispositivos de pequeno porte, sendo o J2ME a tecnologia utilizada neste projeto.

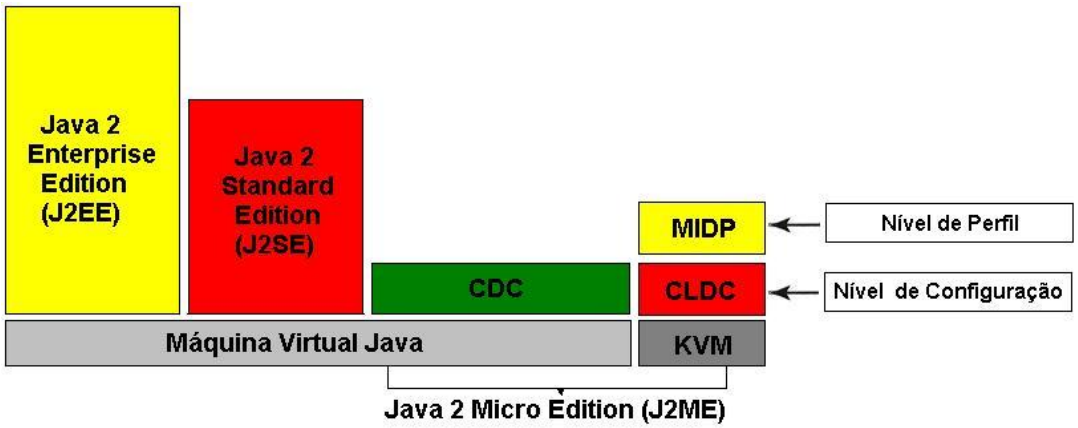


Figura 20. Várias edições do Java, Fonte: MUCHOW.

4.2.1.1 J2ME:

O J2ME é uma derivação da tecnologia J2SE, que é a especificação padrão da tecnologia Java, pois a J2ME foi projetada para dispositivos de pequeno porte, com memória, processamento e vídeo limitados. Os recursos dentro do J2ME podem variar muito, pois há diferenças entre os dispositivos móveis de pequeno porte, tanto configurações de vídeo, como memória e processamento.

Como em desktops, ou servidores há a necessidade da criação de uma maquina virtual Java. Está maquina virtual, também conhecida como JVM (Java Virtual Machine) é uma configuração introduzida pela sun para uma ampla variedade de dispositivos, ou seja, nela é que são definidos os recursos e as bibliotecas básicas para uma configuração em particular.

Basicamente o J2ME é uma versão reduzida do J2SE, onde por questão de eficiência foram removidos vários componentes para ser mantido a eficiência. Um

exemplo é que muitos dispositivos moveis não tem os recursos de tela para fornecer componentes avançados com janelas sobrepostas e menus suspensos.

O J2ME tem algumas configurações e perfis que influenciam na implementação desta tecnologia, pois irá depender da configuração do dispositivo móvel e de seu perfil para que o mesmo suporte ou não funcionalidades do J2ME.

4.2.1.1.1 CDC (CONFIGURAÇÃO DE DISPOSITIVO CONECTADO)

Para esta configuração, é necessário pelo menos 512 Kb de memória para execução do Java, e 256 Kb de memória para alocação em tempo de execução e largura do canal de trafego de informações persistente e alta.

4.2.1.1.2 CLDC (CONFIGURAÇÃO DE DISPOSITIVO CONECTADO LIMITADO)

Para está configuração é necessário 128 Kb para executar o Java, 32 Kb para alocação em tempo de execução, interface restrita ao usuário, normalmente com o dispositivo móvel alimentado por bateria e com largura do canal de trafego de informações baixa e são intermitentes. À medida que a tecnologia de dispositivos de pequeno porte for evoluindo, evoluirá também a tecnologia Java para esses dispositivos, o que já ocorreu, a CLDC evoluiu da versão 1.0 para a versão 1.1 e a MIDP passou de 1.0 para 2.0, como veremos a seguir.

4.2.1.1.3 MIDP (Mobile Information Device Profile) 1.0 e MIDP (Mobile Information Device Profile) 2.0:

Como foi discutido anteriormente, houve uma necessidade de ser criado uma JVM para dispositivos de pequeno porte, denominada KVM, que é uma implementação da Sun de uma JVM que atende as especificações CLDC.

Os recursos considerados no MIDP 2.0 podem incluir HTTPS, *sockets* e datagramas, extensões para interface de baixo nível e um analisador XML.

A tecnologia MIDP evolui conforme o hardware para esses dispositivos evoluem também e a especificações onde a Sun discute com a comunidade as necessidades que podem/devem ser implementadas nas próximas versões, teoricamente qualquer um pode dar sugestões para novas implementações, basta apenas se cadastrar no site da sun e ir à lista de discussões.

4.2.2 BOUNCY CASTLE

A biblioteca Bouncy Castle é um pacote de implementação de algoritmos de criptografia. Este pacote fornece suporte apropriado as API's para o ambiente que será utilizado (J2ME) que foi desenvolvido seguindo as especificações do *framework* JCE (*Java Cryptography Extension*).

Com as bibliotecas Bouncy Castle é possível implementar atribuição de certificados, PCKS12, SMINE e OpenPGP.

Neste trabalho foram utilizadas apenas algumas funcionalidades de muitas existentes na biblioteca Bouncy Castle, mais especificamente algumas funcionalidades de criptografia simétrica e algumas de hash. Abaixo será descrito as funcionalidades utilizadas.

- Interface BlockCipher: Interface que é utilizada por outras classes que implementam cifras de bloco, a Tabela 3 descreve os métodos da interface BlockCipher.

Método	Descrição
getAlgorithmName	Retorna uma String com nome do algoritmo de cifra que implementou esta classe
getBlockSize	Retorna um inteiro com o tamanho do bloco para a cifra implementada
Init	Inicializa a cifragem ou decifragem
processBlock	Processa um bloco de entrada retornando um bloco como saída.
Reset	Reinicializa a cifra

Tabela 3 : Métodos BlockCipher.

- Classe CBCBlockCipher: Implementa o modo de cifra de corrente de bloco (Cipher-Block-Chaining) mais conhecida como CBC:
  A classe CBCBlockCipher contém apenas os métodos que da interface BlockCipher já que a classe CBCBlockCipher implementa a interface BlockCipher.

- Classe DESEngine: Esta classe implementa o algoritmo DES, ela também implementa a interface BlockCipher, sendo assim ela contém os métodos da interface BlockCipher mais os métodos descritos na Tabela 4.

Método	Descrição
desFunc	Retorna um array de byte com o resultado da cifragem.
generateWorkingKey	Gera uma chave baseada na chave secret.

Tabela 4 : Métodos DESEngine.

- Classe AESEngine: Esta classe implementa o algoritmo AES (Rijndael), ela também implementa a interface BlockCipher;
- Classe IDEAEngine: Esta classe implementa o algoritmo IDEA (Algoritmo Internacional de Encriptação de Dados), ela também implementa a interface BlockCipher;
- Classe RC6Engine: Esta classe implementa o algoritmo RC6, ela também implementa a interface BlockCipher;

- Classe BufferedBlockCipher: Esta classe permite que as cifras do bloco sejam usadas processar dados, ela implementa os métodos da interface BlockCipher além de conter os seguintes métodos descritos na Tabela 5:

Método	Descrição
doFinal	Finaliza o processamento de cifragem do bloco.
getOutputSize	Retorna o tamanho do buffer necessário para gerar uma saída.
getUnderlyingCipher	Retorna a cifra utilizada
getUpdateOutputSize	Retorna o tamanho do buffer necessário para gerar uma saída a partir de um buffer de entrada

Tabela 5: Métodos BufferedBlockCipher.

- Classe PaddedBufferedBlockCipher: é uma subclasse da classe BufferedBlockCipher, herdando os métodos getBlockSize, getUnderlyingCipher e reset, e sendo implementado os métodos doFinal, getOutputSize, getUpdateOutputSize e getUpdateOutputSize, contendo também os métodos descritos na Tabela 6:

Método	Descrição
processByte	Processa um único byte, gerando de retorno um array de byte caso necessário.
processBytes	Processa um array de byte, gerando de retorno um array de byte caso necessário.

Tabela 6: Métodos PaddedBufferedBlockCipher.

- Interface Digest: Esta interface é utilizada por outras classes que utilizam algoritmos hash e contém os seguintes métodos descritos na Tabela 7:

Método	Descrição
doFinal	Finaliza o processamento do hash.
getAlgorithmName	Retorna o algoritmo hash utilizado
getDigestSize	Retorna o tamanho do hash gerado
Reset	Reinicializa o hash
Update	Atualiza o hash com um byte de entrada
Update	Atualiza o hash com um array de byte de entrada

Tabela 7: Métodos Digest.

- Classe MD5Digest: contém o algoritmo de hash MD5 e implementa os métodos da classe Digest mais os seguintes métodos descritos na Tabela 8:

Método	Descrição
processBlock	Processa um bloco
processLength	Informa o tamanho do bloco a ser processado
processWord	Processa um bloco passado um array de byte

Tabela 8: Métodos MD5Digest.

- Classe SHA1Digest: contém o algoritmo de hash MD5 e implementa os métodos da classe *Digest* e os métodos de mesmo nome e funcionalidade da classe MD5Digest.

4.3 Especificação do Projeto

Para desenvolver o projeto foi utilizada a tecnologia J2ME para troca de mensagens SMS entre dispositivos móveis, e a biblioteca Bouncy Castle para cifrar, decifrar e calcular o *hash* das mensagens, além disso, para o desenvolvimento das aplicações foi utilizado como ferramenta de desenvolvimento a IDE NetBeans, o MIDway para transferência da aplicação para o celular.

O sistema desenvolvido foi chamado de LRIEncrypt. O LRIEncrypt tem as funcionalidades de enviar e receber mensagens SMS cifradas, validar a integridade

das mensagens SMS, armazenar mensagens SMS no dispositivo móvel, seja as mensagens SMS enviadas ou recebidas, configurar o algoritmo de criptografia e o algoritmo de *hash* que será utilizado para cifrar e decifrar as mensagens trocadas entre os dispositivos móveis.

Conforme mostra a Figura 21, o LRIEncrypt necessita de duas entidades sendo denominada uma de emissora e a outra de receptora, sendo que um dispositivo móvel pode ser o emissor e o receptor. O emissor fornecerá ao sistema a mensagem que deseja enviar ao receptor, o número do telefone celular do receptor e uma senha que terá que ser compartilhada entre as duas entidades. Depois de fornecido estas informações o emissor irá enviar a mensagem cifrada ao receptor. Esta mensagem irá trafegar pela rede de telefonia celular que repassará a mensagem cifrada ao receptor que ao receber a mensagem cifrada informará a senha compartilhada para que esta mensagem possa ser decifrada, restaurando assim a mensagem original.

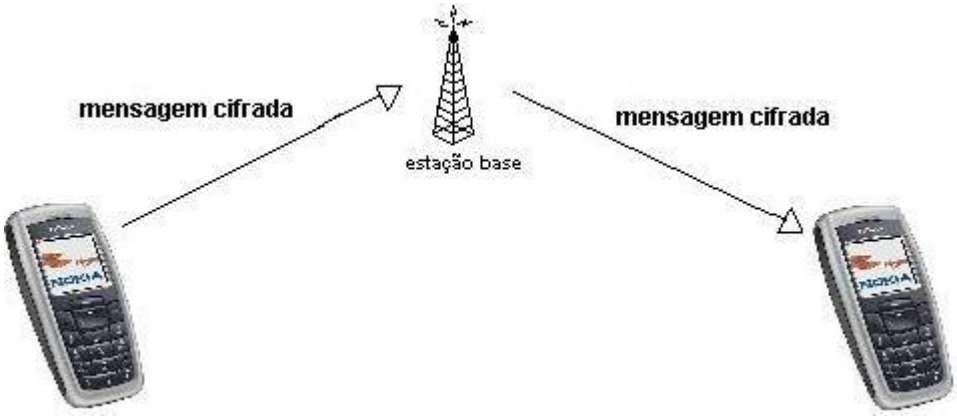


Figura 21. Funcionamento LRIEncrypt.

Para que fosse possível enviar e receber mensagens SMS cifradas com vários tipos de algoritmos de criptografia e algoritmos de *hash* de maneira transparente ao usuário, foi desenvolvido um protocolo que foi separado em 5 partes e o tamanho ocupado por cada parte no protocolo desenvolvido.

1ª Parte: Algoritmo de criptografia utilizada na mensagem, utilizando um byte;

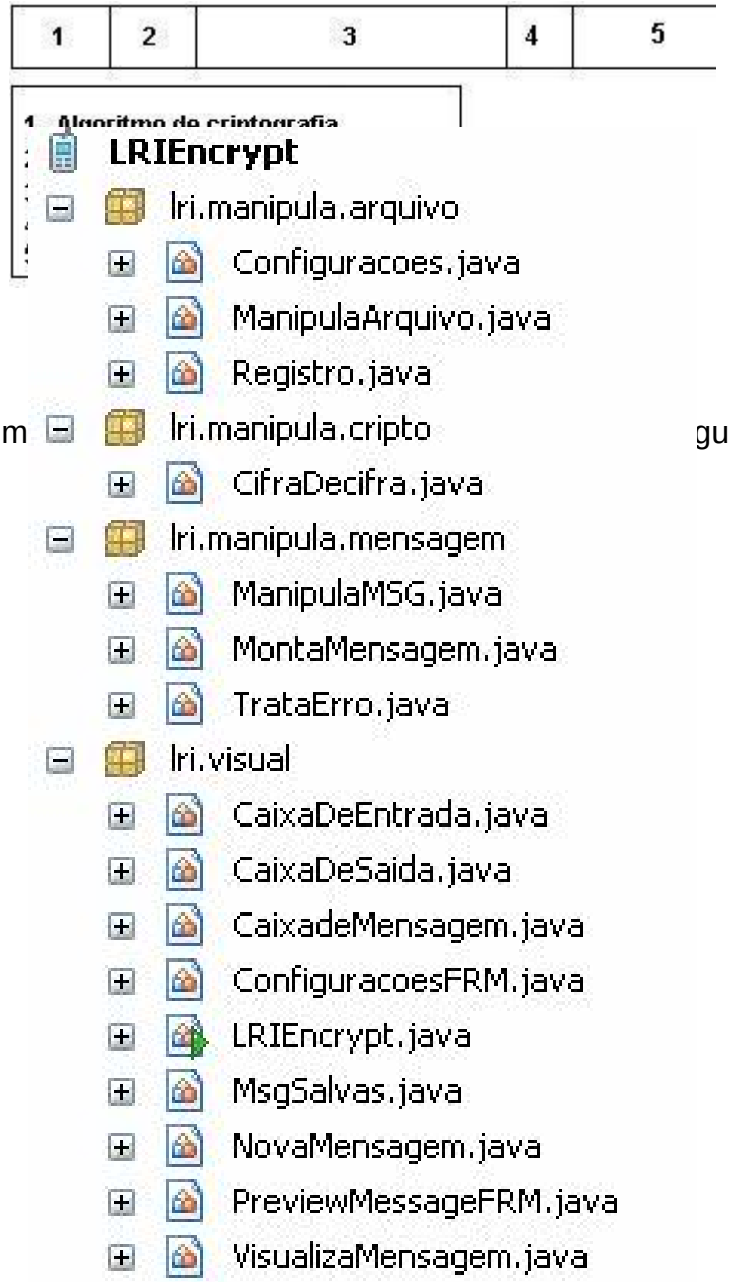
2ª Parte: Algoritmo *hash* utilizada na mensagem, utilizando um byte;

3ª Parte: Mensagem cifrada com as configurações acima, utiliza X bytes, sendo X bytes gerados na cifragem do texto da mensagem SMS, que pode ter no máximo 120 bytes;

4ª Parte: Tamanho da mensagem cifrada, utiliza três bytes;

5ª Parte: *Hashing* da mensagem, depende do algoritmo de hash utilizado, sendo o MD5 utilizando 16 bytes e o SHA1 utilizando 20 bytes;

Caso esta mensagem ultrapasse 160 bytes, que é o tamanho máximo de uma mensagem SMS, a mensagem será dividida em mais pacotes, sendo apenas no primeiro pacote enviado o algoritmo de criptografia e o algoritmo *hash* utilizado na mensagem. A Figura 22 mostra o protocolo desenvolvido para este projeto:



Conforme m guintes classes:

Figura 23, Estrutura das classes.

4.3.1 Estrutura das classes utilizadas no projeto

Nesta seção descreve-se a estrutura de classes utilizadas no sistema LRIEncrypt.

4.3.1.1 Pacote lri.manipula

Este pacote é responsável pela manipulação dos arquivos, as mensagens e a criptografia utilizadas no projeto. A Figura 24 ilustra a modelagem UML do pacote lri.manipula.

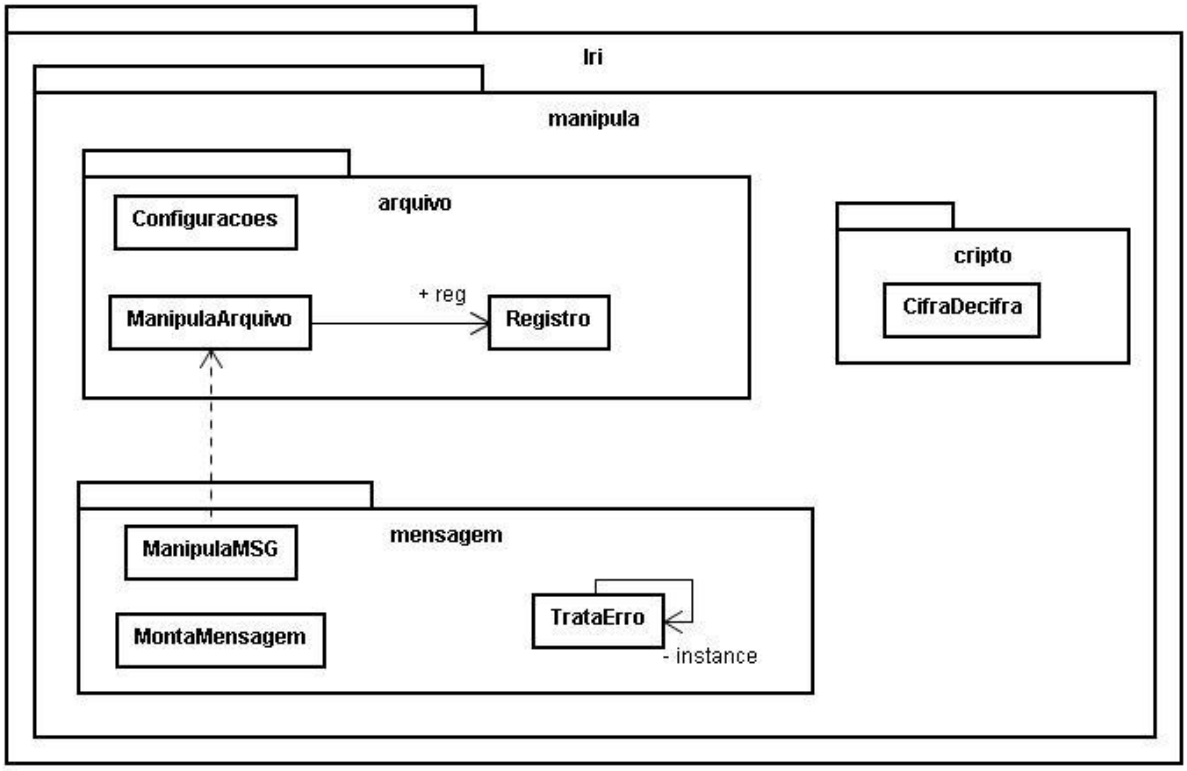


Figura 24, Modelagem UML do pacote lri.manipula.

4.3.1.1.1 Pacote lri.manipula.arquivo

Responsável pela manipulação de arquivos e objetos criados para manipulação de arquivos.

4.3.1.1.1.1 Classe lri.manipula.arquivo.Registro

Responsável pelos dados contidos em uma mensagem, como por exemplo, número do telefone que foi/será enviado a mensagem, assunto, mensagem, imagem para implementações futuras, algoritmo de criptografia, e algoritmo hash;

4.3.1.1.1.2 Classe lri.manipula.arquivo.ManipulaArquivo

Responsável por manipular os dados lidos e gravados das mensagens enviadas, recebidas, e salvas. Nesta classe contém alguns atributos utilizados na manipulação de arquivos, como por exemplo, o nome do arquivo, o arquivo, e um vetor de Registros. Nela é implementada métodos de manipulação de arquivo, como por exemplo, ler e gravar registros no arquivo, abrir/criar arquivo, apagar um ou mais registros e fechar arquivo, o anexo 1 e anexo 2 mostra os dois principais métodos desta classe que são o “gravaregistro” e o “leregistro”:

4.3.1.1.1.3 Classe lri.manipula.arquivo.Configurações

Responsável por manipular os dados lidos e gravados das configurações escolhidas pelo usuário, nesta classe como na anterior tem métodos de leitura e gravação de dados em arquivos, onde os dados são os algoritmos de criptografia que será utilizado na troca de SMS e o algoritmo hash;

4.3.1.1.2 Pacote lri.manipula.mensagem

Responsável pela manipulação das mensagens.

4.3.1.1.2.1Classe lri.manipula.mensagem.ManipulaMSG:

Responsável por manipular a mensagem, lendo e gravando dados no registro da mesma. Nesta classe tem métodos como montar lista de mensagens, tanto enviadas como recebidas, retorna a mensagem corrente;

4.3.1.1.2.2Classe lri.manipula.mensagem.MontaMensagem

Responsável pela montagem e desmontagem do protocolo criado para envio e recebimento das mensagens;

4.3.1.1.2.3Classe lri.manipula.mensagem.TrataErro

Responsável pelo tratamento de erro ocorrido durante a manipulação das mensagens;

4.3.1.1.3 Pacote lri.manipula.cripto

Responsável pela criptografia implementada no projeto.

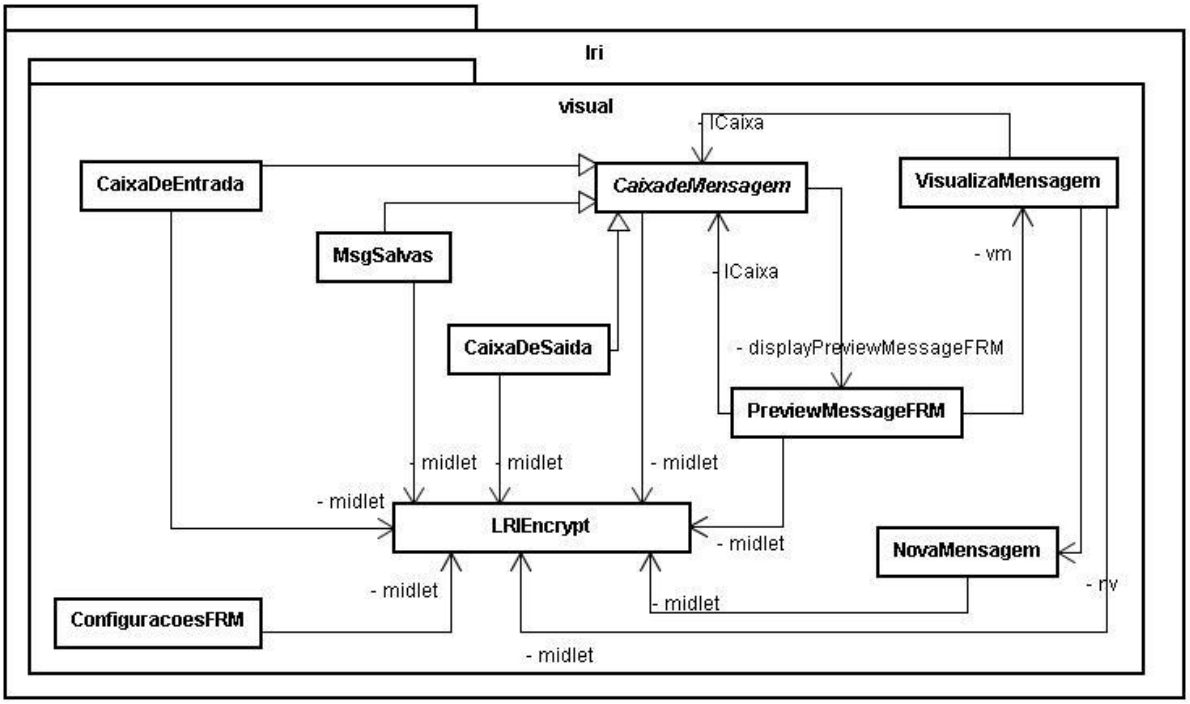
4.3.1.1.3.1Classe lri.manipula.cripto.CifraDecifra

Responsável pela criptografia utilizada para cifrar, decifrar e calcular os hash das mensagens enviadas e recebidas. Como pode ser observado no Anexo 3 foi

criado um método para cifrar informações e no Anexo 4 um método para decifrar as informações.

4.3.1.2 Pacote lri.visual

Este pacote é responsável por toda a parte visual do projeto. A Figura 25 ilustra a modelagem UML do pacote lri.visual.



:

Figura 25, Modelagem UML do pacote lri.visual.

4.3.1.2.1 Classe lri.visual.CaixadeEntrada

Responsável por apresentar e as mensagens recebidas pelo dispositivo móvel;

4.3.1.2.2 Classe lri.visual.CaixadeSaida

Responsável por apresentar as mensagens enviadas pelo dispositivo móvel;

4.3.1.2.3 Classe lri.visual.CaixadeMensagem

Responsável por manipular as mensagens enviadas, recebidas e salvas pelo dispositivo móvel;

4.3.1.2.4 Classe lri.visual.Configurações

Responsável por apresentar e manipular as configurações de algoritmo de criptografia simétrica e *hash* utilizados para o envio da mensagem;

4.3.1.2.5 Classe lri.visual.LRIEncrypt

Classe principal que através dela pode-se chegar as funcionalidades da aplicação tendo um menu de opções para tais funcionalidades descritas acima;

4.3.1.3 Pacote lri.teste

Este pacote é responsável por toda a parte teste do projeto. A Figura 27 ilustra a modelagem UML do pacote lri.visual.

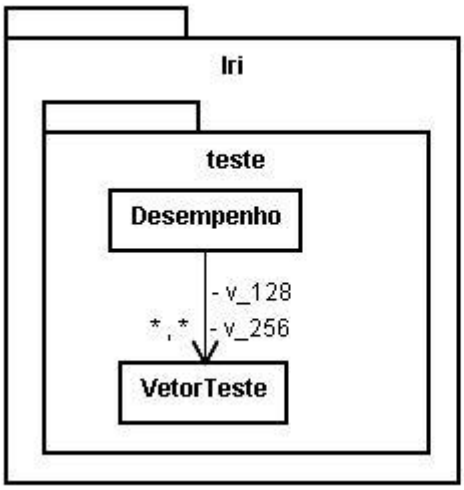


Figura 26, Modelagem UML do pacote lri.teste.

4.3.2 Funcionamento do LRIEncrypt

Nesta seção descreve-se as funcionalidades do sistema LRIEncrypt para envio de mensagens SMS cifradas.

Apresentação: A primeira tela do LRIEncrypt é uma tela de apresentação como ilustrada na Figura 27. Nesta tela é apenas apresentado informações sobre o desenvolvedor.

Figura 27, tela de apresentação. Fazer para todas as telas

Menu Principal: Após a apresentação inicial esta é exibida a tela do menu principal como ilustrado na Figura 28. No menu principal é apresentado ao usuário às opções de escrever uma nova mensagem, ler mensagens enviadas, ler mensagens recebidas, mensagens salvas, configurações do sistema e desempenho.

Figura 28. Menu principal.

Nova Mensagem: A tela de Nova Mensagem é apresentada ao usuário após ter escolhido a opção Nova Mensagem no Menu Principal como ilustrada na Figura 29. A opção de Nova mensagem fornece ao usuário uma interface, onde o usuário informará dados sobre a mensagem SMS, como a mensagem a ser enviada que tem o limite de 120 caracteres, o telefone do destinatário, e uma senha que deve conter no mínimo 8 caracteres, sendo que todas as três informações são obrigatórias. Depois de fornecida estas informações, o usuário poderá escolher voltar ao menu principal ou enviar a mensagem ao destinatário informado.

Figura 29, Nova mensagem.

Alerta de erro de senha: A tela de Alerta de erro de senha é apresentada ao usuário caso informe uma senha que contenha menos de 8 caracteres. A Figura 30 ilustra a mensagem exibida.

Figura 30, Alerta de erro de senha.

Alerta de envio de mensagem: A tela de Alerta de envio de mensagem, ilustrada na Figura 31, é exibida pela própria API do J2ME, para que o usuário confirme o envio da mensagem.

Figura 31, Alerta de envio de mensagem.

Caixa de Entrada: Quando o usuário selecionar no menu principal a opção “Caixa de Entrada”, o LRIEncript irá exibir as mensagens recebidas do usuário como ilustrado na Figura 32, que estão ordenadas pela ordem de recebimento das mensagens, o

assunto é gerado pelo LRIEncrypt que tem como conteúdo o telefone do remetente. O usuário pode navegar por todas as mensagens recebidas. Além de navegar pelas mensagens recebidas o usuário poderá visualizar a mensagem recebida escolhendo a opção visualizar ou retornar ao menu principal caso escolhendo a opção voltar.

Figura 32, Caixa de Entrada.

Caixa de Saída: Quando o usuário selecionar no menu principal a opção “Caixa de Saída”, o LRIEncrypt irá exibir as mensagens enviadas pelo usuário como ilustrada na Figura 33, que estão ordenadas pela ordem de envio das mensagens, o assunto é parte do conteúdo do corpo da mensagem enviada. O usuário poderá navegar por todas as mensagens enviadas, escolher a opção voltar caso deseje voltar ao menu principal ou visualizar para ler a mensagem enviada.

Figura 33, Caixa de Saída.

Informe a senha: A tela Informe a senha ilustrada na Figura 34 é apresentada ao usuário quando escolhido a opção visualizar, tanto na tela “Caixa de Entrada” quanto na tela “Caixa de Saída”. Na tela de Informe a senha o usuário deverá informar a senha referente à mensagem que deseja visualizar.

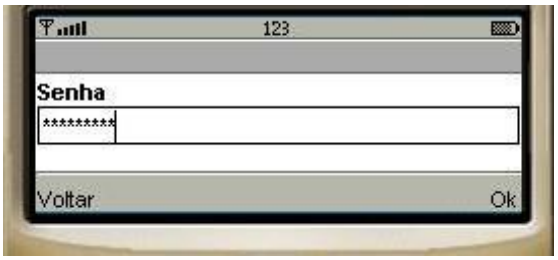


Figura 34, Informe a senha.

Alerta de senha inválida: Como ilustrado na Figura 35 a tela Alerta de senha inválida é apresentada quando o usuário informa uma senha incorreta ou ocorre algum erro na leitura da mensagem, ocorrendo qualquer um dos dois casos será exibido ao

usuário um alerta contendo a seguinte mensagem “Senha inválida ou mensagem não pode ser aberta!”.

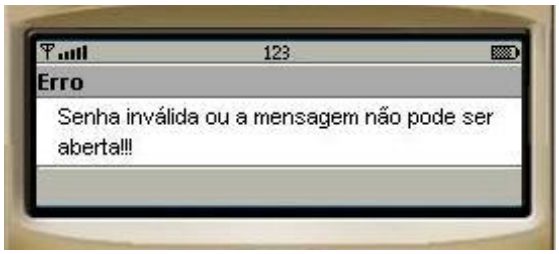


Figura 35, Alerta de senha inválida.

Alerta Integridade Tamanho Hash Diferente: Este alerta é exibido ao usuário quando o LRIEncrypt detecta um problema na integridade da mensagem, quando o mesmo compara o tamanho do hash da mensagem enviada/armazenada com o tamanho do hash calculado após a decifragem da mensagem, ocorrendo qualquer um dos casos é exibido a tela Alerta Integridade Tamanho Hash Diferente, como ilustrado na Figura 36.

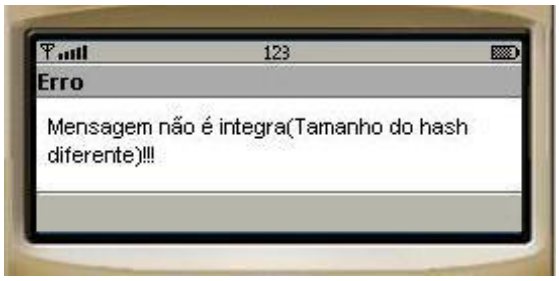


Figura 36, Alerta Integridade Tamanho Hash Diferente.

Alerta Integridade Hash Diferente: Como ilustrado na Figura 37, o Alerta Integridade Hash Diferente é exibido ao usuário quando o LRIEncrypt encontra um problema na integridade da mensagem, comparando o hash da mensagem enviada/armazenada com o hash calculado após decifrar da mensagem.

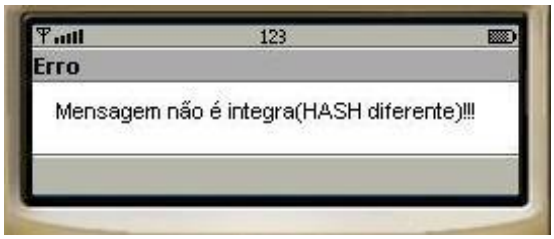


Figura 37, Alerta Integridade Hash Diferente.

Visualização de mensagem: A tela de Visualização de mensagem é exibida quando o usuário informa a senha correta da mensagem enviada/armazenada, que decifra a

mensagem validando sua integridade e exibe o resultado, como ilustrado na Figura 38.

Figura 38, Visualização de mensagem.

Configurações: Como ilustrado na Figura 39 a tela de configurações é apresentada ao usuário quando escolhido a opção de Configurações no menu principal, a tela de Configurações tem como objetivo salvar as configurações de envio de mensagem como o algoritmo de cifragem e o algoritmo de calculo de hash que será utilizado no envio de mensagem. O usuário poderá escolher utilizar o DES, AES, IDEA ou RC6 como algoritmo de cifragem e MD5 e SHA1 como algoritmo de calculo de hash, caso o usuário não deseje salvar suas modificações deve-se escolher a opção voltar que retorna ao menu principal, ou escolhe a opção Salvar que salva as configurações personalizadas pelo usuário. Por padrão o programa já vem com as configurações de DES para algoritmo de cifragem e MD5 para algoritmo de calculo de hash.

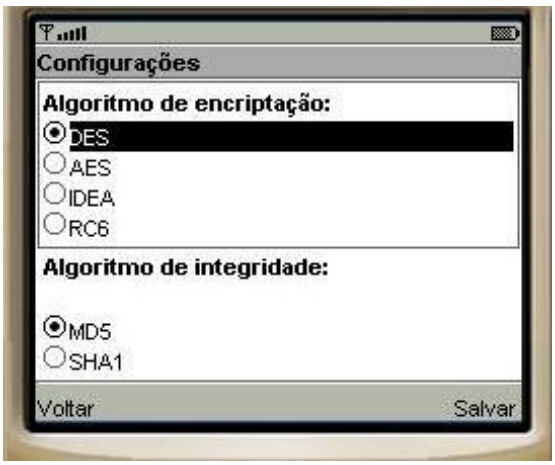


Figura 39, Configurações.

Alerta de configurações salvas: Este alerta é apresentado ao usuário quando escolhido a opção Salvar dentro da tela de Configurações, ilustrado na Figura 40. A tela Alerta de configurações salvas é apresentada ao usuário quando concluído com sucesso as modificações personalizadas pelo usuário.

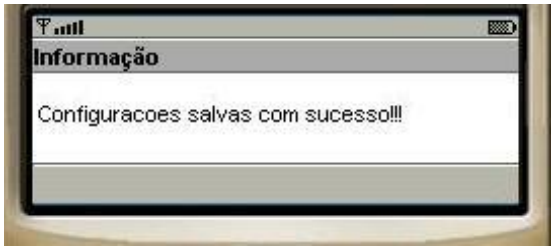


Figura 40, Alerta de configurações salvas.

Desempenho: A opção de desempenho é apresentado ao usuário quando escolhido a opção desempenho no menu principal, como ilustrado na Figura 41. Esta opção tem como objetivo medir o desempenho do dispositivo móvel com relação ao tempo que o dispositivo leva para cifrar uma mensagem combinando todos os quatro algoritmos de criptografia, os dois algoritmos de calculo de *hash* utilizados no LRIEncrypt e vários tamanhos de mensagem que estão descrito na Seção de testes.

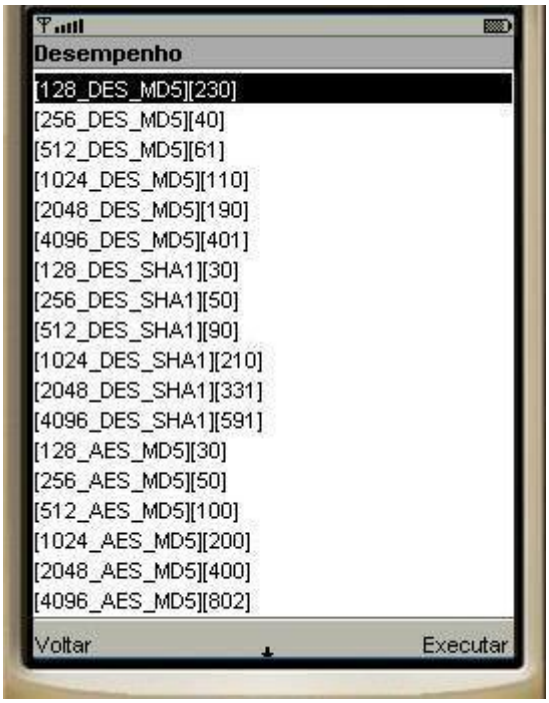


Figura 41, Desempenho.

4.4 Testes

4.4.1 Descrição da metodologia usada

Para testar o impacto da utilização de algoritmos de criptografia simétrica e de funções *hash* foi adicionado uma funcionalidade de teste de desempenho, esta funcionalidade executa a cifragem e o *hashing* de diversos tamanhos de blocos de dados, desde 128 bytes até 256, lembrando que este programa foi desenvolvido para cifrar mensagens SMS que utilizam no máximo 160 bytes, seguindo o padrão do SMS.

4.4.2 Detalhes sobre ambiente de testes

Para que fosse possível a realização de testes em um ambiente real foi utilizado um aparelho celular motorola modelo v185. A Tabela 9 descreve as características deste aparelho celular utilizado para os testes:

Característica	Detalhes
Tecnologia	WAP 2.0. J2ME, SMS, SEM, MMS, AOL/ICQ IM
Conectividade	Mini USB
Sistema Operacional	Motorola
Chipset	I250S1
J2ME	CLCD 1.0, MIDP 2.0
Biblioteca Criptográfica	Bouncy Castle release 1.29
Biblioteca de Otimização	ProGuard 3.5

Tabela 9: características do motorola v185 e tecnologias utilizadas.

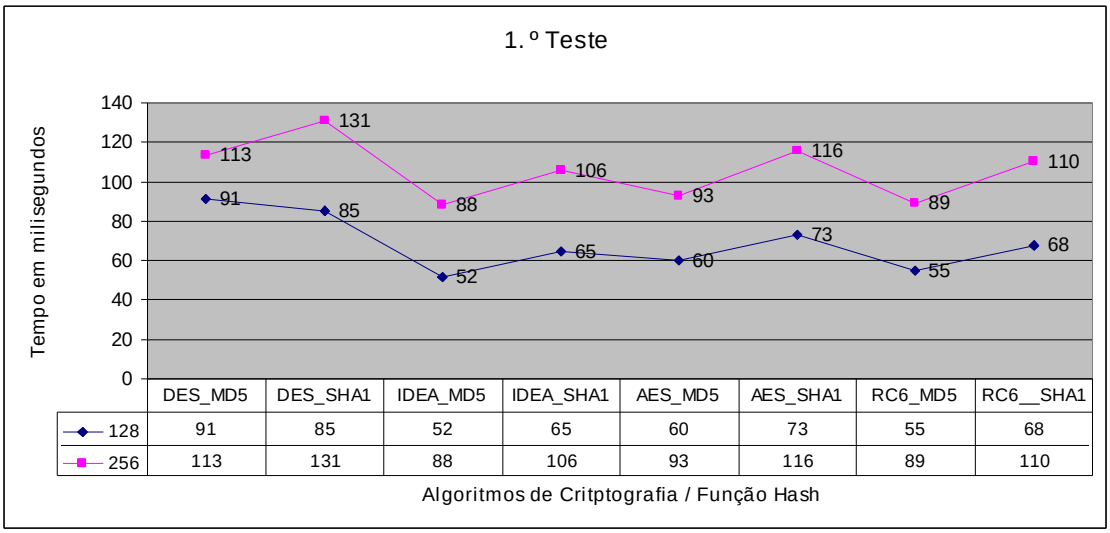
4.4.3 Funcionamento da aplicação de testes

Para executar o teste de desempenho é necessário escolher no menu principal a opção “Desempenho”, após exibida a tela de desempenho existe duas opções, sendo a primeira voltar ao menu anterior, ou seja, o menu principal, e a segunda executar, que executa o teste de desempenho que após calcular o tempo gasto para cifragem de um bloco de dados aleatório de tamanho x, com algoritmo de criptografia simétrica y, mais o algoritmo de função *hash* z, informará na tela do

celular o tempo gasto para cifrar o bloco, sendo que x pode ser 128 bytes e 4096 bytes e y pode ser DES, AES, IDEA e RC6, e z pode ser MD5 e SHA1. Nos testes abaixo descritos foram incluídos apenas blocos de 128 e 256 bytes para a analise.

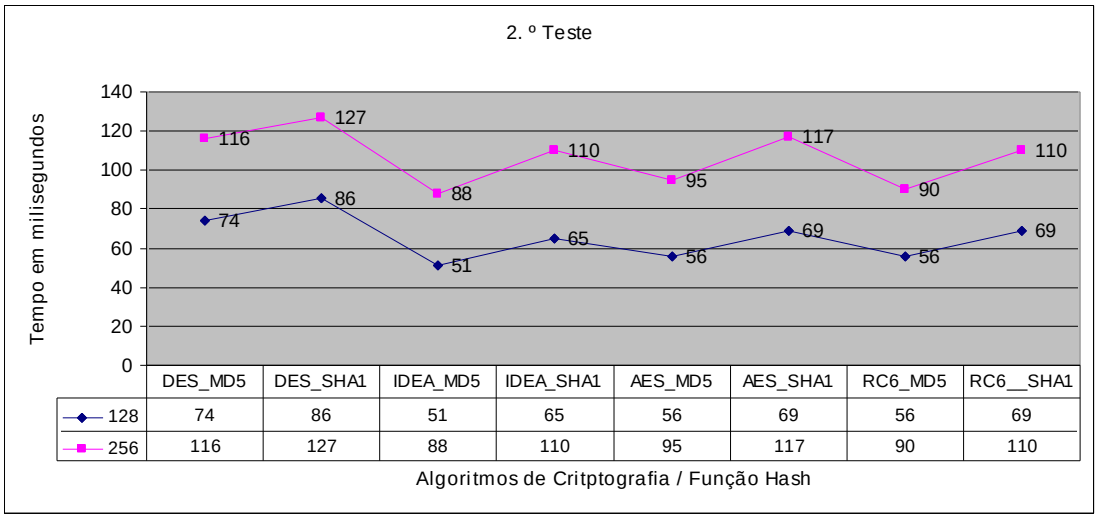
4.4.4 Medidas

Foram realizados três testes que estão ilustrados nas figuras Figura 42, Figura 43 e Figura 44, calculado a média aritmética dos três testes ilustrado na Figura 45 onde a média aritmética foi obtida dividindo-se a soma das observações pelo número delas, onde temos uma série de três valores sendo um de cada teste. E após os gráficos dos testes é feita uma analise do resultado obtido nos testes.



	DES_MD5	DES_SHA1	IDEA_MD5	IDEA_SHA1	AES_MD5	AES_SHA1	RC6_MD5	RC6__SHA1
128	91	85	52	65	60	73	55	68
256	113	131	88	106	93	116	89	110

Figura 42, Gráfico com o 1º Teste de cifragem de mensagem.



	DES_MD5	DES_SHA1	IDEA_MD5	IDEA_SHA1	AES_MD5	AES_SHA1	RC6_MD5	RC6__SHA1
128	74	86	51	65	56	69	56	69
256	116	127	88	110	95	117	90	110

Figura 43, Gráfico com o 2º Teste de cifragem de mensagem.

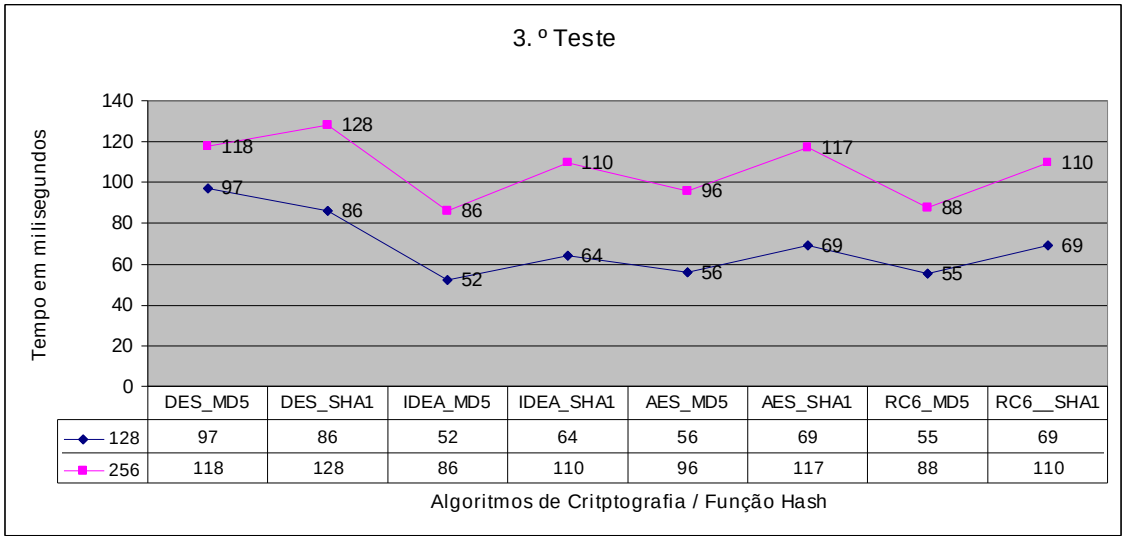


Figura 44, Gráfico com o 3º Teste de cifragem de mensagem.

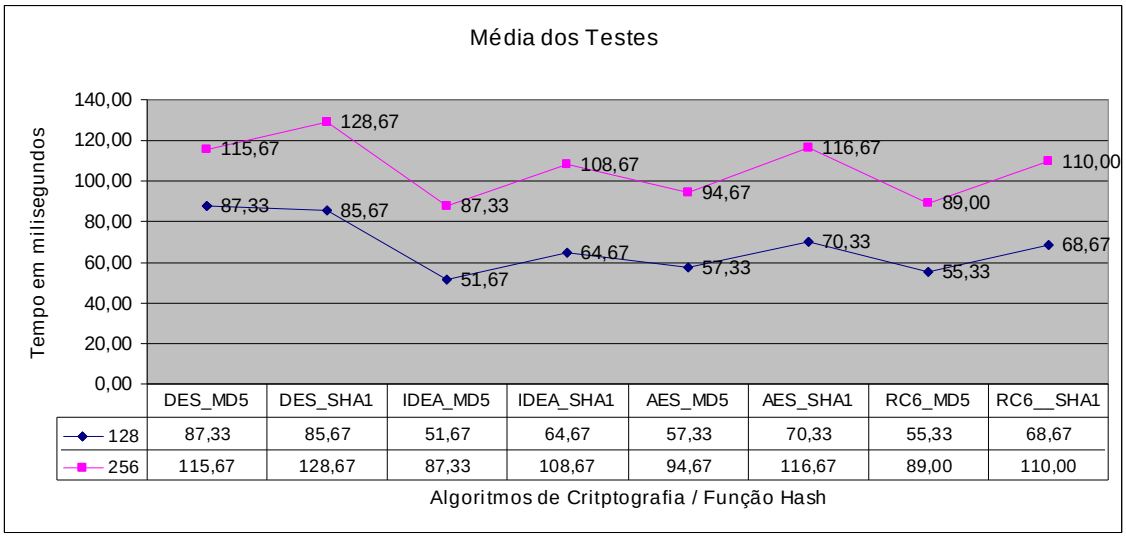


Figura 45, Gráfico com a média de tempo para cifrar mensagens.

Observando o gráfico da média ilustrado na Figura 45 pode-se verificar o desempenho dos algoritmos de criptografia simétrica DES, IDEA, AES e RC6 em combinação dos algoritmos de função *hash* MD5 e SHA1 no ambiente de testes descritos na seção 4.4.2 que a utilização de dos algoritmos de criptografia simétrica DES, IDEA, AES e RC6 com do algoritmo de função *hash* MD5 utiliza menos tempo de processamento do que utilizando o algoritmo de função *hash* SHA1.

Comparando os dados dos algoritmos de criptografia simétrica DES, IDEA, AES e RC6 quando utilizado o mesmo algoritmo de função *hash* MD5 ou SHA1, pode-se verificar que há uma diferença no tempo de processamento onde podemos observar a classificação abaixo do algoritmo de criptografia simétrica que utilizou

menos tempo de processamento para o algoritmo de criptografia simétrica que utilizou mais tempo de processamento:

- IDEA;
- RC6;
- AES;
- DES;

Analisando os dados dos testes realizados, pode-se observar que a diferença de tempo entre os algoritmos de criptografia simétrica em combinação com os algoritmos de função *hash* é pequena levando-se em conta que uma mensagem SMS pode ter no máximo 160 bytes.

5 Conclusão

Pelo estudo realizado, conclui-se a eficiência da utilização da tecnologia J2ME em conjunto com as bibliotecas do projeto Bouncy Castle, garantindo a confidencialidade e a integridade que com isso foi possível adicionar uma camada a mais de segurança das mensagens SMS trocadas entre dispositivos móveis.

Também foi possível verificar o desempenho da utilização de alguns algoritmos de criptografia simétrica e algoritmos de funções *hash* no qual se pode verificar que o tempo gasto para cifrar uma mensagem SMS que tem no máximo 160 bytes, o tempo levado para que estas informações fossem cifradas, foi pequeno, pois levou-se menos de um segundo.

Para que o estudo fosse concluído, foram abordados amplos assuntos, como telefonia celular digital, J2ME, criptografia simétrica, funções hash, bibliotecas do projeto Bouncy Castle, emuladores de dispositivos moveis, MIDway (programa de transferência de arquivos entre o computador e o dispositivo móvel), Sun Java Wireless Tool Kit 2.3, proguard3.5 (ofuscador de código) e sendo utilizando o IDE NetBeans 5.0 da SUN para o desenvolvimento do programa LRYEncrypt.

Ao decorrer do projeto foram surgindo novas idéias que não foram implementadas, como por exemplo:

- Melhoria na usabilidade do programa;
- Realizar testes em vários aparelhos celulares;
- Realizar teste de overhead que é gerado com informações cifradas;
- Utilização de outros algoritmos de criptografia simétrica;
- Utilização de outros algoritmos de funções hash;
- Utilização de MMS (Multimídia Message Service)
- Utilização de Criptografia de Chaves Publicas;

REFERENCIAS BIBLIOGRAFICAS

ALENCAR, Marcelo Sampaio de, Telefonia Celular Digital. São Paulo: Editora Érica, 2004.

BERNSTEIN, Bhimani, Schultz e Siegel.: Segurança na Internet. Ed. Campus, 1997.

GUELFI, Adilson Eduardo, Criptografia. São Paulo, 2005.

MUCHOW, John W. Core J2ME Tecnologia & MIDP. São Paulo: Makron Books, 2004.

SCHNEIER, Bruce. Applied Cryptography – protocols, algorithms and source code in C. New York. John Wiley, 1996.

TANENBAUM, Andrew. Redes de Computadores. Rio de Janeiro: Campus, 1996.

Bouncy Castle. <http://www.bouncycastle.org> Data de acesso em: 25 de março de 2005.

BARKAN Elad, BIHAM, Eli, KELLER, Nathan http://cryptome.org/gsm-crack-bbk.pdf. Data de acesso em: 08/03/2006.

Tecnologia Sun Java <http://java.sun.com> Data de acesso em: 10 de abril de 2005.

STANFORD. STANFORD UNIVERSITY, Desenvolvida por Dan Boneh e Matthew Frankliny Disponível em < http://crypto.stanford.edu/~dabo/papers/ibe.pdf>. Acesso em 10/06/2005.

Microsoft Disponível em <www.microsoft.com/windowsmobile/pocketpc/default.mspx> Acesso em 26/07/2005

Microsoft Disponível em <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/wcedsn40/html/cmconencryptdatausingcryptoapi.asp> Acesso em 26/07/2005

Microsoft Disponível em <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/wcedsn40/html/cmconencryptdatausingcryptoapi.asp> Acesso em 26/07/2005

Symbian Disponível em <http://www.symbian.com> 26/07/2005

PalmOS Disponível em <www.palmsource.com/palmos/> Acesso em 28/08/2005.

GSM Association Disponível em: http://www.gsmworld.com/index.shtml Acesso em: 29/03/2006.

RSA Laboratory Disponivel em: www.rsasecurity.com/rsalabs. Acesso em 30/03/2006.

ITI – Instituto Nacional de Tecnologia da Informação. Disponivel em: www.iti.gov.br. Acesso em: 15/03/2006.

BIBLIOGRAFIA RECOMENDADA

KAUFMAN, C., PERLMAN, R., SPECINER M., Network Security, Private Communication in a Public World. Prentice Hall, 1995.

FABRIN, A. P. D. P. Segurança e mobilidade na Internet. Dissertação (Mestrado). Instituto Tecnológico de Aeronáutica, São José dos Campos, 2001.

LUCCHESI, Cláudio Leonardo, Introdução à Criptografia Computacional. Campinas, SP: Papirus, 1986.

REZENDE, Pedro. <http://www.cic.unb.br/docentes/pedro/segdadtop.htm>. Data de acesso em: 12 de janeiro de 2005.

PESONEN, Lauri http://www.dia.unisa.it/professori/ads/corso-security/www/CORSO-9900/a5/Netsec/netsec.html. Data de acesso em: 08/03/2006.

MARGRAVE, David http://www.hackcanada.com/blackcrawl/cell/gsm/gsm-secur/gsm-secur.html. Data de acesso em: 08/03/2006.

WIKIPEDIA Disponível em http://en.wikipedia.org Acesso em: 17/03/2006

ANEXO 1 – Método gravaRegistro

```
public void gravaRegistro(Registro reg) {
try {
    // escreve dados no byte array interno
    ByteArrayOutputStream strmBytes = new ByteArrayOutputStream();
    // escreve tipos de dados Java no byte array descrito acima
    DataOutputStream strmDataType = new DataOutputStream(strmBytes);
    byte[] record;
    strmDataType.writeUTF(reg.getNumeroTel());
    strmDataType.writeUTF(reg.getAssunto());
    strmDataType.writeUTF(reg.getMensagem());
    strmDataType.writeInt(reg.getImagem());
    strmDataType.writeByte(reg.getCifra());
    strmDataType.writeByte(reg.getHash_Agoritmo());
    strmDataType.writeUTF(reg.getHash_Mensagem());
    strmDataType.flush();
    record = strmBytes.toByteArray();
    rs.addRecord(record, 0, record.length);
    strmBytes.reset();
    strmBytes.close();
    strmDataType.close();
    } catch (Exception e) {
      db(e.toString());
    }
}
```

ANEXO 2 – Método leRegistro

```
public void leRegistro() {
try {
     byte[] recData = new byte[450];
    // lengthde um byte array especifico
    ByteArrayInputStream strmBytes =
                        new ByteArrayInputStream(recData);
    // le tipos de dados Java do byte array
    DataInputStream strmDataType = new DataInputStream(strmBytes);
    // int aux = rs.getNumRecords();
    for (int i = 1; i <= rs.getNumRecords(); i++) {
      // Coloca o dado no byte array
              rs.getRecord(i, recData, 0);
               vReg.addElement( new Registro(strmDataType.readUTF(),
              strmDataType.readUTF(),
              strmDataType.readUTF(),
                strmDataType.readInt(),
                strmDataType.readByte(),
                strmDataType.readByte(),
                strmDataType.readUTF()));
                strmBytes.reset();
    }
    strmBytes.close();
    strmDataType.close();
    } catch (Exception e) {
        db(e.toString());
    }
}
```

ANEXO 3 – Método cifraMensagem

```
public static byte[] cifraMensagem(String plainText, String password,
            byte bAlgoCifra, byte bAlgoHash) throws Exception {
        byte conteudo[] = plainText.getBytes();
        byte chave[] = password.getBytes();
        BufferedBlockCipher cifraBuffer = new
PaddedBufferedBlockCipher(
                createEngine(bAlgoCifra));
        cifraBuffer.init(true, new KeyParameter(chave));
        byte[] cipherText = new
byte[cifraBuffer.getOutputSize(conteudo.length)];
        byte[] hash = null;
        int tamCipherText = cifraBuffer.processBytes(conteudo, 0,
                conteudo.length, cipherText, 0);
        cifraBuffer.doFinal(cipherText, tamCipherText);
        if (bAlgoHash != -1) {
            Digest objHash = createDigest(bAlgoHash);
            int tamHash = objHash.getDigestSize();
            hash = new byte[tamHash];
            objHash.update(conteudo, 0, conteudo.length);
            objHash.doFinal(hash, 0);
        }
        ByteArrayOutputStream out = new ByteArrayOutputStream();
        DataOutputStream dout = new DataOutputStream(out);
        dout.writeShort(cipherText.length);
        out.write(cipherText);
        if (bAlgoHash != -1) {
            out.write(hash);
        }
        return out.toByteArray();
    }
```

ANEXO 4 – Método decifraMensagem

```
public static String decifraMensagem(byte[] mensagem, String password,
            byte bAlgoCifra, byte bAlgoHash)
            throws Exception {
        byte chave[] = password.getBytes();
        ByteArrayInputStream inStream = new
ByteArrayInputStream(mensagem);
        DataInputStream dInStream = new DataInputStream(inStream);
        int tamCipherText = dInStream.readShort();
        byte[] cipherText = new byte[tamCipherText];
        inStream.read(cipherText, 0, tamCipherText);
        byte[] hash = null;
        Digest objHash = createDigest(bAlgoHash);
        if (bAlgoHash != 1) {
            int tamHash = objHash.getDigestSize();
            hash = new byte[tamHash];
            inStream.read(hash, 0, tamHash);
        }// if (bAlgoHash != 1) {
        BufferedBlockCipher cifraBuffer = new
PaddedBufferedBlockCipher(
                createEngine(bAlgoCifra));
        cifraBuffer.init(false, new
KeyParameter(password.getBytes()));
        byte[] plainText = new byte[cifraBuffer
                .getOutputSize(tamCipherText)];
        int tamCifraBuffer = cifraBuffer.processBytes(cipherText, 0,
tamCipherText,
                plainText, 0);
        cifraBuffer.doFinal(plainText, tamCifraBuffer);
        String mensagemRetorno = new String(plainText).trim();
        if (bAlgoHash != 1) {
            plainText = mensagemRetorno.getBytes();
            objHash.update(plainText, 0, plainText.length);
            byte[] bHashMensagem = new byte[objHash.getDigestSize()];
            objHash.doFinal(bHashMensagem, 0);
            if (bHashMensagem.length != hash.length) {
                throw new Exception(
                        "Mensagem não é integra(Tamanho do hash
diferente)!!!");
            }// if (bHashMensagem.length != hash.length) {
            for (int i = 0; i < bHashMensagem.length; i++) {
                if (bHashMensagem[i] != hash[i]) {
                    throw new Exception(
                            "Mensagem não é integra(HASH
diferente)!!!");
                }//if (bHashMensagem[i] != hash[i]) {
            }// for (int i = 0; i < bHashMensagem.length; i++)
        }// FINAL if (bAlgoHash != 1) {
        return mensagemRetorno;
    }// FINAL decifraMensagem;
```