CENTRO UNIVERSITÁRIO DE BRASÍLIA -UniCEUB

CURSO DE ENGENHARIA DE COMPUTAÇÃO

ARTHUR PRIETO COELHO

MECANISMO DE CRIPTOGRAFIA PARA COMUNICAÇÃO VIA BLUETOOTH

Orientador: Prof. Ms.C. Francisco Javier de Obaldía Díaz

Brasília

Junho, 2011

ARTHUR PRIETO COELHO

MECANISMO DE CRIPTOGRAFIA PARA COMUNICAÇÃO VIA BLUETOOTH

Trabalho apresentado ao Centro Universitário de Brasília (UniCEUB) como pré-requisito para a obtenção de Certificado de Conclusão de Curso de Engenharia de Computação.
Orientador: Prof. Ms.C. Francisco Javier de Obaldía Díaz

Brasília

Junho, 2011

ARTHUR PRIETO COELHO

MECANISMO DE CRIPTOGRAFIA PARA COMUNICAÇÃO VIA BLUETOOTH

Trabalho apresentado ao Centro Universitário de Brasília (UniCEUB) como pré-requisito para a obtenção de Certificado de Conclusão de Curso de Engenharia de Computação.
Orientador: Prof. Ms.C. Francisco Javier de Obaldía Díaz

Este Trabalho foi julgado adequado para a obtenção do Título de Engenheiro de Computação, e aprovado em sua forma final pela Faculdade de Tecnologia e Ciências Sociais Aplicadas - FATECS.

_____

Prof. Abiezer Amarilia Fernandez
Coordenador do Curso

Banca Examinadora:

_____

Prof. nome, titulação.
Orientador

_____

Prof. nome, titulação.
Instituição

_____

Prof. nome, titulação.
Instituição

_____

Prof. nome, titulação.
Instituição

AGRADECIMENTOS

Aos meus pais, Marcelo e Jussara, pelo apoio incondicional em todas os momentos, principalmente nos mais difíceis.

À minha irmã Júlia, pelo companheirismo e cumplicidade em todas as horas.

À Solange, muito mais do que qualquer um poderia esperar de uma madrasta.

Aos meus avós, Cybele, Irma e Arnaldo, tios e tias, primos e primas, e demais familiares que me acompanharam nessa jornada, e sempre me acolheram bem.

Ao professor Javier, pela ajuda e orientação no decorrer no projeto.

Ao professor Marco Antônio, pelo acompanhamento no começo do projeto.

Ao amigo Thiago Rider, pela imensurável ajuda no desenvolvimento desse projeto.

À todos que, de alguma forma, ajudaram na confecção e desenvolvimento deste projeto.

À todos os que passaram comigo pelas aulas do curso, pelos bons momentos que sempre passamos juntos.

Por último, mas não menos importante, a todos os amigos que vieram comigo nessa jornada, e continuarão ao longo de outras.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

Um dos grandes problemas encontrados nos dias de hoje para a comunicação via celulares é a vulnerabilidade do protocolo Bluetooth à intrusão de pessoas com acessos indesejados, que podem capturar a informação enquanto ela é transmitida. Com a melhoria da tecnologia Bluetooth, para o alcance de maiores distâncias, esse problema pode aumentar. Uma solução seria a implementação de um algoritmo de criptografia. Ao se utilizar o mesmo na mensagem a ser enviada, o intruso não teria acesso à mensagem original. Este trabalho visa realizar a implementação de um mecanismo de criptografia para este tipo de comunicação para aparelhos móveis que utilizem o sistema operacional Android, visando aumentar a segurança das transferências de dados e informações realizadas através deste protocolo nestes aparelhos. Será utilizado um algoritmo já existente de chaves simétricas, o DES. Especificamente, o projeto tem como objetivo a segurança na troca de mensagens via Bluetooth. Esta segurança será provada através de tentativas de captura da mensagem, mostrando que a mesma está criptografada.

Palavras Chave: Bluetooth, Criptografia, Segurança, DES

ABSTRACT

One of the great problems found these days for mobile phone communication is the vulnerability of the Bluetooth protocol to the intrusion of unwanted people, who can capture the information while it is transferred. With the improvement of the technology, allowing for bigger distances, this problem may increase. A solution would be the implementation of a cryptography algorithm. When using it in a message to be sent, the intruder wouldn't have access to the original data. This project aims to do the implementation of a cryptography mechanism for this kind of communication, for mobile devices that use the Android operational system, aiming to increase the security of data and information transfers done through this protocol, for these devices. An already existing algorithm of symmetric keys, DES, will be used. Specifically, this project has as a goal the security in Bluetooth communication. This security will be shown through attempts of capturing the message, showing that it is encrypted.

Keywords: Bluetooth, Cryptography, Security, DES

CAPÍTULO 1 - INTRODUÇÃO

1.1 – Apresentação do Problema

Nos dias atuais, um dos grandes problemas enfrentados no mundo da informática é sobre como tratar da segurança das informações. Isto inclui, mas não está limitado a, transferência de arquivos, em suas diversas formas. Um protocolo que vem crescendo cada vez mais devido ao seu uso em telefones móveis é o Bluetooth.

O Bluetooth é projetado para baixo consumo de energia e baixo alcance, baseado em microchips transmissores de baixo custo em cada dispositivo. A sua própria natureza define a sua utilização principal em aparelhos celulares, visto que as suas características básicas são exatamente aquelas necessárias para a popularização de um dispositivo na telefonia móvel.

Com a popularização do protocolo e com todos os aparelhos móveis modernos possuindo a opção de envio e recebimento de arquivos por este meio, a utilização do Bluetooth cresce cada vez mais. Até os mesmo os novos computadores, tablets e notebooks para uso corporativo vêm sendo lançados com o protocolo integrado.

Com o uso do Bluetooth aumentando cada vez mais, tanto para uso doméstico quanto para uso corporativo, a segurança no protocolo se torna um fator de risco. Caso as informações não sejam devidamente protegidas de usuários externos, elas podem correr risco de quebra de confidencialidade (veremos mais sobre os pilares da Segurança da Informação no próximo capítulo).

Muitas vezes, os usuários do Bluetooth não sabem que existe risco na utilização do protocolo, assim como existe em qualquer outra forma de comunicação. O alcance pode chegar a até 100 metros, dependendo da classe do protocolo em utilização. Um intruso pode extrair as informações transferidas com uso de um dispositivo pessoal, simples e que pode parecer inofensivo, como um notebook.

Desta forma, o protocolo precisa de segurança para que as informações enviadas através dele sejam ilegíveis no caso de um usuário externo capturar a mensagem que está sendo enviada durante a sua transferência. Como tornar a transferência de dados e arquivos via Bluetooth mais segura para os seus usuários?

1.2 – Objetivos do Trabalho

Objetivo Geral – Aumentar a segurança dos dados e informações trafegados através de comunicação via Bluetooth com uso de criptografia.

Objetivos Específicos – Será especificado, desenvolvido e implementado uma forma de aplicar criptografia simétrica em mensagens trocadas via Bluetooth. Com isto, o remetente

da mensagem irá realizar o processo de cifragem da mesma antes de enviá-la. O destinatário, por sua vez, irá decifrar a mensagem para ter acesso à mensagem original. Não existem hoje softwares amplamente difundidos que façam isso para os aparelhos Android; Será criado um software que irá realizar a cifragem e a decifragem da mensagem. Esta mensagem será criptografada antes de ser enviada pelo aparelho e, quando chegar a seu destino, será decifrada por seu recebedor; Será estudada também a questão de impacto no desempenho com a cifragem aplicada e não aplicada, mostrando a variação na performance da transferência por completo da mensagem.

1.3 – Justificativa e Importância do Trabalho

Esta situação se torna um problema devido ao constante aumento do uso do Bluetooth para transferência de informações. Com isto, é possível que informações confidenciais ou pessoais sejam transferidas para uma outra pessoa. Caso esta comunicação seja interceptada, as informações poderão estar em risco.

Desta forma, se torna necessário tornar esta comunicação mais segura, de forma a permitir a troca de informações sem que haja preocupação com a segurança da transmissão.

1.4 – Escopo do Trabalho

Será implementado um mecanismo que, ao verificar que haverá o envio ou recebimento de uma mensagem Bluetooth, irá realizar a cifragem da mensagem no remetente e a decifragem da mensagem no destinatário, através de um algoritmo de chaves simétricas, o DES. Será estudada também a questão de impacto no desempenho com a cifragem aplicada e não aplicada, mostrando a variação na performance da transferência por completo da mensagem.

1.5 – Resultados Esperados

O mecanismo de criptografia criado deverá funcionar da forma correta, tendo sua inviolabilidade demonstrada através de ataques à rede de comunicação.

1.6 – Estrutura do Trabalho

O primeiro capítulo, que se encerra nesse item, aborda uma introdução ao projeto, apresentando o problema, citando os objetivos, justificativa, importância e escopo do trabalho, assim como os resultados esperados e a estrutura presente no projeto.

O segundo capítulo irá apresentar um referencial teórico do projeto. Começando pela Segurança da Informação, com definição, objetivo e o motivo de sua necessidade. Em

seguida, trataremos de criptografia, com a sua definição e os métodos utilizados para implantação do processo, nos aprofundando no método de chaves simétricas.

Após isso, será comentada a telefonia móvel: definições, vulnerabilidades da comunicação móvel e um aprofundamento no aparelhos móveis que funcionam com sistema operacional Android, que serão utilizados para implementação desse projeto, apresentando a série e mostrando como se dá o desenvolvimento de aplicativos para esses aparelhos.

No terceiro capítulo, veremos o protocolo Bluetooth, com a sua definição e as aplicações do mesmo. Veremos ainda as vulnerabilidades do protocolo Bluetooth.

No quarto capítulo, será mostrada a proposta de segurança para o Bluetooth, mostrando o modelo proposto e a aplicação do mesmo. Será feita a apresentação do modelo e descritas suas etapas e implementação. Na aplicação, será apresentada a área de aplicação do modelo, seguida pela sua descrição. Será feita ainda uma avaliação global do modelo.

CAPÍTULO 2 – CRIPTOGRAFIA E ASPECTOS DE SEGURANÇA DA INFORMAÇÃO EM COMUNICAÇÕES MÓVEIS

Este capítulo apresenta os principais conceitos sobre criptografia e segurança da informação com aplicações em telefonia móvel e redes sem fios, sendo que a transmissão com uso da tecnologia bluetooth pode beneficiar-se de metodologias de segurança como propostas no presente trabalho.

2.1 – SEGURANÇA DA INFORMAÇÃO

2.1.1 – Definição de Segurança da Informação

A segurança da informação tem como objetivo proteger a informação de qualquer perigo que ela possa correr. Isto inclui risco de roubo da informação, indisponibilidade da mesma por qualquer motivo ou ainda o risco de ela não esteja íntegra. A informação é um ativo que, possui um valor para o seu dono, seja ele uma pessoa, uma pequena empresa ou uma grande corporação. Desta forma, ela deve ser devidamente protegida. Conforme definido por Fernando Nicolau Freitas Ferreira (Nicolau, 2003, p.1), "a segurança da informação protege a informação de diversos tipos de ameaças garantindo a continuidade dos negócios, minimizando os danos e maximizando o retorno dos investimentos e das oportunidades."

Nos dias de hoje, em que as informações são os mais importantes patrimônios de uma organização, elas estão também sob um risco em que nunca estiveram antes. Desta forma, a segurança da informação tornou-se crítica para a manutenção dos negócios de uma empresa ou organização.

Quando a informação existia apenas em papel, a segurança era simples, limitava-se à proteção física dos ativos. Isto ainda é necessário, incluindo acesso restrito e proteção contra desastres naturais, falhas estruturais, sabotagens e fraudes. Neste caso, costuma-se fazer um *backup* das informações, e guardá-lo em um outro local.

Hoje, com o número de computadores pessoais aumentando exponencialmente e com as redes ficando cada vez mais abrangentes, cresce a necessidade da segurança lógica das informações. Desta forma, torna-se obrigatório ter uma equipe dedicada à implementação e gerenciamento da segurança das informações. A segurança da informação possui três pilares em que ela se sustenta:

a) Confidencialidade: garantia de que a informação é acessível somente por pessoas autorizadas;
b) Integridade: salvaguarda da exatidão da informação e dos métodos de processamento;
c) Disponibilidade: garantia de que os usuários autorizados obtenham acesso à informação e aos ativos correspondentes sempre que necessário. (FERREIRA, Fernando Nicolau Freitas. Segurança da Informação, Rio de Janeiro, 2003, p.2)

Alguns autores definem ainda mais dois pilares: o não repúdio e a autenticidade "compreendem o que poderia ser denominado de responsabilidade final e, dessa forma, busca-se fazer a verificação da identidade e autenticidade de uma pessoa ou agente externo de um sistema a fim de assegurar a integridade de origem." (Antonio Mendes Silva Filho, 2004, em http://www.espacoacademico.com.br/042/42amsf.htm)

A quebra de um desses pilares pode comprometer sensivelmente a segurança da informação. Caso haja perda de confidencialidade, a informação pode cair na mão de um concorrente, que certamente saberá usá-la a seu favor. Se houver a perda de integridade, a informação deixa de ser confiável, pois estará corrompida ou, na pior das hipóteses, terá sido deletada. Caso haja perda de disponibilidade, pode-se perder uma oportunidade para fechar um negócio ou atender uma necessidade de um cliente. No último caso, o da perda de não repúdio e autenticidade, a situação pode ser ainda pior porque a informação pode ser não ser proveniente do local do qual se acredita que ela veio, havendo uma fraude no processo. Um intruso pode enviar uma informação incorreta, que poderá gerar uma tomada de decisões incorreta. A figura 2.1 mostra como esse processo pode acontecer.

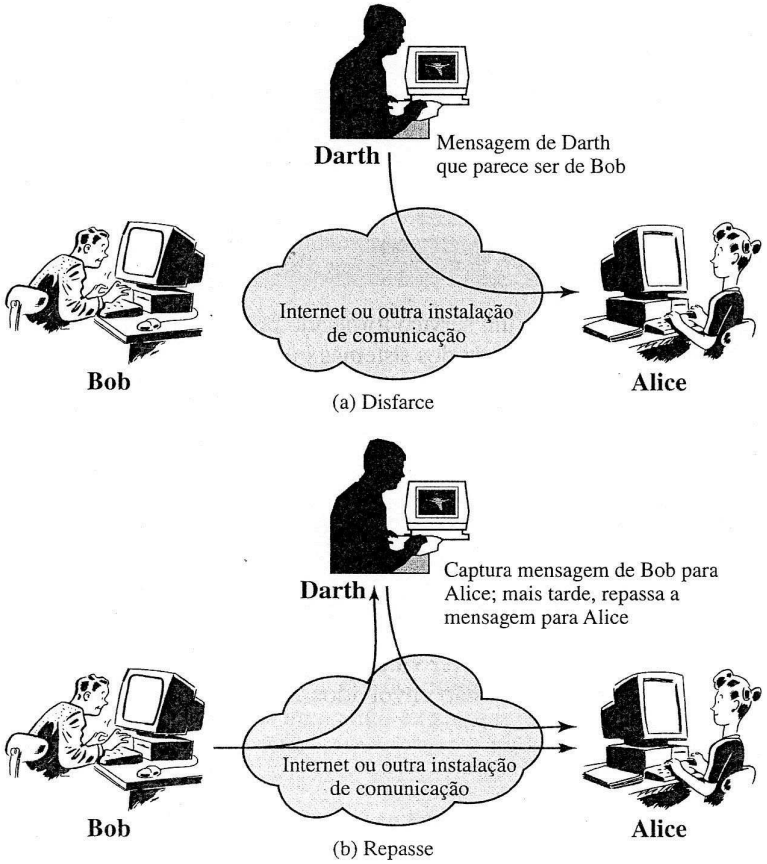

Figura 2.1: Como um intruso pode agir se tiver acesso à comunicação entre dois pontos.

Fonte: Stallings, 2003, 4ª Ed.

2.1.2. – Objetivos da Segurança da Informação

Para podermos implementar a segurança da informação, precisamos definir a área de atuação do mecanismo de proteção. Para tal, devemos responder às seguintes perguntas:

- O que devemos proteger?
- Contra que ou quem?
- Quais as ameaças mais prováveis?
- Qual a importância de cada recurso?
- Qual o grau de proteção desejado?
- Quanto tempo, recursos humanos e financeiros pretendemos gastar para atingirmos os objetivos de segurança desejados?
- Quais as expectativas dos usuários e clientes em relação à segurança das informações?
- Quais as consequências para a organização se os sistemas e informações forem corrompidos ou roubados? (FERREIRA, Fernando Nicolau Freitas. Segurança da Informação, Rio de Janeiro, 2003, p.3)

Com essas respostas em mãos, é possível definir os objetivos e requisitos de segurança da informação que a organização espera que sejam cumpridos após a aplicação da política de segurança.

2.1.3. – Necessidade da Segurança da Informação

A informação, assim como os sistemas, redes e processos de apoio, são ativos de grande importância para uma organização. Os cinco pilares devem ser garantidos, pois são essenciais para manter a competitividade e a lucratividade de um negócio, perante ao mercado e aos clientes. A imagem da empresa deve ser mantida intacta nesse aspecto. Caso haja suspeita de que a organização não é mais segura, haverá desconfiança e a instituição deverá perder negócios.

A cada dia que passa, os sistemas de informação e redes de computadores vêm sendo mais atacados, explorados e testados pelos mais diversos tipos de ameaças, das mais diversas origens. Essas ameaças incluem, mas não estão limitadas a, fraudes eletrônicas, espionagem, sabotagem e vandalismo. Os vírus, hackers e ataques estão cada vez mais complexos, ambicionando cada vez mais, e se tornando cada vez mais comuns.

A dependência crescente dos sistemas de informação e serviços faz com que as organizações estejam cada vez mais vulneráveis à ataques e ameaças de segurança. O crescimento da rede mundial, a conexão das redes internas com as externas e o compartilhamento de recursos aumentam a dificuldade em se controlar o acesso.

A segurança da informação torna-se cada vez mais crítica para o sucesso de uma organização, em um período que mantê-la segura é cada vez mais complicado. Uma das

formas de proteger a informação quando ela é transmitida é por meio da criptografia, que será discutida a seguir.

2.2 – CRIPTOGRAFIA

2.2.1. – Definição de Criptografia

Criptografia é a arte ou ciência de codificar informações, de modo a esconder o conteúdo original de pessoas indesejadas, permitindo que somente o destinatário da mensagem consiga lê-la e compreendê-la de forma íntegra. Em outros termos, a criptografia transforma um texto original, também conhecido como *plaintext* (texto claro), em um texto cifrado, ou *cyphertext* (texto cifrado), que irá parecer informação randômica e ilegível. Para tal, é usada uma técnica de encriptação, chamada de cifra ou sistema criptográfico. O processo de converter o texto claro no texto cifrado é chamado de cifração ou encriptação. O caminho contrário recebe o nome de decifração ou decriptação. (Nicolau, 2003)

De acordo com o dicionário Michaelis, a criptografia é "a arte ou processo de escrita em caracteres secretos ou em cifras." Desde a Antiguidade se conhecia a arte da cifragem, onde se realizava a substituição ou troca dos símbolos com o objetivo de confundir um possível interceptador das mensagens. Na computação, esse princípio é mantido. Porém, com a capacidade de processamento de dados existente, a criptografia foi levada a um outro nível de complexidade e segurança. Existem métodos de criptografia que são tão robustos e complexos que fazem com que a tentativa de quebra das chaves não compense o tempo e o esforço computacional que deverá ser despendido na operação. A tabela 2.1 mostra uma estimativa de tempo que um invasor levaria para quebrar chaves por meio do ataque de força bruta em função da quantidade de bits utilizados na chave:

Tabela 2.1: Estimativa de tempo para quebra de chave por ataque de força bruta

Tamanho da Chave (bits)	Atacante Individual	Pequeno Grupo	Rede Acadêmica	Grande Corporação	Agência de Inteligência Militar
40	Semanas	Dias	Horas	Milissegundos	Microssegundos
56	Séculos	Décadas	Anos	Horas	Segundos
64	Milênios	Séculos	Décadas	Dias	Minutos
80	Inviável	Inviável	Inviável	Séculos	Séculos
128	Inviável	Inviável	Inviável	Inviável	Milênios

Fonte: Nicolau, 2003

2.2.2. – Princípios Básicos

Dos princípios básicos da segurança, dois deles podem ser mantidos com auxílio da criptografia: a autenticidade e a confidencialidade.

A autenticidade é definida pela identificação correta de um usuário ou máquina, O serviço de autenticação implementado deve ser capaz de verificar que a mensagem é realmente originada no local informado em seu conteúdo. Esse processo geralmente é realizado através de um controle de senhas ou por assinatura digital. A autenticidade funciona como medida de proteção contra a personificação por intrusos, e é necessária em todas as etapas do processo de identificação: usuário para sistema, sistema para usuário e sistema para sistema. Uma maneira comum de personificação ocorre quando um usuário externo toma a identidade de um usuário interno, se passando por ele dentro do sistema. Isto pode ser feito com o objetivo de conseguir novos usuários e senhas internos do sistema até que o invasor ache uma combinação que lhe permita o acesso completo.

A confidencialidade é caracterizada pela proteção de informações contra o seu conhecimento por pessoas não autorizadas. Significa proteger a informação contra distribuição para alguém que não tenha permissão do proprietário da informação. Deve-se proteger não apenas a informação enquanto de posse dos usuários, mas também no transporte da mesma por uma rede, evitando que ela seja capturada da rede por um mecanismo ilícito. Isso é feito evitando-se a escuta (meio físico, topologia) e a inteligibilidade dos dados durante a transmissão (criptografia). Um método muito utilizado para captura de informações em uma rede é o uso de *sniffers*. As informações capturadas por *sniffers* podem ser sensíveis (e muitas vezes o são), e podem, futuramente, permitir um ataque por autenticidade, por meio do uso de senhas e outras informações capturadas na rede.

O uso da criptografia irá proteger a informação de visualização alheia, deter alterações nos dados e identificar usuários, mas não irá impedir de forma alguma que a informação seja apagada. (Oliveira, 2001)

2.2.3. – Modelos de Criptografia

Existem dois modelos de criptografia utilizados no dia de hoje: criptografia por chaves simétricas e por chaves assimétricas. (Nicolau, 2003)

Neste trabalho, será utilizada a criptografia por chaves simétricas, cujo método será exposto a seguir. Maiores detalhes sobre esse tipo de criptografia podem ser encontrados em (Stallings, 2003) e (Buchmann, 2002).

2.2.3.1. – Criptografia Simétrica

O método de criptografia simétrica utiliza uma chave única tanto para criptografar como para decriptografar uma mensagem, ou seja, caso se use este método para troca de mensagens, a mesma chave deverá ser usada tanto para criptografar a mensagem quanto para decriptografá-la. Esta chave deve ser mantida em segredo para garantir a eficiência do algoritmo.

Os modelos de criptografia por chaves simétricas são eficientes e de difícil decifração. As chaves consideradas seguras nestes métodos devem possuir pelo menos 128 bits de comprimento.

Uma cifra simétrica possui cinco elementos, relatados abaixo e retratados na figura 2.2:

- Texto claro – A mensagem original, em texto entendível, ou a entrada de um algoritmo;
- Algoritmo de encriptação – Este algoritmo faz diversas substituições e transformações no texto claro;
- Chave secreta – A chave também é entrada do algoritmo de encriptação, porém independente do texto claro. O algoritmo irá gerar uma saída diferente dependendo da chave usada no momento, e a substituições e transformações feitas em cima do texto também irão depender da chave.
- Texto cifrado – É mensagem gerada como saída do algoritmo. Ele é função do texto claro e da chave secreta. Para uma mesma mensagem de entrada, duas chaves diferentes irão produzir duas saídas diferentes. O texto cifrado parece um texto randômico e ininteligível.
- Algoritmo de decriptação – É o algoritmo de encriptação rodado ao inverso. Precisa-se do texto cifrado e da chave para conseguir-se o texto claro novamente.

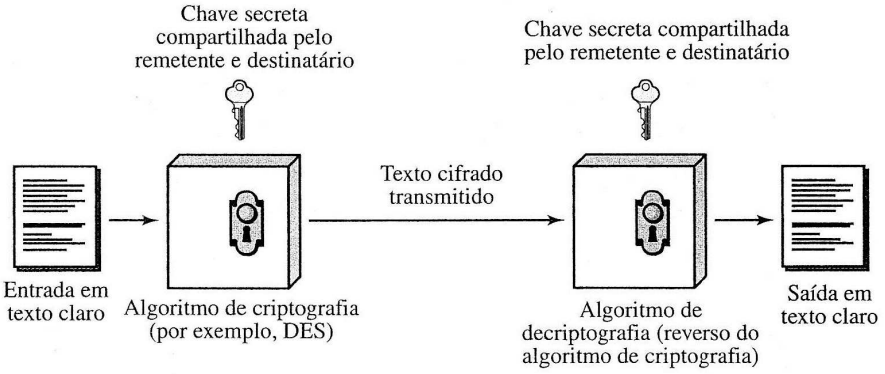

Figura 2.2: Modelo simplificado da criptografia simétrica.

Fonte: Stallings, 2003, 4ª Ed.

Existem dois requerimentos que devem ser alcançados para o uso com sucesso de encriptação simétrica:

- É necessário um algoritmo de encriptação forte. O ideal é que o algoritmo seja tal que, se um intruso souber o algoritmo e obtiver posse de alguns textos cifrados, ele não consiga nem decifrar os textos, nem descobrir a chave. Esse requerimento pode ser colocado de forma mais forte: O intruso que obtiver posse de alguns textos cifrados junto com os textos claros que originaram cada um, não deve conseguir decifrar outros textos cifrados ou descobrir a chave.
- Tanto o remetente como o destinatário devem ter obtido posse da chave de forma segura e devem manter a chave em segurança. Caso a chave seja descoberta e o intruso saiba o algoritmo, qualquer comunicação utilizando essa chave é legível.

É tomado como princípio o fato de que não é prático decriptar uma mensagem estando de posse apenas do texto cifrado e do algoritmo de encriptação utilizado. Isto é, não é preciso manter o algoritmo em segredo; apenas a chave. É essa característica dos algoritmos simétricos que viabilizam o seu uso em grande escala, pois permite que fabricantes desenvolvam suas soluções de encriptação de dados com um baixo custo, e coloquem-nas em seus produtos. O único problema passar a ser a segurança de chave, que garante o algoritmo.

Todo esse processo é mostrado matematicamente da seguinte forma: Uma fonte produz uma mensagem em texto claro, $X = [X_1, X_2, ..., X_M]$. Os M elementos de X são letras em algum alfabeto finito, normalmente as 26 letras tradicionais em maiúscula. Hoje em dia, também se usa o alfabeto binário {0, 1}. Para fazer a encriptação, é gerada uma chave K =

[K_1, K_2, ..., K_J]. Se a chave for gerada pela origem, ela deve ser transmitida de forma segura ao destino. A chave também pode ser gerada por um terceiro que a provém para origem e destino da mensagem.

Com a mensagem X e a chave K como entrada, é realizada a encriptação de forma a gerar o texto cifrado Y = [Y_1, Y_2, ..., Y_N]. A fórmula para a encriptação pode ser descrita da seguinte forma:

$$Y = E_K(X) \quad \text{(Eq. 2.1)}$$

Essa fórmula descreve o texto cifrado Y sendo gerado pelo algoritmo de encriptação E sendo aplicado em cima do texto claro X, com a função específica sendo determinada pela chave K.

Uma vez de posse da mensagem, o destinatário irá realizar a transformação inversa, conforme mostrado abaixo:

$$X = D_K(Y) \quad \text{(Eq. 2.2)}$$

Um intruso, sabendo Y mas não tendo acesso a X ou K, pode tentar recuperar o texto claro, a chave ou até mesmo os dois. Para tal, é assumido que o intruso sabe os algoritmos de encriptação e decriptação. Se o objetivo do intruso for recuperar uma mensagem específica, ele irá focar em descobrir X, gerando uma estimativa do texto claro. No entanto, se o objetivo for recuperar também mensagens futuras, o intruso irá concentrar seus esforços em descobrir a chave K, gerando uma estimativa da mesma.

O modelo de criptografia simétrica é classificado em três famílias: a criptografia simétrica de blocos (*Block Cipher*), a criptografia simétrica de via (*Stream Cipher*) e a criptografia simétrica de resumo (*Hash Functions*).

A criptografia simétrica foi a mais utilizada ao longo da história. Ela permite a implementação em diversos dispositivos, dos mais variados tipos: manuais, mecânicos, elétricos e nos algoritmos atuais, utilizáveis em qualquer computador. A criptografia é aplicada através do uso de diferentes funções sobre a mensagem que se deseja criptografar, de forma que só se possa reverter o processo com o conhecimento da chave.

Este modelo é ainda subdividido em algumas técnicas de criptografia, que serão abordados brevemente a seguir.

2.2.3.1.1 – Técnica de Substituição

Esta técnica consiste em fazer, de forma simples, a substituição de uma letra do alfabeto por outra. A primeira pessoa que se tem registro a usar este tipo de cifragem foi o general romano Júlio César. Na cifra criada por ele, cada letra era substituída pela terceira

subseqüente no alfabeto. Desta forma, o A era trocado por D, o B por E, e assim por diante. O alfabeto se torna cíclico, ou seja, a letra seguinte à Z é o A. Eventualmente, essa cifra evoluiu para se usar uma outra substituição, mantendo o alfabeto em ordem. Hoje, essa cifra é facilmente quebrada por um ataque de força bruta. No entanto, na época de sua primeira utilização, não existe registro de que ela tenha sido quebrada. Isso pode ter ocorrido simplesmente porque um intruso acreditaria estar diante de um alfabeto desconhecido, logo não poderia entender a mensagem. (Stallings, 2003)

A cifra de César evoluiu para a cifra monoalfabética. Essa cifra também pode ser quebrada facilmente, caso se conheça a origem do texto claro (texto em um idioma conhecido, por exemplo). Isso ocorre porque pode-se analisar as regularidades do idioma, como letras e palavras mais comuns.

Eventualmente, foram criadas cifras mais bem trabalhadas, como a de Playfair, a de Hill e a polialfábetica. Em todas já existe a introdução de uma chave, que será usada para a geração do texto cifrado. Com isto, a segurança aumenta, apesar de ainda não ser a ideal, já que a substituição é sempre feita a partir de um símbolo de texto claro para um de texto cifrado.

2.2.3.2.2 – Técnica de Transposição

Nesta técnica de criptografia simétrica, o mapeamento realizado para troca de símbolos é diferente. Ao invés de existir uma tabela ou relacionamento que permita ao utilizador sair de um símbolo de texto claro para um de texto cifrado, é feita uma transposição entre os elementos da mensagem em texto claro. Neste caso, organiza-se a mensagem em colunas, utilizando quantas linhas forem necessárias. Por exemplo, a frase “vamos para vancouver” poderia ser mostrada da seguinte forma:

```
voavcv
asraoe
mpanur
```

E seria escrita assim: voavcvasraoempanur. No entanto, criptografar essa mensagem seria um tanto quanto trivial. Pode-se, no entanto, reordenar as colunas. Com isso, a ordem das mesmas passa a ser a chave do algoritmo. Por exemplo:

```
531642
vamosp
aravan
couver
```

O texto cifrado passaria a ser: maupnrarosaevacovv. No entanto, ainda seria relativamente fácil quebrar essa cifra. Bastaria colocar o texto cifrado em uma matriz e tentar reordenar as colunas. Essa cifra passa a ser bem mais eficiente se for realizada mais de uma rodada de transposição. O resultado passa a ser uma permutação mais complexa, não tão simples de ser decriptada. Basta manter-se a mesma chave (ordem de colunas) e colocar o texto cifrado na matriz para ser encriptado novamente. Este princípio se aplica também à técnica de substituição. (Stallings, 2003)

2.2.3.2.3 – Máquinas de rotor

O princípio da máquina de rotor tornou-se comum no início do século passado, e foi usado para a criação das máquinas Enigma (Alemã) e Purple (Japonesa), utilizadas durante a Segunda Guerra Mundial. Os códigos foram eventualmente quebrados pelos Aliados, o que teve grande influência no resultado do confronto.

O conceito é o seguinte: a máquina possui uma série de cilindros rotatórios independentes, através dos quais pulsos elétricos podem ser transmitidos. Cada cilindro possui 26 pinos de entrada e 26 pinos de saída, sendo cada pino do primeiro conjunto é ligado em um único pino do segundo. Com isto, uma solução com um cilindro se tornaria uma aplicação da cifra monoalfabética, e se tornaria de relativamente fácil solução. A força do mecanismo de máquinas rotores está no uso de múltiplos cilindros. A cada letra digitada, o primeiro cilindro anda uma posição. Quando o primeiro cilindro der volta completa, ou seja, andar as 26 posições, o segundo cilindro move uma posição. Quando o segundo cilindro der a volta, o terceiro anda uma posição. E segue-se assim, para quantos cilindros houver. Conforme definido em [Kahn, 1996, p. 413], uma solução com 5 cilindros seria suficiente para tornar inviável uma tentativa de quebrar a cifra por força bruta.

Esta técnica de criptografia se tornou tão importante que virou a base para o mais utilizado algoritmo de criptografia utilizado na história: o DES (*Data Encryption Standard*). A figura 2.3 mostra o funcionamento da máquina rotor.

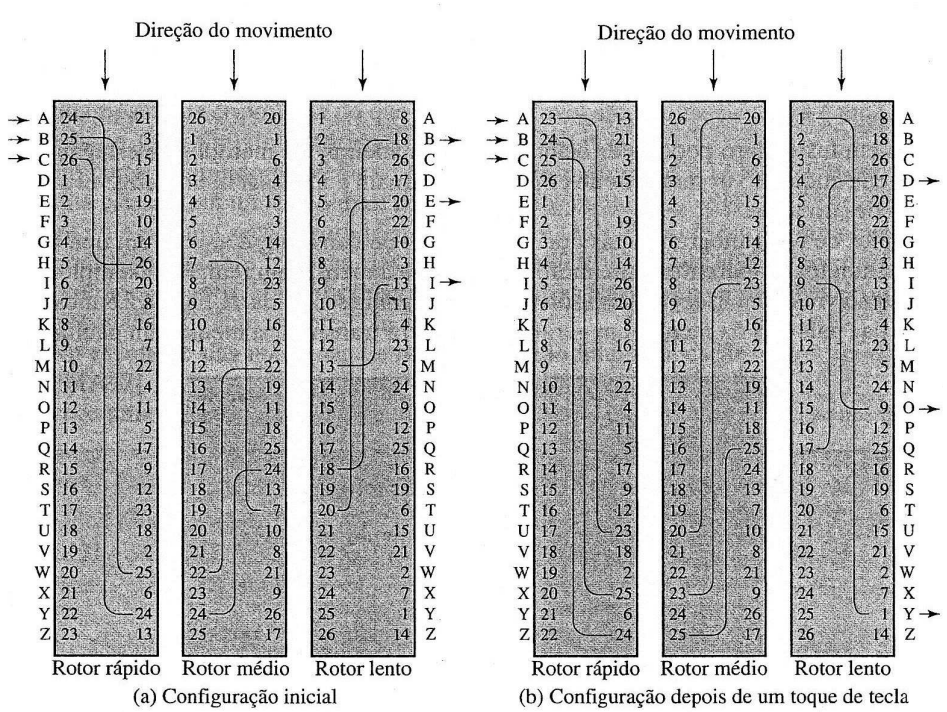

Figura 2.3: Modelo de uma máquina de três rotores

Fonte: Stallings, 2003, 4ª Ed.

2.2.3.2.4 – Esteganografia

A esteganografia trata-se de uma forma diferente de criptografia, pois ao invés de diretamente criptografar a mensagem, o processo é o de escondê-la, por exemplo, dentro de um outro texto. Pode-se escrever um texto inofensivo no qual a primeira letra de cada palavra forma o texto que realmente se quer passar. Ou então, escrever o texto com tinta invisível. Ou ainda, marcar as letras relevantes de uma forma pré-combinada.

O método mais comum de uso da criptografia simétrica é a cifra de *Feistel*, que consiste em aplicar um número finito de iterações de forma que, ao final do processo, se tenha a mensagem codificada como resultado. É assim que funcionam o DES e o Blowfish, entre outros, porém com algumas modificações para aumento de segurança.

2.2.3.2. – Cifras de Bloco

Uma cifra de bloco é um método de encriptação e decriptação em que um pedaço de texto claro é tratado de uma vez e usado para gerar um texto cifrado de igual tamanho. Geralmente são usados blocos de 64 ou 128 bits.

Várias cifras de bloco utilizam a estrutura de Feistel para realizar a criptografia. Tal estrutura consiste em se executar uma mesma rodada de processamento repetidas vezes, de forma idêntica em cada iteração. Em cada rodada, é efetuada uma substituição em uma das

metades do texto que está sendo cifrado, e em seguida é feita uma permutação com a outra metade. A chave original é constantemente expandida, para que uma chave diferente seja usada em cada iteração. Feistel utilizou os seguintes conceitos na concepção de seu método:

- Substituição: Cada elemento ou grupo de elementos do texto claro é unicamente trocado por um elemento ou grupo de elementos de texto cifrado correspondente.
- Permutação: Uma sequência de elementos do texto claro é trocada por uma permutação daquela sequência. Desta forma, nenhum elemento é adicionado, removido ou trocado de posição na sequência, sendo a ordem em que os elementos aparecem na sequência alterada.

A figura 2.4 mostra a sequência de funcionamento da cifra de Feistel:

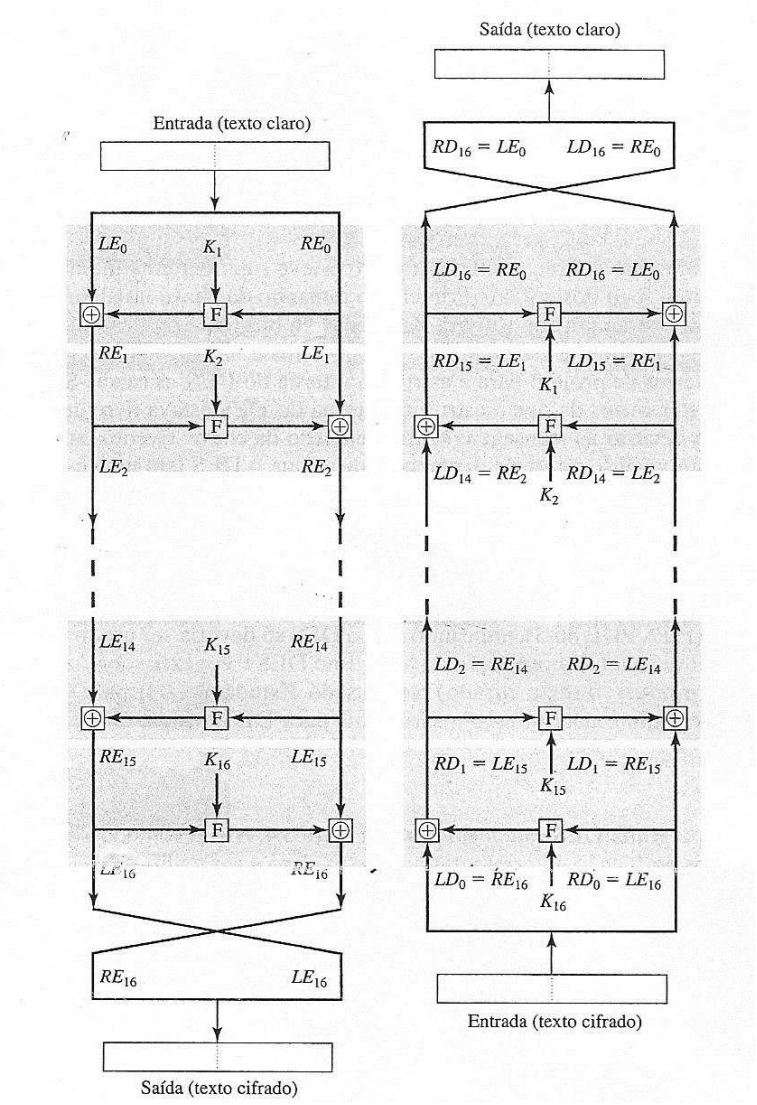

Figura 2.4: Criptografia e decriptografia de Feistel

Fonte: Stallings, 2003, 4ª Ed.

Do lado esquerdo, está retratado o processo de encriptação para um único bloco de texto claro. A entrada para esse algoritmo compreende o bloco de texto claro e a chave K. O bloco é dividido em duas metades, L_0 e R_0, que passarão por *n* rodadas de processamento e em seguida serão combinadas para formar o texto cifrado. Cada rodada *i* terá como entrada L_{i-1} e R_{i-1}, derivadas da iteração anterior, assim como uma subchave K_i, derivada da chave K.

Todas as iterações são feitas da mesma forma. A substituição é feita no lado esquerdo do bloco, através da aplicação de uma função F ao lado direito do bloco e, em seguida, fazendo o ou-exclusivo da saída da função com o lado esquerdo do bloco. A função é a mesma para todas as rodadas de execução, mas acaba parametrizada pela subchave da rodada K_i. Após a substituição, é feita uma permutação entre as duas metades do bloco, invertendo os dois de posição.

A forma como ocorre a cifra de Feistel depende na escolha dos seguintes parâmetros:

- Tamanho de bloco: Um tamanho maior de bloco significa maior segurança, dado que os outros elementos continuem os mesmos. No entanto, a velocidade de encriptação/decriptação fica reduzida para o algoritmo. Em geral, se usam blocos de 64 bits, apesar de no AES o bloco ser de 128 bits.
- Tamanho da chave: Esse parâmetro segue a mesma lógica do anterior: quanto maior o tamanho da chave, maior a segurança, porém a velocidade do algoritmo será comprometida. Chaves de 64 bits ou menos são hoje consideradas inadequadas, com as chaves de 128 bits tendo se tornado as mais comuns.
- Número de rodadas: A base da cifra de Feistel é que uma única iteração não oferece segurança considerável, mas múltiplas rodadas de execução aumentarão a segurança. O usual são 16 rodadas.
- Algoritmo de geração de subchaves: Quanto mais complexo esse algoritmo, maior será a dificuldade de quebrá-lo.
- Função rodada F: Novamente, uma maior complexidade deverá levar a uma maior resistência às tentativas de quebrar o algoritmo.

O processo de decriptação da cifra de Feistel é bastante similar ao de encriptação. O texto cifrado é usado como entrada e o algoritmo é o mesmo, porém as chaves K_i são usadas na ordem inversa. Ou seja, K_n é usada na primeira iteração, em seguida K_{n-1}, e assim por diante, até que K_1 será usada na última rodada.

A figura 2.4 retrata o algoritmo de decriptação funcionando no sentido inverso do algoritmo de encriptação, mostrando que o bloco em cada passo da decriptação é igual àquele correspondente gerado durante a encriptação, mas com os lados invertidos.

A cifra de Feistel foi, e ainda é, utilizada na confecção de diversos algoritmos, como o Blowfish e o DES. É este último que será utilizado na construção do trabalho, e explicado na seção 2.2.4.

2.2.3.3. – Criptografia Assimétrica

"A criação da criptografia de chave pública é a maior, mais importante e talvez a única verdadeira revolução ocorrida na história da criptografia." (Stallings, 2003, Tradução Livre)

A grande revolução ocorreu porque, na criptografia por chaves assimétricas, existem duas grandes diferenças em relação à criptografia por chaves simétricas: Em primeiro lugar, porque é baseada em funções matemáticas ao invés de substituição e permutação. Em segundo lugar, porque utiliza-se de chaves assimétricas, ou seja, usa duas chaves distintas, ao contrário do que acontecia do método de encriptação simétrica, onde existia uma única chave. É utilizado um par de chaves que será usado para realizar as operações de criptografia. Caso uma seja escolhida para fazer a encriptação, a outra será responsável por fazer a decriptação. A transformação que será realizada pelo algoritmo de encriptação depende da chave pública ou privada fornecida na entrada. A figura 2.5 mostra como funciona esse processo.

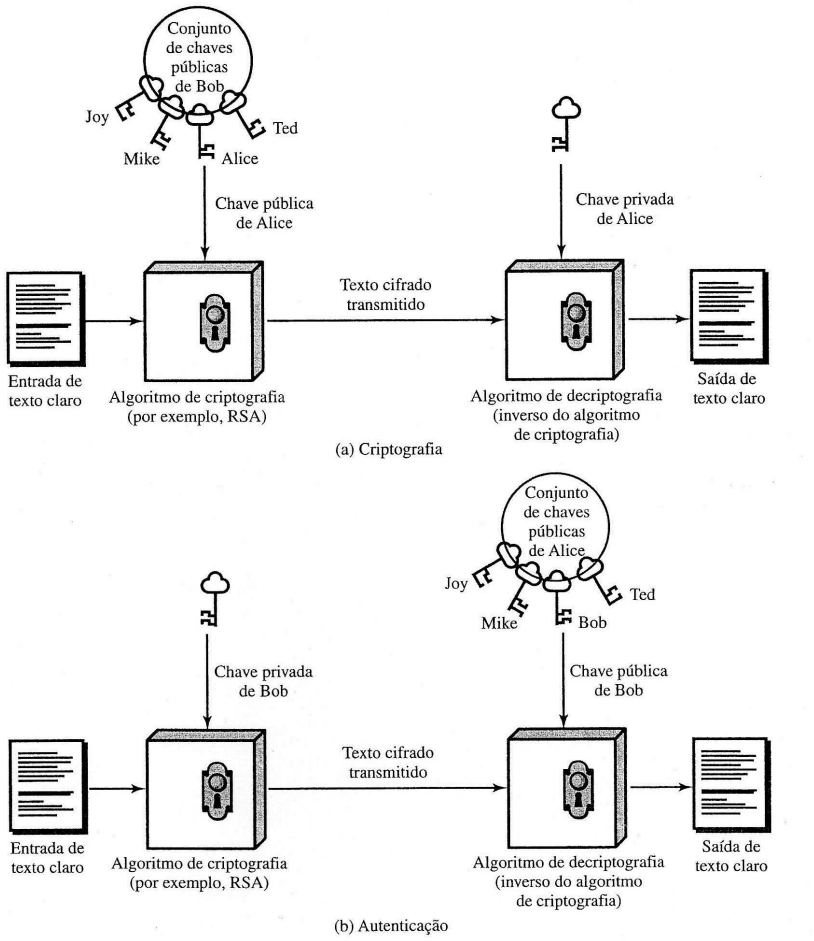

Figura 2.5: Modelo de criptografia de chave pública

Fonte: Stallings, 2003, 4ª Ed.

2.2.4. – DES (Data Encryption Standard)

O mais utilizado esquema de encriptação ao longo da história é fundado no DES (do inglês, Padrão de Encriptação de Dados), que começou a ser utilizado em 1977. O algoritmo em si é chamado de DEA (Algoritmo de Encriptação de Dados). No DES, a informação é encriptada em blocos de 64 bits. (Stallings, 2003)

A construção geral do algoritmo pode ser vista na figura 2.6. Como acontece em qualquer esquema de encriptação, existem duas entradas para a função: o texto claro e a chave. No caso do DES, utilizam-se blocos de 64 bits no texto claro e uma chave que também tem 64 bits, dos quais apenas 56 são de fato utilizados. Os outros 8 podem ser utilizados para paridade ou definidos aleatoriamente.

Olhando o lado esquerdo da figura 2.6, percebe-se que o algoritmo consiste de 3 fases distintas: na primeira, ocorre uma permutação inicial (IP) em cima do texto claro de 64 bits,

que irá rearranjar os bits para gerar a entrada permutada. Na segunda etapa, é executada a função de encriptação por 16 iterações. Essa função consiste de operações de substituição e permutação. A saída da última rodada de encriptação será um conjunto de 64 bits, função do texto claro. As metades esquerda e direita deste resultado são trocadas entre si para gerar uma pré-saída. Em cima desse novo conjunto de bits, é realizada uma permutação (IP^{-1}), que é a inversa da permutação inicial realizada, para gerar o texto cifrado de 64 bits. Com exceção das permutações feitas no início e no fim, o DES se trata de uma aplicação exata da cifra de Feistel, como definido na figura 2.4.

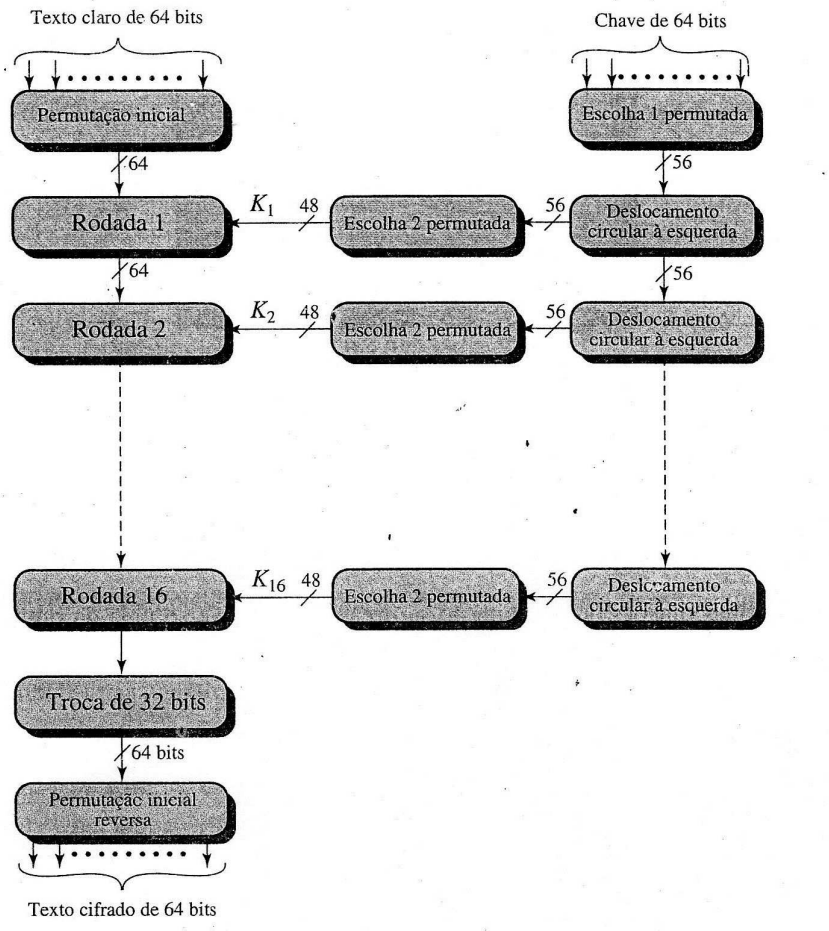

Figura 2.6: Representação geral do algoritmo de criptografia DES

Fonte: Stallings, 2003, 4ª Ed.

O lado direto da figura 2.6 mostra como a chave de 56 bits é utilizada. Em um primeiro passo, a chave passa por uma função de permutação. Então, para cada uma das 16 iterações, uma subchave (K_i) é gerada pela execução de um deslocamento circular à esquerda

e uma permutação. A função de permutação permanece inalterada durante as 16 iterações, mas a nova subchave gerada a cada rodada será sempre diferente da anterior, devido ao deslocamento realizado nos bits de cada subchave.

A permutação inicial e a sua inversão são definidos em tabelas de permutação do DES. Essas tabelas podem ser vistas no conjunto de tabelas 3.2. Especificamente, as tabelas de permutação e permutação inversa são as 3.2a e 3.2b, respectivamente.

Tabela 2.2: Tabelas de Permutação do DES

(a) Permutação Inicial (IP)

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

(b) Permutação Inversa (IP^{-1})

40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

(c) Permutação de Expansão (E)

32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

(d) Função de Permutação (P)

16	7	20	21	29	12	28	17
1	15	23	36	5	18	31	10
2	8	24	14	32	27	3	9
19	13	30	6	22	11	4	25

Fonte: Stallings, 2008

As tabelas de permutações inicial e inversa funcionam da seguinte forma: cada uma recebe como entrada um bloco de 64 bits. Esses bits são numerados de 1 a 64. A tabela de permutação possui 64 posições, que irão indicar como deve ser feita a permutação dos números de 1 a 64. Cada entrada na tabela de permutação indica a posição numerada da entrada no bloco que está sendo encriptado. Para mostrar isso, tomemos como exemplo a função M abaixo:

M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8
M_9	M_{10}	M_{11}	M_{12}	M_{13}	M_{14}	M_{15}	M_{16}
M_{17}	M_{18}	M_{19}	M_{20}	M_{21}	M_{22}	M_{23}	M_{24}
M_{25}	M_{26}	M_{27}	M_{28}	M_{29}	M_{30}	M_{31}	M_{32}
M_{33}	M_{34}	M_{35}	M_{36}	M_{37}	M_{38}	M_{39}	M_{40}
M_{41}	M_{42}	M_{43}	M_{44}	M_{45}	M_{46}	M_{47}	M_{48}
M_{49}	M_{50}	M_{51}	M_{52}	M_{53}	M_{54}	M_{55}	M_{56}
M_{57}	M_{58}	M_{59}	M_{60}	M_{61}	M_{62}	M_{63}	M_{64}

Onde M_i é um dígito binário. Na sequência, a permutação X = IP(M) fica assim:

M_{58}	M_{50}	M_{42}	M_{34}	M_{26}	M_{18}	M_{10}	M_2
M_{60}	M_{52}	M_{44}	M_{36}	M_{28}	M_{20}	M_{12}	M_4
M_{62}	M_{54}	M_{46}	M_{38}	M_{30}	M_{22}	M_{14}	M_6
M_{64}	M_{56}	M_{48}	M_{40}	M_{32}	M_{24}	M_{16}	M_8
M_{57}	M_{49}	M_{41}	M_{33}	M_{25}	M_{17}	M_9	M_1
M_{59}	M_{51}	M_{43}	M_{35}	M_{27}	M_{19}	M_{11}	M_3
M_{61}	M_{53}	M_{45}	M_{37}	M_{29}	M_{21}	M_{13}	M_5
M_{63}	M_{55}	M_{47}	M_{39}	M_{31}	M_{23}	M_{15}	M_7

Se for realizada a permutação inversa em cima de X, poderá ser verificado que os dígitos são retornados à sua posição original.

A figura 2.7 mostra o funcionamento interno de uma única rodada do DES. Do lado esquerdo, é possível ver o processo de encriptação do texto e do lado direito está retratado o trabalho realizado com a chave a cada iteração. Analisando primeiro o lado esquerdo, pode-se ver que cada bloco de 64 bits é tratado como dois blocos separados de 32 bits, chamados de L (esquerda) e R (direita). O processamento em cada rodada pode ser definido da seguinte forma:

$$L_i = R_{i-1} \text{ (Eq. 2.3)}$$

$$R_i = L_{i-1} \oplus F(R_{i-1}, K_i) \text{ (Eq. 2.4)}$$

A chave Ki possui 48 bits, enquanto a entrada R possui 32 bits. Primeiramente, R é expandida para 48 bits através da função de permutação e expansão definida na tabela 2.2c. Os 48 bits resultantes passam por uma operação de XOR com K_i. Esse novo resultado passa por uma função de substituição, com o objetivo de gerar uma saída de 32 bits, que por sua vez é permutada conforme definido na tabela 2.2d.

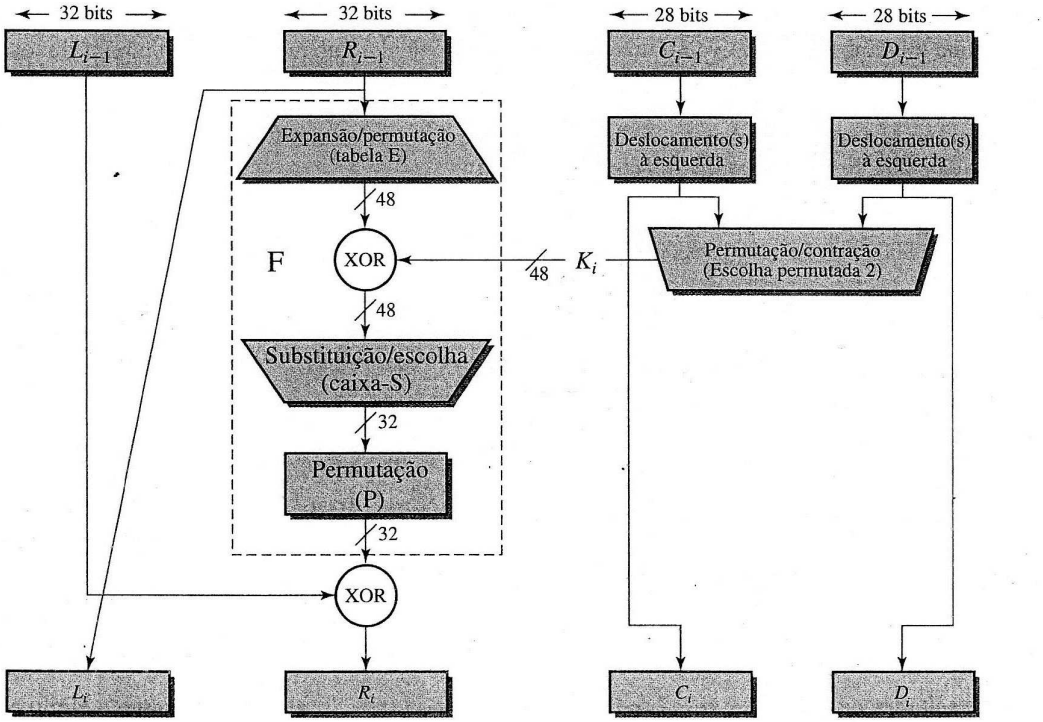

Figura 2.7: Rodada individual do algoritmo DES

Fonte: Stallings, 2003, 4ª Ed.

A definição da caixa-S, presente na função F, é ilustrada na figura 2.8. A substituição consiste, na verdade, de 8 caixas-S. Cada uma dessas caixas recebe uma entrada de 6 bits e gera uma saída de 4 bits. Essas transformações são definidas na tabela 2.3, que deve ser lida da seguinte forma: o primeiro e o último bit da entrada para cada caixa formam números binários de 2 bits. O número gerado para uma caixa vai definir qual das 4 linhas da mesma será lida na operação de substituição, e os 4 bits do meio do bloco de entrada definem a coluna. Por exemplo, para uma entrada 101010, a linha será definida pelo binário 10 (linha 2) e a coluna será definida pelo binário 0101 (coluna 5). Vale ressaltar que a contagem das linhas e colunas tem que sair de zero, e não um.

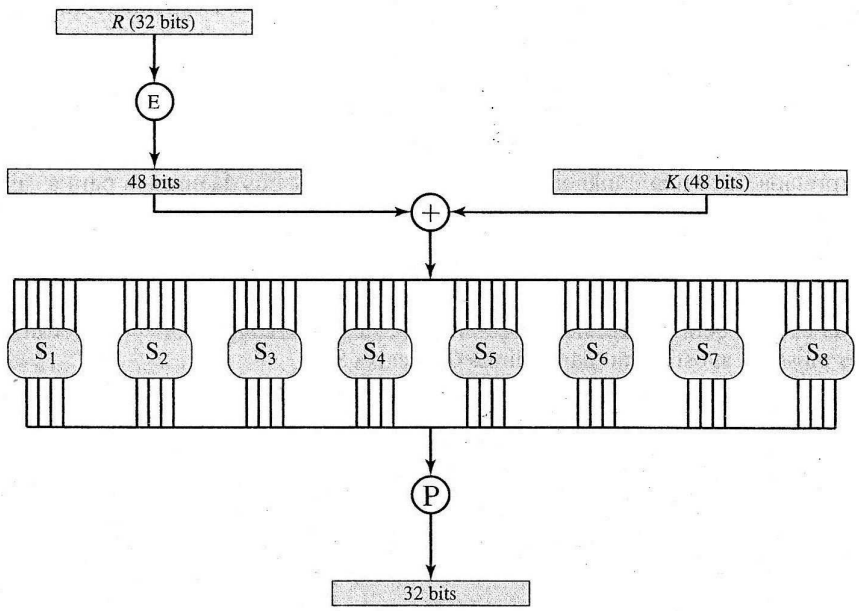

Figura 2.8: Cálculo de F(R, K)

Fonte: Stallings, 2003, 4ª Ed.

Tabela 2.3: Definição das caixas-S do DES

S_1	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

S_2	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15

	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

S_3	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

S_4	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

S_5	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

S_6	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

S_7	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

S_8	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

Fonte: Stallings, 2008

A geração das chaves a cada rodada é feita através de uma sequência de substituições e deslocamentos circulares da chave anterior. A chave de 64 bits é numerada, e passa primeiramente por um processo que ignora todo oitavo bit, conforme mostrado na tabela 2.4a. Essa nova chave de 56 bits passa por um processo de permutação definido na tabela Escolha

Permutada Um (tabela 2.4b). O resultado é tratado como duas chaves distintas de 28 bits, chamadas de C_0 e D_0. No início de cada rodada, C_{i-1} e D_{i-1} são submetidas, separadamente, a um processo de deslocamento circular à esquerda de 1 ou 2 bits, de acordo com o definido na tabela 2.4d. C_{i-1} e D_{i-1} também servem como entrada para a operação chamada de Escolha Permutada Dois (tabela 2.4c), que irá gerar a chave de 48 bits utilizada na função F.

Tabela 2.4: Cálculo de escalonamento de chave do DES

(a) Chave de entrada

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56
57	58	59	60	61	62	63	64

(b) Escolha Permutada Um

57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36
63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

(c) Escolha Permutada Dois

14	17	11	24	1	5	3	28
15	6	21	10	23	19	12	4
26	8	16	7	27	20	13	2
41	52	31	37	47	55	30	40
51	45	33	48	44	49	39	56
34	53	46	42	50	36	29	32

(d) Escalonamento de Deslocamento à Esquerda

Número da rodada	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Bits Escalonados	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

Fonte: Stallings, 2008

A decriptação do DES, assim como com qualquer cifra de Feistel, usa o mesmo algoritmo da encriptação, porém com as chaves sendo utilizadas em ordem inversa.

Como neste trabalho é proposto o uso da criptografia com bluetooth, um tipo de comunicação sem fio em curtas distâncias, popularmente utilizada na comunicação entre celulares e dispositivos próximos, a seguir será feita breve referencia às comunicações sem fio, em especial às que utilizam o aparelho celular como transmissor e receptor. Este tipo de comunicação está cada vez mais sendo utilizada, permitindo a tão deseja mobilidade, mas trazendo riscos de segurança para a informação, se não utilizada com mecanismos de proteção, como é o caso da criptografia.

2.3 – TELEFONIA MÓVEL

2.3.1. – Evolução da Telefonia Móvel

O início da telefonia móvel pode ser contada com o avanço da tecnologia de rádio que começou nos anos 40. As primeiras aplicações vieram nos táxis, carros de polícia, e outros veículos de serviço onde dois rádios se comunicavam entre si, ou um deles se comunicava com uma base central.

O primeiro telefone móvel oficial surgiu na Suécia, em 1946, e era usado pela polícia do país. Em 1947, foi criado pela Bell Labs o modelo para as células de comunicação que viriam a ser utilizadas muitos anos depois. No entanto, algumas tecnologias, principalmente a eletrônica, ainda não haviam sido desenvolvidas e ainda demoraria algum tempo para que isso acontece. A eletrônica usada nos primeiros telefones móveis só viria a ser criada nos anos 60.

Em 1967, se tornou disponível a tecnologia para telefonia móvel. No entanto, o usuário tinha que ficar dentro da área de uma célula. Não era possível a transferência de uma ligação de uma torre para outra, portanto o usuário não conseguir continuar em uma chamada se ele saísse da área de atuação de uma torre. Essa tecnologia foi criada em 1970, por Amos Edward Joel, também da Bell Labs.

O primeiro protótipo que atendia as especificações das novas tecnologias foi o DynaTAC, da Motorola, criado em 1973, e nunca chegou a ser disponibilizado para o público. Dez anos depois, foi lançado, também pela Motorola, o DynaTAC8000X, primeiro aparelho celular disponibilizado ao público, e que marcou o início da primeira geração de celulares.

Para a época, era um aparelho bem leve (795 gramas), que media 33cm x 4,5cm x 9cm, e ganhou o apelido de “tijolo”, devido a sua forma. A figura 2.9 mostra esse aparelho.

Até o final dos anos 80, os telefones móveis ganharam popularidade, apesar de a maioria deles não poderem ser carregados na mão. Em geral, os telefones eram montados para instalação permanente em um veículo. O problema era o consumo de bateria, e a solução encontrada foi usar a saída de força do carro para prover a energia necessária para executar as chamadas.

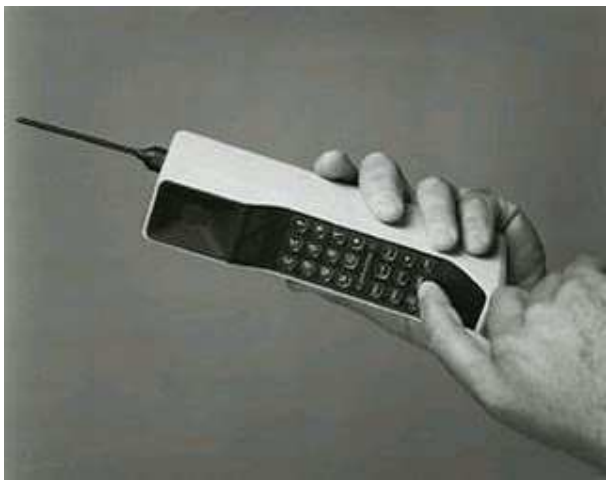

Figura 2.9: o DynaTAC 8000X.
Fonte: http://neuronio-virtual.blogspot.com/2008/10/iphone-o-verdadeiro-celular.html

A segunda geração dos telefones celulares surgiu no início dos anos 90, e foi marcada pela troca das redes analógicas pelas redes digitais. Iniciou-se o uso das redes CDMA, TDMA e GSM, o que aumentou a qualidade do sinal e das chamadas e diminuiu a quantidade de chamadas perdidas. O celular que marcou o lançamento do 2G foi mais uma vez da Motorola: o MicroTAC, marcado pelo seu menor peso, pelo aumento na qualidade e diminuição do tamanho da bateria e pelo seu menor tamanho, possibilitando que fosse de fato um telefone portátil. Com o passar dos anos, alguns serviços começaram a ser implementados, como o SMS, o MMS e o WAP. O MicroTAC pode ser visto na figura 2.10.

Figura 2.10: O Motorola MicroTAC.
Fonte: http://neuronio-virtual.blogspot.com/2008/10/iphone-o-verdadeiro-celular.html

A Motorola lançou também o StarTAC, que pode ser visto na figura 2.11, em 1996.

Figura 2.11: Motorola StarTAC.
Fonte: http://neuronio-virtual.blogspot.com/2008/10/iphone-o-verdadeiro-celular.html

Ao final dos anos 90, foi introduzida a terceira geração de telefones móveis, tecnologia usada até os dias de hoje. O 3G se difere do 2G principalmente por diferenças na tecnologia e nos serviços, embora os padrões possam ser diferentes, de acordo com a rede. Também é marca da geração 3G a disponibilização de novos serviços, que incluem o WiFi, streaming de rádio e TV, videoconferência, entre outros. Hoje, os aparelhos incluem Smartphones, iPhones e até telefones que recebem sinal de TV em alta definição. A figura 2.12 mostra o Nokia N95, um dos celulares com tecnologia 3G lançados pela fabricante finlandesa. Entre as novidades, surgiu também o sistema operacional Android, da Google, que será discutido mais à frente. (http://neuronio-virtual.blogspot.com/2008/10/iphone-o-verdadeiro-celular.html, acessado em

01/06/2011; http://www.tech-faq.com/history-of-cell-phones.html, acessado em 01/06/2011; http://www.tecmundo.com.br/2140-historia-a-evolucao-do-celular.htm, acessado em 01/06/2011)

Figura 2.12: Nokia N95, exemplo de celular com tecnologia 3G.
Fonte: Fonte: http://infodicas.com/blog/?p=122

2.3.2. – O Sistema Operacional Android

O Android é um software de camadas para dispositivos móveis, que inclui um sistema operacional, um middleware e aplicações. Entre as suas características, estão:

- Framework de aplicações, com possibilidade de reutilização e substituição de componentes;
- Máquina virtual Dalvik, otimizada para aparelhos móveis;
- Browser integrado, baseado no mecanismo *open source* Webkit;
- Gráficos otimizados providos por uma biblioteca de gráficos 2D customizada. Gráficos em 3D baseados na especificação OpenGL ES 1.0;
- SQLite para armazenamento estruturado de dados;
- Suporte à mídia para arquivos comuns de áudio, vídeo e imagens;
- Telefonia GSM;
- Bluetooth, EDGE, 3G e WiFi;
- Câmera, GPS, bússola e acelerômetro;
- Um rico ambiente de desenvolvimento, incluindo um simulador de dispositivo móvel, ferramentas para depuração e um plugin para o Eclipse IDE (será discutido a fundo mais à frente).

O Android possui alguns aplicativos nativos, incluindo cliente de e-mail, programa de SMS, calendário, mapas, navegador e contatos, entre outros. Todos os aplicativos são desenvolvidos com uso da linguagem Java.

O Android utiliza na versão 2.6 do Linux para serviços nativos do sistema, como segurança, gerenciamento de memória e processos, pilha de rede e modelo do driver. O kernel também age como uma camada entre o hardware e as camadas de software.

2.3.2.1. – Desenvolvimento para o Android

O Android SDK (Software Development Kit) fornece as ferramentas e APIs (Application Programming Interface) necessárias para realização do desenvolvimento de aplicações na plataforma Android usando a linguagem de programação Java.

Maiores detalhes sobre o desenvolvimento específico realizado para esse projeto serão discutidos no capítulo 4.

2.4. – BLUETOOTH

2.4.1. – O que é o Bluetooth?

De acordo com Michael Miller (Miller, 2001, p.4), "(O *Bluetooth*) é uma tecnologia que promete eliminar a maioria dos cabos que conectam os vários dispositivos de comunicação pessoal," através do uso de ondas curtas de rádio para suas transmissões. A tecnologia *Bluetooth* utiliza, para tal, um pequeno chip de baixa voltagem com radiotransmissor em um dispositivo eletrônico tradicional. Através desse chip, é possível realizar comunicações de dados e voz com outros dispositivos.

Para o uso do *Bluetooth*, foi especificado o uso da banda ISM (*Industrial, Scientific and Medical Band*, ou Industrial, Científica e Médica), com frequências entre 2,4 e 2,48 gigahertz (GHz). A banda ISM não possui licença, ou seja, ele pode ser utilizada por qualquer um sem nenhuma taxa. Inclusive, ela é utilizada em outros tipos de comunicação incompatíveis com o *Bluetooth*).

O *Bluetooth* surgiu da necessidade de se eliminar o cabo em algumas comunicações de nível pessoal. Por exemplo, entre um aparelho celular e o fone de ouvido utilizado pelo usuário para suas conversas telefônicas. Desta forma, se tornou conveniente que a tecnologia fosse de baixo custo e de baixo consumo de energia.

Após anos de estudo na segunda metade dos anos 90, foi lançada em 1999 a versão 1.0 da especificação *Bluetooth* (*Bluetooth Specification*). Esse documento possuía todas as

informações necessárias para assegurar que todos os dispositivos compatíveis com a tecnologia Bluetooth pudessem se comunicar entre si. Este documento, que possui mais de 1500 páginas, tem duas seções: Volume 1 (Principal) e Volume 2 (Perfis).

O primeiro volume possui em seu conteúdo informações técnicas sobre os seguintes itens:

- Pilha de protocolos;
- Rádio Bluetooth;
- Gerenciador de link;
- Camada de transporte;
- Interoperabilidade entre diferentes protocolos de comunicação;
- Teste e concordância.

O volume 2 trata do que é chamado de “modelos de uso”. Um modelo de uso é uma forma específica da aplicação do Bluetooth, e um perfil é a tecnologia detalhada e os procedimentos necessários para implementar uma determinada aplicação. Os perfis definem como usar a pilha do protocolo Bluetooth, como reduzir opções, como precisar parâmetros no padrão básico e como utilizar processos a partir de diversos padrões básicos.

2.4.2. – Aplicações do Bluetooth

O Bluetooth pode ser utilizado para transmissão de dados: os aparelhos celulares de hoje em dia têm, em sua maioria, o protocolo disponível para transferência de arquivos. Muitas vezes é possível também utilizar fones de ouvido que se comunicam por meio do protocolo com o aparelho celular, e o usuário pode usá-lo também para transmissão de voz.

2.4.3. – Vulnerabilidades do Bluetooth

O Bluetooth utiliza a freqüência de rádio no intervalo entre 2,4 e 2,48 GHz. Esta banda é livre para qualquer uso e propósito. Isso tem um lado positivo e um negativo. O positivo é que ela pode ser utilizada livremente, sem custo com taxas de licenciamento. O negativo é que o espaço dentro da banda é limitado e ela é utilizada para outras finalidades, como telefones e redes sem fio. Além disso, por se tratar de uma freqüência aberta, a comunicação está aberta a interferências ou invasões.

A aplicação desenvolvida neste projeto será o sistema operacional para telefones móveis com Android, e irá utilizar o protocolo Bluetooth para envio e recebimento de mensagens encriptadas com o algoritmo DES. O modelo e sua implementação serão discutidos no próximo capítulo.

CAPÍTULO 3 – PROPOSTA DE SEGURANÇA PARA O BLUETOOTH

Este capítulo apresenta o desenvolvimento da aplicação envolvendo criptografia e comunicação sem fio, por meio do protocolo Bluetooth e do algoritmo DES de encriptação. A seguir, será descrito o modelo e o processo de desenvolvimento utilizado.

3.1 – Modelo Proposto

Para a solução do problema apresentado, será desenvolvida uma aplicação para aparelhos móveis que utilizam o sistema operacional Android. Especificamente, será utilizada a versão 2.1 do sistema para desenvolvimento e teste da aplicação.

Esta solução servirá para troca de arquivos por meio do protocolo Bluetooth, com o objetivo de prover encriptação no arquivo antes da sua transmissão, sendo que esta encriptação será realizada por meio do algoritmo DES.

O produto irá utilizar, na demonstração, dois aparelhos com Android, para demonstrar o funcionamento do aplicativo desenvolvido. Na fase de testes, poderão ser utilizadas máquinas virtuais para verificar o funcionamento do aplicativo.

3.2 – Apresentação Geral do Modelo Proposto

O modelo é representado de forma geral na figura 3.1.

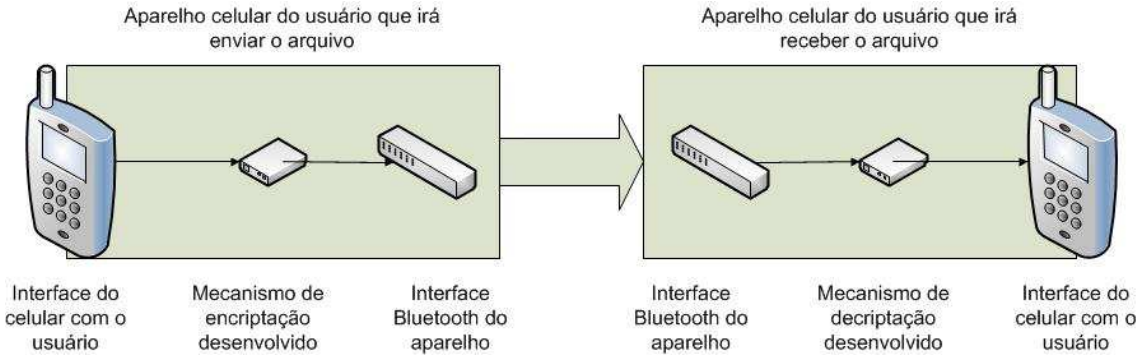

Figura 3.1: Apresentação geral do modelo proposto

As partes integrantes do modelo foram retratadas na ordem em que aparecem no processo. O usuário que irá enviar o arquivo irá interagir com o seu aparelho celular de forma a usar o aplicativo que irá encriptar e enviar a mensagem. O destinatário irá receber a mensagem e utilizar o aplicativo para decriptar o arquivo.

3.2 – Descrição das Etapas do Modelo

O Android usa como linguagem padrão para desenvolvimento de aplicativos o Java. Desta forma, todo o desenvolvimento será feito usando esta linguagem.

Para o desenvolvimento do aplicativo, será utilizado o Eclipse, uma das várias ferramentas de IDE (do inglês, Ambiente de desenvolvimento integrado) gratuitas disponíveis e uma muito comumente utilizada. Para que o desenvolvimento para o Android possa ser realizado no Eclipse, deve ser instalada a SDK (do inglês, Kit de desenvolvimento de software) e configurado o plug-in do Android no Eclipse, para que ele leia a SDK. (http://developer.android.com)

Configurado o Eclipse, cria-se um novo projeto Android na ferramenta, conforme retratado na figura 3.2.

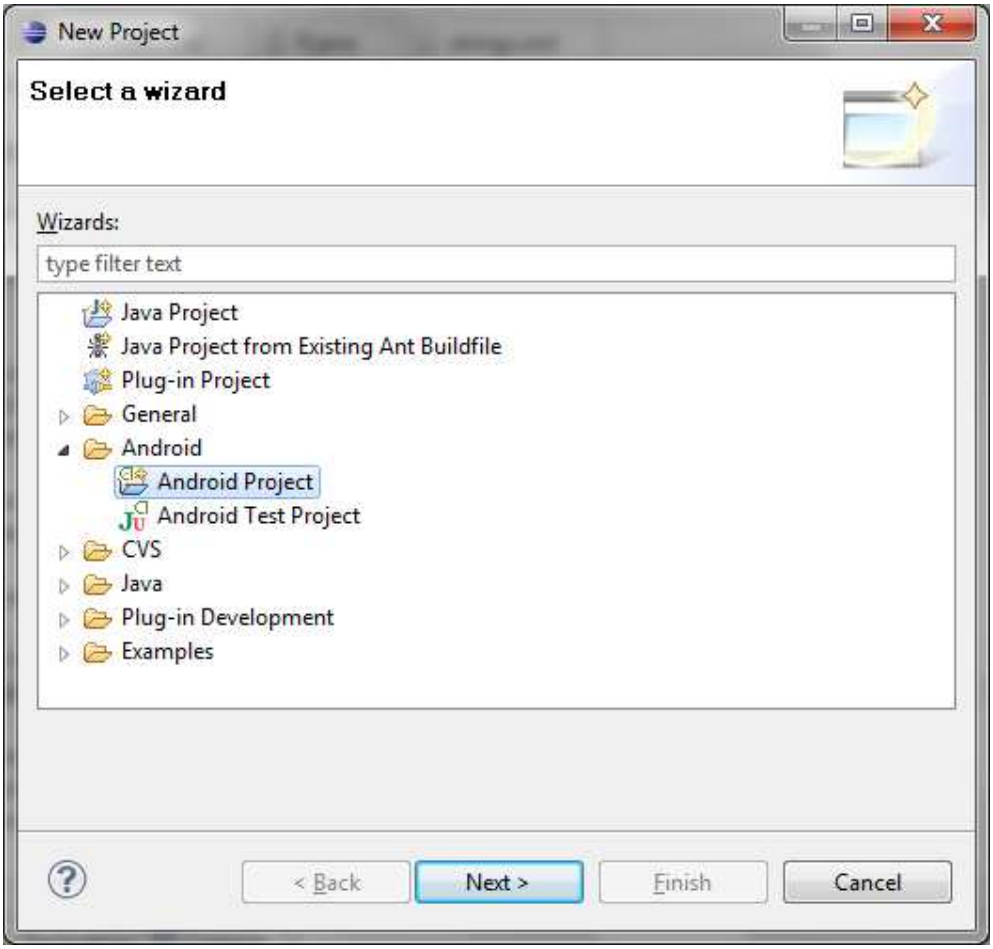

Figura 3.2: Tela do Eclipe IDE mostrando a escolha do projeto

A própria ferramenta irá configurar a estrutura básica necessária para desenvolvimento do aplicativo. O desenvolvimento do código será feito na classe criada pela IDE. No caso do projeto, foi criada a classe BluetoothEncryption.java, dentro do projeto Bluetooth_Encryption, como mostra a figura 3.3.

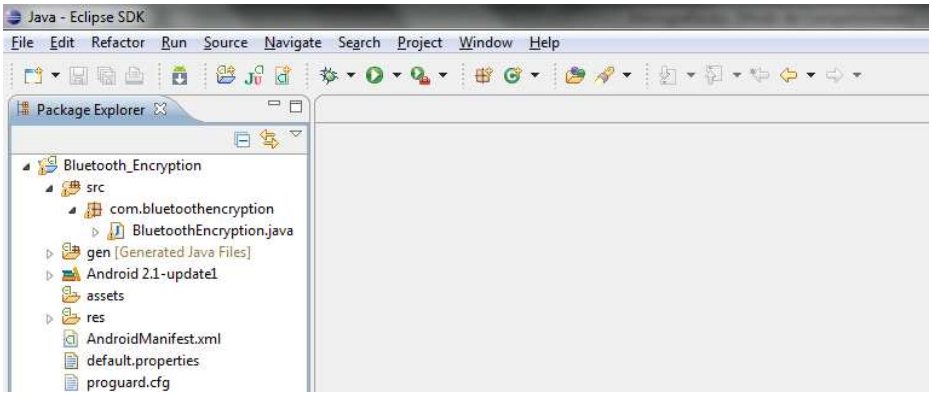

Figura 3.3: Tela do Eclipe IDE mostrando o projeto e a classe criados

A classe já é criada com uma estrutura inicial, necessária para o desenvolvimento de um aplicativo para o Android. O que se faz é desenvolver em cima dessa classe criada, com o código necessário. A implementação deste código será discutida mais adiante.

Para teste do aplicativo, tanto em tempo de desenvolvimento quanto após o seu término, deve-se baixar os pacotes desejados e criar uma máquina virtual Android (AVD – Android Virtual Device). Pode-se, como alternativa para a AVD, utilizar um aparelho para debugar e testar o código.

Os pacotes disponíveis incluem, mas não estão limitados a, plataformas do Android para criação de máquinas virtuais, documentação e plug-ins específicos para determinadas ações, como, por exemplo, o uso do Driver USB.

Para criar uma AVD, deve-se entrar na opção “Android SDK and AVD Manager” disponível no menu “Window”, conforme mostra a figura 3.4.

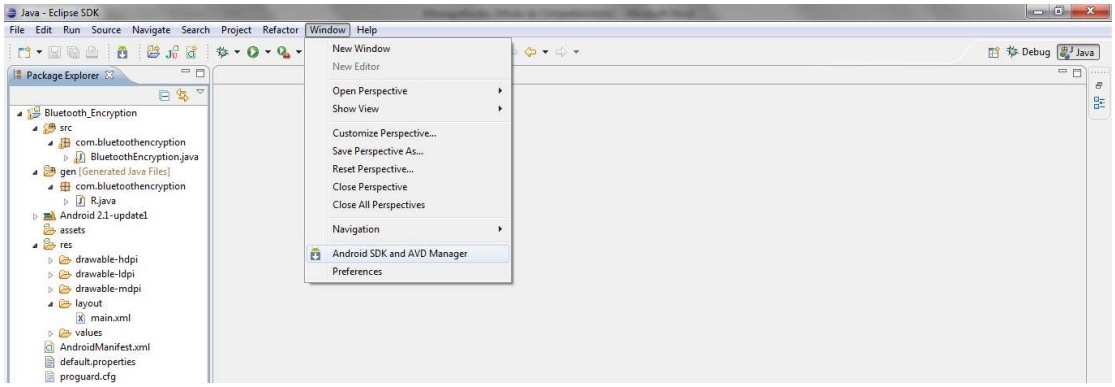

Figura 3.4: Tela do Eclipe IDE mostrando a opção de gerenciamento de SDK e AVD

Na próxima tela, dentro do menu “Virtual devices”, deve-se selecionar a opção para criar uma nova AVD, como retratado na figura 3.5.

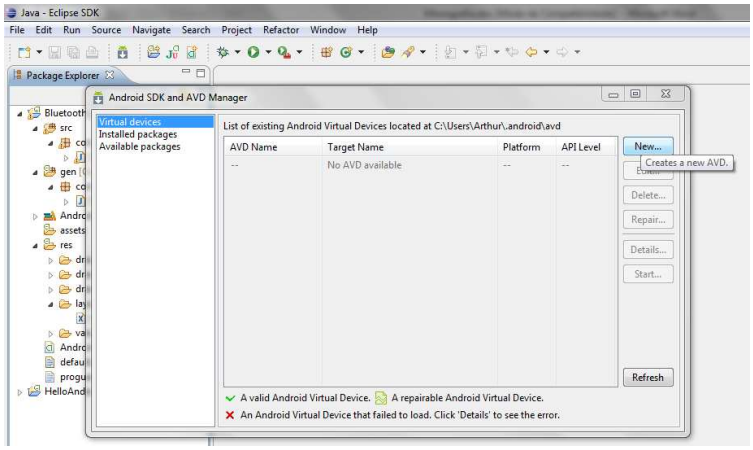

Figura 3.5: Tela do Eclipe IDE mostrando a opção de criação de uma nova AVD

Na tela seguinte, deve-se entrar com os dados da AVD. Para o caso desse aplicativo, só é necessário fornecer o nome e a versão do Android que será utilizada (cujo pacote deve ter sido instalado anteriormente), conforme mostrado na figura 3.6.

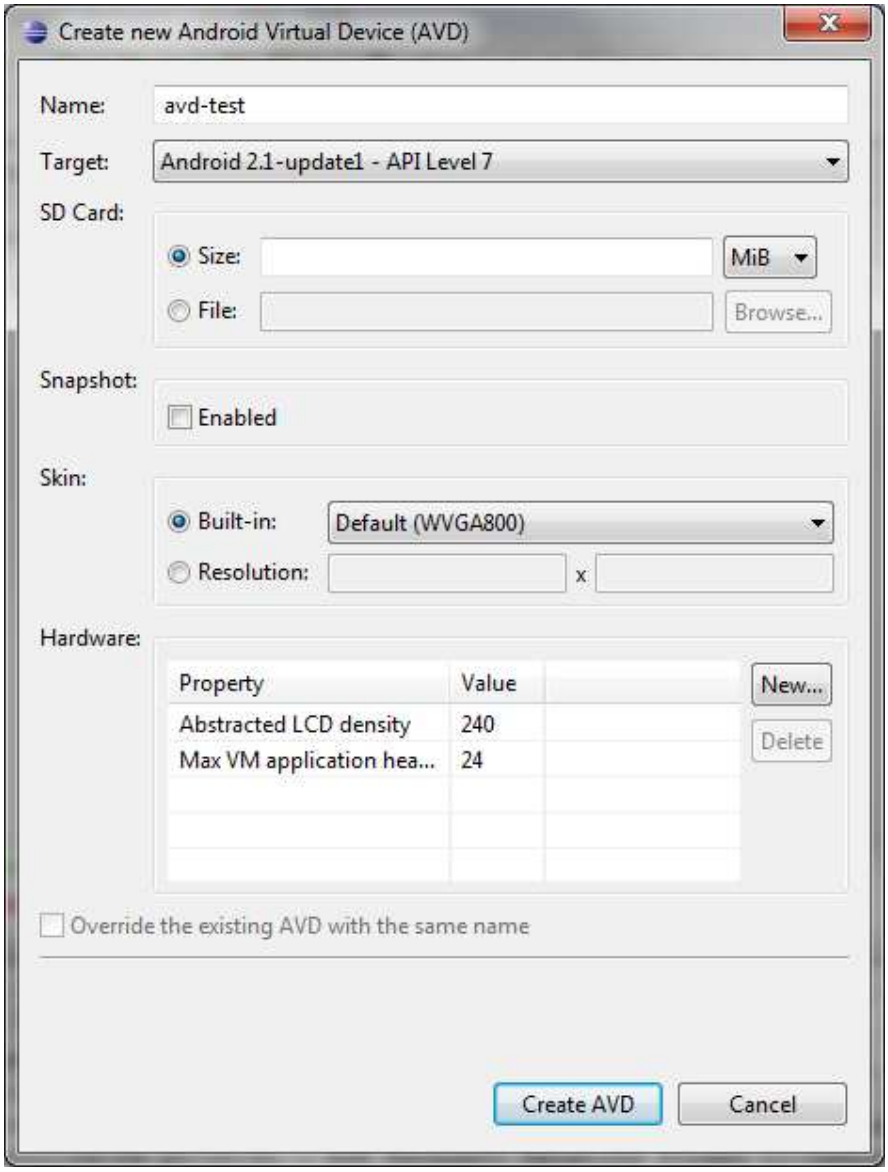

Figura 3.6: Tela do Eclipe IDE mostrando a entrada de dados para criação de uma nova AVD

Após clicar em “Create AVD”, a máquina virtual aparecerá na lista de dispositivos existentes, conforme mostrado na figura 3.7.

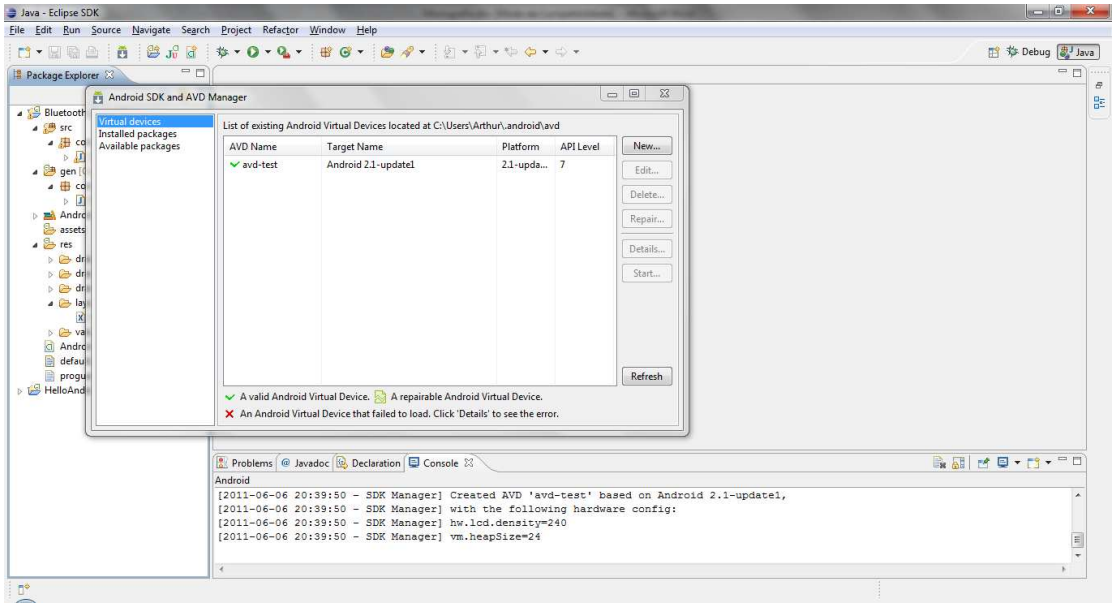

Figura 3.7: Tela do Eclipe IDE mostrando a AVD existente na lista de máquinas virtuais

Após a configuração desse detalhes, pode-se começar o desenvolvimento do aplicativo de fato. O aplicativo desenvolvido irá realizar a encriptação ou decriptação de um arquivo selecionado, de acordo com a seleção do usuário. Qualquer que seja a escolha, será aberta uma tela para a escolha de qual arquivo irá passar pela operação desejada.

O primeiro passo é o usuário abrir o aplicativo. A tela de apresentação já será o menu que irá mostrar as opções de encriptação e envio ou decriptação de um arquivo. Na ordem do modelo apresentado anteriormente, a sequência de ações é a seguinte:

- O usuário que enviará o arquivo irá executar o aplicativo e selecionar a operação “Encriptar e Enviar Arquivo”. Após selecionada essa opção, será aberta uma tela com um explorador de arquivos e pastas, para que se escolha o arquivo que se deseja encriptar. Após escolhido o arquivo, o aplicativo irá encriptá-lo com uso do algoritmo DES descrito anteriormente e fazer o processo de transferência via Bluetooth para o destinatário desejado. Após essa etapa, o arquivo existirá no diretório do aplicativo e poderá ser visto no explorador de arquivos e pastas. No entanto, não poderá ser aberto ou visualizado, já que se tratará do arquivo criptografado.
- O destinatário do arquivo terá de aceitar a transferência a partir de seu aparelho para que o arquivo possa ser recebido. Após esse passo, o usuário poderá ver que o arquivo recebido, ainda criptografado, estará na pasta do aplicativo.

Assim como no caso anterior, o arquivo ainda não poderá ser aberto ou visualizado.

- Após o recebimento do arquivo, o destinatário deve entrar no aplicativo a partir de seu aparelho móvel e, no menu inicial, selecionar a opção “Decriptar Arquivo Recebido”. Após a seleção dessa opção, será aberto o explorador de arquivos e pastas, no qual deve ser escolhido o arquivo que será decriptado. Essa opção irá decriptografar o arquivo recebido, com uso de um processo de decriptografia para o DES, para que o mesmo possa ser lido com sucesso pelo usuário.

O resultado final do desenvolvimento deverá ser o arquivo visto corretamente pelo destinatário, com a encriptação tendo sida corretamente aplicada durante a transferência.

3.3 – Descrição da Implementação

Após a criação do projeto Android no Eclipse, a própria IDE se encarrega de criar a estrutura básica do projeto. Para desenvolvimento deste aplicativo, foi criado o projeto Bluetooth_Encryption. Os arquivos principais, que devem ser observados, são os seguintes:

- Bluetooth_Encryption/src/com.bluetoothencryption/BluetoothEncryption.java – Nesse arquivo é criada, a classe principal, BluetoothEncryption, e um código inicial para setar a tela a ser visualizada como oriunda do main.xml;
- Bluetooth_Encryption/gen/com.bluetoothencryption/R.java – O R é uma classe de definições das constantes que serão usadas no aplicativo. As telas criadas, as strings definidas, todas são referenciadas no R.java para leitura do programa principal. Nenhuma alteração é feita manualmente nesse arquivo, pois a IDE já faz as atualizações necessárias no momento adequado.
- Bluetooth_Encryption/res/layout/main.xml – É onde é criado o layout da tela que será usada no aplicativo.
- Bluetooth_Encryption/res/values/strings.xml – Onde se definem as strings que serão usadas no aplicativo. Por exemplo, os nomes dos botões da tela principal podem ser referenciados pelo strings.xml.

No main.xml será criada a tela que será exibida quando o usuário abrir o aplicativo. Deve-se selecionar o tamanho da tela e, após isso, utilizar as opções de *drag-and-drop* da ferramenta para criar a tela da maneira desejada. A figura 3.8 mostra um exemplo de tela criada pelo Eclipse dentro da ferramenta, e a figura 3.9 retrata o menu dentro da AVD.

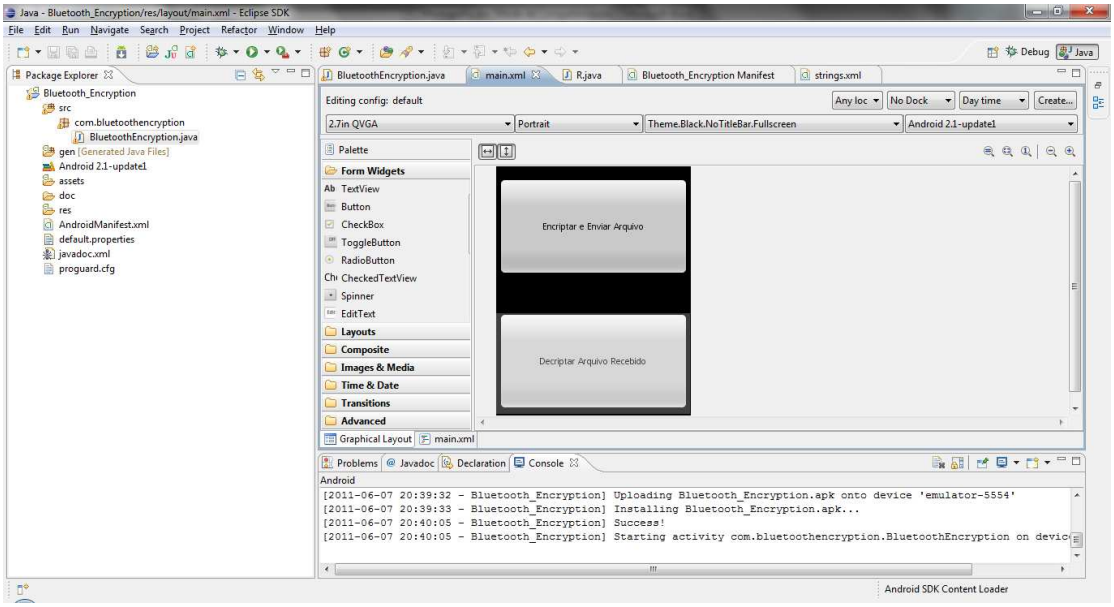

Figura 3.8: Tela do Eclipe IDE com a tela do aplicativo criada

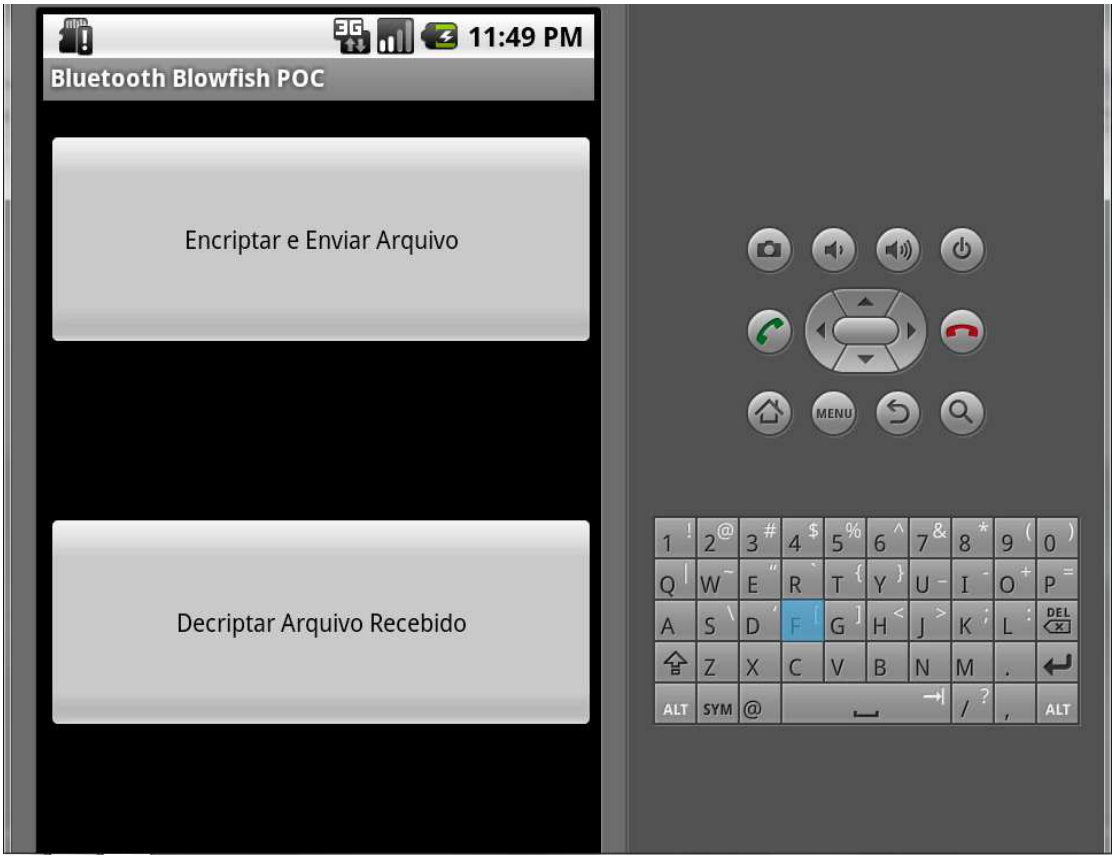

Figura 3.9: Tela da AVD com o menu do aplicativo exibido

O arquivo main.xml já é referenciado dentro da classe R.java, logo a execução do aplicativo em uma AVD ou aparelho físico irá carregar a tela inicial do programa.

As opções do menu devem ser definidas no arquivo strings.xml. Nas figuras 3.8 e 3.9 é possível ver as opções “Encriptar e Enviar Arquivo” e “Decriptar Arquivo Recebido”. Esses textos devem ser definidos através de strings a serem criadas dentro do strings.xml. No caso deste projeto, foram criadas as strings “encrypt_send_file” e “decrypt_file” com a definição de texto a ser exibido por cada uma. Em seguida, define-se a propriedade OnClick em cada um dos botões no main.xml para ler as strings criadas.

Após criada a tela com os botões adequados, pode-se partir para o desenvolvimento do código do aplicativo, na classe BluetoothEncryption.java.

O código inicial da classe criada pelo Eclipse para o projeto Bluetooth_Encryption está retratado no ANEXO A – Código básico gerado pelo Eclipse na criação de um projeto Android, e tem por objetivo apenas a exibição da tela criada no main.xml. A partir daqui é que começam as alterações ao aplicativo.

Em primeiro lugar, deve-se importar as bibliotecas necessárias para o desenvolvimento e uso do aplicativo. Em geral, essas bibliotecas vão sendo descobertas ao longo do desenvolvimento, mas as utilizadas nesse projeto encontram-se no APÊNDICE A – Importação das bibliotecas necessárias para o desenvolvimento e uso do aplicativo:

Em seguida, deve-se criar uma ação de fato para os botões desenhados no leiaute da tela. Na atividade principal do código, é definida uma variável em cima de cada botão do menu, para que o programa possa trabalhar em cima dela depois. No caso, foram criadas duas variáveis, uma de encriptação e outra de decriptação. Em seguida, foram definidas duas funções, uma para encriptação e outra para decriptação, que tem, a princípio, a mesma função: abrir o explorador de arquivos padrão do aparelho para permitir que o usuário escolha o arquivo desejado. No entanto, as duas funções devolvem valores diferentes dentro da variável chamada de requestCode, responsável por dirigir o resto da atividade pelo caminho escolhido pelo usuário (encriptação ou decriptação). Este código pode ser verificado no APÊNDICE B – Código criado para escolha de arquivo por parte do usuário e definição do caminho a ser seguido dentro da atividade.

Para a cópia do conteúdo do buffer para um vetor que poderá ser lido como atributo, foram criados dois métodos: toByteArray e copy. O toByteArray recebe da atividade o streaming de bytes que será lido como entrada, cria um streaming para saída e retorna esta saída para a atividade. Entre estas duas últimas tarefas, é chamado o método copy, que irá realizar a cópia efetiva do conteúdo do buffer para o vetor criado pelo toByteArray. Os dois

métodos estão definidos no APÊNDICE C – Métodos para cópia do arquivo em buffer para um atributo.

No final do código descrito no APÊNDICE B, qualquer que tenha sido a seleção (encriptação ou decriptação) existe a chamada do resultado da atividade, o método onActivityResult. Dentro deste método, o primeiro passo é entrar em uma estrutura de seleção, para definir se o caminho a ser seguido é o de encriptação ou decriptação. Após esse passo, alguns passos ocorrem de forma semelhante: o arquivo é colocado em um streaming de buffer, ocorrendo exceções caso o arquivo não possa ser aberto com sucesso. Em seguida são chamados os métodos que vão realizar a cópia do conteúdo do buffer para um atributo definido como um vetor de bytes. Aqui, começa a ocorrer a diferença nas escolhas: no caso da encriptação, é gerado um arquivo com a extensão do arquivo original mais a extensão .cry. Este é o arquivo que será encriptado com uso do DES e que será transmitido posteriormente. Após a geração do arquivo de saída, é realizado o processo de encriptação do arquivo através da implementação do algoritmo de criptografia e o arquivo é salvo, já encriptado, com o mesmo nome do arquivo de extensão .cry gerado anteriormente; no caso da decriptação, o usuário escolherá o arquivo .cry recebido via Bluetooth para que ele seja transformado novamente no arquivo original. Após essa escolha, o programa irá criar um arquivo igual ao escolhido pelo usuário, porém sem a extensão .cry, em processo inverso ao ocorrido na primeira escolha. Após este passo, o aplicativo realizará a decriptação através do algoritmo implementado, salvando o novo arquivo em formato igual ao original. Este código pode ser visto no APÊNDICE D – Encriptação/decriptação do arquivo escolhido.

Desta forma, o aplicativo é projetado para possibilitar a escolha de um arquivo para envio por parte do usuário de um aparelho com sistema operacional Android. Esse arquivo será encriptado por meio do algoritmo DES, gerando um novo arquivo após este processo, e este novo arquivo será enviado via Bluetooth para o destinatário desejado.

CAPÍTULO 4 – APLICAÇÃO DO MODELO PROPOSTO

4.1 - Apresentação da área de Aplicação do modelo

O modelo proposto tem o objetivo de prover segurança na comunicação Bluetooth entre usuários de aparelhos Android. O aplicativo poderá ser usado tanto em transferências corriqueiras, como um envio de uma imagem para um conhecido ou um documento que se deseja manter protegido, quanto em troca de informações sensíveis, como um plano de negócios ou uma definição estratégica.

Ocorre a hipótese de que o aplicativo seja usado sem que haja necessidade de fato, mas o julgamento fica a critério do usuário. O único problema que poderá ocorrer é com arquivos muito grandes, que podem estourar tamanho de buffer ou memória do aparelho durante os processos de encriptação ou decriptação. Como usualmente se utiliza o Bluetooth para transferência de arquivos de tamanho pequeno a médio (uma música, um arquivo PDF ou até uma apresentação de slides costumam não passar de 10 MB), espera-se que não exista problemas na maioria das transferências.

Desta forma, o previsto é que o aplicativo funcione na maioria da vezes em que for utilizado. Não está definido o tamanho limite do arquivo que será transmitido, até porque este valor poderá variar de acordo com o aparelho em uso.

4.2 – Descrição da Aplicação do Modelo

O aplicativo irá, portanto, realizar a encriptação e decriptação dos dados através dos métodos desenvolvidos para que o processo de envio e recebimento do arquivo possa ocorrer normalmente. Para tal, há uma sequência de acontecimentos que devem ocorrer com sucesso:

1. Ao clicar no ícone do aplicativo no menu do aparelho, ele deve abrir normalmente;
2. O usuário remetente do arquivo irá escolher a opção “Encriptar e Enviar Arquivo” no menu inicial. Ela deve abrir normalmente, habilitando uma janela do Explorer do Android;
3. Dentro dessa janela do Explorer, o usuário poderá navegar através dos diretórios do cartão de memória e escolher um arquivo para ser encriptado e enviado. Tudo isso ocorrerá normalmente e sem erros, a não ser que o usuário escolha um arquivo que não pode ser lido (um arquivo de sistema, por exemplo);

4. Após a escolha do arquivo, o aplicativo encriptará a informação. Um novo arquivo deverá ser gerado e enviado para o destinatário desejado, via Bluetooth, em um processo transparente ao usuário;
5. O destinatário, ao receber o arquivo, deverá abrir a aplicação e escolher a opção "Decriptar Arquivo Recebido". Essa opção irá abrir uma janela do Explorer que, assim como a anterior, permitirá que o usuário navegue pelos diretórios até achar o arquivo recebido. O destinatário selecionará o arquivo desejado, e o arquivo será decriptado corretamente pela aplicação, gerando um novo arquivo igual ao original.

Os resultados obtidos foram satisfatórios, visto que as operações de encriptação e decriptação do arquivo com o uso do DES foram realizadas com sucesso pelo aplicativo desenvolvido para o Android. Alguns dos problemas encontrados foram, mas não estão resumidos a:

- Para que o arquivo possa ser lido e transformado em um vetor de bits que será encriptado e transformado em um novo arquivo, é necessário seguir uma sequência de tipos de variáveis, desde o arquivo, passando por streams, até chegar no vetor de bits. Em caso de nova implementação, recomenda-se consultar o código gerado para esse aplicativo para evitar que algum passo seja esquecido;
- O algoritmo que ia ser utilizado inicialmente era o Blowfish, que é aceito pelas bibliotecas do Java. No entanto, no Android, não existe o algoritmo dentro da biblioteca de desenvolvimento, o que motivou a troca da escolha original pelo DES. Em aplicativos futuros, deve-se tomar o cuidado de escolher um algoritmo que esteja nas bibliotecas disponíveis para o Android (ou para o sistema operacional desejado, caso seja outro), caso contrário, será necessário ou trocar o algoritmo utilizado (a opção escolhida nesse projeto, com o uso do DES) ou fazer o desenvolvimento das funções de criptografia dentro do algoritmo;
- Para testes na AVD, foi necessário instalar um plugin que configurasse o Explorer do Android na máquina virtual. Sem essa instalação, o programa sempre falhava durante a execução, já que a AVD não tem um Explorer padrão instalado. Esse erro só ocorreu na AVD, sendo que em testes em um aparelho celular o aplicativo não enfrenta esse problema.

Recomenda-se também o uso do site http://developer.android.com, que possui todas as instruções necessárias para configuração das ferramentas e criação de projetos. As bibliotecas (chamadas de pacotes na documentação oficial) disponíveis estão catalogadas lá, e é possível consultar todas e verificar a forma de utilização. Foi crucial no desenvolvimento desse projeto, no qual foram utilizadas as seguintes bibliotecas e classes, com os seguintes objetivos:

- java.io – Provê para entrada e saída através de streaming de dados, serialização e sistema de arquivos. As seguintes classes desse pacote foram utilizadas:
  - BufferedInputStream – Encapsula um InputStream existente e o coloca em buffer;
  - BufferedOutputStream – Encapsula um OutputStream existente e o coloca em buffer;
  - ByteArrayOutputStream – Um OutputStream especializado para escrever conteúdo em um array de bytes interno;
  - DataInputStream – Encapsula um InputStream existente e lê informações do mesmo. Geralmente pega o resultado de um DataOutputStream;
  - DataOutputStream – Encapsula um OutputStream existente e escreve informações no mesmo;
  - File – Uma representação de um arquivo definido pelo seu *pathname*, que pode ser absoluto (relativo à raiz do sistema) ou relativo ao diretório no qual o programa está rodando. Apesar do nome da classe, ela pode ser usada também para diretórios.
  - FilterInputStream – Encapsula um InputStream existente e realiza alterações na informação de entrada enquanto a mesma é lida. Essas transformações podem ser, por exemplo, um filtro simples de informação ou uma compressão/descompressão em tempo real;
  - FileOutputStream – Um stream de saída que escreve bytes em um arquivo. Se o arquivo existir, ele pode ser trocado ou complementado. Se não existir, um novo arquivo será criado;
  - InputStream – A classe base para todos os input streams (streams de entrada). Um input stream é uma forma de ler informações de uma fonte byte a byte;

- OutputStream - A classe base para todos os output streams (streams de saída). Um output stream é uma forma de escrever informações em um destino byte a byte;
- FileNotFoundException – Exceção utilizada quando o arquivo especificado por um programa não é encontrado;
- IOException – Exceção utilizada para sinalizar um erro geral, ligado a I/O (entrada/saída);
- UnsupportedEncodingException – Exceção utilizada quando um programa tenta usar um conversor de caracteres não disponível.

- security – Provê as classes e interfaces que constituem o framework de segurança do Java. Foram utilizadas as seguintes exceções:
  - InvalidKeyException – Indica uma condição excepcional, causada por uma chave inválida;
  - NoSuchAlgorithmException – Indica que o algoritmo solicitado não pôde ser encontrado.
- javax.crypto – Provê as classes e interfaces para aplicativos que utilizem criptografia, com implementação de algoritmos para encriptação, decriptação e pareamento de chaves. As seguintes classes desse pacote foram utilizadas:
  - Cipher – Permite acesso à implementações de cifras criptográficas para encriptação e decriptação. Para se utilizar as classes de Cipher, deve-se chamar o método getInstance com o nome da operação desejada;
  - CipherInputStream – Encapsula um stream de entrada e uma cifra, para que métodos de leitura retornem informação que tenham vindo do InputStream e sido passados pela cifra;
  - CipherOutputStream – Encapsula um stream de saída e uma cifra, para que métodos de escrita passem o arquivo pela cifra antes de escrever os dados no stream de saída;
  - NoSuchPaddingException – Exceção utilizada quando o mecanismo de *padding* utilizado não é suportado.
- javax.crypto.spec – Provê as classes e interfaces necessárias para especificar chaves e parâmetros para encriptação. Desse pacote, apenas uma classe é utilizada:

    - SecretKeySpec – Uma especificação de chave secreta, e também uma implementação de chave que é independente de aprovisionamento.
- android.app – Contém classes de alto nível encapsulando o modelo geral do Android. Neste projeto, é utilizada apenas a classe de Atividade:
    - Activity – A atividade é uma coisa simples e focada que um usuário pode fazer. Quase sempre interagem com o usuário, então essa classe se encarrega de criar a janela na qual será realizada a interface com o usuário.
- android.content – Contém classes para acessar e publicar data em um dispositivo. A seguinte classe foi utilizada no projeto:
    - Intent – É uma descrição abstrata de uma operação a ser realizada. Um Intent tem seu uso mais significativo na inicialização de atividades, onde ele pode ser visto como a cola entre as atividades, mantendo a sequência lógica das ações.
- android.os – Provê serviços básicos do sistema operacional, passagem de mensagens e comunicação intra-processo com o dispositivo.
    - Bundle – Mapeia valores de strings para tipos Parcelable.
- android.view – Provê classes que permitem interação básica com o usuário. Foi utilizada uma classe desse pacote:
    - View – Representa o bloco básico de interface com o usuário. Uma View ocupa uma área retangular na tela e é responsável por lidar com os eventos gerados pelo usuário. A View é a classe básica para uso de *widgets*, que são usados para criar componentes interativos de interface com o usuário.
- android.widget – Esse pacote contém elementos de interface com o usuário para serem utilizados na tela do aplicativo. Foram utilizadas as seguintes classes:
    - Button – Permite uso de botões, que podem ser clicados ou pressionados, no aplicativo;
    - Toast – Permite que sejam mostradas pequenas e rápidas mensagens para o usuário durante a execução do aplicativo.

Maiores detalhes sobre os pacotes e classes utilizados, além do resto da biblioteca do Android, podem ser encontrados no site mencionado acima.

4.3 – Avaliação Global do Modelo

A encriptação e decriptação de arquivos foi realizada com sucesso por meio do aplicativo desenvolvido, com a devida estabilidade requerida por uma solução de um aplicativo com esse objetivo.

A verificação do sucesso da encriptação foi feita na tentativa de abrir o arquivo criptografado gerado pelo aplicativo. Através dele, foi possível verificar que o conteúdo original não estava legível. Após o envio do arquivo pelo Bluetooth, foi verificado também no receptor da mensagem o arquivo recebido, e foi visto que o conteúdo estava protegido. Apenas após a execução da decriptação do arquivo foi possível verificar o conteúdo claro do mesmo, com a mensagem original. Não foi invadida a comunicação, mas ficou provado que o arquivo transmitido era seguro e protegido de usuários que não possuem a chave.

Quanto à chave, a mesma está inserida no código do aplicativo, descrito na seção 3.3, sendo que o mesmo pode ser visto no Apêndice deste projeto. Para efeitos de teste, foi definida nesse momento como “chave123”.

CAPÍTULO 5 - CONCLUSÃO

5.1 - Conclusões

Apesar de o aplicativo ter funcionado corretamente e cumprido o seu papel, não foram cumpridos todos os objetivos do projeto. Além da troca segura de mensagens, havia sido definido como um dos objetivos específicos a comparação de desempenho entre as opções com a encriptação aplicada e não aplicada, o que não foi realizado a tempo. Desta forma, os objetivos do trabalho foram alcançados de forma parcial.

No entanto, sendo o principal objetivo do trabalho a implementação de um processo que oferecesse maior segurança na troca de mensagens e arquivos via Bluetooth em aparelhos Android, e tendo esse objetivo sido atingido com sucesso, o resultado do projeto ainda assim foi satisfatório.

A vantagem maior do aplicativo desenvolvido é a segurança fornecida pelo mesmo. Não existe nenhuma grande desvantagem no uso do mesmo (apesar de não terem sido realizados testes específicos de performance, em nenhum momento o tempo de espera pela encriptação do arquivo foi alto). Por se tratar de um aplicativo não-comercial, não existe custo para o uso do mesmo.

Pode-se, portanto, dizer que o aplicativo cumpriu seus objetivos, e que o projeto cumpriu a maioria dos seus, sendo esses que foram atingidos os mais cruciais para a completude do projeto.

5.2 - Sugestões para Trabalhos Futuros

Durante a confecção deste trabalho, foram percebidos alguns pontos que abrem espaço para futuras melhorias:

- Trabalhar com arquivos maiores – No aplicativo atual, o tamanho do arquivo é limitado pelo tamanho do vetor de bytes definido em código (10 MB, ou 10485760 bytes). Uma solução possível é tornar as operações de encriptação e decriptação mais dinâmicas, realizando-as ao longo da leitura do arquivo.
- Definição da chave no momento da transmissão – A solução atual utiliza uma chave pré-definida. Para que a chave possa ser definida no momento da transmissão, deve ocorrer uma troca de mensagens que irá validar a chave e garantir sua segurança.

REFERÊNCIAS

BUCHMANN, Johannes A. **Introdução à Criptografia**, São Paulo: Berkeley, 2002

DIFFIE, W. **The First 10 Years of Public-Key Cryptography**. Proceedings of the IEEE, 1988

FERREIRA, F. N. F.. **Segurança da Informação**, Rio de Janeiro: Editora Ciência Moderna Ltda., 2003

KAHN, D. **The Codebreakers: The Story of Secret Writing**. New York: Scribner, 1996

MILLER, Michael. **Descobrindo Bluetooth**, Rio de Janeiro: Campus, 2001

OLIVEIRA, Wilson José de. **Segurança da Informação: Técnicas e Soluções**, Florianópolis: Editora Visual Books, 2001

STALLINGS, William. **Cryptography and Network Security: Principles and Practices**. New Jersey, Prentice Hall, 2003

STALLINGS, William. **Criptografia e segurança de redes: Princípios e práticas.** São Paulo, Prentice Hall, 2008

http://www.espacoacademico.com.br/042/42amsf.htm

http://neuronio-virtual.blogspot.com/2008/10/iphone-o-verdadeiro-celular.html

http://www.tech-faq.com/history-of-cell-phones.html

http://www.baixaki.com.br/info/2140-historia-a-evolucao-do-celular.htm

http://developer.android.com

APÊNDICE

APÊNDICE A – Importação das bibliotecas necessárias para o desenvolvimento e uso do aplicativo:

```
import java.io.BufferedInputStream;
import java.io.BufferedOutputStream;
import java.io.ByteArrayOutputStream;
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.io.UnsupportedEncodingException;
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
import javax.crypto.Cipher;
import javax.crypto.CipherInputStream;
import javax.crypto.CipherOutputStream;
import javax.crypto.NoSuchPaddingException;
import javax.crypto.spec.SecretKeySpec;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;
```

APÊNDICE B – Código criado para escolha de arquivo por parte do usuário e definição do caminho a ser seguido dentro da atividade:

```
public void onCreate(Bundle savedInstanceState) {

	super.onCreate(savedInstanceState);
	setContentView(R.layout.main);

	Button encrypt = (Button)findViewById(R.id.encrypt);
	Button decrypt = (Button)findViewById(R.id.decrypt);

	encrypt.setOnClickListener(new Button.OnClickListener(){
		@Override
		public void onClick(View arg0) {
			Intent intent = new Intent(Intent.ACTION_GET_CONTENT);
			intent.setType("file/*");
			startActivityForResult(intent,0);
		}
	}
	);

	decrypt.setOnClickListener(new Button.OnClickListener(){
		@Override
		public void onClick(View arg0) {
			Intent intent = new Intent(Intent.ACTION_GET_CONTENT);
			intent.setType("file/*");
```

```
                startActivityForResult(intent,1);
            }
        }
        );
}
```

APÊNDICE C – Métodos para cópia do arquivo em buffer para um atributo:

```
public static byte[] toByteArray(InputStream in) throws IOException {
        ByteArrayOutputStream out = new ByteArrayOutputStream();
        copy(in, out);
        return out.toByteArray();
}

public static long copy(InputStream from, OutputStream to) throws
IOException {
        byte[] buf = new byte[BUFFER_SIZE];
        long total = 0;
        while (true) {
            int r = from.read(buf);
            if (r == -1) {
                break;
            }
            to.write(buf, 0, r);
            total += r;
        }
        return total;
}
```

APÊNDICE D – Encriptação/decriptação do arquivo escolhido**:**

```
protected void onActivityResult(int requestCode, int resultCode, Intent
data) {
        DataInputStream origem = null;
        String FilePath;
        File origemFile;

        switch(requestCode){
        case 0: //BOTÃO ENCRIPTAR
            if(resultCode==android.app.Activity.RESULT_OK){
                FilePath = data.getData().getPath();
                String newFilePath;
                Toast toast;

                DataOutputStream destino = null;

                try {
                  origemFile = new File(FilePath);
                  FileInputStream fis = new FileInputStream(origemFile);
                  BufferedInputStream bis = new BufferedInputStream(fis);
                  origem = new DataInputStream(bis);
                } catch (FileNotFoundException e) {
                  toast = Toast.makeText(this, "Erro: Arquivo não
existe!!!", Toast.LENGTH_SHORT);
                  toast.show();
                } catch (SecurityException e) {
                  toast = Toast.makeText(this, "Erro: Permissão de
leitura negada!!!", Toast.LENGTH_SHORT);
                  toast.show();
                }

                int count=0;
```

```
                byte fileContent[] = new byte[12582912];

                try {
                  count = origem.read(fileContent);
                  toast = Toast.makeText(this, count+"",
Toast.LENGTH_SHORT);
                  toast.show();

                } catch (IOException e1) {
                  toast = Toast.makeText(this, "Erro: Não foi possível
ler o arquivo de origem!!!", Toast.LENGTH_SHORT);
                  toast.show();
                }

                try {
                  newFilePath = FilePath+ ".cry";
                  File destinoFile = new File(newFilePath);
                  FileOutputStream fis = new
FileOutputStream(destinoFile);
                  BufferedOutputStream bis = new
BufferedOutputStream(fis);
                  destino = new DataOutputStream(bis);
                } catch (FileNotFoundException e) {
                  toast = Toast.makeText(this, "Erro: Não foi possível
abrir o arquivo de destino para escrita!!!", Toast.LENGTH_SHORT);
                  toast.show();
                } catch (SecurityException e) {
                  toast = Toast.makeText(this, "Erro: Permissão de
escrita negada!!!", Toast.LENGTH_SHORT);
                  toast.show();
                }

                try {

                %Aqui é iniciada a encriptação do arquivo%

                  String keyString = "chave123";

                  byte[] key = new byte[8];

                  key = keyString.getBytes();
                  Cipher cipher = Cipher.getInstance("DES");
                  SecretKeySpec keySpec = new SecretKeySpec(key, "DES");
                  cipher.init(Cipher.ENCRYPT_MODE, keySpec);
                  CipherOutputStream cos_CIFRADO = new
CipherOutputStream(destino, cipher);
                  cos_CIFRADO.write(fileContent, 0, count);
                  cos_CIFRADO.close();

                %Aqui termina a encriptação do arquivo%


                } catch (NoSuchAlgorithmException e) {
                      toast = Toast.makeText(this, "Erro: Não foi possível
utilizar o algoritmo DES!!!   " + e.getMessage(), Toast.LENGTH_SHORT);
                      toast.show();
                } catch (NoSuchPaddingException e) {
                      toast = Toast.makeText(this, "Erro: O mecanismo de
padding selecionado não é suportado!!!   " + e.getMessage(),
Toast.LENGTH_SHORT);
                      toast.show();
                } catch (InvalidKeyException e) {
                      toast = Toast.makeText(this, "Erro: Chave
inválida!!!   " + e.getMessage(), Toast.LENGTH_SHORT);
```

```
                    toast.show();
                } catch (IOException e) {
                    toast = Toast.makeText(this, "Não foi possível
encriptar!!!", Toast.LENGTH_SHORT);
                    toast.show();
                }
            }
            break;
        case 1: //BOTÃO DECRIPTAR
            if(resultCode==android.app.Activity.RESULT_OK){
            FilePath = data.getData().getPath();
            String newFilePath = null;
            byte[] buffer = new byte[12582912]; //10MB + 20%
            CipherInputStream cis_DECIFRADO = null;
            Toast toast;

            DataOutputStream destino = null;
//
            try {
                origemFile = new File(FilePath);
                FileInputStream fis = new FileInputStream(origemFile);
                BufferedInputStream bis = new BufferedInputStream(fis);
                origem = new DataInputStream(bis);
            } catch (FileNotFoundException e) {
                 toast = Toast.makeText(this, "Erro: Arquivo não
existe!!!", Toast.LENGTH_SHORT);
                toast.show();
            } catch (SecurityException e) {
                 toast = Toast.makeText(this, "Erro: Permissão de leitura
negada!!!", Toast.LENGTH_SHORT);
                toast.show();
            }

            try {

            %Aqui é iniciada a decriptação do arquivo%

                String keyString = "chave123";

                byte[] key = new byte[8];

                key = keyString.getBytes();
                Cipher cipher = Cipher.getInstance("DES");
                SecretKeySpec keySpec = new SecretKeySpec(key, "DES");

                cipher.init(Cipher.DECRYPT_MODE, keySpec);
                cis_DECIFRADO = new CipherInputStream(origem, cipher);

            %Aqui é finalizada a decriptação do arquivo%

            } catch (NoSuchAlgorithmException e) {
                 toast = Toast.makeText(this, "Erro: Não foi possível
utilizar o algoritmo DES!!!   " + e.getMessage(), Toast.LENGTH_SHORT);
                toast.show();
            } catch (NoSuchPaddingException e) {
                 toast = Toast.makeText(this, "Erro: O mecanismo de padding
selecionado não é suportado!!!   " + e.getMessage(), Toast.LENGTH_SHORT);
                toast.show();
            } catch (InvalidKeyException e) {
                 toast = Toast.makeText(this, "Erro: Chave inválida!!!   "
+ e.getMessage(), Toast.LENGTH_SHORT);
                toast.show();
            }
```

```
            try {
                  newFilePath = FilePath.substring(0,
FilePath.indexOf(".cry"));
                  toast = Toast.makeText(this, newFilePath,
Toast.LENGTH_LONG);
                  toast.show();
            } catch (IndexOutOfBoundsException e) {
                  toast = Toast.makeText(this, "Arquivo não suportado!!!",
Toast.LENGTH_SHORT);
                  toast.show();
            }

            File destinoFile = new File(newFilePath);
            FileOutputStream fos = null;
            try {
                  fos = new FileOutputStream(destinoFile);
            } catch (FileNotFoundException e) {
                  toast = Toast.makeText(this, "Não foi possível criar o
arquivo!!!", Toast.LENGTH_SHORT);
                  toast.show();
            }

            BufferedOutputStream bos = new BufferedOutputStream(fos);
            destino = new DataOutputStream(bos);
            int count=0;

            try {
                  count = cis_DECIFRADO.read(buffer);
                  toast = Toast.makeText(this, count+"",
Toast.LENGTH_SHORT);
                  toast.show();
            } catch (IOException e1) {
                  toast = Toast.makeText(this, "Não foi possível escrever o
arquivo!!!", Toast.LENGTH_SHORT);
                  toast.show();
            }

            try {
                  destino.write(buffer, 0, count);
            } catch (IOException e) {
                  toast = Toast.makeText(this, "Não foi possível escrever o
arquivo!!!", Toast.LENGTH_SHORT);
                  toast.show();
            } catch (NullPointerException e) {
                  toast = Toast.makeText(this, "Buffer apontando para
null!!!", Toast.LENGTH_SHORT);
                  toast.show();
            }

            try {
                  destino.close();
            } catch (IOException e) {
                  e.printStackTrace();
            }
      }
      break;
      }
```

ANEXOS

ANEXO A – Código básico gerado pelo Eclipse na criação de um projeto Android

```
package com.bluetoothencryption;

import android.app.Activity;
import android.os.Bundle;

public class BluetoothEncryption extends Activity {
	/** Called when the activity is first created. */
	@Override
	public void onCreate(Bundle savedInstanceState) {
	super.onCreate(savedInstanceState);
	setContentView(R.layout.main);
	}
}
```