Introdução de Mecanismos de Segurança em Sistemas de Correio Eletrônico

Paulo Sergio Pagliusi

Dissertação de Mestrado

Instituto de Computação
Universidade Estadual de Campinas

Introdução de Mecanismos de Segurança em Sistemas de Correio Eletrônico

Paulo Sergio Pagliusi

fevereiro, 98

Banca Examinadora:

- Prof. Dr. Cláudio Leonardo Lucchesi (Orientador)
  Instituto de Computação — UNICAMP
- Prof. Dr. Routo Terada
  Instituto de Matemática e Estatística - USP
- Prof. Dr. Paulo Lício de Geus
  Instituto de Computação — UNICAMP
- Prof. Dr. Ricardo Dahab (Suplente)
  Instituto de Computação — UNICAMP

Co-orientador:

- Prof. Dr. Luiz Eduardo Buzato
  Instituto de Computação — UNICAMP

Introdução de Mecanismos de Segurança em Sistemas de Correio Eletrônico

Este exemplar corresponde à redação final da Dissertação devidamente corrigida e defendida por Paulo Sergio Pagliusi e aprovada pela Banca Examinadora.

Campinas, 27 de Fevereiro de 1998.

Prof. Dr. Cláudio Leonardo Lucchesi
(Orientador)

Dissertação apresentada ao Instituto de Computação, UNICAMP, como requisito parcial para a obtenção do título de mestre em Ciência da Computação.

À memória de meu avô,
Godofredo Pagliusi.

Resumo

Este trabalho tem por objetivo apresentar e avaliar um sistema criado para prover segurança ao ambiente de correio eletrônico do editor Emacs: o programa ProtegeMail. Construído na linguagem Emacs LISP, o ProtegeMail consiste em uma extensão dos subsistemas de correio eletrônico do Emacs (VM, RMAIL, MH-E e GNUS). Ele funciona como uma *interface* modular para chamar funções criptográficas existentes nos programas de segurança de *e-mail* PGP e RIPEM. Inicialmente, este trabalho apresenta e discute a funcionalidade, a segurança e os protocolos, padrões e programas relacionados com o conceito de correio eletrônico. Em seguida, apresenta os aspectos específicos da utilização de criptografia em correio eletrônico. Depois, descreve e compara os pacotes PGP e PEM. Também apresenta e analisa a escolha do Emacs como ambiente alvo. Por fim, descreve e avalia o sistema ProtegeMail, através de suas principais funções e da segurança que proporciona ao usuário.

Abstract

The major goal of this work is to present and to evaluate a system created to provide security for the Emacs editor electronic mail environment: the ProtegeMail program. Developed with Emacs Lisp language, ProtegeMail consists of an Emacs e-mail subsystems (VM, RMAIL, MH-E and GNUS) extension. It operates like a modular interface, calling cryptographic functions existing in PGP and RIPEM e-mail security programs. First, this work presents and discusses the functionality, security, and the protocols, standards and programs related with electronic mail concept. It then presents specific aspects of criptography applied to electronic mail. After this, it describes and compares PGP and PEM packets. It also presents and analyzes the choice of Emacs as a target environment. Finally, it describes and evaluates the ProtegeMail system, through its main functions and the security that it provides to the user.

Agradecimentos

Desejo expressar meus sinceros agradecimentos aos Professores Cláudio Leonardo Lucchesi (Orientador) e Luiz Eduardo Buzato (Co-orientador) pela assistência fornecida na preparação deste trabalho.

Além disso, agradeço ao pessoal do Grupo Cripto do Instituto de Computação da UNICAMP, em especial aos colegas Monteiro, Jerônimo, Oliva, Lucas, Keesje e Luiz Eduardo cuja disposição em conhecer e se familiarizar com as necessidades do ProtegeMail foram bastante úteis durante a fase de implementação e testes deste Projeto.

Agradeço também aos membros do "staff" do Instituto de Computação, aos colegas da Marinha e companheiros de curso Núccio, Marques, Lúcio, Silva Roberto, Augusto e Francisco. E ao pessoal da Diretoria de Telecomunicações da Marinha e do Centro de Análises de Sistemas Navais por suas valiosas contribuições.

É importante ressaltar o apoio de meus pais, Adauto e Nilza, a paciência e a dedicação de minha querida mulher, Márcia, e o carinho dos meus filhos, Daniel e Rodrigo, para a consecução deste projeto.

ÍNDICE ANALÍTICO

LISTA DE FIGURAS

LISTA DE TABELAS

GLOSSÁRIO DE SIGLAS

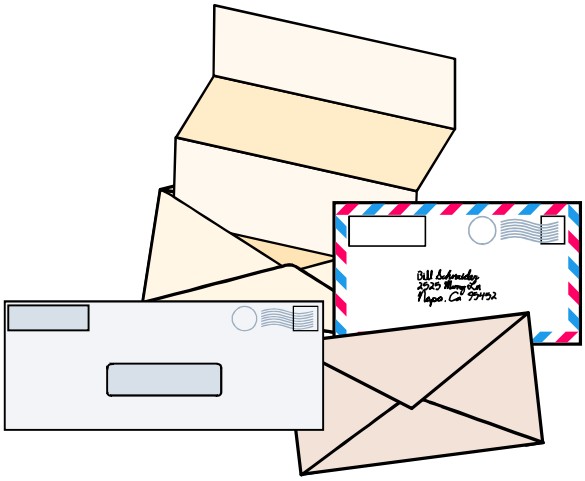
Sigla	*Página(s)*	*Definição*
ASCII	**14**, 124, 127 e 151.	American Standard Code of Information Interchange. Formato de código padrão para troca de informações.
CA	**49**, 60, 61, 87, 88, 93, 141 a 143, 149, 152 e 153.	Certification Authority. Sigla utilizada nas especificações PEM e em outras implementações de pacotes de segurança de *e-mail* para indicar Autoridade de Certificação.
CRL	**153.**	Certificate Revogation Lists. Listas especificadas pelo padrão PEM, contendo informações sobre certificados de chaves revogados.
DEK	**133.**	Data Encryption Key. Denominação dada pelo padrão PEM à chave de sessão única e secreta utilizada para cifrar o conteúdo de uma mensagem.
DES	51, 52, 57, 66, 67, 83, 93, **95** a 101, 104, 135, 138, 140, 169 e 174.	Data Encryption Standard. Algoritmo de ciframento convencional mais utilizado no mundo, ao menos para aplicações não militares. Trata-se de um cifrador de bloco iterativo, com tamanho de chave de 56 bits. Adotado em 1976 como um padrão de ciframento nos EUA, pelo NIST (antigo NBS), foi renomeado para DEA-1, um padrão de ciframento internacional.
DN	**144** e 152.	Distinguished Name. Nome único utilizado como padrão do diretório X.500. Serve para identificar indivíduos, organizações, dispositivos, nomes de listas de discussão, entre outros.
DNS	**18** e 25.	Domain Name System. Sistema destinado a resolver problemas de localização de nomes de máquinas ou de domínios, em uma rede TCP/IP, pelo mapeamento destes nomes a endereços IP, e vice-versa.
DSA	64, 67, **113**, 114 e 138.	Digital Signature Algorithm. Algoritmo de assinatura digital baseado em criptografia de chave pública, proposto pelo NIST.
ECB/CBC/ OFB/CFB	**99**, 123, 138, 140 e 174.	Modos de operação de algoritmos de ciframento convencionais, tais como o DES e o IDEA.
EDI	14 e **56**.	Sigla que simboliza a *Electronic Data Interchange*. Corresponde à transferência, entre computadores, de transações de negócios formatadas.
EMACS	3, 5, **155** a 163, 167 a 169 e 172 a 174.	Editor MACroS. Editor de textos Unix amplamente utlizado. Escolhido como ambiente alvo do sistema ProtegeMail.
E-MAIL	**1** a 42, 46 a 57, 98 a 123, 130 a 146, 151, 152 e 158 a 174.	Electronic Mail. Sigla para mensagem eletrônica. Também utilizada para indicar o endereço de correspondência eletrônica.
FTP	**41** a 70.	File Transfer Protocol. Protocolo de transferência de arquivos na Internet.
IAB	**60**.	Internet Architecture Board. Comitê especial da Internet Society.
IDEA	57, 59, 64 a 68,	International Data Encryption Algorithm. Algoritmo de ciframento

	83, 95, **100** a 104, 118, 123, 124 e 174.	convencional utilizado no PGP, que emprega chaves de 128 bits e operações de adição em módulo de 2^{16}, ou-exclusivo e multiplicação em módulo de $2^{16}+1$.
IETF	**10**, 60 e 62.	Internet Engineering Task Force. Comitê especial da Internet Society.
IK	**133**.	Interchange Key. Denominação dada pelo PEM para a chave utilizada para cifrar a chave DEK.
IPRA	**61**, 141, 143 e 149.	Internet Policy Registration Authority. Autoridade central para a certificação de chaves, na hierarquia definida pelo padrão PEM.
IRTF/PSRG	**60**.	Internet Resources Task Force / Privacy and Security Research Group. Grupo de trabalho especial da Internet Society.
ISO	**10** e 22.	International Standards Organization. Entidade que estabelece padrões internacionais.
ITU-T	**22**.	Sigla para International Telecommunications Union - Telecommunications. Entidade que estabelece padrões internacionais na área de telecomunicações.
KDC	**93**.	Key Distribution Center. Centro de distribuição de chaves do sistema Kerberos, destinado a prover certificação de seus usuários.
LISP	3, **156**, 160, 161, 168 e 174.	LISt Processor. Linguagem interpretada cujos programas consistem de uma série de chamadas de função, com um nome de função seguido de alguns argumentos, todos cercados de parênteses. Os argumentos, por sua vez, também podem ser outras chamadas de função. Disponibilizada pelo Emacs, foi empregada para a elaboração do sistema ProtegeMail.
MD2	**116**, 135 e 138.	Função de espalhamento unidirecional (*one-way hash*) que depende de uma permutação baseada nos dígitos de π. Produz um valor de 128 bits a partir de uma mensagem de tamanho arbitrário.
MD4	114 a **116**.	Algoritmo precursor do MD5, inventado por Ron Rivest. Trata-se de uma função de espalhamento unidirecional.
MD5	59, 64 a 68, 89, **114** a 118, 122, 130, 135, 138 e 141.	Função de espalhamento unidirecional que, a partir de um texto de qualquer tamanho, produz um valor de 128 bits. Utilizada pelo PGP e pelo PEM.
MIC	135 a **138**, 140, 141, 150, 151 e 168.	Message Integrity Code. Código gerado para verificação da integridade e autenticidade de mensagens no padrão PEM.
MIME	40, **62**, 69 e 174.	Multipurpose Internet Mail Extension. Mecanismo para especificar e descrever o formato de conteúdos de mensagens na Internet.
MOAC	**56**.	Message Origin Authentication Check. Sigla para verificação da autenticidade da origem de uma mensagem eletrônica.
MOSS	40, **62** e 174.	MIME Object Security Services. Padrão de segurança de correio eletrônico similar ao PEM, mas com adaptação aos objetos MIME.
MSP	40 e **62**.	Message Security Protocol. Equivalente militar do PEM, padronizado na série de documentos SDN.700 do NIST. Protocolo compatível com o X400 voltado para segurança de correio eletrônico.
MTA	**23**, 25, 26, 33, 70 a 73 e 162.	Message Transfer Agent. Componente de um sistema de correio eletrônico destinado a despachar mensagens, localmente ou entre máquinas. É equivalente

		à uma agência de correio tradicional.
NBS	**95** e 96.	National Bureau of Standards. Nome antigo do NIST.
NIST	**62**, 89, 96, 113 e 173.	National Institute of Standards and Technology. Instituto americano para definições de padrões tecnológicos, supostamente envolvido com as necessidades criptográficas não militares dos EUA. Responsável por tornar padrão o algoritmo DES.
NSA	**62**, 67, 95, 113 e 115.	National Security Agency. Organização oficial de inteligência do governo dos EUA, que conduz pesquisas em criptografia, projeta algoritmos seguros para os militares americanos e técnicas de criptoanálise para a escuta clandestina das comunicações com outros países. Foi criada na década de 40, com a missão de acompanhar e filtrar toda a comunicação dos EUA com outras nações, buscando informações de interesse do governo.
NTEMACS	**159**, 169 e 174.	Versão do Emacs para o sistema operacional Windows NT.
PCA	**61**, 141, 143, 149 e 153.	Policy Certification Authority. Autoridade para a certificação de chaves de CA na hierarquia PEM, subordinada ao IPRA.
PEM	2, 5, 40, **60**, 95, 99 a 116, **132 a 154**, 159 a 161, 168 a 174.	Privacy Enhanced Mail. Conjunto de procedimentos destinados a prover segurança ao correio eletrônico da Internet, especificados nas RFC 1421-1424.
PGP	3, 5, 40, **57** a 68, 89, 95, 99, 104, 111, **118 a 130**, 136, 139, 148 a 154, 161 a 168 e 174.	Pretty Good Privacy. Programa de criptografia que provê recursos de sigilo e de autenticação para mensagens eletrônicas e arquivos. Suas principais funções são ativadas pelo sistema ProtegeMail.
RFC	**10**, 11, 60, 62, 95, 134 e 138.	Request For Comment. Documento que faz parte do processo de padronização de especificações da Internet Society.
RIPEM	2, 61, 133, **144 a 147**, 154, 161 a 168, 170, 172 a 174.	Riordan's Internet Privacy Enhanced Mail. Implementação assimétrica do padrão PEM. Trata-se de um programa de proteção de *e-mail* escrito por Mark Riordan. Suas principais funções são ativadas pelo sistema ProtegeMail.
RSA	59, 62, 64, 67, 85, 89, **104** a 114, 118, 122, 124, 135 a 141, 145 e 174.	Algoritmo de criptografia de chave pública mais amplamente utilizado. R, S e A são as iniciais dos sobrenomes dos autores deste algoritmo. Tanto o PGP quanto o RIPEM utilizam este algoritmo para o gerenciamento de chaves e para a assinatura de *e-mails*.
RSADSI	**62**.	RSA Data Security Inc. Empresa que vende uma grande variedade de produtos de *software* para criptografia de chave pública. É detentora de diversas patentes, tais como a da biblioteca RSAREF.
RSAREF	144 e **145**.	Biblioteca da empresa RSADSI que contém código patenteado de criptografia assimétrica.
S/MIME	40, **62** e 174.	Secure Multipurpose Internet Mail Extensions. Produto do esforço de um consórcio, liderado pela RSADSI e pela Microsoft, que consiste em uma especificação para produzir um padrão Internet de segurança de mensagens eletrônicas no formato MIME.

SHA	64 a 68, 89 e **115**.	Secure Hash Algorithm. Função de espalhamento unidirecional inventada pela NSA, destinada a produzir um valor de 160 bits, a partir de uma mensagem de tamanho arbitrário.
SMTP	5, 22, **24 a 32**, 69, 70, 132, 138 e 172.	Simple Mail Transfer Protocol. Protocolo padrão TCP/IP de transferência de mensagens eletrônicas entre máquinas.
STM	**23** e 162.	Sistema de Transferência de Mensagens. Equivale a um conjunto de MTAs.
TCP/IP	**10**, 18, 22, 25 e 70.	Transmission Control Protocol / Internet Protocol. Protocolo de rede padrão da Internet.
TECO	**157**.	Tape Editor and Corrector. Editor de textos com facilidade de edição de *macro*. Precursor do Emacs.
TGS	**94**.	Ticket_Granting Server. Servidor de *tickets* (ou bilhetes) para acesso a serviços no sistema de autenticação Kerberos.
UA	**22** a 27, 33, 121, 162 e 174.	User Agent. Componente de um sistema de correio eletrônico destinado a prover a *interface* com o ser humano.
UUCP	**69**.	Unix-to-Unix CoPy. Conjunto de programas para transferência de arquivos entre sistemas Unix.
VM/ RMAIL/MH /GNUS	**3**, 5, 155, 161 e 163.	Subsistemas de correio eletrônico do editor Emacs que o sistema ProtegeMail provê suporte de segurança.
X.400	**22**, 69, 132 e 152.	Recomendações do ITU-T que definem o Message Handling System.
X.500	62, **144**, 152 e 153.	Série de recomendações do ITU-T para definir uma estrutura de diretório global, baseada em nomes distintos para localização.
X.509	**62**, 133, **142**, 152 e 174.	Define o relacionamento entre as autoridades de certificação. Faz parte da série X.500.
XEMACS	155, **159**, 161, 167 e 174.	Versão do Emacs com *interface* gráfica.

PREFÁCIO



A Importância do Correio Eletrônico Seguro

[Schneier 95]

A linguagem escrita possibilitou ao ser humano deixar seu testemunho, para ser lembrado ao longo do tempo e da distância. Ela é o principal repositório de cultura de uma civilização. Fazer um testemunho atravessar distâncias é ter poder e influência. Permite a coordenação de atividades envolvendo grandes áreas e populações. A infra-estrutura para se fazer isto é denominada **correio**.

Embora o correio moderno tenha nascido oficialmente com a invenção do selo postal, há 150 anos, ele remonta à Pérsia antiga. Na maioria das vezes considerado uma função governamental, o correio também tem sido uma função exercida pelas igrejas, universidades ou entidades comerciais. Sofreu modificações a cada avanço na tecnologia de transporte. Ele foi uma das primeiras utilizações encontradas para a carruagem, para a estrada de ferro e para a aviação.

Os governos sempre desejaram controlar o correio por razões de segurança de estado e em função da utilização de rendimentos públicos para seu suporte. Desde os tempos do Império Romano, isto se justifica, em parte, pelo papel dos governos na construção de estradas. Entretanto, o controle do correio também tem sido utilizado para justificar a intervenção dos governos em outras áreas. Espionagem industrial e tecnológica se encaixam como exemplos deste tipo de intervenção.

Segurança tem sido uma preocupação do correio desde a Antigüidade, quando as cartas mais importantes eram redigidas pelos soberanos que, não raro, mandavam cortar a língua de seus mensageiros. Segurança ainda é muito importante no correio de hoje.

Segurança no correio significa enviar uma mensagem somente para o destinatário, com confidencialidade. O padrão moderno de confidencialidade no correio consiste no envelope branco simples, dentro de onde quase toda correspondência comercial se movimenta. Uma combinação de cultura, leis e capacidade potencial evidenciam o envelopamento como ato de proteger a confidencialidade.

Somente uma pequena porção de mensagens requer um grau de segurança mais elevado. Para esta porção, são utilizados: dois envelopes, mala postal lacrada de couro ou de lona, ou até mesmo um mensageiro especial com uma maleta algemada no pulso.

Antes do serviço postal moderno, havia muita correspondência que não alcançava seu destino. O emissor reagia, então, encaminhando mensagens importantes múltiplas vezes por múltiplos caminhos. Isto aumentava, por outro lado, a probabilidade do conteúdo da mensagem ser descoberto.

A resposta do correio moderno foi oferecer um serviço diferenciado, com preços idem. Os serviços de primeira classe passaram a contar com opções de retorno ao emissor, no caso da mensagem ou encomenda não atingir seu destinatário. Serviços opcionais para uma entrega segura foram incluídos, tais como: o retorno do recebimento ao emissor e a correspondência registrada e segurada, para prevenir casos de perda ou furto.

Há dezenas de problemas de segurança nos sistemas de correio atuais. Há problemas internos, tais como o furto de correspondência pelos próprios funcionários do correio, muitas vezes em cooperação ilegal com autoridades judiciais ou da segurança nacional. Há problemas externos, tais como o furto de talões de cheque e de cartões de crédito das caixas de correspondência. Há uma enorme gama de fraudes, agravadas também pelo relaxamento das práticas de segurança de boa parte dos usuários, entre elas a não confirmação da mudança de nomes e de endereços.

Mas estes problemas não afetam a real necessidade da utilização continuada dos sistemas de correio. Em parte porque os usuários não reconhecem o risco, em parte porque o aceitam ou então tomam providências para se precaver. Em geral, os problemas de segurança não são graves o suficiente para danificar o sistema.

Há poucas **tecnologias** sustentando a promessa do correio eletrônico. O correio tem prosperado ou ficado debilitado de acordo com a civilização. Ele cresce quando a civilização prospera, e se enfraquece quando a civilização fica em má situação. Ao surgir uma nova era tecnológica, é um ótimo negócio para ambos o fato do outro prosperar.

Quase todas as implementações de correio eletrônico feitas até hoje podem ser descritas como tecnologias de brinquedo. Elas são divertidas e interessantes, contudo não podem ser consideradas sérias. Embora tenham crescido rapidamente, ainda carregam apenas uma pequena quantidade de tráfego de mensagens. Elas não foram projetadas para serem bem sucedidas. Foram apenas concebidas e, com o tempo, aperfeiçoadas. A maioria teve um sucesso muito além do esperado por seus criadores e patrocinadores.

Mas a maioria dos sistemas de correio tradicionais também não foi projetada para ser o que são hoje: eles apenas cresceram. Na verdade, isto faz parte de sua beleza. Os serviços foram adicionados, desenvolvidos e introduzidos lentamente. Às vezes, em resposta a uma necessidade, às vezes em resposta a algum desenvolvimento tecnológico.

O sistema de correio eletrônico está crescendo da mesma maneira. Ele está crescendo pela interconexão de diversos sistemas. Pela exploração do uso do telefone, do fax e até mesmo da tecnologia dos sistemas de páginas. Pela cooperação de milhares de pequenos empreendimentos comerciais com alguns poucos grandes. Em resumo, ele cresce porque seus usuários, sejam indivíduos ou instituições, desfrutam de vantagens econômicas sobre os não-usuários.

Por enquanto, o **papel dos governos** no correio eletrônico tem sido limitado. Nos EUA, o governo patrocinou a Internet, principalmente a pesquisa básica. Em outros países, como no Brasil, por enquanto o governo tem simplesmente ficado indiferente. Sendo assim, o correio eletrônico ainda não é uma função governamental. Ele é mais um esforço cooperativo coletivo. É conduzido mais por acordos e costumes, de modo formal ou informal, do que por leis ou regulamentos.

À medida que o correio eletrônico prove sua importância, pode-se esperar que haja cada vez mais tentativas por parte dos governos de cooptá-lo, da mesma maneira que cooptaram o sistema de correio tradicional. A desculpa que será utilizada estará relacionada à segurança. Mas o real motivo será a manutenção do controle e da influência política.

A força e o valor de um meio de comunicação cresce com o número de suas conexões potenciais e com a velocidade de suas mensagens. O telefone proporciona a habilidade de troca de informações quase instantâneas, mas apenas síncronas e somente entre um número limitado de pessoas ao mesmo tempo. Para a maioria, ele também não deixa nenhum registro. Rádio e televisão são ambos síncronos e do tipo “um-para-muitos”.

O sistema de correio eletrônico consegue ser assíncrono e “muitos-para-muitos”. E, enquanto os sistemas de hoje são limitados a texto, os do futuro prometem uma troca ilimitada de uma mistura de voz, som, fotografia, imagem e dados.

O impacto cultural do correio eletrônico bem sucedido em nossa civilização é potencialmente maior do que o do telefone e rivaliza com o do papel.

Segurança, isto é, confidencialidade, confiabilidade e autenticidade tornam-se essenciais para o sucesso do correio eletrônico. As pessoas não irão utilizar um sistema de correio em que elas não possam confiar para despachar suas mensagens. Embora muitas de suas mensagens poderiam seguir em um cartão postal, elas não irão utilizar um sistema em que todas as suas mensagens devam seguir em um cartão postal. Embora elas não lacrem todos os seus envelopes, elas não irão utilizar um sistema em que não haja envelopes. Embora elas confiem no carteiro, elas não irão utilizar um sistema em que sejam obrigadas a confiar neste funcionário. Embora elas utilizem um sistema de brinquedo para algumas aplicações, elas esperam um sistema maduro e robusto para a maioria de suas aplicações.

O mecanismo fundamental para prover a segurança de mensagens codificadas binárias, em uma rede aberta, consiste na **criptografia**. Ela possibilita a emulação de todos os controles em que temos, historicamente, confiado. A criptografia de chave pública permite a emulação não somente de envelopes, mas também de assinaturas. Embora existam as estruturas primitivas necessárias, não há ainda a também necessária infra-estrutura para ampará-las.

Além de possuir recursos para o armazenamento de uma mensagem, um sistema de correio comum deve ter uma infra-estrutura para seu transporte e distribuição. Ele precisa contar com: caminhões, aviões, agências, um sistema de endereçamento e códigos postais. Do mesmo modo, um sistema eletrônico de envelopes e assinaturas lógicas deve possuir uma infra-estrutura adequada para atribuição de nomes e distribuição de chaves criptográficas.

Assim que o correio eletrônico evoluir de um sistema de brinquedo para um sistema com a devida infra-estrutura, vários de seus problemas serão solucionados. Apesar da segurança não ser o mais significativo dentre eles, ela também não é o menos importante. A **Internet** é o laboratório onde tais problemas são pesquisados e onde os protótipos das soluções são concebidos.

Na condução destes experimentos na Internet, espera-se que seja gerado um modelo análogo ao do correio tradicional. Afinal, a maior parte do que os usuários conhecem sobre segurança e controle em um sistema de correio foi aprendida a partir do modelo de papel. Até o ponto em que o novo sistema seguir este modelo, ele irá explorar os hábitos dos usuários. Ao passo que, se falhar em operar da maneira que os usuários habitualmente esperam, ele provocará erros de utilização.

À medida que as expectativas de segurança dos usuários não forem satisfeitas, o sucesso definitivo do correio eletrônico estará colocado em risco. À medida que o público se tornar cada vez mais apreensivo e preocupado, a intromissão governamental estará sendo solicitada e será justificada.

Este trabalho apresenta um protótipo de solução para alguns dos problemas de segurança de correio eletrônico na Internet: o sistema "**ProtegeMail**".

0. Introdução

O Correio Eletrônico ou simplesmente *E-Mail* (*Electronic Mail*) é, atualmente, o sistema digital de maior alcance para troca de mensagens, tendo se tornado um grande fator de motivação para a interconexão de redes. O *e-mail* tem quebrado as barreiras geográficas. Não se trata apenas da troca de mensagens eletrônicas dentro de empresas: hoje a troca de mensagens eletrônicas entre instituições é corriqueira e intensa. Mais de cem milhões de *e-mails* atravessam as redes de computadores todos os dias [Garfinkel 95].

As vantagens da utilização do *e-mail* são evidenciadas por diversos aspectos. A rapidez, permitindo que uma mensagem atravesse bairros, cidades ou países, sendo entregue ao destino quase imediatamente após o seu envio. Há o fato do recebedor não precisar estar conectado nem precisar interromper suas atividades para receber mensagens, representando uma enorme vantagem sobre o telefone, pois este último exige a disponibilidade imediata do recebedor. Há, também, a possibilidade do recebedor manipular digitalmente a mensagem, da forma que desejar.

Há, ainda, outra característica vantajosa: a concisão. Quando o usuário escreve suas mensagens, costuma organizar melhor suas idéias, resultando em uma forma de comunicação mais breve e produtiva. Além disso, as mensagens eletrônicas podem ser compostas de texto, som, imagem, animação e até código executável, apesar do uso majoritário do *e-mail* ser na troca de mensagens do tipo texto [Cavalcanti 96].

Com todas estas características e vantagens, os sistemas de *e-mail* são o meio de comunicação mais prático da atualidade. O correio eletrônico tem expandido o seu alcance; através dele, há troca de mensagens entre amigos, clientes e fornecedores, parceiros de negócios e até entre órgãos do governo. Seus usuários podem estar conectados por *e-mail*, mesmo quando em viagem, no trânsito, no trabalho ou em casa, independentemente do tipo de sistema de correio eletrônico empregado, de sistema operacional executado ou do computador utilizado.

Mas os *e-mails* trafegam de uma rede a outra até seus destinos, através de canais nem sempre seguros. O maior serviço de *e-mail* existente, o da Internet, não oferece aos seus usuários os recursos mínimos necessários à privacidade e à autenticação das mensagens que encaminha. Deste modo, ele expõe as mensagens dos seus usuários à uma série de ameaças à segurança e a diversos tipos de ataques existentes.

A implementação de mecanismos de segurança fica, invariavelmente, por conta do próprio usuário. Para impedir o acesso indevido ou a manipulação por usuários não autorizados, de seus *e-mails* enviados ou recebidos, é necessário que ele utilize algoritmos de criptografia externos ao seu sistema de correio eletrônico.

Para atender às necessidades de sigilo e de autenticação em sua troca de mensagens pela Internet, o usuário se vê forçado, então, a recorrer aos pacotes de segurança baseados em criptografia atualmente disponíveis, como os programas PGP e RIPEM (sendo este último uma implementação do padrão PEM). O fato destes programas de segurança serem ambientes externos ao sistema de correio eletrônico utilizado pelo usuário implica uma enorme perda de produtividade.

Também demanda a aprendizagem de novos comandos que, muitas vezes, não são nada amigáveis para o usuário final. Assim, para se obter a segurança desejada, além do programa de correio eletrônico, o usuário precisa gastar um tempo adicional para aprender a manusear um programa de criptografia

externo. E também para ficar importando e exportando arquivos, contendo mensagens em claro ou cifradas, de um sistema para outro.

0.1. Objetivo

Este trabalho tem como objetivo tratar o aspecto da segurança na troca de *e-mails*, via Internet, oferecendo uma solução flexível e modular, no nível de aplicação, que embute funções de segurança criptográfica *internas* ao ambiente de correio eletrônico do editor Emacs, com uma boa relação de custo/benefício para o seu usuário: o programa **ProtegeMail**.

Construído na linguagem Emacs LISP, o ProtegeMail consiste, portanto, em uma extensão dos subsistemas de correio eletrônico do Emacs (VM, RMAIL, MH-E e GNUS). Ele funciona como uma *interface* modular para chamar, de dentro do ambiente Emacs, funções criptográficas existentes nos programas de segurança de *e-mail* PGP e RIPEM.

0.2. Importância do Trabalho

Atualmente, com a crescente integração de redes, antes isoladas, à Internet, usuários do mundo inteiro podem ter acesso a qualquer rede interligada, tornando os problemas de segurança mais evidentes. Entre tais problemas, os mais comuns são acessos indevidos às máquinas (especialmente às máquinas Unix) e a captura de pacotes que trafegam pela rede. Assim, o sistema de *E-Mail* da Internet é um alvo fácil de ataque, por não implementar mecanismos de proteção às mensagens. Em geral, elas ficam gravadas em formato texto legível na *mailbox* do usuário ou são encaminhadas por este sem nenhum recurso de segurança. Além disso, não há garantias quanto à integridade dos seus conteúdos nem quanto à autenticidade de seus emissores. Também nada garante que o emissor, posteriormente, não possa negar que tenha enviado uma mensagem que na verdade tenha despachado, caso isto lhe convenha.

Uma das formas de se obter segurança na troca de *e-mails* é ter um programa, no nível de aplicação, que os assine e cifre, já incorporado no próprio sistema de correio eletrônico. Consistindo de uma extensão

oferecendo segurança, este programa precisa, então, garantir: a confidencialidade e a integridade do conteúdo, a autenticidade e o não repúdio da origem da mensagem. Além disso, este programa precisa oferecer não apenas segurança, mas também flexibilidade (por exemplo, podendo ser executado em diferentes plataformas, como Unix ou Windows NT), adaptabilidade (oferecendo o emprego de uma variedade de diferentes algoritmos) e uma boa *interface* com o usuário (contendo menus e comandos amigáveis).

Foi com base nesta idéia que foi desenvolvido o ProtegeMail, que utiliza criptografia para esta finalidade. Dessa forma, obtém-se uma forma segura e amigável de comunicação entre o emissor e o recebedor de uma mensagem, mesmo com ela trafegando por um canal inseguro, como ocorre na Internet.

Acredita-se, portanto, que este programa contribuirá para a melhora deste aspecto relevante que consiste na segurança da troca de *e-mails* via Internet, oferecendo um mecanismo adicional de proteção e autenticação entre as partes que se comunicam por esta rede.

0.3. Organização

Neste capítulo foram apresentadas as características do correio eletrônico e o aspecto da falta de segurança na troca de *e-mails* via Internet, bem como o objetivo e a importância deste trabalho. Para um melhor entendimento, este trabalho encontra-se dividido em quatro partes.

Na Parte I, composta de três capítulos, foi dada ênfase aos assuntos ligados ao Correio Eletrônico. O Capítulo 1 - Funcionalidade - descreve os conceitos básicos, discute sobre a atribuição de nomes, a estrutura da mensagem e a arquitetura do correio eletrônico, com enfoque no protocolo SMTP utilizado na Internet.

O Capítulo 2 - Segurança - descreve o problema da segurança do correio eletrônico. Analisa o problema da coleta e cita as possíveis ameaças à segurança dos *e-mails*. Discorre sobre o gerenciamento das chaves e descreve os principais métodos de ataque existentes. Apresenta os serviços de segurança pertinentes e os padrões e produtos de segurança atualmente em uso. Termina descrevendo as exigências para um sistema de *e-mail* seguro.

O Capítulo 3 descreve o programa mais empregado na troca de mensagens via Internet, intimamente relacionado com o protocolo SMTP: o Sendmail.

Na Parte II, PGP & PEM, composta de quatro capítulos, procurou-se enfatizar a descrição e a comparação entre os dois pacotes de segurança de *e-mail* utilizados pelo ProtegeMail: o PGP e o PEM. Para um melhor entendimento, o Capítulo 4 inicia discorrendo sobre as técnicas criptográficas utilizadas e depois analisa os principais algoritmos criptográficos empregados no ciframento, gerenciamento de chaves e autenticação de mensagens. Em seguida, ressalta os cuidados para a proteção das chaves criptográficas.

O Capítulo 5 descreve o programa PGP e o Capítulo 6, o padrão PEM. O Capítulo 7 compara e analisa estes dois pacotes de segurança de *e-mail*, quanto ao modelo de confiança, aplicações alvo, ciframento, assinatura e gerenciamento de chaves. Termina apresentando os resultados da comparação que levaram à criação do sistema ProtegeMail.

A Parte III é composta de dois capítulos e destina-se a descrever e avaliar o sistema ProtegeMail. O Capítulo 8 - Seleção de Um Ambiente Alvo: o Editor EMACS - descreve e cita os principais comandos do editor de textos que serviu, através de seus subsistemas de correio eletrônico (VM, RMAIL, MH e GNUS), como ambiente alvo do desenvolvimento do ProtegeMail; procura, também, salientar os motivos para tal escolha.

O Capítulo 9 efetua a descrição do programa ProtegeMail, começando pela descrição do sistema que lhe serviu de inspiração; passando depois para suas principais funções e comandos e para a descrição das múltiplas formas de segurança que ele proporciona ao seu usuário. Termina efetuando uma avaliação do ProtegeMail, levando-se em conta as exigências apontadas no Capítulo 2 para um sistema de *e-mail* seguro.

A Parte IV, através do seu Capítulo 10, faz uma conclusão sobre esta dissertação, abordando suas sugestões de pesquisa, autocrítica e trabalhos futuros. O Anexo “A” deste trabalho consiste no Guia do ProtegeMail 2.0. Descreve com detalhes os pré-requisitos, a instalação e a operação deste mecanismo de segurança de correio eletrônico. O Anexo “B” contém um relatório cronológico ou *log* das alterações introduzidas no ProtegeMail, desde o lançamento de sua versão *Beta*.

Parte I

CORREIO ELETRÔNICO

1. Funcionalidade

Este capítulo descreve, na seção 1.1, os conceitos básicos envolvendo correio eletrônico. Na seção 1.2, comenta a atribuição de nomes às *mailboxes*. Apresenta o formato padrão das mensagens, a sintaxe de endereços e a arquitetura do correio eletrônico na Internet, respectivamente nas seções 1.3, 1.4 e 1.5. Termina fazendo, na seção 1.6, uma comparação dos atributos do correio eletrônico com os de outras tecnologias de comunicação.

1.1. Conceitos Básicos

[Sproull 95] [Estrada 93] [Cavalcanti 96]

O correio eletrônico é a facilidade mais largamente utilizada na Internet. A capacidade de se comunicar com pessoas de todo o mundo e de encontrar indivíduos ou grupos que compartilham um interesse ou profissão comum torna-o bastante atraente. Há inúmeras pessoas que se conhecem somente através de trocas de mensagens eletrônicas ou e-*mails*.

Em média, um usuário novo recebe entre 10 e 20 mensagens eletrônicas por dia. Se for um usuário mais ativo, pode facilmente receber entre 50 e 100 *e-mails* por dia. Os usuários mais engajados podem receber 300 ou mais mensagens em um único dia.

Um *e-mail* é um estranho híbrido entre uma escrita à máquina e uma conversa telefônica. Sendo assim, a tecnologia de comunicação eletrônica utiliza processadores de texto e ferramentas de comunicação para prover um serviço de troca de mensagens de alta velocidade.

Qualquer pessoa com acesso a um computador pode criar e enviar uma mensagem eletrônica para outra. Basta que esta outra pessoa possua uma caixa de correspondência ou *mailbox* no mesmo computador ou em outro, que esteja conectado através de uma rede de computadores.

Os computadores desta rede podem estar fisicamente próximos, conectados em uma rede local. Ou podem encontrar-se em diferentes estados, países ou continentes, conectados via telecomunicação de longa distância, formando uma rede permanente ou um elo (*link*) temporário.

Para possibilitar a localização do destinatário, o correio eletrônico utiliza o conceito de endereço. Um endereço de correio eletrônico equivale a um endereço postal: deve conter todas as informações necessárias para o envio de uma mensagem ao destinatário.

Dependendo da sofisticação do programa, a informação enviada pode ser: uma mensagem, um documento, um programa, dados estatísticos, som, imagem, animação ou mesmo uma coleção de mensagens organizadas - uma discussão - reenviada ("*forwarded*") a partir de alguma outra *mailbox*.

Conforme convier ao destinatário, ele pode ler um *e-mail*, editá-lo, salvá-lo, eliminá-lo, movê-lo para outro arquivo, retransmití-lo para alguém e/ou responder ao emissor.

1.2. Atribuição de Nomes

[Garfinkel 95] [Sproull 95]

Cada mensagem eletrônica deve indicar precisamente a *mailbox* a que se destina. Seria ótimo utilizar nomes como: "José da Silva", mas nomes são raramente únicos, e o sistema de correio eletrônico ou E-Mail *(Electronic Mail*) tem informações insuficientes para resolver tais ambigüidades de maneira adequada. Alguns sistemas associam números às *mailboxes*, mas esta técnica obriga o emissor a lembrar-se ou a procurar o número da *mailbox* em uma lista.

Uma solução melhor é adicionar endereços a nomes, da mesma forma que endereços são colocados em envelopes, de modo que nomes e endereços juntos identifiquem uma única *mailbox*. Por exemplo, "José da Silva, do Instituto de Computação, da Universidade Estadual de Campinas, no Brasil" pode ser suficiente para o sistema identificar uma única *mailbox*. E fornecer ao sistema de correio eletrônico a informação necessária para transportar um *e-mail* do emissor ao destinatário.

Levando-se em conta que mais de cem milhões de *e-mails* atravessam as redes de computadores todos os dias, parece evidente que as questões em torno de atribuição de nomes e endereçamento são bem mais complexas do que esta discussão sugere. Planejar um esquema que possa ser utilizado, ao redor do mundo e de maneira harmoniosa, por milhões de pessoas, é um desafio que tem sobrecarregado os comitês que estipulam padrões para o correio eletrônico. Este assunto será detalhado na Seção 1.4.

1.3. Formato da Mensagem

[RFC 822] [RFC 1123] [Frey 94] [Teixeira 96] [Cavalcanti 96] [Costales 94] [Allman 95] [Sproull 95]

Quase todos os sistemas de computação têm seu próprio sistema de correio eletrônico. E, com freqüência, os formatos utilizados nas mensagens são incompatíveis. Porém, há um formato amplamente utilizado em diversas redes de computadores. Este formato é muitas vezes identificado pelo nome do documento que o define: RFC 822[1]. Esta RFC faz parte do processo de padronização da IETF (*Internet Engineering Task Force*) - comitê especial que define os padrões para o protocolo

[1] RFC é uma abreviação para "*Request for Comment*".

TCP/IP (*Transmission Control Protocol / Internet Protocol)*. A RFC 822 é, atualmente, largamente utilizada em toda a parte, em pesquisas e no desenvolvimento de redes mundiais.

1.3.1. O Formato RFC 822

[RFC 822] [Cavalcanti 96] [Costales 94]

O formato de mensagem RFC 822 tornou-se o padrão de transferência "*de facto*" dos sistemas de correio atuais. Este formato torna-se particularmente importante quando as mensagens cruzam de uma rede a outra, através dos *gateways*. Os *gateways* de E-Mail são máquinas que implementam mais de um protocolo de transferência de *e-mail*, resolvendo o problema da interoperabilidade entre sistemas de correio eletrônico distintos.

Tais *gateways* irão reconhecer não apenas seu próprio formato de mensagem, mas também a sintaxe de endereços descrita na RFC 822. Um exemplo consiste no *gateway* TCP/IP para OSI, onde mensagens recebidas utilizando uma porta TCP/IP em uma rede são repassadas para uma porta de outra rede através do MOTIS, o protocolo de correio eletrônico da ISO.

No padrão RFC 822, as mensagens são vistas como tendo um *envelope*[2] e um *conteúdo* (Figura 1.1). O envelope contém a informação necessária para o envio da mensagem. O conteúdo consiste no objeto a ser despachado para o recebedor.

[2] O conceito de *envelope* é análogo ao envelope físico utilizado para o envio de uma carta postal. Se for preciso enviar cópias de um documento para duas pessoas localizadas em lugares distintos, cada cópia do mesmo seguirá em um envelope diferente. O mesmo ocorrerá com uma mensagem eletrônica.

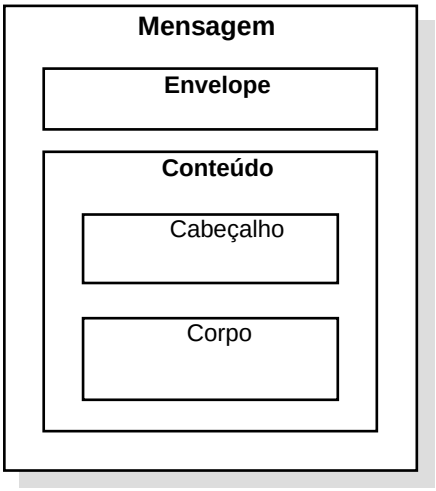


Figura 1.1 - Anatomia de Uma Mensagem [Allman 95].

Este padrão aplica-se somente ao formato e a algumas semânticas do conteúdo da mensagem. Não possui especificações sobre a informação contida no envelope. Entretanto, alguns sistemas de E-Mail podem utilizar informações do conteúdo do *e-mail* para criar o envelope. O padrão RFC 822 facilita a aquisição destas informações por programas. Pelos motivos aqui expostos, o formato RFC 822 será utilizado, de agora em diante, como referência neste trabalho.

1.3.2. Cabeçalho e Corpo da Mensagem

[Allman 95] [Costales 94] [Sproull 95] [Frey 94]

Segue uma mensagem hipotética para ser examinada (Figura 1.2):

From: José da Silva <silva@xingu.dcc.unicamp.br>
To: João de Souza <souza@guttenberg.correionet.com.br>
Cc: ATOM32@DHDCHE11.BITNET (Germanium),
oxygen-lovers@chemistry.mit.edu
Bcc: chaves@printserv.develop.hq.people.com.br
Subject: novos contatos.
Date: Sat, 31 May 97 15:13:38 -0200 (EDT)

Caro João,
Achei suas idéias interessantes e passei adiante seu esboço (através de um "*forward*") para o grupo "oxygen-lovers", nos EUA. Talvez você receba retorno deles, em breve.
Abraços, José.

Figura 1.2 - Exemplo de Uma Mensagem.

O conteúdo de uma mensagem divide-se em duas partes: um *cabeçalho* e um *corpo*, separados por exatamente uma linha em branco. Esta linha é uma parte essencial do formato (Figura 1.3). Muitos sistemas de E-Mail asseguram sua existência de forma automática.

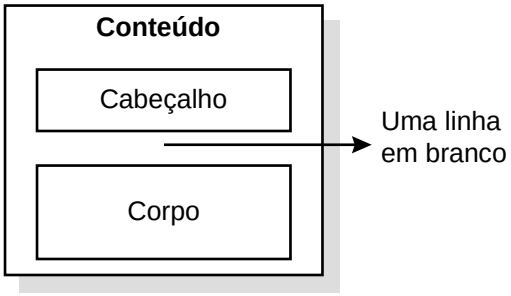


Figura 1.3 - Partes do Conteúdo de Uma Mensagem.

O *cabeçalho* contém informações sobre: o emissor, os recebedores, a data do envio e o assunto da mensagem, entre outras. É organizado em linhas. Cada linha possui um campo formado por uma palavra-chave, seguida de dois pontos de separação e de uma informação. Uma grande parte dos sistemas de E-Mail oferece somente as linhas de cabeçalho essenciais. São elas:

- `To:`
- `From:`
- `Date:`
- `Subject:`

As linhas `From:` e `Date:` são, em geral, inseridas automaticamente, mas pode-se acrescentar mais atributos no cabeçalho de uma mensagem eletrônica. As linhas de cabeçalho mais importantes e seus respectivos significados são apresentados a seguir:

From: Endereço do emissor. Deve haver somente uma destas linhas no cabeçalho. Seu formato será esclarecido na Seção 1.4.

To: Recebedor(es) principal(ais) da mensagem. Esta linha pode especificar mais de um endereço de destino. Neste caso, os endereços devem vir separados por vírgulas ou espaços. Maiores detalhes também serão vistos na próxima Seção.

Cc: Recebedor(es) de cópia. Esta é a cópia carbono (*Cc* é a sigla para *Carbon Copy*) da era eletrônica. Quem estiver nesta linha, recebe uma cópia da mensagem apenas para sua informação.

Bcc: Recebedor(es) de cópia "cega" (*Bcc* é a sigla para *Blind Carbon Copy*). Para o caso de alguém querer enviar uma cópia para terceiros, sem notificar o(s) outro(s) recebedor(es) da mensagem.

Subject: Assunto da mensagem. Este é um texto livre. Deve-se escolher um assunto curto e significativo, sem esquecer-se da pontuação. Embora a presença desta linha não seja obrigatória, seu uso é altamente recomendável.

Date: Data e hora em que a mensagem foi enviada. É importante que a notação da data siga um formato padrão, pois muitos programas podem estar capacitados a classificar mensagens pela data na *mailbox*. A maioria dos sistemas de E-Mail insere linhas *Date:* corretamente formatadas de modo automático (exemplo: 02 Jun 97 16:03:17 -0500). Alguns não permitem a inclusão manual de informações neste campo.

Message-Id: Identificador único da mensagem. É fornecido pelo *host* do emissor. A seguir, observa-se um exemplo:

```
<199706021926.QAA22414@xingu.dcc.unicamp.br>.
```

Esta linha é utilizada para se fazer referências cruzadas automáticas entre mensagens. É inserida pela maioria dos sistemas de E-Mail.

Received: Informação de rastreamento. Utilizada para a análise de problemas no envio de mensagens. Geralmente é composta de múltiplas linhas, que demonstram quando e em que máquinas a mensagem passou.

Resent-From: Endereço da pessoa ou programa de onde a mensagem veio. Campos começando com **Resent -** indicam que a mensagem foi reenviada (*forwarded*). No caso do **Resent-From:**, pela pessoa identificada na linha**.** Pode haver outras linhas, tais como: **Resent-To:** e **Resent-Cc:** .

Reply-To: O endereço da pessoa a quem responder. Em muitos casos, contém o endereço do emissor. Consiste também numa oportunidade para automaticamente reenviar todas as respostas diretamente para outro lugar ou pessoa, sem precisar perguntar pelo endereço do emissor na própria mensagem. Em adição, podem existir outras linhas, tais como **Sender:**, para identificar o emissor, caso este seja diferente do autor demonstrado no endereço da *mailbox* (por exemplo, no caso de **From:** *postmaster*).

O *corpo da mensagem* contém o texto propriamente dito da mesma, em formato ASCII imprimível de 7 bits. As mensagens não-texto, do tipo: voz, fax, imagens, EDI (*Electronic Data Interchange*), código binário, exigem caracteres de 8 bits, precisando ser, de alguma maneira, codificadas em caracteres de 7 bits.

Em grande parte dos sistemas de Correio Eletrônico, não é permitido o envio de binários de 8 bits, sem antes executar alguns procedimentos especiais. O problema consiste no "oitavo bit", que pode ser perdido quando transferido em formato ASCII ou EBCDIC.

Para salvar todos os códigos de controle ou caracteres especiais ligados ao oitavo bit, é necessário codificar antes os binários em 7 bits ASCII ou EBCDIC. E, em seguida, efetuar a transferência e posterior decodificação do formato de 7 bits para o de 8 bits no destino. Na maioria dos sistemas de E-Mail, há programas que realizam tais procedimentos de forma transparente ao usuário.

Alguns *sites* estipulam o tamanho máximo de uma mensagem em 10.000 bytes. Outros, como por exemplo, os da rede Eunet e CSRG Berkeley UNIX, estipulam este limite em 100.000 bytes. Há *sites* que não estipulam limite para o tamanho da mensagem, como os da rede UUNET. Em geral, o limite de 64.000 bytes é um limite seguro a ser observado.

1.4. Sintaxe de Endereços

[Frey 94] [Cavalcanti 96] [Chaves 96] [Brito 97] [Bellovin 94]

A seguir, são observadas três formas de um endereço hipotético de emissor, que podem ser encontradas no cabeçalho de uma mensagem eletrônica:

```
From: silva@xingu.dcc.unicamp.br (José da Silva)
From: José da Silva <silva@xingu.dcc.unicamp.br>
```

```
From: silva@xingu.dcc.unicamp.br
```

A cadeia de caracteres *José da Silva*, embora forneça o nome do emissor, não faz parte do endereço utilizado pelo sistema para rotear mensagens. O sistema de E-Mail tratará todos os três endereços acima de maneira idêntica, porque ele observa apenas a cadeia de caracteres *silva@xingu.dcc.unicamp.br*. Esta cadeia de caracteres consiste no endereço de *e-mail* ou, simplesmente, *e-mail*.

Neste exemplo, o nome *José da Silva* é considerado um comentário que é repassado, sem modificações, pelo sistema de E-Mail. Pode-se colocar o comentário entre parêntesis, ( ), ou o *e-mail* entre os sinais < >. Se o comentário incluir caracteres de pontuação, deve-se empregar a forma entre parêntesis, como a seguir:

```
From: lima@xingu.dcc.unicamp.br (Dr. Antônio Lima)
```

O endereço de *e-mail* será, agora, dividido em duas partes, à esquerda e à direita do caracter arroba, @ (deve haver somente um), para análise.

```
silva    @     xingu.dcc.unicamp.br
```

A parte da esquerda é chamada de parte local e indica o nome da *mailbox*. A parte da direita é o domínio. O domínio especifica onde uma *mailbox* determinada está localizada. Se o endereço não tiver um @, como em:

```
From: silva
```

ele deverá ser um endereço local, significando uma *mailbox* local na organização. A maioria dos sistemas de E-Mail permite a omissão do domínio quando ocorre o envio de mensagem para uma *mailbox* local. Porém, o domínio completo precisará ser adicionado pelo sistema antes do despacho da mensagem.

1.4.1. Nome da Mailbox

[Frey 94] [Cavalcanti 96]

A caixa de correspondência eletrônica ou *mailbox* pode pertencer a um usuário ou a um grupo de usuários. Ou pode ser, também, o lugar para deixar mensagens para alguém com uma função específica, como o *postmaster*.

Ela consiste, portanto, em uma área de armazenamento, onde as mensagens permanecem até que o usuário execute uma ação de eliminação ou de transferência para outra área. A *mailbox* é a versão eletrônica da caixa postal do sistema de correio tradicional.

Há convenções para a determinação da forma de nomes de *mailboxes*, utilizadas para fins especiais:

x-lovers Nomes de *mailboxes* com travessões no interior são provavelmente especiais. Se o nome parecer-se com um que pertença a um grupo de pessoas, ele é possivelmente uma lista de distribuição ou de discussão. O envio de um *e-mail* para nomes deste tipo redistribui a mensagem para todos aqueles que estiverem inscritos na lista.

x-lovers-request Nomes terminando em - *request* são endereços administrativos de listas de discussão. É para onde normalmente são enviados os pedidos de inscrição (*subscribe*) ou de cancelamento de inscrição (*unsubscribe*) em listas. No caso da lista ser moderada, é também para endereços deste tipo que são enviadas as sugestões ao moderador[3].

Postmaster Supõe-se que todo domínio ou *site* possua uma caixa de correspondência *postmaster*. É para onde são encaminhadas as dúvidas e são relatados os problemas envolvendo o sistema de mensagens.

Mailer-daemon Este é o agente do sistema de correio amigável propriamente dito. Mensagens vindas dele são, na maioria, relatos de problemas ocorridos com uma mensagem enviada ou notificação do seu envio. Exemplo de um erro bastante comum: *Host Unknown* ou *User Unknown* - indicando que a mensagem seguiu com um endereço incorreto.

Local%domain A maioria dos sistemas interpreta nomes de *mailbox* com um caracter % no meio como sendo um endereço completo de *e-mail*, onde o primeiro % é substituído por uma arroba, @, e o *mail* é redirecionado de acordo com este novo endereço formado.

[3] O moderador é um editor voluntário que proporciona uma melhor qualidade à um lista de discussão. Ele age como um "filtro humano" para tópicos relevantes.

José.Silva Um crescente número de sistemas de correio vem permitindo o endereçamento do usuário pelo seu nome completo. Para isto, o primeiro nome e os nomes seguintes que compõem o sobrenome devem vir separados por um ponto (.) ou por um sinal de sublinhado (_) .

xyz123AB Há organizações que utilizam números ou códigos para nomes de *mailbox*. Este procedimento é comum em redes EARN e BITNET. Nestes casos, é uma boa prática acrescentar um comentário contendo o nome do destinatário no endereço.

jds Em sistemas UNIX, é bastante comum o uso de partes do nome do usuário para denominar uma *mailbox*. Utiliza-se, também, apelidos ou as iniciais de nome, como *jds* para *José da Silva*.

1.4.2. Domínio

[Bellovin 94] [Frey 94] [Cavalcanti 96] [Chaves 96] [Brito 97]

A parte da direita do sinal de arroba, *@*, é chamada de nome de *domínio*. Indica o computador ou o domínio a que pertence o computador onde a *mailbox* está situada. Para descobrir sua localização, existe o *Domain Name System* (DNS)[4].

O DNS é o sistema que resolve problemas de localização de nomes de computadores ou de domínios, numa rede TCP/IP, pelo mapeamento destes nomes a endereços IP e vice-versa . Uma de suas funções consiste em definir um mecanismo universal para encontrar os computadores destinatários de mensagens endereçadas a um domínio específico. Para isto, divide o nome de domínio onde há pontos, (.), separando-o em segmentos denominados *sub-domínios*.

Deve haver, a princípio, de três a seis sub-domínios. O sub-domínio à direita indica sempre um nível superior em relação ao corrente. O da extrema direita, mais geral, é o primeiro nível ou o *Domínio do nível de topo*, composto normalmente por duas letras indicando o código do país (por exemplo, *br* para Brasil) ou uma designação de rede.

[4] Os nomes DNS formam um conjunto de nomes registrados, obedecendo a um esquema de endereçamento hierárquico, onde o topo é o nome situado mais à direita.

Há várias exceções. Nos Estados Unidos e em alguns *sites* do Canadá, o nível de topo indicando o país é omitido. Utiliza-se, como segmento mais à direita, um código para indicar o tipo de *site*, ou seja, a natureza da organização a que o computador pertence. Este código é, via de regra, utilizado como segundo nível (sub-domínio 1) nos demais países (Tabela 1.1) [5].

Código	*Natureza da Organização do Site*
COM	Comercial
EDU	Educacional
GOV	Governamental
MIL	Militar
NET	Organizações de Rede
ORG	Outras Organizações (Por exemplo: Organizações Não-Governamentais - ONGs).
INT	Internacional
NATO	Pertencentes à NATO

Tabela 1.1- Códigos para Indicar o Tipo do *Site.*

Na Europa, a notação dos sub-domínios é sujeita a um acordo comum entre as redes de cada país. Alguns países, como o Reino Unido, utilizam AC e CO no lugar de EDU e COM, para designar, respectivamente, membros acadêmicos e companhias.

No Brasil, há também uma exceção: omite-se o sub-domínio EDU no caso de *sites* de natureza educacional. Portanto, ao invés de *unicamp.edu.br* temos, no exemplo dado, *unicamp.br*. Deste modo, o nome da instituição passa para o segundo nível (sub-domínio 1).

Em outros países, como a Suíça, há também várias entidades que omitem o segmento apropriado para indicar a natureza da organização. Por exemplo: *who.ch*[6](*World Health Organization*, Suíça), que deveria ser, pela regra geral, *who.org.ch* .

[5] No Brasil, há uma medida do Comitê Gestor da Internet (CGI) para dividir o sub-domínio COM, responsável por 90 % dos endereços brasileiros, em oito novas terminações. Entre elas, temos: INF, ETC e PSI.

[6] O código internacional da Suíça é CH, derivado de "Confederatio Helvetica" (latim), "Confédération Hélvetique" (francês) ou "Schweiz" (alemão).

O nome da organização é, exceções à parte, normalmente utilizado como terceiro nível (sub-domínio 2), como em *embratel.net.br* (Empresa Brasileira de Telecomunicações). É a própria organização que sugere o nome deste sub-domínio que, estando disponível, será concedido[7]. Os demais segmentos, à esquerda, detalham ainda mais a informação sobre o recebedor.

Como quarto nível (sub-domínio 3), é comum empregar-se o nome do computador ou, se houver um quinto nível, o nome da área dentro da organização (nome do Departamento, entre outras áreas) em que se situa o computador. Mas este segmento pode ser omitido.

O quinto nível (sub-domínio 4), também não obrigatório, indica normalmente o nome da máquina que provê suporte para o *site*. Ou, caso haja um sexto domínio, indicará o nome da sub-área (dentro da área da organização) a que pertence o computador.

O sexto nível (sub-domínio 5), se existir, indicará o nome do computador. A sintaxe genérica para os endereços de *e-mail* é:

mailbox@sub-domínio*n*...sub-domínio*2*.sub-domínio*1*.domínio-nível-de-topo

Na mensagem do exemplo (Figura 1.2), os endereços:

silva@xingu.dcc.unicamp.br,
souza@guttenberg.correionet.com.br,
oxygen-lovers@chemistry.mit.edu,
chaves@printserv.develop.hq.people.com.br

podem ser divididos em níveis ou domínios, conforme as regras expostas (Tabela 1.2).

[7] A partir do terceiro nível, a atribuição dos níveis seguintes e a escolha de seus nomes são de exclusiva responsabilidade da instituição.

Nível	*Domínio*	*Descrição*
Local	**Nome da Mailbox:**	*silva, souza, oxygen-lovers* e *chaves*.
6º.	**Sub-domínio 5:**	*printserv* é nome de computador.
5º.	**Sub-domínio 4:**	*.develop* indica a sub-área, referindo-se à Divisão de Desenvolvimento.
4º.	**Sub-domínio 3:**	*xingu* e *guttenberg* são nomes de máquinas. Já *.hq* indica a área *Headquarters* dentro da empresa *People*.
3º.	**Sub-domínio 2:**	*.dcc* e *chemistry* indicam o departamento. Já *.correionet* e *people* seguem a regra geral e indicam a instituição.
2º.	**Sub-domínio 1:**	*.unicamp* e *.mit* indicam a instituição. Já *.com* segue a regra geral, indicando o tipo de *site*.
1º.	**Domínio de Topo:**	*.br* indica o país e *.edu* o tipo do *site* (localizado nos EUA).

Tabela 1.2 - Exemplo de Divisão de Endereços de E-Mail.

1.5. Arquitetura do Correio Eletrônico

[Allman 95] [Comer 95] [Cavalcanti 97] [RFC 821] [RFC 1123] [Sproull 95] [Teixeira 96] [Costales 94] [Cavalcanti 96]

Um sistema de correio eletrônico é formado por componentes que interagem para haver troca de mensagens. Esta troca pode ocorrer entre usuários de uma mesma máquina; ou entre usuários situados em máquinas diferentes, ligadas em rede. Neste caso, pode haver alguma incompatibilidade entre os sistemas.

Assim, um dos maiores problemas com sistemas de correio eletrônico ocorre quando os sistemas de comunicação rodando nas diferentes máquinas não são compatíveis. Para solucionar este problema, foram desenvolvidos protocolos específicos para tratar as comunicações de correio eletrônico.

Os dois grandes fornecedores de padrões para sistemas de correio eletrônico são:

- a ISO/ITU-T (*International Standards Organization / International Telecommunications Union - Telecommunications*), através do protocolo X.400 (X.400 - *Recommendations for Message Handling Systems*); e

- a Internet, através do protocolo SMTP (*Simple Mail Transfer Protocol*).

Os protocolos de transferência de mensagens X.400 e o SMTP são considerados, respectivamente, o padrão "de juri" e o padrão "de facto" para protocolos de correio eletrônico.

A Internet é notoriamente conhecida como a maior rede de computadores do mundo. Ela baseia-se no conjunto de protocolos TCP/IP. No TCP/IP, a transferência de *e-mail* ocorre através do protocolo SMTP. Sendo assim, em função da infra-estrutura disponível na Internet, neste trabalho será enfocado o SMTP - o protocolo de correio eletrônico mais utilizado em todo o mundo.

1.5.1. Componentes Básicos

[Cavalcanti 97] [Costales 94] [Teixeira 96] [Cavalcanti 96]

Os sistemas de E-Mail são constituídos de duas partes distintas. Uma parte provê a *interface* com o ser humano. Esta parte é chamada de *Agente Usuário* (ou UA, de *User Agent* [8]). O UA consiste no programa de correio eletrônico que permite ao usuário confeccionar, enviar, receber e ler mensagens, bem como manipular a caixa de correspondência ou *mailbox*. Equivale à caneta e ao papel utilizados na composição de uma carta convencional.

A outra parte é a que provê o envio da mensagem ao recebedor, chamada de Agente de Transferência de Mensagem (ou MTA, de *Message Transfer Agent*). O MTA é o programa que encaminha a mensagem para a máquina de destino, utilizando um protocolo de transferência de mensagens.

Deste modo, o MTA[9] é o equivalente eletrônico de uma agência de correio tradicional. Se o recebedor for local (na mesma vizinhança ou na mesma máquina), somente um único MTA ou uma única agência estará envolvida. Se for distante, a mensagem será despachada da agência local (MTA local) para outra distante (MTA remoto), para ser entregue ao recebedor (*mailbox*). Um Sistema de Transferência de Mensagens (ou STM) é a denominação dada a um conjunto de MTAs (Figura 1.4).

[8] Algumas referências empregam a sigla MUA, de *Mail User Agent*., no lugar de UA.

[9] As máquinas que executam programas MTA são conhecidas como servidoras de E-Mail.

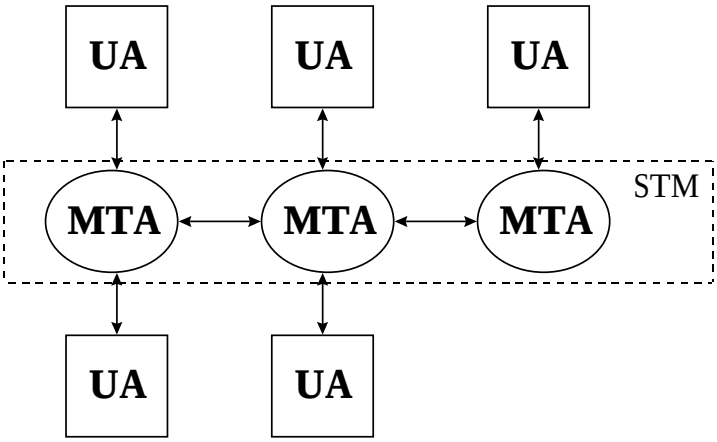


Figura 1.4 - Modelo Geral dos Sist. de Correio Eletrônico [Cavalcanti 96].

Os sistemas de correio eletrônico constituem serviços da camada de aplicação. O modelo empregado pelos sistemas de correio eletrônico segue o paradigma cliente/servidor para desenvolvimento de aplicações distribuídas.

1.5.2. Procedimentos do SMTP

[Teixeira 96] [RFC 821] [RFC 1123] [Cavalcanti 97]

O SMTP consiste em um protocolo puro da camada de aplicação, não se preocupando, portanto, com os serviços de transporte que o apóiam. Trata-se de um protocolo simples, orientado a textos e projetado para transferir mensagens de maneira confiável e eficiente.

A estrutura de uma máquina que implementa o protocolo de transferência de mensagens SMTP inclui (Figura 1.5):

- uma ou mais *Áreas para Enfileiramento de Mensagens* - consistem nas *áreas de spool*, ou seja, são as áreas onde as mensagens em trânsito são armazenadas em uma fila de submissão, visando futura transmissão ou remoção para outra área.

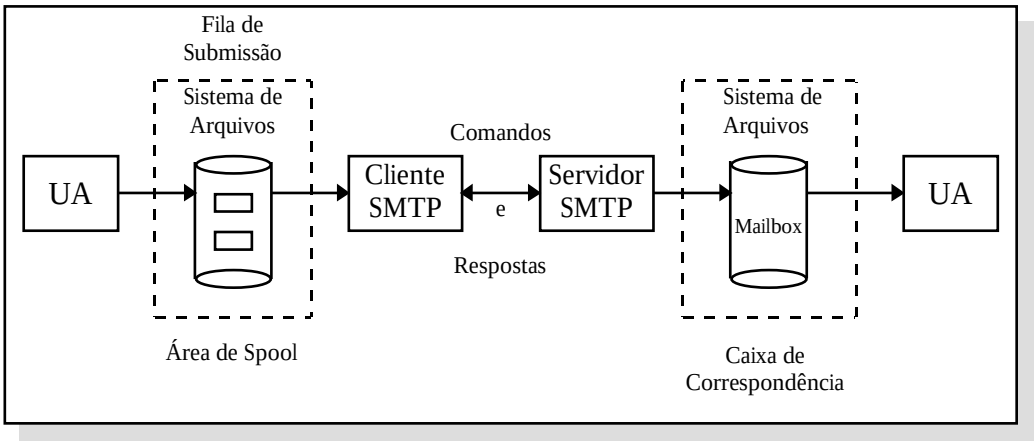


Figura 1.5 - Modelo Funcional do SMTP [Cavalcanti 97].

- um ou mais processos *Daemons* - programas executando na retaguarda (em *background*), para entrega e recebimento de mensagens, de forma a permitir que o usuário continue com suas atividades normais de uso da máquina . Durante o diálogo entre duas máquinas para transferência de uma mensagem, a máquina de origem da mensagem age como c*liente SMTP*. Por sua vez, a máquina de destino age como *servidora SMTP*, aceitando as mensagens que serão colocadas, posteriormente, na *mailbox* do recebedor.

- as *mailboxes* dos usuários - como já exposto, são as áreas de armazenamento onde as mensagens permanecem até que o usuário as elimine ou as transfira para outra área.

O SMTP normalmente interage com o sistema de correio eletrônico e não diretamente com o usuário. Portanto, ele fica à parte de qualquer transferência local para uma determinada máquina.

Deste modo, o SMTP somente entra em cena quando uma mensagem deve ser transferida para uma máquina diferente ou quando do recebimento de uma mensagem remota.

Sendo assim, após o usuário escrever uma mensagem e solicitar seu envio através do UA, havendo necessidade de encaminhar cópia para alguma máquina remota, este a deposita na área de *spool* local para mensagens de saída.

O MTA vasculha periodicamente a área de *spool* citada em busca de mensagens não despachadas. Quando encontra alguma, ele extrai do seu envelope as informações necessárias para seu envio, via SMTP, para a máquina remota destinatária.

No SMTP, antes de enviar uma correspondência eletrônica, o processo *cliente* faz a conversão do nome da máquina destino para seu respectivo endereço IP, através do serviço de diretórios DNS. Dessa forma, é estabelecida uma conexão TCP/IP (aberta em uma porta bem conhecida - no caso, a porta 25) com *o* processo *servidor* em execução naquela máquina.

Se a conexão tiver êxito, o processo *cliente* envia uma cópia da mensagem para o *servidor* remoto. No caminho percorrido por uma mensagem que trafega na rede, um caminho reverso também é executado, tornando possível a notificação ao emissor original sobre possíveis falhas.

O *servidor* armazena temporariamente a cópia em uma área de *spool* para mensagens de entrada. Quando *cliente* e *servidor* concordam com o fato da cópia da mensagem ter sido corretamente transferida e armazenada, o *cliente* remove a cópia local de sua área de *spool*. E o *servidor* também remove sua cópia, transferindo-a, normalmente, para a *mailbox* de destino.

Deste modo, tudo o que o usuário precisa fazer é executar uma *interface* (UA) para depositar ou receber mensagens da área de *spool*. Todas as transferências são tratadas por processos em execução na retaguarda (MTA). As *filas de submissão*, tanto de entrada quanto de saída de mensagens, situam-se entre o sistema de E-Mail local e as partes *cliente* e *servidor* do SMTP.

O processo *cliente* SMTP está ligado à inicialização da transferência de mensagens para outros sistemas de correio eletrônico. O processo *servidor* SMTP cuida do recebimento de mensagens oriundas dos sistemas remotos (Figura 1.6).

Caso a conexão não tenha sucesso - por não ter sido estabelecida ou devido a falha ocorrida durante a comunicação:

- o processo de transferência anota a hora em que tentou a entrega e termina a sua execução;
- o processo de transferência *em background* vasculha periodicamente a área de *spool* procurando mensagens não despachadas;
- quando encontra alguma, ele tenta despachá-la para o usuário recebedor; e
- passado um longo período de tentativas (por exemplo, três dias), ele envia uma mensagem de erro para o emissor.

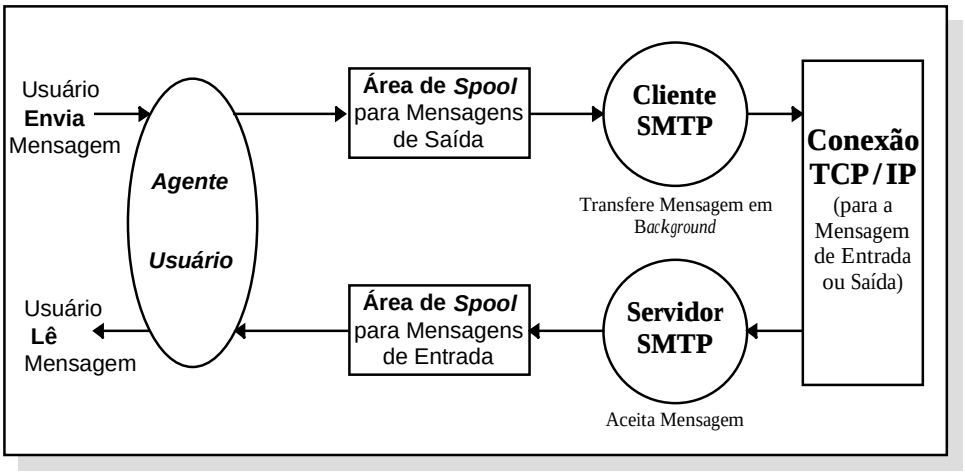


Figura 1.6 - Componentes de Um Sistema de Correio Eletrônico SMTP.

1.5.3. A Transferência de Mensagens pelo SMTP

[Allman 95] [Teixeira 96] [Cavalcanti 97] [RFC 821] [RFC 1123] [Cavalcanti 96]

A transferência de e-mails através do SMTP envolve uma troca de pacotes especiais PDU's (*Protocol Data Units*) conhecidos por comandos e respostas SMTP. Desta forma, a comunicação entre o cliente e o servidor SMTP é um diálogo controlado pelo cliente. Este envia um comando ao servidor que lhe retribui com uma resposta.

Neste diálogo, todos os comandos são compostos por códigos de quatro caracteres alfabéticos, não importando se maiúsculos ou minúsculos. Já as respostas aos comandos SMTP são compostas por um código numérico de três dígitos transmitidos como três caracteres alfanuméricos, seguido por um texto.

A troca de mensagens pelo SMTP está baseada no seguinte modelo de comunicação: como resultado de um pedido de envio de mensagem eletrônica feito por um usuário a um UA, o cliente SMTP da máquina local abre uma conexão de transporte com o servidor SMTP da máquina recebedora [10].

Tão logo iniciado o canal de transmissão solicitado pelo cliente, o servidor SMTP lhe envia a resposta 220 (*Ready for mail*). Em seguida, o cliente envia o comando HELO se apresentando e o servidor devolve a resposta 250, seguida de sua identificação. Uma vez estabelecido o canal de comunicação, o

[10] A máquina recebedora que roda o servidor SMTP pode ser ou a máquina destinatária ou uma intermediária.

cliente SMTP envia um comando MAIL indicando o emissor da mensagem eletrônica. Se o servidor SMTP puder receber e-mails, ele encaminha uma resposta 250 (*OK*).

O cliente SMTP envia um comando RCPT identificando o recebedor existente no e-mail. Se o servidor SMTP puder aceitar mensagens para o recebedor em pauta, ele devolverá uma resposta 250 (*OK*); caso contrário, o servidor responderá rejeitando aquele recebedor - por exemplo, através do código 550 (*No such user here*)[11]. O cliente e o servidor SMTP podem negociar vários recebedores[12]. Após todos os recebedores terem sidos negociados, o cliente SMTP encaminha o comando DATA, informando ao servidor que está pronto para transferir o conteúdo do e-mail.

O servidor envia a resposta 354 (*Enter mail, end with* .), avisando que está pronto para recebê-lo. O cliente SMTP envia, então, o conteúdo da mensagem eletrônica, indicando seu término através de uma seqüência especial de caracteres[13]. Se o servidor SMTP processar com sucesso os dados do e-mail, ele responderá com o código 250 (*OK*).

Para encerrar, o cliente SMTP envia um comando QUIT, respondido pelo servidor SMTP com um 221 (*Service closing transmission channel*), concordando com o término. O diálogo ocorre passo a passo (Figura 1.7).

[11] Neste caso, apenas o recebedor em pauta é rejeitado. O servidor SMTP não rejeita toda a transação.

[12] Tantos quantos forem os destinatários nos campos To:, Cc: e Bcc: constantes do cabeçalho do e-mail.

[13] Esta seqüência consiste, normalmente, de uma linha com um ponto único (<*CR*><*LF*>.<*CR*><*LF*> ou <\n>.<\n>, respectivamente em ambiente MS-DOS ou Unix).

```
S:  220      guttenberg.correionet.com.br    Simple Mail Transfer Service Ready
C:  HELO     xingu.dcc.unicamp.br
S:  250      guttenberg.correionet.com.br
C:  MAIL     From: <silva@xingu.dcc.unicamp.br>
S:  250      ok
C:  RCPT     To:     <souza@guttenberg.correionet.com.br>
S:  250      ok
C:  DATA
S:  354      Enter mail, end with .
C:  From:    José da Silva <silva@xingu.dcc.unicamp.br>
C:  To:      João de Souza <souza@guttenberg.correionet.com.br>
C:  Cc:      ATOM32@DHDCHE11.BITNET(Germanium),oxygen-lovers@chemistry.mit.edu
C:  Subject: novos contatos.
C:  Date:    Sat,  31 May 97 15:13:38 -0200 (EDT)
C:
C:  Caro     João,
C:  Achei suas idéias interessantes e passei adiante seu esboço (através de um “forward”)
C:  para o grupo “oxygen-lovers”, nos EUA. Talvez você receba retorno deles,  em  breve.
C:  Abraços,                                 José.
C:  .
S:  250      ok
C:  QUIT
S:  221      guttenberg.correionet.com.br    Service closing transmission channel
```

Figura 1.7 - Exemplo de Diálogo Cliente/Servidor SMTP

Cada comando SMTP gera uma resposta. As respostas garantem, deste modo, a sincronização das solicitações e da execução de suas respectivas ações na transferência de um e-mail. O primeiro caracter de cada resposta indica se a resposta é boa, ruim ou incompleta. Há cinco valores possíveis (Tabela 1.3).

1º. Caracter do Código de Resposta	***Significado***
1	Indica que o comando foi aceito, mas requer a confirmação da informação contida na resposta.
2	Comando aceito e ação requisitada completada com sucesso.
3	Comando aceito, porém aguarda-se mais informações para a requisição ser atendida.
4	Ocorreu um erro e a ação requisitada não será executada, mas poderá ser solicitada outra vez.
5	Ocorreu um erro e a mesma ação não deverá ser novamente solicitada.

Tabela 1.3 - Significado do Primeiro Caracter do Código de Resposta SMTP.

O segundo caracter da resposta indica a categoria da informação. Há quatro valores possíveis (Tabela 1.4):

2º. Caracter do Código de Resposta	***Significado***
0	Erro de sintaxe, de comando não especificado ou de comando supérfluo.
1	Informação de *status* ou de ajuda.
2	Resposta referente à conexão.
5	*Status* de uma transferência ou de uma ação solicitada.

Tabela 1.4 - Significado do Segundo Caracter do Código de Resposta SMTP.

Comando SMTP	*Descrição*
HELO	Identifica o cliente SMTP para o servidor SMTP.
MAIL	Inicia uma transação de *e-mail* <*Reverse-path*>.
RCPT	Identifica o recebedor (individual) do conteúdo do *mail*.
DATA	O servidor SMTP trata as linhas seguintes como sendo o conteúdo do *e-mail* enviado pelo cliente SMTP.
SEND	Envia o conteúdo do *mail* para um ou mais terminais.
SOML	envia o *mail* para um ou mais terminais, se o usuário recebedor estiver ativo; senão, envia para sua *mailbox*.
SAML	Envia o conteúdo do *mail* para um ou mais terminais e também para as *mailboxes* dos usuários recebedores.
RSET	Indica que a transação corrente está para ser abortada.
VRFY	<*User name*>. Pede ao servidor SMTP para confirmar se o argumento identifica um usuário.
EXPN	<*Mailing list*>. Pede ao servidor SMTP para confirmar se o argumento identifica uma lista de discussão e, caso identifique alguma, pede a relação dos seus membros .
HELP	Solicita informação de ajuda ao servidor SMTP, pedindo os comandos disponíveis e instruções para utilizá-los .
NOOP	Não indica nenhuma ação; apenas solicita uma resposta *ok* do servidor SMTP.
QUIT	Especifica que o servidor SMTP deve enviar uma resposta *ok* e fechar o canal de transmissão.
TURN	Solicitação de inversão de papéis entre cliente e servidor SMTP. O servidor SMTP pode negar ou enviar uma resposta *ok* e, então, assumir o papel do cliente.

Tabela 1.5 - Os Comandos SMTP.

Os procedimentos para transferência de mensagens, os comandos SMTP[14] (Tabela 1.5) e suas respectivas respostas (Tabela 1.6) são definidos na RFC 821, que especifica o SMTP.

[14] Além dos comandos já citados, o SMTP especifica também os seguintes comandos: VRFY, EXPN, SEND, SOML, SAML, TURN, RSET, NOOP e HELP.

Código de Resposta	***Descrição***
500	Erro de sintaxe, comando não reconhecido.
501	Erro de sintaxe nos parâmetros ou argumentos.
502	Comando não implementado.
503	Seqüência incorreta de comandos.
504	Parâmetro de comando não implementado.
211	*Status* do sistema ou resposta de ajuda do sistema.
214	Mensagem de ajuda.
220	(Domínio) Serviço pronto.
221	(Domínio) Serviço fechando o canal de transmissão.
421	(Domínio) Serviço indisponível, fech. canal transmissão.
250	Ação pedida concluída com sucesso.
251	Usuário não local; o servidor reenviará para<*forward-path*>.
450	Ação não realizada: *mailbox* indisponível (Ex. cheia).
550	Ação não realizada: *mailbox* indisponível (Ex. não achada).
451	Ação pedida abortada: erro no processamento.
551	Usuário não local; tente reenviar para<*forward-path*>.
452	Ação não realizada: memória insuficiente.
552	Ação pedida abortada: excedeu alocação de memória.
553	Ação não realizada: nome de *mailbox* inválido.
354	Inicie entrada do conteúdo do *mail*; termine com ponto.
554	A transação falhou.

Tabela 1.6 - Códigos de Resposta SMTP por Grupos de Função.

1.5.4. A Expansão de Apelidos

[Teixeira 96]

A maioria dos sistemas de correio eletrônico possui um programa para transporte de mensagens que inclui um mecanismo de expansão de apelidos de correspondências, permitindo a conversão de identificadores utilizados em endereços de e-mail para novos endereços (um ou mais).

O uso de apelidos aumenta a funcionalidade e conveniência dos sistemas de correio eletrônico. Permite que um *site* associe grupos de usuários por meio de um único identificador. Permite também que um usuário possua múltiplos identificadores para correspondência eletrônica, incluindo posições e

abreviações, através do mapeamento de um conjunto de identificadores para um único indivíduo. O uso de apelidos também possibilita o estabelecimento de *um expansor de correspondências*. Este programa recebe uma mensagem de entrada e a retransmite para um conjunto de recebedores associados a um único identificador, ou seja, para uma *lista de correspondência eletrônica*.

Após a preparação de uma mensagem pelo usuário e a identificação de cada recebedor ter sido estabelecida, geralmente o UA consulta os apelidos locais em um banco de dados para substituir o nome do recebedor pelo nome mapeado. Depois desta substituição, ele repassa a mensagem ao MTA rodando em *background*. Recebedores não *mapeados*, ou seja, sem apelidos, não sofrem alteração. Tais apelidos também podem ser utilizados para converter nomes de usuários locais para mensagens recebidas remotamente. Assim, tanto mensagens de entrada quanto de saída passam pelo mecanismo de expansão de apelidos (Figura 1.8).

Deve-se tomar cuidado com a expansão de apelidos. Se o apelido especificar um endereço inválido, o emissor irá receber uma mensagem de erro. Ou se dois *sites* especificarem apelidos conflitantes, os sistemas de correio eletrônico de ambos podem entrar em *loop*. Por exemplo, o *site* A pode especificar o endereço estação1 como sendo o endereço estação2 no *site* B. E o *site* B, por sua vez, pode traduzir o endereço estação2 para endereço estação1 no *site* A.

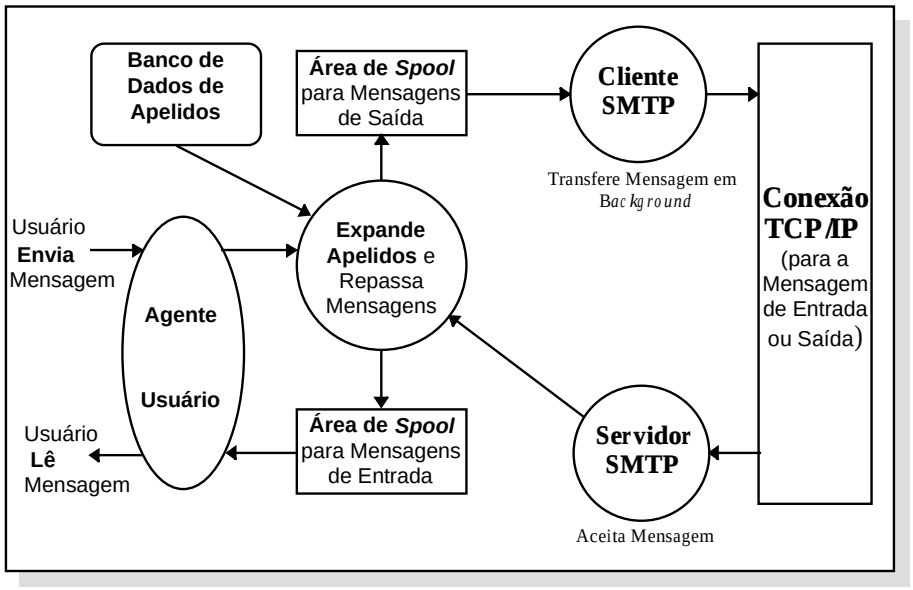


Figura 1.8 - Um Sistema de Correio Eletrônico Completo.

1.6. Atributos Tecnológicos do Correio Eletrônico

[Sproull 95] [Chaves 96]

A tecnologia do correio eletrônico possui seis características que a diferem de outras tecnologias de comunicação. A primeira delas consiste no fato do correio eletrônico ser *assíncrono*. Emissores e recebedores não precisam estar presentes ao mesmo tempo em um determinado canal de comunicação. Ambos podem enviar e receber suas mensagens conforme acharem mais conveniente.

A segunda diz respeito ao fato do correio eletrônico ser *rápido*. Uma mensagem eletrônica pode ser transmitida em segundos ou minutos atravessando um prédio, um continente ou dando a volta ao mundo. As respostas podem fluir com a mesma rapidez. A velocidade não é apenas uma questão de conveniência. Ela possibilita conversação de longa distância, tomada de decisões e uma série de outras interações requerendo curto tempo de resposta.

A terceira característica está ligada ao aspecto do correio eletrônico ser *baseado em texto*. As mensagens, em sua maioria, transmitem caracteres, e não imagens ou sons, embora haja alguns programas de *e-mail* permitindo o envio destes últimos. Além de troca de mensagens, o texto na comunicação eletrônica permite também a troca de documentos.

O mais importante desta característica é que as mensagens parecem-se muito umas com as outras. Não carregam consigo o tom de voz, a inflexão, o sorriso, o olhar de quem as redigiu. Carecem, portanto, de informação social composta de regras e costumes que usualmente regulam as comunicações. Uma tentativa de suprir esta carência consiste no Código de Etiqueta da Rede, desenvolvido pela comunidade da Internet: o *Netiquette Code*.

Os Ícones da Emoção (*Smileys* ou *Emoticons*) têm também como objetivo introduzir alguma comunicação social no texto escrito (Figura 1.9). Outra convenção tácita consiste no uso de maiúsculas, indicando que o emissor está “gritando” um certo trecho da mensagem.

```
:-)     Quem escreveu está brincando ou sorrindo: não leve a sério.
:-|     O leitor não entendeu.
:-o     O leitor ficou surpreso.
8-O     O leitor ficou chocado.
:-(     Quem escreveu está triste.
:-<     Quem escreveu está irritado.
:-#     Quem escreveu mantém segredo.
0:-)    Quem escreveu não tem culpa.
:-\     Quem escreveu está indeciso.
%-(     Quem escreveu está confuso.
:-D     Quem escreveu está rindo.
;-(     Quem escreveu está com vontade de chorar.
;-)     Quem escreveu está piscando (maliciosamente).
```

Figura 1.9 - Os “Emocionícones” dos Sistemas de *E-Mail* [Chaves 96].

A quarta característica diz respeito ao fato do *e-mail* poder ser endereçado para múltiplos recebedores. Este aspecto indica que, sem depender do tempo ou do espaço, as pessoas podem delegar trabalhos, formar novos grupos, encaminhar contribuições ou até mesmo tomar decisões coletivas.

A quinta consiste na memória externa do correio eletrônico. O conteúdo das mensagens eletrônicas pode ser armazenado e posteriormente reenviado. Esta propriedade é importante para a memória social. Por exemplo, os participantes de um grupo eletrônico podem armazenar todas as suas interações ocorridas durante meses ou anos.

Finalmente, a memória produzida é processável por computador. Ela pode ser convenientemente organizada, editada, pesquisada e compartilhada com outras pessoas. Este atributo aumenta a força da memória social por permitir análises de tendências, pontos de consenso, padrões de participação e outras questões do gênero.

Outras tecnologias de comunicação possuem algumas destas características, mas somente o correio eletrônico reúne todos estes seis atributos (Tabela 1.7).

Tecnologia	***Atributos Tecnológicos***					
	Assíncrono	*Rápido*	*Somente Texto*	*Múltiplos Endereços*	*Memória Externa*	*Memória Processável*
Reunião	não	sim	não	sim	não	não
Telefone	não	sim	não	sim	não	não
Carta	sim	não	não	não	sim	não
Telex	sim	sim	sim	não	sim	não
Fax	sim	sim	não	não	sim	não
Mensagem de Voz	sim	sim	não	sim	não	não
Correio Eletrônico	sim	sim	não	sim	sim	sim

Tabela 1.7- Comparação do Correio Eletrônico Com Outras Tecnologias.

2. Segurança

Este capítulo discorre sobre a segurança de correio eletrônico: sobre como a confidencialidade e a autenticidade de uma mensagem em trânsito pode e deve ser protegida. A segurança é, então, considerada uma meta abrangendo vários aspectos, destacando-se o **sigilo** e a **autenticação** como os mais importantes no contexto deste trabalho. Por sigilo, entende-se que somente os usuários autorizados (pessoas ou processos) tenham acesso à informação contida em uma mensagem ou consigam torná-la inteligível. Por autenticação, entende-se o procedimento que visa assegurar a autenticidade da informação: assim, por exemplo, o procedimento pode assegurar a integridade do conteúdo de um *e-mail* ou a identidade de seu emissor.

A seção 2.1 descreve o problema da segurança do correio eletrônico. Faz uma introdução sobre o problema da coleta e sobre algumas possíveis ameaças à segurança dos *e-mails*. Introduz também os serviços de segurança do mundo real e os definidos em padrões. Descreve o gerenciamento das chaves e apresenta alguns padrões e produtos atualmente utilizados. A seção 2.2 descreve as exigências e as feições de um sistema de e-mail seguro.

2.1. O Problema da Segurança dos Sistemas de E-Mail

[Schneier 95] [Levien 96] [Heikkinen 95] [Schneier 96] [Lucchesi 86] [Bellovin 94] [Stinson 95] [Cavalcanti 96] [RFC 1421-1424] [RFC 1521]

No ambiente do correio eletrônico, as mensagens trafegam de uma máquina a outra abertas e disponíveis, como as mensagens escritas no dorso dos cartões postais. Qualquer indivíduo localizado em uma máquina intermediária pode ler as mensagens, do mesmo modo que um carteiro pode ler o verso dos cartões postais manuseados. As pessoas podem optar por não ler, ou podem não possuir os direitos de acesso para ler tão facilmente, mas a única segurança que o usuário do correio eletrônico tem baseia-se na honestidade, ignorância e indiferença daqueles situados nos pontos intermediários. Tais pontos podem ser desde universidades até empresas rivais ou governos estrangeiros. O emissor do *mail* não tem nenhum controle sobre eles.

As redes de *e-mail* são sistemas descentralizados. Os diferentes computadores na Internet possuem tabelas de roteamento. Quando um computador recebe uma mensagem destinada a alguém situado em outra máquina, ele procura por esta máquina em sua tabela de roteamento e encaminha o *e-mail*. É comum a existência de diversos intermediários entre o *emissor* e o *recebedor* de uma mensagem. Apenas para seguir a tradição, doravante o emissor de uma mensagem será chamado de *Alice* e o recebedor de *Bob* .

Se um espião ou intruso[15], também por tradição denominado *Eve* (de *eavesdropper*), sentado em uma destas máquinas intermediárias, desejar ler toda mensagem eletrônica que passar, ele o fará, não importando para quem a mensagem se destina. *Eve* poderá imprimí-la, mostrá-la a um amigo, despachá-la pela rede ou mandar uma cópia à imprensa. Se for esperto o suficiente, poderá também alterar a mensagem em trânsito.

Assim, *Alice* e *Bob*, ao trocarem uma mensagem pela Internet, são obrigados a confiar na segurança de todas as máquinas por onde a mensagem circular (Figura 2.1). Isto ocorre não apenas na Internet, mas também nas demais redes: CompuServe, GEnie, America Online, AppleLink e MCIMail. Mensagens enviadas através destas redes são tão seguras como as existentes no verso dos cartões postais.

[15] O espião ou intruso pode ser o administrador do sistema, um *haker* habilidoso ou, se a segurança da máquina intermediária for pobre o suficiente, um usuário comum.

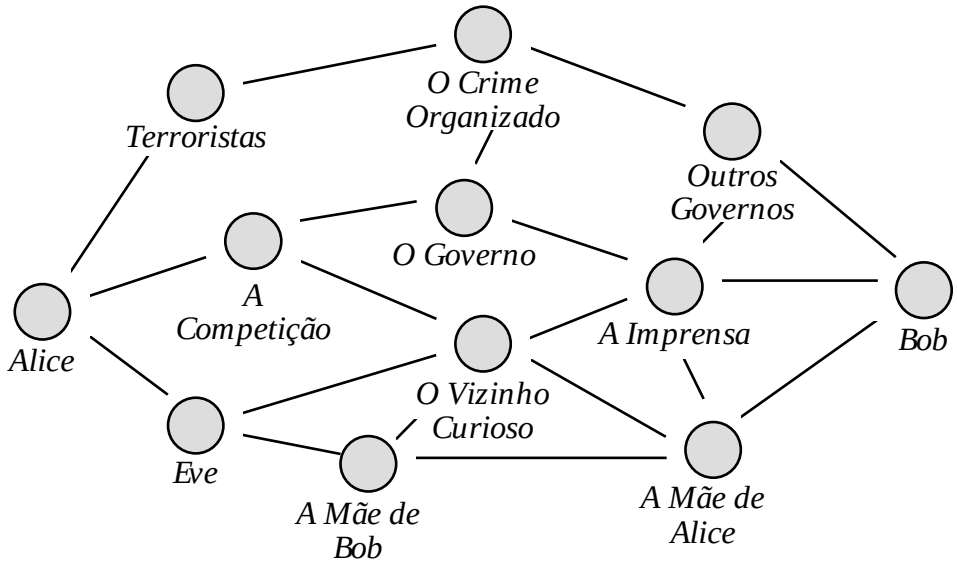


Figura 2.1- Quem Pode Ler Seu *Mail* [Schneier 95].

Se as mensagens eletrônicas são cartões postais, o que se deseja são cartas dentro de envelopes (Figura 2.2).

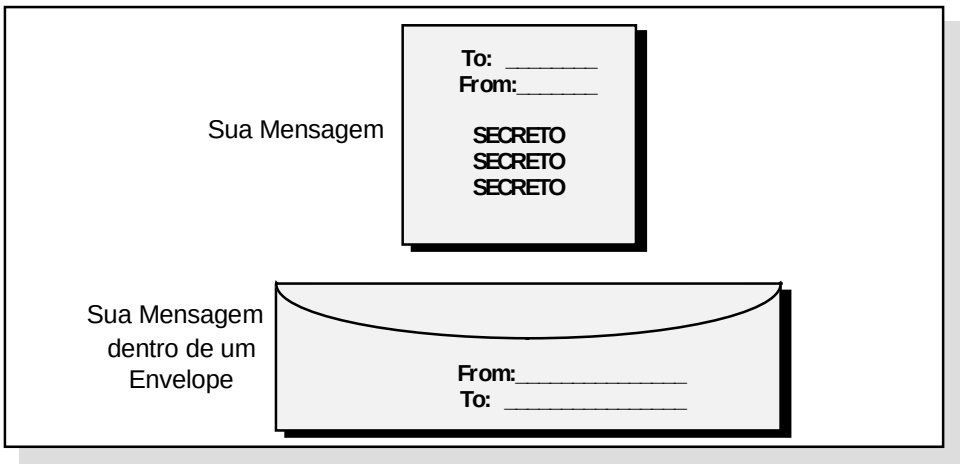


Figura 2.2 - Mensagem e Envelope.

Da mesma forma que os *e-mails*, as cartas são roteadas através de vários pontos intermediários. São colocadas em caixas de correspondências ou *mailboxes*. Um funcionário do correio faz a coleta e as leva à agência de correio local, de onde são roteadas para seu destino através de diversas agências de correio e veículos de transporte. Até que o carteiro as entrega nas caixas de correspondência ou *mailboxes* dos

respectivos recebedores. Dezenas de pessoas manuseiam as cartas durante seu trajeto pelo sistema, mas nenhuma delas pode lê-las. Elas ficam protegidas dentro de seus envelopes.

Como não é possível a colocação de um *e-mail* dentro de um envelope, utiliza-se uma analogia para representá-lo. Assim, pode-se utilizar **criptografia** como um "envelope" eletrônico.

Programas de segurança de correio eletrônico, tais como o *Pretty Good Privacy* (PGP) e implementações dos padrões *Privacy Enhanced Mail* (PEM), *Secure Multipurpose Internet Mail Extensions* (S/MIME) e dos protocolos *MIME Object Security Services* (MOSS) e *Message Security Protocol* (MSP) fazem exatamente isto.

Cifrando seu *e-mail* para que somente *Bob* possa lê-lo, *Alice* garante que *Eve* não poderá ler sua mensagem - mesmo se interceptá-la em trânsito. Adicionando uma assinatura eletrônica em seu *e-mail*, *Alice* pode garantir que *Bob* saberá quem enviou a mensagem. *Eve* não poderá trocar uma mensagem de *Alice* por outra e reivindicar que esta outra tenha sido escrita por *Alice*. Além disso, *Bob* pode até mesmo provar a terceiros que a mensagem foi enviada por *Alice*, num processo que seria o equivalente eletrônico do "reconhecimento de firma".

Portanto, estes programas de segurança de correio eletrônico são melhores do que envelopes físicos. Afinal, *Eve* pode abrir envelopes, velada ou publicamente, e ler seu conteúdo. Pode também interceptar uma carta em trânsito, abrí-la, ler sua mensagem e, em seguida, colocar outra carta em seu lugar dentro de outro envelope. Porém, *Eve* não pode fazer isto com facilidade no mundo digital. A combinação do ciframento com uma assinatura digital proporciona um "envelope" que *Eve* não consegue violar facilmente.

2.1.1. O Problema da Coleta e As Ameaças à Segurança

[Bellovin 94] [Lucchesi 86] [Schneier 95]

O principal obstáculo encontrado por *Eve* na leitura de uma mensagem alheia consiste em localizá-la dentro de um "mar" de outras mensagens eletrônicas. Em geral, além de *e-mails*, costumam circular em um *site* mensagens de *Usenet Newsgroups*, *logons* remotos, conversações de *chat* em tempo real, *downloads* de *ftp* e diversas outras mensagens. Armazenar esta quantidade de dados em um computador, analisá-la e filtrá-la consiste em um grande problema para *Eve*. Schneier denominou este problema de *problema da coleta* (Figura 2.3).

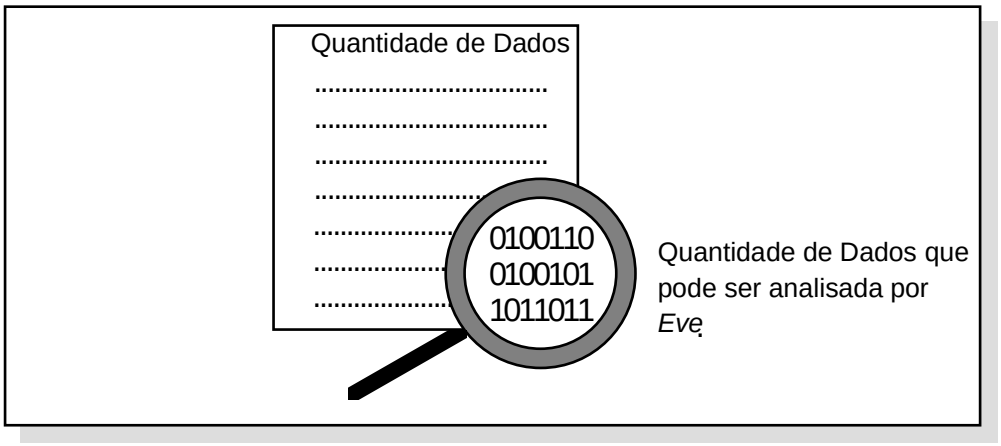


Figura 2.3 - O problema da coleta [Schneier 95].

A filtragem pode basear-se em palavras-chave, tais como: “secreto”, “estratégico” ou “criptográfico”. Se *Eve* for um agente federal, pode preferir palavras-chave do tipo: “assassinato”, “seqüestro”, “explosão”, “contrabando” ou “sonegação”.

Há outras técnicas para a coleta de mensagens. Pode-se procurar por mensagens contendo dados com uma determinada estrutura - a de ciframento do seu conteúdo, por exemplo. Mas a técnica empregada depende da disposição de tempo e dos recursos de *software* ou de *hardware* utilizados por *Eve*.

A coleta é inútil sem a análise das mensagens para verificar se realmente são do interesse de *Eve*. Será preciso uma análise pessoal, pois as palavras-chave podem estar sendo utilizadas fora do contexto esperado. As palavras “explosão”, “assassinato” e “seqüestro” podem estar relacionadas a um filme policial, por exemplo. Ou podem também fazer parte do diálogo eletrônico dos interlocutores *Alice* e *Bob* envolvidos com o crime organizado.

O ciframento torna o trabalho de *Eve* mais difícil por vários motivos. O mais óbvio é que *Eve* não poderá ler todos os *e-mails*. Porém, isto somente será verdade se o método de ciframento for seguro o suficiente para que *Eve* não consiga quebrá-lo. Se for assim, restará ainda a *Eve* a possibilidade de fazer *análise de tráfego*.

A análise de tráfego consiste na verificação de quem envia os *e-mails*, para quem são endereçados, que tamanho possuem, de que assunto (*subject*) tratam e quando foram despachados. Nestes dados há uma série de informações úteis a *Eve*, dependendo de sua perspicácia em saber interpretá-los.

Caso *Eve* consiga quebrar o método de ciframento, então o problema passa a ser apenas uma questão de quanto tempo e de quantos recursos pretende utilizar para ler as mensagens de *Alice* ou de *Bob*. Isto pode significar cinco minutos em um *Pentium* ou então muitas horas em um supercomputador paralelo.

Se *Eve* conseguir quebrar também a assinatura eletrônica de *Alice* ou de *Bob* - ou caso os *e-mails* trocados não sigam assinados eletronicamente, poderá assumir a identidade de um ou de outro. Terá condições, por exemplo, de escrever uma mensagem falsa incriminando *Alice* e entregá-la à imprensa.

Entretanto, *Eve* somente conseguirá ler uma mensagem crucial se conseguir encontrá-la., devido ao problema da coleta já exposto. E isto será mais difícil de acontecer se o ciframento for uma prática largamente utilizada pelos interlocutores. Caso contrário, se *Alice* e *Bob* somente cifrarem umas poucas mensagens, isto servirá como uma bandeira vermelha para sinalizar que tais mensagens são interessantes para *Eve*, facilitando, deste modo, seu trabalho de coleta.

Porém se todas as mensagens, inclusive as rotineiras, forem cifradas, *Eve* não poderá discernir as mensagens cifradas importantes das inócuas. Assim, o ciframento das mensagens, mesmo sendo pobre, pode rapidamente tornar o problema da coleta algo intransponível para *Eve*.

De um modo geral, as ameaças à segurança de sistemas computacionais recaem em quatro amplas classes:

- **Vazamento**: a aquisição de informação por recebedores não autorizados. Um intruso *Eve* pode ser cúmplice de um usuário legítimo (*Alice* ou *Bob*) que "vaza" informações para ele.
- **Violação** (*Tampering*): a alteração não autorizada de informação (incluindo programas).
- **Furto de Recursos**: o uso de facilidades sem autorização.
- **Vandalismo**: interferências nas operações próprias de um sistema sem ganhos para o autor.

Nos sistemas de correio eletrônico, boa parte destas ameaças podem ser atenuadas através do emprego de um gerenciamento de chaves seguro e adequado, assunto tratado no item a seguir.

2.1.2. O Gerenciamento das Chaves

[Heikkinen 95] [Schneier 96] [Stinson 95] [Schneier 95] [Levien 96]

O ciframento de uma mensagem baseia-se em dois componentes : um *algoritmo* e uma *chave*. Um *algoritmo* é uma transformação matemática. Ele converte uma mensagem em claro em uma mensagem cifrada e vice-versa. Quando *Alice* cifra uma mensagem, ela utiliza um algoritmo de ciframento para transformar o conteúdo em claro da mensagem em texto cifrado. Quando *Bob* decifra uma mensagem, ele utiliza o algoritmo de deciframento correspondente para converter o texto cifrado de novo em uma mensagem clara (Figura 2.4).

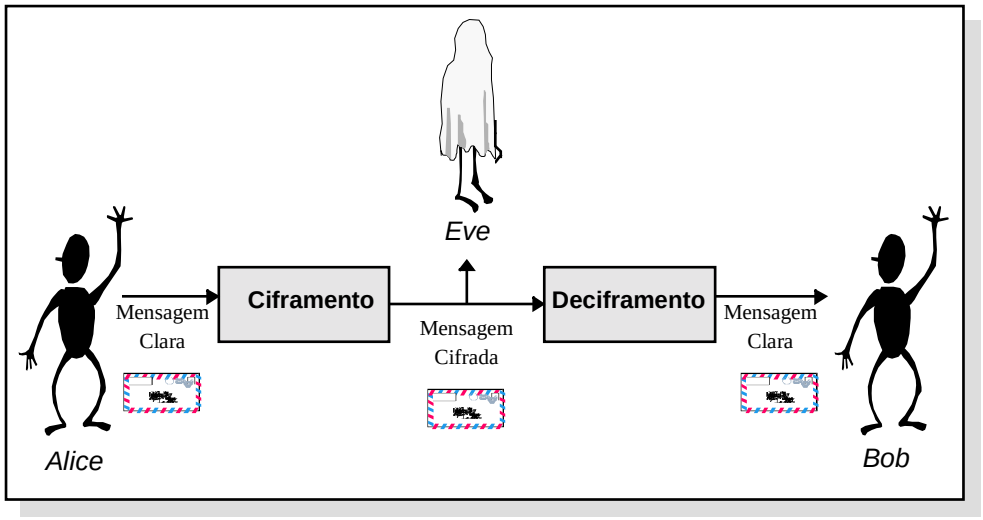


Figura 2.4 - Ciframento e Deciframento.

Antigamente, a segurança do ciframento estava baseada na manutenção do algoritmo em sigilo. Se *Eve* conhecesse o algoritmo sem chave, poderia decifrar uma mensagem cifrada tão facilmente quanto *Bob*.

Manter um algoritmo sem chave em segredo causa uma série de problemas. As organizações militares de todo o mundo gastam muito tempo e energia para fazer isto, mas mesmo assim sempre escapa algum detalhe. Se o algoritmo estiver incorporado em alguma peça de *hardware* - um rádio, por exemplo - eventualmente o inimigo poderá capturar esta unidade e aplicar a engenharia reversa para descobrir o algoritmo.[16]

[16] Em uma rede utilizando este sistema, cada par de usuários precisaria ter seu próprio algoritmo!

Pode-se contornar o problema apresentado utilizando o segundo componente básico da criptografia de mensagens: a *chave*. Uma chave é uma cadeia aleatória de bits utilizada em conjunto com um algoritmo. Cada chave distinta faz com que o algoritmo trabalhe de forma ligeiramente diferente (Figura 2.5).

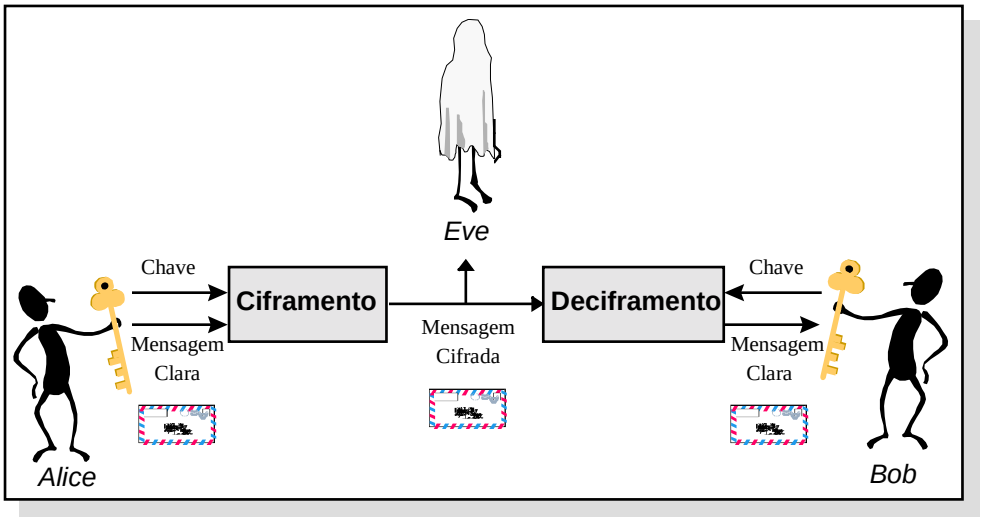


Figura 2.5 - Ciframento e Deciframento com Chave.

Quando *Alice* cifra uma mensagem, ela utiliza um algoritmo de ciframento e uma chave secreta para transformar uma mensagem clara em um texto cifrado. *Bob*, por sua vez, ao decifrar uma mensagem, utiliza o algoritmo de deciframento correspondente e a mesma chave[17] para transformar o texto cifrado em uma mensagem em claro. *Eve*, por não possuir a chave secreta, mesmo conhecendo o algoritmo, não conseguirá decifrar a mensagem[18]. A segurança do sistema passa a residir não mais no algoritmo e sim na chave empregada. É ela que agora, no lugar do algoritmo, deverá ser mantida em segredo por *Alice* e *Bob*[19] . Este tipo de ciframento emprega a criptografia denominada *simétrica* ou *convencional*.

O *gerenciamento das chaves* consiste na parte mais difícil da criptografia de mensagens eletrônicas. É fácil escolher um algoritmo seguro para ciframento e implementá-lo. É simples cifrar mensagens em uma ponta e decifrá-las em outra. O maior desafio consiste justamente na distribuição segura das chaves.

[17] Na verdade a chave de deciframento ou é a mesma que a utilizada para o ciframento ou é uma chave facilmente derivada desta última.

[18] Funciona igual a uma fechadura. Mesmo se duas pessoas utilizarem duas fechaduras idênticas, não poderão abrir a porta uma da outra, a não ser que tenham ambas a mesma chave.

[19] Uma rede inteira de usuários pode se comunicar seguramente utilizando apenas um algoritmo e muitas chaves secretas distintas.

Antes de *Alice* enviar uma mensagem cifrada para *Bob*, ambos precisam entrar em um acordo quanto à chave empregada. Eles podem se encontrar e combinar previamente a chave a ser utilizada. *Alice* também pode gerar a chave sozinha e enviá-la para *Bob* através de um mensageiro de confiança. Feito isto, eles terão duas opções:

- armazenar a chave em algum lugar, correndo risco de furto, até que ela venha a ser necessária; ou
- memorizá-la até sua utilização, sob risco de esquecimento.

Historicamente, as organizações de inteligência militar têm obtido muito mais sucesso no ataque a mecanismos de gerência de chaves de outros países do que no ataque a seus algoritmos de ciframento. A rede de espionagem *Walker* enviou durante anos, à antiga União Soviética, cópias das chaves de ciframento da Marinha Americana. Não importava se o algoritmo de ciframento da Marinha dos Estados Unidos da América era bom; os soviéticos podiam ler tudo.

Ainda bem que há uma solução melhor do que *Alice* e *Bob* encontrarem-se em um beco escuro para combinarem uma chave. Os programas de segurança de *e-mail* modernos permitem que *Alice* envie uma mensagem a *Bob* sem ter que convencionar uma chave secreta previamente. Não precisam nem mesmo se conhecer ou sequer confiar um no outro. Este prodígio é possível devido a existência da *criptografia de chave pública* (ou *assimétrica*) .

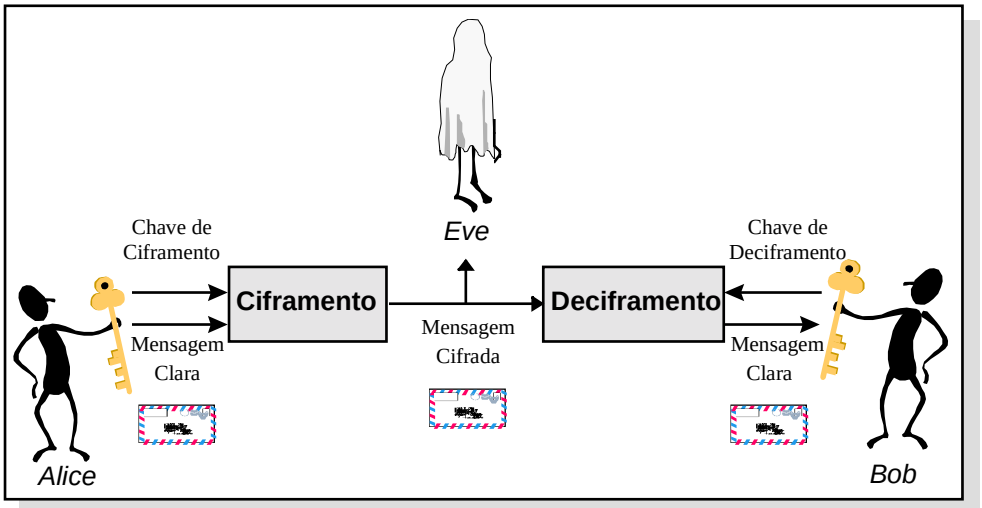


Figura 2.6 - Ciframento e Deciframento com Chave Pública

Na criptografia de chave pública há duas chaves diferentes: uma para ciframento e outra para deciframento (Figura 2.6). Elas existem em pares; uma chave de ciframento trabalha junto com uma chave de deciframento específica. Como não se pode derivar uma chave a partir da outra, então com apenas a chave de ciframento, ninguém conseguirá decifrar mensagens.

De uma forma simplificada, o sistema funciona assim: *Bob* e todos os que desejam comunicar-se de modo seguro geram uma chave de ciframento e sua correspondente chave de deciframento. Ele mantém secreta a chave de deciframento; esta é chamada de sua *chave privada*. Ele torna pública a chave de ciframento: esta é chamada de sua *chave pública*.

A chave pública realmente condiz com seu nome. Qualquer pessoa pode obter uma cópia dela. *Bob* inclusive encoraja isto, enviando-a para seus amigos, publicando-a em boletins ou em seu arquivo *.plan*. Assim, *Eve* não tem nenhuma dificuldade em obtê-la. Quando *Alice* deseja enviar uma mensagem a *Bob*, precisa primeiro encontrar a chave pública dele. Pode-se fazer isto de várias maneiras:

- pode-se obtê-la diretamente de *Bob*;
- pode-se obtê-la de um banco de dados centralizado; ou
- pode-se obtê-la de seu próprio banco de dados.

Feito isto, ela cifra sua mensagem utilizando a chave pública de *Bob*, despachando-a em seguida. Quando *Bob* recebe a mensagem, ele a decifra facilmente com sua chave privada. *Eve*, que interceptou a mensagem em trânsito, não conhece a chave privada de *Bob*, embora conheça sua chave pública. Mas este conhecimento não ajuda a decifrar a mensagem. Mesmo *Alice*, que foi quem cifrou a mensagem com a chave pública de *Bob*, não pode decifrá-la agora.

Desta forma, mesmo alguém que nunca tenha encontrado *Bob* antes, pode obter sua chave pública de uma banco de dados e enviar-lhe um *e-mail* cifrado sem antes precisar compartilhar um segredo com ele. Pode-se fazer duas analogias para um melhor entendimento deste conceito:

- imaginar este sistema como uma caixa de correspondências, onde qualquer pessoa (*Alice*) pode inserir uma carta através da sua abertura (mensagem cifrada com a chave pública). Somente o recebedor autorizado (*Bob*), aquele com a chave da citada caixa (a chave privada), pode abri-la e ler a carta nela depositada (decifrar a mensagem).

- imaginar este sistema como uma caixa de metal onde alguém (*Alice*) insere uma carta e a tranca, utilizando um cadeado de combinação colocado na caixa por outra pessoa (mensagem cifrada com a chave pública de *Bob*). Esta outra pessoa (*Bob*), portanto, passa a ser a única a poder abrir a caixa de metal, pois somente ela conhece a combinação para a abertura do cadeado (a chave privada de *Bob*).

Com um sistema de chave pública, o gerenciamento de chaves passa a ter dois novos aspectos: primeiro, deve-se previamente localizar a chave pública de qualquer pessoa com quem se deseja comunicar e, segundo, deve-se obter uma garantia de que a chave pública encontrada seja proveniente daquela pessoa (*Bob*).

Sem esta garantia, um intruso *Eve* pode convencer os interlocutores (*Alice* e *Bob*) de que chaves públicas falsas pertencem a eles. Estabelecendo um processo de confiança entre os interlocutores, *Eve* pode fazer-se passar por ambos.

Deste modo, quando um interlocutor (*Alice*) enviar uma mensagem ao outro (*Bob*) solicitando sua chave pública, o intruso poderá interceptá-la e devolver-lhe uma chave pública forjada por ele. Ele também pode fazer o mesmo com o recebedor (*Bob*), fazendo com que cada lado pense que está se comunicando com o outro, quando na verdade estão sendo interceptados pelo intruso.

Eve então pode decifrar todas as mensagens, cifrá-las novamente ou, se preferir, pode até substituí-las por outras mensagens, dando início ao ataque do tipo *Man-In-The-Middle*, descrito no item 2.1.3. Através deste ataque, um intruso pode causar tantos danos ou até mais do que causaria se conseguisse quebrar o algoritmo de ciframento empregado pelos interlocutores.

A garantia para evitar este ataque é representada pelos *certificados de chave pública*. Tais certificados consistem em chaves públicas assinadas por uma pessoa de confiança. Servem para evitar tentativas de substituição de uma chave pública por outra [Schneier 96]. O certificado de *Bob* contém algo mais do que sua chave pública: contém informações sobre *Bob* - seu nome, endereço e outras dados pessoais - e é assinado por alguém em quem *Alice* deposita sua confiança: uma *autoridade de certificação* ou *CA* (*Certification Authority*).

Pela assinatura da chave pública e das informações sobre *Bob*, a CA garante que a informação sobre *Bob* está correta e que a chave pública em questão realmente pertence a *Bob*. *Alice*, por sua vez, confere a assinatura da CA e então utiliza a chave pública em pauta, segura de que esta pertence a *Bob* e a

ninguém mais. Certificados desempenham um importante papel em um grande número de protocolos e padrões utilizados na proteção de sistemas de correio eletrônico, conforme descrito no item 2.1.5.

No próximo item (2.1.3), serão descritos alguns dos principais métodos de ataque a sistemas de correio eletrônico utilizados por intrusos. Quando o gerenciamento de chaves é inadequado, observa-se um melhor desempenho em boa parte dos ataques efetuados.

2.1.3. Métodos de Ataque

Nos sistemas de correio eletrônico, os principais métodos de ataque utilizados por *Eve* incluem (Figura 2.7) :

- **Escuta clandestina** ("***eavesdropping***"): obtenção de cópias de mensagens sem autorização, diretamente de uma rede, canal de comunicação ou pelo exame de informações armazenadas e inadequadamente protegidas.
- **Mascaramento** : envio ou recepção de mensagens utilizando a identidade de outro interlocutor (*Alice* ou *Bob*) sem sua autorização. Possibilita uma forma de ataque conhecido como *Man-In-The-Middle* - onde *Eve* se mascara de *Alice* para *Bob* e se faz passar por *Bob* perante *Alice*, se interpondo na troca de mensagem.
- **Violação de mensagens**: interceptação de mensagens e alteração de seus conteúdos antes de passá-los para o recebedor (*Bob*) a que se destinam. É um tipo de ataque também conhecido como *ataque ativo*. Inclui também a inserção e a remoção de mensagens.
- **Repetição *(Replaying)***: armazenamento de cópias de mensagens e reenvio delas em uma data posterior. Por exemplo, após a autorização para o uso de um recurso ter sido revogada. Mesmo que não se conheça o claro de algum texto, um intruso pode, simplesmente, repetir o cifrado durante um protocolo para ganhar acesso a outras informações. Também conhecido como *ataque da meia-noite.*

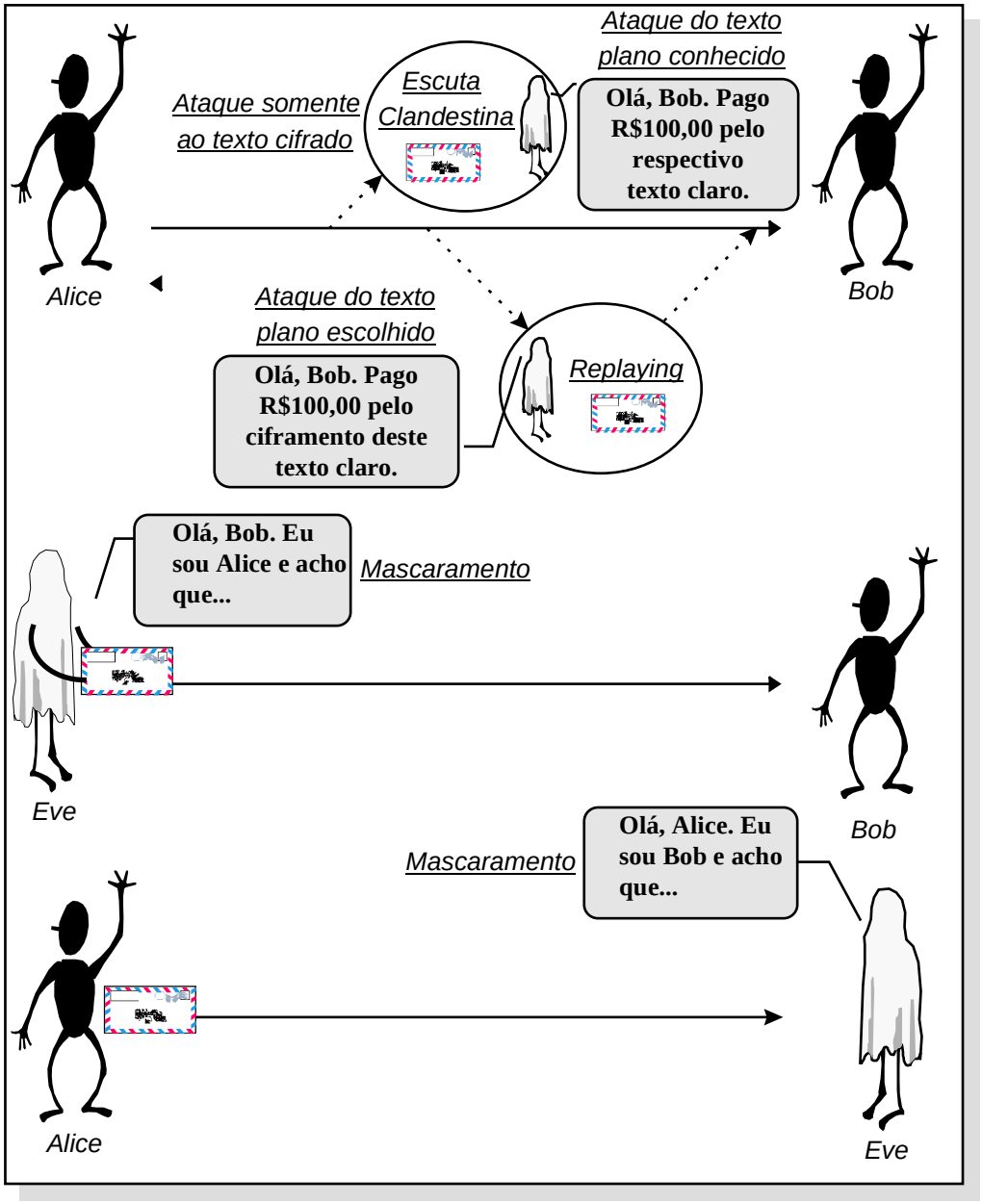


Figura 2.7 - Métodos de Ataque à Comunicação por E-Mail.

- **Inferência**: tentativa de obter algum item de informação através de cálculos matemáticos, por criptoanálise. A *criptoanálise* consiste na ciência - ou arte - de ler tráfego cifrado de mensagens sem conhecer, a princípio, a chave utilizada [Bellovin 94]. Ou seja**,** ler o conteúdo dos textos cifrados sem que se seja o legítimo destinatário. O objetivo usual é obter os conteúdos somente com o auxílio dos textos cifrados, mas isso pode ser muito difícil; então, a criptoanálise também pode ser

baseada em texto escolhido ou texto conhecido. Para algoritmos de bloco, existem métodos próprios de ataque, tais como a Criptoanálise Diferencial e Criptoanálise Linear. A tabela a seguir apresenta resultados de criptoanálise do DES (Tabela 2.1).

ATAQUE (DES)	***TIPO***	***COMPLEXIDADE***
exaustão	claro conhecido	2^55
Diferencial	claro conhecido	2^55
Diferencial	claro escolhido	2^47
Linear	claro conhecido	2^43

Tabela 2.1 - Criptoanálise do DES.

Como exemplo de inferência por criptoanálise, pode-se citar o furto de informação pela derivação de uma chave criptográfica.

- **Busca exaustiva**: tentativa de se utilizar todas as chaves possíveis. É um ponto de referência para os outros tipos de ataque, também conhecida como método da *força bruta.* É um ataque sempre possível; não há como prevení-lo. O melhor a fazer é tornar este tipo de ataque tão dispendioso em tempo e recursos de modo a não valer a pena sequer iniciá-lo. O número total de chaves possíveis para um tamanho de chave *n* é igual a 2^n. Por exemplo: uma chave de tamanho igual a *128* bits possibilita 2^{128} chaves distintas, ou seja, cerca de 10^{38}.

- **Ataque somente ao texto cifrado:** um criptoanalista (*Eve*) possui alguns *e-mails* cifrados com um algoritmo conhecido, sem ter acesso ao respectivo texto original das mensagens ou às chaves utilizadas. Ele tem como meta encontrar os correspondentes textos em claro trocados por *Alice* e *Bob*.

- **Ataque do texto plano conhecido**: é comum um intruso *Eve* ter, além do algoritmo, um ou mais pares de mensagens de *Alice* com um texto cifrado e seu respectivo texto claro, empregando a mesma chave. Estes pares, conhecidos como *cribs*, podem prestar auxílio à criptoanálise. *Eve* tem como meta encontrar a chave utilizada na comunicação.

- **Ataque do texto plano escolhido:** é suposto que o intruso *Eve* escolhe um ou mais pares de textos claros e cifrados correspondentes, cada par com uma chave. Pode ser que os claros escolhidos não estejam presentes em nenhuma mensagem real. Assim, neste ataque, *Eve* escolhe um pedaço de um

texto em claro para ser cifrado com um determinado algoritmo. Ele conhece o texto claro que escolheu, seu correspondente texto cifrado e o algoritmo, mas não conhece a chave. Tem como meta encontrar esta chave, para poder desvendar outras mensagens cifradas por *Alice* ou *Bob* com o mesmo algoritmo e a mesma chave.

- **Corte e cola**: dadas duas mensagens cifradas com a mesma chave, é possível, às vezes, combinar porções de duas ou mais mensagens para produzir uma nova. Pode ser que não se conheça exatamente o que elas dizem, mas um intruso (*Eve*) pode utilizá-las para enganar um usuário autorizado (*Alice* ou *Bob*), de modo a este fazer o que ele deseja.
- ***"Man-In-The-Middle"***: através de uma dupla personificação, um usuário pode interferir em um processo de comunicação por chave pública (Figura 2.8).

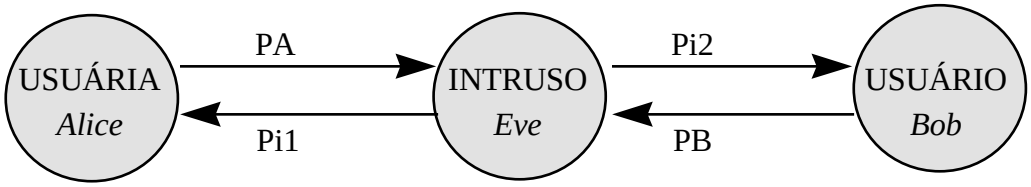


Figura 2.8 - Ataque do Tipo *Man-In-The-Middle*.

Protocolo utilizado:

1) A usuária *Alice* envia para *Bob* sua chave pública *PA*, e pede que *Bob* lhe envie sua chave pública *PB*.

2) O intruso *Eve* intercepta a mensagem de *Alice* e envia uma chave Pi_1 para ela, que passa a achar que o intruso é *Bob*. Em seguida, *Eve* manda uma chave pública PUi_2 a *Bob*, e pede que *Bob* lhe envie sua chave pública.

3) O usuário *Bob* envia sua chave pública PB ao intruso, pensando que ele é a usuária *Alice*. Deste modo, todo texto cifrado de *Alice* para *Bob* e de *Bob* para *Alice* é lido por *Eve*.

Para lançar os ataques citados a um sistema de correio eletrônico, o intruso deve ter acesso ao sistema de modo a executar o programa que implementa o ataque. A maioria dos ataques são lançados por um dos usuários legítimos de um sistema. Eles abusam de suas autorizações executando programas designados para "carregar" uma das formas de ataque.

Para usuários ilegítimos, um simples método de infiltração é através de suposição de *passwords* ou pelo uso de programas "quebra senha" para obter a senha de um usuário conhecido. Em adição a estas formas diretas de ataque, há alguns métodos mais sutis. Eles incluem:

- **Vírus**: um programa que é anexado a um programa hospedeiro legítimo e se instala no ambiente alvo quando o programa hospedeiro está sendo executado. Uma vez instalado, ele executa suas ações criminosas quando lhe convier, freqüentemente utilizando uma data como seu gatilho.
- **Verme** (*worm*): um programa que explora facilidades para executar processos remotamente em sistemas distribuídos.
- **Cavalo de Tróia** (*Trojan Horse*): um programa oferecido a usuários de um sistema como desempenhando uma função útil, mas que tem uma segunda função dissimulada escondida dentro dele. Por exemplo, o *spoof login*, um programa que se apresenta a usuários com *prompts* que não são distinguíveis de *login* regulares e diálogos de *password*, mas de fato armazenam a entrada inocente do usuário em um arquivo conveniente para posterior uso ilícito.

A conclusão a que se chega sobre a discussão dos métodos de ataque com suas respectivas formas de infiltração é que, para se produzir um sistema seguro e, particularmente, um **sistema de correio eletrônico seguro**, deve-se projetar os componentes do sistema (por exemplo, os diretórios) assumindo-se que as outras partes (pessoas ou programas) não são confiáveis, até que se prove o contrário.

2.1.4. Serviços de Segurança

[Heikkinen 95] [Schneier 96] [Levien 96]

A maioria dos serviços de segurança discutidos neste item não são específicos para *e-mail*. Por exemplo: qualquer documento confidencial pode ser cifrado e assinado. Entretanto, os serviços debatidos são, na maior parte das vezes, utilizados para correio eletrônico. São eles:

- **Confidencialidade de Conteúdo .** Confidencialidade significa proteger a mensagem contra a divulgação a usuários não autorizados. Ou seja, a mensagem não deve ser revelada a ninguém, exceto ao recebedor tencionado pelo emissor. Isto é obtido pelo ciframento da mensagem,

utilizando-se algoritmos simétricos (de chave secreta) ou assimétricos (com duas chaves distintas). A submissão de mensagens a múltiplos recebedores exige técnicas simétricas, sozinhas ou combinadas com técnicas assimétricas. Já o emprego de técnicas assimétricas isoladas para criframento sai muito caro computacionalmente, o que significa que não é prático utilizá-las para cifrar mensagens inteiras.

- **Autenticidade da Origem da Mensagem.** A autenticidade da origem da mensagem simplesmente fornece uma resposta à seguinte pergunta: *"Quem enviou esta mensagem?"* Deve ser possível a um recebedor de uma mensagem averiguar sua origem. Um intruso *(Eve)* não deve ser capaz de se fazer passar pelo emissor legítimo. Ela é tipicamente provida pela integridade do conteúdo. Sendo assim, a autenticidade da origem da mensagem é proporcionada pela utilização de *assinaturas digitais*. Estas são muito semelhantes a assinaturas feitas a mão. Por serem não reutilizáveis, inimitáveis e autênticas, elas provam que o documento não foi alterado e também não podem ser repudiadas pelo emissor. Mensagens com assinaturas digitais não são reutilizáveis de um modo geral mas, apesar disso, elas podem ser repetidas (*replayed*). Embora o conteúdo da mensagem não possa ser modificado, assinaturas digitais não garantem necessariamente que linhas de cabeçalho, tais como `Date` e `Subject`, não sejam alteradas. A assinatura e o criframento podem ser combinados de três maneiras distintas:

  primeiro cifrando o documento, depois assinando-o;

  primeiro assinando o documento, depois cifrando somente o seu conteúdo, deixando a assinatura em claro; ou

  primeiro assinando o documento, depois cifrando seu conteúdo junto com a assinatura.

  A terceira solução é a melhor, sob o ponto de vista de proteger a identidade do emissor perante intrusos.

- **Autenticidade da Integridade do Conteúdo.** A autenticidade da integridade do conteúdo assegura que o conteúdo da mensagem não foi modificado. Deve ser possível ao recebedor de uma mensagem *(Bob)* verificar que ela não foi modificada em trânsito; um intruso *(Eve)* não deve ser capaz de substituir uma mensagem legítima por outra falsa. A autenticidade da origem da mensagem é automaticamente provida pela integridade do seu conteúdo. De fato, ambas devem sempre ser utilizadas em conjunto: a manutenção da integridade não vale nada se a origem da

mensagem não puder ser confirmada; a origem autenticada não possui valor algum se a integridade não puder ser preservada. Verificar a integridade do conteúdo de uma mensagem muito grande é uma operação exaustiva. Então é calculado um pequeno valor *hash* [20] em separado da mensagem e este valor segue incluso nela. Um valor *hash* consiste em uma pequena e única quantia de informação, também chamada de *message digest*, associada a uma mensagem em particular, porém muito menor do que a mensagem em si. Como conseqüência, a verificação da autenticidade da origem da mensagem (*MOAC - Message Origin Authentication Check)* e da integridade do seu conteúdo torna-se uma operação bem mais simples: a autenticidade completa da mensagem será averiguada com base na conferência de um pequeno valor *hash* previamente assinado.

- **Não Repúdio.** Não repúdio da origem, em sua concepção mais simples, consiste apenas na autenticação da origem da mensagem (e também da integridade do conteúdo) combinada com algoritmos de ciframento assimétrico. Deste modo, um emissor (*Alice*) não poderia falsamente negar, mais tarde, que tenha enviado uma mensagem. Já na utilização do não repúdio do recebimento existem alguns problemas. Por exemplo, mensagens de trote não podem ser detectadas antecipadamente. Para resolver isto, o recebedor *Bob* poderia primeiro confirmar a impressão digital de uma mensagem (*digital fingerprint*), mas neste caso ele poderia rejeitar a mensagem e, portanto, recusar o seu recebimento. Além disso, não há nenhuma definição consensual do termo *recebimento*. Normalmente, ele significa leitura, mas não há nada dito sobre o entendimento da mensagem ou sobre tomar alguma ação a respeito dela. Os sistemas EDI (*Electronic Data Interchange*) permitem requisitar que os recebedores não possam ler as mensagens sem uma notificação de recebimento assinada. Isto pode ser obtido pelo uso do *digital certified mail protocol.* Este protocolo é teoricamente válido, mas complicado e pouco prático, exigindo o envio de 200 mensagens entre o emissor e o recebedor. Baseia-se na revelação de chaves secretas, *bit* a *bit*, para ambas as partes. Outra maneira consiste na utilização de um árbitro, que impediria o recebedor de ler uma mensagem antes do envio de um recibo assinado.

[20] Um valor *hash* funciona como uma impressão digital que possibilita a distinção entre uma mensagem e outra, mesmo se ambas diferirem por apenas um bit.

2.1.5. Padrões e Produtos

[Levien 96] [Schneier 96] [Heikkinen 95] [Schneier 95] [Cavalcanti 96] [RFC 1421-1424] [RFC 1521]

Na prática, algoritmos criptográficos de chave pública são pesados e complicados. Por este motivo, ninguém os utiliza para cifrar mensagens eletrônicas inteiras, como citado no item 2.1.2; levaria muito tempo. Assim, os programas de segurança de correio eletrônico existentes utilizam a criptografia de chave pública voltada para o gerenciamento de chaves, e não para o ciframento de mensagens inteiras.

Para cifrar as mensagens, são utilizados algoritmos de criptografia convencional ou de chave secreta - bem mais rápidos, tais como o DES e o IDEA, ambos detalhados no Capítulo 4 - Criptografia. A cada novo ciframento de mensagem, os programas de ciframento de *e-mail* geram uma nova chave secreta a ser utilizada pelo algoritmo convencional - também conhecida como *chave de sessão aleatória*. Desta maneira, somente esta chave de sessão é cifrada pela criptografia de chave pública. O algoritmo de chave pública é, portanto, utilizado não no ciframento de mensagens inteiras e, sim, na *distribuição* da chave de sessão aleatória citada.

Seguem alguns exemplos de padrões e produtos para tornar mais claro como funcionam, na prática, os sistemas de segurança de correio eletrônico.

- **Pretty Good Privacy - PGP.** Foi implementado em 1991. É o produto para segurança de *e-mail* com maior sucesso - tanto que tornou-se o padrão de "facto" para ciframento de *e-mail* na Internet. Destinado a ser utilizado com todos os sistemas de correio eletrônico existentes, encontra-se disponível em quase todas as plataformas. Considerado um programa forte em termos criptográficos, levantou muita discussão desde que seu criador, Phil Zimmermann, foi indiciado por facilitar sua exportação ilegal dos Estados Unidos, tornando o programa disponível via Internet. No PGP, uma mensagem é assinada e cifrada como descrito a seguir (Figura 2.9):

    1. *Alice* escreve a mensagem.
    1. É gerado um valor com base no cálculo de uma função de espalhamento unidirecional (*one-way hash*) aplicada ao conteúdo da mensagem - função MD5, no caso do PGP, detalhada no Capítulo 4 - Criptografia.
    1. O valor *hash* é assinado; utiliza-se, para isto, a assinatura do emissor, *Alice*. Para assinar, o PGP utiliza o algoritmo de chave pública RSA, também descrito no Capítulo 4.

1. A mensagem e sua respectiva assinatura são concatenadas.
1. É criada uma chave de sessão aleatória.
1. A mensagem assinada é cifrada com a chave de sessão, empregando-se um algoritmo de chave secreta - o IDEA, no caso do PGP.
1. *Alice* obtém a chave pública do recebedor, *Bob*.
1. A chave de sessão aleatória é cifrada com a chave pública de *Bob*, empregando-se um algoritmo de chave pública - o RSA, no caso do PGP, detalhado no Capítulo 4 (Criptografia).
1. A mensagem cifrada é concatenada à chave de sessão, também cifrada, para se conseguir, finalmente, a mensagem segura.
1. *Alice* despacha a mensagem assinada e cifrada para *Bob*.

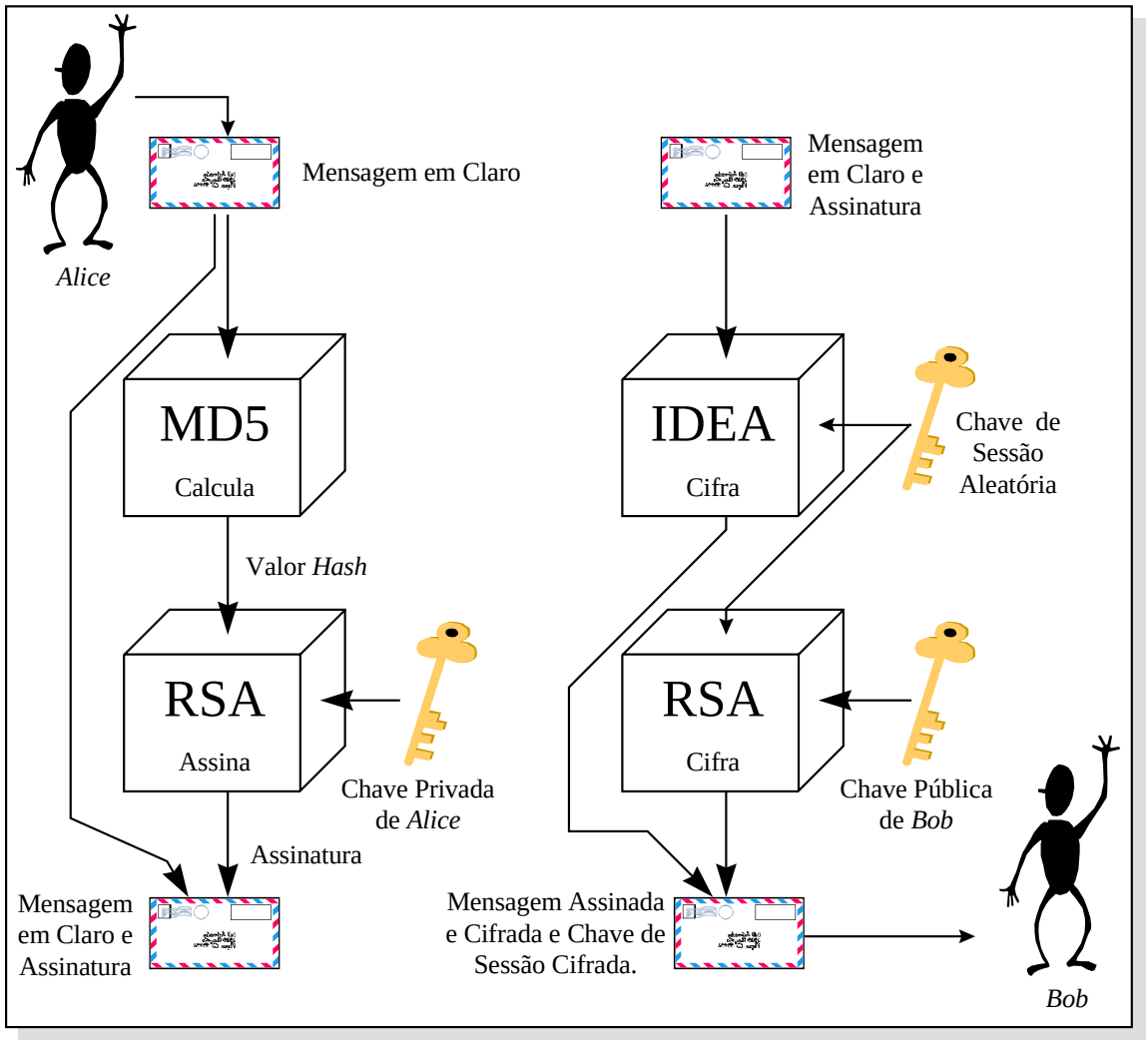


Figura 2.9 - Assinatura e Ciframento de Mensagem Eletrônica pelo PGP.

O aspecto mais interessante do PGP é o seu modelo de distribuição para o gerenciamento de chaves. Não há hierarquia de autoridades para certificação; ao invés disso, o PGP provê uma *teia de confiança*, composta por uma rede distribuída de indivíduos. No PGP, vale a frase: "eu conheço quem você é porque eu confio em alguém que acredita que você é quem diz ser". Todo usuário gera e distribui sua própria chave pública. Ela pode ser distribuída por *e-mails* ou através de páginas *Web*, arquivos *.plan*, servidores de chave ou boletins internos.

Os usuários assinam, conforme desejarem, as chaves públicas uns dos outros, criando, assim, uma comunidade (informal) de usuários PGP interligados. Eles decidem em quem confiam para atestá-los para outros usuários. Assim, cada usuário pode obter as chaves diretamente, tornando-se sua

própria origem da confiança, e cada terceira pessoa torna-se uma autoridade de certificação ou CA (*Certification Authority*).

Esta propriedade também trás consigo problemas de escala, ligados à administração de chaves: é o caso de uma rede composta por milhares de usuários PGP, onde um usuário não pode verificar a validade de todas as outras chaves. Por outro lado, o PGP não requer nenhuma infra-estrutura para seu funcionamento. Duas pessoas podem começar a utilizá-lo para comunicarem-se instantaneamente. Mais usuários podem ser adicionados e a rede cresce rapidamente. O PGP também não proporciona autenticação do recebimento. O Capítulo 5 - Descrição do PGP - contém mais detalhes sobre este programa.

- **Privacy-Enhanced Mail - PEM.** É o padrão adotado pela IAB (*Internet Architecture Board*) para prover segurança de correio eletrônico na Internet. Projetado inicialmente pelo IRTF/PSRG (*Internet Resources Task Force / Privacy and Security Research Group*), foi repassado para o IETF / PEM Working Group, que publicou os documentos finais em 1993. Encontra-se definido nas RFCs 1421-1424. A primeira RFC (antiga 989) foi publicada em 1987. O padrão PEM engloba a essência dos serviços de segurança OSI: ciframento, autenticação, integridade de mensagens e gerenciamento de chaves, sendo esta baseada em uma hierarquia de certificados.

  Como o PGP, o PEM também não proporciona autenticação do recebimento e pode ser executado em quase todos os sistemas de *e-mail* existentes. Os protocolos e procedimentos PEM foram projetados para serem compatíveis com uma certa variedade de modelos de gerenciamento de chaves. Isto faz com que o PEM inclua tanto o esquema de criptografia simétrica quanto o de chave pública para o ciframento das chaves de sessão.

  O padrão PEM permite diferentes combinações entre alguns algoritmos que apóia. Também utiliza criptografia simétrica para cifrar o conteúdo das mensagens. E algoritmos *hash* criptográficos para garantir a integridade dos *e-mails*. Não é possível despachar, pelo PEM, mensagens sem assinatura, o que obriga a assinatura de todas as mensagens, algo bastante inconveniente e demorado.

  O aspecto que mais se destaca no PEM é a sua distribuição hierárquica de chaves. O controle centralizado é obtido através de um pequeno número de servidores *root* (formando a IPRA - *Internet PCA Registration Authority*, sendo PCA a sigla para *Policy Certification Authority*) que são a origem de toda a confiança. A confiança, no caso do PEM, é obrigatória, e não uma escolha individual de um

determinado usuário. Vale a frase: "eu conheço quem você é porque seu CA assinou por você, seu PCA pertinente assinou por seu CA e o IPRA assinou por seu PCA" - onde IPRA, PCA e CA representam entidades organizadas em uma estrutura hierárquica (Figura 2.10).

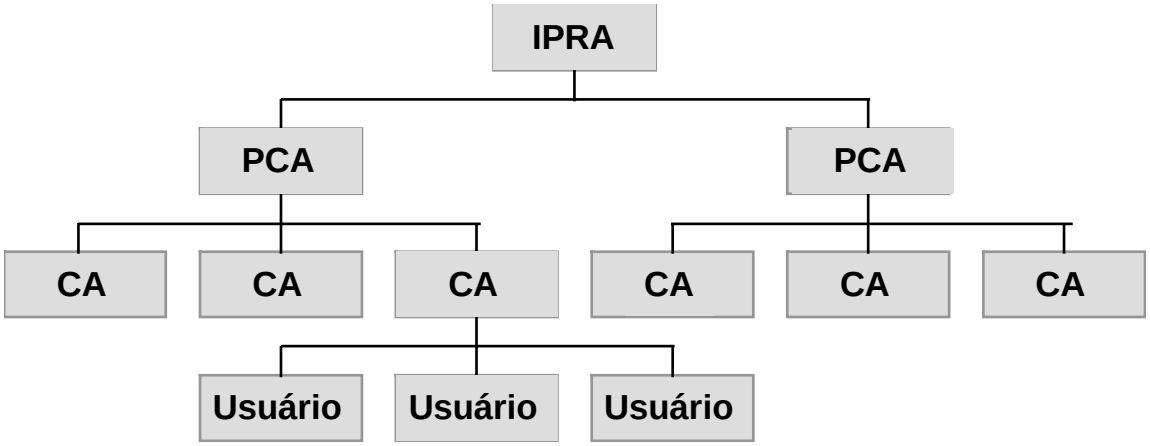


Figura 2.10 - A hierarquia de certificados PEM.

Deste modo, dois usuários sempre possuirão alguma autoridade hierarquicamente superior em comum que tenha assinado ambas as chaves para possibilitar uma comunicação mútua segura e confiável. Em última instância, esta autoridade seria o IPRA raiz. Esta autoridade central, raiz da hierarquia de certificados, cobra uma taxa para o seu funcionamento. Além disso, todas as chaves precisam ser certificadas, demandando suporte e custo adicional.

Embora na teoria seja altamente escalável, o padrão PEM ainda carece, na prática, de infra-estrutura e de diretórios públicos para apoiar sua escalabilidade. Deste modo, produtos desenvolvidos segundo a especificação PEM, como o RIPEM, não implementam a totalidade de suas especificações. O RIPEM (*Riordan's Internet Privacy-Enhanced Mail*), por exemplo, não utiliza a certificação para autenticação de chaves prevista no padrão X.509[21] (RFC 1422). O Capítulo 6 - Descrição do PEM - contém mais detalhes sobre o padrão PEM e o programa RIPEM.

- **Outros padrões e produtos.** O padrão PGP/MIME, especificado pela IETF, adiciona ao PGP a habilidade de manusear objetos MIME[22] (*Multipurpose Internet Mail Extensions).* Já o PEM foi

[21] A especificação X.509 define o relacionamento entre as autoridades de certificação. Faz parte das séries X.500 de recomendações para uma estrutura de diretório global, baseada em nomes distintos para localização.

[22] A RFC 1521 descreve as extensões MIME.

adaptado aos objetos MIME através do MOSS[23], definido pelas RFCs: 1848-*MIME Object Security Services (MOSS)* e 1847-*Security Multiparts for MIME: Multipart/Signed and Multipart/Encrypted* [24].

O S/MIME[25] (*Secure Multipurpose Internet Mail Extensions*) consiste em um esforço de um consórcio de empresas, liderado pela RSADSI e pela Microsoft, para adicionar segurança a mensagens eletrônicas no formato MIME. A especificação S/MIME está baseada em dois *Internet Drafts*: *S/MIME Message Specification* e *S/MIME Certificate Handling*[26].

O *Message Security Protocol* (MSP) é tido como o equivalente militar do PEM[27], sendo criado para ser empregado com o *Defense Secure Data Networking System* da agência americana NSA e para ser utilizado com o correio da Internet no formato MIME. O MSP foi padronizado nas séries de documentos SDN.700 do NIST (*National Institute of Standards and Technology*), enquanto que sua integração MIME atualmente é um *Internet Draft*.

2.2. Exigências Para Um Sistema de E-Mail Seguro

[Schneier 95]

Um sistema de *e-mail* precisa ser seguro e fácil de utilizar. No item 2.1.2, verificou-se a necessidade de um adequado gerenciamento de chaves. No item 2.1.4 verificou-se também a necessidade dos seguintes serviços para prover a segurança:

- Confidencialidade de Conteúdo;

[23] O MOSS não exige certificados e não provê serviços de ciframento simétricos como o PEM.

[24] O formato *Multipart / Signed and Multipart / Encrypted* permite que diferentes partes de uma mensagem sejam cifradas e assinadas individualmente.

[25] O modelo de gerenciamento de chaves do S/MIME é um híbrido situado entre a hierarquia de certificados e a teia de confiança.

[26] Um inconveniente potencial do S/MIME é que ele permite a utilização de chaves muito pequenas para garantir a segurança adequada. Isto ocorre porque todas as implementações do S/MIME devem incluir o RC2, um algoritmo proprietário da RSA cuja versão de exportação possui uma chave de apenas 40 bits. Como o RC2 representa o menor denominador comum, quaisquer duas implementações do S/MIME podem definitivamente comunicar-se entre si - porém às custas de um algoritmo inseguro.

[27] Trata-se de um protocolo do nível de aplicação compatível com o X.400. Como o PEM, as implementações do MSP são flexíveis e projetadas para acomodar uma variedade de algoritmos para funções de segurança, incluindo: assinatura, *hashing* e ciframento. Possui duas características importantes: uma capacidade de recebimento assinado - ou *não repúdio do recebimento* - bastante forte, em termos criptográficos; e a habilidade de classificar as mensagens - como ultra-secretas, por exemplo. Assim, um cliente MSP rejeitará as mensagens se os usuários não possuírem o credencial ou a autorização apropriada para lê-las.

- Autenticidade da Origem da Mensagem;
- Autenticidade da Integridade do Conteúdo; e
- Não Repúdio.

Prover tais características a um sistema de correio eletrônico exige um certo *conjunto de ferramentas*. Do lado do emissor (*Alice*), é preciso:

- *Ciframento de Chave Pública.* O ponto central de um sistema de segurança de *e-mail* consiste na habilidade de prover comunicação segura com outras pessoas sem haver a necessidade de trocar chaves secretas com elas primeiro. Criptografia de chave pública é uma maneira de se fazer isto.
- *Assinatura Digital.* É um modo de se prover autenticação e integridade de mensagem. A criptografia de chave pública é uma forma de se fazer isto eficientemente.

Do lado do recebedor (*Bob*), é preciso:

- *Deciframento por Chave Pública*; e
- *Verificação de Assinatura Digital*.

Adicionalmente, é preciso toda uma infra-estrutura de chaves públicas, de modo que um emissor qualquer possa enviar uma mensagem cifrada a qualquer recebedor, sem terem que se encontrar previamente. É preciso também gerar e distribuir chaves, e às vezes revogar aquelas que forem furtadas ou perdidas.

É necessário que isto tudo esteja em um pacote fácil de utilizar, que interaja com todos os sistemas de correio eletrônicos da Internet e que possa ser empregado rotineiramente.

2.2.1. A Exigência de Segurança

[Schneier 95]

Primeiro e antes de tudo, um sistema de proteção de *e-mail* precisa ser seguro. Do contrário, não há razão para utilizá-lo. E, como já observado na seção 2.1, segurança é muito mais do que escolher um bom par de bons algoritmos criptográficos. Segurança é como uma corrente; ela é tão resistente quanto

o seu elo mais fraco. Em um sistema de segurança de mensagens eletrônicas, há uma porção de elos com que se preocupar:

- Algoritmos convencionais (de chave secreta) para o ciframento do conteúdo da mensagem (por exemplo, o IDEA);
- Algoritmos de chave pública para o gerenciamento de chaves (por exemplo, o RSA);
- Algoritmos de chave pública para assinaturas digitais (por exemplo: o RSA, o DSA);
- Funções de espalhamento unidirecional para emprego com assinaturas digitais (por exemplo: o SHA, o MD5);
- Geração de números aleatórios, para utilização na criação de chaves de sessão (para o algoritmo convencional);
- Geração de números primos, para uso na criação de chaves públicas e privadas;
- Armazenamento de chaves públicas e privadas;
- Procedimentos de gerenciamento de chaves;
- Cuidados ao apagar arquivos;
- *Interface* com o usuário.

Esta lista não é exaustiva. Se um órgão de inteligência do governo desejar bisbilhotar a comunicação de alguém, ele não o fará através de quebra do IDEA ou do Triplo DES. Afinal, a Criptografia é uma disciplina acadêmica bastante madura: vários algoritmos na literatura publicada vêm sendo analisados por inúmeras pessoas e têm sido considerados seguros. Ele provavelmente o fará pela exploração de alguma fraqueza em uma seção obscura do programa de segurança de correio eletrônico utilizado na comunicação.

Para validar a afirmação do parágrafo anterior, o algoritmo convencional utilizado deve possuir, atualmente, um tamanho de chave de pelo menos 112 bits. O algoritmo de chave pública deve possuir um tamanho de chave de pelo menos 1024 bits. Seguem comentários sobre alguns dos ítens mencionados:

- A geração das chaves de sessão aleatórias é mais difícil do que a escolha e implementação de um algoritmo criptográfico. Números realmente aleatórios não podem ser gerados num

computador digital. O que se pode fazer é gerar números pseudo-aleatórios, isto é, seqüências de números com uma determinada lei de formação: o conhecimento da regra de recorrência permite determinar um elemento da seqüência[28], a partir do anterior.

- O gerenciamento de chaves, como já exposto anteriormente, é também uma tarefa árdua. Há uma série de ataques possíveis contra as chaves, e um bom sistema deve levá-los todos em conta. Deve se resguardar contra chaves falsas substituindo chaves legítimas, chaves furtadas ou antigas sendo armazenadas para posterior reutilização, entre outros ataques.
- A análise de tráfego deve ser dificultada ao máximo. É um problema complexo, que não pode ser solucionado, normalmente, apenas através de um programa de segurança de *e-mail*.

Finalmente, é mais fácil quebrar um programa de segurança do que provar que ele não pode ser quebrado. Pode-se provar que uma certa pessoa não pode quebrá-lo com uma determinada quantidade de recursos em um tempo pré definido, mas isto não diz nada sobre outras pessoas tentando quebrá-lo com mais recursos, mais tempo e mais conhecimento técnico.

2.2.2. A Flexibilidade

[Schneier 95]

Um bom programa de segurança de correio eletrônico precisa ser flexível. Ele deve permitir aos usuários o envio de mensagens cifradas não assinadas, assinadas não cifradas e mensagens assinadas e cifradas. Deve cifrar as mensagens que armazena. Deve possibilitar o envio de mensagens tanto para um único quanto para múltiplos recebedores.

Deve também estar disponível para todo tipo de plataforma, seja UNIX ou MS-DOS, por exemplo. Pessoas que utilizam um sistema operacional devem ser capazes de enviar mensagens seguras para pessoas que utilizam outro sistema. Em resumo, quanto mais flexível for o programa de segurança, maior utilidade ele terá. Quanto maior sua utilidade, maior sua presença nos mais variados ambientes e máquinas - objetivo final de todo programa de segurança de *e-mail*.

[28] O primeiro elemento da seqüência em pauta é denominado *semente*.

2.2.3. A Adaptabilidade

[Schneier 95]

Um bom programa de segurança de *e-mail* deve permitir o emprego de uma variedade de diferentes algoritmos. Alguns usuários podem preferir o DES, outros o IDEA ou o Triplo DES. Se a comunidade científica descobrir que um determinado algoritmo criptográfico pode ser quebrado, seria uma simples questão de substituí-lo por outro. Bruce Schneier faz uma afirmação bastante interessante em um de seus livros [Schneier 95]:

"No mínimo, acho que um programa de segurança de mensagens eletrônicas deveria utilizar os seguintes algoritmos:

- Algoritmos para Ciframento de Dados:

  Triplo DES.

- Algoritmo para Gerenciamento de Chaves:

  ElGamal.

- Funções de Espalhamento Unidirecional:

  SHA (sigla para *Secure Hash Algorithm*, da NSA. Produz um valor *hash* de 160 bits a partir de uma mensagem de tamanho arbitrário*).

- Algoritmo para Assinatura Digital:

  ElGamal.

Outros possíveis algoritmos, embora em minha opinião (na dele*) escolhas menos ideais, são:

- Algoritmos para Ciframento de Dados:

  DES

  IDEA

- Algoritmo para Gerenciamento de Chaves:

  RSA

- Funções de Espalhamento Unidirecional:

  MD5

- Algoritmo para Assinatura Digital:

  RSA

  DSA."

Motivos que Bruce Schneier alega para as escolhas efetuadas:

"Prefiro ElGamal a RSA devido a questões de patente. ElGamal entra em domínio publico em 1997. Os vários algoritmos de chave publica - RSA, ElGamal e DSA deveriam possuir chaves de pelo menos 1024 bits. Prefiro Triplo DES (112 bits*) no lugar do DES (56bits*) porque é mais seguro. O DES está atingindo o final de sua vida criptográfica útil.

Prefiro Triplo DES a IDEA porque o Triplo DES foi analisado muito mais exaustivamente do que o IDEA. Finalmente, prefiro o SHA (valor *hash* de 160 bits*) ao MD5 (valor *hash* de 128 bits*) porque o SHA tem um valor *hash* mais longo e aparenta ser mais seguro."

2.2.4. A Interface com o Usuário

[Schneier 95]

Se for perguntado a um comandante de um submarino se ele deseja ou não um novo componente em seu barco, ele responderá: "somente se ele não provocar nenhum ruído, não ocupar nenhum espaço, não utilizar nenhuma energia de bordo e contribuir para a melhora do desempenho do submarino". Segurança computacional é muito parecido com isto. As pessoas a desejam, porém somente se ela não exigir nenhuma memória, não afetar o desempenho do sistema e for transparente ao usuário. Infelizmente, isto é impossível.

Mesmo assim, deseja-se que um programa de segurança de correio eletrônico chegue o mais perto possível desta meta. O envio de uma mensagem eletrônica assinada e cifrada deveria ser a opção rotineira em qualquer programa deste tipo. O usuário deveria realizar uma tarefa extra para o envio de mensagem sem assinatura e/ou não cifrada.

O programa deveria também estar apto a produzir mensagens cifradas que passam através de todos os sistemas de *e-mail* sem alteração. E também a decifrar mensagens no destino que tenham passado

* As informações entre parênteses foram inseridas pelo autor deste trabalho.

através de uma grande variedade de *gateways* de *e-mail*. Em geral, isto deveria ser efetuado com a mínima intervenção do usuário; no caso ideal, com nenhuma intervenção.

Em suma, uma boa *interface* de usuário é mais difícil de se conseguir do que boa segurança. O mundo da Internet está repleto de diferentes computadores, sistemas operacionais, programas de *e-mail* e usuários. Todas as pessoas têm suas próprias preferências, e é virtualmente impossível fazer um programa que trabalhe bem em todos os ambientes.

3. O Programa Sendmail

Existem vários protocolos, padrões e programas utilizados na troca de mensagens, dentre os quais podem ser citados: os protocolos MIME, POP3 e SMTP, o padrão X.400, o conjunto de programas UUCP e o programa Sendmail. No Capítulo 1 - Funcionalidade - foi enfatizado o protocolo SMTP, utilizado na Internet. Neste Capítulo, será enfatizado o programa Sendmail, intimamente ligado ao SMTP e importante para um melhor entendimento do sistema ProtegeMail.

A seção 3.1 provê um breve histórico e uma descrição sucinta do Sendmail. Pretende-se, na seção 3.2, mostrar como funciona seu arquivo de configuração (*sendmail.cf*). A seção 3.3 fornece uma visão global do que vem a ser o diretório de enfileiramento (*/var/spool/mqueue*). O arquivo de pseudônimos e o reenvio automático de mensagens são apresentados na seção 3.4.

3.1. Descrição do Sendmail

[Allman 95] [Costales 94] [Cavalcanti 96] [Chaves 96]

O programa *sendmail* foi originariamente escrito por Eric Allman, na Universidade da California, em Berkeley. Seu precursor, o programa *delivermail*, foi lançado em 1979. Em 1980, houve uma série de mudanças na ARPAnet[29], entre elas a conversão ocorrida do NCP (*Network Control Protocol*) para o TCP (*Transport Control Protocol*), fazendo com que os *e-mails* não mais fossem enviados pelo protocolo *ftp* e sim pelo SMTP.

Em resposta a estas mudanças, Allman converteu o *delivermail* no programa *sendmail*. Mas ele não foi o único a trabalhar com o *sendmail*. Em 1987, Lennart Lovstrand, da Universidade de Linköping, Suécia, desenvolveu os incrementos IDA para a versão BSD 5 deste programa, injetando uma série de aperfeiçoamentos e consertando vários *bugs*. Neil Rickert e Paul Pomes, desenvolveram a versão UIUC IDA e Paul Vixie, a versão KJS do *sendmail*, dando continuidade ao trabalho de Lennart.

O *sendmail* é bem mais do que um simples programa. Capaz de receber e enviar *e-mails* via SMTP, é o MTA da Internet mais utilizado em ambiente UNIX[30] . Composto por uma coleção de programas, arquivos, diretórios e serviços, ele oferece uma solução para a coexistência de diferentes protocolos de troca de mensagens em um mesmo servidor de *e-mail* [31]. Age como um despachante e não se restringe a ser apenas um MTA SMTP; ao contrário, é um sistema aberto que escolhe que MTA é mais adequado para o envio de um determinado *e-mail* (Figura 3.1).

---

[29] ARPAnet é o antigo nome da Internet. Criada em 1969 pelos militares americanos, inicialmente interligava computadores de grande porte com o NCP. A partir de 1983, quando começou a utilizar o protocolo TCP/IP, foi se tornando cada vez mais uma rede de redes e cada vez menos uma rede de computadores. Data desta época a oficialização e o uso do termo "Internet".

[30] O *sendmail* pode ser uma porta aberta a abusos. Sua má utilização ou configuração mal feita pode conduzir a um sistema de correio eletrônico inseguro e possivelmente comprometido. Como normalmente é instalado para executar como um processo *suid root*, se não for adequadamente instalado pelo administrador, ele pode ser explorado por um intruso para a obtenção de privilégios de *root* ou para a geração de *e-mails* falsos com a intenção de obter a senha de algum usuário.

[31] A variedade de protocolos e seus correspondentes MTA complicam a configuração e o suporte de um servidor de *e-mail* ligado a redes distintas. Mensagens em redes TCP/IP são, em geral, enviadas pelo SMTP via *sendmail*; em redes UUCP, são encaminhadas via *uux* ou *rmail*; um outro programa (o UA) encaminha *e-mails* locais, e assim por diante.

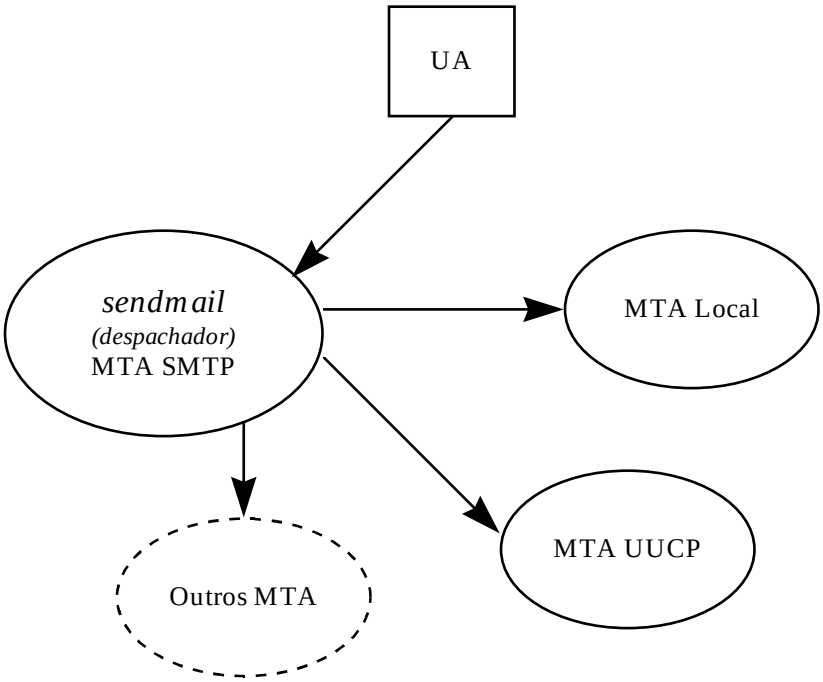


Figura 3.1 - Roteamento de mensagem via *sendmail*.

A chave do *sendmail* consiste em seu arquivo de configuração (*configuration file*), que define a localização e o comportamento de todas as suas partes e inclui regras para a reescrita de endereços. Um diretório de enfileiramento (*queue directory*) armazena a mensagem não entregue até que ela possa ser distribuída. Um arquivo de pseudônimos (*aliases file*) permite a utilização de nomes alternativos para usuários e possibilita o desenvolvimento de listas de discussão pelo reenvio de mensagens.

Os formatos dos endereços diferem de um MTA para outro. Por isto, o *sendmail* utiliza o endereço do recebedor de uma mensagem como critério para a escolha de um MTA entregador. A diferença no formato de endereço, entre MTA distintos, também gera a necessidade da reescrita de endereços do emissor e do recebedor. A reescrita é feita de acordo com o formato exigido pelo MTA selecionado pelo *sendmail*. Surge, então, a necessidade de *regras para a reescrita* de endereços, detalhadas no arquivo de configuração (*sendmail.cf*).

3.2. O Arquivo de Configuração (sendmail.cf)

[Allman 95] [Costales 94] [Cavalcanti 96]

A posição do programa *sendmail* na hierarquia do sistema de arquivo local pode ser vista como uma árvore. O *sendmail*, ao ser executado[32], lê e analisa primeiro o seu arquivo de configuração */etc/sendmail.cf*, situado na raiz desta árvore (Figura 3.2). O arquivo de configuração do programa *sendmail* consiste em um arquivo de texto contendo todas as informações de que o *sendmail* necessita para funcionar corretamente.

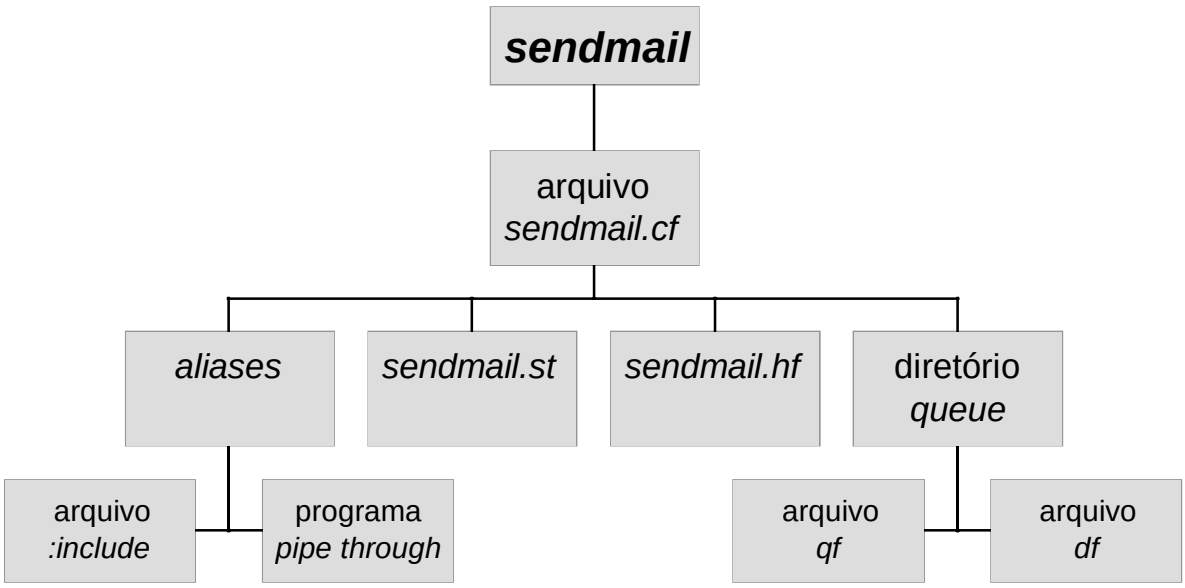


Figura 3.2 - O *sendmail.cf* Conduz a Todos os Demais Arquivos [Costales 94].

Assim, toda a configuração do *sendmail*, com exceção do serviço de pseudônimos (que utiliza o arquivo */etc/aliases*), é feita através do arquivo */etc/sendmail.cf*. Ele contém as informações requeridas para o encaminhamento de *e-mails*, incluindo como deve ser o roteamento das mensagens e as regras de reescrita de endereços, conforme o MTA a ser utilizado. Assim, o arquivo de configuração contém opções para a modificação do comportamento do *sendmail*.

Cada uma de suas linhas contém uma parte destas informações. Um tipo de linha proporciona uma maneira de representar endereços e outros textos simbolicamente. Estes símbolos podem, posteriormente, substituir o texto original. Um outro tipo de linha permite que múltiplos itens de texto

[32] O *sendmail* é executado toda vez que um *e-mail* é enviado.

sejam representados por um único símbolo. Há também um tipo para prover informações necessárias ao *sendmail*, tais como: localizações dos arquivos importantes, permissões e modos de operação.

Regras para reescrita (comando R) e *conjuntos de regras* (comando S) são linhas utilizadas para converter um endereço em outro que venha a ser requerido para a distribuição da mensagem. Consistem no cerne do *sendmail*. Através delas, é possível o mapeamento de um endereço no formato utilizado pelo usuário para o atendimento das necessidades dos diversos MTA (Tabela 3.1).

Conjunto de Regras	***Propósito***
0	Determina um MTA para a entrega do *e-mail*, a partir do endereço do recebedor.
1	Processa o endereço de um emissor.
2	Processa o endereço do recebedor.
3	Pré processa todos os endereços.
4	Pós processa todos os endereços.
5	Processa os cabeçalhos do emissor (somente versão IDA).
5	Reescreve usuários locais sem pseudônimos (somente versão 8).
6	Processa os cabeçalhos do recebedor (somente versão IDA).

Tabela 3.1 - Os propósitos dos Conjuntos de Regras [Costales 94].

Segue um exemplo de um arquivo *sendmail.cf* completo (Figura 3.3).

```
##################################################################
# A sendmail.cf file.
#
# $Date: 1993/09/28 19:30:09 $
##################################################################
# Other names for the local host
Cwlocalhost
# This is the name of the mail hub
DHhub.your.domain
# This is the name of the mail relay
DRmailhost
# Our official canonical hostname.
Dj$w
## Standard macros
# name used for error messages
DnMailer-Daemon
# UNIX header format
DlFrom $g  $d
# seperator (operator) characters
Do.:%@!^=/[]
# format of a total name
Dq<$g>
# Process messages in the background.
Odbackground
# Default permissions for files
OF0600
# Default user and group (daemon/daemon)
Og1
Ou1
# The logging level
OL9
# Accept oldstyle addresses
Oo
# Send a copy of bounced messages to postmaster
OPPostmaster@$H
# SMTP read timeout
Or15m
# Note, we do queue, in case hub and MX's are all down.
OQ/var/spool/mqueue
# Time to live in the queue
OT5d
###   Message precedences (all of equal weight. let hub decide.)
Pfirst-class=0
Pspecial-delivery=0
Pjunk=0
###   Trusted users
T root daemon
## Required headers
HReceived: $?sfrom $s $.by $j ($v) id $i; $b
H?D?Date: $a
H?F?From: $q
H?M?Message-Id: <$t.$i@$j>
H?D?Resent-Date: $a
H?F?Resent-From: $q
H?M?Resent-Message-Id: <$t.$i@$j>
H?x?Full-Name: $x
S0 # Punt to hub
R$*                                     $#ether $@$R $:$1
S3 # local users made to look like they are from the hub
R$*<$*<$*>$*>$*              $3                      denest
R$*<$+>$*      $2                       basic RFC822 parsing
R$*<>$*                      $n                      RFC1123 <>
R$-                          $@ $1 @ $H     user => user@hub
R$+@$+                       $: $1 @ $[$2$] canonify the hostname
R$+@$=w                      $@ $1 @ $H     user@thishost => user@hub
R$=w!$+                      $@ $2 @ $H     thishost!user => user@hub
R$+%$=w                      $@ $>3 $1 @ $2 handle % hack thishost
R$*                          $@ $1                   default, unchanged
Mether,        P=[IPC], F=mDFMuCX, S=0, R=0, A=IPC $h
Mlocal, P=xxx, A=Required by sendmail but unused
Mprog,  P=xxx, A=Required by sendmail but unused
```

Figura 3.3 - Exemplo de Um Arquivo *sendmail.cf* .

3.3. O Diretório de Enfileiramento

[Allman 95] [Costales 94] [Cavalcanti 96]

Uma mensagem pode ficar temporariamente sem ser distribuída por uma série de razões, tais como a queda de uma máquina remota ou um problema temporário de disco. Para garantir que esta mensagem seja difundida, o programa *sendmail* a armazena em seu diretório de enfileiramento (*/var/spool/mqueue*) até que ela possa ser entregue. A linha OQ do arquivo de configuração indica ao *sendmail* onde encontrar seu diretório de enfileiramento de *e-mails*:

```
OQ/var/spool/mqueue
```

Quando uma mensagem é enfileirada, ela é dividida em duas partes, cada uma delas armazenada dentro do diretório de enfileiramento em um arquivo separado. A informação do cabeçalho fica armazenada em um arquivo cujo nome começa com os caracteres *qf*. O corpo da mensagem é guardado em um arquivo cujo nome começa com os caracteres *df*. O *sendmail* periodicamente segue a fila e tenta entregar cada mensagem. Caso não seja bem sucedido, ele devolverá o *e-mail* ao seu emissor, acrescido de uma mensagem de erro.

3.4. O Arquivo de Pseudônimos e o Reenvio de Mensagens

[Allman 95] [Costales 94] [Cavalcanti 96]

Um pseudônimo permite que o envio de uma mensagem eletrônica a um determinado endereço possa ser redirecionado para outro endereço. Também possibilita que as mensagens sejam armazenadas em arquivos ou processadas através de programas, formando a base das listas de discussão. Uma utilidade do serviço de pseudônimos é converter um nome genérico, como *root*, em um nome de usuário verdadeiro. Outra é converter um nome em uma lista contendo vários nomes (como nas listas de discussão).

O cerne do serviço de pseudônimos é o arquivo *aliases*. Sua localização é determinada pela linha de opção OA do arquivo *sendmail.cf*. Por exemplo:

```
OA/etc/aliases
```

No arquivo *aliases*, as linhas começando com # são comentários. Linhas vazias são ignoradas. Modificações feitas neste arquivo somente se tornam válidas após a execução do comando *newaliases*. Há duas linhas que são obrigatórias em todo arquivo *aliases* : *postmaster* e *MAILER-DAEMON*. A linha *postmaster* é necessária porque os *e-mails* sobre problemas no correio eletrônico são sempre enviadas para este pseudônimo. A linha *MAILER-DAEMON* é importante porque, quando uma mensagem retorna por não poder ter sido entregue por qualquer motivo, ela sempre será emitida pelo *MAILER-DAEMON* .

Ambas estão na forma mais simples de pseudônimo: um nome, seguido de dois pontos e de outro nome. Assim, o nome à esquerda dos dois pontos será convertido no nome à direita. Há cinco formas que uma linha em um arquivo *aliases* pode assumir, conforme exemplo a seguir (Figura 3.4):

```
# Pseudônimos obrigatórios.
postmaster :  root
MAILER-DAEMON :     postmaster

# As cinco formas de linhas de definição de pseudônimos.
root :          pagliusi
xpres:          figueiredo,sarney,collor,itamar,fhc
oldlist:        :include:       /usr/local/oldguys
nobody:         /dev/null
ftphelp:        |/usr/local/bin/sendhelp
```

Figura 3.4 - Exemplo de Um Arquivo */etc/aliases* .

- A primeira forma é a mais simples, utilizada nas linhas obrigatórias já mencionadas. No exemplo, um *e-mail* enviado para o endereço *root*, será entregue para o usuário cujo nome de *login* é *pagliusi*.

- A linha *xpres:* ilustra que um nome pode ser expandido em uma lista contendo vários nomes. É entregue uma cópia da mensagem eletrônica a cada um destes novos nomes, sendo que cada um deles se torna um novo nome para um futuro processo de verificação de pseudônimo.

- A linha *oldlist :* demonstra que uma lista de discussão (ou de distribuição) pode ser lida a partir de um arquivo. A expressão *:include:* faz com que o *sendmail* leia um arquivo e utilize os nomes contidos nele como a lista de recebedores.

- A linha *nobody:* ilustra que um nome pode ter um arquivo como um pseudônimo. Assim, a mensagem eletrônica será anexada ao arquivo indicado. O arquivo */dev/null* do exemplo é um arquivo especial. Este arquivo é um "buraco" vazio, onde a mensagem simplesmente desaparece.

- A linha *ftphelp:* demonstra que um nome pode ser convertido no nome de um programa. O caracter | faz o *sendmail* processar (*to pipe*) a mensagem eletrônica através do programa cujo caminho encontra-se indicado. No exemplo, o caminho é */usr/local/bin/sendhelp*.

A atribuição de pseudônimos também encontra-se disponível para o usuário individual através de um arquivo chamado *.forward* disponível em seu diretório *home*. Para ativar o reenvio das mensagens para outro endereço, o conteúdo do *.forward* deve ser composto do endereço eletrônico desejado.

No tratamento automático de mensagens recebidas, o *sendmail* também oferece, além dos recursos de reenvio de mensagens, o envio automático de mensagens em resposta às mensagens recebidas. A resposta automática torna-se bastante útil em situações em que o usuário não está podendo consultar sua *mailbox* ; por exemplo, quando tira férias. Este recurso também pode ser configurado através do arquivo *.forward*, utilizando-se o comando *vacation*. Este comando editará também o arquivo *.vacation.msg*, contendo a mensagem de resposta desejada (informando, por exemplo, que o usuário está ausente e que data está prevista para seu regresso).

Finalizando, o arquivo de pseudônimos do *sendmail* tem o potencial de tornar-se bastante complexo. Contudo, ele pode ser utilizado na solução de vários problemas especiais de correio eletrônico.

Parte II

PGP & PEM

4. Criptografia

A **criptografia** consiste na ciência de escrever a informação em cifras, de forma a que somente as pessoas autorizadas possam compreendê-la. O seu objetivo é a construção de cifras invioláveis que garantam a privacidade e a autenticidade das mensagens transmitidas [Cavalcanti 96].

Este capítulo tem como propósito, na seção 4.1, enfatizar as técnicas criptográficas como ferramentas para a elaboração de um sistema de correio eletrônico seguro. Além disso, pretende-se, na seção 4.2, complementar os conceitos sobre criptografia, já introduzidos no Capítulo 2, pela descrição dos algoritmos criptográficos utilizados para o ciframento, o gerenciamento de chaves e a autenticação de

mensagens eletrônicas nele citados. Os cuidados para a proteção de chaves criptográficas também serão abordados neste capítulo, através da sua seção 4.3.

4.1. Técnicas Criptográficas

[Stinson 95] [Lucchesi 86] [Schneier 96] [Cavalcanti 96] [Schneier 95] [Tanenbaum 92] [Coulouris 94] [Bellovin 94]

A resposta para muitos dos problemas apresentados no Capítulo 2 é o uso de técnicas criptográficas. Por técnicas criptográficas entende-se que são as *funções e mecanismos de segurança* que vão ser usados para garantir a privacidade e a autenticação no sistema de correio eletrônico. *Funções de segurança* são funções específicas, relevantes para a segurança, utilizadas para prover um ou mais serviços de segurança. E *mecanismos de segurança* são métodos específicos que implementam uma ou mais funções ou serviços de segurança. As técnicas criptográficas básicas são descritas nos ítens a seguir.

4.1.1. Ciframento

[Stinson 95] [Lucchesi 86] [Schneier 95]

O ciframento consiste em tornar ininteligível um certo texto originalmente claro, pelo uso de uma função que contém um parâmetro secreto chamado *chave* (*trapdoor* ou alçapão). A maneira de se fazer isso é usar uma função unidirecional (*one-way*), que é uma função com as seguintes características:

- dado x, é fácil calcular $f(x)$;
- dado $f(x)$, é muito difícil calcular $f^{-1}(x)$; ou seja, $f(x)$ é computacionalmente inviável de se calcular.

Quando tais funções incluem uma chave (*trapdoor)* - que permite a quem a possui inverter facilmente a função, elas são denominadas funções unidirecionais com *trapdoor*. Estas são as principais funções utilizadas para a construção de algoritmos de ciframento.

Dessa forma, mesmo que o intruso *Eve* conheça uma função unidirecional com *trapdoor*, ele não poderá obter o texto claro correspondente ao texto cifrado, sem antes conhecer sua chave (isto será computacionalmente impossível de se fazer).

Há dois tipos principais de algoritmos de ciframento que implementam funções unidirecionais com *trapdoor:* algoritmos de chave secreta (também chamados de convencionais ou simétricos) e algoritmos de chave pública (ou assimétricos).

- Os algoritmos **de chave secreta, simétricos** ou **convencionais** são utilizados para ocultar a informação que fica exposta em algumas partes do sistema de *e-mail*, entre elas os canais físicos de comunicação. Este uso corresponde ao uso tradicional da criptografia em atividades militares e de diplomacia. Ele explora o fato de que uma mensagem cifrada por um emissor (*Alice*), com uma chave secreta de ciframento, somente pode ser decifrada por um recebedor (*Bob*) que também conheça a chave em pauta - a chave de deciframento ou é a mesma utilizada para o ciframento ou é uma chave facilmente derivada desta última. A figura a seguir (Figura 4.1) mostra o esquema de um algoritmo criptográfico convencional ou de chave secreta. Nele, o espião (pessoa ou processo) é o intruso (*Eve*) que deseja burlar o sistema, mas não consegue, pois somente possui acesso ao *texto cifrado* da mensagem O *texto original* (ou seja, a mensagem clara) está bem guardado, através de uma função criptográfica de ciframento (que pode ser inclusive um algoritmo contendo várias funções) e de uma chave secreta de ciframento.

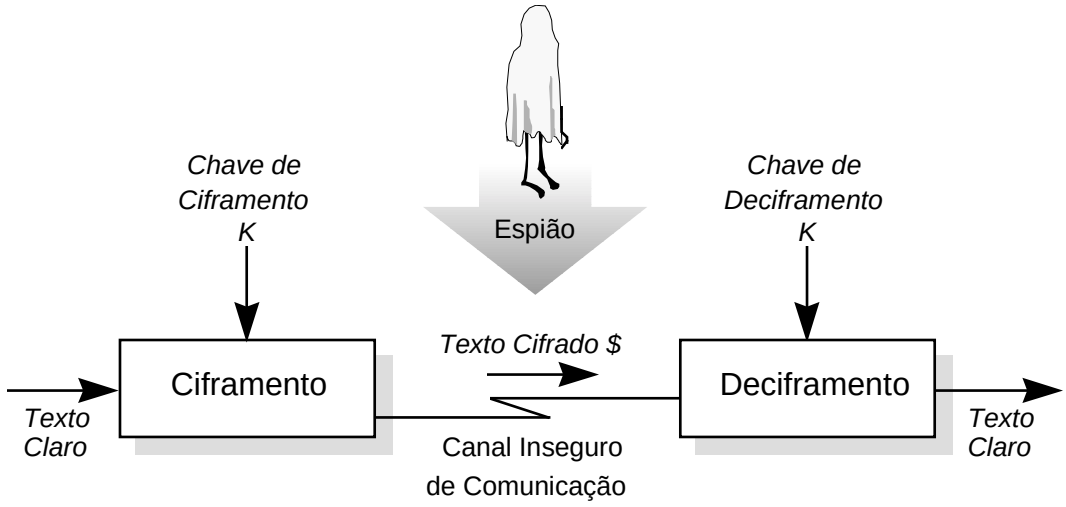


Figura 4.1 - Sistema Criptográfico Convencional [Lucchesi 86].

Assim, um algoritmo de chave secreta implementa as seguintes operações:

$ = f (K , claro); e

claro = f (K , $).

Onde:

f é o algoritmo de chave secreta empregado para cifrar e decifrar ;

K é a chave secreta;

$ é o texto cifrado produzido por *Alice*; e

claro é o texto original de *Alice* a cifrar.

A tabela a seguir apresenta e compara alguns algoritmos de ciframento convencional ou de chave secreta (Tabela 4.1).

Algoritmo De Chave Secreta	*Tamanho Da Chave (em bits)*	*Tamanho do bloco (em bits)*	*Observações*
DES	56	64	Chave pequena.
DES Triplo	112	64	Lento.
Blowfish	var. (< 448)	64	Faltam ataques.
Khufu	64	64	Patenteado, chave pequena.
FEAL-32	64	64	Patenteado, chave pequena.
LOKI-91	64	64	Chave pequena.
REDOC II	160	80	Patenteado.
REDOC III	variável	64	Patenteado.
IDEA	128	64	Patenteado.
RC2	variável	64	Proprietário.
Skipjack	80	64	Algoritmo Secreto.
GOST	256	64	Faltam especificações.
MMB	128	128	Inseguro.
SAFER	64	64	Patenteado, chave pequena.

Tabela 4.1 - Tabela Comparativa de Algoritmos de Chave Secreta.

- os **algoritmos de chave pública** ou **assimétricos** são normalmente utilizados no suporte a mecanismos de autenticação de comunicação entre pares de agentes ativos *Alice* e *Bob* (agentes que

podem ser pessoas ou processos). Eles utilizam duas chaves distintas: uma para cifrar e outra para decifrar. Um delas, a *chave pública*, é divulgada sem restrições; a outra, a *chave privada*, é mantida em segredo, sendo jamais divulgada. A segurança de sistemas deste tipo baseia-se no fato de ser computacionalmente caro deduzir a chave privada (SA) a partir da chave pública (PA). Um agente que decifra com sucesso uma mensagem, usando uma chave pública, como pode ser observado na figura a seguir (Figura 4.2), assume que a mensagem é autêntica se ela contiver algum valor esperado.

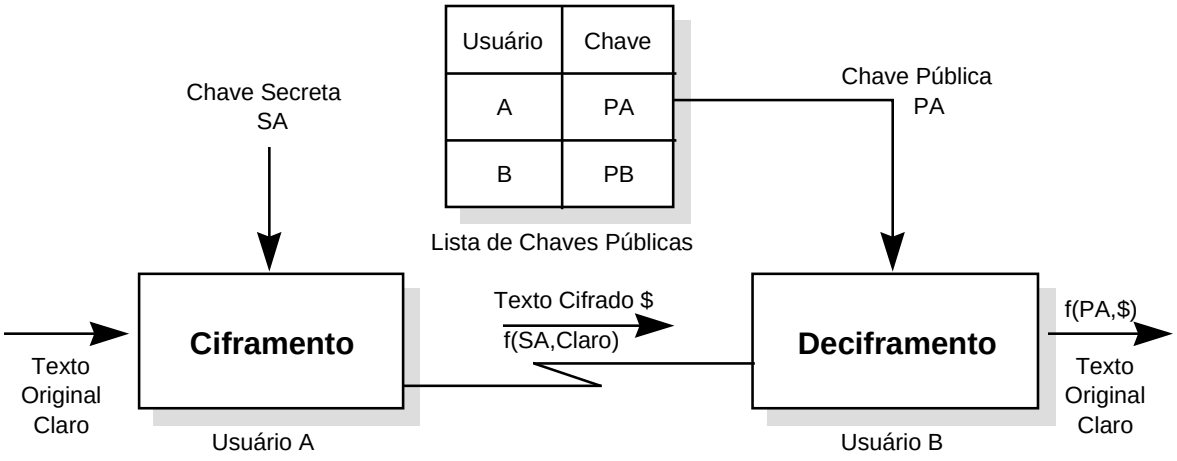


Figura 4.2 - Sistema de Chave Pública Para Autenticação.

Uma grande aplicação de chave pública é em *protocolos de autenticação***,** através dos quais garante-se que a mensagem, além de ser autêntica, não é uma repetição, ou seja, um *replay*. Estes protocolos asseguram, portanto, que a mensagem é recente, não se tratando de reenvio de uma mensagem antiga.

Um algoritmo de chave pública implementa as seguintes operações para cifrar uma mensagem[33] (Figura 4.3):

[33] Note-se o efeito de confidencialidade obtido ao se cifrar só com a chave pública, e o efeito de autenticidade ao se cifrar com a chave privada.

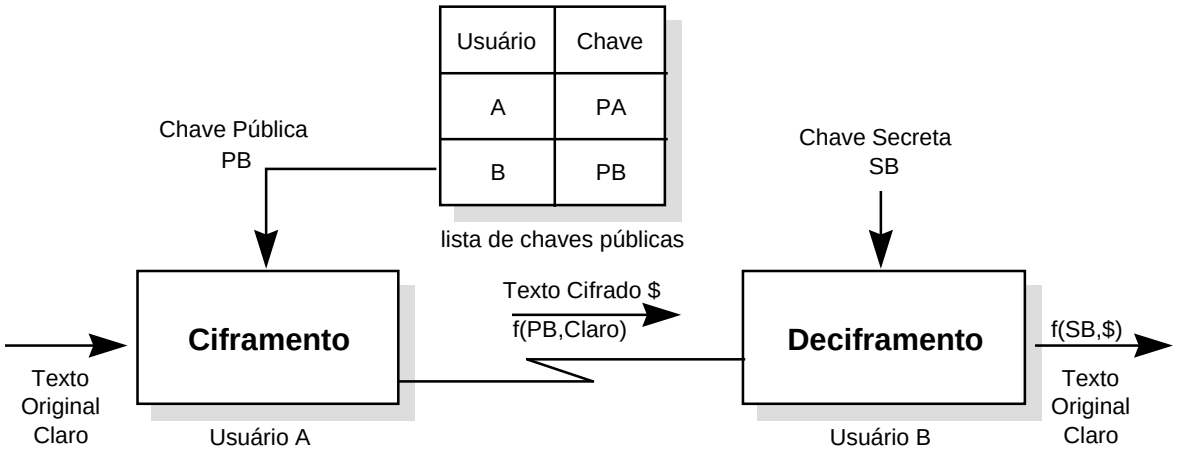


Figura 4.3 - Sistema de Chave Pública para Ciframento.

$ = f (PB, claro); e

claro = f (SB , $).

Onde:

f é o algoritmo de chave pública empregado para cifrar e decifrar;

PB é a chave pública de *Bob*, inserida em um catálogo público;

SB é a chave privada de *Bob*;

$ é o texto cifrado produzido por *Alice*; e

claro é o texto de *Alice* original a cifrar.

Um exemplo: o RSA.

Os algoritmos também podem ser classificados quanto ao comportamento da chave dentro do algoritmo, e então podem ser cifras *de bloco* ou *de stream*. Nas cifras de bloco, a chave não se altera durante o ciframento de um texto. Se a mesma entrada em claro ocorrer, é produzido o mesmo texto cifrado. Nas cifras de *stream*, também chamadas de *running-key generators*, o preenchimento interno da chave se altera durante o ciframento de uma mensagem, de modo que a mesma entrada produzirá saídas diferentes ao longo do ciframento.

Os algoritmos de ciframento podem ser utilizados para implementar não apenas confidencialidade, autenticação e integridade, como também protocolos de controle de acesso e de identificação. Os algoritmos de chave pública são particularmente adequados à distribuição de chaves.

4.1.2. Assinatura Digital

[Stinson 95] [Lucchesi 86] [Schneier 95]

Uma assinatura digital caracteriza-se por ser: fácil de se produzir, fácil de se reconhecer, e difícil de se falsificar. Assim, a assinatura digital imita a regra das assinaturas convencionais. Há, portanto, uma comprovação perante terceiros (que desempenham o papel de cartório) de que uma mensagem é uma cópia inalterada de uma produzida por um agente específico.

A capacidade de proporcionar assinaturas digitais depende de haver algo que o emissor (*Alice*) original possa fazer que outros não possam. Isto pode ser obtido, por exemplo, solicitando-se a uma terceira pessoa confiável, que tem prova da identidade do emissor original, para cifrar uma mensagem ou, então, uma forma reduzida da mensagem, denominada *digest*.

A mensagem cifrada resultante ou o *digest* agem como uma assinatura convencional do emissor, que acompanha a mensagem. Ela pode ser comprovada por qualquer recebedor que peça à mesma terceira pessoa ou a um cartório idôneo (podendo este ser inclusive um *servidor de autenticação*) para cifrar a mensagem de novo. Se o resultado conferir, a assinatura está comprovada.

É possível, ainda, obter assinatura digital também por chave pública. Dado um texto original ***claro***, um usuário (*Alice*) o assina com um algoritmo de chave pública ***f***, usando uma chave privada ***SA***, produzindo um texto cifrado ***$***, da seguinte maneira (conforme a Figura 4.2):

$$\$ = f(SA, claro).$$

Assim, somente *Alice* pode ter assinado este texto. Para ler o texto original assinado desta forma, é só usar a chave pública de *Alice*[34]:

$$claro = f(PA, \$).$$

[34] Nem todo sistema de chave pública é comutativo, ou seja, $f(PA, f(SA, claro)) = claro$ para todo texto original claro.

Possíveis aplicações das assinaturas digitais: na autenticação de mensagens, aí incluídas a identificação, integridade e não repúdio - produção de provas. Outra aplicação interessante, onde se obtém autenticação e confidencialidade simultâneas, usando-se um algoritmo de chave pública ***f***, é:

$ = f (PB , f (SA , claro));

Onde:

SA é chave privada do emissor do texto ***claro*** (no exemplo, *Alice*); e

PB é a chave pública do recebedor do texto ***$***, assinado e cifrado (no exemplo, ***Bob***).

A aplicação acima é útil em todo protocolo que envolva a necessidade de autenticação, confidencialidade, auditoria e distribuição de chaves.

4.1.3. Certificados

[Stinson 95] [Lucchesi 86] [Schneier 95]

Um certificado é uma mensagem autenticada de modo criptográfico - por exemplo, contendo uma chave criptográfica. Um certificado é uma credencial que serve como prova junto a terceiros. O uso de certificados responde à seguinte pergunta*: a chave é autêntica ?* Por isso, a resposta a essa pergunta normalmente precisa ser dada por uma terceira pessoa de confiança - a autoridade de certificação CA (o "tabelião"), que emite certificados confiáveis.

Dado um algoritmo de chave pública ***f***, obtém-se um certificado cifrado ***C$*** como a seguir:

C$ = f (ST , f (PA , Alice));

onde:

Alice é o nome do dono do certificado;

PA é a chave pública de *Alice*; e

ST é a chave privada do CA (*T* de “tabelião”).

Aplicação : Autenticação - prova de autorização e gerenciamento de chaves.

4.1.4. Funções Hash, Message Digest, Message Authentication Code

[Stinson 95] [Lucchesi 86] [Schneier 95]

Normalmente, cifrar uma mensagem inteira para usar como assinatura digital é muito demorado. Assim sendo, é mais rápido usar uma função especial chamada de função de espalhamento (ou *hash*) que possui as seguintes propriedades (Figura 4.4):

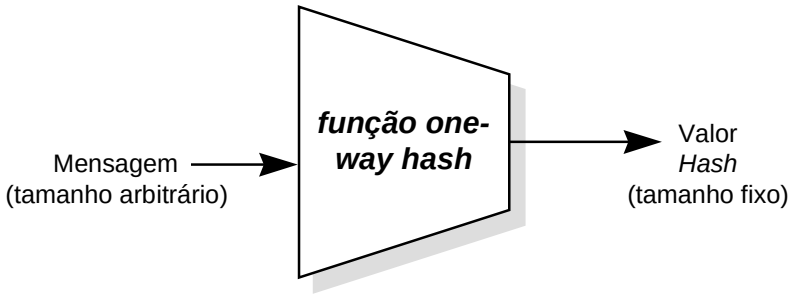


Figura 4.4 - Função *One-Way Hash* [Schneier 95].

- gera uma saída pequena de tamanho fixo (p. ex.: 128 bits), para uma entrada de qualquer tamanho;
- é computacionalmente impossível invertê-la[35]; e
- É computacionalmente impossível achar duas mensagens com o mesmo valor *hash*, ou seja, o mesmo *message digest* .

Outra aplicação das funções hash é na detecção de modificação de mensagens. Isto pode ser feito usando-se uma técnica chamada de *cryptographic checksum* (ou MAC - *Message Authentication Code* ou MDC). Assim, não se consegue adulterar um texto e manter o mesmo resultado ou valor *hash*.

Exemplos de algoritmos que implementam funções *hash*: SHA (criado pelo NIST), MD5 (da firma RSA Data Security, é usado no PGP).

Aplicação das funções *hash*:: Autenticação - assinatura digital e integridade.

[35] Ou seja, é impossível recuperar a mensagem original a partir do valor *hash*.

4.1.5. Timestamp

[Stinson 95] [Lucchesi 86] [Tanenbaum 92]

Apesar de não ser exatamente uma função criptográfica, é um serviço que é parte importante de muitos protocolos de segurança de mensagens. Se uma mensagem eletrônica receber o *timestamp* antes de ser verificada sua validade, o recebedor terá em mãos uma garantia de não repúdio da origem, evitando o ataque da meia-noite (ou *Replaying*, descrito no item 2.1.3). Outra maneira de se empregar este serviço, é verificar se uma mensagem demorou mais do que um certo período de tempo para chegar. Caso isto ocorra, então ela pode ser um *replay*.

Um exemplo deste serviço: pode-se criar uma seqüência de mensagens encadeadas, usando-se o chamado *link value* (L_n), da seguinte maneira:

$$L_n = H(T_n, H(claro), n, L_{n-1});$$

onde:

H é uma função *hash* ;

T_n é a data/hora de envio da mensagem;

n é o número da mensagem; e

claro é o texto original em claro.

Aplicação: Autenticação por *timestamp* e não repúdio.

4.1.6. One-Time Passwords e Algoritmos One-Time Pad

[Lucchesi 86] [Schneier 95]

Uma senha descartável (ou *One-Time Password*) consiste em uma forma de se evitar que seja fornecida sempre a mesma senha. Evita-se, assim, que um intruso *Eve* use algum método de ataque para obter informação que permita a ele se autenticar junto ao sistema, derivando daí sua identificação e autorização.

Segue um exemplo:

Nome do usuário (*username*): *pagliusi*

desafio: *4a82d07c*

resposta: *c1f1e904*

Protocolo *One-Time Password*:

1) O computador envia:

$$d = f(SC, f(PU, a))$$;

2) O usuário calcula:

$$a = f(PC, f(SU, d))$$;

3) O usuário envia:

$$r = f(PC, f(SU, a \oplus FFFFFFFF_{16}))$$; e

4) O computador confere:

$$a \oplus FFFFFFFF_{16} \stackrel{?}{=} f(SC, f(PU, r))$$.

Onde:

f é o algoritmo de chave pública empregado para cifrar e decifrar;

PU e SU são, respectivamente, as chaves pública e privada do usuário;

PC e SC são, respectivamente, as chaves pública e privada do computador;

a é um número aleatório;

d é o desafio; e

r é a resposta do usuário.

A chave privada do usuário SU pode ser guardada em um cartão inteligente (*smart card*) com PIN.

Exemplos de produtos já existentes: o CRYPTOcard e o ActivCard.

Já um algoritmo *One-Time Pad* consiste no único esquema de ciframento que provou ser absolutamente inquebrável. Ele é bastante utilizado na manipulação de informações ultra-secretas, por não precisar de nenhum *hardware* para ser implementado e devido à sua segurança absoluta. Durante anos a linha direta (*hotline)* entre Moscou e Washington foi cifrada com um algoritmo *one-time pad.*

O algoritmo exige a geração de muitos conjuntos ou *pads* de chaves de ciframento combinadas. Cada *pad* consiste de um certo número de caracteres de chave aleatórios. Estes caracteres de chave devem ser escolhidos utilizando-se um processo verdadeiramente aleatório. Eles não podem ser gerados por nenhum tipo de gerador de chave criptográfica.

Cada parte envolvida em uma comunicação recebe conjuntos combinados de *pads*. Cada caracter de chave do *pad* é utilizado para cifrar um e somente um caracter do texto claro, não sendo nunca mais reutilizado. Qualquer violação destas condições anula a segurança perfeita provida pelo *one-time pad.*

Como o número de *pads* de chave aleatória gerados precisa ser pelo menos igual ao volume de mensagens em texto claro a ser cifrado, os *pads* de chave precisam ser substituídos depois de um certo período. O algoritmo torna-se inviável nos sistemas de comunicação modernos, em que a quantidade de informação que trafega é muito grande.

4.1.7. Gerência de Chaves Criptográficas

[Lucchesi 86] [Schneier 95] [Coulouris 94] [Bellovin 94]

Como já foi descrito no item 2.1.2, o uso de criptografia e ciframento implica lidar com chaves. Para tratar esse problema, deve-se dar atenção aos seguintes mecanismos:

- **Geração de Chaves**. São mecanismos pelos quais as chaves ou os pares de chaves são gerados, garantindo a imprevisibilidade e boas qualidades criptográficas.
- **Distribuição de Chaves.** Refere-se aos mecanismos pelos quais as chaves são segura e confiavelmente entregues às partes que as requisitam. O problema da distribuição de chaves tem relação com o tipo de algoritmo empregado para cifrar. Se for um algoritmo simétrico, então cada enlace distinto precisa de uma chave. Se *N* usuários quiserem se comunicar entre si, serão necessárias *N(N-1)/2* chaves. No entanto, se forem utilizados algoritmos assimétricos, serão necessárias somente *N* chaves. Essa é uma das razões de se utilizar algoritmos assimétricos para enviar as chaves de um algoritmo simétrico, que é quem vai efetivamente cifrar as mensagens. Assim, com um algoritmo assimétrico, é preciso haver somente uma chave por usuário, o que é uma quantidade bem menor que o número de enlaces possíveis.

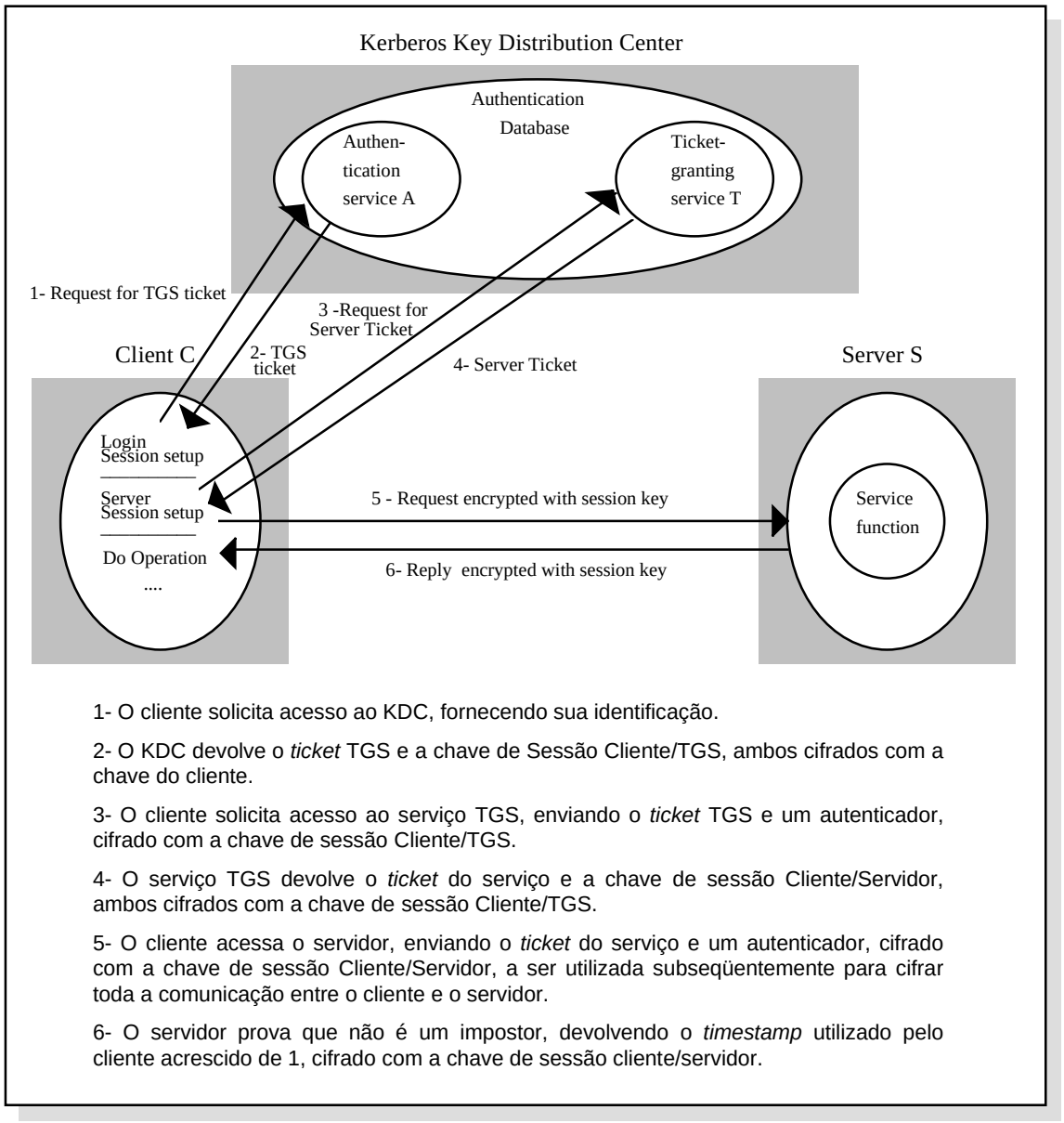
- **Armazenamento de Chaves**. Refere-se aos mecanismos pelos quais as chaves são armazenadas de forma segura e confiável para uso futuro pelas partes envolvidas.
- **Atualização de Chaves**. Refere-se aos mecanismos pelos quais as chaves são segura e confiavelmente atualizadas pelas partes.
- **Destruição de Chaves**. Refere-se aos mecanismos pelos quais se assegura a destruição de chaves antigas e não mais necessárias.
- **Arquivamento de Chaves**. Refere-se aos mecanismos pelos quais as chaves usadas nos "cartórios" (*notarization*) e nas atividades relacionadas ao não repúdio são arquivadas de maneira segura para posterior utilização. Por exemplo, como prova em litígios.
- **Key Distribution Center - KDC.** Os mecanismos acima relacionados são melhor administrados se for utilizado um Centro de Distribuição de Chaves ou KDC. Além disso, é bem mais fácil proteger chaves centralizadas do que chaves distribuídas. O KDC também pode funcionar como um "cartório", efetuando a certificação das chaves necessárias; o KDC normalmente exerce o papel de uma autoridade de certificação (CA) confiável.

  Existem várias implementações que utilizam essa técnica, como o sistema simétrico de autenticação *Kerberos*. O *Kerberos* foi desenvolvido no MIT para prover um alcance de autenticação e facilidades de segurança para uso no sistema *Athena* e em outros sistemas abertos. É um sistema de autenticação de rede, de chave secreta, sendo um cartório confiável que valida a identidade de indivíduos e servidores de rede.

  Baseia-se no modelo de distribuição de chaves desenvolvido por Needham e Schroeder. Permite a entidades, que estão se comunicando através de redes, provarem suas identidades mutuamente, enquanto previne ataques de escuta clandestina ou de *replay*.

  Foram adicionados *timestamps* no modelo de Needham e Schroeder para ajudar na detecção e prevenção de *replays*. O *Kerberos* também provê integridade ao fluxo de dados (quanto à detecção ou modificação) e sigilo (prevenindo leitura não autorizada), utilizando algoritmos criptográficos, tais como o DES.

  Ele opera sem impor nenhuma carga adicional ao usuário, como uma parte do *LOGIN* normal (Figura 4.5).



1- O cliente solicita acesso ao KDC, fornecendo sua identificação.

2- O KDC devolve o *ticket* TGS e a chave de Sessão Cliente/TGS, ambos cifrados com a chave do cliente.

3- O cliente solicita acesso ao serviço TGS, enviando o *ticket* TGS e um autenticador, cifrado com a chave de sessão Cliente/TGS.

4- O serviço TGS devolve o *ticket* do serviço e a chave de sessão Cliente/Servidor, ambos cifrados com a chave de sessão Cliente/TGS.

5- O cliente acessa o servidor, enviando o *ticket* do serviço e um autenticador, cifrado com a chave de sessão Cliente/Servidor, a ser utilizada subseqüentemente para cifrar toda a comunicação entre o cliente e o servidor.

6- O servidor prova que não é um impostor, devolvendo o *timestamp* utilizado pelo cliente acrescido de 1, cifrado com a chave de sessão cliente/servidor.

Figura 4.5 - Sistema de Autenticação *Kerberos* [Coulouris 94].

4.2. Algoritmos Criptográficos

[RFC 1321] [Menezes 93] [Stinson 95] [Schneier 95] [Lucchesi 86] [Cavalcanti 96] [Schneier 96] [Robshaw 97]

Como já foi descrito no item 2.1.2, o uso de criptografia e ciframento também implica lidar com algoritmos. Em geral, utiliza-se algoritmos que são públicos e conhecidos para cifrar o conteúdo das

mensagens - DES, IDEA, etc. - e chaves secretas. Assumindo-se que tais algoritmos são seguros, então as mensagens eletrônicas cifradas com eles são tão seguras quanto a chave. Como já exposto, há também diversos algoritmos de chave pública. Muitos deles destinados a gerenciamento de chaves. Outros destinados a fazerem assinaturas digitais, com a finalidade de prover autenticação às mensagens, em conjunto com as funções de espalhamento unidirecional. Serão discutidos nesta seção somente aqueles que provavelmente serão observados em uso, ou nas versões correntes ou nas versões futuras do programa PGP ou do padrão PEM.

4.2.1. Algoritmos de Ciframento

[Stinson 95] [Schneier 95] [Lucchesi 86] [Cavalcanti 96] [Schneier 96]

Neste item serão descritos três dos principais algoritmos de chave secreta existentes, utilizados para o ciframento de mensagens eletrônicas: o DES, o Triplo DES e o IDEA.

- **DES (*Data Encryption Standard*).** É o algoritmo de ciframento mais amplamente utilizado no mundo, ao menos para aplicações não militares. Foi desenvolvido pela IBM no início da década de 70, a partir de um algoritmo denominado *Lucifer*. Quando a NBS (*National Bureau of Standards*) solicitou algoritmos candidatos a padrão de ciframento federal em 1974, a IBM submeteu uma variação do *Lucifer*. A NSA ajudou a evoluir o algoritmo[36] e a IBM forneceu ao mundo uma licença de não exclusividade com isenção de pagamento de direitos autorais pela utilização do algoritmo.

  Adotado em 1976 como um padrão de ciframento nos EUA pelo NBS, atual NIST, o DES foi renomeado para DEA-1, um padrão de ciframento internacional. Desde então ele vem sendo aplicado em quase todo tipo de comunicação digital e armazenamento de dados. Vem sendo bastante analisado pela comunidade mundial de criptografia e, embora possua algumas fraquezas, ele ainda é o algoritmo eleito para muitas aplicações.

  O DES é chamado de um *cifrador de bloco iterativo*. *Cifrador* é apenas outra denominação para um algoritmo de ciframento. Um *cifrador de bloco* é um algoritmo que cifra dados em pedaços do tamanho de um bloco. O DES tem um tamanho de bloco de 8 bytes (64 bits). Assim, o algoritmo

[36] Por isto, muitos estudiosos de criptografia desconfiam da "mão invisível" da NSA no desenvolvimento do algoritmo DES. Durante o projeto do DES, a IBM desenvolveu algumas técnicas criptográficas que a NSA não quis que se tornassem públicas. Para muitos pesquisadores, as escolhas do projeto - ordens das operações, valores de determinadas constantes, empregos da chave - parecem arbitrárias e misteriosas.

aceita 8 bytes de texto claro e devolve 8 bytes de texto cifrado. Se a mensagem for mais longa do que 8 bytes - como é o caso da maioria das mensagens - o algoritmo cifra a mensagem 8 bytes por vez. Na outra ponta, o algoritmo decifra o texto cifrado em pedaços de 8 bytes.

Iterativo significa que o algoritmo é uma simples função repetida várias vezes. O DES possui 16 iterações. Como uma regra geral, mais iterações, ou *rounds*, proporcionariam maior segurança. Entretanto, o DES foi construído de tal forma que 16 iterações tornam-no tão seguro quanto possível; adicionar mais iterações ao DES praticamente não altera sua segurança.

O DES, portanto, cifra uma *string* de bits de texto claro *x*, de tamanho 64, empregando uma chave *K* que consiste em uma *string* de bits de tamanho 56, obtendo uma *string* de bits de texto cifrado novamente com tamanho igual a 64. De um modo geral, o algoritmo procede em três estágios [Stinson 95]:

1. Dado um texto claro *x*, a *string* de bits x_0 é gerada por uma permutação inicial fixa (*IP*) dos bits de *x*. Assim, $x_0 = IP\ (x\ ) = L_0\ R_0$, onde L_0 abrange os primeiros 32 bits de x_0 e R_0 os últimos 32 bits.

1. Efetua-se 16 iterações de uma determinada função *f* [37]. Calcula-se $L_i\ R_i$ $(1 \leq i \leq 16)$ com base na regra: $L_i = R_{i-1}$ e $R_i = L_{i-1} \oplus f\ (K_i\ , R_{i-1}\ )$, onde cada K_i consiste em uma *string* de bits de tamanho 48, calculada a partir de *K*. A figura, a seguir, descreve uma iteração de ciframento DES (Figura 4.6).

[37] A função *f* tem como entrada um argumento A, que é uma *string* de bits de tamanho 32, e um segundo argumento J, que é uma *string* de bits de tamanho 48, e produz como saída uma *string* de bits de tamanho 32. O argumento A sofre uma expansão E(A) para 48 bits, calcula-se $B=E(A) \oplus J$, escreve-se este resultado B como a concatenação de 8 *strings* de 6 bits cada $B=B_1B_2B_3\ B_4\ B_5\ B_6\ B_7\ B_8$. São utilizadas 8 vetores 4x16 fixos, ou seja, as *S-boxes* $S_1, \ldots , S_8$ para calcular $Cj = S_j\ (\ B_j\ )$, onde $1 \leq j \leq 8$ e Cj é escrito como uma *string* de bits de 32 bits. Assim, escreve-se $C = C_1C_2\ C_3\ C_4\ C_5\ C_6\ C_7\ C_8$ e permuta-se seus valores conforme uma permutação fixa P. A *string* de bits P(C) é definida como sendo a saída da função *f* (A , J).

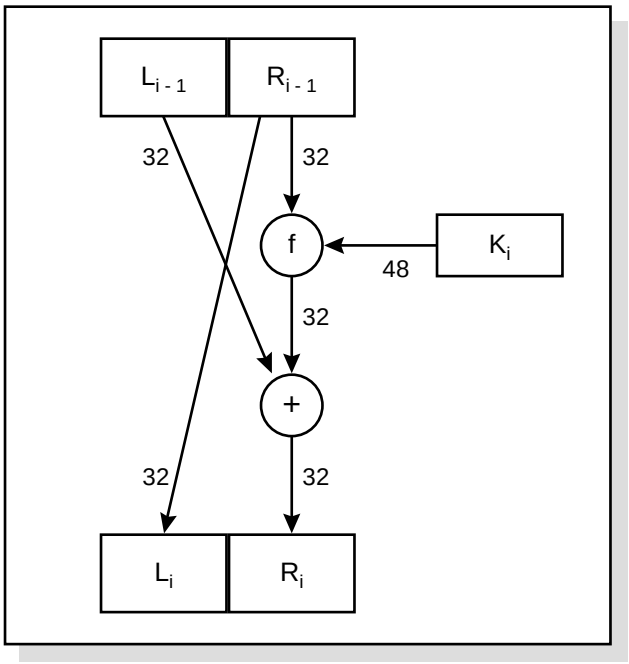


Figura 4.6 - Uma iteração do ciframento DES [Stinson 95].

3. Aplica-se a permutação inversa IP^{-1} à *string* de bits $R_{16}\,L_{16}$, obtendo-se o texto cifrado *y*. Isto é, $y = IP^{-1}(R_{16}\,L_{16})$.

Há vários ataques possíveis ao DES. Em 1979, W. Diffie e M. Hellman provaram que um computador paralelo com o propósito especial de quebra do DES poderia recuperar uma chave de um único par de texto claro/cifrado - isto é, 8 bytes de texto cifrado e 8 bytes de seu correspondente texto claro - em 24 horas. A máquina custaria entre 20 e 50 milhões de dólares para ser construída. Em 1993, uma demonstração similar feita por M. Wiener concluiu que o custo da máquina seria de um milhão de dólares, levando 3,5 horas para recuperar uma chave.

Avanços recentes da criptoanálise trouxeram ataques ao DES que são, na teoria, mais eficientes do que a força bruta mas, na prática, mais difíceis de serem utilizados. A *criptoanálise diferencial* [38] pode recuperar uma chave DES 512 vezes mais rápida do que a força bruta (conforme Tabela 2.1), mas exige 2^{55} blocos de texto claro conhecido ou 2^{47} blocos de texto claro escolhido para funcionar [39]. A

[38] Esta técnica foi inventada pelo menos 3 vezes durante o curso da história: pela NSA (embora eles não reconheçam); pela IBM, durante o desenvolvimento do Lucifer e do DES, no início dos anos 70; e finalmente por Eli Biham e Adi Shamir, nos anos 90, que a tornaram pública [Schneier 95].

[39] É uma mensagem tão longa que caberia em 1,8 milhões de discos de CD-ROM ou tão comprida que levaria 30.000 anos para ser transmitida sobre um *modem* de 9600 bps [Schneier 95] .

criptoanálise linear [40], de M. Matsui, da Mitsubishi do Japão, é mais eficiente do que a criptoanálise diferencial. Porém, este ataque necessita de 2^{43} bytes de texto claro e seu correspondente texto cifrado para trabalhar [41].

M. Hellman e outros pesquisadores estão tentando combinar as duas técnicas para criar um ataque mais poderoso. Enquanto isto, o *ataque da força bruta* continua sendo a maneira mais eficaz de se quebrar o DES. Ela requer apenas um bloco cifrado com seu correspondente texto claro; e o tempo, dinheiro e desejo de implementar a máquina de quebra. Afinal, a chave de 56 bits é realmente muito curta. Por outro lado, um milhão de dólares é um bocado de dinheiro para alguém gastar para ler um *e-mail* [Schneier 95].

O DES e, de fato, qualquer algoritmo de bloco, pode operar em um dos seguintes modos (Tabela 4.2):

[40] "Matsui descobriu relações lineares entre certas posições dos 64 bits de entrada e de saída para o DES, e através de experimentos com uma amostra relativamente pequena de dados de entrada, consegue-se recalcular a chave secreta. Mais recentemente, Matsui tem aplicado o mesmo tipo de análise para outras funções como o FEAL (*Fast Data Encipherment Algorithm*), criado no Japão" (extraído da palestra do prof. Routo Terada, de 03/nov/97, proferida na USP - SP).

[41] Ou seja, "apenas" 117.000 CD-ROM carregados de dados [Schneier 95].

Modo de Operação	*Descrição*
ECB *(Electronic Codebook)*	Este modo trata cada um dos blocos de 8 bytes de forma independente. Oito bytes de texto claro que entram equivalem a oito bytes de cifrado e vice-versa. Se uma mensagem tiver dois blocos de 8 bytes idênticos em determinada posição, o cifrado resultante também os terá na mesma posição, facilitando a criptoanálise e um ataque denominado *block-replay* (repetição de bloco).
CBC *(Cipher Block Chaining)*	Cada bloco cifrado y_i é somado por *XOR* (ou exclusivo) com o próximo bloco em claro x_{i+1} antes deste ser cifrado com a chave *K*. Este modo previne o ataque acima (o da repetição do bloco), fazendo com que os blocos de texto cifrado dependam uns dos outros. Se *Eve* tentar colocar um bloco no lugar de outro ou trocar a ordem de um par de blocos, será detectado. Dificulta a criptoanálise, mas requer a adição de uma variável de 8 bytes à chave: o vetor de inicialização (IV), que não precisa ser mantido em sigilo. Ou seja, $y_i = cifra\ (K, (y_{i-1} \oplus x_i))$, para y_0 = IV.
OFB *(Output Feedback)*	Gera-se uma *keystream* que é então somada por *XOR* com o texto claro. É um cifrador de *stream* síncrono, que produz a *keystream* pelo ciframento repetido do vetor IV: faz-se $z_0 = IV$ e calcula-se a *keystream* z_1z_2... pela regra: $z_i = cifra\ (K, (z_{i-1}))$. A seqüência de texto claro x_1x_2 é então cifrada pelo cálculo de $y_i = x_i \oplus z_i$.
CFB *(Cipher Feedback)*	Como o CBC, também faz com que os blocos de texto cifrado dependam uns dos outros, frustando tentativas de criptoanálise de *Eve*. O tipo de realimentação é diferente, mas os resultados são os mesmos. Também requer o vetor IV: faz-se $y_0 = IV$ e calcula-se a *keystream* z_i pelo ciframento do bloco cifrado *y* prévio. Isto é, $z_i = cifra\ (K, (y_{i-1}))$. Como no OFB, o texto claro x_1x_2 é obtido por $y_i = x_i \oplus z_i$.

Tabela 4.2 - Modos de Operação do DES.

Há dois pacotes de segurança de *e-mail* discutidos com mais detalhe neste trabalho: PGP e PEM. O PEM especifica o emprego do DES no modo CBC[42]. O PGP utiliza o IDEA no modo CFB.

- **Triplo DES.** O triplo DES é uma simples variação do DES: cifra a mensagem três vezes, uma seguida da outra. Se forem utilizadas diferentes chaves para cada ciframento, o resultado é um algoritmo muito mais seguro do que o DES.

  O PEM possui uma opção para o triplo DES. Há várias formas de implementá-lo, e o método de implementação tem considerável importância na sua segurança. Os projetistas do PEM prestaram atenção a este tópico, e implementaram o triplo DES corretamente. A versão utilizada no PEM é denominada modo EDE (*Encrypt - Decrypt - Encrypt*). Assim, no PEM, o triplo DES é chamado de DES-EDE. O DES-EDE cifra um bloco de texto claro com o DES e uma chave K_1, decifra o

[42] Modo *Cipher Block Chaining*, ou seja, ciframento de blocos com encadeamento.

resultado com o DES^{-1} e uma segunda chave K_2, e cifra novamente este último resultado com o DES e a primeira chave K_1 (Figura 4.7). O deciframento é o reverso: primeiro, decifra o texto cifrado com o DES^{-1} e a primeira chave K_1, então cifra esta saída com o DES e a segunda chave K_2 e, finalmente, decifra o último resultado de novo com o DES^{-1} e a primeira chave K_1.

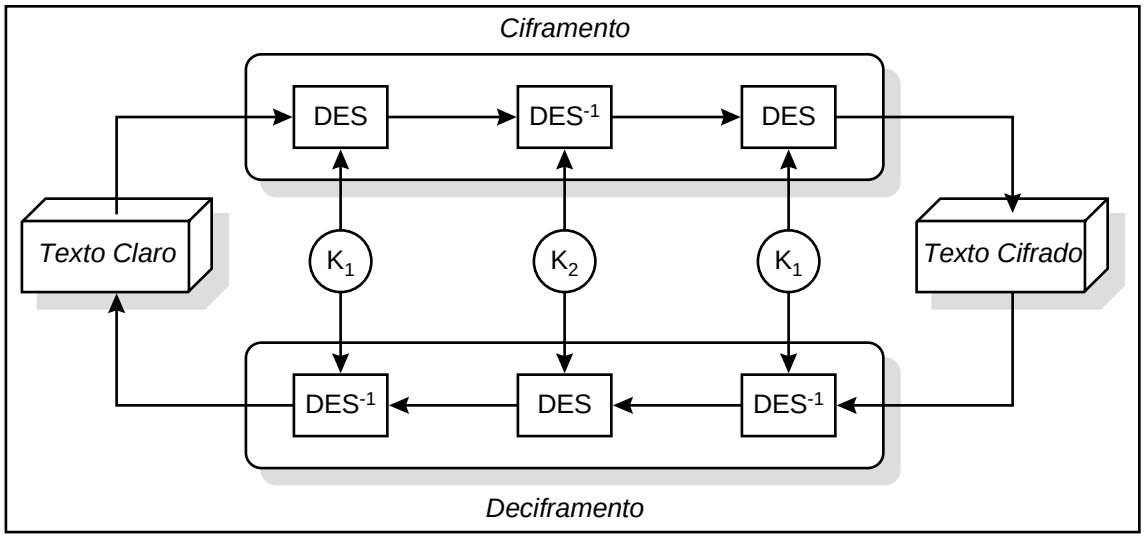


Figura 4.7 - Triplo DES.

O DES-EDE tem uma chave de 112 bits: duas chaves independentes de 56 bits cada. Ela é suficientemente longa para tornar um ataque de força bruta impraticável. O algoritmo DES-EDE é resistente à criptoanálise diferencial e linear. É seguro o suficiente contra ataques em um futuro próximo. Possui também metade da rapidez de um DES regular[43].

- **IDEA (*International Data Encryption Algorithm*).** Foi desenvolvido em 1991 por James Massey e Xuejia Lai, na Suíça. O algoritmo é estruturado seguindo as mesmas linhas gerais do DES. Ele também é um cifrador de bloco iterativo, com um bloco de tamanho igual a 64 bits e um tamanho de chave de 128 bits. Ele possui apenas 8 iterações, em comparação com o DES que possui 16, mas cada iteração IDEA funciona como se fosse uma dupla iteração DES. Na maioria dos microprocessadores, uma implementação por *software* do IDEA é mais rápida do que uma implementação por *software* do DES.

[43] Não possui um terço da rapidez do DES devido a alguns artifícios que os programadores utilizam para implementá-lo.

O IDEA opera sobre blocos de 64 bits: 64 bits de texto claro produzem 64 bits de texto cifrado. O mesmo algoritmo é utilizado para cifrar e para decifrar. A filosofia do projeto do algoritmo consiste na mistura de três grupos algébricos diferentes, todos facilmente implementáveis em *hardware* ou *software*:

XOR (ou exclusivo);

adição módulo 2^{16}; e

multiplicação módulo $2^{16} + 1$ (operação observada nas *s-boxes* do IDEA).

Todas estas operações são efetuadas em sub-blocos de 16 bits, o que colabora para torná-lo eficiente mesmo em processadores de 16 bits. Assim, o bloco em claro de 64 bits é dividido em 4 sub-blocos de 16 bits cada: X_1, X_2, X_3 e X_4 , que são a entrada da primeira iteração do algoritmo. Há um total de 8 iterações. Em cada uma delas, os 4 sub-blocos são somados por *XOR*, adicionados e multiplicados uns com os outros e com 6 sub-chaves de 16 bits cada. Entre as iterações, o segundo e o terceiro sub-bloco são trocados. Finalmente, os 4 sub-blocos são combinados com quatro sub-chaves em uma transformação de saída (Figura 4.8).

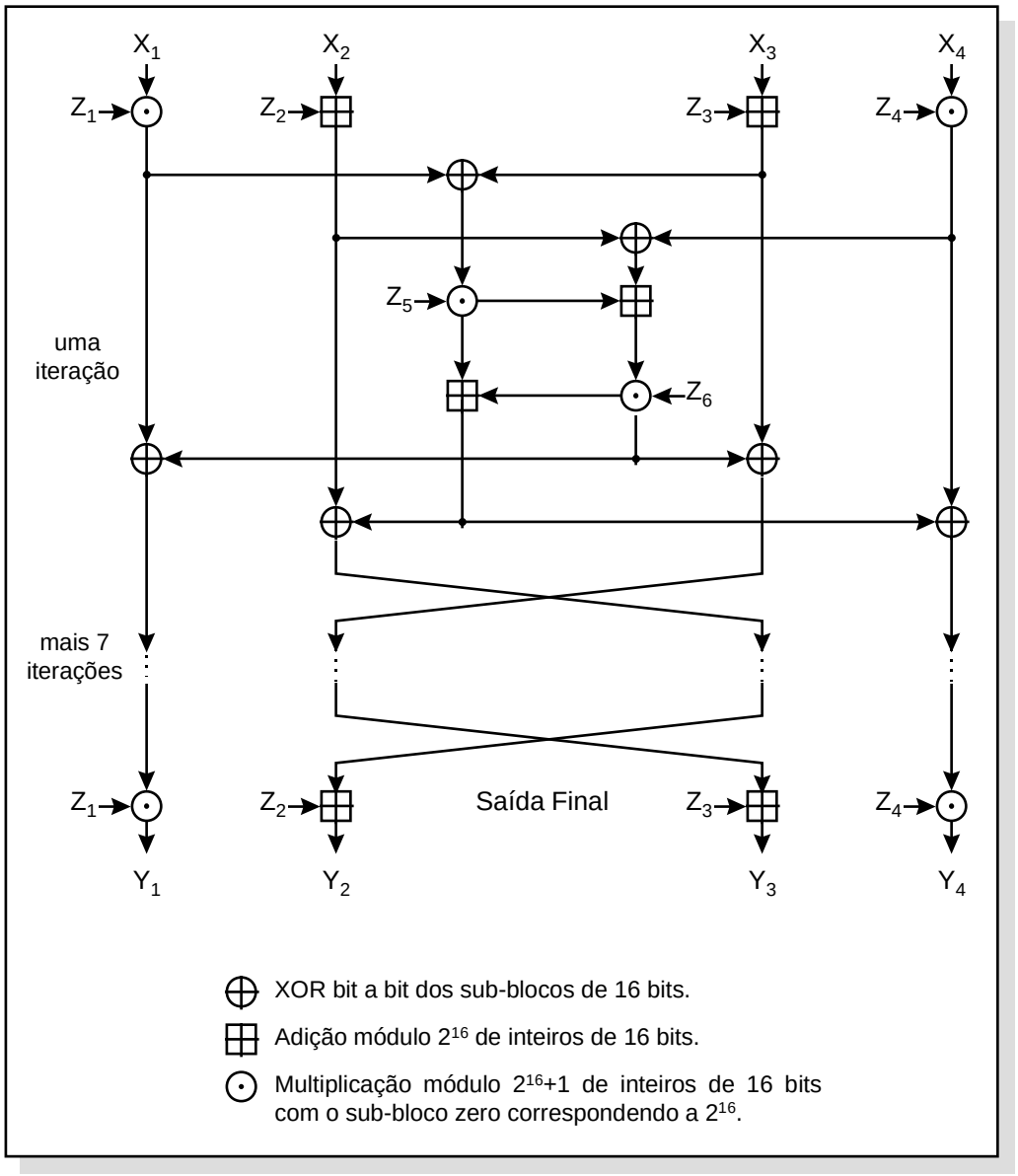


Figura 4.8 - IDEA [Schneier 96].

Em cada iteração, a seqüência de eventos é a existente na tabela a seguir (Tabela 4.3):

Passo	*Descrição*
1	Multiplica-se X_1 pela primeira sub-chave Z_1.
2	Adiciona-se X_2 à segunda sub-chave Z_2.
3	Adiciona-se X_3 à terceira sub-chave Z_3.
4	Multiplica-se X_4 pela quarta sub-chave Z_4.
5	Efetua-se um ou-exclusivo (XOR) com os resultados dos passos 1 e 3.
6	Efetua-se um ou-exclusivo (XOR) com os resultados dos passos 2 e 4.
7	Multiplica-se o resultado do passo 5 pela quinta sub-chave Z_5.
8	Adiciona-se os resultados dos passos 6 e 7.
9	Multiplica-se o resultado do passo 8 pela sexta sub-chave Z_6.
10	Adiciona-se os resultados dos passos 7 e 9.
11	Efetua-se um ou-exclusivo (XOR) com os resultados dos passos 1 e 9.
12	Efetua-se um ou-exclusivo (XOR) com os resultados dos passos 3 e 9.
13	Efetua-se um ou-exclusivo (XOR) com os resultados dos passos 2 e 10.
14	Efetua-se um ou-exclusivo (XOR) com os resultados dos passos 4 e10.

Tabela 4.3 - Passos para uma Iteração do IDEA.

A saída de cada iteração consiste nos 4 sub-blocos resultantes dos passos 11 a 14. Troca-se os dois sub-blocos internos (exceto para a última iteração); esta é a entrada para a próxima iteração. Após a oitava iteração, há uma transformação de saída final (Tabela 4.4).

Passo	*Descrição*
1	Multiplica-se X_1 pela primeira sub-chave Z_1.
2	Adiciona-se X_2 à segunda sub-chave Z_2.
3	Adiciona-se X_3 à terceira sub-chave Z_3.
4	Multiplica-se X_4 pela quarta sub-chave Z_4.

Tabela 4.4 - Passos para a Transformação Final do IDEA.

O IDEA é muito mais rápido do que o triplo DES. Tem uma chave de tamanho mais do que duas vezes maior do que a do DES; seu tamanho de chave é mesmo maior do que a utilizada no triplo DES. Assumindo que o ataque da força bruta seja o mais eficiente, seriam necessários 2^{128} (10^{38}) ciframentos para se recuperar uma chave.

Por outro lado, talvez a força bruta não seja o melhor ataque; e ele ainda é um algoritmo muito novo[44] para qualquer resultado de criptoanálise definitivo. Entretanto, há muito mais teoria matemática construída dentro do IDEA do que há no DES. O IDEA baseia-se na "mistura de operações a partir de grupos algébricos diferentes" [Schneier 95], o que provê à comunidade de criptografia alguma dimensão para confiar na segurança do algoritmo. Porém, amanhã sempre poderá haver uma descoberta que possa quebrá-lo. Isto vale também para o DES. O PGP (a partir da versão 2.0) utiliza o IDEA no modo CFB.

4.2.2. Algoritmos de Gerenciamento de Chaves

[Menezes 93][Stinson 95] [Schneier 95] [Lucchesi 86] [Cavalcanti 96] [Schneier 96] [Robshaw 97]

Neste item, serão descritos dois dos principais algoritmos de chave pública existentes, utilizados para o gerenciamento de chaves em sistemas de segurança de *e-mail*: o RSA e o ElGamal. O sistema criptográfico de curvas elípticas de Menezes-Vanstone, embora não esteja especificado nos pacotes de segurança de correio eletrônico citados neste trabalho, também será apresentado a título de sugestão para futura implementação.

- **RSA.** O RSA é um algoritmo assimétrico que possui este nome devido a seus inventores: Ron Rivest, Adi Shamir e Len Adleman. É, atualmente, o algoritmo de chave pública mais amplamente utilizado, além de ser uma das mais poderosas formas de criptografia de chave pública conhecidas até o momento. O RSA utiliza números primos[45]. A premissa por trás do RSA é que é fácil multiplicar dois números primos para obter um terceiro número, mas muito difícil recuperar os dois primos a partir daquele terceiro número. Isto é conhecido como *fatoração*. Por exemplo, os fatores primos de 3.337 são 47 e 71.

[44] Foram necessários 15 anos de estudo do DES para a invenção da criptoanálise diferencial, algo que a NSA conhecia o tempo todo. Quem pode afirmar algo sobre o que a NSA já descobriu a respeito do IDEA? [Schneier 95]

[45] Ou seja, números que somente são divisíveis por 1 e por eles mesmos.

Gerar a chave pública envolve multiplicar dois primos grandes; qualquer um pode fazer isto. Derivar a chave privada a partir da chave pública envolve fatorar um grande número. Se o número for grande o suficiente e bem escolhido, então ninguém pode fazer isto em uma quantidade de tempo razoável. "Assim, a segurança do RSA baseia-se na dificuldade de fatoração de números grandes. Pelo menos todos acreditam que a única maneira de se quebrar o RSA é através da fatoração de números grandes. Mas é sempre possível que alguém possa descobrir uma forma de quebrá-lo sem fazer isto. Deste modo, a fatoração representa um limite superior do tempo necessário para quebrar o algoritmo" [Schneier 95].

Para se gerar as duas chaves, basta escolher dois grandes (de 100 a 200 ou mais dígitos) números primos aleatórios p e q. Para uma segurança máxima, é bom escolher p e q de igual tamanho. Calcula-se então o produto:

$$n = pq$$

Então escolhe-se aleatoriamente a chave de ciframento e, de maneira tal que e seja primo relativo [46] de $(p - 1)(q - 1)$. Finalmente, utiliza-se o algoritmo de Euclides estendido para se calcular a chave de deciframento d, tal que:

$$e\,d \equiv 1 \pmod{((p - 1)(q - 1))}$$

Ou seja,

$$d \equiv e^{-1} \pmod{((p - 1)(q - 1))}$$

O número n e o número e são a chave pública; o número d é a chave privada. Os dois primos p e q não são mais necessários. Eles podem ser agora descartados, porém nunca revelados.

Para cifrar uma mensagem m, primeiro deve-se dividí-la em blocos numéricos menores do que n (com dados binários, deve-se escolher a maior potência de 2 menor do que n). Isto é, se ambos p e q forem primos de 100 dígitos, então n terá um comprimento abaixo de 200 dígitos e, por conseguinte, cada bloco de mensagem m_i também. A mensagem cifrada c será composta de blocos de mensagem c_i de tamanho similar. A fórmula de ciframento é simples:

$$c_i = m_i^{\,e} \bmod n$$

[46] Ser primo relativo significa não possuir fatores em comum.

Para decifrar uma mensagem, então deve-se pegar cada bloco cifrado c_i e calcular:

$$m_i = c_i^{\,d} \bmod n$$

Desde que:

$$c_i^{\,d} = (m_i^{\,e})^{\,d} = m_i^{\,e\,d} = m_i^{\,k\,(p-1)(q-1)+1} = m_i * m_i^{\,k\,(p-1)(q-1)} = m_i * 1 = m_i ,$$

considerando tudo (mod n), esta fórmula recupera a mensagem.

A mensagem também pode ser cifrada com d e decifrada com e; o RSA é, portanto, um sistema comutativo. Ele também pode ser utilizado para assinatura digital, como descrito no item 4.2.3. A tabela, a seguir, resume o ciframento do RSA (Tabela 4.5).

Valor	***Descrição***
Chave Pública *n, e*	O valor n é o produto de dois primos p e q (p e q devem permanecer secretos). O valor e é relativamente primo a $(p - 1)(q - 1)$.
Chave Privada *d*	O valor d é igual a $e^{-1} \bmod ((p - 1)(q - 1))$.
Ciframento *c*	O valor cifrado c é igual a $m^e \bmod n$.
Deciframento *m*	O valor da mensagem m é igual a $c^d \bmod n$.

Tabela 4.5 - O Ciframento do RSA.

- **ElGamal.** O ElGamal é outro algoritmo de chave pública utilizado para gerenciamento de chaves. Sua matemática difere da utilizada no RSA, mas também é um sistema comutativo. O algoritmo envolve a manipulação matemática de grandes quantidades numéricas. Sua segurança advém de algo denominado *problema do logaritmo discreto*[47], que lembra bastante o problema da fatoração.

  De fato, "qualquer avanço no problema do logaritmo discreto geralmente pode ser aplicado ao problema da fatoração" [Schneier 95]. E não há garantias de não existir uma maneira melhor de quebrar o ElGamal sem ser por logaritmos discretos, embora todos suspeitem de que realmente não haja melhor forma.

[47] O ElGamal obtém sua segurança da dificuldade de se calcular logaritmos discretos em um corpo finito.

Atualmente, nenhum dos principais programas de segurança de correio eletrônico utiliza o ElGamal[48], mas existe a possibilidade dele ser adicionado a uma versão do PEM. Ele não é nem mais nem menos seguro do que o RSA: baseia-se na mesma teoria matemática: a *teoria dos corpos finitos*.

Em um sistema de segurança de *e-mail*, a única razão de se optar por um ao invés do outro é para evitar violações de patente. O algoritmo ElGamal não é patenteado. Mas foi coberto até 29 de abril de 1997 pela patente de Diffie - Hellman, obviamente já expirada. Como o RSA, o ElGamal também pode ser utilizado para assinaturas digitais, como será apresentado no item 4.2.3.

Para se gerar um par de chaves no ElGamal, primeiro deve-se escolher um valor primo p e dois valores aleatórios g e x, tais que ambos sejam menores do que p. Então, calcula-se:

$$y = g^x \bmod p$$

A chave pública é composta por y, g e p. Ambos g e p podem ser compartilhados entre um grupo de usuários. A chave privada é x. Para *cifrar* uma mensagem M, primeiro deve-se escolher um valor k aleatório, tal que k seja primo relativo de p - 1. Então, calcula-se:

$$a = g^k \bmod p$$

$$b = y^k M \bmod p$$

O par a e b forma o texto cifrado. Observe que o texto cifrado é duas vezes o tamanho do texto claro, o que implica um fator de expansão da mensagem igual a 2. Para *decifrar* a e b, calcula-se:

$$M = b / a^x \bmod p$$

Desde que:

$$a^x \equiv g^{kx} \bmod p, \text{ e } b/a^x \equiv y^k M/a^x \equiv g^{xk} M / g^{xk} \equiv M \pmod p,$$

tudo isto funciona bem. A tabela, a seguir, resume o ciframento do ElGamal (Tabela 4.6).

[48] O ElGamal também pode ser implementado em uma curva elíptica, porém há uma dificuldade prática em tal implementação: o fator de expansão da mensagem clara passa de dois (quando implementado em Z_p) para cerca de quatro.

Valor	*Descrição*
Chave Pública ***p, g, y***	O valor p é um primo que pode ser compartilhado entre um grupo de usuários. O valor aleatório $g < p$ também pode. O valor y é igual a $g^x \bmod p$.
Chave Privada *x*	O valor aleatório x também precisa ser $< p$.
Ciframento ***k, a, b***	O valor k é escolhido aleatoriamente, sendo primo relativo de p - 1. O texto cifrado a é igual a $g^k \bmod p$. Já o texto cifrado b (o par a e b forma o texto cifrado) é igual a $y^k M \bmod p$.
Decifamento *m*	O valor da mensagem M (texto claro) é igual a $b / a^x \bmod p$.

Tabela 4.6 - O Ciframento do ElGamal.

- **Curvas Elípticas.** Em 1985, Neal Koblitz e V. S. Miller propuseram de forma independente a utilização de curvas elípticas para sistemas criptográficos de chave pública. Eles não chegaram a inventar um novo algoritmo criptográfico com curvas elípticas sobre corpos finitos, mas implementaram algoritmos de chave pública já existentes[49] usando curvas elípticas.

  Assim, os sistemas criptográficos de curvas elípticas consistem em modificações de outros sistemas (o ElGamal, por exemplo), que passam a trabalhar no domínio das curvas elípticas, em vez de trabalharem no domínio dos corpos finitos. Eles possuem o potencial de proverem sistemas criptográficos de chave pública mais seguros[50] com chaves de menor tamanho.

  "Estima-se que um sistema criptográfico de curvas elípticas, implementado sobre o corpo finito de 160 bits GF (2^{160}), tenha aproximadamente a mesma resistência a um ataque que o RSA de 1024 bits" [Robshaw 97].

  Muitos algoritmos de chave pública, como o Diffie - Hellman, o ElGamal e o Schnorr podem ser implementados em curvas elípticas sobre corpos finitos. Assim, fica resolvido um dos maiores problemas dos algoritmos de chave pública: o grande tamanho de suas chaves.

[49] Como o algoritmo de Diffie e Hellman.

[50] As curvas elípticas proporcionam uma forma de construir "elementos" e "regras de combinação" que produzem grupos. Estes grupos têm propriedades familiares o suficiente para a construção de algoritmos criptográficos, mas não possuem certas propriedades que podem facilitar a criptoanálise.

Porém, os algoritmos de curvas elípticas atuais, embora possuam o potencial de serem rápidos, são em geral mais demorados do que o RSA. Além disso, a implementação do ElGamal em uma curva elíptica, por exemplo, possui a dificuldade prática de dobrar o fator de expansão de mensagem. Quando implementado em Z_p (p primo > 3), a mensagem cifrada possui o dobro do tamanho original do texto claro. Quando implementado em uma curva elíptica, ela possui o quádruplo do tamanho mencionado.

Para resolver este problema, Menezes e Vanstone encontraram uma variação mais eficiente. Nela, a curva elíptica é utilizada como "máscara", e permite-se que os textos claros e cifrados sejam pares ordenados arbitrários de elementos não nulos do corpo (isto é, não se exige que sejam pontos sobre a curva elíptica). Isto produz um fator de expansão de mensagem igual a 2, ao invés de 4. Ou seja, o mesmo fator do sistema criptográfico ElGamal original (Figura 4.9).

Seja E uma curva elíptica definida sobre Z_p (p primo > 3), tal que E contenha um subgrupo cíclico H em que o problema do logaritmo discreto seja intratável.

Faça $P = Z_p^* \times Z_p^*$, $C = E \times Z_p^* \times Z_p^*$, e defina:

$$K = \{(E, \alpha, a, \beta) : \beta = a\,\alpha \},$$

onde $a \in E$. Os valores α e β são públicos, e o valor a é secreto.

Para $K = (E, \alpha, a, \beta)$, para um número aleatório (secreto) $k \in Z_{|H|}$, e para $x = (x_1 , x_2) \in Z_p^* \times Z_p^*$, defina

$$e_k (x, k) = (y_0 , y_1 , y_2),$$

onde

$$y_0 = k\,\alpha ,$$
$$(c_1 , c_2) = k\,\beta ,$$
$$y_1 = c_1\ x_1 \bmod p , \quad \text{e}$$
$$y_2 = c_2\ x_2 \bmod p .$$

Para um texto cifrado $y = (y_0 , y_1 , y_2)$, defina

$$d_k (y) = (y_1\ c_1^{-1} \bmod p , y_2\ c_2^{-1} \bmod p),$$

onde

$$a\ y_0 = (c_1 , c_2).$$

Figura 4.9 - O Sistema de Curvas Elípticas de Menezes-Vanstone [Stinson 95].

Os sistemas de curvas elípticas também possuem a vantagem de não estarem patenteados por completo: na verdade são patenteados por família de curvas. Assim, uma dada curva elíptica pode estar patenteada, mas uma outra pode não estar por pertencer a uma família distinta.

Diante do exposto até aqui, é possível concluir que os sistemas criptográficos de curvas elípticas são bastante interessantes e podem ser implementados com vantagem nos pacotes de segurança de correio eletrônico. A matemática por trás dos sistemas de curvas elípticas é complexa e foge ao escopo deste trabalho, mas pode-se encontrar maiores detalhes no excelente livro de Alfred Menezes [Menezes 93].

4.2.3. Algoritmos de Autenticação

[RFC 1321] [Stinson 95] [Schneier 95] [Lucchesi 86] [Cavalcanti 96] [Schneier 96]

Os algoritmos de chave pública não se destinam somente a gerenciamento de chaves. Neste item, serão descritos os algoritmos de chave pública destinados a fazer assinaturas digitais, com a finalidade de prover autenticação às mensagens nos sistemas de segurança de *e-mail* estudados. Como normalmente tais algoritmos são utilizados em conjunto com funções de espalhamento unidirecional, também serão descritas algumas funções deste tipo.

- **RSA.** Como já mencionado, o RSA também é comutativo e pode ser utilizado para a geração de assinatura digital. A matemática é a mesma: há uma chave pública e uma chave privada, e a segurança do sistema baseia-se na dificuldade da fatoração de números grandes. O PGP e o PEM utilizam o RSA para assinatura digital. Basta inverter a ordem de utilização das chaves apresentada no ciframento.

  Utilizando-se a chave RSA privada de *Alice* para ciframento, obtém-se um texto assinado que somente pode ter sido cifrado por uma única pessoa: *Alice.* E a assinatura de *Alice* no texto é facilmente conferida por *Bob* ou qualquer outra pessoa (inclusive *Eve*), pelo uso da chave pública de *Alice.* Afinal, todos possuem acesso a esta informação. A tabela, a seguir, descreve o esquema da assinatura digital do RSA (Tabela 4.7).

Valor	*Descrição*
Chave Pública ***n, e***	O valor n é o produto de dois primos p e q (p e q devem permanecer secretos). O valor e é relativamente primo a $(p - 1)(q - 1)$.
Chave Privada *d*	O valor d é igual a $e^{-1} \bmod ((p - 1)(q - 1))$.
Assinatura *y*	O valor y, que corresponde ao valor x assinado, é calculado assim: y = assina (x, chave privada) = $x^d \bmod n$.
Verificação	A verificação da assinatura é feita como a seguir: verifica (x, y, chave pública) = verdadeira $\Leftrightarrow x \equiv y^e \bmod n$.

Tabela 4.7 - O Esquema de Assinatura do RSA.

- **ElGamal.** Como o RSA, o ElGamal também é comutativo, podendo ser utilizado tanto para assinatura digital quanto para gerenciamento de chaves; assim, ele obtém sua segurança da dificuldade do cálculo de logaritmos discretos em um corpo finito. A geração das chaves é idêntica à descrita no item 4.2.2: primeiro deve-se escolher um valor primo p e dois valores aleatórios g e x, tais que ambos sejam menores do que p. Então, calcula-se:

$$y = g^x \bmod p$$

Apenas para lembrar: a chave pública do ElGamal é composta por y, g e p; ambos g e p podem ser compartilhados entre um grupo de usuários; e a chave privada é x. Para *assinar* uma mensagem M, primeiro deve-se escolher um número aleatório k, tal que k seja primo relativo de $p - 1$. Então, calcula-se:

$$a = g^k \bmod p$$

e utiliza-se o algoritmo de Euclides estendido para obter o valor de b na seguinte equação:

$$M = (x\, a + k\, b) \bmod (p - 1)$$

O par a e b forma a assinatura. O valor aleatório k deve permanecer secreto.

Para *verificar uma assinatura*, deve-se confirmar que:

$$y^a * a^b \bmod p = g^M \bmod p$$

Cada assinatura ElGamal exige um novo valor aleatório de *k*. Se um intruso *Eve* recuperar um valor *k* utilizado por *Alice*, ou se conseguir duas mensagens assinadas ou cifradas com o mesmo valor *k* (mesmo sem saber o valor de *k*, neste caso), ele poderá recuperar a chave privada de *Alice*, ou seja, o valor *x*.

Atualmente, nenhum programa de segurança de *e-mail* utiliza o ElGamal para assinatura, mas existe a possibilidade dele vir a ser utilizado em versões futuras do PEM. A tabela, a seguir, resume o esquema de assinatura do ElGamal (Tabela 4.8).

Valor	***Descrição***
Chave Pública ***p, g, y***	O valor p é um primo que pode ser compartilhado entre um grupo de usuários). O valor aleatório $g < p$ também pode. O valor y é igual a $g^x \bmod p$.
Chave Privada *x*	O valor aleatório x também precisa ser $< p$.
Assinatura ***k, a, b***	O valor k é escolhido aleatoriamente, sendo primo relativo de $p - 1$. O valor a (assinatura) é igual a $g^k \bmod p$. Já o valor b (o par a e b forma a assinatura) é tal que $M = (x\,a + k\,b) \bmod (p - 1)$.
Verificação	Aceita-se como válida se $y^a * a^b \bmod p = g^M \bmod p$.

Tabela 4.8 - O Esquema de Assinatura do ElGamal.

- **DSA.** O *Digital Signature Algorithm* foi proposto pelo NIST, em agosto de 1991, para utilização no seu padrão DSS (*Digital Signature Standard*). Adotado como padrão final em dezembro de 1994, trata-se de uma variação dos algoritmos de assinatura ElGamal e Schnorr. Foi inventado pela NSA e patenteado pelo governo americano.

  A empresa RSA Data Security lançou uma campanha contra o algoritmo, alegando haver nele um possível *trapdoor* que permitiria ao governo dos EUA quebrar o DSA; e queixou-se de sua velocidade, do fato dele não poder ser utilizado para ciframento e do tamanho de sua chave.

  A única questão genuína de segurança levantada foi sobre o tamanho da chave. Quando o padrão foi inicialmente proposto, previa uma chave de 512 bits - muito pequena. O padrão final permite o uso de chave até 1024 bits, um tamanho bem mais seguro.

O DSA obtém sua segurança a partir do problema do logaritmo discreto, similar ao ElGamal. Assim, não há muita vantagem em termos de segurança na utilização de um algoritmo sobre o outro. Atualmente, nenhum programa de segurança de *e-mail* utiliza o DSA para assinatura, mas existe a possibilidade dele vir a ser utilizado, como ocorre com o ElGamal, em uma versão do PEM, em substituição ao algoritmo de assinatura RSA.

- **MD5.** É uma função de espalhamento unidirecional inventada por Ron Rivest, do MIT, que também trabalha para a RSA Data Security. A sigla "MD" significa *Message Digest*. Este algoritmo produz um valor *hash* de 128 bits para uma mensagem de entrada de tamanho arbitrário.

  Foi inicialmente proposto em 1991, após alguns ataques de criptoanálise terem sidos descobertos contra a função de espalhamento unidirecional prévia de Rivest: a MD4. O algoritmo foi projetado para ser rápido, simples e seguro. Seus detalhes são públicos, e têm sido analisados pela comunidade de criptografia.

  Foi descoberta uma fraqueza em parte do MD5, mas até agora ela não afetou a segurança global do algoritmo. Entretanto, o fato dele produzir uma valor *hash* de somente 128 bits é o que causa maior preocupação; é preferível uma função de espalhamento unidirecional que produza um valor maior. Ambos PGP e PEM utilizam o MD5 como função de espalhamento unidirecional. Segue uma descrição da função MD5.

  Após iniciada, a função MD5 processa o texto de entrada em blocos de 512 bits cada, divididos em 16 sub-blocos de 32 bits. A saída do algoritmo é um conjunto de quatro blocos de 32 bits, que concatenados formam um único valor *hash* de 128 bits.

  Primeiro, a mensagem é estendida até que seu tamanho fique congruente a 448, módulo 512 (ou seja, seu tamanho fique exatamente 64 bits menor do que um múltiplo de 512). A expansão é efetuada da seguinte maneira: um bit 1 é acrescentado no final da mensagem, seguido de tantos bits zeros quantos necessários. Depois, uma representação de 64 bits do tamanho original da mensagem (antes dos bits de expansão serem adicionados) é anexada ao resultado. Estes dois passos servem para tornar o tamanho da mensagem um múltiplo exato de 512 bits, exigido para o restante do algoritmo[51].

[51] Além de assegurarem que mensagens diferentes não fiquem muito parecidas após a expansão.

Então, são iniciadas quatro variáveis *A*, *B*, *C* e *D*, de 32 bits cada, com quatro valores hexadecimais fixos:

A = 01234567;

B = 89abcdef;

C = fedcba98; e

D = 76543210.

O *loop* principal possui quatro iterações, todas bastante parecidas. Cada iteração utiliza uma mesma operação 16 vezes. Cada operação utiliza uma função não linear em três das quatro variáveis auxiliares (Figura 4.10).

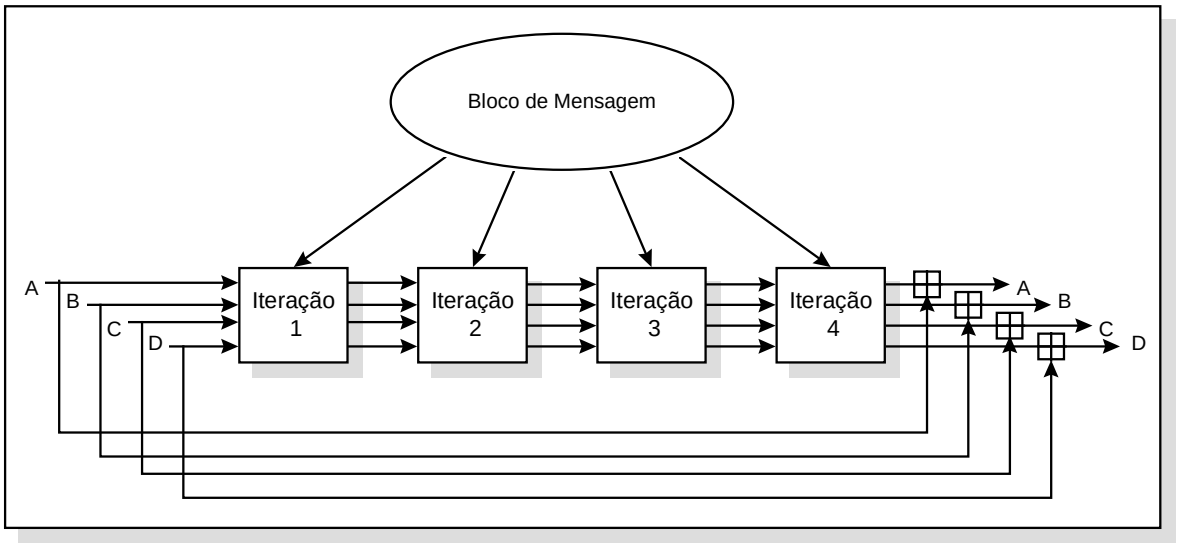


Figura 4.10 - *Loop* Principal do MD5 [Schneier 96].

- **SHA.** O *Secure Hash Algorithm*, uma função de espalhamento unidirecional inventada pela NSA, gera um valor *hash* de 160 bits, a partir de um tamanho arbitrário de mensagem. O funcionamento interno do SHA é muito parecido com o observado no MD4, indicando que os estudiosos da NSA basearam-se no MD4 e fizeram melhorias em sua segurança.

  De fato, a fraqueza existente em parte do MD5, citada anteriormente, descoberta após o SHA ter sido proposto, não ocorre no SHA. Atualmente, não há nenhum ataque de criptoanálise conhecido contra o SHA. Mesmo o ataque da força bruta torna-se impraticável, devido ao seu valor *hash* de

160 bits. Porém, não há provas de que, no futuro, alguém não possa descobrir como quebrar o SHA.

Nem o PGP nem o PEM atualmente fazem uso do SHA, mas é esperado que ambos ofereçam-no, ao menos como uma opção, em breve.

- **MD2 e MD4.** O MD4 é o precursor do MD5, tendo sido inventado por Ron Rivest. Após terem sido descobertas algumas fraquezas no MD4, Rivest escreveu o MD5. O MD4 fez parte das especificações originais do PEM, mas não é mais utilizado.

  O MD2 é uma função de espalhamento unidirecional simplificada, e produz um *hash* de 128 bits. A segurança do MD2 é dependente de uma permutação aleatória de bytes. Esta permutação é fixa e dependente dos dígitos de π. "É uma opção no PEM; não é recomendável sua utilização, pois, em geral, é mais lento do que as outras funções *hash* citadas e acredita-se que seja menos seguro" [Schneier 95].

4.3. Cuidados Para Manter A Chave Protegida

[Schneier 95] [Schneier 96]

A força de um programa de segurança de *e-mail* é totalmente dependente do sigilo da chave privada. A chave privada identifica uma pessoa; é a parte de informação que somente uma pessoa deve conhecer.

Se esta pessoa (*Alice*) revelar sua chave privada a alguém (*Eve*), este alguém pode se fazer passar por ela, além de poder ler toda sua correspondência eletrônica pessoal. Assim, a chave privada não deve ser revelada a ninguém. Não importa o quanto isto facilite as coisas ou quanto se confia em outra pessoa. Não se trata, aqui, de uma questão de confiança; é uma questão de identidade. Fornecer a alguém a chave privada equivale a fornecer sua própria identidade. Isto não deve ser feito; e também não se deve facilitar o furto de sua chave privada.

Como ela geralmente é um número muito grande, o programa de segurança armazena-a cifrada em um disco. O que é preciso memorizar é uma *password* ou *passphrase* [52] que desbloqueia a chave privada. Deste modo, deve-se tomar alguns cuidados básicos:

- Não se deve escrever a *passphrase* em lugar algum, muito menos na agenda pessoal - local mais visado por alguém que deseja furtá-la.
- Não se deve armazená-la em um arquivo de computador.
- Deve-se mantê-la na memória, tornando-a parte de si próprio.
- Deve-se escolher uma *passphrase* que não seja fácil de adivinhar. Para isto, não se deve utilizar:

  uma *passphrase* que seja formada pelo próprio nome, iniciais ou nome de alguém próximo (inclusive animais de estimação);

  datas importantes, tais como aniversários;

  números associados à pessoa, tais como número da identidade ou CPF;

  o nome da rua ou do bairro onde se mora, o nome do modelo do carro que se dirige ou qualquer outra palavra que possa ser facilmente ligada à própria pessoa;

  palavras que sejam facilmente visíveis quando se está sentado no computador (tais como o modelo da máquina ou a legenda do quadro na parede em frente ao computador); e

  palavras muito curtas; elas devem ser tão longas quanto possíveis e deve-se sempre evitar palavras existentes em qualquer dicionário, procurando-se misturar letras, números e caracteres especiais, tais como marcas de pontuação.

[52] Uma *passphrase* é parecida com uma *password* ou senha; porém, ela pode ser uma frase inteira ou uma sentença, e não apenas uma única palavra.

5. Descrição do PGP

O *Pretty Good Privacy* (PGP) é um programa de segurança de correio eletrônico originariamente projetado por Philip Zimmermann. Ele emprega: o IDEA, para ciframento de dados; o RSA (com chaves de tamanho até 2047 bits), para gerenciamento de chaves e assinaturas digitais; e o MD5, como função de espalhamento unidirecional.

Encontra-se disponível na Internet, em versões gratuitas ou pagas, operando em uma grande variedade de plataformas de *hardware* e de *software*. Conforme já discutido no item 2.1.5, desde que foi introduzido, em 1991, ele foi amplamente disseminado em todo o mundo e é, atualmente, um dos mais populares programas de segurança de *e-mail*.

O PGP provê uma série de serviços de segurança para correio eletrônico, todos já discutidos no item 2.1.4:

- Confidencialidade.
- Autenticidade da Origem dos Dados.

- Integridade de Mensagem.
- Não Repúdio da Origem.

Embora o PGP tenha sido projetado para prover automaticamente confidencialidade e autenticidade, é possível enviar uma mensagem protegida pelo PGP sem prover confidencialidade. Também é possível o envio de uma mensagem protegida pelo PGP sem prover autenticidade ou integridade.

As estruturas de dados em uma mensagem PGP encontram-se todas em binário. Portanto, são bastante eficientes. Significa também que, ao se olhar para uma mensagem PGP, é bem difícil obter algum detalhe interessante. Um arquivo PGP é uma concatenação de um ou mais pacotes[53] binários que podem ser transformados por ciframento, compressão ou codificação.

O exterior do arquivo da mensagem PGP permite visualizar apenas:

- o resultado *K(a)* do ciframento da mensagem assinada *a* feito pela chave de sessão aleatória *K*;
- o resultado do ciframento $P_B\ (K)$ da chave de sessão *K*, utilizando-se a chave pública P_B de *Bob*; e
- uma identificação $id(P_B)$ da chave pública P_B de *Bob*, utilizada no ciframento da chave *K*.

Somente após o deciframento da mensagem seus pacotes internos poderão ser lidos; então, poderá ser extraída a chave de ciframento aleatória utilizada, ou seja, a chave de sessão (Figura 5.1).

A seção 5.1 deste capítulo descreve o ambiente PGP. A seção 5.2 descreve os passos necessários para o envio de uma mensagem PGP, enquanto que a seção 5.3 descreve os passos necessários para o seu recebimento. A seção 5.4 discute a certificação e a distribuição das chaves PGP. As principais funções do PGP são acionadas pelo programa "ProtegeMail", descrito na Parte III deste trabalho.

[53] Um pacote, no caso, consiste em um envelope digital com dados em seu interior.

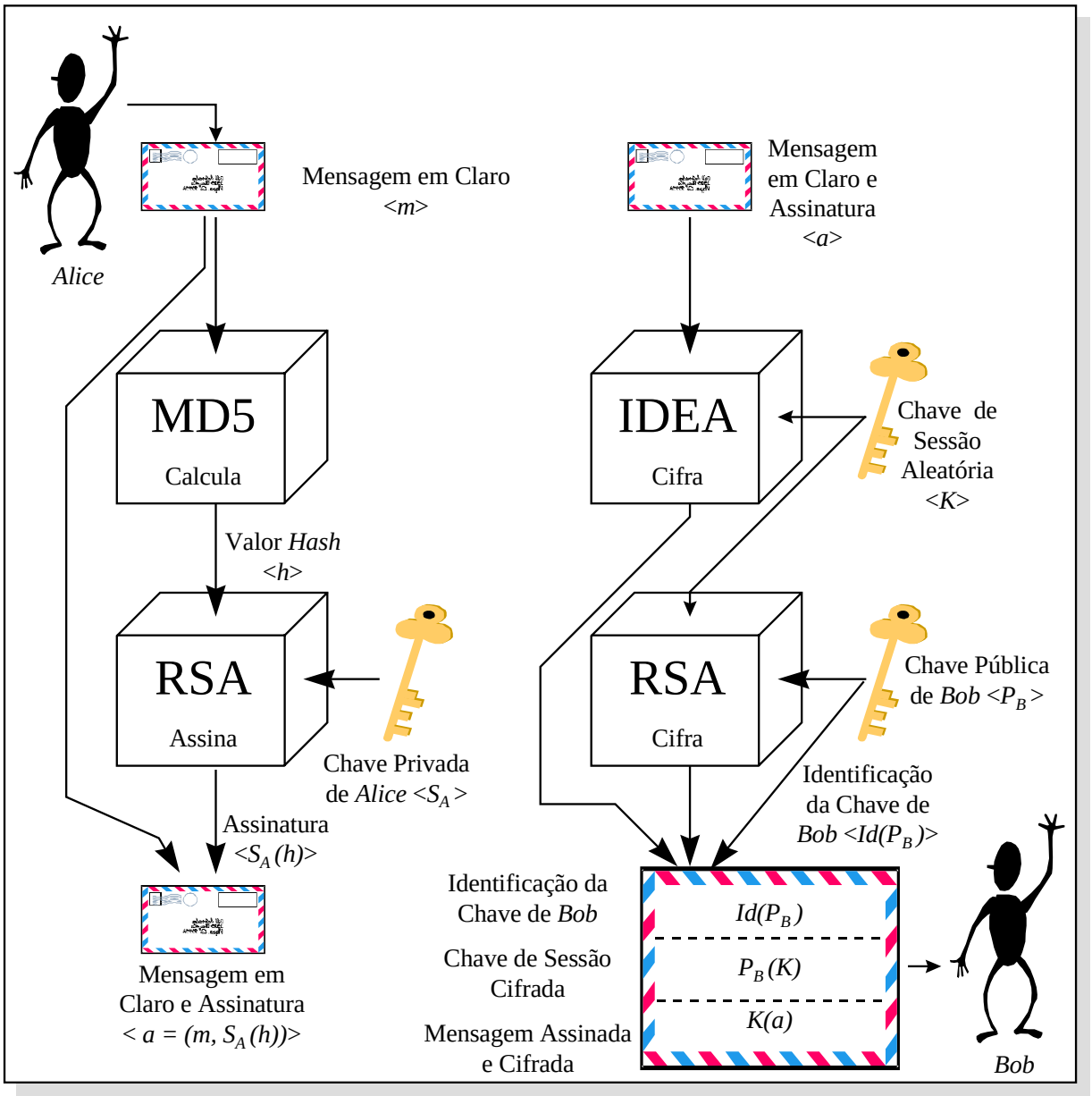


Figura 5.1 - Assinatura, Ciframento e Estrutura de Uma Mensagem PGP.

5.1. O Ambiente PGP.

[Zimmermann 94] [Cavalcanti 96] [Schneier 95] [Schneier 96]

O PGP é projetado para trabalhar junto com sistemas de correio eletrônico, com ênfase naqueles utilizados na Internet. Mas ele não se encontra integrado com nenhum programa de *e-mail* (UA) ou editor de texto. Ou seja, é um *software* totalmente independente do ambiente de correio eletrônico. Isto significa que o usuário precisa escrever a mensagem fora do ambiente de *e-mail*, utilizar o PGP para assiná-la e/ou cifrá-la, importar o arquivo assinado e/ou cifrado para o ambiente de correio eletrônico para, finalmente, enviar sua mensagem. De maneira análoga, na recepção da mensagem citada, é preciso primeiro exportá-la para depois poder decifrá-la e/ou verificar sua assinatura.

O fato do PGP ser um ambiente externo ao ambiente do correio eletrônico implica uma enorme perda de produtividade na sua utilização. Também demanda a aprendizagem de novos comandos que, muitas vezes, não são nada amigáveis para o usuário final. Assim, no PGP, para se obter a segurança desejada, além do programa de correio eletrônico, o usuário precisa aprender a manusear um programa de criptografia.

Além disso, no PGP, conforme já exposto no item 2.1.5, os usuários assinam as chaves uns dos outros para criar uma comunidade de usuários PGP interligados. Isto faz com que, o usuário que deseja se comunicar de forma segura, necessite realizar a tarefa adicional de descobrir, de algum modo, a chave pública do recebedor e verificar se ela é confiável.

5.2. Enviando uma Mensagem PGP.

[Zimmermann 94] [Cavalcanti 96] [Schneier 95] [Schneier 96]

O envio de uma mensagem PGP consiste de quatro passos (Figura 5.2):

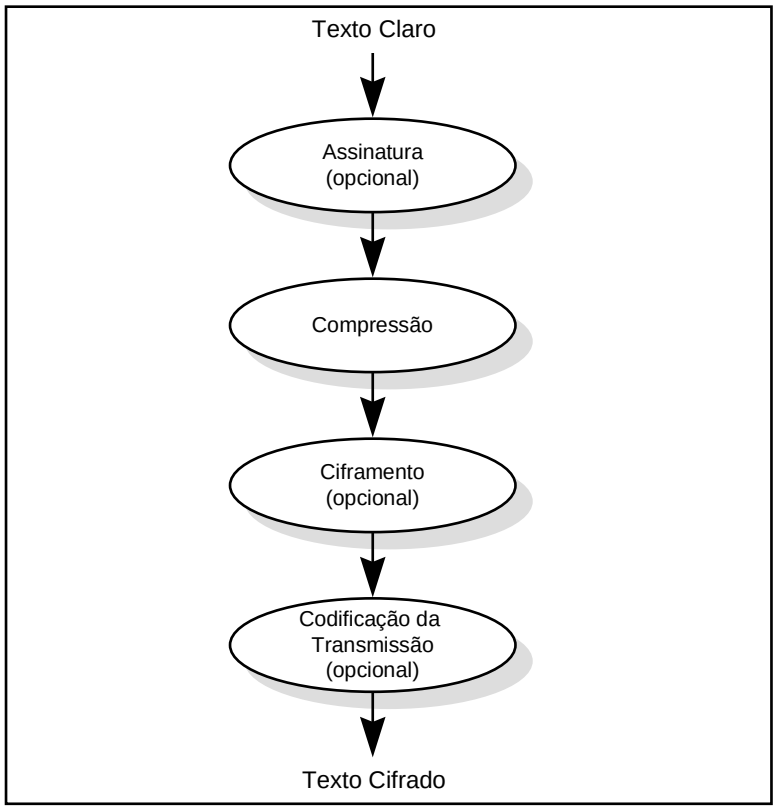


Figura 5.2 - Preparação de Uma Mensagem PGP [Schneier 95].

- **Assinatura (opcional).** As mensagens PGP podem ser assinadas pelo emissor. As assinaturas são opcionais; o emissor não é obrigado a assinar sua mensagem. Há casos em que um emissor não deseja assinar sua mensagem. Por exemplo, quando envia um *e-mail* através de um serviço de *remailer* anônimo. As assinaturas PGP permitem que o recebedor verifique tanto a identidade do emissor quanto o fato de que a mensagem não foi falsificada durante o trânsito.

  A assinatura digital do PGP envolve o uso da função MD5 e do algoritmo de chave pública RSA. Para criar uma assinatura digital, primeiro o PGP calcula o valor *hash* MD5 da mensagem. Então, ele cifra o valor *hash* utilizando a chave privada do emissor. Finalmente, o valor *hash* cifrado é anexado à mensagem.

  Embora as assinaturas digitais sejam normalmente anexadas às mensagens ou aos arquivos que assinam, este não é o único caso. O PGP também permite assinaturas destacadas. Uma assinatura destacada pode ser armazenada e transmitida em separado da mensagem que assina. Isto pode ser

útil para manter um arquivo de *log* de assinaturas separado de todas as mensagens enviadas ou recebidas.

Pode servir também quando mais de uma parte deve assinar um contrato legal. Cada assinatura pessoal é independente e, portanto, é aplicada somente sobre a mensagem. Isto é, ambos *Alice* e *Bob* assinam somente a mensagem; o segundo assinante não assina a primeira assinatura[54].

- **Compressão.** Após a assinatura, o PGP compacta a mensagem. Esta compressão reduz o tamanho da mensagem. Contudo, o simples ato de reduzir o tamanho da mensagem elimina qualquer redundância no texto claro; isto, em geral, dificulta a criptoanálise.

  O algoritmo de compressão é o mesmo utilizado no programa de compressão de domínio público ZIP 2.0. É importante observar que o PGP compacta a mensagem após aplicar a assinatura (porém, antes do ciframento). Assim, a assinatura é compactada junto com o restante da mensagem.

- **Ciframento (opcional).** O PGP provê a confidencialidade cifrando as mensagens a serem transmitidas ou arquivos de dados a serem armazenados localmente, utilizando o algoritmo de ciframento convencional (de chave secreta) IDEA.

  Contudo, o PGP não obriga o emissor a cifrar a mensagem. Isto pode ser útil quando um emissor deseja enviar uma mensagem para grupos públicos, como os *Usenet Newsgroups*. Neste caso, ele deseja que qualquer um possa ler o *e-mail* e, por isto, não utiliza ciframento. No máximo, o emissor assina a sua mensagem.

  O PGP utiliza, para cifrar mensagens, o algoritmo de ciframento IDEA, no modo CBC, descrito no item 4.2.1. Deste modo, ele gera uma chave IDEA aleatória de 128 bits e um vetor IV aleatório de 64 bits toda vez que cifra uma nova mensagem. Esta chave e este vetor são utilizados pelo IDEA para cifrar o conteúdo da mensagem.

  Após o ciframento da mensagem, o PGP cifra a chave de sessão utilizada através do algoritmo RSA empregando, para isto, a chave pública do recebedor (*Bob*). Se houver mais de um recebedor, o PGP cifra múltiplas cópias desta chave, uma cópia para cada chave pública de recebedor existente.

[54] De outra forma, as assinaturas seriam aninhadas, com o segundo assinante assinando tanto o documento quanto a primeira assinatura.

O PGP pode assinar e cifrar uma única mensagem. Primeiro, deve-se gerar a assinatura e anexá-la à mensagem. Depois, a mensagem e sua assinatura são cifradas pelo IDEA, utilizando-se uma chave aleatória de ciframento (chave de sessão). Finalmente, a chave de sessão é cifrada pelo RSA e anexada ao bloco cifrado (conforme já foi descrito na Figura 5.1).

Observa-se que a assinatura digital é cifrada junto com a mensagem. Neste caso, é impossível verificar-se a assinatura, feita sobre uma mensagem assinada e cifrada desta maneira, sem antes decifrá-la.

- **Codificação da transmissão (opcional).** Com o PGP, pelo menos parte do bloco a ser transmitido segue cifrado. Se for utilizada uma assinatura destacada, então o valor *hash* será cifrado (com a chave privada do emissor). Se for utilizado ciframento, a mensagem e sua assinatura (se presente) serão ambas cifradas (com uma chave convencional aleatória única).

  Assim, ou parte ou todo o bloco resultante consiste de uma *stream* de bytes arbitrários. Entretanto, muitos sistemas de correio eletrônico somente permitem a utilização de blocos de texto ASCII. Para acomodar-se a esta restrição, o PGP converte a *stream* binária de 8 bits de texto cifrado para uma *stream* de caracteres ASCII imprimíveis[55].

5.3. Recebendo uma Mensagem PGP.

[Zimmermann 94] [Cavalcanti 96] [Schneier 95] [Schneier 96]

Receber uma mensagem PGP envolve desemaranhar tudo o que foi feito pelo emissor. Felizmente, o programa PGP cuida de tudo. Primeiro, o programa procura ver que versão do PGP foi utilizada pelo emissor. No momento, a versão corrente gratuita é a 5.0 (antiga 3.0), embora por enquanto seja mais comum encontrar pessoas utilizando a versão internacional 2.6.3i; a versão corrente comercial da ViaCrypt é a 2.7.1, junto com a 4.0.

Há também várias versões comerciais disponíveis do PGP 5.0 para Windows95/NT e para Macintosh. Para obter uma posição atualizada das versões, sugere-se consultar a página Web <*http://www.ifi.uio.no/pgp/versions.shtml*> (*Current PGP Versions*).

Para se receber uma mensagem PGP, é necessário passar pelos seguintes passos:

- **Decodificação da transmissão.** Se a mensagem estiver codificada para transmissão, ela será decodificada do formato ASCII imprimível de volta para o texto cifrado de 8 bits.
- **Deciframento.** O PGP verifica se a mensagem está cifrada. Se estiver, ele tenta decifrá-la. Primeiro, o programa procura pelo pacote que contém a chave de ciframento aleatória (ou chave de sessão) cifrada, para ver se consegue decifrá-la. Pode haver mais de um pacote deste tipo[56]. Se o PGP puder, ele então decifra a chave de sessão aleatória utilizando a chave privada do recebedor. Então, ele decifra o conteúdo da mensagem empregando a chave de sessão aleatória.
- **Verificação da Assinatura.** O PGP determina se a mensagem está assinada. Se estiver, o PGP decifra o valor *hash* utilizando a chave pública do emissor. Então, ele gera um novo valor *hash* para a mensagem recebida e o compara ao valor decifrado. Se ambos forem idênticos, a mensagem é aceita como válida e autêntica.
- **Disposição da Mensagem.** Após o recebedor ler a mensagem PGP, ele tem uma série de opções. Ele pode armazenar a mensagem decifrada sem a assinatura anexa. Pode também armazenar a mensagem decifrada com a informação da assinatura. Esta forma de armazenamento é apropriada se o recebedor desejar enviar adiante uma mensagem assinada por terceiros, e também para prover proteção contra modificações enquanto a mensagem ficar guardada. Ou, se desejar proteger sua confidencialidade, pode também armazenar a mensagem no formato cifrado.

5.4. Certificando e Distribuindo Chaves Públicas PGP.

[Garfinkel 95][Cavalcanti 96] [Schneier 95] [Zimmermann 94] [Schneier 96]

O item 2.1.5 já apresentou uma discussão sobre o gerenciamento das chaves públicas feito pelo PGP. Pretende-se, agora, complementar a apresentação, discorrendo sobre alguns comandos para gerenciamento de chaves PGP (Tabela 5.1) e sobre conceitos ainda não abordados.

---

[55] O esquema utilizado para este propósito é a conversão *radix-base64*. Este formato também anexa um código para detectar erros de transmissão, sendo o mesmo algoritmo de codificação utilizado pelo PEM.

[56] Podem existir múltiplos recebedores, ou um único recebedor pode possuir múltiplas chaves públicas.

Comandos	*Descrição*
pgp -kg	Gera o próprio par de chaves pública / privada.
pgp -ks <nome_usuário> [-u <próprio_nome>]	Assina e certifica a chave pública de alguém no chaveiro público pessoal.
pgp -krs <nome_usuário>	Remove a assinatura selecionada de alguém no chaveiro público pessoal.
pgp -ka <chave_pública>	Acrescenta a chave pública de alguém no chaveiro público pessoal.
pgp -ke <nome_usuário>	Edita os parâmetros de confiança da chave pública de alguém selecionado no chaveiro público pessoal.
pgp -kc <nome_usuário>	Verifica o conteúdo, os atributos de confiança e as assinaturas de certificação da chave pública de alguém no chaveiro público pessoal.
pgp -kvc <nome_usuário>	Obtém a impressão digital (*fingerprint*) da chave pública de alguém no chaveiro público pessoal.
pgp -kx <nome_usuário> <arquivo_chave>	Extrai a chave pública de alguém, do chaveiro público pessoal, para passá-la adiante. A chave fica armazenada no <arquivo_chave>.

Tabela 5.1 - Comandos para Gerenciamento de Chaves PGP.

5.4.1. Certificação da Chave Pública PGP.

[Zimmermann 94] [Cavalcanti 96] [Schneier 95]

Uma chave pública PGP é composta pelos componentes de chave e por uma identificação de usuário (em geral, um nome seguido de um endereço Internet). Podem existir, também, assinaturas anexadas à chave. Como já exposto no item 2.1.5 mencionado, cada pessoa gera seu próprio par de chaves pública/privada (opção -kg). Sua chave pública é assinada pelas pessoas que a conhecem. Estas assinaturas servem como *certificados* da validade da chave, podendo ajudar a convencer outras pessoas sobre este fato.

Um certificado (ou seja, uma chave pública assinada) parece-se com um pequeno arquivo ASCII. Pode ser colocado em um disco e ser entregue a alguém. Pode ser enviado a outra pessoa pelo sistema de correio eletrônico. Pode também ser colocado em um servidor de chaves PGP ou mesmo no arquivo *.plan* do próprio usuário. Deve ser distribuído o mais amplamente possível, pois é o que as pessoas utilizarão para o envio de mensagem seguras para o seu dono.

A certificação no PGP funciona assim: *Alice* extrai (opção -kx) e passa fisicamente sua chave pública para o usuário *Bob*, que a assina. *Bob*, então, guarda uma cópia da chave pública de *Alice*, assinada, e devolve-lhe uma cópia também assinada.

Vamos supor que *Carol* seja uma usuária que também possui a chave pública de *Bob* e confia nele para certificar chaves. *Alice* deseja se comunicar com *Carol* e envia para ela sua chave pública. *Carol* verifica, então, que *Bob* assinou a chave de *Alice* e, após isto, finalmente aceita a chave de *Alice* como válida. *Alice* também pode confiar no usuário *Dave* e utilizar o mesmo procedimento.

5.4.2. Verificação da Chave PGP Alheia.

[Zimmermann 94] [Garfinkel 95][Cavalcanti 96] [Schneier 95]

Toda chave pública PGP tem dois atributos referentes à verificação da legitimidade da chave. São eles: a confiança e a validação. Na realidade, o atributo confiança é relativo ao dono da chave, enquanto o atributo validação é relativo à chave propriamente dita.

O valor da confiança é atribuído pelo próprio usuário e pode ter, tipicamente, os seguintes valores: indefinida, negativa, completa e absoluta. Esses valores indicam o quanto se confia no proprietário da chave para certificar (assinar) outras chaves. Se a confiança for *indefinida*, significa que você ainda não definiu, ou não sabe definir, se confia ou não naquele proprietário. Se *negativa*, significa que você não confia naquele proprietário para autenticar outras chaves. Se *completa* ou *absoluta*, significa que você confia naquele proprietário para certificar outras chaves.

A diferença entre a confiança completa e absoluta é que, neste último caso, o proprietário da chave é o próprio usuário. A confiança absoluta é automaticamente atribuída à chave pública do próprio usuário quando gerada. Para as demais chaves, esses valores podem ser informados pelo usuário em duas ocasiões: quando for acrescentar uma chave ao seu arquivo pessoal de chaves públicas conhecido

como chaveiro público, discutido no item 5.4.3 a seguir (opção -ka) ou durante o processo de edição dos parâmetros de confiança de uma chave pública (opção -ke).

Já o atributo de validação da chave terá seu valor definido, automaticamente, conforme o valor do atributo de confiança. Os valores possíveis, normalmente, são: indefinida, ignorada, negativa, e completa. Será indefinida se não contiver nenhuma assinatura, ou nenhuma assinatura de usuário de confiança definida. Será ignorada se só contiver uma ou mais assinaturas desconhecidas, isto é, de usuários cujas chaves não constem do chaveiro público.

Será negativa se só contiver assinatura de um ou mais usuários de confiança igualmente negativa. Será completa se contiver pelo menos uma assinatura de usuário de confiança absoluta. Será também completa se contiver pelo menos uma assinatura de usuário de confiança completa e se a chave referente a essa assinatura tiver sua validação igualmente completa, isto é, se essa assinatura pertencer a uma chave[57].

É esse atributo, portanto, que dá o grau de legitimidade de determinada chave pública. Naturalmente, por princípio, só se deve empregar chaves públicas de validação completa ou definida. Não obstante, a observância ou não desse preceito é da própria conta e risco do usuário.

A verificação dos atributos de confiança e validação é realizada com a opção -kc. Assim executado, o PGP, num primeiro bloco, lista todas as chaves do chaveiro público. Num segundo bloco mostra, para cada chave, o valor do atributo validação e para cada assinatura o valor do atributo confiança.

O PGP também permite uma opção para obter a impressão digital (*fingerprint*) de uma chave. Uma impressão digital é o valor *hash* MD5 de uma chave pública. Ela facilita a verificação de uma chave. Se *Alice* desejar enviar para *Bob* sua chave pública, ela pode enviá-la por *e-mail*.

Quando *Bob* a receber, precisará verificar se a chave é válida. Se a chave de *Alice* não for assinada por ninguém em quem *Bob* confie, ele pode telefonar para *Alice* e pedir que lhe dite a impressão digital de sua chave pública. Como é um valor pequeno (16 bytes), isto pode ser feito rapidamente. Após comparar a impressão digital conseguida pelo telefone com a impressão digital (resultado do cálculo do valor *hash*) da chave pública obtida via *e-mail*, se ambos os valores combinarem, *Bob* pode confiar na validade da chave pública de *Alice*.

[57] Estes parâmetros podem ser modificados para se ajustarem à "paranóia" de cada um.

5.4.3. Armazenando Chaves PGP.

[Zimmermann 94] [Garfinkel 95] [Schneier 95]

As chaves do PGP são armazenadas em chaveiros (*key rings*). Um chaveiro é um arquivo que faz o papel de um molho de chaves. Os usuários PGP armazenam cópias das chaves públicas de outros usuários.

Assim, se *Alice* se comunica com *Bob* regularmente, precisa manter uma cópia da chave pública de *Bob* em seu chaveiro público. Se deseja enviar uma mensagem segura para *Bob* e não possui a chave pública dele em seu chaveiro, *Alice* precisa obtê-la de algum modo (pode pedir a *Bob*, dar uma olhada em seu arquivo *.plan* através de um comando *finger* ou pesquisar em algum servidor de chaves PGP) . Após conseguí-la, *Alice* a adiciona em seu chaveiro público para poder enviar mensagens seguras a *Bob*.

O chaveiro público é armazenado de forma não cifrada. Afinal, não precisa ser mantido secreto. Mas o PGP mantém dois chaveiros: um chaveiro público, contendo todas as chaves públicas de quem o usuário conhece, e um chaveiro secreto, que contém uma ou mais chaves privadas associadas ao usuário (podem estar associadas ao seu nome ou ao(s) seu(s) pseudônimo(s)). Estas chaves privadas ficam protegidas por uma *passphrase*, como já abordado na seção 4.3.

6. Descrição do PEM

O PEM não é um produto; é um padrão. Conforme já apresentado no item 2.1.5, o PEM é um padrão proposto que define ciframento de mensagem e procedimentos de autenticação de forma a prover serviços de *e-mail* acrescidos de privacidade na transferência de mensagens via Internet. Ele é normalmente utilizado com o protocolo SMTP (discutido na seção 1.5), mas pode ser empregado com outro esquema de mensagens eletrônicas, como o padrão X.400, por exemplo.

O PEM foi projetado para ser compatível com uma série de modelos de gerenciamento de chaves. Ele possui mecanismos para o emprego de criptografia convencional (de chave secreta) ou de chave pública para o gerenciamento de chaves (conforme exposto no item 2.1.5). Entretanto, como todas as implementações atuais do PEM utilizam criptografia de chave pública para o gerenciamento de chaves, será dada ênfase a este modelo.

O PEM utiliza dois tipos de chaves: a *Data Encryption Key* (DEK), chave única (chave de sessão secreta) utilizada para cifrar o conteúdo de uma mensagem; e a *Interchange Key* (IK), utilizada para cifrar a chave DEK.

Este capítulo encontra-se dividido da seguinte forma: a seção 6.1 descreve o ambiente PEM. A seção 6.2 compara os tipos de mensagens PEM. A seção 6.3 descreve os passos para o envio de uma mensagem PEM, enquanto que a seção 6.4 descreve os passos para seu recebimento. A seção 6.5 descreve os certificados de chave pública utilizados pelo padrão PEM (X.509) e, finalmente, a seção 6.6 discute sobre uma das principais implementações do padrão PEM: o programa RIPEM, cujas funções são acionadas pelo programa "ProtegeMail", descrito na Parte III deste trabalho.

6.1. O Ambiente PEM.

[RFC 1423] [Schneier 95] [Schneier 96]

O PEM foi projetado para trabalhar com os sistemas de *e-mail* existentes, principalmente os sistemas de *e-mail* utilizados na Internet [58] . Sendo assim, o PEM foi projetado para adaptar-se às arquiteturas de sistema de *e-mail* existentes. As mensagens PEM podem ser criadas com editores de texto, e a maioria dos sistemas de *e-mail* não corrompem as mensagens PEM em trânsito. O PEM possui utilidade em uma variedade de aplicações de segurança, em uma variedade de contextos e em uma variedade de plataformas de *hardware* e de *software*. As principais características de projeto do PEM são:

- O PEM não se restringe a uma máquina ou a um sistema operacional em particular; ao invés disso, permite interoperabilidade entre uma extensa gama de sistemas. Ele independe de qualquer característica do programa de *e-mail*, do sistema operacional, do *hardware* ou da rede.
- O PEM é compatível com o correio eletrônico inseguro normal. E também com uma variedade de sistemas de *e-mail*, protocolos e *interfaces* de usuários.
- O PEM provê proteção da privacidade de *e-mail* endereçado a listas de discussão.
- O PEM é compatível com uma variedade de modelos de distribuição de chaves, incluindo (mas não limitado a eles): a distribuição manual; a distribuição centralizada, baseada em criptografia convencional; e o uso de certificados de chave pública.

[58] O PEM também trabalha com a rede CompuServe, America Online, GEnie e Delphi, entre outras.

O PEM provê os meios para a identificação do algoritmo utilizado em cada mensagem, para cada recebedor. Atualmente, um conjunto particular de algoritmos está identificado para utilização pelo PEM, mas o padrão permite o emprego de diferentes algoritmos.

Os projetistas esperam que os novos algoritmos sejam também agrupados em conjuntos; do contrário, seria facilmente possível haver muitas combinações diferentes de algoritmos utilizados por diferentes pessoas. Um dado emissor e um determinado recebedor poderiam não possuir os mesmos algoritmos. Contudo, é possível para uma implementação PEM permitir aos seus usuários efetuarem variações (*switch*) entre algoritmos.

O conjunto atual de algoritmos do PEM, especificado pela RFC 1423, encontra-se definido na tabela a seguir (Tabela 6.1).

Conjunto de Algoritmos para:	***Algoritmo Utilizado***
Ciframento de Dados.	DES, no modo *Cipher Block Chaining*.
Gerenciamento de Chaves.	DES, no modo *Electronic Codebook.*
	DES, no modo *Cipher Block Chaining*.
	RSA.
Verificação da Integridade da Mensagem.	RSA e MD2.
	RSA e MD5.
Assinatura Digital.	RSA e MD2.
	RSA e MD5.

Tabela 6.1 - Conjunto de Algoritmos Atuais do PEM.

6.2. Tipos de Mensagens PEM.

[RFC 1421-1424] [Cavalcanti 96]

O cabeçalho de uma mensagem PEM possui uma série de campos especiais. O primeiro deles é o *Proc-Type:*, que identifica o tipo de processamento executado sobre a mensagem. Há três tipos de mensagem

PEM, sendo que cada tipo é processado de uma determinada forma, oferecendo ao usuário uma diferente combinação de características. São eles:

- **MIC-CLEAR.** Uma mensagem do tipo MIC-CLEAR provê integridade e autenticação, mas não confidencialidade. A mensagem é enviada em claro. Qualquer pessoa pode ler a mensagem, mas os usuários PEM também serão capazes de verificar a autenticidade da mensagem. Esta opção é útil para o envio de mensagens para grupos de discussão *Usenet Newsgroups*, por exemplo.

- **MIC-ONLY.** É o mesmo que uma mensagem MIC-CLEAR, exceto que possui um passo de codificação adicional. O esquema utilizado para esta codificação é a conversão *radix-base64*, similar ao PGP. Esta codificação permite que as mensagens PEM passem incólumes através de vários *gateways* de *e-mail*.

- **ENCRYPTED.** Uma mensagem ENCRYPTED possui todas as características de uma mensagem MIC-CLEAR - integridade, autenticação - com a adição da confidencialidade. Somente o recebedor tencionado de uma mensagem deste tipo pode ler seu conteúdo. Um intruso não pode ler a mensagem, mas pode utilizar o PEM para verificar a origem e a integridade da mensagem. Mensagens ENCRYPTED passam pela mesma codificação das mensagens MIC-ONLY.

  A figura, a seguir, mostra a estrutura de uma mensagem ENCRYPTED utilizando gerenciamento de chave pública (Figura 6.1).

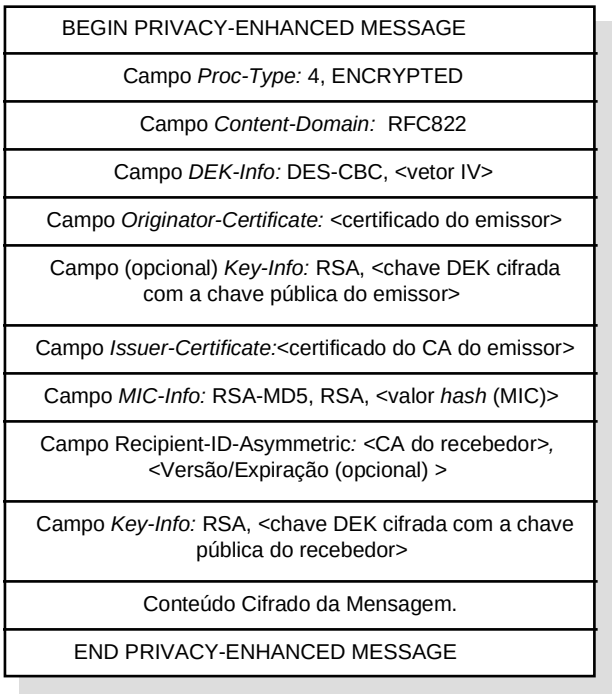


Figura 6.1 - Estrutura de uma mensagem PEM ENCRYPTED (Chave Pública).

6.3. Enviando uma Mensagem PEM.

[RFC 1421-1424] [Cavalcanti 96] [Schneier 95] [Schneier 96]

O envio de uma mensagem PEM envolve quatro passos [Schneier 95]:

1. Padronização.
1. Autenticação da Origem e da Integridade da Mensagem.
1. Ciframento (opcional).
1. Codificação da Transmissão (opcional).

Estes passos são transparentes ao usuário. Ele apenas precisa informar o tipo de mensagem a enviar e, se estiver utilizando ciframento, para quem está enviando a mensagem. O PEM faz o restante.

Cada mensagem PEM não segue exatamente cada passo. Uma mensagem MIC-CLEAR segue somente os passos 1 e 2. Uma mensagem MIC-ONLY segue os passos 1, 2 e 4. Uma mensagem ENCRIPTED segue os passos de 1 a 4.

Será apresentada uma breve descrição do que ocorre em cada passo:

- **Padronização.** O primeiro passo traduz a mensagem, de qualquer representação em particular, utilizada por um processador de texto ou programa de correio eletrônico, para uma representação comum, padrão entre todas as plataformas e todas as máquinas na rede.

  Assim, uma mensagem enviada por *Alice*, a partir de um determinado computador, pode ser recebida por *Bob* utilizando uma plataforma de computador totalmente diferente. O padrão PEM deseja, então, assegurar-se de que *Bob* sempre possa ler a mensagem de *Alice*.

  O PEM permite diferentes transformações de padronização em diferentes ambientes de mensagem. As transformações são indicadas no campo *Content-Domain:* (vide Figura 6.1) do formato de mensagem PEM. Na Internet, utiliza-se a mesma padronização empregada pelo SMTP, especificada na RFC 822 (vide item 1.3.1). Correntemente, é a única transformação prevista nos documentos PEM.

- **Autenticação da Origem e da Integridade da Mensagem.** Este passo começa pelo cálculo do valor *hash* da mensagem (chamado de MIC). Este é o código que diz ao recebedor que a mensagem não foi modificada em trânsito.

  O padrão permite a escolha entre algoritmos MIC; atualmente, há dois especificados nos documentos PEM: MD2 e MD5[59]. O valor *hash* ou MIC é calculado com base na versão padrão da mensagem.

  Para prover autenticação do emissor e integridade da mensagem, o MIC deve ser protegido de alguma maneira que o relacione ao emissor. As assinaturas digitais resolvem este problema. Deste modo, o MIC é assinado pelo emissor da mensagem. Assim, a assinatura digital pode ser verificada por qualquer recebedor - inclusive um intruso - mas *não pode ser falsificada*.

[59] Recomenda-se o MD5, ao invés do MD2, por ser mais seguro.

O PEM permite uma variedade de algoritmos de assinatura; porém, o padrão somente especifica o uso do RSA atualmente (conforme descrito na Tabela 6.1). Versões futuras do PEM poderão permitir o emprego do DSA ou do ElGamal.

O algoritmo MIC e o algoritmo de assinatura digital são especificados no campo *MIC-Info:* da estrutura de mensagem PEM (vide Figura 6.1).

- **Ciframento.** O terceiro passo é o ciframento. Ele é opcional; o PEM somente cifra uma mensagem se o seu tipo for ENCRYPTED. O padrão PEM pode apoiar múltiplos algoritmos de ciframento, mas atualmente especifica somente um: o DES, no modo CBC (Tabela 6.1). O algoritmo de ciframento é especificado no campo *DEK-Info:*, do cabeçalho de mensagem PEM, junto com qualquer dado adicional que o algoritmo necessite. Como, por exemplo, o vetor de inicialização IV, no caso do DES modo CBC (vide Figura 6.1 e Tabela 4.2).

  A mensagem é cifrada com uma chave aleatória inicialmente conhecida apenas pelo emissor. Para transferir esta chave de modo seguro para o recebedor, o emissor precisa utilizar criptografia de chave pública. Atualmente, o RSA é o único algoritmo definido nas especificações PEM para este propósito (Tabela 6.1). A chave cifrada segue no campo *Key-Info:*, junto com o nome do algoritmo de chave pública empregado; o nome do recebedor autorizado segue no campo *Recipient-ID-Asymetric:* (Figura 6.1).

  Como no PGP, mesmo que a mensagem tenha múltiplos recebedores, ela será cifrada apenas uma vez. Uma cópia da chave de ciframento da mensagem é cifrada com a chave pública de cada recebedor autorizado. Neste caso, podem existir múltiplos campos *Recipient-ID-Asymetric:* [60] no cabeçalho PEM, cada um seguido por um diferente *Key-Info:* . Cada par destes campos de cabeçalho PEM provê a informação necessária para um recebedor decifrar a mensagem.

  O PEM também possibilita a distribuição de chave sem o uso de criptografia assimétrica. Neste caso, a mensagem ainda é cifrada com uma chave de ciframento aleatória, sendo esta chave cifrada por um algoritmo convencional, uma vez para cada recebedor, utilizando chaves previamente combinadas entre as partes.

[60] A informação contida no campo *Recipient-ID-Asymetric :* consiste do nome de quem emitiu o certificado de chave pública do recebedor e de um número serial que identifica este certificado; uma combinação que individualiza o recebedor.

- **Codificação.** O último passo do envio de uma mensagem PEM é a codificação para transmissão. É um passo opcional; somente as mensagens do tipo ENCRYPTED ou MIC-ONLY são codificadas. Como já exposto na seção 5.2, o esquema de codificação corrente é o mesmo utilizado no PGP. Contudo, o esquema de codificação é outro parâmetro PEM que pode ser modificado.

  A mensagem PEM é, portanto, encapsulada para não sofrer modificação feita pelos sistemas de *e-mail* durante o trânsito. Em seguida, ela é enviada para seu destino.

6.4. Recebendo uma Mensagem PEM.

[RFC 1421-1424] [Cavalcanti 96] [Schneier 95] [Schneier 96]

Da mesma forma que no envio, no recebimento de uma mensagem as implementações PEM cuidam de todos os detalhes; o usuário não precisa se preocupar com nenhum dos passos do recebimento.

Quando um programa PEM recebe uma mensagem, ele primeiro vasculha toda a mensagem até encontrar os limites PEM [61] da mensagem recebida; a seguir, verifica no cabeçalho que versão do PEM foi empregada no processamento da mensagem. Então ele prossegue no trato da mensagem.

- **Decodificação.** Se a mensagem for do tipo ENCRYPTED ou MIC-ONLY, o primeiro passo é inverter o passo da codificação realizado no lado do emissor. Se a mensagem for cifrada, então a codificação (no formato de 6 bits) é revertida para o texto cifrado no formato 8-bit. Se não estiver cifrada, então ela é revertida para o texto claro padrão.

- **Deciframento.** Se a mensagem for do tipo ENCRYPTED, o PEM precisa decifrar a mensagem. O programa PEM do recebedor vasculha o cabeçalho da mensagem PEM para localizar o campo *Recipient-ID-Asymetric:* que o individualiza (pode haver mais de um campo deste tipo, indicando múltiplos recebedores). Então, ele examina o próximo campo, o *Key-Info:*, que especifica o algoritmo de ciframento da mensagem e também contém a chave cifrada. Assumindo-se o uso de criptografia de chave pública para a distribuição de chave, o recebedor utiliza sua chave privada para recuperar a chave empregada no ciframento da mensagem.

[61] Linhas BEGIN PRIVACY-ENHANCED MESSAGE e END PRIVACY-ENHANCED MESSAGE .

O campo *DEK-Info:* especifica o algoritmo de ciframento, o modo e qualquer outro parâmetro necessário. Assim, o recebedor pode utilizar a chave de mensagem recém decifrada e (supondo o emprego do DES-CBC) o vetor IV contido no campo *DEK-Info:* para decifrar o texto da mensagem. Após o deciframento, o conteúdo da mensagem encontra-se no mesmo estado de processamento de uma mensagem MIC-ONLY ou MIC-CLEAR. O restante desta sessão, portanto, se aplica aos três tipos de mensagem.

- **Verificando a integridade da mensagem e a autenticidade do emissor.** O programa recebedor primeiro verifica o cabeçalho PEM para determinar que algoritmos MIC e de assinatura digital foram utilizados para a mensagem. Assumindo-se que foram utilizados o MD5 e o RSA, o recebedor pega a chave pública do emissor (contida no certificado do emissor existente no campo *Originator-Certificate:* da Figura 6.1) e decifra o valor (*hash*) MIC assinado.

  Finalmente, o programa PEM recebedor calcula o valor MIC sobre a forma padrão da mensagem, e então compara este valor com o valor MIC recém decifrado. Se ambos combinarem, então a mensagem foi autenticada. Caso contrário, foi forjada ou adulterada na transmissão.

  Para haver uma verdadeira identificação do emissor, o programa recebedor deveria examinar não somente o certificado do emissor, mas também o certificado da pessoa (CA) que certifica o emissor (campo *Issuer-Certificate:* da Figura 6.1), o certificado da pessoa que certifica esta última (PCA) e assim por diante, até chegar ao do IPRA mencionado no item 2.1.5 (vide Figura 2.10).

  O padrão PEM permite a inclusão de todos estes certificados no cabeçalho PEM de uma mensagem. Mas isto não ocorre na prática, pois as mensagens ficariam muito longas. Ao invés disso, espera-se que os usuários armazenem certificados localmente, para uso posterior.

- **Tradução.** Finalmente, a forma padrão da mensagem é traduzida para qualquer representação apropriada ao sistema do recebedor, sendo então tornada disponível para a leitura do recebedor. O sistema PEM recebedor informa ao usuário sobre a verificação da integridade da mensagem PEM e sobre a autenticação da identidade do emissor.

- **Disposição da Mensagem.** Após o recebedor ler a mensagem PEM, ele tem uma série de opções. Ele pode armazenar a mensagem na forma decifrada, sem estar padronizada e sem nenhum cabeçalho PEM. Pode também armazenar a mensagem decifrada, padronizada e com o cabeçalho PEM necessário para a verificação da assinatura. Esta forma de armazenamento é apropriada se o

recebedor desejar enviar adiante uma mensagem assinada por terceiros, e também para prover proteção contra modificações enquanto a mensagem ficar guardada.

Ou, se desejar proteger sua confidencialidade, pode também armazenar a mensagem no formato cifrado.

6.5. Os Certificados de Chave Pública X.509.

[Schneier 95] [RFC 1422]

Um certificado de chave pública é uma estrutura de dados especial utilizada para relacionar, de forma segura, uma chave pública a um grupo de atributos. Ele identifica *Alice* como uma pessoa com uma chave pública. Seu certificado é formado por sua chave pública, seu nome e algumas outras informações pessoais, todas agrupadas e assinadas por uma pessoa de confiança (o CA, na linguagem PEM, mencionado no item 2.1.5). Assim, qualquer pessoa do universo PEM pode verificar a assinatura do CA e assegurar-se de que a chave pertence a *Alice*.

O PEM utiliza certificados em conformidade com o padrão X.509. Tais certificados associam uma chave pública a um nome de diretório, ou seja, um endereço único na Internet, e identificam seus respectivos donos. A tabela, a seguir, descreve os campos contidos em um certificado PEM (Tabela 6.2).

Campo	*Descrição*
Version	Indica o número da versão do formato do certificado. Por enquanto, só existe a versão "0".
Serial Number	Individualiza o certificado entre aqueles assinados por quem o emitiu (o CA).
Signature	Identifica o algoritmo de assinatura digital empregado e especifica seus parâmetros associados.
Issuer Name	Contém o nome de quem assinou o certificado (nome do IPRA, PCA ou CA).
Validity Period	Consiste em um par de indicação de data/hora, representando o início e o fim do período de validade do certificado.
Subject Name	Nome do dono do certificado.
Subject Public Key	Contém a chave pública do dono do certificado e uma indicação do algoritmo assimétrico utilizado, junto com seus parâmetros necessários.

Tabela 6.2 - Conteúdo de Um Certificado PEM.

Como visto na Figura 2.10, os certificados PEM são organizados em uma estrutura hierárquica, no formato de uma árvore, tendo como raiz o IPRA e, logo abaixo, os PCA. Deste modo, os CA, subordinados aos PCA, assinam os certificados de outros CA ou dos usuários PEM. Há três tipos diferentes de CA identificados pelo PEM:

- ***Organizational***. Este tipo inclui companhias, organizações governamentais, instituições acadêmicas e associações profissionais.
- ***Residential***. Alternativamente, os usuários podem receber certificados baseados no lugar onde residem. Assim, um CA pode emitir certificados para as pessoas que moram em uma certa região geográfica.
- ***Personal***. Finalmente, os usuários podem se registrar com base apenas em seus nomes ou, se desejarem, seus pseudônimos. Obviamente, o usuário deste último grupo teria que conseguir uma *mailbox* com o mesmo pseudônimo com que se registrou. Este tipo de certificado permite ao usuário usufruir das características de segurança do PEM e permanecer anônimo.

Todos os usuários PEM possuem um nome único chamado de *Distinguished Name* (DN). O PEM não utiliza o endereço da Internet pessoal como seu nome; ele segue o padrão do diretório X.500. Este padrão provê um formato para identificar não apenas nomes de indivíduos, mas também organizações, dispositivos, nomes de listas de discussão, entres outros.

Esta convenção de nomes funciona como uma árvore. Na base, fica o atributo mais abrangente: o nome do país. Os atributos ulteriores detalham mais o DN: o nome do estado ou da província, da cidade, da organização, até chegar ao nome do usuário. Por exemplo, o DN do autor deste trabalho ficaria assim:

{C= BR, S= São Paulo, L= Campinas, O = UNICAMP, CN = Paulo Sergio Pagliusi}.

6.6. O RIPEM.

[Riordan 93] [Schneier 95]

O Riordan's Internet Privacy Enhanced Mail (RIPEM) é uma implementação assimétrica do padrão PEM, escrita originariamente por Mark Riordan[62] entre maio e julho de 1992. A versão atual do RIPEM não implementa todo o padrão PEM[63], mas mesmo assim é um programa de segurança de *e-mail* muito útil. O RIPEM permite que as mensagens eletrônicas tenham as quatro facilidades providas pelo PEM: ciframento (opcional), autenticidade do emissor, avaliação da integridade da mensagem e não repúdio da origem (sempre). Além disso, possui a grande vantagem de se optar, no ciframento da mensagem, pelo Triplo DES (opção *-e*, seguida de *-A des-ede-cbc*), bem mais seguro do que o DES (opção *-e*, seguida de *-A des-cbc*).

A versão atual do RIPEM é a 2.1. A maioria do seu código é de domínio público, a exceção do código RSA, que é uma biblioteca chamada de RSAREF, licenciada pela RSA Data Security Inc. em março de 1992. Esta biblioteca é considerada munição pelo governo americano. Portanto, ficou restrita sua exportação, pela legislação americana, para distribuição a quem não é cidadão ou pessoa com residência permanente nos EUA ou Canadá.

---

[62] Como o RIPEM competia com o PGP, houve interesse do pessoal da RSA Data Security Inc. em colocá-lo em seu *site ftp* e em acrescentar nele melhorias e otimizações substanciais, a ponto dele ganhar o prêmio de "Melhor Aplicação Construída sobre a RSAREF" em 1992.

O RIPEM foi desenvolvido para as plataformas: MS-DOS, Macintosh, OS/2, Windows NT e vários sistemas UNIX (incluindo: NeXT, SunOS, Solaris, ULTRIX, AIX, HP/UX, Irix, MPIS RISC/os, V/88, Apollo, SCO, 386BSD, Linux e ESIX). Ele funciona junto com a maioria dos sistemas de *e-mail*. O RIPEM utiliza:

- um arquivo de entrada ou *stream* contendo uma mensagem a ser processada (ou seja, a ser cifrada (opcional), assinada (obrigatório), decifrada ou verificada a assinatura);
- um arquivo de saída ou *stream* do processamento;
- um arquivo contendo a chave RSA privada cifrada do usuário; e
- um arquivo contendo as chaves públicas em claro dos (potencialmente) muitos usuários.

A figura, a seguir, contém a sinopse das principais opções de comando do RIPEM (Figura 6.2).

```
ripem { -e \ -d \ -g \ -c }
[-r recebedor(es)]  [-b # de bits]  [-A alg_ciframento]
[-m {encrypted \ mic-only \ mic-clear}]
[-u nome_emissor]  [-h ipr]  [-T amn]
[-p arq_entrada_ch_pública]  [-s
arq_entrada_ch_privada]
[-k {chave_para_chave_privada\-}]
[-P arq_saída_ch_pública]  [-S arq_saida_ch_privada]
[-y nome_servidor_chave_pública]  [-Y fs]
[-i arq_entrada]  [-o arq_saída]
[-D nível_debug ]  [-Z arq_debug]
[-R cfkms]  [-F arquivo_aleatório]  [-C string_aleatória]
<in>out
```

Figura 6.2 - Sinopse das Opções de Comando do RIPEM [Riordan 93].

Seguem alguns exemplos de comandos do RIPEM utilizando a plataforma UNIX:

1. para a geração de um par de chaves, seguida de seu registro em um servidor de chaves PEM, pode-se utilizar:

   ripem -g -S .ripemprv -P .ripempub -R eks

[63] O RIPEM não utiliza a certificação para autenticação de chaves prevista no padrão X.509.

mail ripem-register-keys@ripem.msu.edu <.ripempub

2. para a alteração da *password* da chave privada:

ripem -c -s .ripemprv -S newprv

Enter password to private key: (digitar a password antiga aqui.)

Enter new password to private key: (digitar a nova password aqui.)

Enter again to verify: (digitar a nova password de novo para verificação)

mv newprv .ripemprv

3. Para cifrar um *e-mail* utilizando o programa "mail" (a linha de comando *~| ripem* pode ser substituída por inteiro pela seqüência *~e*, convertida para um comando de ciframento na maioria dos sistemas UNIX):

pinheiros% mail lucchesi@dcc.unicamp.br

Subject: Olah

Esta eh uma mensagem de teste

~| ripem -e -r lucchesi@dcc.unicamp.br

Enter password to private key: (digitar a password aqui.)

(continue)

(Digitar Ctrl-D)

pinheiros%

Como já foi citado, as principais funções do RIPEM são acionadas pelo programa "ProtegeMail", descrito na Parte III deste trabalho.

7. Comparação entre PGP e PEM

O PGP e o PEM são ambos voltados para a segurança de correio eletrônico. Ambos assinam e cifram mensagens. Estão ambos baseados em criptografia de chave pública. Contudo, eles possuem diferentes filosofias. À primeira vista, a diferença pode ser entendida assim: O PEM baseia-se no conceito de uma organização hierárquica, enquanto o PGP baseia-se em uma rede distribuída de indivíduos.

Assim, o PEM é mais adequado para aplicações em companhias, governos e organizações militares. O PGP é definitivamente mais adequado para pessoas se comunicando de maneira informal via Internet. Por sinal, esta é uma das razões para seu sucesso. As seções deste capítulo comparam o PGP e o PEM em maiores detalhes. A seção 7.1 compara os modelos de confiança assumidos por ambos, enquanto a seção 7.2 enfoca as aplicações alvo. As duas seções seguintes comparam o ciframento e a assinatura (seção 7.3), e também o gerenciamento de chaves (seção 7.4) utilizado em cada programa. Finalmente, a seção 7.5 analisa os resultados da comparação, que foram decisivos para a idéia do desenvolvimento do sistema ProtegeMail, descrito na Parte III, objeto de estudo deste trabalho.

7.1. Modelo de Confiança.

[Riordan 93] [Schneier 95] [Zimmermann 94]

O PEM assume uma distribuição hierárquica de chaves. O controle centralizado advém de um pequeno número de servidores *root* (IPRAs) que são a fonte de toda a confiança. Assim, no PEM, “eu confio em você porque, acima de nós, há uma autoridade (um terceiro hierarquicamente superior) em comum que ampara esta confiança distribuindo certificados, ou diretamente para nós ou para nossas respectivas CA e PCA. Ou seja, porque seu CA assinou por você, seu PCA pertinente assinou pelo seu CA e o IPRA assinou pelo PCA”. No PEM, sempre haverá uma autoridade superior (em último caso, os IPRAs).

Já o PGP assume uma distribuição em rede (teia) da confiança: "eu conheço quem é você porque eu confio em alguém que acredita que você é quem diz ser". A confiança, neste caso, é uma decisão solitária do usuário. Ele decide em quem confia para atestá-lo para outros usuários. Assim, cada usuário pode obter todas as chaves diretamente, tornando-se sua própria origem de confiança, e cada terceira pessoa de confiança torna-se um CA confiável. Não há nada equivalente ao PCA ou ao IPRA, neste caso.

Deste modo, o PEM assume que o usuário confia na cadeia de certificados averiguados de uma mensagem assinada, ao passo que o PGP assume que somente o usuário está qualificado para decidir em quem confiar.

7.2. Aplicações Alvo.

[Schneier 95]

O PEM foi projetado especificamente para trabalhar com correio eletrônico. No padrão PEM, a autenticação é considerada mais importante do que a privacidade (é impossível enviar uma mensagem PEM que não esteja autenticada).

O PGP considera a privacidade pelo menos tão importante quanto a autenticação, e talvez até mais importante. Ele foi originariamente projetado para segurança de correio eletrônico, mas o PGP também é muito útil para a segurança de arquivos binários arbitrários. Em adição, o PGP comprime os

arquivos binários antes de cifrá-los. Isto reduz inclusive o tamanho dos arquivos convertidos pelo *radix-64*.

O projetistas do PEM concentraram-se mais na autenticidade de identidade, e menos nos graus de confiança. O criador do PGP possui uma visão mais pragmática, gerando um modelo de confiança flexível. Ainda que, às vezes, a confiança e a identidade ficam igualadas.

7.3. Ciframento e Assinatura.

[Riordan 93] [Schneier 95] [Zimmermann 94]

Ambos PGP e PEM podem gerar mensagens assinadas e cifradas; ambos podem gerar mensagens assinadas em claro. Somente o PGP pode cifrar mensagens não assinadas. Esta é uma diferença significativa. Com o PEM, é impossível o envio de mensagens não assinadas. Os usuários PEM não podem se esconder por trás de *remailers* anônimos, assunto discutido no item 9.3.1. Uma mensagem cifrada pelo PGP pode estar sem assinatura e anônima. A próxima tabela resume estas diferenças (Tabela 7.1).

É impossível verificar uma assinatura PGP se a mensagem ainda estiver cifrada[64]. É possível verificar uma assinatura PEM mesmo se a mensagem estiver cifrada. Esta é uma diferença sutil, mas significa que qualquer pessoa (inclusive *Eve*) pode verificar quem enviou uma mensagem PEM cifrada, o que pode permitir análise de tráfego. Não somente é impossível o envio de uma mensagem PEM anônima cifrada, como também é impossível o envio de uma mensagem PEM que seja anônima para a rede, mas que possa ser autenticada pelo recebedor autorizado.

[64] É claro que a assinatura pode ser feita após o ciframento, mas esta não é a forma correta de se trabalhar com o PGP.

Tipo de Segurança	*Terminologia PEM*	*Opções da Linha de Comando PGP*
Texto claro, assinado.	MIC-CLEAR	-sta +clearsig=on
Somente assinado.	MIC-ONLY	-sta
Assinado e cifrado.	ENCRYPTED	-stea
Somente cifrado.	<não há>	-tea

Tabela 7.1 - Tipos de Mensagens PGP e PEM [Schneier 95].

7.4. Geração, Distribuição e Revogação de Chaves.

[Riordan 93] [Schneier 95] [Zimmermann 94]

Um usuário PGP gera seu próprio par de chave pública/privada. O certificado de chave pública está baseado em:

- Uma *string* ASCII arbitrária (normalmente: *"nome_usuário <e-mail_usuário>"*).
- Um número aleatório derivado da análise das características de datilografia durante a introdução de um texto também aleatório[65].

O usuário protege sua chave privada com uma *passphrase*, introduzida a cada vez que o PGP realiza o acesso a esta chave.

A *string* arbitrária consiste na forma com que as outras pessoas farão referência à sua chave pública. Normalmente, a fim de garantirem que são únicas, as pessoas utilizam o formato de endereçamento da Internet (o padrão X.400) para definirem suas *strings*. Não há, contudo, nenhum formato imposto. Porém, o PGP funciona bem sem impor nenhum modelo ou padrão.

No PEM, os certificados X.509 são gerados pelo usuário, por uma pessoa designada ou por um dispositivo de *hardware*. O certificado é assinado pelo CA do usuário e é armazenado em um servidor de diretório. Os detalhes do manuseio da chave privada dependem da implementação.

A chave pública do usuário é referenciada pelo uso de um nome X.500 (o DN). Ele é verdadeiramente único, e o seu padrão é projetado para ser descritivo e escalável (vide exemplo no final da seção 6.5). As chaves PEM incluem um período de validade. Após este período, a validade fica expirada.

As chaves públicas do PGP são tipicamente distribuídas por *e-mail*, boletins internos ou servidores de chaves na Internet. Para prevenir falsificações, as chaves são assinadas por terceiros. Cada usuário PGP decide em que terceiros ele confia. Deve-se obter as chaves públicas de pessoas em quem se confia por meios seguros.

Se for encontrada uma chave pública assinada por alguém em quem se confia, então esta chave será considerada confiável. O "introdutor de confiança" consiste na forma utilizada para a disseminação das chaves públicas. Mas esta propriedade não é transitiva[66] .

A versão do PGP para uma *Certification Authority* (CA) é um repositório de chaves públicas amplamente utilizado. A noção de hierarquia de repositórios é mencionada, mas não está desenvolvida. O PGP opera tipicamente em um ambiente com uma rede arbitrária de confiança, ao invés de uma estrita hierarquia.

As chaves públicas PEM podem ser distribuídas por qualquer meio, normalmente por correio eletrônico ou por buscas no diretório X.500. As chaves públicas são assinadas pela chave privada do CA (o emitente do certificado), podendo ser verificadas pela obtenção das correspondentes chaves públicas. Estas podem ser obtidas nos CA superiores na hierarquia.

No PGP, a distribuição de chaves é ampla e muitas vezes feita verbalmente. Assim, é impossível garantir a revogação de um certificado, se este for comprometido. Pode-se emitir um *certificado de revogação de chave*[67], mas não há garantia de que ele alcançará todas as pessoas que possuam a chave pública de um determinado usuário em seus respectivos chaveiros.

No PEM, existem as *Certificate Revocation Lists* (CRL). Elas são guardadas no diretório X.500 ou nas *mailboxes* mantidas pelos PCA. Portanto, os certificados podem ser rapidamente revogados, desde que a

---

[65] Este texto também entra na composição do número citado, por isso é importante que ele seja realmente aleatório.

[66] Se *Alice* confia em *Bob* e *Bob* confia em *Carol*, então *Alice* não tem que confiar em *Carol*.

[67] O certificado de revogação de chave PGP é assinado com a chave privada do usuário. Assim, se ele perdê-la, não há como revogar sua chave pública.

infra-estrutura de acesso ao banco de dados das CRL permaneça intacta. Assim, o CRL pode ser distribuído a todo usuário que mantém um *cache* local de certificados PEM.

7.5. Resultados da Comparação.

[Schneier 95]

O PEM parece muito bem organizado na teoria. Foi desenvolvido cuidadosamente pela Internet Society, constando de quatro RFCs. Mas na prática, carece de infra-estrutura e de diretórios públicos. Além disso, no PEM não é possível enviar mensagem sem assinatura, o que obriga a introdução de *passwords* a cada envio de mensagem. Embora seja altamente escalável, falta no PEM uma infra-estrutura para apoiar sua escalabilidade.

No PGP, um usuário não pode verificar a validade de todas as outras chaves PGP e nem mesmo confiar na maioria das pessoas usuárias do PGP que encontrar na Internet. Assim, ele não é muito escalável. Entretanto, não há razões técnicas para que as chaves públicas, na prática, não sejam administradas por autoridades confiáveis.

O PGP não tem a escalabilidade oferecida pelo PEM mas, em compensação, não requer nenhuma infra-estrutura. Duas pessoas podem começar a utilizá-lo para se comunicarem instantaneamente. Mais usuários poderão ser adicionados e a rede cresce rápida e facilmente (há um perigo aqui ... por exemplo, um chaveiro contendo milhares de chaves é algo muito pesado para a filosofia de gerenciamento do PGP).

Em resumo, as seguintes características fazem com que o PGP seja perfeito para o ambiente distribuído da Internet e menos ideal para organizações hierárquicas:

- falta de uma política explícita para certificados;
- falta de revogação garantida de certificados; e
- falta de um gerenciamento rigoroso do espaço de nomes (gerando o problema da escalabilidade).

Em suma, o PGP é um programa de segurança de correio eletrônico pragmático que evolui conforme seus limites são atingidos. O PEM é por vezes mais elegante, porém a infra-estrutura que exige o torna

muito pesado. Cada um possui suas próprias vantagens e limitações, seu próprio ambiente e seu próprio mercado. Decidir entre um ou outro, dependendo da situação, não é uma escolha fácil.

Foi baseado neste dilema que foi criado o sistema ProtegeMail, objeto de estudo desenvolvido neste trabalho, descrito na Parte III.

Assim, no sistema ProtegeMail, estão embutidos tanto o PEM (implementado pelo programa RIPEM) quanto o PGP. Portanto, o ProtegeMail, em sua versão 2.0, propicia maior flexibilidade e rapidez no trato de mensagens cifradas e/ou assinadas por usufruir das vantagens destes dois programas de segurança, além de oferecer várias outras facilidades.

Parte III

O SISTEMA PROTEGEMAIL

8. Seleção de Um Ambiente Alvo: o Editor EMACS

O sistema ProtegeMail é um pacote de programas feitos em Emacs Lisp que serve para implementar mecanismos de segurança nos sistemas de correio eletrônico existentes no editor Emacs ou no XEmacs . Tais sistemas são: o VM, o RMAIL, o MH e o GNUS.

Este capítulo pretende apresentar o editor de textos Emacs (seção 8.1) e expor os motivos para sua escolha como ambiente alvo do ProtegeMail (seção 8.2).

8.1. Descrição do Emacs.

[Gildea 93] [Gobbel 97] [Schneier 95]

O nome Emacs significa editor de macros (***E****ditor* ***MAC****ro****S***). Assim, nesta seção será apresentado, primeiro, o conceito de *macro*. Depois será feita uma descrição do editor Emacs.

Uma *macro* é uma maneira de se empacotar um grupo de comandos em um único comando, conhecido como comando de *macro*. Assim, uma *macro* destina-se a evitar que se repita uma mesma seqüência de ações de forma tediosa.

Pode-se aplicar o conceito de macros em todo tipo de contexto. Pode-se empacotar sob um único comando de *macro* vários comandos complexos ou uma série de comandos simples, como por exemplo a opção de negrito de um editor de textos. As *macros* também podem possuir parâmetros.

Atualmente, as macros do Emacs são na verdade programas completos, escritos em uma linguagem denominada LISP (de ***LIS****t* ***P****rocessor*).

LISP é uma linguagem muito simples, com quase nenhuma sintaxe. Os programas LISP consistem de uma série de *chamadas de função*, com um *nome de função* seguido de alguns argumentos, todos cercados de parênteses. Os argumentos também podem ser chamadas de funções, o que pode levar à existência de diversos parênteses em um único programa LISP.

O Emacs possui comandos para tornar os programas LISP mais fáceis de serem lidos, e para ajudar a localizar os pares de abertura/fechamento de parênteses. Os usuários podem utilizar o LISP para criar suas próprias extensões para o Emacs. O sistema ProtegeMail foi construído desta forma, como será apresentado no Capítulo 9.

Usualmente descrito como um editor de textos Unix, o Emacs é bem mais do que isto: trata-se de um sistema de janelas orientado a texto. Diferentemente de outros editores de texto Unix, como o vi, ed e ex, não é preciso entrar nenhum comando especial para introduzir textos no Emacs. Por *default*, o que se digita já é introduzido como texto. São utilizadas seqüências de teclas de controle para ativar as funções de *macro*. Segue um breve histórico do Emacs.

Um dos primeiros editores de textos para computadores foi o TECO (*Tape Editor and Corrector*). O TECO possuía uma característica de *macro*, e as pessoas começaram a escrever *macros* cada vez mais

complicadas com o passar dos anos. Um dos pacotes de *macro* mais interessantes que surgiram para o TECO, foi o Emacs, escrito por Richard Stallman.

O Emacs percorreu um longo caminho desde então. A versão corrente do GNU Emacs descende diretamente da pilha de macros escrita na metade da década de 70, e foi escrita também por Richard Stallman, que continua trabalhando neste sistema.

Os comandos Emacs mais básicos são caracteres únicos. Porém, os comandos que são utilizados com menos freqüência são chamados pela digitação de *um caracter de prefixo* seguido de um ou mais caracteres. Há dois caracteres de prefixo principais: *o Control-x* e o *Control-c*. Os comandos *Control-x* são utilizados em todos os contextos, e os comandos *Control-c* são específicos para subprogramas dentro do Emacs. Assim, por exemplo, as funções do sistema ProtegeMail são iniciadas por esta última seqüência de caracteres de prefixo.

Há também aqueles conhecidos como comandos *Meta-x*. Estes são chamados pela digitação de uma tecla *Meta*[68] seguida do caracter *x* e do nome do comando, que consiste em uma *string* descritiva. Todos os comandos Emacs podem ser chamados como comandos *Meta-x*, mesmo os mais simples. Por exemplo, *M-x forward-char* equivale a *C-f* [69].

O Emacs possui um excelente *tutorial* construído em seu interior. Para executá-lo e seguí-lo passo a passo, basta digitar *C-h t*. Outros tipos de comandos de ajuda do Emacs encontram-se descritos na tabela a seguir (Tabela 8.1).

[68] Alguns comandos utilizam teclas conhecidas como *Meta Keys*. Teclas como *Shift, Control, Command* e *Option* são conhecidas como *teclas modificadoras*. *Meta* era uma tecla modificadora existente nos terminais do laboratório onde o Emacs foi inventado. Como nem todo teclado possui uma *Meta Key*, pode-se digitar a tecla *Escape* (Esc) em seu lugar.

[69] Nos comandos Emacs, a tecla *Meta* é normalmente abreviada para *M* e a tecla *Control* para *C*.

Comando	*Descrição*
C-h a	Encontra um comando contendo um nome (palavra-chave) em particular. (*Command apropos*).
C-h k	Descreve o que faz a digitação de uma tecla de comando. (*Key help*).
C-h w	Descreve que tecla digitar para se obter um comando (*Where is*).
C-h i	Ativa o leitor da documentação de informação do Emacs (*Information*).
C-h ?	Explica as várias opções do comando *Help*.

Tabela 8.1 - Comandos de Ajuda do Emacs.

8.2. Motivos para a escolha do ambiente alvo Emacs.

[Gildea 93] [Gobbel 97] [Schneier 95]

Há programas no Emacs para se fazer quase tudo que se pode fazer no Unix. Além dos subsistemas de leitura de *e-mail* mencionados, há subsistemas no Emacs para processamento de textos, manutenção de uma agenda, edição de quase todas as linguagens de programação, navegação nas páginas Web, leitura de *netnews*, é só o usuário escolher. Alguns destes pacotes são melhores do que outros, mas a maioria dos programas são muito bem escritos[70].

Este fato, aliado à sua popularidade e à facilidade de construção de *macros*, já forneceria motivos suficientes para a escolha do Emacs como ambiente de desenvolvimento do sistema ProtegeMail. Mas podem ser acrescentadas mais três características que justificam esta opção do Emacs como ambiente alvo para o desenvolvimento de uma ferramenta de proteção de correio eletrônico.

Desta maneira, o Emacs é um sistema:

[70] Segue uma sugestão: *C-h p* encontra pacotes Emacs por palavra-chave.

- **Simples**. Há literalmente centenas de comandos no Emacs, mas pode-se utilizá-lo mesmo conhecendo somente alguns deles. Seus criadores trabalharam bastante para fazer com que os comandos seguissem um esquema regular e previsível, facilitando as suposições sobre que comandos farão o quê, no caso dos comandos mais avançados.

  Como os comandos Emacs são muito curtos, é fácil acidentalmente digitar algumas teclas e fazer com que o Emacs entre em um estado não desejado. A seqüência *Control-g* é bastante segura e sai de quase todo estado deste tipo. Além disso, o Emacs possui múltiplos níveis de comandos *undo* (comandos para *desfazer* uma ação realizada), e mantém 20.000 caracteres de informação disponível para serem desfeitos[71]. O comando desfazer pode ser acionado por:

  Control-underscore(_);

  Control-slash(/); ou

  Control-x u.

  O Emacs possui várias facilidades de ajuda embutidas, de forma a se poder encontrar quase toda a informação necessária sem precisar sair de seu ambiente (conforme observado nos exemplos da Tabela 8.1).

- **Universal.** O Emacs funciona bem em qualquer tipo de terminal, do mais simples ao mais sofisticado. Mesmo que se tenha apenas um terminal simples baseado em caracteres, pode-se utilizar todos os subsistemas do Emacs.

  Mas se tiver um terminal com um *mouse*, pode-se utilizar uma *interface* mais interessante, repleta de menus, cores e ícones gráficos, como é o caso do *XEmacs*, uma versão gráfica do Emacs. Ou como no caso do *NTEmacs*, uma versão do Emacs para o Windows NT.

  Assim, pode-se utilizar o Emacs, por exemplo, para leitura de *e-mails* no ambiente de trabalho, baseado em plataforma Unix; em casa, no PC, pode-se executá-lo para ler mensagens em ambiente Linux. Ou mesmo ler mensagens eletrônicas durante uma viagem internacional, utilizando o Emacs em sistemas Windows NT ou Macintosh. E tudo isto sempre com a mesma facilidade.

[71] Ou muito mais, desde que se informe a ele a quantidade desejada.

- **Completo.** O Emacs tem tudo. No Emacs, como já apresentado, há programas para leitura de *e-mail*, e também para percorrer diretórios de arquivos (mesmo pertencentes a outras máquinas diferentes da que se está executando) e navegar na WWW (*World Wide Web*). Ele possui jogos, facilidades especiais para programas de edição, execução e de rastreamento de erros, processadores de texto e muito mais.

Além dos motivos acima expostos, convém salientar que o sistema ProtegeMail baseou-se no sistema Mailcrypt, descrito na seção 9.1 deste trabalho, já desenvolvido previamente no ambiente Emacs em linguagem LISP.

9. Descrição do Sistema ProtegeMail

O ProtegeMail é um programa criado para implementar mecanismos de segurança nos subsistemas de Correio Eletrônico do editor Emacs ou do XEmacs (vide Capítulo 8). Tais subsistemas são: o VM, o RMAIL, o MH-E e o GNUS (este último é, na verdade, um sistema de manuseio de *netnews*). O ProtegeMail é carregado automaticamente quando da ativação de um destes módulos.

Construído na linguagem Emacs LISP, o ProtegeMail consiste em uma extensão do editor Emacs, agindo como uma *interface* modular para chamar funções criptográficas para: ciframento, deciframento, assinatura de mensagens, busca, geração e absorção de chaves, entre diversas outras existentes nos programas PGP e RIPEM (*Pretty Good Privacy* e *Riordan Privacy Enhanced Mail*, descritos, respectivamente, nos Capítulos 5 e 6 deste trabalho). Ele é ativado em conjunto com os subsistemas de correio eletrônico existentes no Emacs.

A figura, a seguir, demonstra a posição do sistema ProtegeMail no modelo geral de sistema de correio eletrônico utilizado na Internet. Ela descreve, portanto, sua posição em relação aos subsistemas de *e-*

mail do Emacs (ou seja, os UA), os programas PGP e RIPEM e o Sistema de Transferência de Mensagens (STM), formado pelo conjunto de MTAs[72] (Figura 9.1).

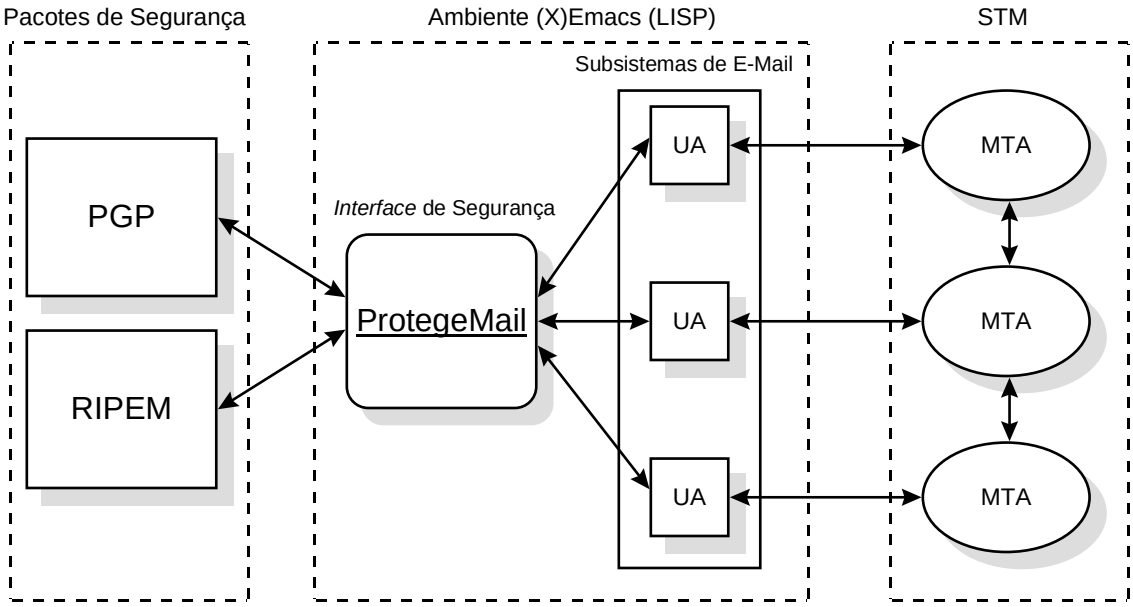


Figura 9.1 - O ProtegeMail no Modelo de Correio Eletrônico da Internet.

O usuário pode optar pelo pacote criptográfico desejado - PGP ou RIPEM, para proteção de seus *e-mails* ou *netnews* enviados e/ou recebidos. O ProtegeMail, em sua versão 2.0, torna disponível esta opção tanto no modo de leitura de *e-mails* quanto no modo de escrita. Uma vez estando na memória, ele pode ser acionado através de:

- um menu disponível na barra de menus principal do (X)Emacs, menu *pull down* denominado *ProtegeMail*;
- um aperto no botão direito do *mouse*, em cima de uma mensagem a ser lida ou que está sendo escrita, ativando o menu *ProtegeMail*; ou
- uso da seqüência de caracteres de prefixos *C-c/* (tecla *Control* junto com a tecla *"c"*, soltando-as e teclando a barra *"/"*), seguido de uma letra que varia conforme a função desejada.

[72] Um exemplo de MTA consiste no programa *Sendmail*, anteriormente descrito.

O ProtegeMail foi baseado no software Mailcrypt-3.4, de Patric LoPresti, que provê *interface* entre o Emacs e o PGP, descrito na seção 9.1 deste capítulo. A seção 9.2 descreve as principais funções do ProtegeMail versão 2.0. A seção 9.3 discorre sobre a segurança que ele provê ao subsistema de correio eletrônico Emacs utilizado. Finalmente, a seção 9.4 realiza uma avaliação do sistema ProtegeMail quanto ao atendimento às exigências de um bom sistema de proteção de *e-mail*.

9.1. O Sistema Mailcrypt

[LoPresti 95] [Pagliusi 97]

O sistema Mailcrypt consiste em um pacote feito em Emacs Lisp que provê aos *e-mails* e *netnews* uma *interface* para algumas funções criptográficas do PGP. Assim, o ciframento passa a ser uma parte integrante do ambiente de manuseio de mensagens.

Este sistema foi desenvolvido por Patrick J. LoPresti e Jin Choi, do MIT, em 1995, tendo sido inspirado no programa pgp.el, de Gray Watson. Foi projetado para utilização com o GNU Emacs versão 19. Trata-se de um *software* livre e gratuito, podendo ser redistribuído e/ou modificado sob os termos da *GNU General Public License*, como publicado pela Free Software Foundation, de Cambridge, EUA.

O Mailcrypt exige que se tenha instalado previamente o PGP e que se esteja familiarizado com suas funções básicas. Embora possa ser utilizado para o processamento de dados em *buffers* arbitrários do Emacs, ele é mais útil em conjunção com os subsistemas de manuseio de *e-mails* e *netnews*. Provê suporte especializado para o RMAIL, o VM, o MH-E e o GNUS.

Todos os comandos Mailcrypt são ativados por seqüências de três caracteres, começando com os caracteres de prefixos *C-c/*. Pode-se obter um resumo dos comandos disponíveis teclando-se a seqüência *C-h m* . Suas quatro operações mais comuns são descritas na tabela a seguir (Tabela 9.1).

Comando	***Descrição***
C-c / e	Cifra uma mensagem PGP utilizando a(s) chave(s) pública(s) do(s) recebedor(es).
C-c / d	Decifra uma mensagem PGP utilizando a chave privada do usuário.
C-c / s	Assina uma mensagem em claro utilizando a chave privada do usuário.
C-c / v	Verifica a assinatura de uma mensagem assinada em claro utilizando a chave pública do emissor.

Tabela 9.1 - Principais Operações do Mailcrypt.

O Mailcrypt serviu de base para o sistema ProtegeMail. Suas funções de i*nterface* com o pacote PGP (incluindo a busca de chaves públicas e o suporte para *remailers* utilizando o PGP) são comandos de *macro* em Emacs Lisp e, desta forma, puderam ser integralmente aproveitadas para o sistema ProtegeMail, tendo sido inclusas em seus menus *pull down*.

Nos menus do ProtegeMail, para facilitar o *gerenciamento das chaves* pelo usuário, foram acrescentadas *3 (três) novas* chamadas de funções do PGP, inexistentes no Mailcrypt:

- "Mostra Lista de Usuários PGP (*C-c / t*)";
- "Avalia Lista de Usuários PGP (*C-c / h*)"; e
- "Impressão Digital de Usuários PGP (*C-c / w*)".

Além disso, o ProtegeMail versão 2.0 também realiza a *interface* com outro pacote de segurança de correio eletrônico: o RIPEM. Assim, seus menus de leitura e de escrita de *e-mails* e de *netnews* incluem mais *8 (oito)* novas funções criptográficas, destinadas à *interface* com o RIPEM.

Ao contrário do Mailcrypt, limitado ao PGP, o ProtegeMail, portanto, permite ao usuário usufruir das vantagens de dois programas de segurança, conforme sua escolha, seja na escrita ou na leitura de uma mensagem: o PGP e o RIPEM.

Como descrito na próxima seção (9.2), todas as funções do ProtegeMail constituem parte integrante do ambiente de manuseio de mensagens, propiciando maior flexibilidade e rapidez no trato de mensagens cifradas e/ou assinadas.

9.2. Principais Funções do ProtegeMail

[Pagliusi 97]

Todas as funções do ProtegeMail são (por *default*) ativadas através de menu *pull down* e também pela seqüência de três caracteres de prefixo *C-c/*. O ProtegeMail opera proporcionando duas maneiras de realizar *interface* com as funções criptográficas (do PGP ou do RIPEM): o *modo de leitura* (ativando a função *pm-modo-leitura*) e o *modo de escrita* (ativando a função *pm-modo-escrita*).

O *modo de leitura* propicia a *interface* de segurança para o processamento de mensagens criptográficas recebidas pelos subsistemas de *e-mail* do Emacs. Por sua vez, o *modo de escrita* propicia a *interface* para o processamento de mensagens que estão prontas para ser enviadas pelos subsistemas, porém com o texto ainda em claro e/ou não assinado.

Estes dois modos ficam disponíveis ao usuário pelo menu *ProtegeMail*, situado na *barra principal* de menus do (X) Emacs. Como já exposto, este menu também pode ser ativado pelo botão direito do *mouse*, desde que um dos seus modos tenha sido previamente carregado na memória [73].

O menu *ProtegeMail* é carregado automaticamente, quando da ativação de um dos subsistemas de *e-mail* citados. Este menu também pode ser carregado manualmente na memória, através de uma das seguintes seqüências de caracteres, conforme o modo desejado pelo usuário:

- *M-x (Tecla "Esc" seguida da tecla "x") pm-modo-escrita*[74]*;* ou
- *M-x pm-modo-leitura*[75] .

[73] Esta característica é útil, pois existe a possibilidade de uso das funções do ProtegeMail em situações não relacionadas a correio eletrônico. Como no caso de assinatura de arquivos de texto contidos em *buffers* arbitrários.

[74] Para facilitar a ativação do modo de escrita, pode-se utilizar a seqüência: *C-c/j*.

[75] Para facilitar a ativação do modo de leitura, pode-se utilizar a seqüência: *C-c/y*.

Os dois modos ativam as funções do PGP (pelo submenu "*Utilizando PGP*") e do RIPEM (pelo submenu "*Utilizando PEM*") relacionadas, respectivamente, com a leitura ou com a escrita de uma mensagem. Contudo, há algumas funções que ficam sempre disponíveis em ambos os modos.

As funções que o ProtegeMail 2.0 engloba encontram-se nas três tabelas a seguir. Estão divididas conforme a respectiva associação com o modo de leitura (Tabela 9.2), com o modo de escrita (Tabela 9.3) ou com ambos os modos de ativação existentes (Tabela 9.4).

Submenu	***Função***	***Caracteres de Prefixo***
Utilizando PGP	Deciframento de Mensagem.	*C-c/d*
	Verificação de Assinatura.	*C-c/v*
	Absorção de Chaves.	*C-c/a*
Utilizando PEM	Deciframento de Mensagem ou Verificação de Assinatura.	*C-c/i*
	Validação do Emissor e Absorção de Sua Chave Pública.	*C-c/l*

Tabela 9.2 - Funções ProtegeMail Somente do Modo de Leitura.

O anexo "A" deste trabalho - "*Guia do ProtegeMail 2.0"* descreve com mais detalhes todas as funções mencionadas nesta seção. Também contém instruções para instalação (no Emacs ou XEmacs, no sistema Unix / Linux) e operação do sistema ProtegeMail 2.0. Este sistema encontra-se disponível para *download*, por http, no seguinte endereço WWW: <*http://www.dcc.unicamp.br/~pagliusi/mo1000.html*>.

Submenu	***Função***	***Caracteres de Prefixo***
Utilizando PGP	Ciframento de Mensagem.	*C-c/e*
	Assinatura de Mensagem.	*C-c/s*
	Inserção de Chave Pública.	*C-c/x*
	Ciframento para *Remailer(s).*	*C-c/r*
	Inserção de Pseudônimo.	*C-c/p*
	Inserção de Bloco de Resposta.	*C-c/b*
Utilizando PEM	Ciframento de Mensagem.	*C-c/c*
	Assinatura de Mensagem - Texto Codificado (MIC-ONLY).	*C-c/o*
	Assinatura de Mensagem - Texto Claro (MIC-CLEAR).	*C-c/n*

Tabela 9.3 - Funções ProtegeMail Somente do Modo de Escrita.

Submenu	***Função***	***Caracteres de Prefixo***
Utilizando PGP	Busca de Chaves.	*C-c/k*
	Esquecimento de *Passphrase(s).*	*C-c/f*
	Listagem de Usuários PGP.	*C-c/t*
	Avaliação da Lista de Usuários PGP.	*C-c/h*
	Impressão Digital de Usuários PGP.	*C-c/w*
Utilizando PEM	Geração de Par de Chaves Pública/Privada.	*C-c/g*
	Mudança de *Password.*	*C-c/m*
	Listagem de Usuários PEM Catalogados.	*C-c/u*

Tabela 9.4 - Funções ProtegeMail Disponíveis em Ambos os Modos.

9.3. Segurança Provida pelo ProtegeMail

[LoPresti 95] [Pagliusi 97] [Schneier 95]

Pode-se afirmar que a segurança provida pelo sistema ProtegeMail possui seu limite superior coincidente com a segurança provida pelos programas PGP e RIPEM. E que estes dois programas possuem seus limites superiores de segurança coincidentes com a segurança provida pelos algoritmos convencionais e de chave pública que utilizam, todos já discutidos na seção 4.2 deste trabalho.

O PGP e o RIPEM proporcionam um boa segurança quando utilizados corretamente. Para aproveitar bem a segurança que oferecem, é importante a familiarização com suas funções. Isto pode ser obtido pela leitura das instruções contidas em seus respectivos guias de usuários [Zimmermann 94] e [Riordan 93] [76].

Todas as operações do ProtegeMail colocam as saídas do PGP e do RIPEM no *buffer *ProtegeMail temp**. É interessante verificá-lo ocasionalmente, para ler as mensagens de *status* e de avisos importantes.

A segurança oferecida pelo ProtegeMail está fundamentada na *password* e na chave privada RIPEM, e na *passphrase* e na chave privada PGP do usuário. Por isto, nunca é demais frisar que a *password* ou a *passphrase* deve ser mantida em sigilo absoluto. Trata-se de uma questão de identidade e não de confiança, como já exposto na seção 4.3.

Além disso, é recomendável armazenar os arquivos com a chave privada RIPEM e com a chave privada PGP em local seguro, mesmo levando-se em conta que ambos ficam protegidos por senha. Pode-se guardá-los, inclusive, em um disquete, caso o usuário esteja utilizando o ProtegeMail e o Emacs em um ambiente de rede desprotegido.

Deve-se tomar cuidado com uma facilidade oferecida pelo ProtegeMail, herdada do Mailcrypt: o armazenamento da *passphrase* PGP durante um certo período de tempo. Este armazenamento tem, como propósito, evitar que o usuário precise introduzir sua *passphrase* mais de uma vez quando assina ou decifra mensagens PGP em série. O período de tempo padrão é de 300 segundos (5 minutos), mas este valor pode ser modificado, conforme instruções contidas no Anexo "A" deste trabalho.

[76] Estes guias são distribuídos junto com o pacote de instalação do ProtegeMail 2.0.

É uma característica bastante confortável, porém deve-se tomar a precaução de, ao utilizá-la, procurar sempre chamar a função [*Esquece Passphrase(s)*], caso o usuário precise se afastar da máquina durante algum tempo, deixando nela o Emacs ativo.

Caso contrário, qualquer pessoa fica "autorizada" a utilizar durante algum tempo a chave privada PGP do usuário. Ou, na pior situação, pode até descobrir sua p*assphrase.*

Se o ambiente de trabalho do usuário ProtegeMail for inseguro, sugere-se alterar o valor <*tempo*> para *nil* ou 0. O valor *nil* ou 0 desativa por completo o armazenamento temporário da *Passphrase* PGP.

9.3.1. Remailers, Blocos de Resposta e Pseudônimos.

[LoPresti 95] [Pagliusi 97]

Outra facilidade do ProtegeMail, também herdada do Mailcrypt, com que se deve tomar cuidado consiste na utilização dos *remailers* disponíveis na Internet. Os *remailers* são programas anônimos que recebem um *e-mail*, retiram dele todas as informações de cabeçalho que identificam o emissor da mensagem, e encaminham o *e-mail* para o recebedor designado.

São muito úteis para prevenir a análise de tráfego, discutida no item 2.1.1 deste trabalho. Porém, este esquema simples torna-se inseguro se o *remailer* escolhido pelo usuário estiver comprometido. Deste modo, quem controlar o *remailer*, terá acesso à identidade dos emissores e dos recebedores dos *e-mails* que nele circularem.

Por isto, sugere-se o emprego de cadeias de *remailers* que enviam mensagens cifradas, como observado na figura a seguir (Figura 9.2).

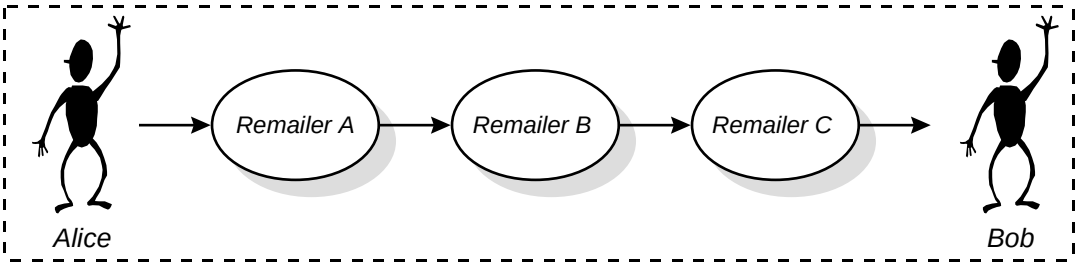


Figura 9.2 - Uma Cadeia de *Remailers* para o Envio de *E-Mail.*

Na figura, *Alice* deseja enviar uma mensagem anônima e cifrada para *Bob*, através da cadeia formada pelos *remailers A*, *B* e *C*. Então, ela escreve a mensagem, cifrando-a com a chave pública de *Bob*. Em seguida, cifra o resultado obtido (inclusive o cabeçalho que indica que *Bob* é o recebedor) utilizando a chave pública do *remailer C*. Então, ela cifra este último resultado fazendo uso da chave pública do *remailer B*. Em seguida, ela cifra o resultado obtido, utilizando a chave pública do *remailer A*, e envia sua mensagem para *A*.

Quando *A* recebe a mensagem, ele a decifra com sua chave privada de modo a produzir algo cifrado para *B*, verifica que *B* é o próximo *remailer* destinatário, retira fora do cabeçalho a informação de que a mensagem veio de *Alice* e a envia para *B*. *B*, por sua vez, repete o mesmo procedimento de *A*, retirando fora a informação de que a mensagem veio de *A* e a encaminha para *C*. *C* também decifra a mensagem com sua chave privada, descobre que o recebedor é *Bob*, retira fora a informação de que a mensagem veio de *B* e envia o resultado para *Bob*, que o decifra, finalmente, com sua chave privada.

Se a usuária *Alice* desejar, ela pode incluir um *bloco de resposta* cifrado na sua mensagem anônima, enviada a *Bob*. Este bloco de resposta define um *remailer* ou uma cadeia de *remailers* que *Bob* pode utilizar para responder a *Alice*, mesmo sem saber para quem responde. Basta que o último *remailer* da cadeia de *remailers* escolhida (se houver uma) saiba quem é o recebedor final.

Alice também pode estipular um *pseudônimo* para utilização na assinatura de seus *e-mails* anônimos, para que *Bob* saiba que os *e-mails* recebidos, ainda que anônimos, sempre são oriundos da mesma pessoa.

O ProtegeMail, através do seu submenu “Utilizando PGP”, inclui facilidades para o envio de mensagem por *remailers*, para a definição de cadeias de *remailers*, para a geração de blocos de resposta e para o uso de pseudônimos pelo usuário. A figura a seguir ressalta a ordem natural destas operações, no caso de se desejar realizar todas elas, desde a composição da mensagem até o seu envio (Figura 9.3).

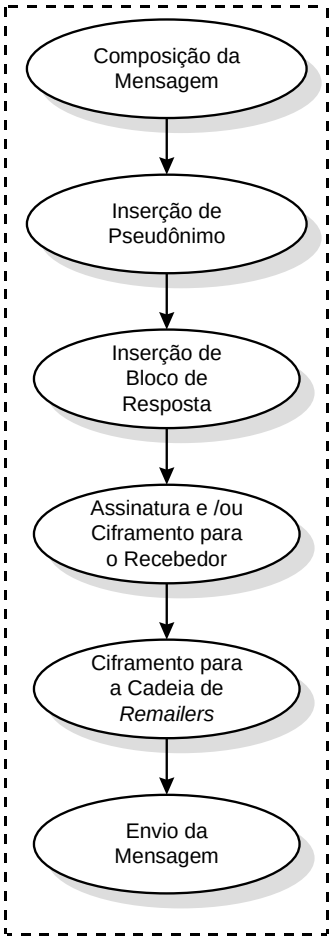


Figura 9.3 - Ordem Natural das Operações Envolvendo *Remailers*.

9.3.2. Remoção de Arquivos, Cavalos de Tróia e Efeito Tempest.

[Pagliusi 97] [Schneier 95]

Quando o usuário for apagar arquivos que não serão mais utilizados pelo Emacs ou pelo ProtegeMail, em especial arquivos contendo mensagens em claro, ele deve se lembrar que, em geral, o que está sendo apagado é apenas o nome do arquivo do diretório. Os bits que, na verdade, compõem o arquivo permanecem intactos no disco rígido ou, no máximo, sofrem alguma modificação trivial. Há diversos programas que podem recuperá-los, como os utilitários *Norton* (para *PCs* e *Macs*).

Para apagá-los de forma que não possam ser mais lidos por tais pacotes, deve-se escrever algo fisicamente múltiplas vezes sobre todos os seus bits. Há diversos utilitários que efetuam esta operação.

Mesmo assim, experimentos recentes feitos no NIST com microscópicos eletrônicos de alta precisão sugerem que esta operação pode não ser suficiente. Os resultados apontam que o único jeito de se apagar mídia magnética seria destruí-la fisicamente[77].

Além dos cuidados na remoção de arquivos, deve-se ater também para a possibilidade do PGP ou do PEM instalados na máquina do usuário serem, na verdade, *Cavalos de Tróia* : uma cópia levemente modificada do programa de segurança. Esta cópia se parece e se comporta exatamente como o programa original, mas possui um *trapdoor* em seu interior. Ela pode ser colocada em um *site* aparentemente inocente destinado a *download* dos arquivos de instalação do PGP ou do PEM.

Talvez esta nova versão do PGP ou do PEM possa, sorrateiramente, despachar uma cópia em claro de toda mensagem que for cifrada, para um determinado endereço pertencente a *Eve*. Ou possa conter uma rotina de geração de chaves modificada, que gera somente 1000 chaves de sessão possíveis. Ou mesmo utilizar um algoritmo de ciframento ou de gerenciamento de chaves enfraquecido a ponto de tornar mais fácil a criptoanálise do intruso *Eve*. Sendo assim, o ProtegeMail fica também comprometido.

Outro ponto a ser levado em conta é que o computador do usuário irradia dados. Ele o faz através do monitor; ele o faz através da linha de força de corrente alternada. Com o equipamento adequado, um intruso sofisticado pode ler o que o usuário está digitando em seu computador a centenas de metros de distância; talvez de uma caminhonete estacionada fora do seu escritório ou residência. Não é necessário dizer que a criptografia não parece ser muito boa se o texto claro ficar disponível desta maneira.

Para prevenir este tipo de ataque, foi desenvolvido um padrão militar americano conhecido como *Tempest*. Normalmente, os computadores não são construídos para evitar o efeito *Tempest*, podendo facilmente ser monitorados. Assim, os militares projetam seus equipamentos seguindo o padrão *Tempest* - blindados especialmente para não irradiarem.

Uma solução mais barata é utilizar um computador *laptop* funcionando com baterias. Assim, não haverá emissão de realimentação das linhas de energia elétrica; e a quantidade de energia utilizada para alimentar o *display* sendo consumida pelo *laptop* provê um sinal muito fraco para permitir que seja monitorado à distância.

[77] Há alguns órgãos de inteligência americanos que recomendam ao seu público interno destruir seus discos rígidos passando-os em um esmeril. E derretendo, depois, as sobras.

Este é um cenário realmente paranóico, mas ilustra um ponto importante: no final das contas, não se consegue obter nenhuma segurança totalmente garantida com pacotes de segurança de correio eletrônico, seja com o ProtegeMail ou com qualquer outro. A menos que o computador esteja trancado em um lugar blindado que ninguém tenha acesso, ele estará em condições de aparentar ser um computador seguro, mas que na verdade não é.

9.4. Avaliação do Sistema ProtegeMail.

[Pagliusi 97] [Schneier 95]

O ProtegeMail é um programa robusto e seguro. Mas não é somente isto que se espera de um sistema de proteção de *e-mail*. Como exposto na seção 2.2, as exigências que o ProtegeMail precisa atender para ser um bom sistema de segurança de correio eletrônico podem ser resumidas em:

- **Segurança.** O sistema ProtegeMail atende à exigência da segurança, quando utilizado corretamente, pois utiliza pacotes de proteção de correio eletrônico baseados em algoritmos amplamente empregados e analisados pela comunidade de criptografia. Porém, como já mencionado, é mais fácil quebrar um programa de segurança do que provar que ele não pode ser quebrado.

- **Flexibilidade.** O sistema ProtegeMail permite aos usuários o envio de mensagens cifradas não assinadas (pelo emprego do PGP), assinadas não cifradas (pelo PGP e PEM) e mensagens assinadas e cifradas (pelo PGP e PEM). Ele permite o ciframento das mensagens armazenadas e possibilita o envio de mensagens tanto para um único quanto para múltiplos recebedores.

  Além disso, o ProtegeMail encontra-se disponível para a plataforma Unix / Linux, possibilitando sua operação em diversos subsistemas de correio eletrônico do Emacs ou XEmacs. E encontra-se em fase de testes na plataforma MS-DOS, através da sua implementação no NTEmacs (que também utiliza a linguagem interpretada LISP).

  Ele permite também que pessoas utilizando um dado sistema operacional sejam capazes de enviar, via Internet, mensagens seguras para uma pessoa utilizando um sistema distinto. Em resumo, pode-se afirmar que o ProtegeMail é um sistema que atende à exigência da flexibilidade.

- **Adaptabilidade.** O ProtegeMail é um sistema modular. Assim, por permitir o emprego de uma variedade de diferentes algoritmos, possui uma boa adaptabilidade. Usuários que confiam mais no DES, podem utilizá-lo se optarem pelo submenu do PEM. Outros que preferem o IDEA, podem empregá-lo caso optem pelo submenu do PGP.

  Mesmo aqueles que preferem outro tipo de algoritmo, como o Triplo DES ou um algoritmo de ciframento proprietário, podem passar a utilizá-lo através do ProtegeMail, bastando que instalem uma implementação do PEM que o tenha como opção de ciframento de mensagem. O mesmo vale para aqueles que desejam outra opção de algoritmo de gerenciamento de chaves ou de assinatura digital, diferente do RSA.

- **Boa *interface* com o usuário.** Como já foi citado, obter uma boa *interface* com o usuário é mais difícil de se conseguir do que uma boa segurança. O mundo da Internet está repleto de diferentes computadores, sistemas operacionais, programas de *e-mail* e usuários. Todas as pessoas têm suas próprias preferências, e é virtualmente impossível fazer um programa que trabalhe bem em todos os ambientes.

  Apesar disso, o ProtegeMail, através de seus menus *pull down* e de suas opções por seqüências de teclas, totalmente incorporado não somente a um, mas a vários sub-sistemas de *e-mail*, de um dos mais populares editores de texto utilizados em todo o mundo - o Emacs, procura atender a esta exigência. Deste modo, no ProtegeMail permanece transparente ao usuário uma série de operações de segurança realizadas pelo sistema. Somente são solicitadas ao usuário informações consideradas imprescindíveis.

  Mesmo assim, caso ele queira inteirar-se do andamento das operações, ele pode sempre consultar o *buffer *ProtegeMail temp**, como descrito no Anexo "A" deste trabalho, para ler as mensagens de *status* e de avisos importantes do PGP ou do RIPEM.

Parte IV

10. Conclusão

O correio eletrônico é um dos principais protagonistas do crescimento fantástico da Internet. E sua segurança tem sido um grande problema, desde que o *e-mail* foi utilizado pela primeira vez. A mensagem eletrônica pode sofrer uma série de ataques, que vão desde a escuta clandestina, o emprego de identidade falsa, o *Man-In-The-Middle*, a violação de mensagens e a inferência por criptoanálise até os ataques da meia-noite, do corte e cola e o método da força bruta. E o que é pior: muitos dos ataques ocorrem sem o conhecimento do emissor e do recebedor legítimo. Não há segurança alguma no serviço de *e-mail* oferecido pela Internet, onde as mensagens parecem circular como cartões postais; ou seja, ficam totalmente expostas a quem quiser lê-las ou falsificá-las.

A criptografia pode, então, servir como assinatura e envelope eletrônico para os inúmeros *e-mails* que trafegam diariamente pela Internet. O usuário que faz emprego dela em sua correspondência

eletrônica, mesmo para os assuntos atinentes ao seu cotidiano, livra-se da preocupação de ter sua privacidade invadida ou, na pior das hipóteses, de ter a sua identidade corrompida. Além disso, ela contribui para aumentar o problema da coleta por parte do intruso. Em um mar de *e-mails*, onde a maioria encontra-se cifrada, torna-se complexo identificar aqueles que são de seu real interesse e que compensam o esforço e o custo do ataque.

A solução padrão para a segurança em *e-mail* Internet, o PEM, com sua hierarquia de chaves e o PGP, com sua teia de confiança, a solução mais amplamente utilizada, possuem vantagens complementares, porém estes programas não são compatíveis um com o outro. E ambos possuem o inconveniente de serem externos ao programa de correio eletrônico.

Além disso, o comandos do PGP e do RIPEM, a implementação mais popular do padrão PEM, não são nada amigáveis ao usuário final; exigem o gasto de um tempo valioso, tanto para o correto aprendizado de seus comandos quanto para a constante importação e exportação de textos em claro e cifrado, do ambiente de correio eletrônico para o ambiente de criptografia externo, e vice-versa. Portanto, o desejável é uma solução que alie as vantagens destes dois pacotes de segurança, mas sem os inconvenientes que ambos apresentam.

Assim, o emprego de um sistema que ofereça ao usuário segurança na troca de *e-mails* via Internet, permitindo a opção pelo algoritmo criptográfico desejado e que faça parte do ambiente de correio eletrônico utilizado (com o que ele já esteja acostumado) representa uma solução vantajosa para o usuário.

Nos capítulos anteriores, foram discutidas as principais características de um sistema de correio eletrônico e a questão da sua segurança, bem como os principais protocolos e programas utilizados na troca de mensagens, com ênfase no SMTP e no Sendmail. Introduziu-se a criptografia, seus conceitos básicos, suas técnicas e seus principais algoritmos voltados para mensagens eletrônicas. Foram descritos e comparados os pacotes de segurança de *e-mail*, PGP e PEM, e analisadas suas características e vantagens complementares, a fim de colocá-los como opção no sistema de segurança desenvolvido.

Foram expostos os motivos da escolha do editor Emacs e de seus subsistemas de correio eletrônico como ambiente alvo deste sistema e, em seguida, feita a descrição do sistema de segurança **ProtegeMail**; através da apresentação do sistema que lhe serviu de base, seguida de suas principais funções, da descrição da segurança provida e de uma avaliação final, em face às exigências de um

sistema de *e-mail* seguro. O anexo "A", Guia do ProtegeMail 2.0 complementa a descrição deste sistema e o anexo "B" contém um relato cronológico de suas alterações desde seu lançamento.

O objetivo deste trabalho, descrito na seção 0.1, foi tratar o aspecto da segurança na troca de *e-mails*, via Internet, oferecendo uma solução flexível e modular, no nível de aplicação, que embute funções de segurança criptográfica *internas* ao ambiente de correio eletrônico do editor Emacs, com uma boa relação de custo/benefício para o seu usuário.

Acredita-se que o sistema ProtegeMail atende a todas as exigências mencionadas, garantindo aos seus usuários um ambiente seguro para troca de *e-mail* pela Internet, oferecendo uma boa relação de custo/benefício.

10.1. Sugestões de Pesquisa

Acredita-se que o objetivo proposto foi atingido e que o sistema ProtegeMail desenvolvido e disponível para *download*, pela Internet, possa vir a ser bastante utilizado. As diversas pessoas que se ofereceram para testá-lo já o incorporaram aos seus hábitos de manuseio de *e-mail*. Seguem algumas sugestões de pesquisa, que podem contribuir para maior segurança do ambiente de *e-mail* proposto:

- Inclusão, no ProtegeMail, de mais uma opção para a chamada de funções de um pacote de segurança de *e-mail* com a habilidade de manuseio de objetos MIME, como o PGP/MIME, o MOSS e o S/MIME.
- Migração do ProtegeMail para os ambientes operacionais Windows 95 e Windows NT, que também possuem interpretadores Emacs LISP e os mesmos subsistemas de correio eletrônico do Emacs utilizado na plataforma Unix (e Linux).
- Acréscimo de suporte para um completo gerenciamento de chaves na opção pelo PGP incluindo, por exemplo, a geração de chaves, existente na opção pelo RIPEM.
- Inclusão de uma nova opção para chamar funções de um pacote de segurança de *e-mail* com uma capacidade de recebimento assinado - ou seja, não repúdio do recebimento.

- Inclusão de uma nova opção, de interesse militar ou governamental, para chamar funções de um pacote de segurança de *e-mail* com a habilidade de classificar as mensagens - como ultra-secretas, por exemplo. Assim, um UA poderia rejeitar as mensagens, se os usuários não possuírem o credencial necessário ou a autorização apropriada para lê-las.

- Acréscimo de um novo submenu e de uma nova seqüência de teclas associadas a partir da opção de *Ciframento de Mensagem (C-c/c)* do menu *Utilizando PEM*, que permita ao usuário optar pelo Triplo DES ou pelo DES-CBC, quando for cifrar uma mensagem pelo RIPEM.

10.2. Autocrítica

O sistema ProtegeMail é um programa poderoso (Figura 10.1), mas não é uma *interface* completa do PGP e do RIPEM. Talvez alguma versão futura consiga este feito. Enquanto ela não surge, é necessário o emprego da *interface* para a linha de comando existente no (X) Emacs para algumas operações. Os ítens seguintes trazem algumas observações sobre funções que a versão atual (2.0) não implementa:

- **Completo Gerenciamento de Chaves.** O suporte de gerenciamento de chaves na opção pelo PGP do ProtegeMail é limitado à adição e à extração de chaves dos chaveiros. Ele não provê suporte para geração, remoção ou revogação de chaves, nem para edição de parâmetros de confiança e de identificação ou para certificação de chaves. Ele também ignora os avisos do PGP quando o usuário utiliza uma chave que não está totalmente certificada. O gerenciamento de chaves na opção pelo RIPEM oferece alguns recursos, como a geração do par de chaves pública/privada e a mudança de *password*. Porém, ele fica limitado pelo próprio fato do RIPEM não oferecer a certificação para autenticação de chaves prevista na especificação X.509.

- **Ciframento de Chaves com Criptografia Convencional.** Embora tanto o PGP quanto o padrão PEM ofereçam a possibilidade do uso de criptografia convencional

(ou seja, não baseada em chave pública) para ciframento e deciframento de chaves de sessão, o ProtegeMail implementa apenas o deciframento, em suas duas opções de pacotes de segurança.

- **Assinaturas Destacadas.** O ProtegeMail não permite a criação nem a verificação de assinaturas PGP ou RIPEM destacadas.

- **Proteção contra "Cavalos de Tróia".** Um intruso *Eve* pode facilmente substituir o PGP ou o RIPEM por um programa que não seja distinguível do original, mas que contenha uma fraqueza dissimulada em seu interior para favorecer a quebra da segurança na troca de *e-mails* entre *Alice* e *Bob*. Por exemplo: um programa que gere somente 1000 chaves de sessão. O ProtegeMail não prevê ataques deste tipo. O usuário precisa se precaver, efetuando, por exemplo, *downloads* do PGP e do RIPEM a partir de endereços WWW ou *ftp* confiáveis. Pode também utilizar o próprio PGP para assinar os códigos-fonte dos programas copiados, permitindo uma posterior conferência da assinatura destacada.

- **Proteção contra Substituição Ilícita de Chaves Públicas.** Nada impede que *Eve* gere uma chave pública falsa contendo a identificação de *Alice* ou de *Bob*, deixando-a disponível para consulta para personificar um ou outro. Como já mencionado, o ProtegeMail não efetua um completo gerenciamento das chaves utilizadas. Contudo, *Alice* pode ler os avisos do RIPEM ou do PGP quando utilizar uma chave que não esteja totalmente certificada, abrindo o *buffer *ProtegeMail temp**, como descrito na seção 9.3. Além disto, ela pode utilizar recursos como a obtenção da impressão digital da chave pública PGP de *Bob* e conferi-la por telefone (recurso oferecido pela opção *C-c/w* do ProtegeMail). Todavia, nada garante que do outro lado da linha não esteja um intruso *Eve* imitando a voz de *Bob* para confirmar a impressão digital falsa. Fraquezas como esta são "calcanhares de Aquiles" não apenas do ProtegeMail, mas inerentes aos sistemas de criptografia baseados em chave pública, como o PGP e o RIPEM.

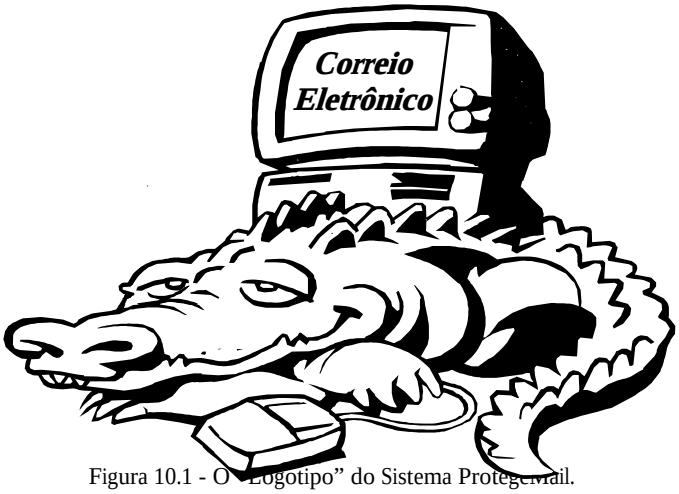


Figura 10.1 - O "Logotipo" do Sistema ProtegeMail.

Referências Bibliográficas

[Allman 95] Eric Allman: "*Sendmail Inside and Out*". Pangaea Reference Technologies, 9th Systems Administration Conference, Monterey, CA, EUA, 17 a 22 de Setembro de 1995.

[Bellovin 94] William R. Cheswick & Steven M. Bellovin: *"Firewalls and Internet Security"*. Addison-Wesley Publishing Company 1994, EUA, pp. 213.

[Brito 97] Manoel Francisco Brito: "*Seção Hipertexto*", revista Veja número 1499, 11 de Junho de 1997, pp. 15.

[Cavalcanti 96] Edjozane C. Cavalcanti, Walfredo C. Cirne Filho, Francisco V. Brasileiro: "*Introduzindo Segurança em E-Mail Internet*". Dissertação de Mestrado em Informática da UFPB, Centro de Ciências e Tecnologia, Dep. de Sist. e Computação, Lab. de Sistemas Distribuídos, Campina Grande - PB, 1996.

[Cavalcanti 97] Edjozane C. Cavalcanti, Walfredo C. Cirne Filho, Francisco V. Brasileiro: "*Introduzindo Segurança no Correio Eletrônico Internet*". ANAIS / XV Simpósio Brasileiro de Redes de Computadores - SBRC, São Carlos-SP, Maio de 1997.

[Chaves 96] Eduardo O. C. Chaves: "*Internet-se: Um Guia Prático*". People Brasil Informática Ltda., Campinas, 1996.

[Comer 95] Douglas E. Comer: "*Internetworking with TCP/IP*". Prentice Hall, New Jersey, 1995.

[Costales 94] Bryan Costales, Eric Allman, Neil Rickert: "*Sendmail*". O'Reilly & Associates Inc., 1994, Sebastopol, CA, EUA, pp. 3-16.

[Coulouris 94] G. Coulouris, J. Dollmore & T. Kindberg*: "Distributed Systems: Concepts and Design"*, 2a. edição, Addison-Wesley Publishing Co.,1994, pp. 495-502.

[Denning 82] D. E. R. Denning : *"Cryptography and Data Security*". Addison-Wesley, 1982.

[Diffie 76] W. Diffie & M.E. Hellman: *"New Directions in Cryptography"*. IEEE Transactions on Information Theory IT-22, Novembro de 1976, pp. 644-654.

[Diffie 88] W. Diffie: *"The First Ten Years of Public Key Cryptography"*. Proceedings of the IEEE, 76 (5), Maio de 1988, pp.560-577.

[Estrada 93] Susan Estrada: "*Connecting to The Internet*". O'Reilly & Associates

Inc., 1993, Sebastopol, CA, EUA, pp. 23 e 24.

[Franco 95] João Henrique de A. Franco: *"Sistemas Criptográficos de Chave Pública"*. Artigo "Tecnologias", Revista da Telebrás, Abril de 1995, pp. 6 a 9.

[Frey 94] Donnalyn Frey: *"!%@:: A Directory of Electronic Mail Addressing & Networks"*. O'Reilly & Associates Inc., 1994, Sebastopol, CA, EUA, pp. 1-19.

[Garfinkel 95] Simson Garfinkel: *"PGP Pretty Good Privacy"*, O'Reilly & Associates Inc., 1995.

[Gildea 93] Stephen Gildea: *"GNU Emacs Reference Card"*, Free Software Foundation, Inc., v2.0, para GNU Emacs versão 19 sobre Unix,<http://www.geek-girl.com/emacs/ refcard.txt>, 1993.

[Gobbel 97] Randy Gobbel: *"An Introduction to Emacs"*, CogSci 3 Emacs Lecture,<http://cogsci.ucsd.edu/~gobbel/Emacs. html>, 1997.

[Heikkinen 95] Jyrki Heikkinen: *"Secure Electronic Mail"*. ICL Personal Systems, 1995. TeamWARE Division / Enterprise Messaging Development. End.: *<http://www.tcm.hut.fi/Opinnot/Tik110.501/1995/Secure-email.html>*.

[Levien 96] Raph Levien: *"Protecting Internet E-Mail From Prying Eyes"*. University of California, Berkeley, Maio de 1996. Endereço: *<http://www.data.com/Tutorials/Protecting_ Internet_Email.html>*.

[Linden 76] T. A. Linden: "O*perating System Structures to Suport Security and Reliable Software*". Computing Surveys, vol.8, Dezembro de 1976, pp.409-445.

[LoPresti 95] Patrick J. LoPresti: "*Mailcrypt: An Emacs Interface to PGP "* Free Software Foundation, sistema Mailcrypt versão 3.4, Distribution, EUA, Outubro de 1995.

[Lucchesi 86] Cláudio Leonardo Lucchesi: "*Introdução à Criptografia Computacional"*. Editora Papirus / UNICAMP, 1986, pp.1-15.

[Menezes 93] Alfred Menezes: "*Elliptic Curve Public Key Cryptosystems*", Kluwer Academic Publishers, EUA, 1993.

[Nechvatal 92] J. Nechvatal: *"Public Key Cryptography"*. In Contemporary Cryptology: The Science of Information Integrity, IEEE Press, EUA, 1992, pp.177-288.

[Needham 78] R. M. Needham & M. D. Schroeder: *"Using Encryption for*

Authentication in Large Networks of Computers". Comm. of ACM, 21,12, Dezembro de 1978.

[Needham 87] R. M. Needham & M. D. Schroeder: "*Authentication Revisited"*. ACM Operating System Review 21 (1): 7, Janeiro de 1987.

[NSA C. O. CAPI Team 95] National Security Agency Cryptographic Application Program Interface Team: "*Security Service API: Cryptographic API Recommendation"*. 12 de Junho de 1995, pp. 1-25.

[O'Reilly 92] Tim O'Reilly e Grace Todino: "*Managing UUCP and USENET*". O'Reilly & Associates Inc., 1992.

[Pagliusi 97] P. S. Pagliusi, C. L. Lucchesi & L. E. Buzato: "*Guia do ProtegeMail 2.0 - Uma Interface (X)Emacs para o PGP e o RIPEM*", Distribuição, UNICAMP, Agosto de 1997, *<http://www.dcc.unicamp.br/~pagliusi/mo1000.html>*.

[Popek 79] G. J. Popek e C. S. Kline: "*Encryption and Secure Computer Networks"*. Computing Surveys, 11, 1979, pp.331-356.

[RFC 1123] Robert Braden: "*Requirements for Internet Hosts - Aplication and Suport*", Internet Engineering Task Force, Outubro de 1989.

[RFC 1321] Ron Rivest: "*The MD5 Message Digest Algorithm"*. MIT Laboratory for Computer Science and RSA Data Security Inc., Abril de 1992.

[RFC 1421] John Linn: "*Privacy Enhancement for Internet Electronic Mail: Part I - Message Encryption and Authentication Procedures"*. IAB IRTF PSRG, IETF PEM WG, Fevereiro de 1993.

[RFC 1422] Steve Kent: "*Privacy Enhancement for Internet Electronic Mail: Part II - Certificate-Based Key Management*". IAB IRTF PSRG, IETF PEM WG, Fevereiro de 1993.

[RFC 1423] David Balenson: "*Privacy Enhancement for Internet Electronic Mail: Part III - Algorithms, Modes, and Identifiers"*. IAB IRTF PSRG, IETF PEM WG, Fevereiro de 1993.

[RFC 1424] Burton S. Kaliski, Jr. : "*Privacy Enhancement for Internet Electronic Mail: Part IV - Key Certification and Related Services"*. IAB IRTF PSRG, IETF PEM WG, Fevereiro de 1993.

[RFC 1521] N. Borestein & N. Freed: "*Multipurpose Internet Mail Extensions"*. Bellcore & Innsoft, Setembro de 1993.

[RFC 1685] H. Alvestrand: "*Writing X.400 O/R Names*". UNINETT,

Agosto de 1994.

[RFC 819] Jonathan B. Postel & Zaw-Sing Su: *"The Domain Naming Convention for Internet User Applications"*. University of Southern California, Marina de Rey, California, EUA, Agosto de 1982.

[RFC 821] Jonathan B. Postel: *"Simple Mail Transfer Protocol"*. University of Southern California, Marina de Rey, California, EUA, Agosto de 1982.

[RFC 822] David H. Crocker: *"Standard for the Format of ARPA Internet Text Messages"*. University of Delaware (UDEL), EUA, Agosto de 1982.

[RFC 976] Mark R. Horton: *"UUCP Mail Interchange Format Standard"*. Bell Laboratories, EUA, Fevereiro de 1986.

[Riordan 93] Mark Riordan: *"RIPEM User's Guide"*. RIPEM versão1.2 Distribution, Junho de 1993.

[Rivest 78] Ronald L. Rivest, Adi Shamir & Leonard Adleman: *"A Method of Obtaining Digital Signatures and Public-Key Cryptosystems"*. Comm. of the ACM, 21(2), Fevereiro de 1978, pp. 120-126.

[Robshaw 97] M. J. B. Robshaw & Yiqun Lisa Yin: *"Elliptic Curve Cryptosystems"*. RSA Laboratories Technical Note, 27 de junho de 1997.

[Saltzer 75] J. H. Saltzer & M. D. Schroeder: *"The protection of Information in Computer Systems"*. Proceedings, IEEE, vol. 63, Setembro de 1975, pp. 1278-1308.

[Satyanarayanan 89] M. Satyanarayanan: *"Integrating Security in a Large Distributed System"*. ACM Transactions on Computer Systems, 7, 3, Agosto de 1989, pp. 247-280.

[Schneier 95] Bruce Schneier: *"E-Mail Security - How To Keep Your Electronic Messages Private"*. John Wiley & Sons, Inc., EUA, 1995.

[Schneier 96] Bruce Schneier: *"Applied Cryptography"*. John Wiley & Sons, Inc., EUA, 1996.

[Sproull 95] Robert F. Sproull: *"Connections: New Way Of Working In The Networked Organization"*. Cambridge, Mass. MIT, 1995, pp. 59-65 e 177-184.

[Steiner 88] Jennifer Steiner, B. Clifford Neuman & Jeffrey I. Schiller: *"Kerberos: An Authentication Service for Open Network Systems"*. In Proc. Winter USENIX Conference, EUA, 1988, pp. 191-202.

[Stinson 95] Douglas R. Stinson: *"Cryptography - Theory and Practice"*. CRC Press, EUA, 1995.

[Tanenbaum 92] A. S. Tanenbaum: *"Modern Operating Systems"*. Prentice Hall, 1992, pp. 598-629.

[Teixeira 96] José Helvécio Teixeira Júnior: *"Suporte a Sistemas de Correio Eletrônico com SMTP"*. ANAIS / SBRC, Workshop de Administração e Integração de Sistemas, Fortaleza, Maio de 1996.

[Wray 93] John Wray: *"Generic Security Service API: C-bindings"* RFC 1509, Setembro de 1993.

[Zimmermann 94] Philip Zimmermann: *"PGP User's Guide - Volume I: Essential Topics"* PGP Version 2.6, Distribution, Maio de 1994.