Segurança de E_mail
(mensagens eletrônicas)

Guilherme Elias Belassiano & Marita Maestrelli

*Centro Brasileiro de Pesquisas Físicas – CAT/CBPF/CNPq
Rua Dr. Xavier Sigaud, 150
22290-180 – Rio de Janeiro, RJ - Brasil

Sumário

Resumo

O presente documento tem o objetivo de fornecer uma idéia geral do funcionamento do serviço de correio eletrônico (e-mail).

Abordamos o conceito de *third party relay* e o software denominado Sendmail. Explicamos a necessidade de atualização deste programa de forma transparente para o usuário final.

A nova versão do sendmail foi exaustivamente testada durante três meses antes de ser implantada na estação servidora, no intuito de evitar a paralisação, ainda que temporária do serviço de correio eletrônico. Conforme explicado adiante, a não atualização do software poderia acarretar conseqüências indesejáveis.

Palavras chave: sendmail, criptografia, PGP, SMTP, e-mail, server.

Introdução

Primeiro era a Arpanet. Depois veio a Bitnet. E de repente era uma porção de redes de computadores. Hoje temos a Internet, que pode ser vista como “uma rede de redes”.

Há bem pouco tempo atrás, palavras como *web*, *site*, *homepage*, e *e-mail* eram distantes da realidade cotidiana da maioria das pessoas. Já é difícil imaginar a produção deste Centro sem o serviço de e-mail (mensagens eletrônicas).

De fato, o e-mail é uma solução para os problemas de comunicação entre pesquisadores de diversas partes do mundo. Mas toda solução gera novos problemas, e com esta ferramenta não poderia ser diferente. Sempre há pessoas que procuram usar a nova tecnologia para fins distintos daqueles para os quais ela foi idealizada.

No caso do e-mail, há pessoas que enviam centenas (as vezes milhares) de mensagens não solicitadas de conteúdo pornográfico ou publicitário. Essas pessoas são conhecidas como *spammers* e o tipo de correspondências que enviam leva o apelido de *junk mail*.

Para enviar essas mensagens, freqüentemente é feito o uso de um servidor poderoso que, devido a uma brecha na segurança, dá a possibilidade de ser “pego emprestado” para enviar tais mensagens. Assim, uma das coisas que podemos fazer é eliminar a possibilidade de “emprestarmos” nosso servidor para tais fins.

Abordaremos aqui também a violabilidade das mensagens enviadas por e-mail e porque incentivaremos o uso de criptografia.

“Pegando um servidor emprestado”

Este problema é conhecido como *third-party mail relay*. Descreveremos o que isso significa e porque há quem queira fazer isso. Também será explicado porque isso é uma grande ameaça a uma organização séria e os motivos para a sua prevenção.

O uso de um terceiro servidor ocorre quando um servidor de mensagens (*mail server*) processa uma mensagem eletrônica na qual nem o remetente nem o destinatário é um usuário local. No exemplo ilustrado na figura 1, ambos (remetente e destinatário) estão fora do domínio. O *mail server* é um terceiro nesta operação. Não há nenhuma razão que justifique a passagem da mensagem por esse servidor.

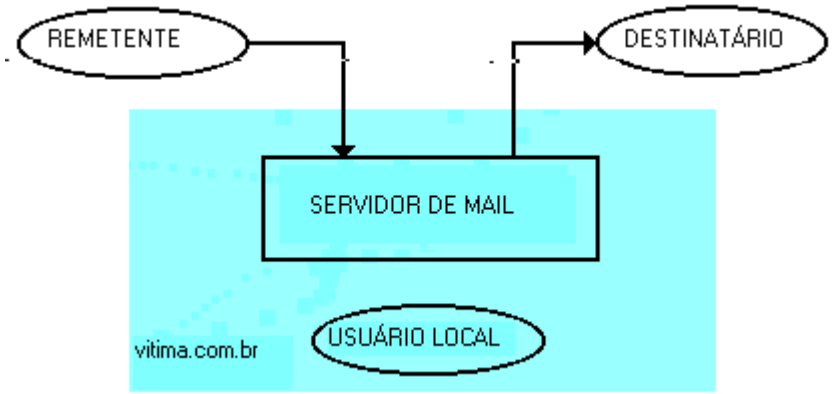

Figura 1 – Esquema de um “relay”

O uso de um terceiro servidor tem alguns poucos usos “legítimos”. Alguns administradores de rede têm-se valido deste recurso para depurar conectividade de *mail*. Também é usado para “driblar” alguns problemas conhecidos de *mail*. Apesar disso ser raramente necessário, esse provou ser um recurso útil nessas ocasiões.

De algum tempo para cá, no entanto, o uso legítimo desse recurso tem sido superado pelo seu uso indevido, quando um servidor é “seqüestrado” para o envio indiscriminado de mensagens inúteis. Isso é uma ameaça significativa a operações na internet.

Por que usar um terceiro servidor?

Há algumas razões para que os *spammers* (pessoas que propagam suas mensagens não solicitadas na internet) façam isso.

- Há um grande número de operações dedicadas ao *spam* na internet, espalhando *e-mails* não solicitados. Alguns administradores de rede começaram a filtrar as conexões e reter aquelas de *sites* que enviam tais mensagens. Assim, os *spammers* precisaram desenvolver novas técnicas para transpassar esse bloqueio. A preferida no momento é a de seqüestrar um *mail server*. Assim, eles transpassam o filtro, pois fazem suas mensagens serem enviadas a partir de um servidor sério.

- Usando um servidor alheio, é possível enviar um número incrivelmente maior de mensagens. Uma pessoa sentada na frente de um PC conectado a uma linha telefônica pode mandar um número limitado de mensagens. Mas se conseguir “pegar emprestado” um servidor poderoso com uma conexão muito mais rápida do que se pode conseguir com um modem, é possível enviar centenas de vezes mais mensagens. Além disso, se um *spammer* conseguir usar vários servidores paralelamente, poderá encher a rede de lixo. O pensamento dessas pessoas é: “por que pagar caro por recursos computacionais e conexão rápida, se é possível usar recursos dos outros?”

- Usando o servidor de um terceiro, um *spammer* consegue esconder sua identidade muito mais facilmente. Apesar de não conseguirem um anonimato completo, eles podem despistar os administradores de rede e direcionar uma boa parte das reclamações ao dono do servidor seqüestrado. Na verdade, muitos *spammers* modificam os cabeçalhos de e-mail para forçar esse redirecionamento.

Assim, o objetivo de “seqüestrar” um *mail server* é aumentar o número de mensagens espalhadas sem nenhum custo. Evidentemente, isso congestiona o sistema seqüestrado e pode até mesmo danificá-lo.

O sistema pode ficar sobrecarregado ou danificado

Um *spammer* está roubando serviço quando usa um servidor alheio. Estão sendo roubados a capacidade da rede, algum espaço em disco e o poder de processamento. Se roubarem demais, o sistema congestiona, podendo travar, e deixando a vítima sem *e-mail* por algum tempo, além de correr o risco de perder dados valiosos.

Depois do “seqüestro”, a conta pode ser muito alta

O uso de um servidor é apenas o início dos problemas. Depois disso, serão necessárias muitas horas de trabalho restaurar as mensagens legítimas que se conseguir, além de implementar as defesas que deveriam ter sido tomadas antes. Além de ter de explicar a queda do *e-mail* por várias horas, ainda será necessário lidar com as reclamações das vítimas do *spammer*.

A reputação e o crédito serão afetados

Trabalhamos duro para conseguir um nome para nossa instituição. Se um *spammer* chegar a usar nosso servidor, todo esse trabalho pode ser destruído em questão de horas, uma vez que milhares (até mesmo milhões) de mensagens não solicitadas vão circular pela rede com o nome de nosso servidor.

Podemos ser incluídos numa lista negra

Um número crescente de organizações está bloqueando mensagens originadas de servidores envolvidos em abuso da rede. Uma vez tendo o servidor seqüestrado, é possível ser listado. Não importa quais sejam nossas intenções, mas apenas o fato de as mensagens não solicitadas terem sido originadas no nosso servidor.

Funcionamento do e-mail

Quando tentamos contatar um computador pela Internet, temos a ilusão de que uma máquina está falando diretamente com a outra, sem intermediários. Mas isso não é verdade. Alguns programas mostram a rota percorrida pelos bits do remetente até o destinatário. Exemplos destes programas são o traceroute (disponível nas máquinas SUN) e o tracert (disponível nos PCs). Por exemplo, do CBPF até o MIT uma rota possível é:

```
 1    2 ms    1 ms    1 ms  cisco7513-CBPF.cat.cbpf.br [152.84.253.1]
 2    2 ms    1 ms    1 ms  rt-rederio-anel.pop-rj.rnp.br [200.159.254.253]

 3   11 ms    9 ms   11 ms  cisco-ufrj-s11.rederio.br [192.80.209.73]
 4  357 ms  370 ms  396 ms  s2-0.san-bb3.cerf.net [134.24.26.100]
 5  470 ms  405 ms  403 ms  h2-0.lax-bb3.cerf.net [134.24.29.10]
 6  413 ms  394 ms  398 ms  134.24.29.17
 7  493 ms  352 ms  448 ms  atm3-0-155M.sfo-bb1.cerf.net [134.24.29.42]
 8  338 ms  445 ms  401 ms  fe9-0-0.sfo-bb2.cerf.net [134.24.29.118]
 9  381 ms  422 ms  403 ms  atm3-0-155M.sjc-bb3.cerf.net [134.24.29.22]
10  408 ms  394 ms  383 ms  h5-1-0.paloalto-br1.bbnplanet.net [4.0.3.81]
11  453 ms  461 ms  429 ms  p2-0.paloalto-nbr1.bbnplanet.net [4.0.2.193]
12  455 ms  465 ms    *     p3-1.chicago1-nbr1.bbnplanet.net [4.0.3.165]
13    *       *       *     Request timed out.
14  468 ms  479 ms  480 ms  h2-0.boston1-br1.bbnplanet.net [4.0.1.126]
15    *       *       *     Request timed out.
16    *       *     486 ms  h1-0.cambridge2-br1.bbnplanet.net [4.0.1.186]
17  493 ms    *       *     ihtfp.mit.edu [192.233.33.3]
18    *       *       *     Request timed out.
19    *       *     448 ms  DANDELION-PATCH.MIT.EDU [18.181.0.31]
Trace complete.
```

As rotas até o destino são dinâmicas, isto é, normalmente vai existir mais de uma alternativa e os roteadores tratam de levar os bits através do caminho menos congestionado e mais curto.

Mail Server

A cada vez que enviamos um e-mail, a mensagem trafega pela rede até o servidor SMTP[1], a partir de onde será enviado ao destino. Para que isso ocorra é necessário que haja um programa servidor na origem e no destino. O mais famoso desses programas é o *sendmail* (usado em quase todos os *sites*). Outros exemplos são o Qmail, o Dixie Mail e o Dmail, mas são muito poucos os *sites* que os utilizam. Aqui no CBPF usamos a versão 8.9.1 do sendmail.

Este software se encarrega de disponibilizar o computador para receber conexões de outros computadores, seja para envio de mensagens ou para recebimento. O software coloca cada mensagem numa fila e as libera conforme a disponibilidade do processador e do link com o servidor de destino. Além disso, ele se encarrega de distribuir as mensagens recebidas para os diversos usuários do CBPF.

[1] Servidor SMTP – Programa que, entre outras coisas, se encarrega de fazer o envio das mensagens de correio eletrônico. Geralmente esse programa roda em uma máquina remota.

No anexo 1, exemplificamos o arquivo de configuração do sendmail, o sendmail.cf .

Problemas com a versão antiga

Até recentemente usávamos a versão 8.6 do sendmail. Esta versão enviava e recebia perfeitamente os nossos e-mails. Mas deixava uma porta aberta para pessoas externas fazerem *relay*, o que é extremamente indesejável por motivos previamente expostos.

Vantagens e desvantagens da nova versão

Para evitar tais conseqüências, atualizamos o sendmail para a versão 8.9.1. Essa versão impede o *relay* dos domínios externos. Quer dizer, protege nosso servidor de locatários caloteiros. Como efeito colateral, nos impede de enviar mensagens de e-mail através dele se estivermos fora da rede (exceto por *telnet*). Ou seja, se estivermos usando um servidor de acesso remoto em outra rede que não a do CBPF, não podemos enviar mensagens através do servidor do CBPF. Podemos, contudo, visualizar as mensagens que chegam em nossas contas no CBPF.

Resumindo, podemos sempre baixar as mensagens que estão aqui, ainda que estejamos em outro lugar; mas não podemos mandar de outro lugar parecendo que foi daqui, exceto quando já estivermos *logados* em alguma máquina (através de *telnet*) dentro do CBPF, autenticados no sistema por uma senha.

Esta versão do sendmail permite o bloqueio e até mesmo o descarte de mensagens de qualquer servidor que não julguemos confiável ou de um endereço particular indesejável.

Outros programas servidores de mail

DixieMail – mail server totalmente desenvolvido em Java e por isso mesmo, compatível com todas as plataformas (windows, unix, mac). Por ser o Java uma linguagem interpretada (e não compilada), sua execução é meio lenta,

o que pode se tornar crítico numa aplicação como essa. O programa permite extensões escritas pelo usuário, tais como servidores de listas e jogos.

CSM Mail Server para windows 9x/NT – suporta os protocolos SMTP e POP3. Dizem os autores que esse programa consegue combinar o poder dos servidores para UNIX com a facilidade de uso de um ambiente windows.

Dmail – servidor de e-mail e de listas, escrito pensando em provedores de internet e intranet. Faz automaticamente a estatística de conexões para cobrança.

Qmail – a intenção deste programa é substituir o sendmail. O qmail pretende ser um “pacote seguro” e até agora parece ser, pois foi oferecido um prêmio de US$ 1.000.000 para quem demonstrasse o contrário. Por enquanto, ninguém ganhou. Esse programa já tem solucionado o *bug* do milênio.

Mail client

Os clientes de mensagens são os programas que usamos nos computadores pessoais para enviar mensagens para o servidor e ler as mensagens que chegam até ele e estão armazenadas nas nossas caixas postais. Alguns exemplos são o Microsoft Outlook (figura 3), o Netscape Messenger e o Qualcomm Eudora (figura 4).

Esses programas, ao contrário do programa servidor, têm uma interface bastante amigável. Esses programas são, ao mesmo tempo, clientes SMTP (a cada vez que enviam uma mensagem) e clientes POP3 (a cada vez que recebem mensagem).

Comunicação entre cliente e servidor

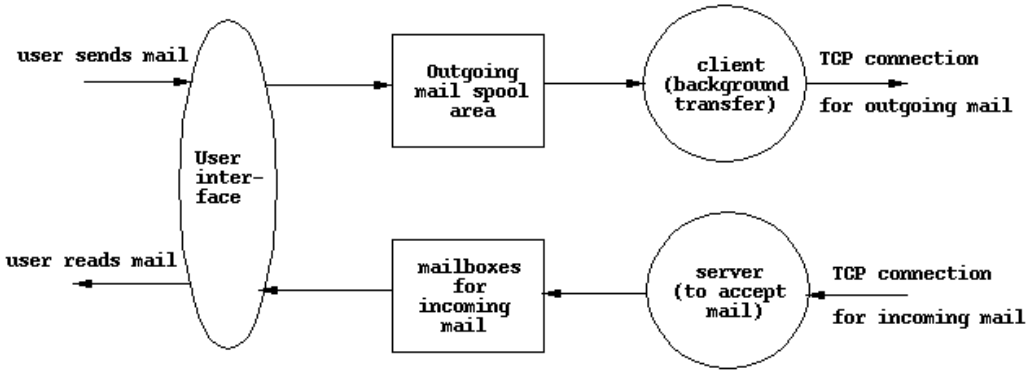

Figura 2 – Esquema de uma conexão com servidor SMTP para envio e para leitura de mensagens.

Exemplos de interfaces de e-mail com o usuário final. Esses programas, além de clientes SMTP, também são clientes POP.

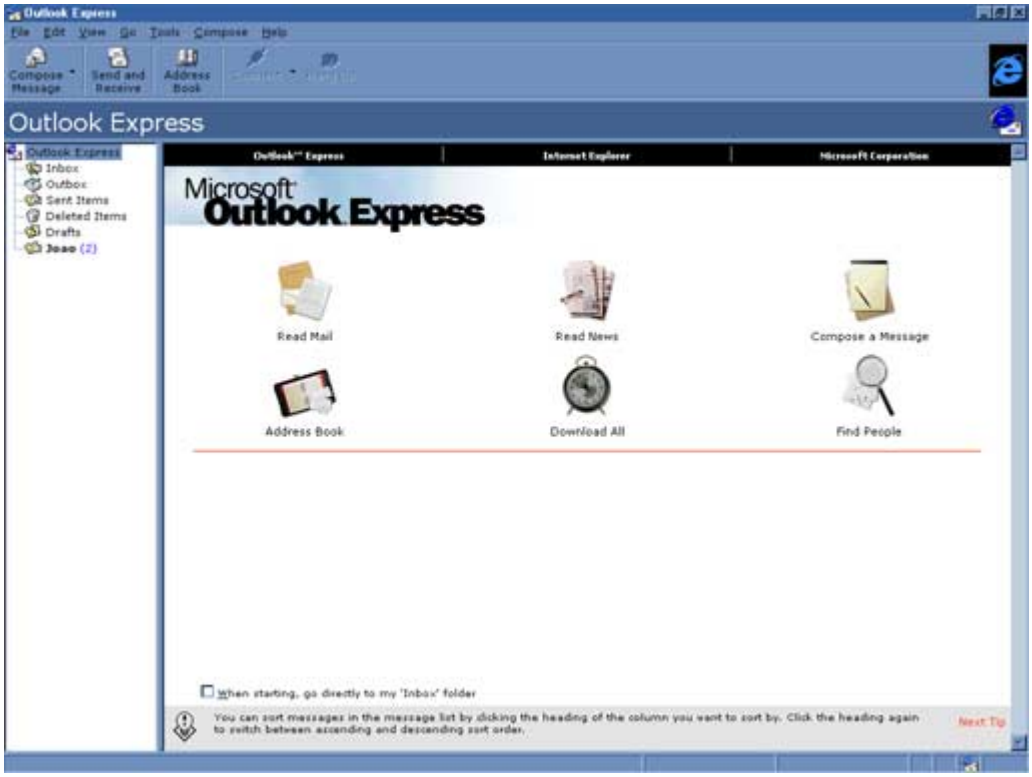

Figura 3 (Microsoft Outlook Express)

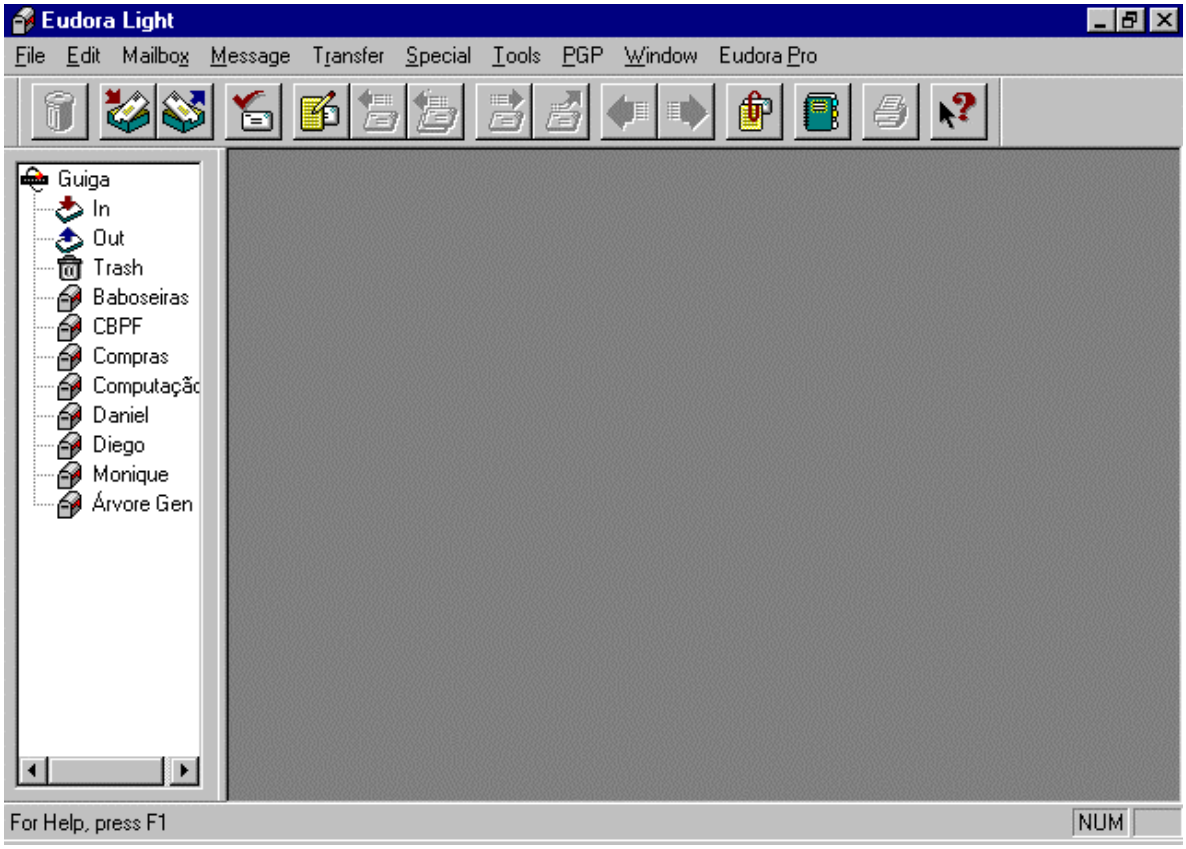

Figura 4 (Qualcomm Eudora Light)

Segurança das mensagens

Em qualquer máquina por onde a mensagem passa no caminho até o destino pode ficar uma cópia. Pode até mesmo ser lida e alterada. Alguns especialistas em segurança dizem que “o e-mail é um cartão postal escrito a lápis”. É claro que isso não é feito de maneira sistemática devido ao enorme número de mensagens que passam por essas máquinas. Além disso, seus administradores costumam estar muito ocupados com tarefas mais nobres do que bisbilhotar mensagens alheias.

Para tornar impraticável a leitura de uma mensagem, recomenda-se o uso de criptografia nas mensagens enviadas por e-mail.

Criptografia

Conforme nos direcionamos a uma sociedade de informações os meios tecnológicos para a disponibilização de dados de milhões de indivíduos se fazem disponíveis para muitos governos. A Criptografia tornou-se uma das principais ferramentas de privacidade, confiança, controle de acesso, pagamentos eletrônicos, segurança corporativa e muitos outros campos.

Em criptografia, a mensagem é chamada de **texto claro**. A codificação do conteúdo da mensagem de forma a escondê-la de terceiros é chamada **encriptação**. A mensagem encriptada (ou criptografada) é chamada de **texto cifrado**. O processo de recuperação do texto claro a partir do texto cifrado é chamado **decriptação**. Encriptação e decriptação geralmente fazem uso de uma **chave** e o método de codificação é tal que sua decodificação é quase impossível sem o conhecimento da chave.

Tipos de Criptografia

Existem duas classes de criptografia baseada em chave: **simétrica** (ou de chave secreta) e **assimétrica** (ou de chave pública). A diferença é que a criptografia simétrica usa a mesma chave para encriptação e decriptação (ou uma chave é facilmente obtida a partir da outra), enquanto que a criptografia assimétrica usa duas chaves diferentes: uma (pública) para encriptação e outra (privada) para decriptação. Uma chave não pode ser obtida a partir da outra.

A chave pública, como o próprio nome diz, pode ser publicada até mesmo num jornal de grande circulação, permitindo a qualquer um criptografar a mensagem. Mas apenas o destinatário poderá recuperar a mensagem, fazendo uso da chave privada.

Existem softwares próprios para a criptografia de mensagens enviadas por e-mail. O mais popular deles é o PGP.

Força dos algoritmos de criptografia

Em teoria, qualquer método criptográfico pode ser quebrado se todas as possibilidades forem tentadas. Mas é possível construir sistemas que na prática não podem ser quebrados (embora isso não possa ser provado). E isso não aumenta de maneira significativa o esforço de implementação.

Se o uso da força bruta[2] for a única opção, o poder de computação exigido para quebrar uma chave aumenta exponencialmente de acordo com o número de bits dela. Vide tabela 1.

Chaves privadas de 32 bits	Exigiriam 2^{32} (aproximadamente 10^9) tentativas. Podem ser quebradas por uma amador com o uso de um computador doméstico.
Chaves privadas de 40 bits	Aproximadamente 10^{12} tentativas. Podem ser quebradas por muitas universidades e pequenas organizações.
Chaves privadas de 56 bits	Aproximadamente 10^{16} tentativas. Exigem um esforço considerável, mas dentro das possibilidades de governos poderosos e grandes organizações.
Chaves privadas de 64 bits	Aproximadamente 10^{19} tentativas. Talvez sejam quebráveis por governos poderosos e grandes organizações. Deve ficar vulnerável em pouco tempo também para as pequenas organizações.
Chaves privadas de 80 bits	Poderão ser quebradas em um futuro próximo.
Chaves privadas de 128 bits	Devem continuar impenetráveis por força bruta até onde é possível prever.

Tabela 1 – Alguns comprimentos de chave pública e a correspondente dificuldade para serem quebradas por força bruta.

Chaves ainda maiores são possíveis. Encontraremos um limite para o comprimento de uma chave quando a energia consumida pela computação de todas as tentativas exceder a energia da massa do Sol ou mesmo do universo.
Entretanto, o comprimento da chave não é o único ponto relevante. É possível romper muitos métodos sem que sejam tentadas todas as possibilidades.

O comprimento das chaves usadas em criptografia de chave pública é normalmente muito maior que os usados em criptografia de chave privada, uma vez que aí o problema não é adivinhar a chave, mas obtê-la a partir da chave pública. No caso do RSA[3], um módulo de 256 bits pode ser fatorado por pessoas comuns; um módulo de 384 bits, por grupos de pesquisa em universidades e companhias; 512 bits está dentro do alcance de governos poderosos. Chaves de 768 bits são inseguras a longo prazo. 1024 bits é considerado seguro a longo prazo, a menos que grandes avanços sejam feitos em algoritmos de fatoração. 2048 bits é considerado seguro por várias décadas.

PGP

PGP (*Pretty Good Privacy*) é um programa de encriptação que usa chave pública. Nos últimos anos este programa ganhou muitos adeptos no mundo todo e tornou-se quase um padrão para a encriptação de e-mail na Internet. A versão 5.5.3i inclui a opção de instalar o PGP como plug-in do Eudora ou do MS Outlook.

O uso de criptografia de chave pública torna possível privacidade na troca de mensagens por correio eletrônico, sem que as partes envolvidas tenham que se encontrar para combinar uma senha, pois a chave que “fecha” a mensagem é diferente daquela que a “abre”. Veja o exemplo

[2] Força bruta – por esta expressão, entende-se busca por exaustão, ou seja, a tentativa de todas as possibilidades, uma a uma.
[3] RSA - As iniciais dos inventores deste algoritmo, que é baseado na dificuldade de fatorar grandes números. Enquanto é relativamente fácil encontrar grandes primos, é computacionalmente inviável fatorar o produto de dois primos.

no quadro abaixo:

Maria escreveu para João: “Amanhã vou viajar.”

Maria criptografa a mensagem com a chave pública de João. O resultado fica algo como:

-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: PGPfreeware 5.5.3i for non-commercial use <http://www.pgpi.com>

mQGiBDXrFAoRBADWTnCwUDBMNM+WfIrUffCZDXs8fBT5n3rBO98R3epXj0Jlb1P8
8Pi3w/IRsQ90M8KsPmYgP6Epcx4pDcnfS+wQj74KggIbCTXltU04a3z7BFMeYc/g
wEEby0OOBS+GWxKiqp9uTiR3enxMyASZXLoBURnoMI7ExujaATSNnoLvIwCg/2RN
2RsaTpk9STFGtTUR3khbZpED/0A+BOM751YF/IeLDZWqAv+MCMjvY76DxIlxWkCo
MhdJAq4zjSdrqN800p/xoqVGnfOruNIH4kG3mgzu2oUlVhIk4mFHp2U0Ujd+Cy9P
-----END PGP PUBLIC KEY BLOCK-----

Como se vê, a mensagem fica ilegível. E é assim que João a recebe. Além da mensagem, o PGP envia uma cópia criptografada do cabeçalho (origem da mensagem, data, destino, etc.), o que dificulta sua alteração durante o trajeto. Para recuperar a mensagem, João precisa ter a chave privada (senha).

Ao decriptar a mensagem com sua chave privada, João obtém de volta a mensagem:

“Amanhã vou viajar.”
Atente para o detalhe de que, encriptada desta forma, apenas a chave privada de João decripta a mensagem. Se Maria quisesse enviar a mesma mensagem para outra pessoa, seria necessário “fechar” com outra chave pública. No caso de mais de um destinatário para a mesma mensagem, o PGP se encarrega de fechar cada mensagem com a chave apropriada. Para isso, o PGP cria arquivos “chaveiros” (*keyring files*[4]), com as chaves públicas de todas as pessoas para quem você pode enviar e-mail desta forma.

Criando um par de chaves com o PGP

Uma chave pública não tem sentido se não existir uma chave privada correspondente. Por isso, ao criar uma chave privada, o PGP cria também uma chave pública. Esta chave pública deverá ser enviada em forma de arquivo para todas as pessoas com quem você deseja trocar e-mail criptografado. Como a chave pública serve apenas para “fechar” a mensagem, alguém que a intercepte não poderá “abrir” uma mensagem cujo destinatário seja você.

A figura 5 é a tela inicial para a geração de um par de chaves no PGP.

[4] Estes arquivos guardam chaves públicas das pessoas com quem se deseja trocar correspondência segura.

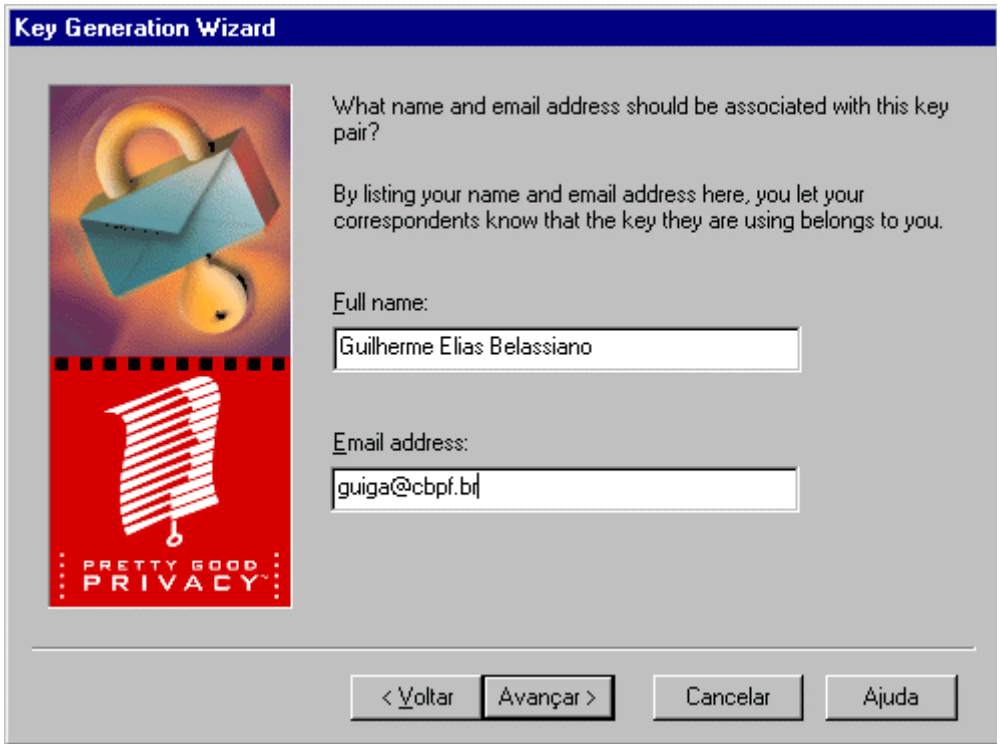

Figura 5 - PGP

No segundo passo, o programa pede que se escolha um tipo de chave. Se você pretende trocar correspondência segura com um usuário antigo do PGP, é sugerido que se escolha o tipo RSA. Caso contrário, é recomendada a utilização de chaves que usam o algoritmo de Diffie-Helman[5].

Figura 5.1 – Tipo de chave

Em seguida, o programa pede uma data de validade das chaves. Isso vai depender do teor das mensagens de cada usuário.

[5] Este algoritmo é baseado no problema do logaritmo discreto

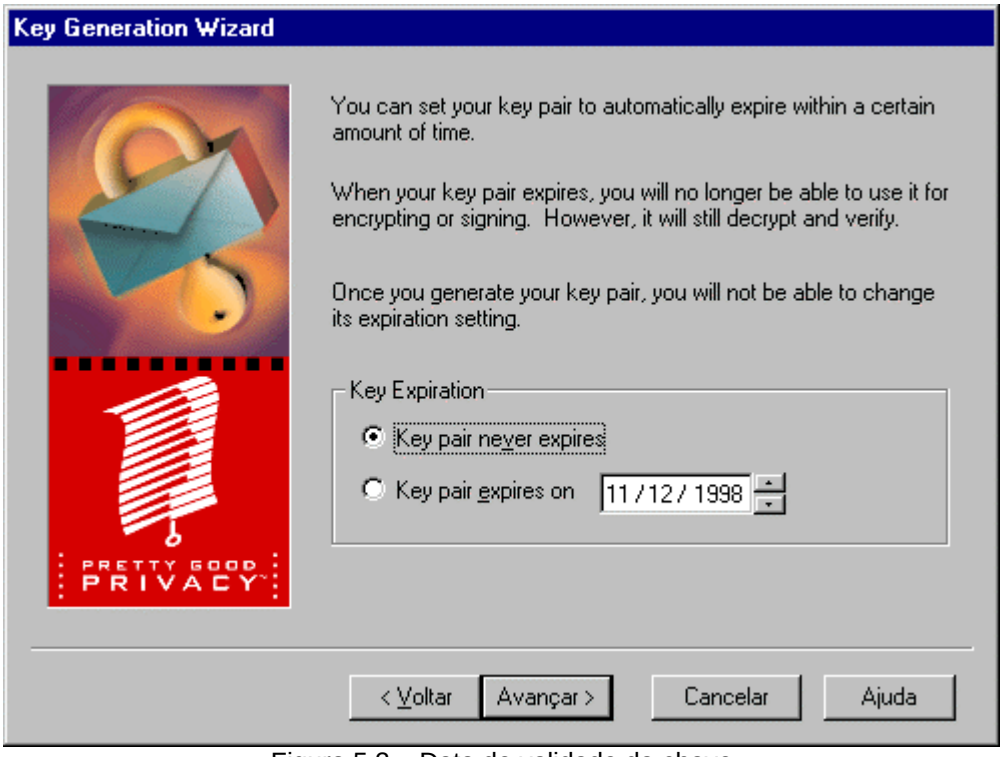

Figura 5.2 – Data de validade da chave

Na tela seguinte, o programa pede uma “passphrase”. Aqui, a senha do usuário será uma frase, ao contrário das senhas de uma palavra (“password”). A frase deve ser memorizada pelo usuário, mas difícil de ser deduzida por outros.

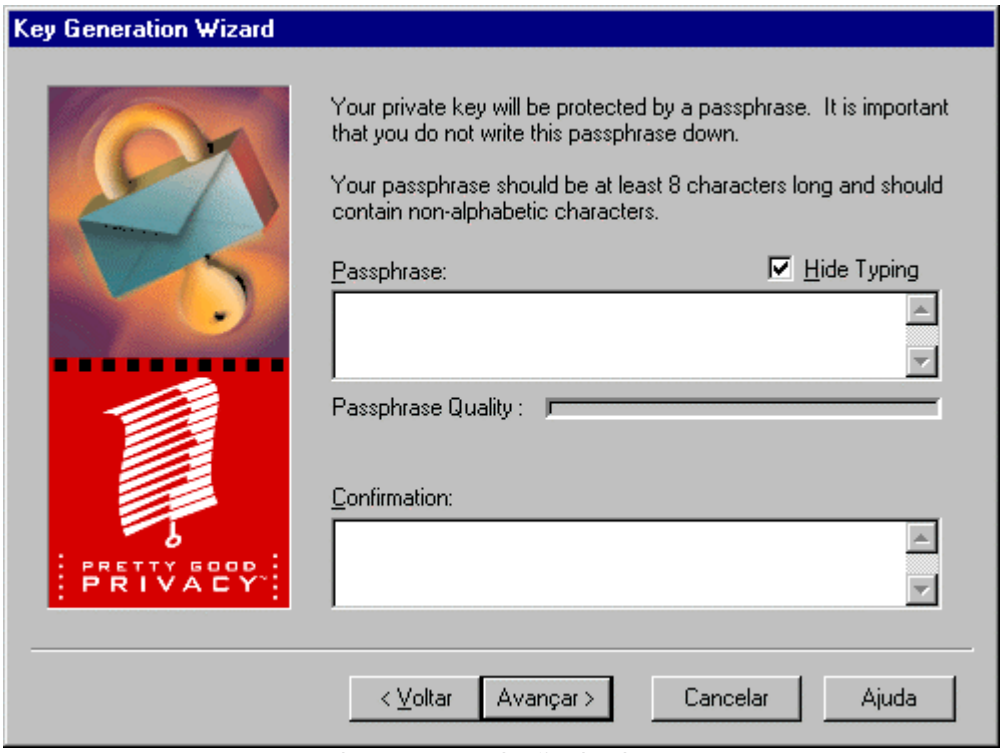

Figura 5.3 – Criação da chave

É interessante enviar a chave para um servidor, pois assim é mais fácil para outras pessoas acharem a sua chave. Além disso, caso você venha a perdê-la é possível recuperá-la a partir do servidor. Aqui, vale lembrar que a chave pública (que o programa gera) é função da chave privada (que você escolhe). A chave pública fica arquivada no computador e ninguém precisa memorizá-la.

Figura 5.4 – Envio da chave

Esta é a tela de confirmação

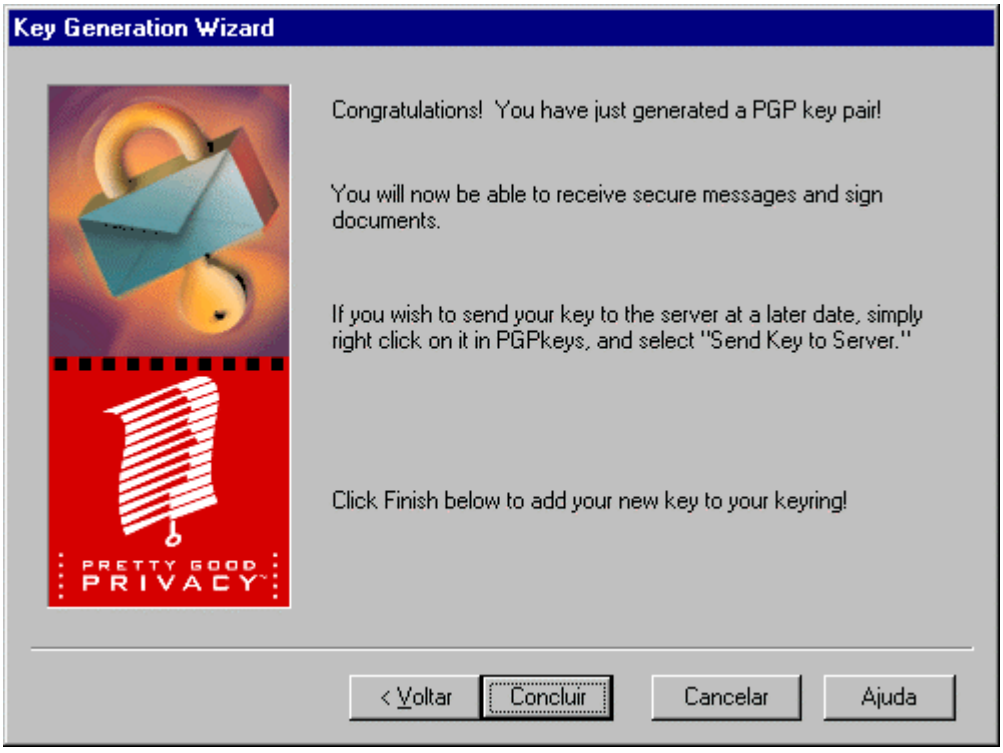

Figura 5.5 – Geração do par de chaves

Conclusão

Um servidor de e-mail é seqüestrado quando envia mensagem de um remetente alheio à rede, para um destinatário igualmente alheio. Em sua época, isso foi uma ferramenta útil, mas pouco usada. Hoje, no entanto, é usado para espalhar mensagens não solicitadas pela Internet.

Impedir essa prática tornou-se parte do trabalho dos administradores de sistemas.

Por tudo isso, optamos pela instalação da versão 8.9.1 do sendmail, a primeira que proíbe o relay a partir de qualquer computador. Para que o relay seja permitido, é necessário que o computador onde a mensagem foi originada esteja previamente autorizado.

Quando enviamos um *e-mail*, ele passa por diversas máquinas até chegar no servidor de destino. Em cada uma delas, o conteúdo pode ser visto e até mesmo alterado por quem tenha acesso pleno a uma dessas máquinas (seja um administrador ou um *hacker*).

Para garantir a segurança das mensagens, recomenda-se o uso de criptografia de chave pública, através de softwares como o PGP.

Glossário

Internet Protocol (IP)
Protocolo “connectionless” (sem estabelecimento de circuito) da camada “internet” na arquitetura TCP/IP, responsável pela conexão lógica entre as redes.

Simple Mail Transfer Protocol (SMTP)
Sistema de correio eletrônico para o transmissor e o receptor.

User Agent (UA)
Também conhecido como Mail Client. Programa que interage com o usuário, responsável pela obtenção de mensagens a serem transmitidas e a retirada de mensagens a serem recebidas. Como exemplo temos o Eudora, o Netscape Mail e o MS Outlook.

Mail Transport Agent (MTA)
Também chamado de programa servidor. Programa responsável pelo transporte de mensagens entre os pontos envolvidos, locais ou através da internet. Um bom exemplo é o Sendmail.

Mail Boxes
Caixas postais onde são armazenadas as mensagens recebidas

Mail Box Manager
Programa responsável pelo gerenciamento das caixas postais, necessário especialmente quando os programas UA e MTA não residem no mesmo equipamento.

Roteador
Equipamento dedicado, cuja finalidade é enviar dados pelo melhor caminho. Por melhor caminho, entende-se o mais curto e o menos congestionado.

Criptografia
Arte ou ciência de manter secretas as mensagens.

Protocolo
Forma como os bits de informação estão codificados. A grosso modo, dois computadores precisam trocar informações usando o mesmo protocolo da mesma que duas pessoas trocam informações no mesmo idioma.

Junk mail
Correspondência eletrônica não solicitada, geralmente comercial.

Spam
Envio de mensagens indesejadas (*junk mail*) para múltiplos destinatários.

Spammer
Indivíduo que faz spam.

Debugar
Aportuguesamento do termo em inglês “debug”. Significa depurar, reparar os erros no código fonte ou na implementação de um software.

Hacker
Pessoa obsecada por computadores. Atualmente, o termo é usado como para referir-se a invasores digitais, ou seja pessoas que entram em sistemas sem autorização.

Referências

What is mail third party relay?
Mail Abuse prevention system
Disponível na internet em: http://mpas.vix.com/tsi/ar-what.html

Sendmail installation and operation guide

The international crypto page
Universidade de Helsinki, Finlândia
Disponível na internet em http://www.cs.hut.fi/crypto

Tucows network
Disponível na internet em http://www.tucows.com

Guia de conectividade
Cyclades

Apêndice – Um exemplo de arquivo de configuração: sendmail.cf

```
#
# Copyright (c) 1998 Sendmail, Inc.  All rights reserved.
# Copyright (c) 1983, 1995 Eric P. Allman.  All rights reserved.
# Copyright (c) 1988, 1993
#           The Regents of the University of California.  All rights reserved.
#
# By using this file, you agree to the terms and conditions set
# forth in the LICENSE file which can be found at the top level of
# the sendmail distribution.
#
#

##########################################################################
##########################################################################
#####
#####                  SENDMAIL CONFIGURATION FILE
#####
##### built by gshapiro@knecht.Sendmail.ORG on Thu Jul 2 11:05:45 PDT 1998
##### in /home/knecht/a/eric/src/sendmail/cf/cf
##### using ../ as configuration include directory
#####
##########################################################################
##########################################################################

#####  @(#)cfhead.m4          8.22 (Berkeley) 5/19/98  #####
#####  @(#)cf.m4   8.29 (Berkeley) 5/19/98  #####
#####  @(#)generic-solaris2.mc          8.8 (Berkeley) 5/19/98  #####

#####  @(#)solaris2.m4        8.15 (Berkeley) 5/19/98  #####

#####  @(#)generic.m4         8.9 (Berkeley) 5/19/98  #####

#####  @(#)redirect.m4        8.10 (Berkeley) 5/19/98  #####



#####  @(#)use_cw_file.m4   8.6 (Berkeley) 5/19/98  #####



#####  @(#)proto.m4           8.223 (Berkeley) 6/30/98  #####


# level 8 config file format
V8/Berkeley

# override file safeties - setting this option compromises system security
# need to set this now for the sake of class files
#O DontBlameSendmail=safe

###################
#   local info   #
###################

Cwlocalhost
# file containing names of hosts for which we receive email
Fw/etc/mail/sendmail.cw



# my official domain name
# ... define this only if sendmail cannot automatically determine your domain
#Djargonio.cat.cbpf.br
```

```
CP.

# "Smart" relay host (may be null)
DS


# operators that cannot be in local usernames (i.e., network indicators)
CO @ % !

# a class with just dot (for identifying canonical names)
C..

# a class with just a left bracket (for identifying domain literals)
C[[




# Resolve map (to check if a host exists in check_mail)
Kresolve host -a<OK> -T<TEMP>

# Hosts that will permit relaying ($=R)
FR-o /etc/mail/relay-domains

# who I send unqualified names to (null means deliver locally)
DR

# who gets all local email traffic ($R has precedence for unqualified names)
DH

# dequoting map
Kdequote dequote

# class E: names that should be exposed as from this host, even if we masquerade
# class L: names that should be delivered locally, even if we have a relay
# class M: domains that should be converted to $M
#CL root
CE root
CM argonio.cat.cbpf.br

# who I masquerade as (null for no masquerading) (see also $=M)
DM

# my name for error messages
DnMAILER-DAEMON


CPREDIRECT

# Configuration version number
DZ8.9.1


################
#   Options   #
################

# strip message body to 7 bits on input?
O SevenBitInput=False

# 8-bit data handling
O EightBitMode=pass8


# wait for alias file rebuild (default units: minutes)
O AliasWait=10

# location of alias file
O AliasFile=/etc/mail/aliases
#O AliasFile=/etc/mail/majordomo.aliases
```

```
# minimum number of free blocks on filesystem
O MinFreeBlocks=100

# maximum message size
#O MaxMessageSize=1000000

# substitution for space (blank) characters
O BlankSub=.

# avoid connecting to "expensive" mailers on initial submission?
O HoldExpensive=False

# checkpoint queue runs after every N successful deliveries
#O CheckpointInterval=10

# default delivery mode
O DeliveryMode=background

# automatically rebuild the alias database?
#O AutoRebuildAliases

# error message header/file
#O ErrorHeader=/etc/sendmail.oE

# error mode
#O ErrorMode=print

# save Unix-style "From_" lines at top of header?
#O SaveFromLine

# temporary file mode
O TempFileMode=0600

# match recipients against GECOS field?
#O MatchGECOS

# maximum hop count
#O MaxHopCount=17

# location of help file
O HelpFile=/etc/mail/sendmail.hf

# ignore dots as terminators in incoming messages?
#O IgnoreDots

# name resolver options
#O ResolverOptions=+AAONLY

# deliver MIME-encapsulated error messages?
O SendMimeErrors=True

# Forward file search path
O ForwardPath=$z/.forward.$w+$h:$z/.forward+$h:$z/.forward.$w:$z/.forward

# open connection cache size
O ConnectionCacheSize=2

# open connection cache timeout
O ConnectionCacheTimeout=5m

# persistent host status directory
#O HostStatusDirectory=.hoststat

# single thread deliveries (requires HostStatusDirectory)?
#O SingleThreadDelivery

# use Errors-To: header?
O UseErrorsTo=False

# log level
O LogLevel=9

# send to me too, even in an alias expansion?
#O MeToo

# verify RHS in newaliases?
```

```
O CheckAliases=False

# default messages to old style headers if no special punctuation?
O OldStyleHeaders=True

# SMTP daemon options
#O DaemonPortOptions=Port=esmtp

# privacy flags
O PrivacyOptions=authwarnings

# who (if anyone) should get extra copies of error messages
#O PostMasterCopy=Postmaster

# slope of queue-only function
#O QueueFactor=600000

# queue directory
O QueueDirectory=/var/spool/mqueue

# timeouts (many of these)
#O Timeout.initial=5m
#O Timeout.connect=5m
#O Timeout.iconnect=5m
#O Timeout.helo=5m
#O Timeout.mail=10m
#O Timeout.rcpt=1h
#O Timeout.datainit=5m
#O Timeout.datablock=1h
#O Timeout.datafinal=1h
#O Timeout.rset=5m
#O Timeout.quit=2m
#O Timeout.misc=2m
#O Timeout.command=1h
#O Timeout.ident=30s
#O Timeout.fileopen=60s
O Timeout.queuereturn=5d
#O Timeout.queuereturn.normal=5d
#O Timeout.queuereturn.urgent=2d
#O Timeout.queuereturn.non-urgent=7d
O Timeout.queuewarn=4h
#O Timeout.queuewarn.normal=4h
#O Timeout.queuewarn.urgent=1h
#O Timeout.queuewarn.non-urgent=12h
#O Timeout.hoststatus=30m

# should we not prune routes in route-addr syntax addresses?
#O DontPruneRoutes

# queue up everything before forking?
O SuperSafe=True

# status file
O StatusFile=/etc/mail/sendmail.st

# time zone handling:
#  if undefined, use system default
#  if defined but null, use TZ envariable passed in
#  if defined and non-null, use that info
#O TimeZoneSpec=

# default UID (can be username or userid:groupid)
#O DefaultUser=mailnull

# list of locations of user database file (null means no lookup)
#O UserDatabaseSpec=/etc/userdb

# fallback MX host
#O FallbackMXhost=fall.back.host.net

# if we are the best MX host for a site, try it directly instead of config err
#O TryNullMXList

# load average at which we just queue messages
#O QueueLA=8

# load average at which we refuse connections
```

```
#O RefuseLA=12

# maximum number of children we allow at one time
#O MaxDaemonChildren=12

# maximum number of new connections per second
#O ConnectionRateThrottle=3

# work recipient factor
#O RecipientFactor=30000

# deliver each queued job in a separate process?
#O ForkEachJob

# work class factor
#O ClassFactor=1800

# work time factor
#O RetryFactor=90000

# shall we sort the queue by hostname first?
#O QueueSortOrder=priority

# minimum time in queue before retry
#O MinQueueAge=30m

# default character set
#O DefaultCharSet=iso-8859-1

# service switch file (ignored on Solaris, Ultrix, OSF/1, others)
#O ServiceSwitchFile=/etc/service.switch

# hosts file (normally /etc/hosts)
#O HostsFile=/etc/hosts

# dialup line delay on connection failure
#O DialDelay=10s

# action to take if there are no recipients in the message
#O NoRecipientAction=add-to-undisclosed

# chrooted environment for writing to files
#O SafeFileEnvironment=/arch

# are colons OK in addresses?
#O ColonOkInAddr

# how many jobs can you process in the queue?
#O MaxQueueRunSize=10000

# shall I avoid expanding CNAMEs (violates protocols)?
#O DontExpandCnames

# SMTP initial login message (old $e macro)
O SmtpGreetingMessage=$j Sendmail $v/$Z; $b

# UNIX initial From header format (old $l macro)
O UnixFromLine=From $g  $d

# From: lines that have embedded newlines are unwrapped onto one line
#O SingleLineFromHeader=False

# Allow HELO SMTP command that does not include a host name
#O AllowBogusHELO=False

# Characters to be quoted in a full name phrase (@,;:\()[] are automatic)
#O MustQuoteChars=.

# delimiter (operator) characters (old $o macro)
O OperatorChars=.:%@!^/[]+

# shall I avoid calling initgroups(3) because of high NIS costs?
#O DontInitGroups

# are group-writable :include: and .forward files (un)trustworthy?
#O UnsafeGroupWrites
```

```
# where do errors that occur when sending errors get sent?
#O DoubleBounceAddress=postmaster

# what user id do we assume for the majority of the processing?
#O RunAsUser=sendmail

# maximum number of recipients per SMTP envelope
#O MaxRecipientsPerMessage=100

# shall we get local names from our installed interfaces?
#O DontProbeInterfaces

############################
#   Message precedences   #
############################

Pfirst-class=0
Pspecial-delivery=100
Plist=-30
Pbulk=-60
Pjunk=-100

######################
#   Trusted users   #
######################

# this is equivalent to setting class "t"
#Ft/etc/sendmail.ct
Troot
Tdaemon
Tuucp

#########################
#   Format of headers   #
#########################

H?P?Return-Path: <$g>
HReceived: $?sfrom $s $.$?_($?s$|from $.$_)
        $.by $j ($v/$Z)$?r with $r$. id $i$?u
        for $u; $|;
        $.$b
H?D?Resent-Date: $a
H?D?Date: $a
H?F?Resent-From: $?x$x <$g>$|$g$.
H?F?From: $?x$x <$g>$|$g$.
H?x?Full-Name: $x
# HPosted-Date: $a
# H?l?Received-Date: $b
H?M?Resent-Message-Id: <$t.$i@$j>
H?M?Message-Id: <$t.$i@$j>
#
```

```
##########################################################################
##########################################################################
#####
#####                          REWRITING RULES
#####
##########################################################################
##########################################################################

###############################################
###  Ruleset 3 -- Name Canonicalization  ###
###############################################
S3

# handle null input (translate to <@> special case)
R$@                     $@ <@>

# strip group: syntax (not inside angle brackets!) and trailing semicolon
R$*                     $: $1 <@>                       mark addresses
R$* < $* > $* <@>  $: $1 < $2 > $3              unmark <addr>
R@ $* <@>               $: @ $1                         unmark @host:...
R$* :: $* <@>           $: $1 :: $2                     unmark node::addr
R:include: $* <@>  $: :include: $1              unmark :include:...
R$* [ $* : $* ] <@>  $: $1 [ $2 : $3 ]          unmark IPv6 addrs
R$* : $* [ $* ]         $: $1 : $2 [ $3 ] <@>           remark if leading colon
R$* : $* <@>            $: $2                           strip colon if marked
R$* <@>                 $: $1                           unmark
R$* ;                     $1                            strip trailing semi
R$* < $* ; >              $1 < $2 >                     bogus bracketed semi

# null input now results from list:; syntax
R$@                     $@ :; <@>

# strip angle brackets -- note RFC733 heuristic to get innermost item
R$*                     $: < $1 >               housekeeping <>
R$+ < $* >                < $2 >                strip excess on left
R< $* > $+                < $1 >                strip excess on right
R<>                     $@ < @ >                        MAIL FROM:<> case
R< $+ >                 $: $1                           remove housekeeping <>

# make sure <@a,@b,@c:user@d> syntax is easy to parse -- undone later
R@ $+ , $+              @ $1 : $2               change all "," to ":"

# localize and dispose of route-based addresses
R@ $+ : $+              $@ $>96 < @$1 > : $2            handle <route-addr>

# find focus for list syntax
R $+ : $* ; @ $+     $@ $>96 $1 : $2 ; < @ $3 >   list syntax
R $+ : $* ;             $@ $1 : $2;                     list syntax

# find focus for @ syntax addresses
R$+ @ $+                $: $1 < @ $2 >                  focus on domain
R$+ < $+ @ $+ >         $1 $2 < @ $3 >                  move gaze right
R$+ < @ $+ >            $@ $>96 $1 < @ $2 >             already canonical

# do some sanity checking
R$* < @ $* : $* > $*    $1 < @ $2 $3 > $4       nix colons in addrs

# convert old-style addresses to a domain-based address
R$- ! $+             $@ $>96 $2 < @ $1 .UUCP >       resolve uucp names
R$+ . $- ! $+           $@ $>96 $3 < @ $1 . $2 >        domain uucps
R$+ ! $+             $@ $>96 $2 < @ $1 .UUCP >       uucp subdomains

# if we have % signs, take the rightmost one
R$* % $*             $1 @ $2                         First make them all @s.
R$* @ $* @ $*           $1 % $2 @ $3                    Undo all but the last.
R$* @ $*             $@ $>96 $1 < @ $2 >             Insert < > and finish

# else we must be a local name
R$*                     $@ $>96 $1


################################################
###  Ruleset 96 -- bottom half of ruleset 3  ###
################################################

S96
```

```
# handle special cases for local names
R$* < @ localhost > $*          $: $1 < @ $j . > $2          no domain at all
R$* < @ localhost . $m > $*   $: $1 < @ $j . > $2          local domain
R$* < @ localhost . UUCP > $*   $: $1 < @ $j . > $2          .UUCP domain
R$* < @ [ $+ ] > $*          $: $1 < @@ [ $2 ] > $3          mark [a.b.c.d]
R$* < @@ $=w > $*          $: $1 < @ $j . > $3          self-literal
R$* < @@ $+ > $*          $@ $1 < @ $2 > $3          canon IP addr




# if really UUCP, handle it immediately

# try UUCP traffic as a local address
R$* < @ $+ . UUCP > $*          $: $1 < @ $[ $2 $] . UUCP . > $3
R$* < @ $+ . . UUCP . > $*    $@ $1 < @ $2 . > $3

# pass to name server to make hostname canonical
R$* < @ $* $~P > $*          $: $1 < @ $[ $2 $3 $] > $4

# local host aliases and pseudo-domains are always canonical
R$* < @ $=w > $*          $: $1 < @ $2 . > $3
R$* < @ $j > $*          $: $1 < @ $j . > $2
R$* < @ $=M > $*          $: $1 < @ $2 . > $3
R$* < @ $* $=P > $*          $: $1 < @ $2 $3 . > $4
R$* < @ $* . . > $*          $1 < @ $2 . > $3


######################################################
###  Ruleset 4 -- Final Output Post-rewriting  ###
######################################################
S4

R$* <@>                $@                    handle <> and list:;

# strip trailing dot off possibly canonical name
R$* < @ $+ . > $*   $1 < @ $2 > $3

# eliminate internal code -- should never get this far!
R$* < @ *LOCAL* > $*          $1 < @ $j > $2

# externalize local domain info
R$* < $+ > $*          $1 $2 $3          defocus
R@ $+ : @ $+ : $+  @ $1 , @ $2 : $3          <route-addr> canonical
R@ $*          $@ @ $1                    ... and exit

# UUCP must always be presented in old form
R$+ @ $- . UUCP          $2!$1                    u@h.UUCP => h!u

# delete duplicate local names
R$+ % $=w @ $=w          $1 @ $2                    u%host@host => u@host


##################################################################
###   Ruleset 97 -- recanonicalize and call ruleset zero   ###
###                (used for recursive calls)                ###
##################################################################

S97
R$*                $: $>3 $1
R$*                $@ $>0 $1


########################################
###   Ruleset 0 -- Parse Address   ###
########################################

S0

R$*                $: $>Parse0 $1          initial parsing
R<@>                $#local $: <@>          special case error msgs
R$*                $: $>98 $1          handle local hacks
R$*                $: $>Parse1 $1          final parsing
```

```
#
#  Parse0 -- do initial syntax checking and eliminate local addresses.
#            This should either return with the (possibly modified) input
#            or return with a #error mailer.  It should not return with a
#            #mailer other than the #error mailer.
#

SParse0
R<@>                    $@ <@>                  special case error msgs
R$* : $* ; <@>          $#error $@ 5.1.3 $: "List:; syntax illegal for recipient addresses"
#R@ <@ $* >             < @ $1 >            catch "@@host" bogosity
R<@ $+>                 $#error $@ 5.1.3 $: "User address required"
R$*                     $: <> $1
R<> $* < @ [ $+ ] > $*  $1 < @ [ $2 ] > $3
R<> $* <$* : $* > $*    $#error $@ 5.1.3 $: "Colon illegal in host name part"
R<> $*                  $1
R$* < @ . $* > $*    $#error $@ 5.1.2 $: "Invalid host name"
R$* < @ $* .. $* > $*   $#error $@ 5.1.2 $: "Invalid host name"

# now delete the local info -- note $=O to find characters that cause forwarding
R$* < @ > $*            $@ $>Parse0 $>3 $1              user@ => user
R< @ $=w . > : $*   $@ $>Parse0 $>3 $2              @here:... -> ...
R$- < @ $=w . >         $: $(dequote $1 $) < @ $2 . >dequote "foo"@here
R< @ $+ >               $#error $@ 5.1.3 $: "User address required"
R$* $=O $* < @ $=w . >  $@ $>Parse0 $>3 $1 $2 $3   ...@here -> ...
R$-                     $: $(dequote $1 $) < @ *LOCAL* >    dequote "foo"
R< @ *LOCAL* >          $#error $@ 5.1.3 $: "User address required"
R$* $=O $* < @ *LOCAL* >
                        $@ $>Parse0 $>3 $1 $2 $3   ...@*LOCAL* -> ...
R$* < @ *LOCAL* >       $: $1

#
#  Parse1 -- the bottom half of ruleset 0.
#

SParse1
# handle numeric address spec
R$* < @ [ $+ ] > $* $: $>98 $1 < @ [ $2 ] > $3     numeric internet spec
R$* < @ [ $+ ] > $* $#esmtp $@ [$2] $: $1 < @ [$2] > $3    still numeric: send


# short circuit local delivery so forwarded email works
R$=L < @ $=w . >  $#local $: @ $1             special local names
R$+ < @ $=w . >         $#local $: $1                   regular local name


# resolve remotely connected UUCP links (if any)

# resolve fake top level domains by forwarding to other hosts



# pass names that still have a host to a smarthost (if defined)
R$* < @ $* > $*         $: $>95 < $S > $1 < @ $2 > $3       glue on smarthost name

# deal with other remote names
R$* < @$* > $*          $#esmtp $@ $2 $: $1 < @ $2 > $3             user@host.domain

# handle locally delivered names
R$=L                    $#local $: @ $1                 special local names
R$+                     $#local $: $1                   regular local names

########################################################################
###   Ruleset 5 -- special rewriting after aliases have been expanded   ###
########################################################################

S5

# deal with plussed users so aliases work nicely
R$+ + *                 $#local $@ $&h $: $1
R$+ + $*            $#local $@ + $2 $: $1 + *

# prepend an empty "forward host" on the front
R$+                     $: <> $1


# see if we have a relay or a hub
```

```
R< > $+                         $: < $H > $1                    try hub
R< > $+                         $: < $R > $1                    try relay
R< > $+                         $: < > < $1 $&h >               nope, restore +detail
R< > < $+ + $* > $*   < > < $1 > + $2 $3                        find the user part
R< > < $+ > + $*    $#local $@ $2 $: @ $1                       strip the extra +
R< > < $+ >                     $@ $1                           no +detail
R$+                             $: $1 $&h                       add +detail back in
R< local : $* > $*    $: $>95 < local : $1 > $2      no host extension
R< error : $* > $*    $: $>95 < error : $1 > $2      no host extension
R< $- : $+ > $+                 $: $>95 < $1 : $2 > $3 < @ $2 >
R< $+ > $+                      $@ $>95 < $1 > $2 < @ $1 >


######################################################################
###  Ruleset 95 -- canonify mailer:[user@]host syntax to triple     ###
######################################################################

S95
R< > $*                         $@ $1                           strip off null relay
R< error : $- $+ > $*           $#error $@ $(dequote $1 $) $: $2
R< local : $* > $*              $>CanonLocal < $1 > $2
R< $- : $+ @ $+ > $*<$*>$*   $# $1 $@ $3 $: $2<@$3>     use literal user
R< $- : $+ > $*                 $# $1 $@ $2 $: $3   try qualified mailer
R< $=w > $*                     $@ $2                           delete local host
R< $+ > $*                      $#relay $@ $1 $: $2             use unqualified mailer

######################################################################
###  Ruleset CanonLocal -- canonify local: syntax               ###
######################################################################

SCanonLocal
# strip trailing dot from any host name that may appear
R< $* > $* < @ $* . >           $: < $1 > $2 < @ $3 >

# handle local: syntax -- use old user, either with or without host
R< > $* < @ $* > $*             $#local $@ $1@$2 $: $1
R< > $+                         $#local $@ $1    $: $1

# handle local:user@host syntax -- ignore host part
R< $+ @ $+ > $* < @ $* >     $: < $1 > $3 < @ $4 >

# handle local:user syntax
R< $+ > $* <@ $* > $*           $#local $@ $2@$3 $: $1
R< $+ > $*                      $#local $@ $2    $: $1

######################################################################
###  Ruleset 93 -- convert header names to masqueraded form     ###
######################################################################

S93


# special case the users that should be exposed
R$=E < @ *LOCAL* >          $@ $1 < @ $j . >                leave exposed
R$=E < @ $=M . >  $@ $1 < @ $2 . >
R$=E < @ $=w . >  $@ $1 < @ $2 . >

# handle domain-specific masquerading
R$* < @ $=M . > $*$: $1 < @ $2 . @ $M > $3      convert masqueraded doms
R$* < @ $=w . > $* $: $1 < @ $2 . @ $M > $3
R$* < @ *LOCAL* > $*        $: $1 < @ $j . @ $M > $2
R$* < @ $+ @ > $* $: $1 < @ $2 > $3             $M is null
R$* < @ $+ @ $+ > $*        $: $1 < @ $3 . > $4             $M is not null

######################################################################
###  Ruleset 94 -- convert envelope names to masqueraded form   ###
######################################################################

S94
R$* < @ *LOCAL* > $*        $: $1 < @ $j . > $2

######################################################################
###  Ruleset 98 -- local part of ruleset zero (can be null)     ###
######################################################################

S98
```

```
# addresses sent to foo@host.REDIRECT will give a 551 error code
R$* < @ $+ .REDIRECT. >                 $: $1 < @ $2 . REDIRECT . > < ${opMode} >
R$* < @ $+ .REDIRECT. > <i>             $: $1 < @ $2 . REDIRECT. >
R$* < @ $+ .REDIRECT. > < $- >          $# error $@ 5.1.1 $: "551 User has moved; please try " <$1@$2>



##########################################################################
###  ParseRecipient --          Strip off hosts in $=R as well as possibly
###                             $* $=m or the access database.
###                             Check user portion for host separators.
###
###         Parameters:
###                 $1 -- full recipient address
###
###         Returns:
###                 parsed, non-local-relaying address
##########################################################################

SParseRecipient
R$*                             $: <?> $>Parse0 $>3 $1
R<?> $* < @ $* . > <?> $1 < @ $2 >              strip trailing dots
R<?> $- < @ $* >    $: <?> $(dequote $1 $) < @ $2 >                 dequote local part

# if no $=O character, no host in the user portion, we are done
R<?> $* $=O $* < @ $* >         $: <NO> $1 $2 $3 < @ $4>
R<?> $*                         $@ $1

R<NO> $* < @ $* $=R >           $: <RELAY> $1 < @ $2 $3 >
R<RELAY> $* < @ $* >            $@ $>ParseRecipient $1
R<$-> $*                $@ $2

##########################################################################
###  check_relay -- check hostname/address on SMTP startup
##########################################################################

SLocal_check_relay
Scheck_relay
R$*                             $: $1 $| $>"Local_check_relay" $1
R$* $| $* $| $#$*       $#$3
R$* $| $* $| $*                 $@ $>"Basic_check_relay" $1 $| $2

SBasic_check_relay
# check for deferred delivery mode
R$*                             $: < ${deliveryMode} > $1
R< d > $*               $@ deferred
R< $* > $*                      $: $2



##########################################################################
###  check_mail -- check SMTP `MAIL FROM:' command argument
##########################################################################

SLocal_check_mail
Scheck_mail
R$*                             $: $1 $| $>"Local_check_mail" $1
R$* $| $#$*                     $#$2
R$* $| $*               $@ $>"Basic_check_mail" $1

SBasic_check_mail
# check for deferred delivery mode
R$*                             $: < ${deliveryMode} > $1
R< d > $*               $@ deferred
R< $* > $*                      $: $2

R<>                             $@ <OK>
R$*                             $: <?> $>Parse0 $>3 $1                  make domain canonical
R<?> $* < @ $+ . > $*           <?> $1 < @ $2 > $3                      strip trailing dots
# handle non-DNS hostnames (*.bitnet, *.decnet, *.uucp, etc)
R<?> $* < $* $=P > $*           $: <OK> $1 < @ $2 $3 > $4
R<?> $* < @ $+ > $*             $: <? $(resolve $2 $: $2 <PERM> $) > $1 < @ $2 > $3
R<? $* <$->> $* < @ $+ > $*
                                $: <$2> $3 < @ $4 > $5

# handle case of @localhost on address
R<$+> $* < @localhost >         $: < ? $&{client_name} > <$1> $2 < @localhost >
```

```
R<$+> $* < @localhost.$m >
                        $: < ? $&{client_name} > <$1> $2 < @localhost.$m >
R<$+> $* < @localhost.UUCP >
                        $: < ? $&{client_name} > <$1> $2 < @localhost.UUCP >
R<? $=w> <$+> $*  <?> <$2> $3
R<? $+> <$+> $*         $#error $@ 5.5.4 $: "553 Real domain name required"
R<?> <$+> $*            $: <$1> $2


# handle case of no @domain on address
R<?> $*                 $: < ? $&{client_name} > $1
R<?> $*                 $@ <OK>                                 ...local unqualed ok
R<? $+> $*              $#error $@ 5.5.4 $: "553 Domain name required"
                                                                ...remote is not
# check results
R<?> $*                 $@ <OK>
R<OK> $*                $@ <OK>
R<TEMP> $*              $#error $@ 4.1.8 $: "451 Sender domain must resolve"
R<PERM> $*              $#error $@ 5.1.8 $: "501 Sender domain must exist"

##########################################################################
###  check_rcpt -- check SMTP `RCPT TO:' command argument
##########################################################################

SLocal_check_rcpt
Scheck_rcpt
R$*                     $: $1 $| $>"Local_check_rcpt" $1
R$* $| $#$*             $#$2
R$* $| $*       $@ $>"Basic_check_rcpt" $1

SBasic_check_rcpt
# check for deferred delivery mode
R$*                     $: < ${deliveryMode} > $1
R< d > $*       $@ deferred
R< $* > $*              $: $2

R$*                     $: $>ParseRecipient $1          strip relayable hosts


# anything terminating locally is ok
R$+ < @ $=w >           $@ OK
R$+ < @ $* $=R >  $@ OK


# check for local user (i.e. unqualified address)
R$*                     $: <?> $1
R<?> $+ < @ $+ >  $: <REMOTE> $1 < @ $2 >
# local user is ok
R<?> $+                 $@ OK
R<$+> $*        $: $2

# anything originating locally is ok
R$*                     $: <?> $&{client_name}
# check if bracketed IP address (forward lookup != reverse lookup)
R<?> [$+]               $: <BAD> [$1]
# pass to name server to make hostname canonical
R<?> $* $~P             $: <?> $[ $1 $2 $]
R<$-> $*        $: $2
R$* .                   $1                              strip trailing dots
R$@                     $@ OK
R$=w                    $@ OK
R$* $=R                 $@ OK

# check IP address
R$*                     $: $&{client_addr}
R$@                     $@ OK                   originated locally
R0                      $@ OK                   originated locally
R$=R $*                 $@ OK                   relayable IP address
R$*                     $: [ $1 ]           put brackets around it...
R$=w                    $@ OK                   ... and see if it is local


# anything else is bogus
R$*                     $#error $@ 5.7.1 $: "550 Relaying denied"

#
##########################################################################
```

```
######################################################################
#####
#####                    MAILER DEFINITIONS
#####
######################################################################
######################################################################


#####################################################
###   Local and Program Mailer specification   ###
#####################################################

#####  @(#)local.m4        8.30 (Berkeley) 6/30/98  #####

Mlocal,         P=/bin/mail, F=lsDFMAw5:/|@qSnE9, S=10/30, R=20/40,
                T=DNS/RFC822/X-Unix,
                A=mail -f $g -d $u
Mprog,          P=/bin/sh, F=lsDFMoqeu9, S=10/30, R=20/40, D=$z:/,
                T=X-Unix,
                A=sh -c $u

#
#  Envelope sender rewriting
#
S10
R<@>                    $n                      errors to mailer-daemon
R@ <@ $*>               $n                      temporarily bypass Sun bogosity
R$+                     $: $>50 $1              add local domain if needed
R$*                     $: $>94 $1              do masquerading

#
#  Envelope recipient rewriting
#
S20
R$+ < @ $* >            $: $1                   strip host part

#
#  Header sender rewriting
#
S30
R<@>                    $n                      errors to mailer-daemon
R@ <@ $*>               $n                      temporarily bypass Sun bogosity
R$+                     $: $>50 $1              add local domain if needed
R$*                     $: $>93 $1              do masquerading

#
#  Header recipient rewriting
#
S40
R$+                     $: $>50 $1              add local domain if needed

#
#  Common code to add local domain name (only if always-add-domain)
#
S50

#######################################
###   SMTP Mailer specification   ###
#######################################

#####  @(#)smtp.m4        8.38 (Berkeley) 5/19/98  #####

Msmtp,          P=[IPC], F=mDFMuX, S=11/31, R=21, E=\r\n, L=990,
                T=DNS/RFC822/SMTP,
                A=IPC $h
Mesmtp,         P=[IPC], F=mDFMuXa, S=11/31, R=21, E=\r\n, L=990,
                T=DNS/RFC822/SMTP,
                A=IPC $h
Msmtp8,         P=[IPC], F=mDFMuX8, S=11/31, R=21, E=\r\n, L=990,
                T=DNS/RFC822/SMTP,
                A=IPC $h
Mrelay,         P=[IPC], F=mDFMuXa8, S=11/31, R=61, E=\r\n, L=2040,
                T=DNS/RFC822/SMTP,
                A=IPC $h

#
#  envelope sender rewriting
```

```
#
S11
R$+                     $: $>51 $1              sender/recipient common
R$* :; <@>              $@                      list:; special case
R$*                     $: $>61 $1              qualify unqual'ed names
R$+                     $: $>94 $1              do masquerading


#
#  envelope recipient rewriting --
#  also header recipient if not masquerading recipients
#
S21
R$+                     $: $>51 $1              sender/recipient common
R$+                     $: $>61 $1              qualify unqual'ed names


#
#  header sender and masquerading header recipient rewriting
#
S31
R$+                     $: $>51 $1              sender/recipient common
R:; <@>                 $@                      list:; special case

# do special header rewriting
R$* <@> $*              $@ $1 <@> $2            pass null host through
R< @ $* > $*            $@ < @ $1 > $2          pass route-addr through
R$*                     $: $>61 $1              qualify unqual'ed names
R$+                     $: $>93 $1              do masquerading


#
#  convert pseudo-domain addresses to real domain addresses
#
S51

# pass <route-addr>s through
R< @ $+ > $*            $@ < @ $1 > $2          resolve <route-addr>

# output fake domains as user%fake@relay

# do UUCP heuristics; note that these are shared with UUCP mailers
R$+ < @ $+ .UUCP. >     $: < $2 ! > $1                  convert to UUCP form
R$+ < @ $* > $*         $@ $1 < @ $2 > $3       not UUCP form

# leave these in .UUCP form to avoid further tampering
R< $&h ! > $- ! $+   $@ $2 < @ $1 .UUCP. >
R< $&h ! > $-.$+ ! $+   $@ $3 < @ $1.$2 >
R< $&h ! > $+           $@ $1 < @ $&h .UUCP. >
R< $+ ! > $+            $: $1 ! $2 < @ $Y >     use UUCP_RELAY
R$+ < @ $+ : $+ >  $@ $1 < @ $3 >               strip mailer: part
R$+ < @ >               $: $1 < @ *LOCAL* >             if no UUCP_RELAY


#
#  common sender and masquerading recipient rewriting
#
S61

R$* < @ $* > $*         $@ $1 < @ $2 > $3       already fully qualified
R$+                     $@ $1 < @ *LOCAL* >             add local qualification

#
#  relay mailer header masquerading recipient rewriting
#
S71

R$+                     $: $>61 $1
R$+                     $: $>93 $1
```