Tese apresentada à Pró-Reitoria de Pós-Graduação e Pesquisa do Instituto Tecnológico de Aeronáutica, como parte dos requisitos para obtenção do título de Mestre em Ciências no Curso de Engenharia Eletrônica e Computação, Área de Informática

Alexandre de Barros Barreto

UMA ARQUITETURA DE TELEFONIA IP PARA PROTEÇÃO DA MÍDIA FIM-A-FIM

Tese aprovada em sua versão final pelos abaixo assinados:

Prof. Dr. Antonio Candido Faleiros
Orientador

Prof. Dr. Homero Santiago Maciel
Pró-Reitor de Pós-Graduação e Pesquisa

Campo Montenegro
São José dos Campos, SP - Brasil
2007

Dados Internacionais de Catalogação-na-Publicação (CIP)

Divisão Biblioteca Central do ITA/CTA

Barreto, Alexandre de Barros

Uma Arquitetura de Telefonia IP para Proteção da Mídia Fim-a-Fim / Alexandre de Barros Barreto.

São José dos Campos, 2007.

155f.

Tese de Mestrado – Curso de Engenharia Eletrônica e Computação. Área de Informática – Instituto Tecnológico de Aeronáutica, 2007. Orientador: Prof. Dr. Antonio Candido Faleiros. .

1. Telefonia IP. 2. Criptografia. 3. Multimídia. I. Centro Técnico Aeroespacial. Instituto Tecnológico de Aeronáutica. Divisão de Ciência da Computação. II. Título.

REFERÊNCIA BIBLIOGRÁFICA

BARRETO, Alexandre de Barros. **Uma Arquitetura de Telefonia IP para Proteção da Mídia Fim-a-Fim**. 2007. 155f. Tese de Mestrado – Instituto Tecnológico de Aeronáutica, São José dos Campos.

CESSÃO DE DIREITOS

NOME DO AUTOR: Alexandre de Barros Barreto

TITULO DO TRABALHO: Uma Arquitetura de Telefonia IP para Proteção da Mídia Fim-a-Fim.

TIPO DO TRABALHO/ANO: Tese / 2007

Alexandre de Barros Barreto

R. H9, bloco B, ap. 402 Campus do CTA

CEP 12.228-611 – São José dos Campos–SP

UMA ARQUITETURA DE TELEFONIA IP PARA PROTEÇÃO DA MÍDIA FIM-A-FIM

Alexandre de Barros Barreto

Composição da Banca Examinadora:

Prof. Dr.	Waldecir João Perrella	Presidente	-	ITA
Prof. Dr.	Antonio Candido Faleiros	Orientador	-	ITA
Prof. Dr.	Paulo Sérgio Licciardi Messeder Barreto	Membro Externo	-	USP
Prof. Dr.	Wagner Chiepa Cunha	Membro	-	ITA
Prof. Dr.	Tânia Nunes Rebello	Membro	-	ITA

ITA

Dedico essa tese a meu pai, que enquanto vivo me ensinou os princípios mais importantes que um filho pode aprender: *moral* e *honra*.

Agradecimentos

Primeiramente, gostaria de agradecer a Deus por ter me dado força e sabedoria para conduzir adequadamente mais essa etapa de minha vida profissional.

A meus pais (Isnei e Vera) por terem me dado a oportunidade de estudar e me ensinado os conceitos de honra e moral que ilumina meus passos.

A minha esposa e querida filha por terem compreendido a minha ausência e ter me apoiado nos momentos mais complexos e difíceis dessa jornada.

Ao Prof. Dr. Antônio C. Faleiros, pela orientação e confiança depositada na realização deste trabalho.

"A ciência tem as raizes amargas mas os frutos são muito doces."
— Aristóteles

Resumo

O uso da tecnologia de voz sobre IP (VoIP) vem se tornando cada vez mais comum no mundo todo. No entanto, a implantação dessa tecnologia associada à necessidade de garantia de sigilo do canal compatível a sua precursora, a telefonia comutada, ainda é um desafio. Apesar de haver certo consenso em como prover o transporte seguro do dado multimídia transportado em uma conferência, ainda existe discussões de como efetuar a negociação de forma protegida das chaves que permitirão a criação do canal criptografado. Quando se deseja criar esse canal de forma independente de qualquer infra-estrutura de segurança existente, tais como nos cenários fim-a-fim, onde apenas os terminais envolvidos devem ser os responsáveis por essa proteção, as alternativas se resumem a apenas três: o uso dos protocolos S/MIME, MIKEY ou ZRTP. Apesar desses frameworks resolverem satisfatoriamente o problema, eles possuem limitações que os impedem de atender todos os cenários de implementação existentes. A presente dissertação propõe um novo modo de transporte para o MIKEY, o MIKEY-DHHMAC-SAS, que mantém as características principais do protocolo, mas adiciona outras existentes no ZRTP, solucionando assim as limitações dos protocolos existentes, podendo ser usado em qualquer conferência segura fim-a-fim.

Abstract

Voice over IP (VoIP) technology has become more common. However, its implementation associated to the necessity of guaranteeing the security of the transmission channel compatible to the commuted telephony, is still a challenge. Despite of existing some consensus on how to provide a safety transmission on a VoIP conference, some discussions still exist on how to provide a protected form of key exchange that allows the creation of the cryptography channel. When the scenario involves the creation of this channel independent of any existing security infrastructure, the alternatives are the S/MIME, MIKEY or ZRTP protocols. Although these protocols solve the problems cited, they have limitations as it does not allow them to attend to all existing implementation scenarios. This document introduces a new form of transmission to MIKEY, the MIKEY-DHHMAC-SAS, in which main characteristics from the MIKEY protocol are kept, but it adds some existing characteristics of the ZRTP.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

CSB	Crypto Session Bundle
HTTP	HyperText Transfer Protocol
IETF	Internet Engineering Task Force
IP	Internet Protocol
ISO	International Organization for Standardization
ITU-T	International Telecommunications Union-Telecommunication
MITM	Man-in-the-Middle
NAT	Network Address Translator
NSA	National Security Agency
NIST	National Institute of Standards and Technology
PABX	Private Automatic Branch eXchange
PSTN	Public Switched Telephone Network
RFC	Request For Comment
SDP	Session Description Protocol
SIP	Session Initiation Protocol
SMTP	Simple Mail Transport Protocol
SRTP/SRTCP	Secure Real Time Protocol/Secure Real Time Control Protocol
TEK	Traffic-Encrypting Key

TGK	TEK Generation Key
URL	Uniform Resource Locator
VoIP	Voz over IP

1 Introdução

Esforços em usar uma rede de pacote para trafegar voz e vídeo datam da década de 70, quando estudos do Network Working Group desenvolveram o protocolo *NVP* (*Network Voice Protocol*) (COHEN, 1977), cujo objetivo era trafegar voz sobre a então existente *ARPANET*. Porém a primeira iniciativa de sucesso em escala industrial foi dada pelo desenvolvimento da pilha de protocolos *H.323* pela *International Telecommunications Union-Telecommunication* (*ITU-T*), no ano de 1996. Esta pilha de protocolos apresentava uma adaptação dos padrões telefônicos existentes nas redes privadas de telefonia (*Public Switched Telephone Network - PSTN*) para a plataforma IP.

Com o desenvolvimento pela *Internet Engineering Task Force* (*IETF*) do padrão *SIP* (*Session Initiation Protocol*) que possuía como principais características a simplicidade e aderência a padrões já existentes na Internet, o padrão *H.323* foi substituído.

Apesar de os protocolos *SIP* e *H.323* possuírem estruturas de funcionamento bem diferentes, ambos usam a pilha *RTP/RTCP* como framework de transporte de mídia em tempo real. A pilha *RTP* surgiu em 1996 para prover uma camada de transporte ao então *NVP*, porém sua robustez e eficiência fizeram com que ela tenha sido adotada como protocolo padrão de transporte multimídia.

Apesar de pesquisas recentes mostrarem que a tecnologia de voz sobre IP (VoIP)

possui índices de crescimento expressivos de 112% no Brasil, mercado que representa 36% do segmento mundial de telefonia (TELEGEOGRAPHY, 2005), ainda não existe uma forma padronizada e universalmente aceita para realizar chamadas seguras onde a confidencialidade dos dados e a autenticação dos usuários sejam requisitos primários. Um requisito adicional a esse cenário é que o meio deve ser transparente à chamada, ou seja, toda a proteção deve ser somente garantida pelos terminais envolvidos, o que é chamado de proteção fim-a-fim.

Diferentemente da rede convencional de telefonia, as redes VoIP são baseadas na rede de pacotes IP, rede altamente descentralizada que, sabidamente, apresenta várias vulnerabilidades. Esse fato ocasiona a inexistência de um controle adequado sobre o canal por onde irá trafegar a voz da conversação. Isto faz com que, quando comparada a sua antecessora, seja vista com restrições para uso em cenários onde a segurança deve ser prezada.

Apesar da existência de alguns padrões que definem como proteger a chamada VoIP, eles são muito recentes e apresentam alguns problemas em sua implantação, tornando-os apenas adequados para cenários específicos. Esse fato resulta na não existência de uma análise ainda bem formada sobre a potencialidade do seu uso.

Um complicador para os projetistas de protocolos de segurança VoIP é que as proteções introduzidas não devem aumentar muito os parâmetros que definem a qualidade de uma chamada IP.

1.1 Objetivo

1.1.1 Objetivo Geral

O objetivo deste trabalho de Mestrado é prover uma arquitetura de negociação de chaves para sistemas baseados na tecnologia de telefonia IP fim-a-fim, ou seja, sem a necessidade de qualquer intermediação ou recurso da infra-estrutura. Essa arquitetura deve implementar as potencialidades já existentes, minimizando ou mesmo eliminando as suas vulnerabilidades e limitações.

1.1.2 Objetivos Específicos

Para esta pesquisa foram definidos os seguintes objetivos específicos:

- Identificar as arquiteturas de segurança VoIP existentes ou em desenvolvimento na IETF que implementam uma arquitetura de proteção fim-a-fim;
- Analisar as arquiteturas levantadas anteriormente, detectando as possíveis limitações e vulnerabilidades, bem como apreciar as potencialidades das mesmas;
- Propor uma nova arquitetura de segurança fim-a-fim para VoIP; e
- Desenvolver uma implementação de referência que possibilite analisar as principais características da nova arquitetura.

1.1.3 Limitação de Escopo

Nem todos os protocolos e ambientes de telefonia IP serão abordados no presente documento, que se deterá apenas naqueles que usam a especificação SIP (ROSENBERG

et al., 2002) como meio de sinalização, uma vez que ele é o padronizado pela IETF (Internet Engineering Task Force), órgão que padroniza os protocolos que funcionam na Internet.

1.2 Estruturação da Tese

- **Capítulo 1 - Introdução:** Contém uma introdução do trabalho, onde são expostos o objetivo e a motivação do trabalho;
- **Capítulo 2 - Conceitos de Telefonia IP:** apresenta os componentes necessários para uma comunicação por voz, tendo como foco descrever a arquitetura dos protocolos SIP e a pilha RTP/RTCP;
- **Capítulo 3 - Requisitos para a Proteção da Mídia:** serão abordadas as questões referentes à segurança em VoIP, onde se definirá os conceitos básicos de segurança e criptografia que permearão todo o documento. Também serão descritos os requisitos de segurança e qualidade definidos na IETF e ITU-T e metodologias existentes para a avaliação dos mesmos;
- **Capítulo 4 - Arquiteturas de Segurança VoIP Fim-a-Fim:** serão discutidas as arquiteturas de segurança fim-a-fim desenvolvidas para prover segurança a uma conferência de telefonia IP;
- **Capítulo 5 - MIKEY-DHHMAC-SAS:** Extensão ao MIKEY: é apresentada a arquitetura desenvolvida na presente pesquisa que tem por finalidade a resolução do problema apontado na introdução do presente trabalho; e
- **Capítulo 6 - Conclusão:** apresenta uma conclusão do estudo realizado e apresenta

algumas recomendações para o desenvolvimento de trabalhos futuros.

2 Conceitos de Telefonia IP

Para ser possível o estabelecimento de um canal de voz entre dois usuários através da arquitetura de Voz sobre IP (VoIP) é necessário que uma série de processos sejam realizados e que diversas entidades interajam entre si.

Para entender como se deve implementar qualquer mecanismo de segurança nesse ambiente, é requisito primário a correta compreensão dos processos envolvidos e do papel de cada elemento participante da arquitetura.

Dessa forma, o presente capítulo tem por finalidade apresentar a arquitetura básica de funcionamento da telefonia IP. Para isso, inicialmente serão abordados alguns conceitos gerais acerca de redes telefônicas e, posteriormente, a estrutura dos protocolos SIP e RTP/RTCP.

2.1 Funcionamento Básico de um Sistema Telefônico

Segundo (RANSOME; RITTINGHOUSE, 2005), a telefonia IP refere-se aos sistemas que possibilitam o transporte de voz, vídeo, texto e qualquer outro tipo de mídia de tempo real através de uma rede de pacotes IP.

Apesar de alguns autores (AVAYA, 2006) distinguirem os conceitos de VoIP e Telefonia

IP, neste documento os dois termos serão considerados sinônimos.

As redes VoIP seguem um modelo de funcionamento muito similar ao existente nas redes telefônicas tradicionais, onde o seu pilar básico é o conceito de sinalização. Conforme definido em (OLSSON, 2000), a sinalização é o processo de comunicação que coordena a troca de mensagens de controle entre os diversos elementos que participam de uma comunicação, fazendo com que eles obtenham as informações sobre as capacidades e localização de seus pares, viabilizando a criação e manutenção de um canal de dados (voz, vídeo ou texto).

A sinalização é um processo constituído por quatro fases. A primeira, denominada fase livre, demarca o momento em que os terminais pertencentes ao canal de comunicação estão em repouso, ou seja, não possuem nenhuma conexão multimídia aberta.

A fase livre finaliza no momento em que um dos terminais decide estabelecer uma conferência com algum dos pares existentes na rede. Para que isso ocorra é necessário a determinar a localização física, a disponibilidade e os recursos multimídias existentes no terminal destino. A fase da sinalização que realiza esse conjunto de tarefas é a de estabelecimento de comunicação.

Uma vez realizada corretamente a fase anterior, as estações multimídia estão aptas a iniciar a comunicação e a transferir dados (voz ou vídeo) entre si. Estas tarefas constituem a fase de transferência.

A última fase a ser realizada no processo de sinalização é a de liberação. Esta fase é muito importante para o processo como um todo, pois sem ela os recursos utilizados durante uma chamada não seriam liberados, ocasionando uma sobrecarga indevida do sistema e, desta forma, exigindo um super-dimensionamento da infra-estrutura necessária

para realizar a sinalização.

O processo de sinalização e as tarefas realizadas em cada uma de suas fases são apresentadas na figura 2.1.

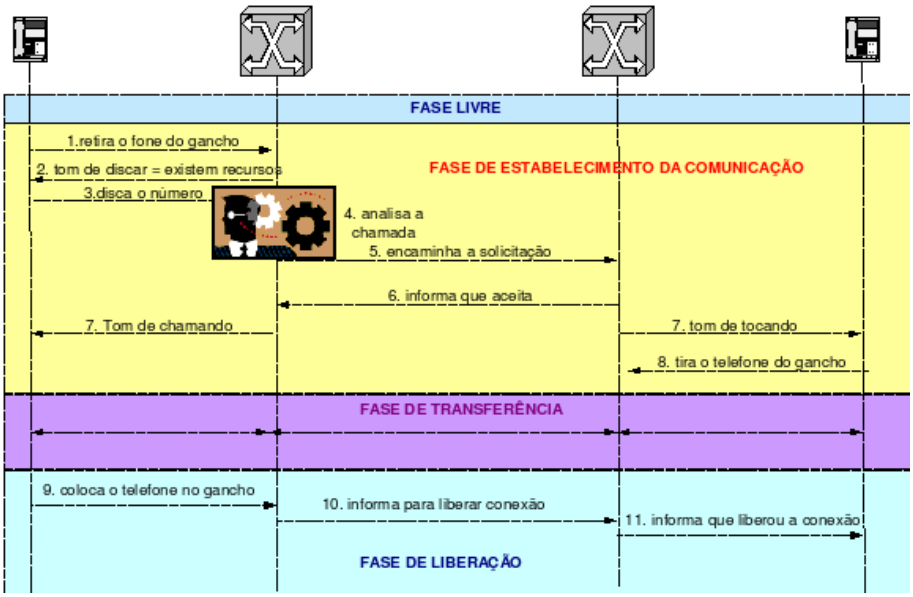

FIGURA 2.1 – Processo de Sinalização.

Ao analisar a figura 2.1, percebe-se que, além dos terminais multimídia, existe um outro elemento muito importante para o funcionamento da rede telefônica como um todo: o *Private Automatic Branch eXchange* (PABX). O PABX é a entidade central de uma rede telefônica convencional. Ele tem como responsabilidade ser uma interface entre os terminais multimídias localizados localmente e a rede telefônica mundial, promovendo o correto encaminhamento das chamadas locais e remotas recebidas e gerenciando o acesso dos terminais locais aos recursos externos a seu domínio.

Uma característica marcante das redes telefônicas convencionais é que seus sistemas PABX são responsáveis pela realização da maior parte das suas funcionalidades. Na maioria dos casos, os terminais telefônicos são apenas entidades capazes de informar à rede o destino desejado e direcionar o dado multimídia ao PABX. É o PABX o responsável por todas as tarefas importantes da rede. Dentre elas, pode-se citar a localização física dos

terminais envolvidos, a negociação de parâmetros, a conversão da mídia para um formato adequado para transmissão e até mesmo o próprio transporte da voz fim-a-fim.

Uma outra característica das redes telefônicas convencionais está no fato de serem baseadas em uma tecnologia comutada de circuitos. Segundo (SOARES; LEMOS; COLCHER, 1995), essa tecnologia, apresentada em 2.2, consiste em uma rede onde os canais de comunicação entre os pares envolvidos no processo são dedicados. Isto significa que, para dois terminais conversarem é necessário existir um canal que os interligue e, durante a conferência, esse canal ficará indisponível para qualquer outro uso.

A exclusividade do canal durante uma conferência em uma rede comutada possui vantagens e desvantagens. Como vantagem, destaca-se a garantia assegurada ao terminal iniciador de que a rede terá todos os meios necessários para a realizar a conferência com qualidade, durante todo o período da chamada. Porém, esse tipo de rede exisge um super-dimensionamento da infra-estrutura de telefonia, uma vez que ela precisa deixar canais ociosos para conseguir atender as demandas de todos os terminais, mesmo àqueles que não usam constantemente a rede. Esta necessidade pode ser extremamente onerosa para o mantenedor desse tipo de arquitetura.

A grande diferença entre as redes VoIP e as redes telefônicas tradicionais está na técnica de comutação utilizada, onde, no caso das redes de telefonia IP, é usado uma técnica de comutação de pacotes ou datagramas ao invés de circuitos. Nesse tipo de redes (apresentado em 2.2) não é necessário o estabelecimento de um canal dedicado para que ocorra uma comunicação entre dois pares (SOARES; LEMOS; COLCHER, 1995), ao invés disso, a estação que deseja se comunicar, adiciona o endereço do par remoto na mensagem a ser trafegada e esta informação deverá ser lida em todos os elementos intermediários da rede, que são responsáveis por encaminhar corretamente a mensagem até o seu destino.

Nesse tipo de técnica de comutação, as informações (sinalização ou multimídia) são encapsuladas dentro de pequenas mensagens e somente assim encaminhadas, eliminando a necessidade de haver um canal dedicado.

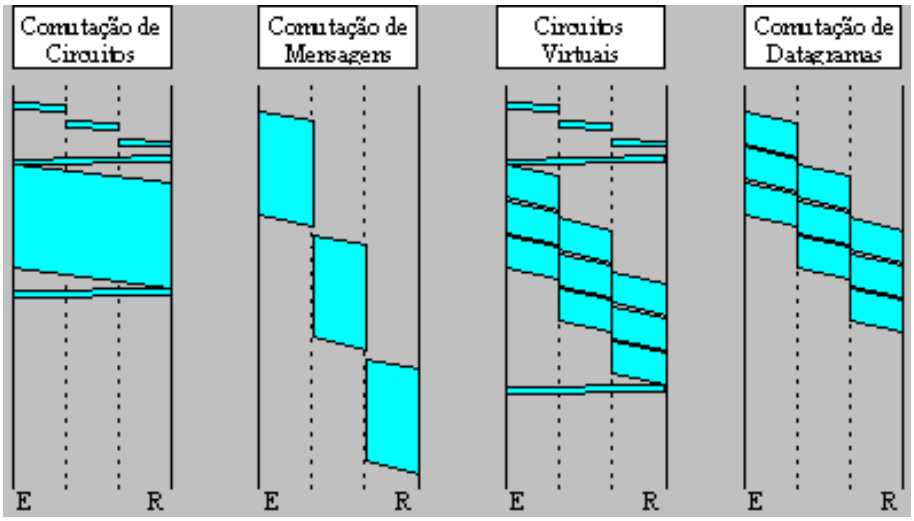

FIGURA 2.2 – Técnicas de Sinalização. Fonte (MOREIRA, 2007)

A figura 2.2 apresenta as diversas técnicas de comutação existentes em (SOARES; LEMOS; COLCHER, 1995). Nela deve-se atentar apenas para as técnicas de comutação de circuito e de pacotes (ou datagramas), técnicas usadas respectivamente nas redes telefônicas tradicionais e VoIP.

Na figura citada percebe-se que, na comutação de circuito, após as primeiras mensagens (fase de estabelecimento da comunicação), o circuito é representado com uma figura contínua, indicando a exclusividade do canal durante a existência da conferência. Já na representação da comutação de pacotes, percebem-se pequenos segmentos de mensagens que são trocadas durante toda a existência do canal.

O uso de pequenas mensagens (pacotes) para realizar a troca de informações em um canal, apesar de tratar melhor a questão do dimensionamento da rede telefônica, possui um problema de solução complexa. Uma vez que não existe reserva de recursos para construír o canal por onde os dados serão trafegados, pode-se, ocasionalmente, construir

um canal sem as condições mínimas para realizar uma conferência com qualidade na voz trafegada.

Por se basear em uma rede de pacotes, as arquiteturas VoIP podem apresentar pequenas diferenças na implementação do seu processo de sinalização. Boa parte das arquiteturas possuem protocolos diferentes para tratar as fases de negociação de parâmetros (livre, estabelecimento da conexão e liberação) e as de transferência dos dados multimídia.

A segunda diferença é que muitas das tarefas desenvolvidas nessas fases são realizadas pelos próprios terminais, que, diferentemente dos existentes nas redes convencionais, possuem várias responsabilidades e, normalmente, bastante poder de processamento.

Por fim, a última diferença existente em uma rede VoIP consiste no fato que os elementos intermediários, apesar de realizarem tarefas de apoio ao processo de sinalização, não são obrigatoriamente necessários em todos os cenários de utilização da tecnologia.

Apesar de (OLSSON, 2000) considerar como parte da sinalização telefônica a transferência do dado multimídia (fase de transferência de dados), no escopo da presente tese, o termo sinalização irá se referir apenas às fases anterior e posterior à transferência do dado multimídia, que constituem as tarefas de apoio para a construção e manutenção do canal multimídia.

Conforme limitação de escopo citado em 1.1.3, serão estudadas apenas as arquiteturas VoIP fim-a-fim implementadas através dos protocolos SIP (ROSENBERG *et al.*, 2002), SDP (HANDLEY; JACOBSON; PERKINS, 2006) e das pilhas RTP/RTCP (SCHULZRINNE *et al.*, 2003), mesmo existindo outros protocolos que implementam uma conferência com base nessa tecnologia.

2.2 O Session Initiation Protocol - SIP

Segundo (ROSENBERG *et al.*, 2002), o Session Initiation Protocol (SIP) é um protocolo localizado na camada de aplicação do modelo TCP/IP cujo objetivo é o de estabelecer, modificar e terminar conferências multimídias entre dois ou mais usuários. Esse conceito define de forma bem clara que o SIP não é responsável por realizar o transporte multimídia, ou seja, pela fase de transferência de dados, mas sim pela negociação dos parâmetros necessários para a construção do canal entre os pares da comunicação.

O SIP tem sua estrutura baseada em dois protocolos bastante comuns na Internet, o HyperText Transfer Protocol (HTTP) e o Simple Mail Transport Protocol (SMTP). Do HTTP ele herda o seu esquema de funcionamento baseado em requisições e respostas, onde todas as transações necessárias para a construção da conferência são inicializados através de uma requisição, que deverá ser respondida por uma ou maismensagens.

Do SMTP, ele herda seu esquema de endereçamento. O SIP usa uma extensão do padrão Uniform Resource Locator (URL) . Nesse padrão, todo recurso que pertence à Internet deve ter sua localização determinada de forma única por um apelido, possibilitando assim que usuários possam recuperar as informações existentes nesses objetos assim identificados.

A URL é formada por três partes: o nome do protocolo a ser usado, o nome do recurso e, por fim, o domínio (rede) onde ele está localizado. Um exemplo de SIP URL é dado na figura 2.3, onde pode-se ver que o protocolo é o sip, o nome do recurso é kabart e o domínio é ita.br.

Conforme dito anteriormente, o SIP implementa as fases relacionadas à sinalização telefônica, que compreendem as fases de estabelecimento e encerramento da comunicação.

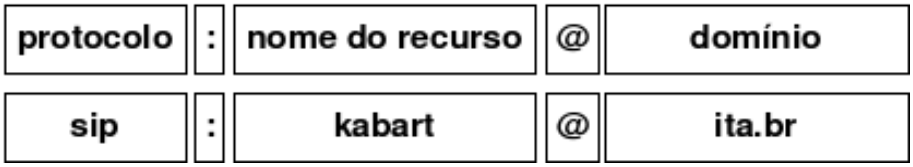

FIGURA 2.3 – Exemplo de URL SIP

Para isso ele usa um esquema de requisições e respostas, onde a requisição identifica o tipo de fase que o terminal iniciador deseja empreender.

A figura 2.4 apresenta esquematicamente o funcionamento do SIP. Nela se pode observar o processo de troca de mensagens (requisições e respostas) entre as diversas entidades participantes no processo de estabelecimento de uma conferência.

Para um completo entendimento do processo apresentado na supra-citada figura, alguns conceitos anteriores devem ser esclarecidos. O primeiro deles é o de *User Agent*, que é qualquer entidade capaz de enviar ou receber requisições em uma rede VoIP, ou seja, qualquer elemento que tenha participação ativa no processo de construção de uma conferência multimídia.

Os User Agent podem ser de dois tipos. Eles serão um *User Agent Client* (UAC), quando eles são os responsáveis por criar a requisição inicial e enviar ao outro terminal. Ao terminal que recebe a requisição dá-se o nome de *User Agent Server* (UAS).

Outro conceito importante definido em (ROSENBERG *et al.*, 2002) é o que define os tipos de UA existentes na arquitetura. Apesar de existirem muitos tipos, basta explicar o conceito de *proxy* para entender a figura 2.4.

Um servidor proxy consiste em um UA responsável por intermediar as transações SIP entre os pares. Um proxy fundamentalmente recebe uma requisição de um terminal iniciador (UAS), localiza o seu destino e direciona para o terminal destino essa mensagem (UAC). Assim que o destino responde à requisição inicial, o proxy redireciona essa resposta

para o terminal de origem, fazendo com que durante todo o processo de sinalização, os terminais nunca interajam diretamente.

Adicionalmente, o proxy é responsável por autenticar os diversos terminais SIP pertencentes à sua rede e gerenciar o acesso aos recursos existentes na rede como, por exemplo, o acesso a outras redes não SIP como a PSTN.

De posse dos conceitos anteriormente apresentados é possível entender a figura 2.4. Nela o processo é iniciado através de uma requisição enviada pelo terminal à esquerda da figura, o terminal iniciador, para o proxy responsável pela gerência da rede local à qual ele pertence. Essa requisição é do tipo *INVITE*, responsável por identificar na rede o terminal com o qual aquele que inicia deseja realizar uma conferência multimídia.

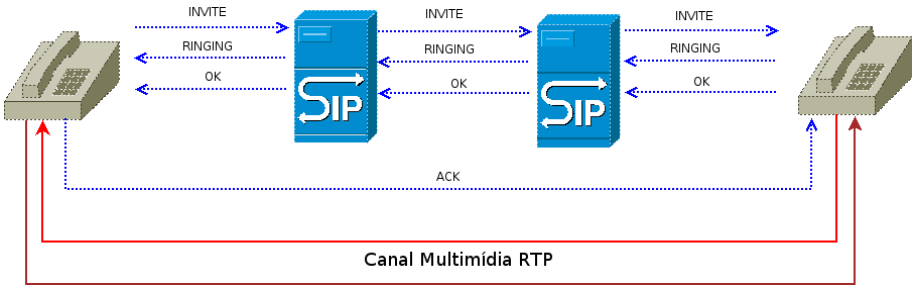

FIGURA 2.4 – Processo de Estabelecimento de uma Chamada SIP

Entre outras informações existentes dentro de uma requisição *INVITE*, existem os parâmetros com os quais o terminal iniciador deseja realizar a conferência e o endereço do terminal destino. Nesses parâmetros pode-se especificar qual é o protocolo de transporte desejado, os tipos de canais multimídia com os quais ele deseja realizar a conferência (voz, vídeo e/ou texto), os CODEC's utilizados para a conversão do dado digital em um áudio analógico, as portas em que ele estará aguardando receber os canais multimídia anteriormente citados, bem como o seu endereço final, visto que durante a maior parte da sinalização sua conversa com o outro par será intermediada sempre por um ou mais proxy's existentes na rede SIP.

Ao receber a requisição *INVITE*do terminal iniciador, o proxy local irá verificar as credenciais do seu usuário, checando se ele possui ou não permissão para realizar a chamada desejada. Após isso, o proxy irá analisar o endereço do terminal destino, averiguando se ele pertence à sua rede local. Se este for o caso,direcionará a chamada para o endereço local que está sendo chamado.

No caso da figura 2.4, o terminal destino não pertence à rede do proxy do originador da chamada. Neste caso ele tem que localizar qual o endereço físico do proxy responsável pela rede do terminal destino e para lá encaminhar a requisição.

Ao receber a requisição *INVITE*, o proxy da rede destino verificará o endereço lógico do terminal chamado e para ele encaminhará a requisição.

O terminal destino, no momento que recebe a requisição *INVITE*, deverá responder ao seu proxy o seu estado, informando se está disponível ou não para realizar a conferência solicitada. Essa informação é realizada através de uma mensagem padronizada pelo framework SIP, sendo do tipo *180 Ringing* no caso de estar disponível (ver figura), ou *486 Busy Here*, quando o terminal estiver ocupado com alguma chamada anteriormente criada. Essa resposta e outras requisições que poderão vir posteriormente serão sempre gerenciadas pelos proxys intemediários.

Para que os UA's consigam associar uma requisição a suas diversas respostas, o SIP identifica as suas mensagens com um rótulo de *transação*. Uma transação consiste em um conjunto de operações iniciadas por uma requisição que invoca um determinado método e que ocasiona uma série de respostas, onde uma delas é responsável por finalizar esse processo. No caso de uma conferência como a apresentada na figura 2.4, a transação é iniciada pela requisição *INVITE* e finalizada pela resposta *200 OK*.

Se o terminal iniciador receber uma resposta do tipo *486 Busy Here*, ele deverá encerrar a transação criada pela requisição inicial. Recebendo uma mensagem *180 Ringing*, ele deverá aguardar pois em breve receberá outra resposta do tipo *200 OK*.

O terminal destino, ao receber a requisição *INVITE* e enviar a resposta que está disponível para a conferência, deverá analisar os parâmetros enviados pelo iniciador e verificar se ele suporta todos eles. No caso de não suportar um dos parâmetros o terminal deverá enviar uma mensagem de erro. Caso suporte os parâmetros solicitados pelo iniciador, mas não concorde com algum deles, deverá informar isso em sua resposta *200 OK* que, além dessas informações, conterá os parâmetros específicos do destino para concretizar a conferência. As mensagens poderão conter informações sobre as portas de comunicação e o endereço físico do usuário.

Ao enviar a mensagem OK, o terminal destino deverá abrir o seu canal multimídia local, preparando-se para receber os dados enviados por seu par.

O iniciador deverá, ao receber a mensagem *200 OK*, analisá-la para recuperar os diversos parâmetros do terminal destino. Encerrando esta tarefa, deverá abrir seu canal multimídia local destinado ao recebimento dos pacotes multimídias enviados pelo terminal destino.

Nota-se que são criados dois canais multimídias, pois conforme será visto em 2.4, eles são unidirecionais. Outra informação importante é que as informações trafegadas por esses canais são fim-a-fim, ou seja, não são intermediadas pelos proxys usados durante o processo de sinalização.

Para garantir que os terminais tenham recebido adequadamente todas as mensagens referentes à transação criada com a requisição *INVITE*, é enviado pelo terminal iniciador,

após receber a mensagem *200 OK*, uma nova requisição do tipo *ACK*. Essa requisição é muito importante no processo de estabelecimento de uma conferência, pois no caso de ter sido extraviada a resposta *200 OK*, o terminal destino nunca iria saber dessa situação e ficaria com um canal multimídia aberto desnecessariamente. Sendo assim, no caso de o terminal destino não receber uma requisição *ACK* depois de um determinado tempo, ele deverá providenciar o fechamento do canal multimídia.

Para possibilitar a identificação da conferência e posteriormente encerrá-la corretamente, o SIP utiliza o conceito de *diálogo*. Um diálogo consiste na relação fim-a-fim existente entre dois terminais SIP, que é iniciada através de uma requisição do tipo *INVITE* e somente é finalizada quando uma das partes deseja terminar a sessão.

Esse conceito permite que um dos pares finalize a conferência desejada sem haver o risco de finalizar outra, considerando-se que um terminal pode possuir várias conferências abertas simultaneamente.

Para finalizar uma conferência é usada uma nova requisição, a *BYE*. Essa requisição é criada pelo terminal que deseja encerrar a transação e deve ser respondida pelo outro par com uma resposta *200 OK*. Assim como todas as mensagens SIP, essa é intermediada pelos proxys existentes nas redes do iniciador e destino. Sua maior importância reside no fato de anunciar a todos os envolvidos na conferência que podem desmobilizar os recursos usados em sua construção, liberando-os para a realização de outras tarefas.

Outro tipo de requisição importante existente no framework SIP é a *REGISTER*. Esse tipo de requisição permite que os diversos terminais sejam autenticados nos proxys responsáveis pela gestão do domínio em que estão localizados. Apesar dessa importância, esse tipo de requisição não será abordado no presente documento, uma vez que ele não interfere no modelo de segurança abordado no mesmo.

Conforme já comentado, o funcionamento da arquitetura SIP usa um modelo baseado em requisições que geram uma ou mais respostas. Sendo assim, para ser possível entender a plenitude do protocolo, é importante entender o formato de suas mensagens.

Uma requisição possui em sua constituição um identificador de tipo, alguns cabeçalhos e o corpo da mensagem. A linha de requisição define de forma explícita o tipo da mensagem (INVITE, REGISTER, ACK, BYE, etc), sendo a primeira linha a ser lida pelas entidades SIP, pois é ela que definirá o algoritmo a ser usado para realizar a análise da mensagem.

Já uma resposta possui, no lugar da linha de requisição, uma linha de status (200, 180, 486, etc), mantendo no resto de sua estrutura total similaridade com a arquitetura de uma requisição. Assim como a linha de requisição, a linha de status é de suma importância, pois nela se define a resposta da UAS à requisição enviada, norteando a ação a ser desenvolvida pela UAC.

Enquanto as linhas de requisição e de status definem o comportamento da UA, os cabeçalhos definem o roteamento correto e a parametrização do canal a ser criado pelos terminais. Um cabeçalho possui o formato *nome do cabeçalho=valor*, onde os mais comuns usados pelo protocolo SIP são apresentados a seguir.

Entre os cabeçalhos existentes em todas as mensagens SIP estão o *From* e o *To*, que identificam, respectivamente, o alias do UAC (terminal que criou a transação) e do UAS.

Outro cabeçalho bastante comum e importante é o Via, que determina a rota que a resposta a uma requisição deve seguir. Isso é importante para garantir que as requisições e respostas percorram o mesmo caminho, garantindo assim que seja possível bilhetar e tarifar uma chamada VoIP corretamente.

Uma vez que o protocolo SIP possui a possibilidade de ser estendido, atualmente

existem especificações (RFC) que implementam funcionalidades adicionais ao protocolo (instant message, segurança, etc), é necessário que, ao enviar uma requisição, o terminal possa determinar quais são as funcionalidades mínimas que ele está disposto a disponibilizar para realizar a conferência. Para isso ele deve usar o cabeçalho *Require*.

Um outro cabeçalho que deve ser comentado é o *Content-Type* que define o protocolo a ser usado pelo SIP para descrever a sessão. Entende-se como descrição da sessão qualquer informação que define os parâmetros fim-a-fim necessários para a construção do canal multimídia. Apesar de (ROSENBERG *et al.*, 2002) deixar em aberto o protocolo a ser usado, por vezes, ele deixa claro a preferência pelo uso do Session Description Protocol (SDP), que será utilizado durante o transcorrer do presente documento.

Para identificar que está sendo usado o protocolo SDP, o cabeçalho *Content-Type* deverá possuir um valor igual a *application/sdp*.

É importante mencionar a diferença o papel do SIP e do SDP. Basicamente o SDP provê ao SIP os meios necessários para anunciar os parâmetros pelos quais os terminais envolvidos na comunicação desejam estabelecer o canal multimídia. Desta forma, o SIP concentra-se apenas no correto encaminhamento das mensagens de sinalização, onde dentre outras informações, estarão os dados SDP, fazendo com que essas mensagens cheguem aos seus destinatários finais.

2.3 Session Description Protocol- SDP

Durante o processo de configuração de uma sessão multimídia, os terminais necessitam trocar informações sobre as suas capacidades e as configurações que desejam usar durante a conferência a ser criada. Apesar de a especificação do SIP prever outras formas

TABELA 2.1 – Atributos SDP.

Cabeçalho	**Descrição**
v*	Define a versão do protocolo SDP a ser usado.
o*	Identifica o proprietário da sessão.
s*	Define um nome para a sessão.
i	Apresenta um texto informativo sobre a sessão.
c	Campo que define o endereço onde a sessão deve ser criado.
b	Informa a banda disponível para o canal multimídia a ser criado.
z	Tem a finalidade de ajustar o clock dos terminais.
a	Define um atributo da sessão.
t*	Apresenta o tempo que a sessão está ativa.
m*	Descreve um canal de mídia a ser criado.

para realizar a negociação dos parâmetros multimídia entre os pares, na presente tese será utilizado o protocolo Session Description Protocol (SDP) (HANDLEY; JACOBSON; PERKINS, 2006).

Para realizar sua tarefa, o SDP possui uma diversidade de cabeçalhos que possibilitam aos terminais definirem de forma padronizada a mídia a ser transportada, o seu protocolo de transporte, informações relacionadas à temporização do canal, além da negociação de parâmetros necessários para a criação de sessões privadas.

Uma mensagem SDP, assim como a SIP, é simples de ser entendida. Consiste de um conjunto de informações do tipo *atributo=valor*, onde o atributo define a propriedade da conferência a ser descrita seguido pelo seu respectivo valor.

Na tabela 2.1 são apresentados os principais atributos que podem ser definidos no SDP e uma breve descrição de cada um. Apesar de todos os atributos serem importantes para descrever a mídia, apenas alguns são de uso obrigatório e serão apresentados abaixo com o símbolo (*).

Entre os atributos definidos em (HANDLEY; JACOBSON; PERKINS, 2006) pode-se comentar que um dos mais importantes é o que descreve a mídia, o atributo *media (m)*.

É esse o atributo que vai identificar para o par remoto a porta de dados e o protocolo no qual o terminal local deseja realizar a conferência multimídia.

Outro atributo de igual importância é o owner (o), que define o proprietário da sessão. Nesse atributo será definido o endereço IP real do usuário. Esta informação é importante pois os cabeçalhos SIP apenas trafegam a URL SIP do terminal que identifica ao par remoto o alias do usuário e o domínio no qual ele está inserido, mas não seu endereço IP real. A única entidade de uma rede SIP que possui essa informação é o proxy local do usuário. Todavia, ele não participa da fase de transferência de dados multimídia pois, conforme especificado, esta tarefa é realizada entre os terminais envolvidos diretamente. Sendo assim, o responsável por informar ao par remoto o real endereço que ele está usando é o atributo *(m)*.

Um dos cabeçalhos mais poderosos do SDP é atributo *(a)*, pois é ele que permite a extensão do SDP sem a necessidade de alterar o núcleo do protocolo. O formato geral desse atributo é: *a=<attribute>:valor*, onde *attribute* é o nome do atributo SDP estendido.

O atributo *a* pode ser usado para várias finalidades. A própria especificação do SDP apresenta uma diversidade de extensões já padronizadas para seu uso. Um exemplo pode ser apreciado na figura 2.5, onde é se apresenta o uso do atributo *a=rtpmap*, que possibilita ao par em comunicação a opção da escolha dinâmica do formato de mídia que deseja usar em uma conferência. Essa escolha é considerada dinâmica, pois cabe ao terminal, ao receber a mensagem SDP, escolher, dentre as opções ofertadas, a que melhor lhe convier.

A figura 2.5 apresenta um fragmento de uma mensagem SDP que informa a outro terminal remoto o desejo de quem chama de realizar uma conferência de áudio, na porta 49230, usando o protocolo RTP/AVP (ver seção 2.4), além de permitir o uso de qualquer formato de CODEC entre os 96, 97 ou 98, que são definidos no protocolo RTP/RTCP

```
m=audio 49230 RTP/AVP 96 97 98
a=rtpmap:96 L8/8000
a=rtpmap:97 L16/8000
a=rtpmap:98 L16/11025/2
```

FIGURA 2.5 – Atributos SDP

(SCHULZRINNE *et al.*, 2003), que será a seguir.

2.4 Real Time Protocol / Real Time Control Protocol - RTP/RTCP

No decorrer da presente tese, foram apresentados os protocolos SIP e SDP que possibilitam a terminais multimídias estabelecerem um canal de comunicação entre si. Esses protocolos são importantes e determinantes para que esse canal seja criado de forma eficiente. Entretanto, eles não implementam o canal de dados multimídia fim-a-fim entre os pares pois essa tarefa é a função da pilha de protocolos RTP/RTCP (SCHULZRINNE *et al.*, 2003).

Inicialmente, poder-se-ia pensar em usar os protocolos TCP ou UDP para a realização desse transporte, uma vez que eles são especificados para essa tarefa pelo IP. Todavia, ambos possuem problemas que os tornam inadequados para realizar a tarefa de transportar dados em tempo real.

Conforme (PERKINS, 2003) o uso do TCP em aplicações multimídia possui algumas vantagens não oferecidas pelo UDP. O TCP é confiável, pois garante a entrega do pacote através da confirmação do seu recebimento. Entretanto, a sobrecarga (overhead)

ocasionada pelo processo de garantia em um ambiente onde a banda é pequena, nem sempre garante a disponilidade dos recursos necessários para o perfeito funcionamento da aplicação (redes best-effort), fato muito prejudicial para o usuário.

Outra opção existente na especificação do protocolo IP seria a utilização o UDP. Todavia, é necessário manter algum controle sobre a entrega dos pacotes enviados, funcionalidade inexistente no UDP, tornando-o inadequado para esse fim.

Diante deste dilema e da inexistência de um protocolo adequado para transportar os dados em tempo real, a *Internet Engineering Task Force (IETF)* desenvolveu, em 1996, a especificação da pilha de protocolos RTP/RTCP (Real Time Protocol / Real Time Control Protocol), atualizando-a em 2003 (SCHULZRINNE *et al.*, 2003).

De forma resumida, o RTP cuida do transporte da mídia com o mínimo overhead possível, muito semelhante ao UDP, enquanto o RTCP cria um canal de controle do tráfego RTP, gerando estatísticas de qualidade e garantia parcial de entrega, possibilitando assim algum controle sobre a mídia, tal como no TCP.

Uma das principais características da pilha de protocolos RTP/RTCP reside em sua arquitetura fim-a-fim (PERKINS, 2003), ou seja, ele abstrai a complexidade da rede e dá aos terminais envolvidos na conferência a inteligência necessária para desenvolver a conversação. Essa arquitetura impede a interferência direta da rede na aplicação, evitando a necessidade de prepará-la para comportar uma conferência baseada nesse protocolo.

Sendo o RTP/RTCP um *framework* com o objetivo de atender aos mais variados tipos de dados multimídia em uma rede de pacotes, ele foi organizado em perfis (profile), onde cada um deles tem por objetivo atender a um tipo específico de mídia ou determinado cenário de uso.

Atualmente existem diversos perfis padronizados pela IETF para o RTP e, na presente dissertação, alguns serão abordados. O primeiro deles, que será foco deste capítulo, é o apresentado por (SCHULZRINNE *et al.*, 2003) para a criação e gerenciamento de uma conferência de áudio e vídeo com um mínimo controle. Também serão abordados em capítulos futuros alguns perfis que tratam do estabelecimento de canais seguros usando a pilha RTP/RTCP.

Uma das características principais do RTP/RTCP é que ele procura interferir o mínimo na rede, evitando aumentar a banda necessária para que o dado multimídia seja trafegado. Para isso ele aperfeiçoa ao máximo o envio de pacotes de dados em detrimento dos de controle.

Para prover a funcionalidade de controle da qualidade da conferência multimídia, o *framework* trabalha com a confecção de relatórios que são gerados esporadicamente e enviados sempre que houver banda disponível e ociosa, onde muitas vezes um único pacote RTCP contém diversos relatórios de controle acumulados.

Um pacote RTP é bastante simples de ser entendido. Ele consiste de um conjunto de cabeçalhos totalizando aproximadamente 40 bytes e um campo de carga (*payload*) onde irão trafegar os dados multimídia propriamente ditos.

Na figura 2.6 é apresentada a estrutura do pacote RTP, onde na primeira linha e coluna é apresentado o número de bytes que cada cabeçalho ocupa. Deve-se observar que a linha organiza o pacote em conjunto de 32 bits, enquanto a coluna diz qual é a posição do primeiro bit onde a informação será armazenada. Deve-se comentar a existência do símbolo (+), que significa a operação soma aritmética. Ele é usado para definir a posição inicial do bit utilizado na informação quando não se sabe exatamente esse valor, ou seja, onde o dado antecessor possui um valor variável.

+ Bits	0-1	2	3	4-7	8	9-15	16-31
0	Ver.	P	X	CC	M	PT	Sequence Number
32	Timestamp						
64	SSRC identifier						
96	... CSRC identifiers ...						
96+(CC×32)	Additional header (optional), indicates length "AHL"						
96+(CC×32) + (X×(AHL+16))	Data						

FIGURA 2.6 – Pacote RTP. Fonte: (WIKIPEDIA, 2007c)

Apesar de existirem vários cabeçalhos previstos na especificação do RTP, descreve-se em seguida aqueles que são fundamentais para o entendimento dos conceitos envolvidos na presente pesquisa.

O primeiro desses cabeçalhos é o *payload type* (PT), que tem por finalidade descrever o tipo de mídia contido no pacote bem como auxiliar a aplicação a decodificar adequadamente o pacote recebido, uma vez que ele é quem informará a aplicação que CODEC será utilizado para digitalizar o áudio. O tipo de áudio a ser usado na conferência, conforme apresentado em 2.3, é negociado anteriormente através de transações efetuadas pelo protocolo SIP e SDP.

O campo *sequence number* provê funcionalidades importantes de controle ao *framework*. De posse desse valor, o terminal pode perceber com clareza se os pacotes estão chegando na ordem seqüencial correta e se existe alguma descontinuidade no seu recebimento.

Porém, essa informação sozinha não é capaz de definir a situação real que se encontra a rede. Para obter essa informação, o RTP utiliza outro cabeçalho bastante importante, o *timestamp*, que tem por finalidade definir o horário no qual o pacote RTP foi gerado em

seu emissor.

Um dos problemas da informação existente no cabeçalho *timestamp* é que ela é local, ou seja, está relacionada ao tempo de geração no remetente, não sendo necessariamente uma informação válida no destinatário, uma vez que a sincronização dos relógios entre os pares da comunicação não é definida como obrigatória pela especificação do protocolo. Para que essa informação seja útil para fins de monitoramento da rede, faz-se necessário combiná-la com aquelas existentes em outros pacotes (conforme (PERKINS, 2003)) ou estabelecer algum mecanismo de sincronização durante a sinalização SIP, podendo ser usado para isso os campos de *timestamp* existentes no SDP.

Outro cabeçalho existente em um pacote RTP é o *Synchronization Source Identifier (SSRC)*. Seu objetivo é o de identificar de forma única um determinado terminal em uma conferência, possibilitando-lhe conversar com uma infinidade de outros terminais de forma simultânea.

Nos casos onde a mídia trafegada foi criada por vários terminais (áudio-conferência) é usado o cabeçalho *Contributed Source Identifiers (CSRC)* para identificar cada *SSRC* que contribuíram na formação da mídia (áudio ou vídeo) final recebida pelo usuário. Para ser possível a aplicação saber quantos *CSRC* existem no pacote, o atributo *CC* é usado para definir esse número.

Para possibilitar que o RTP seja uma plataforma extensível, em seu corpo existem dois campos que podem ser usados para realizar essa tarefa. O primeiro deles define que o pacote trafegado não é uma versão original do protocolo (SCHULZRINNE *et al.*, 2003), mas sim um perfil derivado dele. Para informar isso ao cliente, o cabeçalho X deverá ser igual a 01 (um), definindo que após a aplicação obter os dados contidos no *CSRC*, ela irá ler os dados específicos da extensão usada (AHL) e não a informação multimídia

(*payload*).

O RTP possui outros cabeçalhos que possuem caráter informativo e auxiliar. O primeiro deles é o *V* que indica a versão do RTP usada para construir a mensagem. Na seqüência existe o *P* (*padding*) que informa a aplicação se o pacote sofreu algum tipo de completamento para fazer com que o mesmo possuísse um tamanho múltiplo de 32. Por fim existe o o *M* (*marker*), que é usado para efeitos de otimização do canal e não será objeto de estudo desse documento.

Diferentemente do RTP, os pacotes RTCP não possuem uma arquitetura comum de formação, pois conforme já comentado, em um único pacote pode haver vários relatórios de tipos diferentes, fazendo com que cada pacote tenha um formato diferente. Dessa forma apenas os principais relatórios que podem ser construídos através do referido protocolo serão apresentados em seguida.

O primeiro deles é o *Relatório do Receptor (RR)*, que tem por finalidade apresentar ao par remoto da conferência, como o terminal local está percebendo a qualidade da rede. Entre as informações existentes nesse relatório encontra-se a fração de pacotes perdidos, último número de seqüência recebido do par remoto, variação média dos tempos de chegada dos pacotes.

Por sua vez, um *Relatório do Emissor (SR)* apresenta o número de pacotes e bytes enviados por *SSRC* que participa da conferência, possibilitando que cada par remoto possa calcular o estado da rede.

Além dos relatórios RR e SR existem outros que são informativos e não estão diretamente associados à qualidade da conexão.

O primeiro desses relatórios é o *SDES (Source Description Itens)*, que permite o

mapeamento entre os *SSRC's* existentes na conferência e seu endereço real.

Ainda existe o relatório *BYE* que informa aos diversos participantes da conferência quando algum elemento deseja sair.

Por fim, para prover a extensão do protocolo, existe um protocolo do tipo *APP* e este possibilita que projetistas de aplicações desenvolvam relatórios específicos para atender suas necessidades.

2.5 Integrando os Protocolos SIP, SDP e RTP/RTCP

Até o presente, os protocolos de telefonia foram apresentados de forma isolada, suas funções específicas e sua arquitetura individual foram descritas. Porém, para uma completa compreensão de como uma conferência multimídia ocorre se faz necessário o entendimento de como eles colaboram entre si. Se um deles não funcionar adequadamente, não é possível realizar a chamada.

A presente seção mostrará como a conferência apresentada na figura 2.4 ocorre na prática. Para isso, a cada conjunto de mensagens serão apresentadas as ações desencadeadas pelo recebimento e análise das mensagens recebidas. Para simplificar, não se fará a análise do interior dos *proxy's* intermediários, uma vez que nos cenários de interesse para a presente tese, esses elementos não devem interferir nas mensagens que trafegam através deles.

Todo o processo que usa o protocolo SIP deve ser iniciado através de uma requisição, que no caso de uma conferência multimídia, é do tipo INVITE. Conforme se vê na figura 2.7, o *user1*, iniciador do processo, constrói uma requisição INVITE e insere o destino final de sua requisição que é o *user2@domain2.com*. Nota-se que ele não precisa saber qual é o

servidor responsável pelo domínio *domain2.com*, já que essa tarefa é de responsabilidade do *proxy1.domain1.com*.

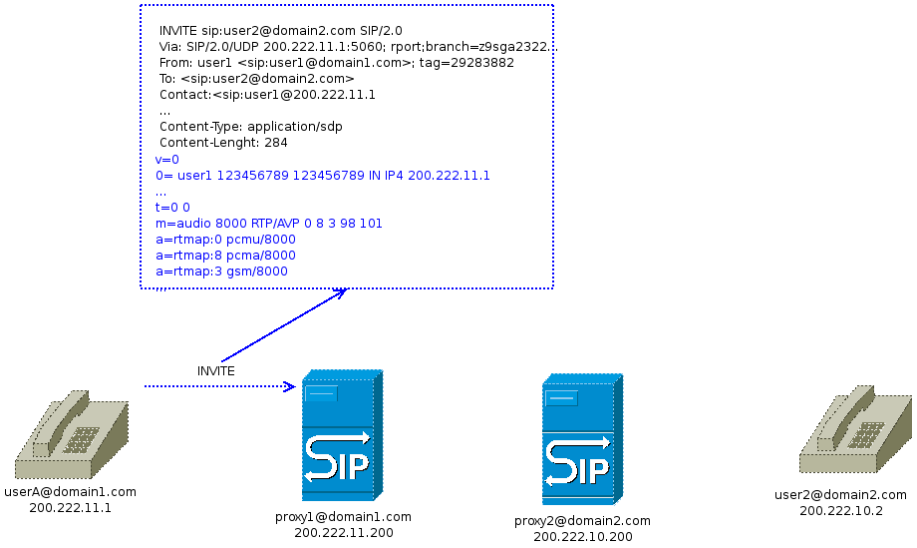

FIGURA 2.7 – Usuário 1 envia um convite para conferência.

Uma outra informação importante sobre a supra-citada requisição é que no campo *Contact* é inserido o endereço final real do usuário, informação que será usada pelo par remoto para responder ao ACK e mensagens SIP posteriores que possam requerer um contato fim-a-fim.

Deve-se notar que, diferentemente do campo *Contact*, os campos *To* e *From* possuem um alias dos usuários remotos e locais respectivamente,.

Uma das informações mais importantes existentes em uma requisição INVITE é seu payload (pacote SDP), pois é nele que o *user1* descreve como deseja construir a conferência. Nesse campo, apresentado na cor azul (ver 2.7), o usuário local informa ao outro par que deseja realizar a conferência usando a porta local 8000, o protocolo RTP/RTCP e está localizado no endereço 200.222.11.1.

É interessante mencionar que o user1 informa ao seu par remoto o CODEC a ser usado para codificar e decodificar o áudio, usando para isso o atributo *a=rtpmap*.

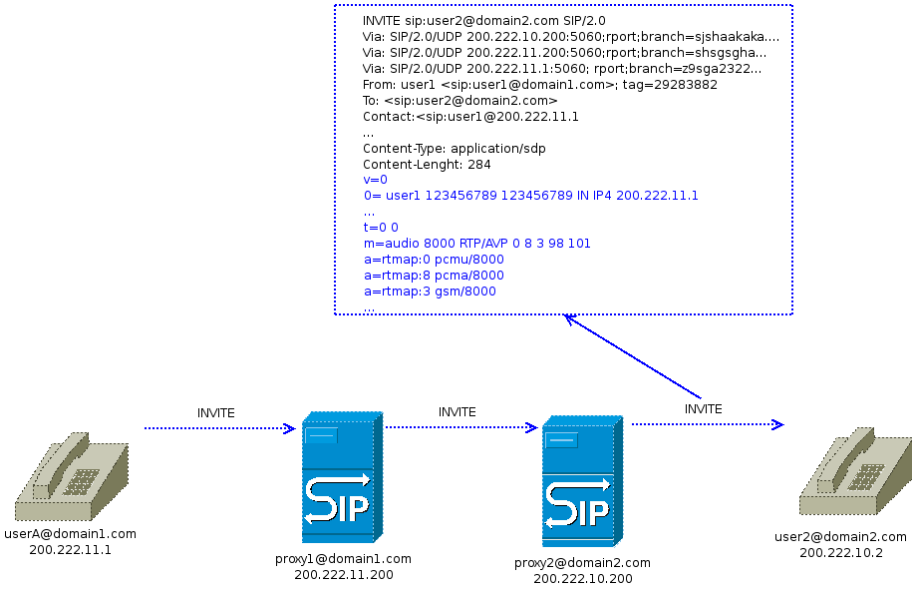

FIGURA 2.8 – Usuário 2 recebe o convite para conferência.

A figura 2.8 apresenta a requisição INVITE do *user1* sendo recebida pelo seu destino. Nessa figura deve-se notar que o campo Via, recebe o endereço de todos os elementos que de alguma forma tiveram acesso à mensagem, ou seja, que a rotearam. Isso é importante para que, ao responder a mensagem, ela percorra o mesmo caminho realizado anteriormente.

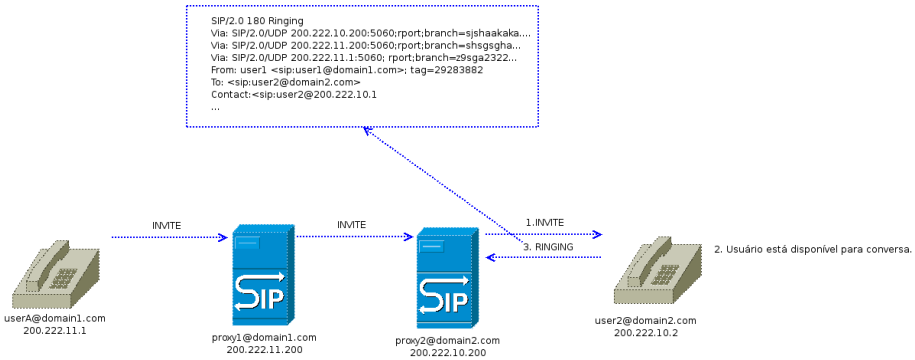

FIGURA 2.9 – Usuário 2 informa que está disponível para a conferência.

Após receber o *INVITE*, o terminal remoto (*user2*) deverá informar se está disponível ou não para a conferência, usando para isso ou uma resposta *180 Ringing* ou uma mensagem de erro. No caso da figura 2.9 é informado ao *user1* que o terminal remoto está disponível para a conferência. Deve-se notar nessa resposta que ela não possui payload (pacote SDP), apenas cabeçalhos SIP.

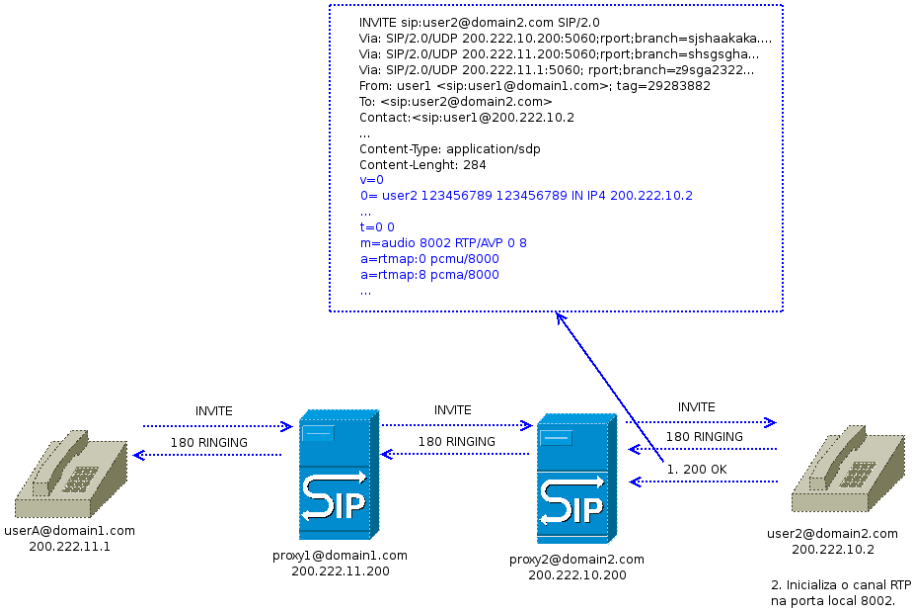

FIGURA 2.10 – Usuário 2 constrói uma resposta 200 OK.

Após analisar o INVITE recebido em detalhes, extraindo todas as informações relativas a como o user1 deseja construir a conferência, o terminal remoto constrói uma resposta *200 OK*, inserindo nela suas informações pessoais a respeito de endereços, bem como o(s) CODEC(s) que ela suporta ou deseja conversar e que, no caso da figura 2.10, é informado que ele suporta apenas os CODEC's PCMU e PCMA.

Após enviar a mensagem acima, o *user2* inicializa o seu canal multimídia usando como parâmetros os dados inseridos na resposta supra-citada, ou seja na porta 8002.

Após receber e analisar a resposta *200 OK* do *user2*, o terminal iniciador irá construir a sua porta multimídia local (8000) e se preparar para codificar o áudio em um dos formatos escolhidos pelo seu par remoto (PCMU ou PCMA). Após isso, ele construirá uma nova requisição (*ACK*), que informará ao seu par remoto que a resposta *200 OK* foi devidamente recebida e que a conferência pode ser iniciada. Somente após o *user2* receber essa nova requisição é que ele começará a enviar dados RTP para o *user1*.

Uma informação importante que não foi apresentada nas figuras acima, é que durante

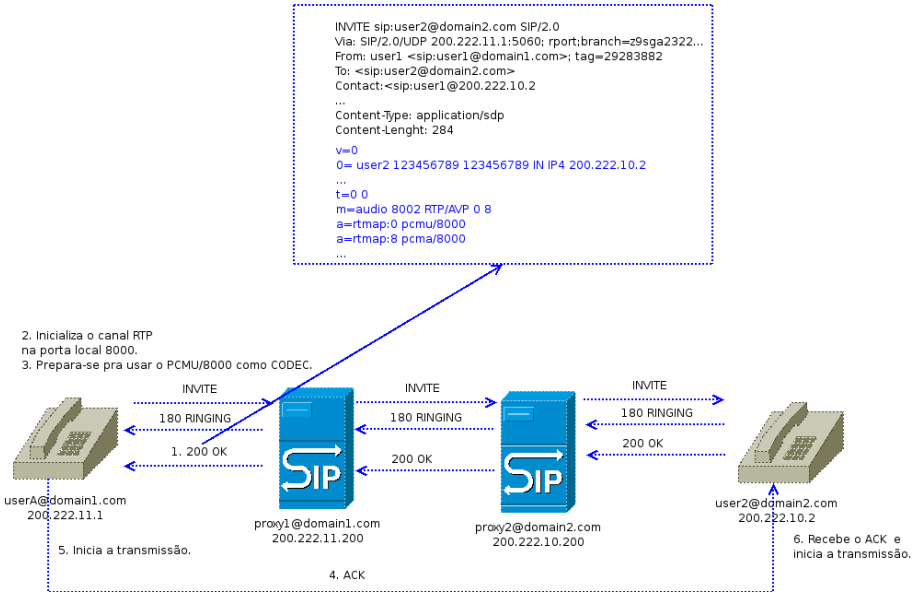

FIGURA 2.11 – Usuário 1 inicializa o seu canal.

o tempo em que o canal estiver aberto, alguns relatórios RTCP deverão ser trocados entre os pares, de forma a aperfeiçoar o canal para uma melhor qualidade dos dados multimídia trafegados entre eles.

De posse desse cenário, é simples verificar a integração de todos os protocolos (SIP, SDP e RTP/RTCP) para a realização de uma conferência, sendo possível inclusive entender porque os protocolos são interdependentes.

No próximo capítulo, será iniciada a discussão acerca de como prover conferências privadas fim-a-fim. Nessa ocasião serão apresentadas as definições básicas sobre segurança que permearão toda presente tese, bem como a problemática de se prover segurança sem interferir na qualidade da voz percebida pelo usuário. Ao final, serão delineados os requisitos básicos que devem ser implementados em todos os protocolos e arquiteturas destinados a prover segurança, tornando possível a análise das arquiteturas apresentadas em capítulos futuros do presente documento.

3 Requisitos para a Proteção da Mídia

Uma das fragilidades que impedem que os sistemas VoIP sejam adotados em larga escala em ambientes empresariais está relacionada ao conjunto de vulnerabilidades herdadas do seu protocolo base, o protocolo IP. Dentre elas, uma que desperta especial interesse é a que trata da simplicidade em realizar escutas clandestinas em uma rede VoIP, problema esse que é o foco de estudo da presente tese.

Em uma rede IP genérica, essa fragilidade pode ser resolvida através de mecanismos de segurança simples, onde o mais conhecido é a arquitetura de túneis criptográficos, que são chamados de Virtual Private Networks (VPN).

Porém em uma rede de voz, existem alguns complicadores adicionais, pois os dados precisam ser transportados com características de tempo real, fazendo com que esse parâmetro seja considerado no esquema a ser adotado.

Outra característica que diferencia as soluções genéricas das VoIP, é que as de telefonia IP em muitos cenários necessitam implementar uma infra-estrutura fim-a-fim, ou seja, uma arquitetura onde o segredo seja compartilhado apenas entre os terminais. Nesse caso, o uso de túneis não é adequado.

O presente capítulo tem por objetivo iniciar a discussão sobre esse assunto. Para isso, inicia-se com uma breve conceituação sobre segurança e criptografia. Posteriormente, serão apresentados os requisitos mínimos para prover uma voz com qualidade e, por fim, serão discutidos os requisitos específicos de segurança que devem ser impostos a um ambiente multimídia de tempo real, formando o arcabouço teórico necessário para analisar as arquiteturas propostas para tratar essa questão que serão estudadas no próximo capítulo (4).

3.1 Conceitos Básicos de Segurança

Segundo (CARUSO; STEFFEN, 1999), a informação é um dos principais ativos de uma empresa e como tal deve ser provida de algum tipo de segurança. Nas empresas atuais, a informação se sobrepõe aos ativos existentes nas linhas de montagem, pois é ela que garante a competitividade e muitas vezes a sobrevivência de uma empresa.

A (ISO/IEC, 2005) define como segurança da informação a *proteção da informação contra vários tipos de ameaça para garantir a continuidade do negócio, minimizar o risco, maximizar o retorno sobre os investimentos e as oportunidades de negócio.* Para ser possível esta proteção, é necessário que três atributos principais da informação sejam mantidos: a *integridade*, a *confidencialidade* e a *disponibilidade*.

A integridade é a *propriedade de salvaguardar a exatidão e a completeza de uma determinada informação* (ISO/IEC, 2004). É essa propriedade que garante ao usuário que um determinado conteúdo que ele esteja tendo acesso é realmente o original, sem qualquer tipo de adulteração.

A confidencialidade é a capacidade de salvaguardar a informação de acesso indevido,

ou seja, garante que somente os usuários autorizados terão acesso à informação.

Finalizando, a disponibilidade garante que uma informação esteja acessível e utilizável sob demanda para as entidades autorizadas (ISO/IEC, 2004).

Ao analisar os conceitos acima apresentados, percebe-se que todas as propriedades só possuem sentido se forem garantidas em seu conjunto. Como exemplo, pode-se imaginar uma informação para a qual tenham sido criados mecanismos para garantir a sua confidencialidade e a sua integridade. Entretanto, se o seu conteúdo não estiver disponível, ela não poderá ser utilizada, logo não terá nenhuma valia.

Existem outros atributos associados ao conceito de informação segura que, apesar de não serem fundamentais em alguns cenários, tornam-se complementares e obrigatórios. Um desses atributos é a *autenticidade*, cujo objetivo é o de garantir que o remetente e o destinatário de uma determinada mensagem sejam corretamente identificados, evitando que um terceiro elemento (atacante) assuma o papel de um deles e realize operações ilegais. A autenticidade associada à integridade é muito importante para alguns cenários bastante usuais atualmente, como, por exemplo, no comércio e no correio eletrônico.

Para prover um sistema seguro é necessário que, em igual intensidade, sejam garantidos todos os atributos vistos na presente seção. Para garanti-los, a (ISO/IEC, 2005) sugere o uso de mecanismos criptográficos, que é o objeto de estudo da próxima seção.

3.2 Conceitos Gerais de Criptografia

Conforme definido em (TRAPPE; WASHINGTON, 2001), a criptografia é a área do conhecimento que permite o desenvolvimento de sistemas capazes de realizar uma comunicação segura através de um canal inseguro. Seu uso remonta da Antiguidade,

quando sistemas baseados em substituição eram usados para proteger a comunicação de César com seus generais.

Apesar de sua existência na antiguidade, a criptografia como é conhecida atualmente somente foi possível com o aparecimento do computador, que permitiu o uso de algoritmos matemáticos complexos.

O conceito ora apresentado pode ser mais bem compreendido através da observação da figura 3.1. Nela existe um canal inseguro entre os dois usuários, Beto (*Bob*) e Alice, que desejam realizar uma comunicação segura. Nesse mesmo canal existe um outro elemento, Eva (*Eve*), que procura interceptar a informação transferida entre os usuários e realizar algum tipo de ação que corrompa algum dos atributos da informação apresentados na seção anterior.

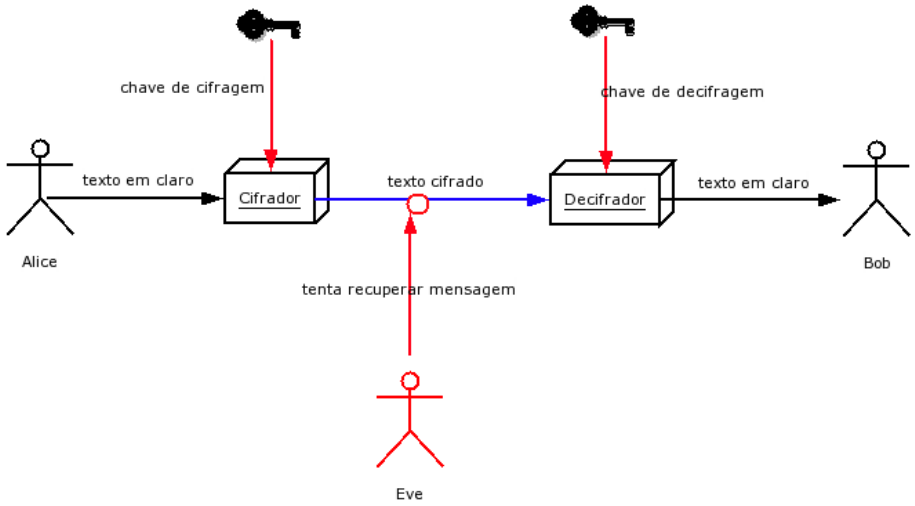

FIGURA 3.1 – Cenário Básico de Uso de Criptografia.

Para proteger o conteúdo da comunicação contra as ações ilegítimas de Eva (*Eve*), insere-se no canal transmissor um sistema cifrador e um sistema decifrador no receptor. O cifrador tem por finalidade o embaralhamento da informação a ser transmitida, de forma que Eva não consiga fazer uso da mesma. Já o decifrador, tem a finalidade de recompor a mensagem original de forma que o receptor possa ter acesso ao conteúdo da informação

transmitida.

O objetivo primário do sistema apresentado na figura 3.1 é o de proteger a informação contra o acesso indevido, mantendo-a confidencial. Com mecanismos adicionais, que serão vistos oportunamente, é possível garantir todos os atributos explanados na seção 3.1 do presente capítulo.

Observando ainda a figura 3.1, percebe-se o uso de um conjunto de chaves para realizar as atividades de cifrar e decifrar a informação a ser transmitida. Quando é usada uma chave específica para cifrar e outra decifrar, o esquema é chamado de *assimétrico*. Já no caso onde uma única chave é utilizada para realizar ambos os processos, o esquema é chamado de *simétrico*.

Eva, para realizar suas ações ilícitas precisa decifrar o conteúdo criptografado que percorre o canal de dados, precisa ter acesso à chave utilizada . Apesar da existência de várias formas de se obter essa informação, a mais simples, porém mais trabalhosa, é a que visa obter a chave correta através do teste de todas as combinações possíveis, o que é chamado de *ataque por força bruta*.

Esse tipo de ataque é sempre possível de ser realizado, porém nem sempre o tempo computacional necessário para sua efetivação é prático, ou seja, o tempo de validade da informação sigilosa (utilidade) é menor do que o tempo necessário para se descobrir a chave usada no processo.

Atualmente, quanto maior o tamanho da chave melhor, uma vez que diariamente surgem computadores com mais capacidade de processamento. O tamanho da chave não é suficiente para garantir a segurança do canal, pois a robustez do algoritmo também deve ser levada em consideração nessa análise (TRAPPE; WASHINGTON, 2001).

Conforme comentado anteriormente, quando é usada uma chave para cifrar e outra para decifrar a informação que percorre um canal, o esquema é chamado de assimétrico. Esse esquema, também conhecido como *criptografia de chave pública*, consiste no fato de um determinado usuário, que deseja receber uma informação através de um canal criptográfico, gerar um conjunto de chaves. Uma delas, aquela usada para cifrar, é chamada de pública, pois pode ser distribuída abertamente e entregue sem qualquer tipo de proteção a qualquer pessoa ou entidade que queira lhe enviar uma mensagem sigilosa. A outra chave, usada para decifrar, é chamada de privada e deve ser mantida em completa segurança, pois, se for capturada por um atacante, ele será capaz de decifrar todas as mensagens endereçadas ao usuário.

Apesar de permitir a distribuição aberta das chaves públicas, esse esquema é muito menos eficiente que o de chave simétrica, fazendo com que seja seu uso considerado inadequado em muitos cenários práticos, inclusive os de tempo real (BUCHMANN, 2002).

Para solucionar essa limitação foi desenvolvido um esquema combinado, onde é utilizado um esquema assimétrico para negociar uma chave simétrica chamada de *chave de sessão*, conforme pode ser visto na figura 3.2.

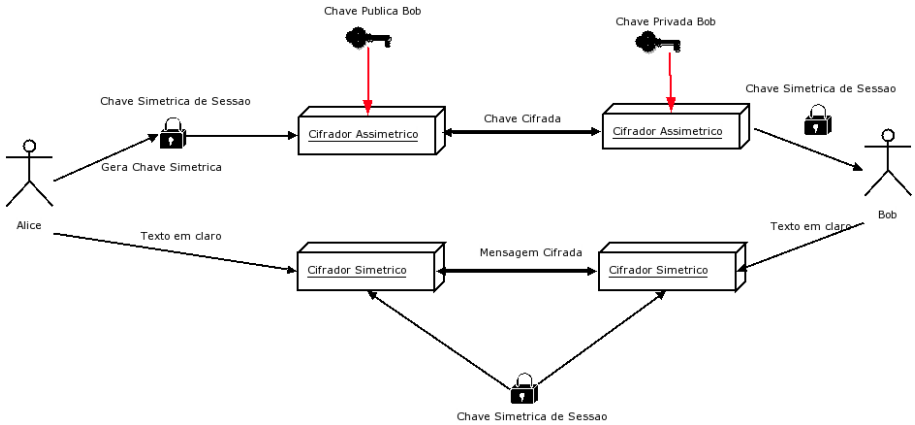

FIGURA 3.2 – Esquema de Criptografia de Sessão.

No esquema apresentado na figura acima, o transmissor (Alice) utiliza a chave pública

(chave usada para cifrar) do destinatário (Beto - *Bob*) para cifrar a chave simétrica da sessão, gerada por ele, e que permeará a proteção do canal seguro a ser criado. Ao receber a chave de sessão cifrada, o destinatário usa sua chave privada para decifrar a mensagem e ter acesso à chave contida nela. De posse dessa chave simétrica, comum a ambos os usuários, é possível estabelecer uma comunicação bilateral segura e mais eficiente.

Esse esquema resolve o problema da ineficiência dos algoritmos assimétricos e da inflexibilidade dos algoritmos simétricos, restando outro que ainda permanece em aberto, que é a necessidade do usuário distribuir sua chave pública para cada elemento que ele deseja estabelecer uma comunicação segura. Em cenários onde existem apenas dois usuários esse fato não chega a ser um problema, porém em cenários mais amplos (Internet), isso se torna inviável, uma vez que para um transmissor distribuir sua chave para n usuários que desejam estabelecer um canal seguro, ele deverá enviá-la para *n-1* usuários. Por analogia pode-se concluir que, para comutar as chaves de todos n usuários entre eles seriam necessárias *n(n-1)/2* comutações.

Para solucionar essa questão, foi desenvolvido a idéia de *repositórios públicos*, local onde todas as chaves públicas devem ser armazenadas. Dessa forma quando um usuário desejar estabelecer um canal seguro com outro, ele obtém a chave pública do destinatário nesse repositório, ao invés de ser necessária a existência prévia de uma cópia dela localmente.

O esquema de repositórios públicos associados ao de chave de sessão garante que uma informação seja trafegada de forma segura mesmo quando usado um canal público (inseguro), garantindo a confidencialidade de uma informação.

Apesar disso, não existe nenhuma garantia de que a chave que está sendo usada para cifrar a mensagem seja realmente do destinatário desejado, pois nem sempre é comprovar

a sua identidade. Também no esquema até então apresentado, não se garante a identidade do emissor, o que impede a verificação da autenticidade da informação.

A identificação dos usuários em um sistema seguro pode ser realizada de duas maneiras: através de mecanismos simétricos ou assimétricos. Na primeira delas, a autenticidade das chaves públicas é garantida mediante a instituição de uma *Autoridade Certificadora (AC)*, entidade cujo objetivo é ser um órgão central no qual todos os elementos existentes no processo de comunicação confiam. Essa AC, utilizando sua chave privada, cifra as chaves públicas dos usuários de sua rede de confiança, gerando um novo documento conhecido como *Certificado Digital*.

Um *certificado digital* é normalmente composto por um e-mail ou alias (apelido) que identifica o usuário, de um alias que identifica a AC responsável por autenticá-lo e a chave pública do usuário cifrada. Para um usuário checar a autenticidade da chave pública existente no certificado, basta que ele o decodifique usando a chave pública da AC que a assinou. Caso obtenha sucesso, o certificado realmente existe e foi assinado pela AC.

Porém, conforme comentado anteriormente, o uso de criptografia assimétrica perde em desempenho quando comparado ao esquema simétrico. Associado a isso, pode ser que exista algum segredo compartilhado entre os pares, fazendo com que não seja necessário o uso de algoritmos assimétricos para garantir a autenticidade da mensagem. Nesses casos é usado um esquema de autenticação baseado em funções de *hash* assinadas, onde a chave simétrica compartilhada é usada para autenticar a mensagem.

Uma função de *hash* é aquela capaz de receber uma entrada de tamanho variável e gerar como saída um texto com tamanho fixo (*fingerprint*), que será um representante resumido da entrada, onde qualquer alteração realizada no texto de entrada irá gerar, com alta probabilidade, um valor diferente na saída, permitindo detectar qualquer adulteração

na mensagem de entrada, garantindo dessa forma a sua integridade.

A figura 3.3 oferece um exemplo de saídas geradas por uma função de hash.

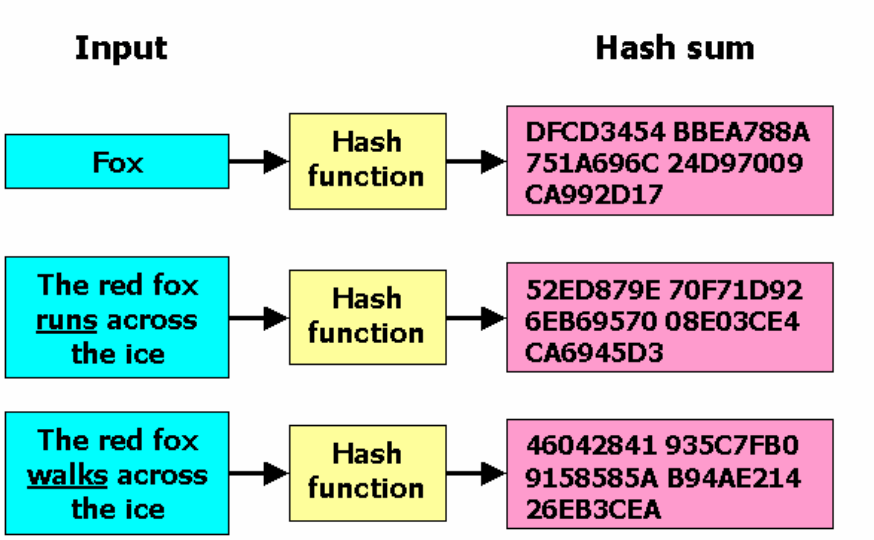

FIGURA 3.3 – Função de Hash. Fonte: (WIKIPEDIA, 2007b)

Para autenticar as mensagens, são usadas as chamadas *Message Authentication Code (MAC)*, que são funções de hash que utilizam uma chave para compor o *fingerprint* da mensagem (STINSON, 2002).

De forma resumida, para prover apenas a integridade de uma mensagem basta o uso de uma função de *hash* comum (*unkeyed function*), porém quando se deseja autenticação associado à integridade, deve-se usar uma função *MAC*.

Para que uma função de hash possa ser considerada segura para fins criptográficos, é necessário que ela seja resistente à *primeira pré-imagem*, à *segunda pré-imagem* e à *colisão*.

Sendo h uma função de hash, x a mensagem que eu desejo autenticar e $y=h(x)$ o fingerprint gerado, considera-se que a função h é resistente quanto a sua primeira pré-imagem quando, obtido o seu fingerprint y, é computacionalmente muito difícil gerar a

mensagem x que a originou.

A função h é resistente à sua segunda pré-imagem quando, tendo o fingerprint *y=f(x)* da mensagem x, é muito difícil obter outra mensagem z, distinta de da primeira e tal que *y=f(z)*.

Por fim, uma função de hash é resistente à colisão, quando é computacionalmente muito difícil obter duas mensagens *m1* e *m2* diferentes, que gerem o mesmo fingerprint *h(m1)=h(m2)*.

As funções MAC são implementadas através de algoritmos simétricos, usando a chave anteriormente acordada (ou uma variação da mesma) com o objetivo de autenticar e garantir a integridade da mensagem.

Apesar de ser computacionalmente mais eficiente, existem casos onde não existe um segredo compartilhado entre os pares e, nesses casos só resta o uso de assinaturas digitais. Porém, graças à sua ineficiência, não se recomenda o uso desses algoritmos em segmentos de dados muito grandes. Isso faz com que sejam usadas funções de hash sobre a mensagem e a assinatura assimétrica é aplicada somente sobre o fragmento comprimido e não sobre a mensagem inteira, artifício que agiliza o processo.

3.3 Requisitos para Ambientes VoIP Seguros e de Qualidade

Conforme apresentado na introdução (1.1), o objetivo da presente tese é o de analisar arquiteturas VoIP que implementem uma segurança fim-a-fim para a mídia trafegada na conferência, ou seja, garantir que o conteúdo conversado entre dois pares seja protegido

contra escutas clandestinas.

Para que se consiga atingir esse objetivo, uma série de problemas precisa ser resolvida, podendo-se apontar as seguintes questões principais:

- Como criar canais com qualidade?
- Como negociar os parâmetros necessários para a criação dos canais de mídia seguros?
- Como criar canais de mídia seguros?

A primeira questão é simples de ser entendida, pois de nada adianta criar um canal multimídia seguro, se não for possível garantir um mínimo de qualidade na voz trafegada através dele.

Já o segundo tópico foi o objetivo do estudo desenvolvido na presente pesquisa, sendo assim será melhor abordado durante o desenvolvimento deste documento.

Por fim, para ser possível a criação de um canal protegido, um problema anterior precisa ser bem delineado, que é a questão de como negociar os parâmetros criptográficos necessários para a criação do canal: se eles não forem acordados de forma segura, o canal também não terá essa característica.

A presente seção abordará cada um desses tópicos, analisando as principais literaturas existentes e apontando os requisitos necessários para responder a essa questão.

3.3.1 Requisitos Mínimos para Prover uma Chamada IP de Qualidade

Antes de iniciar um estudo sobre requisitos de proteção da sinalização e da mídia em um ambiente VoIP, é necessário analisar quais são os critérios mínimos (independentes de

tecnologia) para prover um canal multimídia com qualidade.

Conforme estudo apresentado em (ITU-T, 1996), para que uma chamada multimídia seja considerada de boa qualidade, é necessário garantir um nível mínimo de controle sobre três parâmetros: a *latência*, o *jitter* e a *taxa de perdas de pacote*.

A *latência* consiste no atraso que um pacote pode experimentar ao percorrer o trecho do remetente ao destinatário. Em (PERKINS, 2003) é apresentado como a latência é formada. Segundo ele, a latência pode ser imposta por uma série de fatores em uma rede de pacotes, desde o tempo necessário para que a mídia analógica seja capturada e formatada digitalmente, até o tempo imposto pelos mecanismos de controle de fila em dispositivos de interconexão e finalmente o tempo de decodificação do sinal digital em seu formato analógico original.

Segundo (ITU-T, 1996) para que um conversação não seja clara é necessário que a latência da rede tenha uma valor superior à 400ms (ver figura 3.4). Essa situação é bastante difícil de ocorrer, mesmo quando o meio não possui nenhum controle de admissão (Internet). Uma situação especial ocorre em sistemas compostos por mais de um segmento de satélite, pois a soma dos atrasos individuais inseridos pelos tempos de transmissão entre um satélite e outro, muitas vezes ultrapassa esse valor.

O segundo parâmetro que influencia na qualidade da voz é a *taxa de perdas de pacotes* da rede. A perda de pacotes pode ser ocasionada por uma série de fatores da rede, dentre os quais destacam-se o congestionamento nos dispositivos de interconexão e os problemas na construção dos pacotes nas fontes.

Nas aplicações tradicionais para rede IP, normalmente os pacotes perdidos são tratados através do seu reenvio ou até mesmo na sua reconstrução no destinatário com base nas

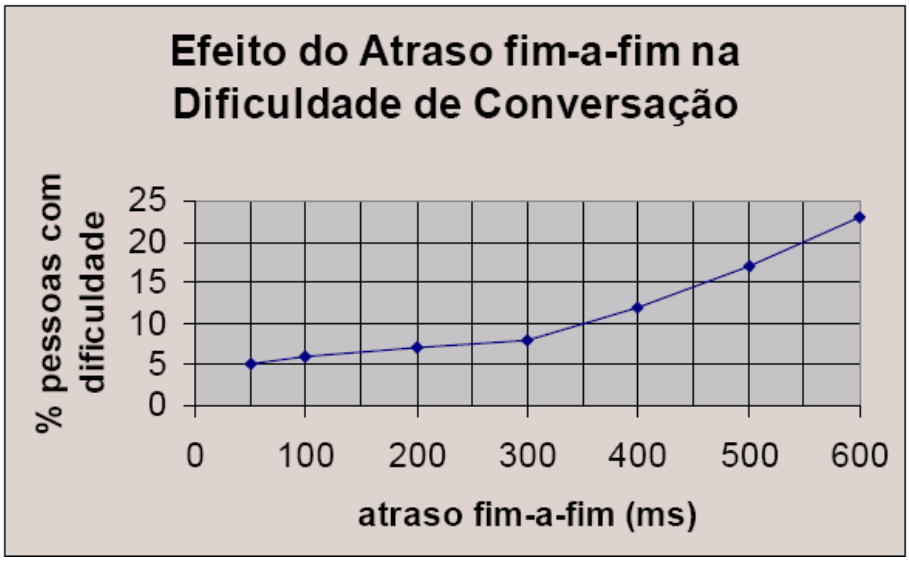

FIGURA 3.4 – Influência da Latência na Legibilidade da Voz. Fonte: (LEWIS, 2000)

informações contidas em pacotes anteriores (TANENBAUM, 2003).

Porém, nas aplicações de que envolve a transmissão de voz, o acréscimo de informação ocasionado por essas retransmissões, associado à característica do cérebro humano, que consegue reconstruir fragmentos de voz mesmo quando eles sofrem pequenas interrupções na sua recepção, faz com que as aplicações multimídia não se preocupem com esse problema.

Apesar dessa característica, a faixa em que essa anomalia pode ocorrer sem que haja interferência na interpretação da informação transmitida é bastante pequena. Segundo (ITU-T, 1996), taxas de perdas de pacotes superiores a 5% dificultam a legibilidade da comunicação e taxas superiores a 10% praticamente torna a mensagem incompreensível. Esse fato é apresentado na figura 3.5.

Dentre os diversos fatores que influem na qualidade da voz em uma rede de pacotes, o *jitter* é o elemento que mais afeta a inteligibilidade de uma da aplicação multimídia. Segundo (PERKINS, 2003), o jitter pode ser entendido como a variação no tempo e na seqüência de entrega dos pacotes devido à mudanças nas condições da rede. Esse conceito

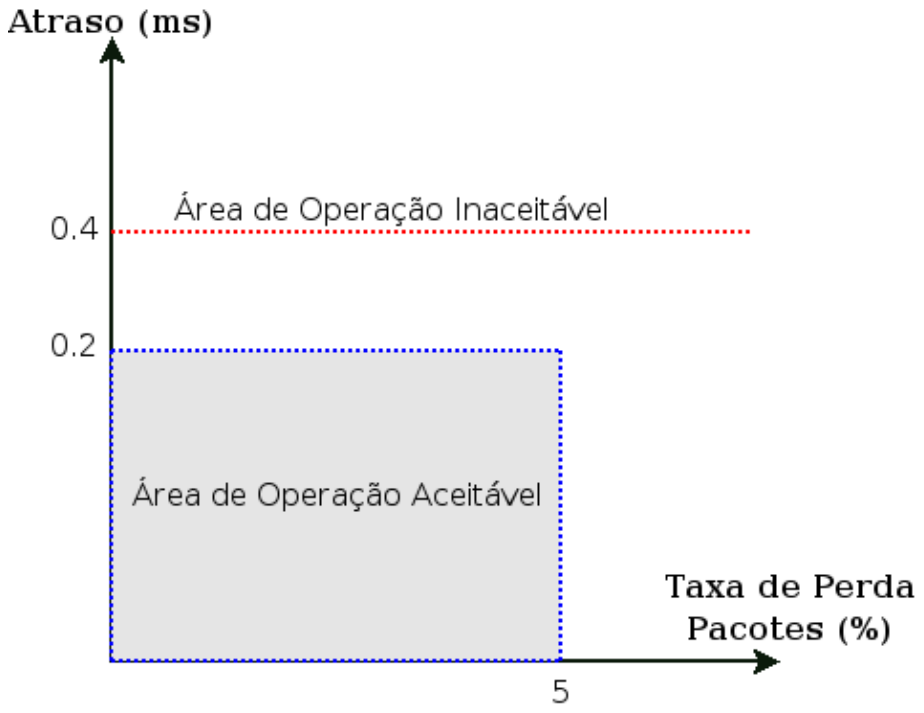

FIGURA 3.5 – Influência da Taxa de Erro na Legibilidade da Voz. Fonte: (LEWIS, 2000)

pode ser entendido através da análise da figura 3.6, que mostra a presença de um emissor que envia diversos pacotes e, devido a alguma anomalia na rede, eles são entregues a intervalos de tempo variáveis e numa seqüência incorreta.

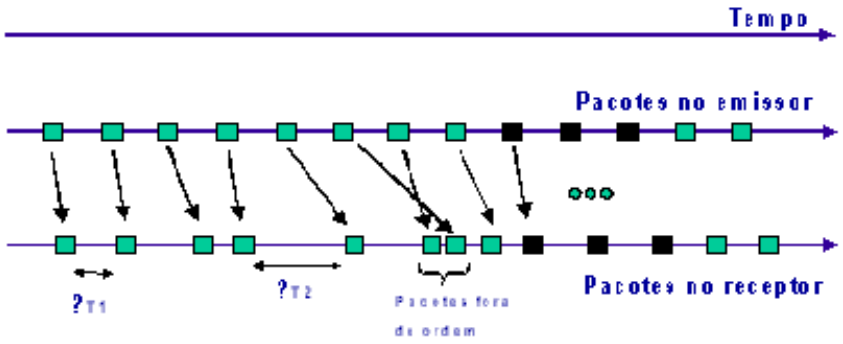

FIGURA 3.6 – Conceito de Jitter. Fonte: (LEWIS, 2000)

Para combater esse fenômeno, os dispositivos de comunicação introduzem em seus algoritmos de codificação/decodificação (CODEC) algum sistema de compensação. A idéia básica desses sistemas consiste na inserção de um buffer que armazena amostras de som e as libera para reprodução a uma taxa constante, mesmo que a taxa de recebimento

desses pacotes de voz seja variável.

O tamanho do *buffer de jitter* é bastante diferente de uma implementação para outra, podendo ser encontrado algumas com buffer estático e outras com buffer dinâmico, onde esse último é adaptativo ao comportamento da rede, o que gera um maior consumo de processamento nos terminais onde está instalado.

De posse do que foi apresentado até o momento nessa seção, pode-se concluir que um projetista, ao decidir usar alguma aplicação multimídia na rede, deve projetar com cuidado o seu ambiente, calculando com riqueza de detalhes a sua topologia, a política de qualidade de serviço e a vazão necessária, de forma a obter os parâmetros mínimos apresentados anteriormente.

No preparo da rede para inserir um novo serviço esse cuidado deve ser redobrado. No caso da inserção de controles de segurança, automaticamente se gera um overhead na rede existente pois, além do processamento normal de codificação, transporte e decodificação da voz, sobrepõem-se aqueles causados pelos algoritmos e mecanismos criptográficos inseridos para garantir as propriedades apresentadas anteriormente no capítulo 3.1.

Na próxima sub-seção serão apresentados os requisitos específicos de segurança que devem ser empreendidos para se negociar os parâmetros necessários para a criação de um canal multimídia seguro. Porém, conforme já comentado anteriormente, deve-se atentar para o fato de que os requisitos ora apresentados devem sempre ser alcançados pois, em caso contrário, a qualidade da voz final percebida pelo usuário ficará deturpada, impedindo a compreensão da mensagem a ser trafegada.

3.3.2 Requisitos Mínimos para Prover a Negociação de Parâmetros Criptográficos

Conforme apresentado em 2.1, para que dois usuários quaisquer possam realizar uma conferência multimídia é necessário que duas tarefas sejam realizadas: a negociação de parâmetros mínimos para a criação de um canal entre os usuários (sinalização) e o transporte dos dados multimídias que constituem a conferência.

Apesar de serem tarefas individuais, percebe-se que não faz sentido falar em proteção da mídia, sem pensar na sinalização, pois é nessa fase que, possivelmente, ocorrerá a definição dos parâmetros de segurança (chaves, tamanho de chaves, algoritmos, etc) que estarão estabelecidos durante a conferência, tarefa conhecida como *Key Agreement*.

Em (BILLIEN, 2003) é apresentado uma série de requisitos necessários para prover a fase de *key agreement* de forma segura, onde alguns são genéricos a qualquer tipo de canal de dados a ser criado e outros específicos para ambientes VoIP.

Entre os requisitos genéricos existem alguns obrigatórios e outros opcionais. Entre os que são obrigatórios encontra-se a *confidencialidade* e a *proteção contra ataques de downgrading*. Conforme já apresentado na seção 3.1, a confidencialidade é a propriedade mediante a qual se garante que um segredo somente estará disponível entre as partes interessadas e que possuem permissão para tal. No contexto da primeira fase da sinalização VoIP, é ela que garante que alguns atributos necessários para o estabelecimento de um canal seguro (chaves, algoritmos, etc) sejam negociados em sigilo.

Já a imunidade a *downgrading atack* é importante, pois evita que um atacante consiga, através da substituição do algoritmo usado por um menos robusto, ter acesso às informações trocadas de forma confidencial entre os membros da conferência.

Segundo o mesmo autor, os requisitos não obrigatórios podem ser de seis tipos: *autenticação fim-a-fim*, *proteção contra ataques de replay*, *perfect forward secrecy*, *garantia de irretratabilidade*, *simplicidade de reenvio da chave* e *proteção contra ataques de negação de serviço*.

A *autenticação fim-a-fim* consiste na capacidade de os pares de uma conferência serem identificados mutuamente de forma única e correta. Essa propriedade, associada à *garantia de irretratabilidade*, possibilita que os terminais envolvidos realizem uma conversa com a absoluta certeza da identidade de seus pares, bem como a incapacidade de qualquer um dos terminais negar a sua ocorrência.

A proteção contra *ataques do tipo replay* consiste na capacidade do sistema identificar se um pacote recebido é original ou foi reinserido na rede. Esse tipo de ataque é bem simples de ser realizado. Para que ele ocorra, basta que o atacante, munido de um analisador de rede (*sniffer*), capture os pacotes de uma conversação legítima e, posteriormente, os reinsira na rede como se fossem válidos.

Outro requisito igualmente importante, porém não obrigatório, é o *perfect forward secrecy*. Esse requisito consiste na capacidade de um sistema, mesmo quando comprometido, prover o mínimo de informação a respeito das mensagens protegidas por ele. Um exemplo de um sistema que **NÃO** tem essa propriedade é apresentado em (BILLIEN, 2003). No referido exemplo, é apresentado um canal que, para negociar uma chave de sessão, a cifra com uma chave pública. Com a revelação indevida da chave privada a um atacante, toda informação trafegada usando a chave de sessão anteriormente acordada também está comprometida.

Uma vez que os protocolos criptográficos são fracos quando necessitam proteger com uma única chave muitos pacotes, os sistemas computacionais seguros devem prover alguma

forma de negociar uma nova chave durante o funcionamento de um canal pré-estabelecido.

Por fim, o último requisito genérico e opcional apresentado por (BILLIEN, 2003) é a proteção contra *ataques de negação de serviço*. Esse tipo de ataque consiste na exploração de uma falha no projeto do protocolo ou dispositivo usado, fazendo com que o sistema alvo fique sobrecarregado a ponto de não conseguir mais responder as requisições dirigidas a ele. Apesar desse tipo de ataque ser bastante comum e muitas vezes simples de ser realizado, não é foco da presente tese analisá-lo.

Apesar dos requisitos acima apresentados serem importantes, eles não são específicos a ambientes VoIP, ou seja, são necessários para qualquer ambiente computacional que necessite ser protegido. Para sistemas de telefonia IP existem dois requisitos específicos que devem ser considerados ao se desenvolver sistemas seguros de telefonia: o *uso de poucos recursos computacionais* e a necessidade de *não inserir mais delay (atraso)* no processo já existente.

A observância da primeira propriedade é bastante importante, pois sistemas VoIP normalmente são embarcados em dispositivos com escassos recursos de memória e processador (palm's, pockets pc's, smartphones e celulares), o que faz com que sistemas criptográficos que usem muitos recursos computacionais (processamento e memória) não sejam apropriados para funcionar nesses tipos de dispositivos.

Já o segundo requisito, apesar de menos crítico, deve ser observado com critério, pois os ambientes VoIP inserem uma latência adicional à sinalização telefônica. Isso ocorre pelo fato de ser uma estrutura baseada em pacotes, onde a transferência da informação precisa ser roteada por diversos elementos intermediários e muitas vezes sem experimentar qualquer priorização. Esse fato faz com que a inserção de um algoritmo de *key agreement* no meio desse processo tenha que ser feita com bastante critério, pois em (OHTA, 2006) é

apresentado um estudo demonstrando que a tolerância de um usuário a um atraso nessa fase é limitado a um tempo máximo, acima do qual ele desiste da transação.

É importante ressaltar que os requisitos apresentados por (BILLIEN, 2003) afetam apenas a sinalização VoIP, ou seja, existem para possibilitar a negociação de forma sigilosa dos parâmetros necessários para a criação da conferência multimídia. Em nosso estudo essa fase é implementada através dos protocolos SIP (2.2) e SDP (2.3).

Dessa forma, os requisitos ora apresentados, não definem como um canal seguro deve ser implementado, apenas viabilizam sua realização. Um estudo detalhado dos requisitos necessários para o estabelecimento de uma conferência multimídia com proteção contra escutas clandestinas será realizado na próxima subseção.

3.3.3 Requisitos Mínimos para Prover um Canal Multimídia Seguro

Em (BAUGHER *et al.*, 2004) é apresentado a especificação de um protocolo seguro para realizar o transporte multimídia que será foco de estudo no próximo capítulo da presente tese. Porém, antes de apresentar a estrutura e funcionamento desse protocolo, (BAUGHER *et al.*, 2004) apresenta uma série de requisitos que permearam a definição do mesmo, fato este importante para compreender como deve ser projetado uma arquitetura para esse fim.

Segundo este autor, um protocolo de transporte multimídia fim-a-fim tem por objetivos principais a garantia da *confidencialidade* e da *integridade* dos pacotes a serem trafegados através do canal. Nota-se que a disponibilidade da informação não consta entre os objetivos principais apresentados, pois a obtenção dessa propriedade necessita de

proteções e cuidados no projeto e implementação de todos os dispositivos que colaboram durante o processo de comunicação (enlaces de dados, dispositivos de interconexão, codec's, etc). Como a maior parte desses dispositivos estão fora do contexto dos dispositivos finais, essa propriedade não foi considerada.

Ainda, nesse mesmo trabalho, se estabelece que um protocolo seguro multimídia fim-a-fim deve complementarmente prover um *framework capaz de ser atualizado e estendido*, possibilitando a incorporação de novos protocolos criptográficos. Além disso, deve *interferir mínimamente no canal pré-existente*, fazendo com que as operações criptográficas realizadas não aumentem muito a banda consumida pela informação a ser trafegada.

Apesar de ambos serem requisitos importantes, o segundo, sob o ponto de vista do usuário, apresenta conseqüências mais sensíveis, pois é ele quem interfere na percepção final da qualidade da voz ou vídeo recebida nos dispositivos finais.

Devido a esse fato, (BAUGHER *et al.*, 2004) apresenta uma série de atributos embutidos em seu protocolo que visam obter um baixo custo de banda. O primeiro desses atributos é que o protocolo deve oferecer um *baixo custo de processamento*, fazendo com que ele possa funcionar em dispositivos com processadores de baixa capacidade (pda's e celulares).

O segundo atributo faz com que as informações relacionadas aos protocolos criptográficos inseridos (chaves, listas de proteção contra replay's, etc) sejam pequenos, ou seja, possuam um pequeno *footprint*. Esse atributo, apesar de ser independente, tem também como motivação possibilitar que dispositivos de baixa capacidade de memória possam rodar o protocolo.

Uma característica que coloca o SRTP em grande vantagem em relação às tecnologias

concorrentes (IPSEC, TLS), conforme será visto no próximo capítulo, é a sua capacidade de praticamente não aumentar o tamanho do pacote protegido em relação ao pacote original (4.1.1). Esse atributo é importante e desejável pois o aumento do tamanho do pacote final tem por conseqüência o aumento da banda necessária para transportar a informação, o que muitas das vezes não é possível (redes sem fio e de longa distância), ocasionando um aumento da *taxa de erros*, *latência* e *jitter*.

Por fim, um outro atributo importante citado por (BAUGHER *et al.*, 2004) e que também deve ser obtido em qualquer protocolo com a sua finalidade, é a sua independência dos protocolos de transporte e de redes específicas para seu funcionamento. Essa característica pode ser obtida dotando o próprio protocolo de capacidade de tratar falhas e perda de pacotes.

Com base no que foi apresentado até o momento, é possível estruturar um conjunto de requisitos necessários para prover um canal multimídia seguro, desde a sua sinalização inicial, momento de negociação de parâmetros do canal, até a hora do transporte dos dados entre os usuários (voz e vídeo).

Esse conjunto de requisitos servirão de base para a análise a ser realizada no próximo capítulo (4), onde serão apresentadas os padrões e estudos existentes na IETF, bem como serão apontadas suas vulnerabilidades e potencialidades.

4 Arquiteturas de Segurança VoIP Fim-a-Fim

Durante o desenvolvimento da presente tese foram apresentados os conceitos básicos acerca de telefonia IP e quais seriam os requisitos necessários para viabilizar a construção de um canal seguro baseada em uma arquitetura fim-a-fim.

A importância desse tipo de arquitetura reside na existência de alguns cenários onde é impossível estabelecer um relacionamento de confiança entre todos os elementos intermediários por onde trafega a informação. Pode-se citar como exemplo, um usuário que possui uma relação pessoal de confiança com outro usuário, porém não compartilha esta confiança com as empresas que lhes fornece o serviço de telefonia. Nesse caso, somente uma arquitetura de segurança fim-a-fim garante a confidencialidade nesse canal.

Neste capítulo algumas arquiteturas (padronizadas e em estudo) para prover um canal seguro serão apresentadas. Apesar de arquiteturas baseadas em túneis não implementarem um canal fim-a-fim, eles serão abordados na primeira seção 4.1, pois alguns autores (STREDICKE, 2006) defendem seu uso, além de estar contido nas formas previstas em por (ROSENBERG *et al.*, 2002) para se proteger um canal multimídia.

Na seqüência será apresentada a arquitetura padrão para o transporte seguro fim-a-

fim de dados multimídia, a pilha de protocolos SRTP/SRTCP (BAUGHER *et al.*, 2004), protocolo no qual existe um consenso sobre sua eficiência (STREDICKE, 2006).

De posse de como realizar o transporte do dado multimídia, serão estudadas algumas alternativas de como negociar de forma segura os parâmetros necessários para a criação de um canal de dados protegido. Esse estudo será finalizado com uma análise comparativa entre essas diversas arquiteturas, onde serão apontadas as principais potencialidades e vulnerabilidades de cada uma delas.

4.1 Arquiteturas de Proteção Baseado em Túnel

Na arquitetura de túnel criptográfico, também conhecida como *Virtual Private Network (VPN)*, as informações existentes nos protocolos VoIP (SIP/RTP) são encapsuladas em um novo pacote protegido, transportados e, no destino, desempacotados e entregues.

Essa solução é bastante interessante pelo fato de não necessitar de qualquer adaptação nas aplicações VoIP existentes, fazendo com que muitos fabricantes usem essa tecnologia como base para prover telefonia com segurança a um baixo custo.

Porém o uso de túneis tem uma série de inconvenientes quando aplicados em cenários VoIP. O primeiro deles diz respeito à necessidade de todos os elementos intermediários, que necessitem manipular o pacote em questão, devem possuir um relacionamento de confiança, caso contrário a comunicação seria impossível de ser realizada. Essa limitação faz com que túneis sejam muito difíceis de serem implementados em cenários onde o usuário não possui um controle sobre todo o canal onde irá ocorrer a comunicação.

O segundo inconveniente diz respeito ao aumento do tamanho dos pacotes de voz, ocasionado pela adição de novos cabeçalhos para prover segurança, uma vez que o IPv4

(versão 4 do protocolo IP) não possui uma segurança nata. Isso ocasiona, visando uma manutenção do nível de qualidade da voz, um aumento da vazão mínima necessária para trafegar o pacote, o que pode não ser viável em muitos ambientes.

No presente documento, serão comentadas duas técnicas de túneis para prover segurança a um ambiente VoIP. A primeira opção consiste em construir esse túnel na camada de rede usando o protocolo IPSEC (KENT; SEO, 2005), enquanto a solução alternativa prevê o mesmo desenvolvimento só que na camada de transporte, através do uso do protocolo TLS (DIERKS, 2006).

4.1.1 O Protocolo IPSEC

O IPSEC foi o primeiro protocolo a ser usado em larga escala na Internet para criar túneis criptográficos. Ele tem por finalidade garantir a privacidade e autenticidade do conteúdo do pacote IP, visando proporcionar ao usuário a capacidade de trafegar dados de natureza sensível pela Internet.

Para proporcionar isso, o IPSEC adiciona ao protocolo IP dois cabeçalhos auxiliares, o *Encapsulation Security Payload (ESP)*, cujo objetivo é proteger o conteúdo do pacote IP (payload) e o *Autentication Header (AH)*, cuja finalidade é garantir a possibilidade de identificar a origem e a adulteração de um pacote.

O protocolo também proporciona em sua especificação um modo seguro de realizar a negociação de parâmetros de segurança, os chamados *Security Association (SA)*.

Para proporcionar essa segurança, o IPSEC possui dois modos de funcionamento, conforme pode ser visto na figura 4.1.No primeiro modo, denominado modo de transporte, o IPSEC não altera o cabeçalho IP, fazendo com que dispositivos de interconexão que não

o suportem ou não pertençam a SA (4.1.1) envolvida, possam realizar o roteamento do pacote. Nesse modo apenas o payload do pacote IP é autenticado e cifrado.

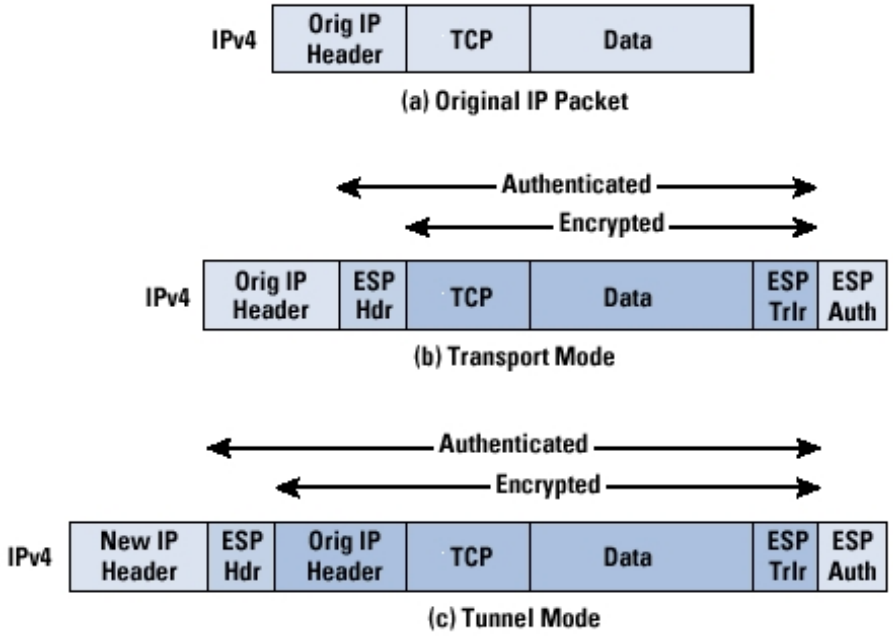

FIGURA 4.1 – Modos de Funcionamento do IPSEC. Fonte: (ANDREOLI, 2000)

Já no modo túnel, é criado um novo cabeçalho IP para o pacote, fazendo com que os endereços reais do originador e destinatário fiquem ocultos a dispositivos que não pertençam à SA (4.1.1) em questão. O maior inconveniente desse modo de funcionamento consiste no fato de que todos dispositivos envolvidos no processo de comunicação devem pertencer à mesma rede de segurança (SA).

Apesar de o IPSEC não necessitar de nenhum tipo de integração com a aplicação SIP, é preciso que os elementos intermediários que manipulam as informações contidas no payload IP pertençam ao mesmo domínio de confiança dos terminais envolvidos. Isso ocorre porque o SIP está localizado na camada de aplicação e as informações necessárias para que os *proxy's* e outros dispositivos possam encaminhar e tomar decisões adequadas, estão localizadas no payload do IP, que está criptografado, o que impede o uso do IPSEC para implementar uma proteção fim-a-fim.

Outro complicador é apresentado em (KUHN; WALSH; FRIES, 2005), onde é comentado o aumento da vazão mínima da rede, ocasionado pela adição da segurança IPSEC. Esta afirmação é fácil de ser comprovada, pois um pacote de voz ocupa em média 40 bytes, contabilizados os cabeçalhos IP, UDP e RTP. Esse mesmo pacote quando usado a compressão oferecida pelo RTP pode chegar a 03 (três) bytes. A inserção da proteção oferecida pelo IPSEC no pacote de voz, ocasiona um overhead de 40bytes ao pacote, fazendo que ao final seja necessário que a rede possua uma vazão média superior a 40Kbps, enquanto sem o uso dessa proteção, essa vazão é de apenas 9Kbps.

Em seu relatório, (KUHN; WALSH; FRIES, 2005) ainda aborda a impossibilidade de autenticar o originador de um pacote quando a rede usar mecanismos de conversão de endereços (*Network Address Translator - NAT*), o que impede o uso de algumas proteções oferecidas pelo protocolo.

Um problema adicional está associado à forma como o protocolo lida com as associações de segurança (SA) (4.1.1). Nele as relações de confiança geradas estão associadas aos dispositivos e não à sessão, fazendo com que todos os tráfegos, não somente os de voz tenham acesso ao serviço, gerando uma necessidade de vazão de rede adicional.

O aumento do tamanho do pacote pelo IPSEC limita bastante seu uso em aplicações multimídia. Em (ANWAR *et al.*, 2006) é demonstrado que para uma simples conferência entre dois pares usuários de *proxy's* diferentes, seriam necessárias a negociação de três SA. Este fato poderia acarretar um atraso na negociação de parâmetros da chamada de aproximadamente 20 segundos, o que em cenários reais de utilização de voz sobre IP é insustentável (OHTA, 2006).

Apesar de (ROSENBERG *et al.*, 2002) fazer referência ao IPSEC como uma alternativa para proteção do tráfego VoIP baseado na pilha SIP/RTP, no decorrer dessa seção fica

visível que esse protocolo não é uma alternativa eficiente para desenvolver esse ambiente seguro. Mesmo assim, com a incorporação do IPSEC no núcleo (kernel) da especificação do substituto do IPv4, uma nova análise desse cenário se faz necessária, uma vez que provavelmente as limitações atuais do seu uso devam ser superadas (KUHN; WALSH; FRIES, 2005), porém isso não é objeto de estudo desta tese.

4.1.2 O Protocolo TLS

Uma alternativa ao IPSEC para prover proteção de uma conferência multimídia é o TLS (DIERKS, 2006), protocolo esse que (ROSENBERG *et al.*, 2002) deixa explícito ao sua preferência.

Tanto o IPSEC quanto o TLS provêm os serviços de sigilo, autenticação e não repúdio. Diferentemente do IPSEC, o serviço de segurança do TLS é realizado na camada de transporte da pilha IP, sendo usado o protocolo TCP como base para tal tarefa.

Seu funcionamento consiste em uma fase inicial (*handshake*) onde serão negociados os parâmetros de segurança entre as partes, bem como a autenticação de cada elemento necessário para o estabelecimento do canal. Essa autenticação pode ser feita de forma bidirecional, onde todos os elementos se autenticam mutuamente (tanto servidores como os clientes), bem como de forma unidirecional, onde somente um dos pares é autenticado.

Durante a fase de *handshake* é negociada uma chave de sessão, que possibilita a transferência de dados entre os terminais envolvidos. Esse esquema é apresentado na figura 4.2.

Assim como o IPSEC, o TLS acarreta overhead a comunicação. Além disso seu uso depende também da existência de algum tipo de relacionamento de confiança entre os

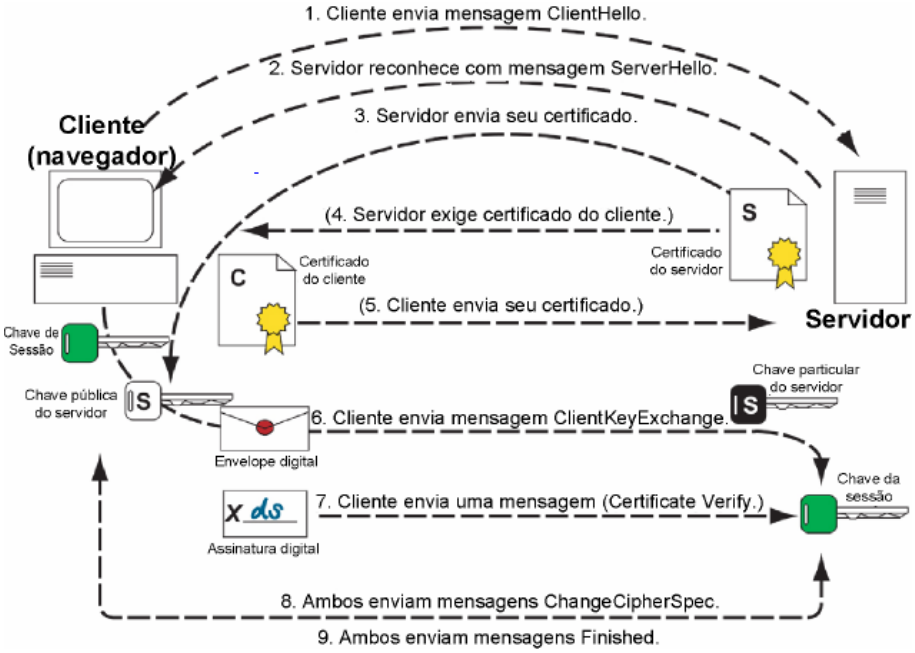

FIGURA 4.2 – Funcionamento do TLS. Fonte: (ANDREOLI, 2000)

elementos envolvidos no processo de comunicação, o que impossibilita seu uso em cenários fim-a-fim onde os pares não podem realizar esses relacionamento transitivo.

Uma vantagem do TLS em relação ao IPSEC é que ele não possui problemas em redes que usam NAT (4.1.1). Uma vez que o TLS protege apenas as informações da camada de transporte e superiores, o endereço IP irá trafegar em claro, possibilitando assim que um servidor NAT possa realizar a conversão devida.

Porém, (ROSENBERG *et al.*, 2002) potencializa o uso do TLS para prover a negociação dos parâmetros criptográficos necessários para a criação de um canal SRTP de forma protegida, ao invés desse tipo de túnel implementar a proteção inclusive do canal multimídia. Essa abordagem será vista em 4.3.1.

4.2 Arquitetura para Transporte Fim-a-Fim Seguro da Mídia - O Protocolo SRTP/SRTCP

Para prover o transporte fim-a-fim de uma mídia de forma segura foi desenvolvido a pilha de protocolos *Secure Real Time Protocol / Secure Real Time Control Protocol (SRTP/SRTCP)* (BAUGHER *et al.*, 2004), que diferentemente das arquiteturas de negociação dos parâmetros necessários para seu funcionamento, possui um certo consenso da comunidade científica sobre sua eficiência.

O SRTP/SRTCP é um perfil específico do RTP/RTCP (SCHULZRINNE *et al.*, 2003) que provê confidencialidade, integridade para todo o pacote RTP/RTCP, além de oferecer uma proteção contra ataques do tipo replay-packets (3.3.2).

Assim como no protocolo RTP/RTCP, a pilha ora apresentada é composta de dois protocolos específicos. O primeiro visa proteger o conteúdo das mensagens multimídia e se chama *Secure Real Time Protocol (SRTP)*. Já o segundo, visa proteger o conteúdo das mensagens de controle e se chama *Secure Real Time Control Protocol (SRTCP)*.

A pilha possui como principal característica a capacidade de interferir o mínimo nos pacotes multimídia originais, sendo uma boa opção aos protocolos de túnel (IPSEC e TLS). O overhead gerado ao pacote multimídia original (RTP) é de no máximo 64 bits e, nos pacotes de controle (RTCP), esse valor pode chegar a 96 bits. Quando comparado ao IPSEC, que gera um overhead de 40bytes fica claro a vantagem do seu uso.

Além de sua função, os pacotes SRTP e SRTCP diferenciam-se por mais um ponto. Enquanto no SRTCP é obrigatório a cifragem e autenticação dos pacotes, no SRTP, apenas a cifragem do payload multimídia é obrigatória, porém recomenda-se fortemente o uso da

autenticação, possibilitando que a funcionalidade de proteção contra replay-atacks (3.3.2) seja empreendida.

4.2.1 Conceitos Gerais do SRTP/SRTCP

O conceito mais importante para o funcionamento do SRTP/SRTCP é o *contexto criptográfico*. Um *contexto criptográfico* é uma coletânea de todas as informações necessárias para o estabelecimento de um canal seguro (chaves, protocolos envolvidos, etc.). Essas informações devem ser mantidas de forma segura pelos dois pares da comunicação durante o tempo de existência do referido canal.

É importante ressaltar que o contexto criptográfico não consiste apenas da informação da chave de sessão acordada, mas também de outras informações importantes para a criação do canal, como a descrição dos protocolos envolvidos, sua forma de funcionamento, número de pacotes que podem utilizar as referidas chaves de sessão, entre outras informações.

Para prover uma proteção contra os replays-atacks, o contexto criptográfico possui um vetor que armazena os pacotes autenticados mais recentes (*replay-list*), fazendo com que, ao ser recebido um pacote de voz, seja verificado o seu índice nesse vetor. Somente no caso desse índice não constar na *replay-list* é que o pacote será processado.

De forma a viabilizar a eficiência dessa funcionalidade, é necessário que todos os pacotes sejam autenticados. Sem essa autenticação, um atacante poderia inserir na rede pacote forjados, gerando um aumento de trabalho do processador do terminal alvo, fazendo com que ele tenha que dividir o processamento do áudio verdadeiro com o forjado, diminuindo dessa forma a qualidade da voz/vídeo recebido.

Desta forma, somente quando a autenticação de pacotes está ativada, é que a proteção contra *replay-atacks* é usada.

Outra variável existente no contexto criptográfico é a *taxa de derivação da chave*, que é um fator usado pelo protocolo para calcular quantos pacotes devem ser utilizados com o uso de uma única chave mestra.

Para aumentar a robustez do protocolo, o SRTP ainda utiliza um processo de derivação de chaves (4.2.5) onde, mediante uma única chave de sessão negociada entre os pares e transportadas por um protocolo de sessão, se gera um conjunto de chaves individuais que se encarregam da proteção (autenticação e cifragem) oferecida pelo protocolo.

Essa técnica de derivação provê uma segurança adicional ao processo, pois uma vez quebrada uma das chaves individuais, apenas as mensagens relacionadas a ela estarão comprometidas. Por exemplo, no caso de comprometimento da chave de autenticação, as mensagens de áudio ou vídeo trafegadas entre as partes ainda continuarão seguras.

Outra informação mantida no contexto criptográfico e que tem por finalidade aumentar a proteção das chaves existentes no processo de proteção oferecido é a *master salt*, ou *chave mestre de salgamento*. Segundo (TRAPPE; WASHINGTON, 2001), o salgamento de uma chave consiste no ato de preencher (*padding*) uma chave anteriormente calculada com valores aleatórios, fazendo com que um eventual atacante tenha mais dificuldade de calcular o seu valor.

No SRTP, conforme será analisado posteriormente, o processo de salgamento de uma chave é realizado pacote a pacote, onde o valor usado para o processo de proteção (chave de autenticação ou cifragem) é calculado através de um vetor inicial, a *Master Salt*.

Além das informações acima apresentadas, o contexto criptográfico contém outro atri-

buto chamado *rollover counter (ROC)*. O *ROC* é um índice que define quantas vezes o valor do número de seqüência do pacote SRTP foi zerado, já que esse índice é composto de 16 bits, possuindo um valor máximo de 65536. Esse valor irá ter outra grande importância no processamento dos pacotes SRTP, pois ele determinará o índice final do pacote, que é usado no processo de derivação da chave de cifragem e decifragem prevista pelo SRTP/SRTCP (4.2.5).

Deve-se lembrar que o processo de definição (negociação e estabelecimento) do contexto criptográfico ocorre anteriormente ao funcionamento dos protocolos SRTP/SRTCP, sendo considerado pelo mesmo como uma entrada esperada. Dessa forma, a construção desse contexto deve ocorrer através de um protocolo de negociação de chaves (Key Agreement) podendo ser associado à sinalização SIP (descrição no SDP) ou através de mensagens iniciais no canal multimídia (ZIMMERMANN; JOHNSTON; CALLAS, 2006), soluções essas que serão vistas posteriormente no presente capítulo.

4.2.2 Estrutura do Pacote SRTP/SRTCP

A estrutura dos pacotes SRTP/SRTCP é bem parecida com as dos protocolos que o originaram, o que proporciona ao projetista multimídia a capacidade de reutilizar os códigos anteriormente desenvolvidos para as ferramentas baseadas no RTP/RTCP.

A diferença do SRTP em relação ao RTP, reside no seu payload, que é cifrado, e dois novos cabeçalhos opcionais: um de autenticação (*authentication tag*) e outro para negociação de chaves (*Master Key Indicator (MKI)*), conforme é apresentado na figura 4.3.

Já o protocolo SRTCP, conforme apresentado na figura 4.4, possui algumas diferenças

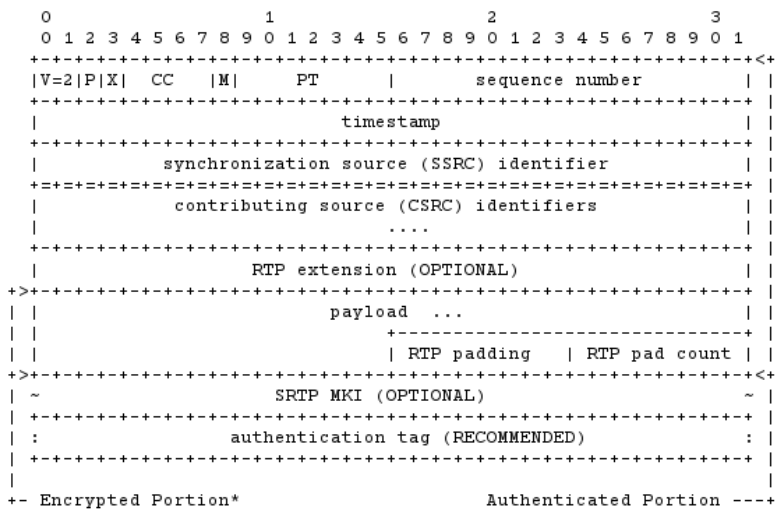

FIGURA 4.3 – Pacote SRTP.Fonte: (BAUGHER *et al.*, 2004)

mais marcantes quando comparado ao seu antecessor. A primeira dessas diferenças é o campo e-flag. Esse campo indica se o pacote SRTP está cifrado ou não, informando ao receptor se ele necessitará decifrar ou não a mensagem.

O segundo campo específico do cabeçalho SRTCP é o índice do pacote, que possui 31 bits e serve para identificar de forma única o pacote SRTCP. Esse índice nada tem a ver com os índices internos existentes nos pacotes que compõem a mensagem e deve ser iniciado por zero no primeiro pacote SRTCP enviado.

Na mensagem SRTCP ainda existem as tags de autenticação (*authentication tag*) e a *Master Key Indicator (MKI)*. Assim como no SRTP, a primeira é um hash criptográfico sobre uma parte da mensagem, enquanto a segunda é usada para trafegar uma nova chave mestra, sempre que a taxa de derivação de chaves (existente no contexto criptográfico) assim determinar.

Uma das características mais importantes de uma mensagem SRTCP é que seus primeiros 64 bits não são cifrados, garantindo a correta identificação do remetente da

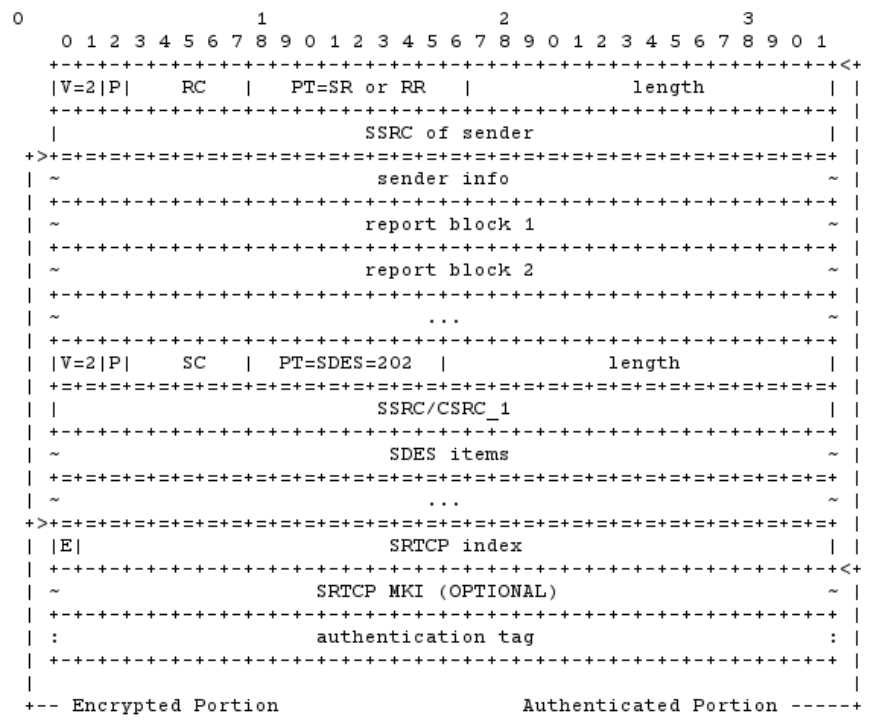

FIGURA 4.4 – Pacote SRTCP. Fonte: (BAUGHER *et al.*, 2004)

mensagem pelo recebedor, possibilitando a correta associação da mensagem ao contexto criptográfico, além de detectar o tipo do primeiro relatório existente em seu corpo.

Tanto o SRTP, quando usando a funcionalidade de autenticação de pacotes, como o SRTCP devem prover uma proteção contra *replay-atacks*. Isso cria a necessidade de existência de um repositório de índices de pacotes específico para cada tipo de mensagem recebida. Essa questão tem de ser analisada com critério em dispositivos de baixa capacidade de armazenamento (smartphones e celulares), pois em (SCHULZRINNE *et al.*, 2003) é definido que cada repositório deve ser capaz de armazenar pelo menos 64 índices .

4.2.3 Algoritmo de Cifragem do SRTP/SRTCP

O algoritmo de cifragem especificado na pilha SRTP/SRTCP objetiva a criptografia do payload multimídia (SRTP) ou de informações (SRTCP). Para isso ele usará um segmento criptográfico (*keystream segment*), gerado através de uma função criptográfica

(*keystream generator*), que sofrerá uma operação *XOR* com o payload em claro do pacote (SRTP/SRTCP), cifrando o mesmo.

A operação supra-citada é apresentada na figura 4.5, onde pode-se perceber que o *keystream segment* é dividido em um sufixo e um prefixo. O prefixo sempre é usado como semente para realizar a operação de autenticação do pacote, enquanto que o sufixo, para a operação de cifragem.

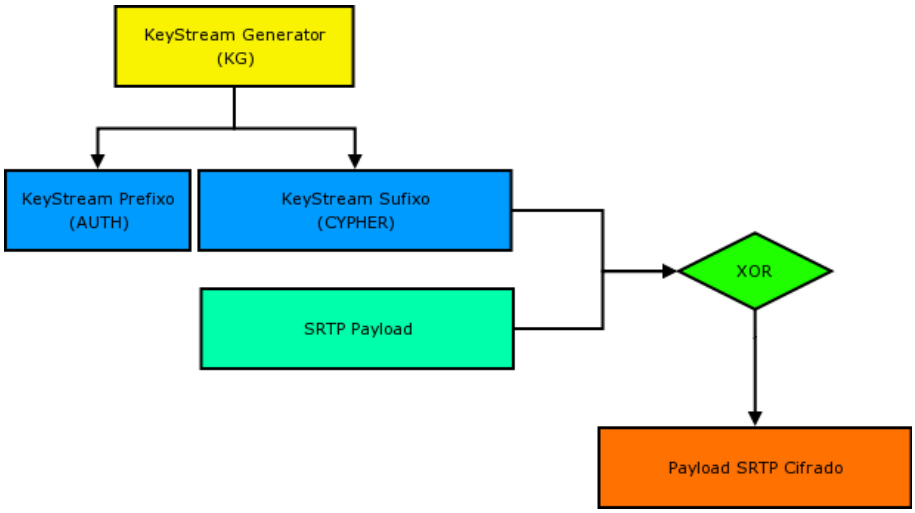

FIGURA 4.5 – Algoritmo de Cifragem do SRTP/SRTCP.

Como é usada a operação *XOR* para cifragem, o processo de decifragem fica muito simples de ser realizado, bastando apenas à aplicação novamente da mesma operação sobre o conteúdo cifrado, gerando dessa forma a mensagem original. Esse fato garante que ambos os terminais terão o mesmo custo computacional para realizar as operações de cifragem e decifragem das mensagens SRTP/SRTCP.

Na especificação SRTP é definido o AES como algoritmo criptográfico padrão, podendo ser usado tanto o modo Contador (*AES-CM*) como o f8 (*AES-f8*). Dado que apenas o modo *AES-CM* é de implementação obrigatória e é encontrado em um maior número de implementações, apenas ele será abordado no presente documento.

O AES é um algoritmo simétrico de criptografia, desenvolvido por *Vicent Rijmen*

e *John Daemen*, que o denominaram originalmente de *Rijndael*. Ele foi adotado como padrão para criptografia simétrica pelo *National Institute of Standards and Technology (NIST)* em 2000, substituindo o algoritmo *DES*, o que faz dele um algoritmo criptográfico obrigatório para aplicações seguras.

Apesar de o AES somente prever o uso de chaves de 128, 192 ou 256 bits, a especificação original, o *Rijndel*, não limita o tamanho das chaves a serem usadas. Porém, no caso do SRTP, esse valor é definido como 128 bits para todos os seus processos.

De uma forma simplificada, o processo de cifragem/decifragem do AES consiste de dez rodadas, onde cada rodada utiliza uma chave diferente, derivada da chave de sessão recebida como entrada pelo algoritmo.

No SRTP são usadas chaves e blocos de 128 bits, onde os blocos são organizados em matrizes de 4x4 bytes. Essas matrizes vão ser processadas em 10 rodadas diferentes, onde em cada rodada são efetuadas quatro operações (deslocamentos e permutações) sobre as linhas e colunas da matriz, sendo a única exceção a última rodada, onde em vez de quatro operações ocorrem apenas três.

O SRTP usa o modo *AES-CM* (ver figura 4.6). Nesse modo de funcionamento, o algoritmo AES é visto como uma caixa preta (**E**) que recebe de entrada uma chave de sessão (**K**) e um vetor inicial (**ctr**) para ser cifrado. Esse *vetor inicial (ctr)* é composto de um contador que é incrementado a cada pacote recebido. O resultado da cifragem sofre uma operação *XOR* com a mensagem que se deseja cifrar(**M**), gerando dessa forma o texto cifrado (**C**). O processo de decifragem ocorre de forma semelhante ao de cifragem.

Segundo (LIPMAA; ROGAWAY; WAGNER, 2006), esse modo de funcionamento aumenta a segurança do conteúdo a ser protegido, uma vez que, a cada pacote a ser proces-

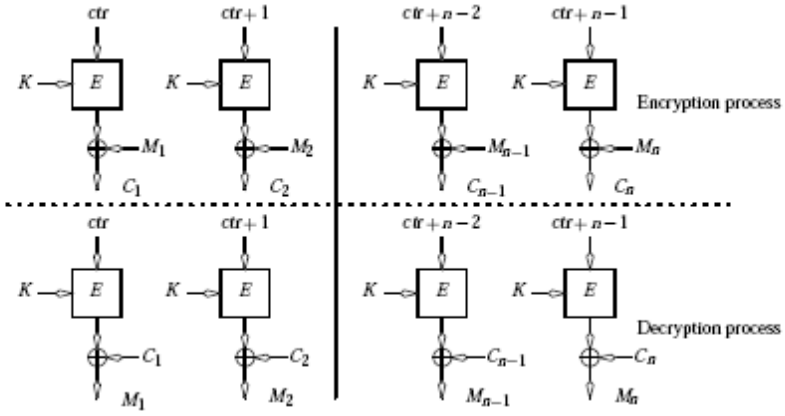

FIGURA 4.6 – AES Modo CTR. Fonte: (LIPMAA; ROGAWAY; WAGNER, 2006).

sado, uma composição de chaves e um vetor inicial diferentes são utilizados, fazendo com que o valor a ser operado (XOR) com o texto claro seja único e não se repita.

Porém, algumas dificuldades de implementação existem nesse modo. A primeira delas consiste na necessidade de compartilhamento dos contadores, além, é claro, das chaves de cifragem. Isto cria a necessidade de que essa informação seja também negociada no processo de construção do contexto criptográfico.

A segunda delas trata da propagação dos erros, já que um erro ocorrido na geração de algum contador intermediário pode perpetuar-se em todos os demais pacotes seqüenciais.

No SRTP, o segmento criptográfico (*keystream segment*) deve ser do tamanho do payload em claro RTP (conteúdo multimídia), visando dessa forma viabilizar a realização da operação XOR. Já que esse segmento tem um tamanho padrão (128 bits, saída do AES) e o payload não possui tamanho fixo, a sua composição pode necessitar de algum mecanismo de preenchimento (padding). Esse preenchimento é realizado através da concatenação do segmento gerado para cifrar o pacote atual com os segmentos gerados anteriormente,

sendo aproveitado os bits existentes nos segmentos criptográficos mais recentes.

$$S = E(k, IV)\|E(k, IV + 1 mod 2^{128})\|E(k, IV + 2 mod 2^{128})... \quad (4.1)$$

O processo de construção do segmento criptográfico (**E**) é apresentado na equação 4.1, onde é possível notar que o vetor de inicialização (IV), contador do modo CTR, é incrementado de uma unidade a cada pacote recebido. Também é fácil perceber que os bits a serem usados pelo SRTP em seus processos de cifragem e decifragem são sempre os existentes nas posições mais à esquerda, ou seja, os existentes nos pacotes mais rescentes.

$$IV = (ks * 2^{16})XOR(SSRC * 2^{64})XOR(i * 2^{16}) \quad (4.2)$$

A expressão 4.2 apresenta a composição do vetor de inicialização (IV) na pilha SRTP/SRTCP. Nesta expressão, o valor ks é a chave de salgamento da sessão, o $SSRC$ é o valor de 32 bits que identifica o remetente da mensagem (SCHULZRINNE *et al.*, 2003) e i o índice do pacote. A existência do índice do pacote para compor o vetor inicial faz com que o mesmo não possa ser pré-calculado, necessitando assim que essa informação seja extraída do pacote recebido e só nesse momento é possível calcular o seu valor, o que diferencia o modo CTR do SRTP do modelo apresentado em (LIPMAA; ROGAWAY; WAGNER, 2006).

4.2.4 Algoritmo de Autenticação do SRTP/SRTCP

O processo de autenticação do SRTP/SRTCP é realizado sobre todo o conteúdo existente em seu pacote. Ele tem por finalidade garantir a integridade do pacote e a correta

identificação do originador da mensagem, além de viabilizar a proteção contra os replay-atacks.

Apesar de ter sido definido que uma parte do segmento criptográfico (sufixo) será destinada para o funcionamento da autenticação, no modo de autenticação padrão do SRTP (HMAC-SHA1), esse segmento não é usado.

O HMAC-SHA1 pertencente a uma família de algoritmos MAC (3.2) desenvolvidos pela *National Security Agency (NSA)*, sendo a sua divulgação realizada em 1995. Ele produz uma mensagem autenticada de 160 bits e tem como entrada uma mensagem de tamanho variável (máximo de 2^{64} bits) e uma chave de 80 bits.

Apesar de estar em uso bastante intenso atualmente e previsto como protocolo de autenticação padrão no SRTP, existem vários estudos comprovando a sua fragilidade. Mesmo assim, o NIST anunciou que pretende usar o protocolo até 2010 (NIST, 2004), quando deve o substituir pela sua versão mais moderna, o HMAC-SHA2.

4.2.5 Algoritmo de Derivação de Chaves no SRTP/SRTCP

Conforme já apresentado, a pilha SRTP utiliza uma *chave mestra* e uma *master salt* para derivar as chaves que serão utilizadas no processo de autenticação e cifragem da mensagem, chaves essas cuja negociação estão fora do escopo do SRTP/SRTCP e devem ser negociadas de forma segura através de um mecanismo de *key agreement*.

De posse dessas chaves, o protocolo pode então gerar, através de seu algoritmo de derivação (ver figura 4.7), seis novas chaves distintas, que serão usadas para o transporte da mídia e para os relatórios de controle.

A primeira das chaves geradas é a a encr-key, que é usada no processo de cifragem e

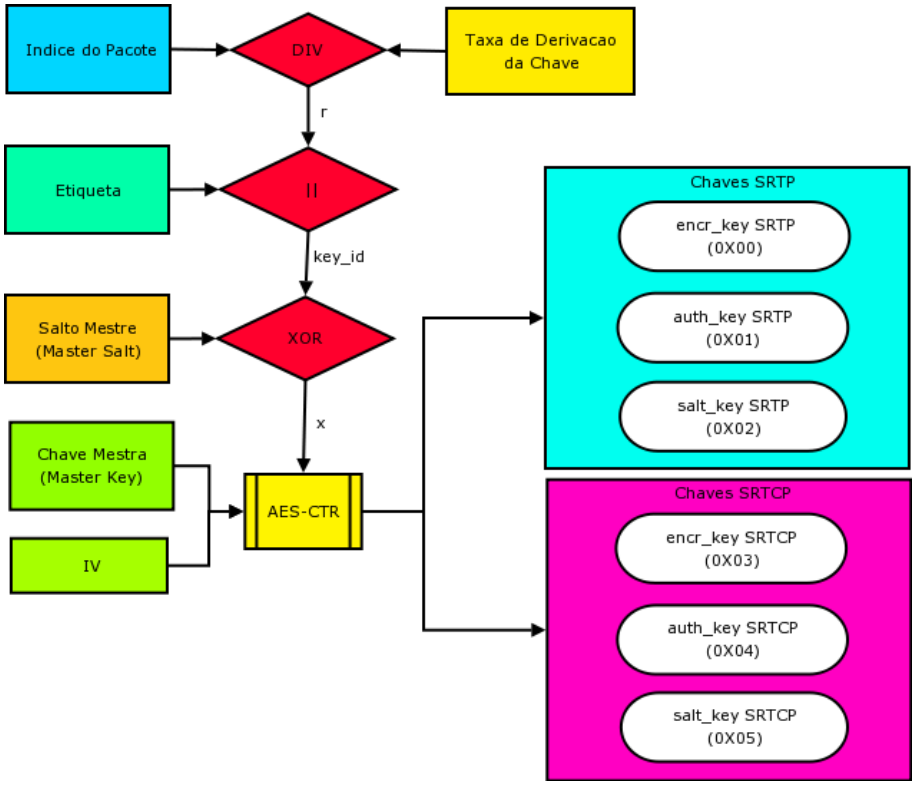

FIGURA 4.7 – Algoritmo de Derivação de Chaves.

decifragem dos payloads dos pacotes SRTP e SRTCP, a auth-key é utilizada no processo de autenticação, e a salt-key, o valor aleatório utilizado no processo de salgamento das diversas chaves antes do seu uso.

O algoritmo de derivação de chave inicia-se com o processo de divisão do índice do pacote (i) pela taxa de derivação da chave, negociada anteriormente e existente no contexto criptográfico. Essa operação matemática gera um novo índice chamado de r (4.3). Uma particularidade dessa operação de divisão (DIV) é que seu resultado será zero se a taxa de derivação da chave também for zero.

$$r = index(DIV)key_derivation_rate \tag{4.3}$$

De posse do valor do índice r, é possível calcular o valor da variável *key-id*, que é igual ao resultado da operação de concatenação de uma etiqueta pré-definida com o índice r

anteriormente calculado (4.5). A etiqueta usada nesse processo é uma constante escolhida com base no tipo de chave a ser criada pelo algoritmo de derivação (BAUGHER *et al.*, 2004).

$$key - id = etiqueta \| r \tag{4.4}$$

Com key-id calculado, pode-se determinar o valor da variável x, que é igual ao resultado da operação XOR entre *key-id* e a *master-salt* (4.5).

$$x = key - id(XOR)master - salt \tag{4.5}$$

Calculado o valor de x é finalmente possível gerar a chave de sessão desejada, que pode ser uma chave de cifragem, autenticação ou de salgamento (ver figura 4.7). O processo de geração da chave de sessão (session-key) consiste do resultado de uma função de cifragem baseada no algoritmo AES-CM (PRF) e tem como vetor de inicialização (IV) o valor $x * 2^{16}$. A expressão que denota a construção da chave de sessão é apresentada na equação 4.6.

$$session - key = PRF(master - key, x) \tag{4.6}$$

Um fato importante sobre o resultado da operação de construção da chave de sessão, é que ele será truncado nos seus n primeiros bits, fazendo com que ele tenha o tamanho planejado para ser a chave do algoritmo a ser utilizado. Como exemplo, pode-se pensar na referida função (PRF) para calcular a chave de sessão para cifragem de um payload SRTP. Nesse caso a session-key será truncada de forma a gerar uma chave de 128 bits.

Para seu correto funcionamento, o algoritmo AES-CM necessita como entrada básica de uma master-salt de *112 bits* e de uma chave-mestre de *128 bits*.

Nota-se que mesmo que o contexto criptográfico possua uma taxa de derivação de chaves igual a zero, o algoritmo de derivação de chaves será chamado pelo menos uma vez durante o funcionamento da pilha SRTP/SRTCP. Isso é devido à necessidade de geração de pelo menos um conjunto de chaves para uso no canal.

4.3 Arquitetura para Negociação de Contexto Criptográfico Fim-a-Fim

Na seção anterior foi apresentado uma arquitetura que realiza o transporte da voz e/ou vídeo de forma a garantir a privacidade do seu conteúdo e a correta identificação da origem dos dados envolvidos na conferência. Porém foi dito que a referida arquitetura precisava receber como entrada alguns parâmetros (contexto criptográfico) para que pudesse realizar seu trabalho, dados esses, que não podem ser transmitidos em claro.

A presente seção tem por finalidade abordar algumas soluções que visam realizar a negociação do contexto criptográfico de forma segura. Inicialmente serão abordadas as funcionalidades existentes na própria especificação original do SIP (ROSENBERG *et al.*, 2002). Na seqüência será apresentado a arquitetura MIKEY (ARKKO *et al.*, 2004) e para finalizar o estudo um protocolo experimental, o ZRTP (ZIMMERMANN; JOHNSTON; CALLAS, 2006).

4.3.1 Negociação de Chaves no SIP

A especificação do SIP (ROSENBERG *et al.*, 2002) prevê duas possibilidades para a negociação de um contexto criptográfico, através do uso do TLS (DIERKS, 2006) ou

usando cabeçalhos cifrados e/ou autenticados S/MIME (RAMSDELL, 1999).

O uso do TLS para a negociação de chaves é bem diferente de quando sua finalidade é implementar um túnel. Nesse caso, o TLS protegerá apenas as mensagens SIP, de forma a garantir que um contexto criptográfico possa ser transmitido de forma segura. Já em um túnel, o TLS protegerá não somente a sinalização, mas sim a conferência como um todo.

Para fazer uso do TLS no SIP, o usuário deve utilizar uma URL (2.2) especial chamada *SIPS*, que informa aos terminais envolvidos com a conferência que preferencialmente deve ser usado para realizar a sinalização SIP o protocolo TLS.

O uso do TLS na sinalização SIP afeta a escalabilidade das soluções VoIP, uma vez que, para o encaminhamento correto de uma chamada, o ambiente necessita que todos os *proxy's* necessitem ter algum relacionamento de confiança estabelecido.

Para contornar essa limitação, o suporte ao TLS não é obrigatório, sendo assim, um *proxy*, ao identificar que o próximo elemento intermediário (*proxy*) não pertence a nenhuma rede de confiança ou não possua suporte ao TLS, deve encaminhar a mensagem em claro, usando uma URL SIP comum. Isto possibilita que a mensagem consiga ser corretamente encaminhada, porém abre uma brecha para *downgrading atack's* (3.3.2).

A necessidade de criação de redes de relacionamento confiáveis pode fazer com que a existência de uma infra-estrutura de chaves públicas seja obrigatória e seja ajustada em tempo de projeto. Para contornar essa limitação, (ROSENBERG *et al.*, 2002) propõe o uso de *certificados auto-assinados*. Um certificado auto-assinado é aquele criado localmente e sem qualquer referência a uma Autoridade Certificadora, o que impossibilita garantir a validade e autenticidade do mesmo.

O uso de certificados auto-assinados possui como principal vulnerabilidade os ataques de *man-in-the-middle (MITM)*. Um *MITM* consiste em um ataque no qual o inimigo localiza-se entre a vítima e o seu par de comunicação, intercepta as suas mensagens, modifica seu conteúdo e se comunica com o par como se fosse a parte legítima. Esse ataque é possível quando o atacante consegue participar do processo de negociação de parâmetros criptográficos, momento em que ele negocia a sua chave fazendo-se passar pelas partes legítimas. Esse tipo de ataque pode ser visto na figura abaixo.

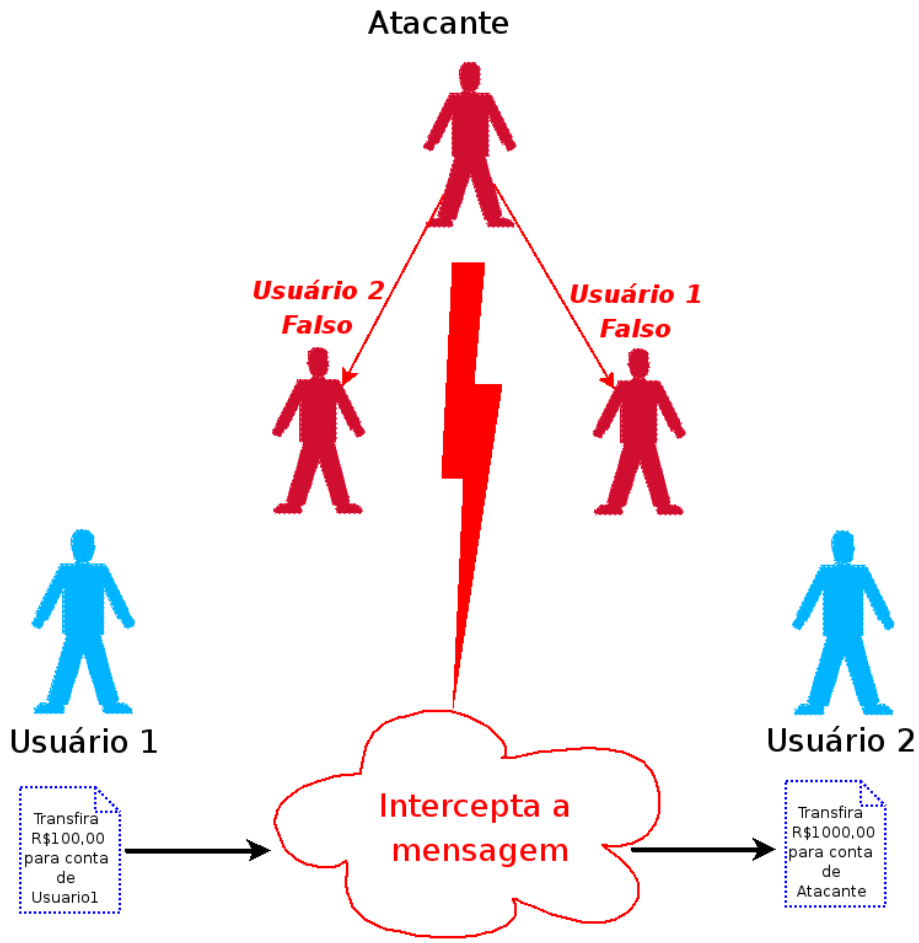

FIGURA 4.8 – Ataque do Homem do Meio (MITM).

Essa vulnerabilidade existente em certificados auto-assinados fez com que (ROSENBERG *et al.*, 2002), apesar de apresentá-lo como alternativa ao problema da estrutura de chaves públicas, desaprove fortemente o seu uso.

Além das limitações relacionadas à escalabilidade, o uso de mensagens SIPS oferece um problema de desempenho. Para que um canal seja sinalizado através do uso do TLS, é necessário que em cada *proxy* existente seja estabelecido uma relação de confiança antes de propriamente ser endereçada a primeira mensagem SIP (JENNINGS, 2004). O overhead gerado por esse processo acarreta um acréscimo de tempo para o estabelecimento de um canal que pode ser muito grande em enlaces com pequena banda disponível, fazendo o usuário desistir da chamada antes dela se completar (OHTA, 2006).

A alternativa baseada em TLS como pode-se perceber não implementa uma proteção fim-a-fim, pois necessita que os diversos proxy's tenham que possuir um segredo em comum (relação de confiança). Para prover uma arquitetura onde a proteçào é baseada apenas em segredos compartilhados unicamente pelos terminais VoIP envolvidos na conferência, (ROSENBERG *et al.*, 2002) propõe o uso do *Security Multiporpose Internet Mail Extensions (S/MIME)* (RAMSDELL, 1999).

O S/MIME consiste em uma sintaxe de mensagem cifrada para a transferência de conteúdo não textual (*MIME*). O padrão foi desenvolvido inicialmente para o envio de e-mail, mas pode ser utilizado em qualquer tipo de mensagem que necessite trafegar de forma protegida pela Internet, como por exemplo, o conteúdo dos cabeçalhos SIP e SDP.

O funcionamento do S/MIME é baseado no uso de chaves assimétricas e certificados para assinar e cifrar o conteúdo a ser protegido. Apesar de necessitar de uma infraestrutura de chaves públicas (*PKI*), quando comparada às soluções de VPN oferecidas pelo IPSEC e o TLS, possui a vantagem de não necessitar de nenhum controle do meio, bastando apenas que os usuários obtenham o certificado válido de seu par para poderem realizar uma comunicação segura.

Em (ROSENBERG *et al.*, 2002) é comentado que o S/MIME pode ser usado para

garantir a confidencialidade e integridade tanto das mensagens SIP como as SDP, porém, em ambos os casos, alguns cuidados no seu uso devem ser tomados.

Quando o S/MIME é aplicado para garantir a integridade ou o sigilo do cabeçalho SIP, deve-se lembrar que a modificação de alguns cabeçalhos por *proxy's* intermediários pode ser legítimo, como por exemplo quando o terminal usa um serviço de tradução de endereços (NAT). Essa particularidade pode ser resolvida não inserindo esses cabeçalhos no conteúdo a ser protegido. Isso garante, nos casos de assinatura, a semelhança no *fingerprint* gerado na origem e o recebido pelo destinatário. No caso de cifragem do conteúdo SIP e SDP, essa solução garante que os elementos intermediários possam encaminhar corretamente a mensagem recebida.

A segunda solução é específica para cifragem. Ela consiste na repetição das informações que podem sofrer modificação no *payload* protegido e no claro. O problema nesse caso, é que o usuário destino, ao usar os cabeçalhos repetidos, deve utilizar as informações externas ao conteúdo protegido, pois são elas que armazenam as informações manipulada pelos *proxy's*, que são as corretas.

Quando analisada a arquitetura SIP e SDP, percebe-se que o local onde deve ser inserida o contexto criptográfico é no corpo do SDP, pois esta informação descreve parâmetros do canal a ser criado pelo SIP. Dessa forma, quando usando o S/MIME, não há a necessidade de proteger todo a mensagem SIP, apenas o conteúdo do SDP.

Para garantir a integridade e autenticidade de um cabeçalho SDP, o S/MIME usa um tipo MIME chamado *signed-data*. Já quando o objetivo é garantir o sigilo sobre o conteúdo SDP, é usado outro tipo MIME chamado *enveloped-data*. Um tipo consiste em uma terminologia usada pelo protocolo MIME para especificar a natureza do dado inserido dentro do corpo de sua mensagem.

A figura abaixo apresenta uma mensagem SIP que contém o conteúdo SDP cifrado. Nela, um quadrado composto de asteriscos (*) é usado para identificar o conteúdo protegido pelo protocolo. Pode-se observar também que é usado o tipo S/MIME (smime-type) *enveloped-data*, identificando que o conteúdo da mensagem é cifrado.

```
INVITE sip:bob@biloxi.com SIP/2.0
Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bKnashds8
To: Bob <sip:bob@biloxi.com>
From: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 314159 INVITE
Max-Forwards: 70
Contact: <sip:alice@pc33.atlanta.com>
Content-Type: application/pkcs7-mime; smime-type=enveloped-data;
     name=smime.p7m
Content-Disposition: attachment; filename=smime.p7m
   handling=required

**********************************************************
* Content-Type: application/sdp                          *
*                                                        *
* v=0                                                    *
* o=alice 53655765 2353687637 IN IP4 pc33.atlanta.com    *
* s=-                                                    *
* t=0 0                                                  *
* c=IN IP4 pc33.atlanta.com                              *
* m=audio 3456 RTP/AVP 0 1 3 99                          *
* a=rtpmap:0 PCMU/8000                                   *
**********************************************************
```

FIGURA 4.9 – Mensagem SDP protegida pelo S/MIME. Fonte: (ROSENBERG *et al.*, 2002)

Apesar do S/MIME ser a única alternativa ofertada por (ROSENBERG *et al.*, 2002) para prover uma sinalização segura fim-a-fim, ela possui uma grande limitação quando o usuário iniciador não possui o certificado do destino. Nesse caso, é necessário usar algum mecanismo externo ao protocolo de forma a possibilitar a transferência desse certificado.

Na especificação SIP é dado como solução a esse problema o uso de requisições *OPTIONS* . Esse tipo de requisição tem por finalidade questionar um par remoto sobre suas capacidades. No caso do uso com S/MIME, o par remoto deve responder a requisição com uma mensagem OK, inserindo um atributo SDP *S/MIMECapabilities*, que apresenta o seu certificado, além de outras capacidades criptográficas suportadas pelo mesmo.

O problema do uso das mensagens OPTIONS consiste em sua vulnerabilidade a ata-

ques do tipo MITM (4.3.1), uma vez que a mensagem pode ser interceptada e adulterada por um atacante que esteja monitorando o canal.

4.3.2 Descrição de Parâmetros Criptográficos usando o SDP e o SDES

Para descrever o contexto criptográfico, o SDP (HANDLEY; JACOBSON; PERKINS, 2006) possui apenas um único atributo para realizar essa tarefa a transferência, o *encription key (k)* que deve transportar a chave que será usada como mestra na sessão a ser criada.

Caso o contexto criptográfico seja composto de mais informações, como ocorre no caso do SRTP/SRTCP, o esquema oferecido por (HANDLEY; JACOBSON; PERKINS, 2006) não é adequado. Para este objetivo desenvolveu-se em 2006 uma extensão do SDP original, que foi chamada de *SDES* (ANDREASEN; BAUGHER; WING, 2004).

O SDES usa o mesmo modelo oferta/resposta do SDP original, onde o remetente propõe os parâmetros que deseja estabelecer no canal multimídia e o destinatário responde com os parâmetros aceitos, que podem ser os mesmos existentes na oferta ou novos escolhidos pelo destinatários.

No protocolo SDES a descrição do contexto criptográfico é realizada através do atributo chamado *crypto*, formado por uma etiqueta (*tag*), pela descrição do algoritmo de criptografia desejado (*crypto-suite*), alguns parâmetros da chave e, opcionalmente, outros parâmetros específicos da sessão. Esse atributo é apresentado na equação 4.7.

$$a = crypto :< tag >< crypto - suite >< key - params > [< session - params >] \quad (4.7)$$

O campo *tag* tem por finalidade identificar de forma única um atributo criptográfico dentro do escopo do SDP. Isso é necessário, pois no processo de oferta do protocolo, o remetente pode enviar mais de um atributo *crypto*, de forma a possibilitar ao destinatário a escolha do mecanismo de construção do contexto criptográfico mais adequado a sua realidade. Nesse caso, o destinatário, ao responder a requisição, deve informar com quais atributos ele concorda em realizar a conversação, sinalizando a *tag* para isso.

Outro atributo existente na descrição do contexto criptográfico é o *crypto-suite*. Esse atributo identifica o algoritmo usado para autenticar e cifrar a mensagem, bem como os respectivos parâmetros utilizados (tamanhos de chaves, master salt, etc).

Na figura 4.10 é apresentado o conjunto de algoritmos criptográficos padronizados em (ANDREASEN; BAUGHER; WING, 2004). Por exemplo, se um usuário receber uma mensagem com o atributo *crypto-suite* igual a *AES_CM_128_HMAC_SHA1_80*, significa que seu par deseja realizar uma conferência SRTP usando o algoritmo AES no modo CTR com chaves de 128 bits para cifragem e, para autenticação, o algoritmo HMAC-SHA1, onde a tag de autenticação terá 80 bits.

O segundo campo do atributo crypto a ser definido é o *key-param*, que define a chave a ser negociada, é construído de acordo com a expressão 4.8. A primeira parte da referida expressão identifica o método usado (*key-method*) para transportar a chave, que no caso do SDES é definido apenas o *inline*, que informa à aplicação que a chave é transportada no próprio conteúdo da *key-param*.

$$< key - method > \ '' : '' < key - info > \tag{4.8}$$

No caso de o método ser *inline*, a *key-info* é composta da chave a ser negociada e de

```
INVITE sip:bob@biloxi.com SIP/2.0
Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bKnashds8
To: Bob <sip:bob@biloxi.com>
From: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 314159 INVITE
Max-Forwards: 70
Contact: <sip:alice@pc33.atlanta.com>
Content-Type: application/pkcs7-mime; smime-type=enveloped-data;
     name=smime.p7m
Content-Disposition: attachment; filename=smime.p7m
   handling=required

*******************************************************
* Content-Type: application/sdp                       *
*                                                     *
* v=0                                                 *
* o=alice 53655765 2353687637 IN IP4 pc33.atlanta.com *
* s=-                                                 *
* t=0 0                                               *
* c=IN IP4 pc33.atlanta.com                           *
* m=audio 3456 RTP/AVP 0 1 3 99                       *
* a=rtpmap:0 PCMU/8000                                *
*******************************************************
```

FIGURA 4.10 – Algoritmos Criptográficos SDES (crypto-suite). Fonte: (ANDREASEN; BAUGHER; WING, 2004)

uma série de informações complementares à mesma. Uma vez que o SRTP necessita de duas chaves para realizar o algoritmo de derivação, esse campo é composto por uma chave simétrica concatenada (||) à master salt.

Após a chave, separada por um símbolo "|", no campo key-info se insere, opcionalmente, o tempo de vida da chave (*lifetime*) e a *MKI* com o seu tamanho (lenght), conforme a expressão 4.9.

$$''inline:'' < key||salt > [''|''lifetime][''|''MKI'' : ''lenght] \tag{4.9}$$

Um exemplo do uso desse campo é apresentado na figura 4.11. Nesta figura percebe-se que o método usado é o *inline*. Na seqüência é apresentada a chave concatenada à master salt. Para possibilitar a identificação do conteúdo que compõe cada uma delas, é necessário conhecer o algoritmo usado (*crypto-suite*).

Após a definição da chave é apresentado o tempo de vida da mesma. Esse campo

```
inline:d0RmdmcmVCspeEc3QGZiNWpVLFJhQX1cfHAwJSoj|2^20|1:4
```

FIGURA 4.11 – Exemplo de uso do campo *key-info*.Fonte: (ANDREASEN; BAUGHER; WING, 2004)

representa o número de pacotes que podem ser processados pela chave transferida pelo método *inline*. Para evitar que seja inserido valores muito altos, essa informação é apresentada como um inteiro formatado na potência de 2. No caso da figura acima é informado que a chave trafegada pode cifrar/decifrar até 2^{20} pacotes, ou seja, 1.048.576 pacotes.

O último campo que pode ser inserido diz respeito à MKI. Esse campo informa se a MKI está presente ou não nos pacotes SRTP e qual o seu tamanho. No caso de estar presente é apresentado o valor 1 seguido do símbolo ":" e do tamanho desse campo em bytes no supracitado protocolo.

Apesar de não ser de uso obrigatório, a última parte do atributo crypto (*session-param*) é muito importante, pois é ele que possibilita que as demais informações necessárias para a criação do contexto criptográfico sejam definidas. Quando o *session-param* não é definido, é necessário que as aplicações envolvidas compartilhem previamente essas informações ou definam algum tipo de padrão.

Apesar de o SDES apresentar outros parâmetros de sessão que podem ser negociados, as de interesse ao contexto a ser utilizado são as relacionadas à taxa de derivação da chave, a informação se o pacote SRTP será autenticado ou não e o tamanho do buffer de proteção contra *replay-atacks*.

Para descrever a taxa de derivação da chave, o SDES usa o atributo **KDR=n**, onde o n é um valor inteiro que representa uma potência de 2, como por exemplo 2^{24}.

Já para informar que o SRTP não suportará autenticação, deve-se usar o atributo **UNAUTHENTICATED_SRTP**. No caso onde o protocolo suporta autenticação, basta

omitir essa informação.

Por fim, o tamanho mínimo do buffer de proteção contra replay-atacks é padronizado na especificação do SRTP (BAUGHER *et al.*, 2004), porém pode ser necessário, devido a requisitos existentes em aplicações específicas que esse valor seja maior. Nesses casos deve-se usar o atributo *Window Size Hint* (**WSH=n**), onde o *n* é um inteiro que define o tamanho do referido buffer.

Apesar dos atributos de sessão serem bastante importante para flexibilizar a construção de clientes SIP seguros, ainda existe um último atributo necessário na inicialização do SRTP que não foi definido por nenhum dos atributos existentes no SDES.

Para inicializar o SRTP, além dos atributos definidos na presente seção, são necessárias três outras informações: o *SSRC*, o *ROC* e o *número de seqüência do pacote* (*SEQ*).

Uma vez que essas informações são necessárias para a inicialização do protocolo, porém só estão disponíveis após seu início, (ANDREASEN; BAUGHER; WING, 2004) propõe uma função chamada *late binding of one or more SSRC's to a crypto context*. A idéia é, através da recuperação do endereço IP do usuário e da sua porta existentes respectivamente nos atributos "c" e "m" do SDP, compor temporariamente o contexto criptográfico. No momento em que essas informações estiverem disponíveis, no recebimento dos primeiros pacotes SRTP/SRTCP, o contexto criptográfico é atualizado.

Para uma melhor compreensão de como o protocolo SDES é usado são apresentadas a seguir a figura 4.12 e 4.13. Na primeira figura é apresentada uma mensagem de oferta, ou seja, a mensagem SDES que será inserida no corpo de uma requisição SIP inicial, como por exemplo um INVITE. Nessa mensagem pode-se perceber que são negociados dois atributos crypto, informando ao destinatário (*userB*) que o usuário remetente (*userA*)

deseja conversar usando o algoritmo AES no modo CTR ou no modo F8.

```
v=0
o=sam 2890844526 2890842807 IN IP4 10.47.16.5
s=SRTP Discussion
i=A discussion of Secure RTP
u=http://www.example.com/seminars/srtp.pdf
e=marge@example.com (Marge Simpson)
c=IN IP4 168.2.17.12
t=2873397496 2873404696
m=audio 49170 RTP/SAVP 0
a=crypto:1 AES_CM_128_HMAC_SHA1_80
 inline:WVNfX19zZW1jdGwgKCkgewkyMjA7fQp9CnVubGVz|2^20|1:4
 FEC_ORDER=FEC_SRTP
a=crypto:2 F8_128_HMAC_SHA1_80
 inline:MTIzNDU2Nzg5QUJDREUwMTIzNDU2Nzg5QUJjZGVm|2^20|1:4;
 inline:QUJjZGVmMTIzNDU2Nzg5QUJDREUwMTIzNDU2Nzg5|2^20|2:4
 FEC_ORDER=FEC_SRTP
```

FIGURA 4.12 – Mensagem SDES Inicial.Fonte: (ANDREASEN; BAUGHER; WING, 2004)

Na figura 4.13 é apresentada a resposta a mensagem de oferta apresentada na figura acima. Nela percebe-se que além do *userB* ter escolhido o contexto apresentado na *tag 1* (AES-CM), ele insere um novo valor de chave, que deve ser usado para o *userA* cifrar e autenticar as mensagens endereçadas a *userB*.

```
v=0
o=jill 25690844 8070842634 IN IP4 10.47.16.5
s=SRTP Discussion
i=A discussion of Secure RTP
u=http://www.example.com/seminars/srtp.pdf
e=homer@example.com (Homer Simpson)
c=IN IP4 168.2.17.11
t=2873397526 2873405696
m=audio 32640 RTP/SAVP 0
a=crypto:1 AES_CM_128_HMAC_SHA1_80
 inline:PS1uQCVeeCFCanVmcjkpPywjNWhcYD0mXXtxaVBR|2^20|1:4
```

FIGURA 4.13 – Mensagem SDES de Resposta. Fonte: (ANDREASEN; BAUGHER; WING, 2004)

Apesar de o SDES apresentar uma forma clara e bem completa de negociar um contexto criptográfico, sua maior limitação é a sua necessidade de ser transportado através de um canal seguro. Esse fator limita bastante o seu uso, pois as alternativas apresentadas até o presente momento (TLS, IPSEC, S/MIME) apresentaram uma infinidade de problemas de implementação.

Na próxima sub-seção será apresentado o MIKEY (ARKKO *et al.*, 2004), que apresenta uma forma segura de negociar o contexto criptográfico, fazendo com que a mesma possa ser realizada em um canal desprotegido, o que é um grande diferencial em relação à arquitetura SDES.

4.3.3 O Multimedia Internet KEYing - MIKEY

Diferentemente do SDES, o Multimedia Internet KEYing (MIKEY) (ARKKO *et al.*, 2004), tem por finalidade não só descrever os parâmetros necessários para a construção do contexto criptográfico, mas também implementar uma proteção a esses atributos, tornando possível seu transporte em um canal desprotegido.

Apesar de sua arquitetura ter sido planejada para ambientes multimídia complexos (um pra um, um pra muitos ou muitos pra muitos), no presente documento o protocolo será analisado apenas no cenário *um pra um*, que consiste em uma conferência formada por somente um par de usuários.

Uma das grandes virtudes do MIKEY é a sua capacidade de negociação dos parâmetros criptográficos em uma única rodada, ou seja, em uma única troca de mensagem de oferta e aceitação (ou recusa), possibilitando assim sua inserção no SDP (ARKKO *et al.*, 2006) e ocasionando uma mínima modificação nos protocolos de sinalização pré-existentes.

O protocolo MIKEY define em sua especificação original ((ARKKO *et al.*, 2004)) três formas para realizar a negociação e transporte do contexto criptográfico: usando uma chave compartilhada (MIKEY-PS), com chaves públicas (MIKEY-PK) e com uso do algoritmo Diffie-Hellman (MIKEY-DH). Devido a limitações existentes nos modos anteriormente citados, foram desenvolvidos mais dois modos de negociação e transporte com-

plementares: usando chaves públicas reversas (MIKEY-RSA-R) e usando Diffie-Hellman com autenticação baseada em chave compartilhada (MIKEY-DHHMAC).

No decorrer desta seção, todos os modos de negociação vistos anteriormente serão abordados em detalhes. Suas principais funcionalidades, formato de suas mensagens e limitações serão analisados, possibilitando a análise comparativa com os demais protocolos realizada no final do presente capítulo.

4.3.3.1 Modo de Transporte baseado em Chave Compartilhada (MIKEY-PS)

A forma mais simples de realizar a negociação de parâmetros usando o MIKEY é através do transporte baseado em chave compartilhada (MIKEY-PS). A estrutura de funcionamento de uma negociação baseada no MIKEY-PS é apresentada na figura 4.14 e tem como principal característica o fato de que as partes envolvidas possuem uma única chave simétrica acordada antes do início da sinalização.

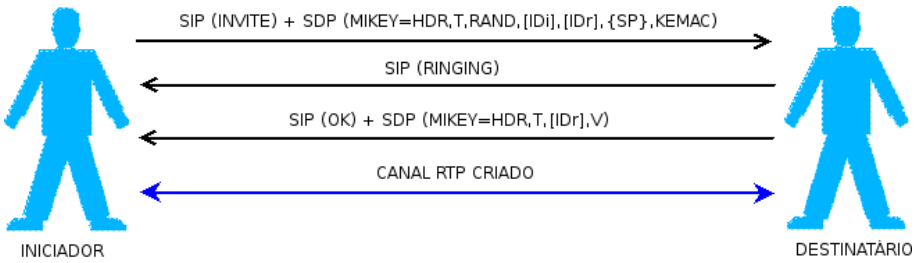

FIGURA 4.14 – Funcionamento do MIKEY-PS.

Apesar de sua simplicidade, esse modo sofre um grande fator limitante, já que o usuário precisa possuir uma chave pré-acordada para cada terminal que ele deseja estabelecer, mesmo que esporádicamente, uma conferência. Esse fato faz com que esse mesmo usuário para realizar uma conferência com um total de n pares remotos, precise manter localmente um total de n-1 chaves. Quando analisado o esforço de troca de chaves de todos os usuários do canal é necessário um total de $n(n-1)/2$ trocas de chaves, o que é um esforço

excessivamente grande em cenários com grande número de usuários.

Conforme apresentado na figura 4.14, o usuário que inicia a conferência envia uma mensagem de iniciação que contém obrigatoriamente um cabeçalho padrão de inicialização (*HDR*), um marcador de tempo (*timestamp*) da mensagem (T), um valor aleatório ($RAND$), um payload que contém o conteúdo a ser protegido ($KEMAC$) e um ou mais parâmetros criptográficos (SP).

Além dos cabeçalhos acima apresentados, pode-se inserir em uma mensagem MIKEY a identificação do iniciador (IDi) e do destinatário da mensagem (IDr), que podem ser seus certificados ou um alias que os identifique.

Um dos cabeçalhos mais importantes da mensagem de iniciação é o *KEMAC*. É ele que transporta de forma protegida a chave mestre (*TEK Generation Key*). Uma *TEK* pode ser entendida como *Traffic-Encrypting Key*, que é a chave de sessão que se deseja negociar.

O KEMAC é composto da TGK (4.3.3.1) cifrada através de um algoritmo simétrico (*AES-CM* - ver 4.2.3) e de uma chave de cifragem (enc_key), que é derivada da chave compartilhada. Esse valor, conforme pode-se ver na expressão 4.10, é concatenado ao cabeçalho autenticado (MAC), que é calculado com base em toda a mensagem MIKEY.

$$KEMAC = E(enc\_key, TGK)||MAC \tag{4.10}$$

Uma informação importante é que o processo de cifragem e de autenticação do pacote MIKEY é realizado através da derivação da chave compartilhada entre os pares. Essa derivação será alvo de análise na sub-seção 4.3.3.6.

Uma vez recebida a mensagem de inicialização, o par remoto deverá analisar a mensagem, extraindo as diversas informações existentes na mesma e criar seu contexto criptográfico. De posse desse contexto, ele deve enviar ao iniciador uma mensagem de resposta, que conterá a confirmação de que a mensagem inicial foi devidamente recebida, processada e acordada.

Conforme apresentado na figura 4.14, essa mensagem é composta de uma série de cabeçalhos padrões. Um deles é o *timestamp* (T), que deve ser o mesmo do encontrado na mensagem de iniciação, garantindo ao iniciador uma forma de descobrir a qual mensagem se refere a resposta recebida, no caso de ele ter enviado mais do que uma requisição.

Existe ainda na resposta um campo de verificação (V), que é um cabeçalho de autenticação calculado com base em toda a mensagem MIKEY de resposta, garantindo ao iniciador a correta identificação do usuário que construiu a resposta.

4.3.3.2 Modo de Transporte baseado em Chave Pública (MIKEY-PK)

O segundo modo de transporte existente no MIKEY é o que realiza a negociação e o transporte dos parâmetros criptográficos através do uso de chaves públicas e privadas, o *MIKEY-PK*. Esse modo de funcionamento tem como principal vantagem a escalabilidade que dá ao sistema VoIP, pois ele possibilita a negociação das chaves em ambientes bastante vastos e heterogêneos, bastando para isso que exista uma infra-estrutura de chaves públicas (PKI).

Um fator complicador desse modo de funcionamento é que os usuários envolvidos necessitam antecipadamente possuir o certificado de seu par, pois sem esse certificado o iniciador não consegue proteger (cifrar) a TGK (4.3.3.1) a ser negociada.

Sem o conhecimento prévio do certificado do par remoto, para ser possível o uso do MIKEY-PK, é necessário que a aplicação possua algum mecanismo externo de consulta segura a um repositório público de chaves, fazendo com que o número de rodadas necessárias para seu funcionamento aumente.

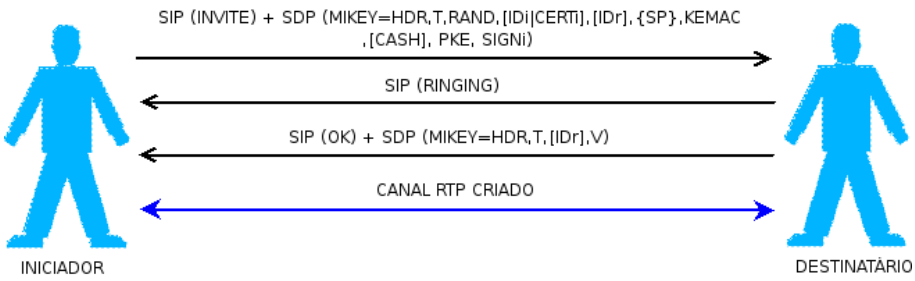

FIGURA 4.15 – Funcionamento do MIKEY-PK.

A figura 4.15 apresenta a estrutura de funcionamento do *MIKEY-PK*. Nela percebe-se que o iniciador deve enviar uma mensagem com uma estrutura semelhante à existente no modo *MIKEY-PS*. A diferença básica consiste no fato de as mensagens MIKEY serem autenticadas e cifradas usando uma chave derivada de uma *enveloped key* (*env_key*). Uma *enveloped key* é um repositório protegido pela chave pública do usuário destino (*PKd*) e encapsulada em um novo cabeçalho chamado *PKE* (*Envelope Data Payload*), conforme é apresentado na expressão 4.11.

$$PKE = E(Pkd, env\_key) \tag{4.11}$$

Outro cabeçalho importante existente na mensagem de iniciação do MIKEY-PK é o *CHASH*, que contém o certificado ou chave pública usada para cifrar a mensagem. Esse cabeçalho é usado sempre que o par remoto possuir mais de uma chave pública ou certificado, informando ao destino qual deles foi usado para gerar a *PKE*.

Diferentemente do modo MIKEY-PS, o cabeçalho *KEMAC* (ver expressão 4.12)possui

uma composição diferente, que é apresentada abaixo. Além dessa diferença, o campo autenticado (MAC) é referente apenas à mensagem $KEMAC$ e não a todo o conteúdo MIKEY, sendo usado para essa função o cabeçalho *SIGNi*, que é assinado usando a chave privada do iniciador.

$$KEMAC = E(enc\_key, Idi||TGK)||MAC \tag{4.12}$$

A resposta construída pelo par remoto é exatamente igual ao processo existente em MIKEY-PS.

4.3.3.3 Modo de Transporte baseado em Diffie-Hellman (MIKEY-DH)

A última alternativa existente no MIKEY para o transporte seguro de um contexto criptográfico consiste na negociação das chaves usando o método proposto por *Diffie-Hellman* (DH).

O Diffie-Hellman (DH) consiste em um algoritmo onde os pares de uma comunicação conseguem compartilhar de forma segura um segredo em um canal desprotegido sem a necessidade de qualquer mecanismo de proteção anteriormente negociado.

A segurança desse algoritmo é baseada na dificuldade de se resolver o chamado problema do logaritmo discreto. Para descrever o problema do logaritmo discreto se faz necessário definir o conceito de *raiz primitiva*. Segundo (TRAPPE; WASHINGTON, 2001), a raiz primitiva módulo de um número primo (p) consiste em um valor que, quando elevado a inteiros menores que p e reduzido módulo p ($mod\ p$), gera valores distintos de *1* a *p-1*. Por exemplo, se a for uma raiz primitiva de p, então $amod(p), a^2mod(p), ..., a^{p-1}mod(p)$ são distintos e percorrem todos os inteiros, desde 1 (um) até p-1.

De posse desse conceito é possível enunciar o problema do logaritmo discreto. Sejam a, b e p números inteiros, com p primo. O logaritmo discreto módulo p de b na base a é o número inteiro i, situado no intervalo $0<=i<=(p-1)$ e tal que $b = a^i mod(p)$. Dados a, p e i, é muito simples calcular o b. Todavia, se a, b e p forem escolhidos adequadamente, pode ser muito difícil calcular o expoente i.

O funcionamento do algoritmo Diffie-Hellman é apresentado na figura 4.16. Nela percebe-se o trabalho do iniciador (*Alice*), que deve escolher aleatoriamente um número primo (p) e uma raiz primitiva (g). De posse desses valores, Alice então calcula o valor público $A = g^a mod(p)$, onde a é selecionado em segredo e aleatoriamente por Alice. Uma vez calculado A, Alice envia a seu par remoto (Bob) os valores de A, p e g.

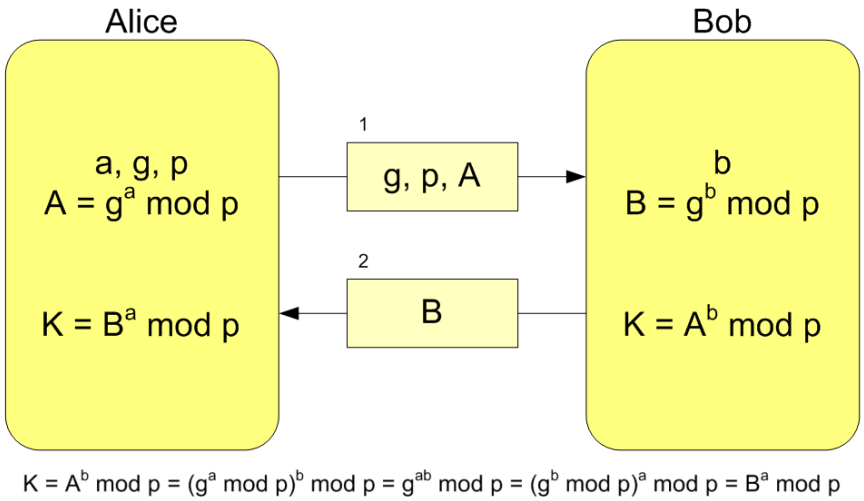

FIGURA 4.16 – Algoritmo Diffie-Hellman. Fonte: (WIKIPEDIA, 2007a)

Ao receber os valores enviados por Alice, Bob realiza localmente o mesmo trabalho realizado por seu par: escolhe aleatoriamente um número inteiro b e o mantém em segredo, calculando com ele o valor público $B = g^b mod(p)$ que envia para Alice.

Quando Bob recebe A, pode calcular $A^b mod(p) = g^{ab} mod(p)$. Quando Alice recebe B, pode calcular $B^a mod(p) = g^{ab} mod(p)$ fazendo com que ambos possuam o mesmo número

$g^{ab}mod(p)$. Se p, g, a e b forem escolhidos convenientemente, alguém que tenha observado a troca de informações, em posse de p, g, A e B não é capaz de calcular $g^{ab}mod(p)$.

A maior virtude encontrada no algoritmo *Diffie-Hellman* é o fato de ele possuir a propriedade *perfect forward secrecy* (3.3.2). Em suas chaves, o DH utiliza apenas valores aleatórios criados no momento de sua necessidade, descartando-os imediatamente após a sua demanda. Outras tecnologias, como aquelas baseadas em infra-estrutura de chave públicas, usam sempre o mesmo conjunto de chaves para realizar as operações de cifragem e autenticação.

Apesar de possuir a funcionalidade apresentada no parágrafo anterior, o Diffie-Hellman tem um grande problema, que é sua vulnerabilidade ao ataque de MITM (4.3.1). A figura 4.17 ilustra um ataque de MITM sobre o Diffie-Hellman. Nela existe um atacante (Eve) que, usando um analisador de redes (*sniffer*), captura os pacotes negociados entre os pares legítimos da comunicação (Alice e Bob). Esses pares em vez de criarem um segredo comum a eles, o fazem com o atacante (Eve) que, desta forma, passa a ter o total controle sobre o canal.

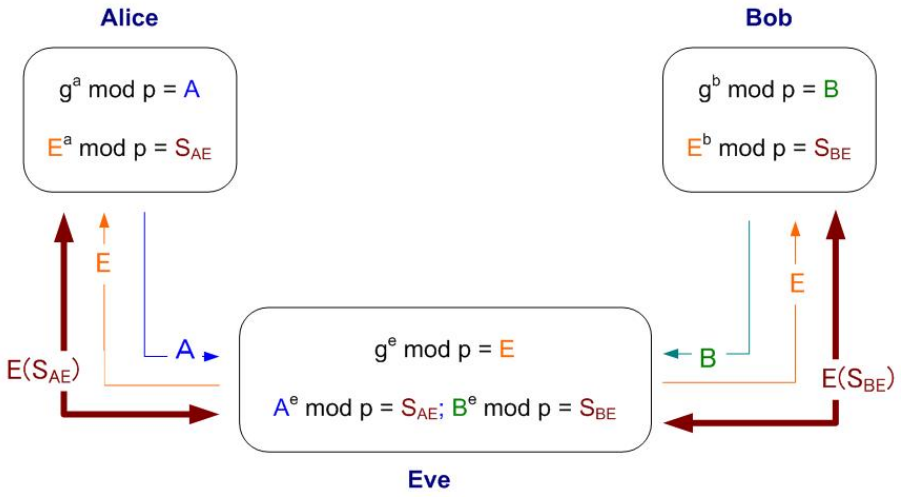

FIGURA 4.17 – Ataque de MITM sobre o Diffie-Hellman. Fonte: (CHEN, 2006)

Uma vez entendido o funcionamento do Diffie-Hellman é possível explicar como ele é

aplicado no MIKEY. A maior vantagem de usar esse algoritmo para transportar o contexto criptográfico através de uma mensagem MIKEY (*MIKEY-DH*) é que ele possibilita aos pares a troca de um segredo comum usando um canal público sem qualquer tipo de segredo previamente agendado.

Por contornar sua vulnerabilidade a ataques do tipo MITM, o algoritmo usa certificados para assinar as mensagens, o que gera ao MIKEY-DH os mesmos problemas encontrados no MIKEY-PK.

O funcionamento do MIKEY-DH (4.18) é similar às outras duas alternativas de transporte, possuindo no corpo de sua mensagem de iniciação os mesmos cabeçalhos *HDR*, *T*, *RAND*, *IDi*, *IDr*, *SP* e *SIGNi*. A diferença desse método para os demais consiste no envio do parâmetro *DHi* (gx^i), que será usado para calcular o segredo compartilhado pelos pares.

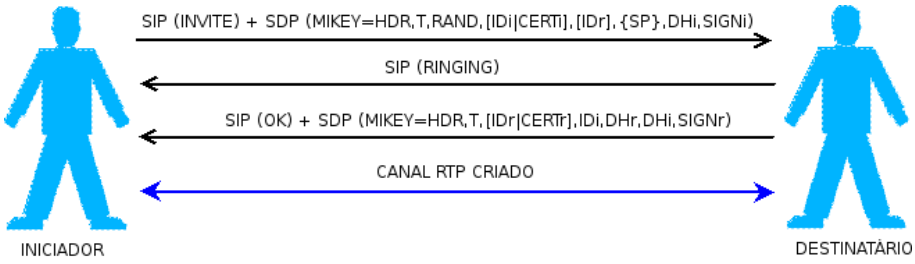

FIGURA 4.18 – Funcionamento MIKEY-DH.

A maior diferença entre o *MIKEY-DH* e os demais modos de transporte é que a mensagem de resposta possui um corpo bem semelhante à requisição inicial, diferenciando-se apenas pela inserção do cabeçalho Hdr (g^{xr}) que, conjuntamente com o *HDi*, gerarão o segredo compartilhado, $TGK = g^{xr*xi}$.

4.3.3.4 Modo de Transporte Reverso de Chaves Públicas (MIKEY-RSA-R)

Conforme foi apresentado em 4.3.3.2, o uso do *MIKEY-PK* é limitado pela necessidade do iniciador possuir o certificado do destinatário, pois sem isso ele não consegue construir o envelope cifrado para transportar a chave de sessão que deseja compartilhar.

Para solucionar esse problema, em novembro de 2006 foi aprovada a especificação do *MIKEY-RSA-R* (IGNJATIC *et al.*, 2006). Esse novo modo de transporte do MIKEY é bastante similar ao MIKEY-PK, diferenciando-se pelo fato de a chave de sessão é inserida na resposta e não na requisição inicial, ou seja, ela é criada pelo destinatário e não pelo iniciador. Essa propriedade é bastante diferente do modo de funcionamento padrão do MIKEY, onde o Iniciador realiza a geração do segredo compartilhado (exceção se faça ao Diffie-Hellman, onde os dois pares contribuem).

Conforme pode-se analisar na figura 4.19, a mensagem de iniciação é bastante simples, contendo apenas os cabeçalhos comuns ao MIKEY e o Certificado do Iniciador. Esse fato faz com que não exista a necessidade da existência de nenhuma informação compartilhada pré-existente entre os pares, pois mesmo o cabeçalho assinado é checado pelo certificado existente na própria mensagem.

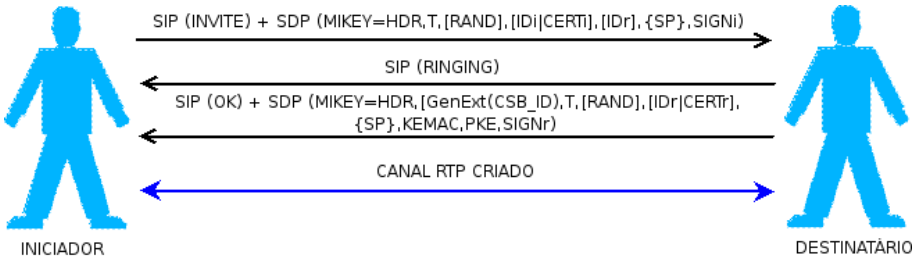

FIGURA 4.19 – Funcionamento MIKEY-RSA-R.

Na mensagem de retorno, aquela confeccionada pelo destinatário, é onde encontra-se o segredo a ser compartilhado entre os pares de forma protegida pelo certificado encontrado na mensagem de inicialização.

Outra limitação dessa arquitetura é sua dependência de uma infra-estrutura de chaves pública (PKI). Esse fato faz com que todos os usuários que desejem ter uma conferência segura necessitem de um certificado e uma chave privada assinados por uma Autoridade Certificadora, o que pode ser oneroso demais em cenários com muitos usuários, como a Internet.

4.3.3.5 Modo de Transporte Baseado em Diffie-Hellman com Autenticação HMAC (MIKEY-DHHMAC)

O uso de Diffie-Hellman em um processo de negociação de chaves é bastante atraente em um cenário fim-a-fim sem qualquer controle do meio, pois ele permite que os pares consigam realizar uma chamada segura sem nenhum requisito de infra-estrutura.

Porém, para garantir a autenticidade de um documento, o modo *MIKEY-DH* necessita fazer uso de certificados, colocando assim a necessidade de um elemento intermediário no processo, a infra-estrutura de chave pública (PKI). Esse novo modo de transporte é apresentado na figura 4.20, onde percebe-se que são usados os mesmos cabeçalhos encontrados em MIKEY-DH.

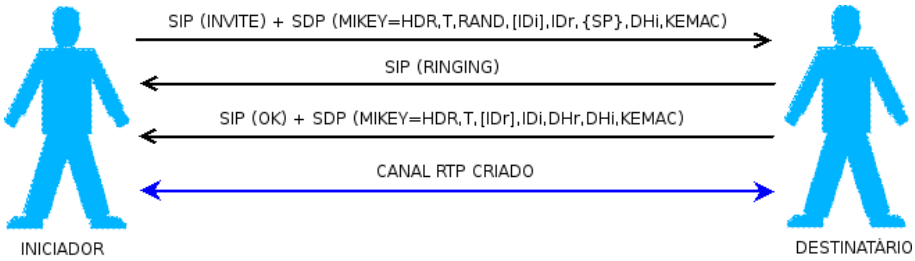

FIGURA 4.20 – Funcionamento MIKEY-DHHMAC.

Para tentar solucionar essa questão, em 2006 foi aprovado um novo modo para o MIKEY, o MIKEY-DHHMAC (EUCHNER, 2006). Esse novo modo de funcionamento consiste no uso do Diffie-Hellman no processo de negociação de parâmetros criptográficos,

porém com a autenticação baseada em um algoritmo simétrico, onde é usado como chave um segredo antecipadamente compartilhado entre as partes.

Em suas mensagens (iniciação e resposta), o cabeçalho *KEMAC* terá um comportamento especial. No caso do MIKEY-PS, esse cabeçalho é usado para transportar o segredo compartilhado. Já no MIKEY-DHHMAC, ele possui a mesma composição da apresentada na sub-seção 4.3.3.3, diferindo-se apenas pelo fato de que, em vez de se transportar a chave simétrica a ser usada para gerar a chave de sessão (*TGK*) de forma segura, é transportado apenas um código de autenticação (MAC) do conteúdo de toda mensagem gerado usando a chave simétrica compartilhada (*pre-shared key*).

Essa arquitetura apesar de resolver o problema do MITM existente no MIKEY-DH, insere uma limitação bastante grande ao protocolo, pois, assim como no *MIKEY-PS*, o uso de chaves compartilhadas gera um problema de escalabilidade muito grande, tornando-o indadequado em cenários como o proposto na presente tese.

4.3.3.6 Derivação de Chaves no MIKEY

Pelo mesmo motivos expostos em 4.2.5, o protocolo MIKEY também possui um esquema próprio de derivação de chaves. Esse esquema visa não só atender a geração de TEK específicas (chaves que serão usadas pelos protocolos multimídia) através de TGK's trafegadas, mas também a geração de chaves para seu uso interno, quando dos processos de cifragem e autenticação de conteúdos dos cabeçalhos KEMAC.

Para realizar essa tarefa, o MIKEY usa uma função pseudo-randômica (*PRF*), que é definida como *PRF(inkey, label)*, onde *inkey* é a chave de entrada do processo usada para realizar o processo de geração de chaves e, label, uma etiqueta que é uma constante

e identifica o tipo de chave a ser gerada.

Segundo (ARKKO *et al.*, 2004) a função *PRF* (expressão 4.14) é baseada na composição, através de operações XOR, de uma segunda função secundária, a função *P*, definida por P(s,label,m), que é calculado pela expressão 4.13.

$$P(s, label, m) = HMAC(s, A_1||label)||HMAC(s, A_2||label)||...||HMAC(s, A_m||label) \tag{4.13}$$

$$PRF(inkey, label) = P(s\_1, label, m)XORP(s\_2, label, m)...XORP(s\_n, label, m) \tag{4.14}$$

Conforme apresentado nas expressões 4.13 e 4.14, para ser possível usar as funções *P* e *PRF*, algumas tarefas adicionais são necessárias. A primeira delas é o particionamento da chave de entrada do processo (*inkey*) em *n* blocos, onde cada bloco corresponde a 160^a parte desta chave ($inkey\_len/160$).A segunda tarefa consiste em calcular *m*, que segundo (ARKKO *et al.*, 2004) é igual a 256^a parte da chave de saída esperada ($outkey\_len/256$).

Outro comentário é sobre o valor de A_x. Essa variável é calculada conforme a seguinte fórmula $A_x = HMAC(s, A_(x-1))$, sendo que quando $x = 0$, então $A_0 = label$.

Para gerar as chaves usadas nos processos internos de cifragem e autenticação, o MIKEY usa como chave de entrada (*inkey*) o valor da chave da *enveloped key* (MIKEY-PK, MIKEY-RSA-R ou MIKEY-DH) ou da *chave simétrica compartilhada* (MIKEY-PS ou MIKEY-DHHMAC).

Já no caso da geração de chaves para uso do protocolo de transporte (*SRTP/SRTCP*), a chave de entrada (*inkey*) é a *TGK* negociada.

Em ambos os casos, existem alguns atributos, que juntamente com uma constante que muda de acordo com o tipo de chave a ser gerada, são usados para gerar a etiqueta (label), que é usada nas funções *P* e *PRF*.

O primeiro desses atributos é o identificador do canal criptográfico (*CSB_ID*) (BAUGHER *et al.*, 2004) e o segundo é o valor *RAND* que é negociado durante a troca das mensagens do protocolo.

É importante mencionar que o *RAND* servirá como um valor de salgamento (*salt*) durante o processo de derivação estipulado pelo algoritmo, uma vez que ele é formado aleatoriamente, o que faz com que não seja possível um atacante estruturar um padrão de formação das chaves derivadas.

4.3.3.7 Integração MIKEY SDP

Conforme foi visto até o presente momento, o MIKEY é um protocolo que possui uma série de cabeçalhos padronizados e provê ao protocolo de sinalização uma forma de realizar a negociação de um contexto criptográfico de forma segura.

Apesar de ser possível inserir o MIKEY diretamente no SIP, é mais apropriado o fazer internamente no SDP. O primeiro motivo para isso está associado à natureza do MIKEY, que é ser um protocolo de negociação de chaves multimídia fim-a-fim. Nesse caso, a inserção de suas mensagens no cabeçalho SIP é um contra-senso, uma vez que esta parte obrigatoriamente sofre uma análise de seu conteúdo em todos os *proxy's* intermediários, aumentando assim o tempo necessário para a comunicação (delay do processo).

O segundo motivo diz respeito à natureza da informação. O contexto criptográfico, assim como os dados de portas e CODEC's a serem utilizados durante a conversação, são informações específicas do canal, logo candidatas a serem trafegadas no corpo do SDP.

Em virtude desses fatos, (ARKKO *et al.*, 2006) propôs uma extensão ao cabeçalho SDP, de forma a possibilitar que ele transportasse informações oriundas do MIKEY.

Diferente do SDES (ANDREASEN; BAUGHER; WING, 2004) que necessita de um canal seguro para ser usado, os atributos SDP propostos por (ARKKO *et al.*, 2006) são independentes de qualquer proteção oferecida pela sinalização, uma vez que são protegidos pelo próprio protocolo que ele transporta.

Um exemplo de seu uso pode ser visto na figura 4.21, onde é apresentada uma mensagem de iniciação MIKEY inserida no corpo do SDP.

```
v=0
o=alice 2891092738 2891092738 IN IP4 w-land.example.com
s=Cool stuff
e=alice@w-land.example.com
t=0 0
c=IN IP4 w-land.example.com
a=key-mgmt:mikey AQAFgM0XflABAAAAAAAAAAAAAAsAyONQ6gAAAAAGEEoo2pee4hp2
UaDX8ZE22YwKAAAPZG9uYWxkQGR1Y2suY29tAQAAAAAAAQAk0JKpgaVkDaawi9whVBtBt
0KZ14ymNuu62+Nv3ozPLygwK/GbAV9iemnGUIZ19fWQUOSrzKTAv9zV
m=audio 49000 RTP/SAVP 98
a=rtpmap:98 AMR/8000
m=video 52230 RTP/SAVP 31
a=rtpmap:31 H261/90000
```

FIGURA 4.21 – Mensagem SDP/MIKEY enviada pelo iniciador. Fonte: (ARKKO *et al.*, 2006)

Apesar de (ARKKO *et al.*, 2006) apresentar em detalhes o uso do SDP em conjunto com o MIKEY, essa especificação é genérica e pode ser usada para qualquer protocolo de negociação de chaves existentes, bastando para isso que seja devidamente padronizada pela IETF.

Outra informação importante é que uma única mensagem SDP pode transportar a oferta de um contexto criptográfico que comporte mais de um protocolo de negociação de

chaves. Neste caso, cabe ao terminal remoto a decisão por qual protocolo usar.

A inserção de mais de um meio para negociar um segredo pode gerar ataques de *downgrading* (3.3.2). Esse tipo de ataque consiste no fato de um agressor inserir um algoritmo ou parâmetro menos seguro (por exemplo, diminuindo o tamanho da chave) de forma a possibilitar que ele tenha o controle do canal.

Para evitar esse problema, a especificação recomenda que existam políticas locais definindo quais protocolos não devem ser usados, maximizando assim o grau de segurança do canal. (ARKKO *et al.*, 2006) também comenta que deve-se garantir que somente mensagens de iniciação possam sugerir protocolos, cabendo às mensagens MIKEY de respostas apenas informar qual protocolo foi aceito para realizar a construção do contexto seguro.

4.3.3.8 Considerações sobre a Segurança do MIKEY

Conforme pode-se verificar na presente seção, o *MIKEY* é uma boa alternativa para prover a negociação de contexto criptográfico para protocolos multimídia. Mesmo assim, ao decidir pelo seu uso, devido as suas limitações, alguns cuidados devem ser observados.

O primeiro deles diz respeito a sua proteção contra *replay atacks*. Essa proteção é semelhante à existente no SRTP/SRTCP, diferenciando-se pela verificação da coerência do *timestamp* antes da checagem da autenticidade da mensagem.

Essa característica dá um nível de segurança maior que o existente no SRTP/SRTCP, porém cria a necessidade de sincronização dos relógios internos dos terminais envolvidos na conferência. Em (ARKKO *et al.*, 2004) é sugerido que essa sincronização seja feita de forma manual ou através de um servidor *NTP* seguro.

Uma das maiores limitações do *MIKEY* diz respeito a necessidade de autenticação das suas mensagens. No caso de cenários fim-a-fim, conforme especificado na presente tese, considera-se minimamente adequado apenas o modelo de transporte *MIKEY-RSA-R*.

Essa arquitetura é considerada minimamente adequada por necessitar do uso de uma infra-estrutura de chaves públicas, o que pode não ser totalmente viável em todos os cenários fim-a-fim, como por exemplo o composto por um número muito grande de usuários.

A problemática do uso de uma infra-estrutura de chaves públicas somente é resolvida apenas pelo modo *MIKEY-DHHMAC*. Porém, ele possui uma limitação bastante grande em seu processo de autenticação, fazendo com que a solução seja pouco escalável.

Apesar das limitações apresentadas, o *MIKEY* pode ser considerado uma excelente alternativa para realizar a negociação de um contexto criptográfico de forma segura, sendo necessário apenas seu uso com bastante critério, pois em alguns modos sua solução não é aplicável a todos os cenários existentes.

4.3.4 O Protocolo ZRTP

Uma alternativa ao MIKEY para estabelecer um contexto criptográfico é o *ZRTP* (ZIMMERMANN; JOHNSTON; CALLAS, 2006). Nesse protocolo, em vez da negociação do contexto ser desenvolvida durante a sinalização, ela é realizada após ser realizada toda a transação iniciada pela requisição INVITE, ou seja, após a abertura do canal RTP. Isso faz com que não sejam necessárias nenhumas modificações nos protocolos de sinalização (SIP e SDP).

A figura 4.22 apresenta a negociação dos parâmetros criptográficos usando o ZRTP. Essa negociação ocorre através de trocas de mensagens texto após o estabelecimento do

canal e antecede o início do transporte do dado multimídia (voz e vídeo) propriamente dito, transporte esse realizado através do SRTP (BAUGHER *et al.*, 2004).

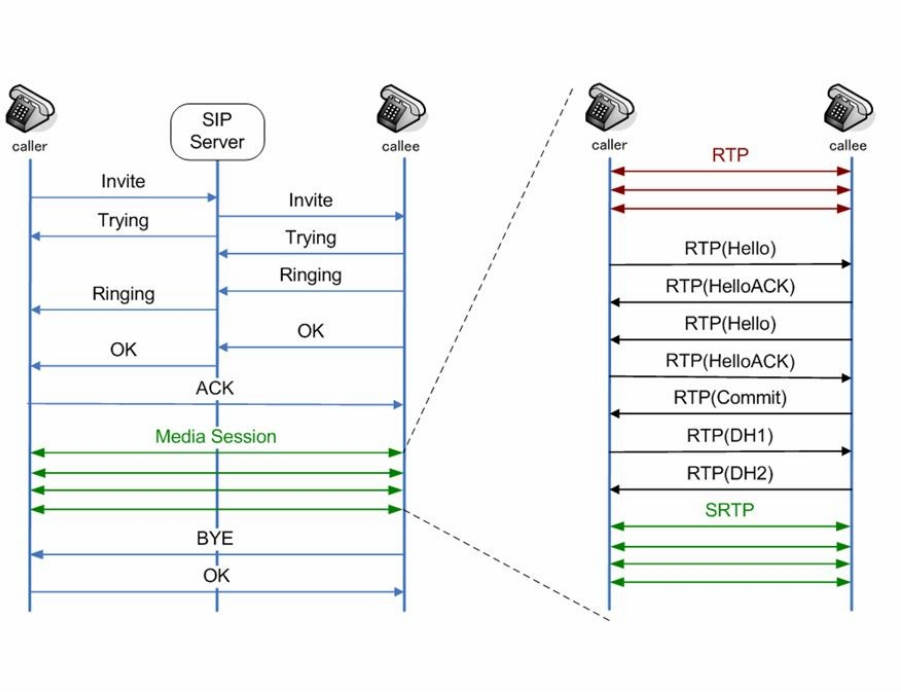

FIGURA 4.22 – Funcionamento do ZRTP. Fonte: (CHEN, 2006)

Para proteger o canal o protocolo utiliza o algoritmo Diffie-Hellman, o que faz o ZRTP precisar de algum mecanismo para garantir a autenticação segura dos pares, evitando assim o problema do MITM.

Para garantir a autenticação correta dos pares o ZRTP usa dois conceitos relativamente simples: o *SAS* (*Short Authentication String*) e a *continuidade da chave*.

A *SAS* consiste em um código de autenticação (MAC) gerado através de um cálculo que usa como base os valores públicos gerados pelo algoritmo Diffie-Hellman (*DHi* e *Dhr*), conforme é apresentado em 4.15.

$$SAS = hash(Dhi||DHr||"ShortStringAuthentication") \tag{4.15}$$

O uso do SAS pelo ZRTP torna necessário que os terminais VoIP possuam uma interface gráfica simples. Essa interface será usada para apresentar o valor do *SAS* aos pares, que deverão confrontar entre si (via voz) esse valor, antes de realizar qualquer conversa com conteúdo sigiloso. No caso de o canal estar sofrendo um ataque de *MITM*, os valores *SAS* apresentados pelos terminais não coincidirão, denunciando assim o ataque de MITM e possibilitando que os mesmos terminem a conversa sem comprometer nenhuma informação sigilosa. Uma demonstração dessa proteção é apresentada na figura 4.23.

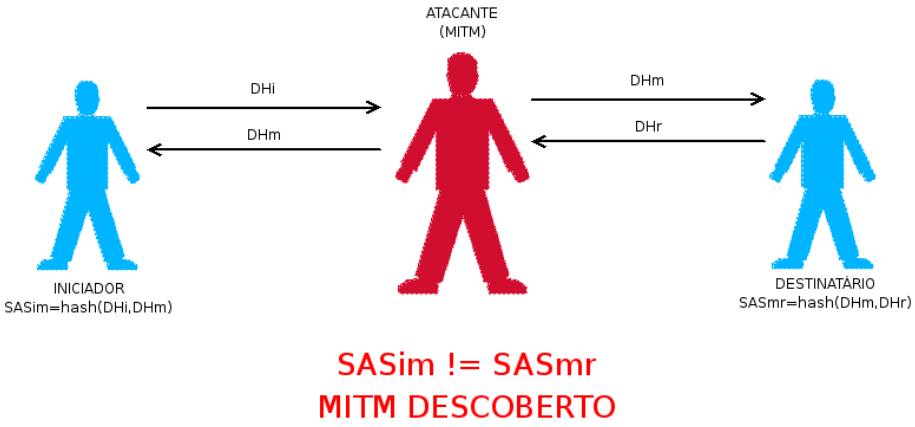

FIGURA 4.23 – Proteção contra MITM através do uso de SAS.

Apesar de o *SAS* garantir uma autenticação segura, ele depende de uma intervenção ativa dos usuários para que seu mecanismo de segurança seja efetivo. Como muitas vezes isso não é uma realidade, o ZRTP propõe uma segunda camada de segurança que é a *continuidade da chave*. Segundo (ZIMMERMANN; JOHNSTON; CALLAS, 2006), entende-se por *continuidade da chave* o fato de a chave de sessão utilizada para proteger a conferência não ser diretamente o valor Diffie-Hellman calculado pelos pares, mas sim um valor derivado de um hash criptográfico formado por esse valor e as chaves acordadas anteriormente pelo ZRTP. A composição da *chave de sessão* usada (*s0*) é apresentada na expressão 4.16, onde *DHSS* é o valor Diffie-Helman ajustado para a sessão e as variáveis *s* (*s1*, *s2* e *s3*) são as chaves de sessão acordadas anteriormente e protegidas localmente

por ambos os pares.

$$s0 = hash(DHSS||s1||s2||s3) \tag{4.16}$$

A camada de proteção oferecida pela propriedade da *continuidade de chave* torna necessário que um eventual atacante tenha que estar presente durante todas as negociações de chaves realizadas pelos pares alvos, uma vez que sua ausência em apenas uma dessas transações anteriores, faz com que ele não tenha condições de gerar o segredo compartilhado pelas partes.

Uma consideração deve ser feita em relação à proteção de *chave continuada*, pois para a efetividade de sua proteção, é necessário que o número de chaves a ser usado para calcular o segredo compartilhado seja relativamente grande, pois senão o trabalho de monitoramento do canal pelo atacante seria simplificado a poucas mensagens.

Outro fator a ser observado, é a necessidade da existência de um mecanismo de validação da chave antes de ela ser armazenada, pois senão chaves comprometidas poderiam fazer parte da composição da chave de sessão, diminuindo assim a robustez oferecida pela propriedade de chave continuada.

Para realizar essa tarefa, o ZRTP usa a própria validação do *SAS*. Segundo (ZIMMERMANN; JOHNSTON; CALLAS, 2006), após a verificação do *SAS* acordado pelos pares, eles devem informar através de uma interface gráfica esse fato ao sistema. Essa informação será transmitida através de um campo especico dentro das mensagens ZRTP de confirmação (*Confirm1* e *Confirm2*), que devem ser transmitidas após o canal seguro (SRTP) ter sido estabelecido.

Conforme apresentado em 4.22, o estabelecimento de um canal seguro usando o ZRTP

é dividido em três fases. A primeira delas é a *discovery*, que é onde os pares irão descobrir suas capacidades, inclusive o suporte ou não do protocolo ZRTP. Essa fase é constituída de uma série de mensagens do tipo *Hello* e *HelloAck*.

Uma característica importante do processo de descoberta é que apesar de ambos os pares enviarem mensagens *Hello*, apenas uma iniciará a escolha dos parâmetros que irão permear a construção do canal, sendo usado para sinalizar essa escolha uma mensagem do tipo *Commit*, que dá a seu emissor o papel de iniciador.

A última fase envolvida no processo de negociação ZRTP é realizada através de mensagens *DHPart*, onde são enviados os valores públicos Diffie-Hellman e as chaves de sessão anteriores acordadas que farão parte do algoritmo de geração da chave de sessão compartilhada (4.16). Para evitar que um atacante nesse momento consiga ter acesso as chaves usadas para a proteção do canal, elas (exceto o valor público Diffie-Hellman) são transmitidas na forma de seus *hash's*, garantindo assim que apenas os usuários legítimos, que possui as chaves armazenadas localmente, possam realizar os cálculos necessários.

Finalmente, depois do canal criado, conferido os valores do SAS e informado essa validação ao sistema são trocadas as mensagens *Confirm* entre os terminais. As mensagens do tipo *Confirm* tem como uma de suas funções garantir ao outro par se o *SAS* já foi validado localmente ou não.

4.3.4.1 Considerações sobre a Segurança do ZRTP

O ZRTP é um protocolo bastante seguro para prover a negociação de um segredo compartilhado. O correto uso dos seus mecanismos de autenticação (*SAS*) garante uma proteção simples e independente de qualquer outro mecanismo externo.

Análises rescentes sobre a segurança do ZRTP demonstram que nos cenários cotidianos de um canal fim-a-fim, o protocolo é seguro. Isso apenas não ocorre em cenários onde o atacantes possui vastos recursos computacionais e em condições bem específicas. Porém nesses casos pequenas adaptações ao protocolo resolvem as vulnerabilidades encontradas (ROBIN; SCHWARTZ, 2006).

Apesar dessas constatações, o fato de sua chave de sessão derivar de um valor armazenado localmente e pré-estabelecido, remove do protocolo a característica de *perfect secrecy forward* existente no Diffie-Hellman. Uma vez que a chave possui em sua constituição chaves anteriormente acordadas, é necessário que algum mecanismo seguro de armazenamento seja utilizado, porém isso não é abordado em (ZIMMERMANN; JOHNSTON; CALLAS, 2006).

O segundo ponto que merece um comentário, apesar de (ZIMMERMANN; JOHNSTON; CALLAS, 2006) propor uma solução, é o fato de todo o funcionamento do *SAS* ser baseado no requisito de existir uma interface gráfica simples nos terminais de usuário, o que é comum, porém existem casos específicos onde isso não é possível.

Pode-se citar como exemplo o cenário constituído por um usuário que necessite estabelecer um canal com uma unidade automática de resposta (*URA*). Nesse caso, a especificação do *ZRTP* apresenta como solução o uso de cabeçalhos SDP para transportar a *SAS*. Nesse caso deve-se utilizar algum mecanismo externo ao ZRTP para prover a proteção do *SAS* (*S/MIME*, *TLS*, *IPSEC*, *MIKEY*), uma vez que ele não apresenta nenhuma segurança interna.

No cenário acima, apesar de ser um problema, não condiz com o apresentado e debatido na presente tese. Mesmo assim, no caso de ser necessário o uso do ZRTP nesse cenário, fica claro que é mais simples usar alguma alternativa anteriormente apresentada.

O maior problema existente do ZRTP não diz respeito a seus mecanismos de segurança, mas sim ao aumento do tempo necessário para ser iniciado o canal de voz. Segundo apresentado na figura 4.22, os usuários, para iniciarem a conversação de forma segura (*SRTP*), necessitam trocar em média um total de sete mensagens.

Apesar de não existirem ainda estudos acerca do comportamento do usuário em relação a atrasos no estabelecimento do canal, acredita-se que os efeitos são bastante similares aos apresentados por (OHTA, 2006).

Essa questão é tão importante, que na própria especificação do protocolo é sugerido à inserção de um ruído de conforto para o usuário, minimizando a sensação de que a chamada foi abruptamente terminada, já que a maioria dos terminais sinalizará graficamente que o canal já está aberto.

Além do fator conforto, o acréscimo de mensagens ao processo de estabelecimento da conferência pode aumentar os problemas de perdas de pacotes em redes congestionadas. Uma vez que o protocolo usa o UDP como transporte, possuindo dessa forma mecanismos próprios de garantia de entrega de suas mensagens. Este fato pode acarretar um efeito cascata, onde com o aumento da perda de pacotes, aumente o número de retransmissões, que acarretam um aumento do congestionamento e dos fatores que ocasionaram a perda inicial de pacotes, piorando dessa forma ainda mais a situação do canal.

Apesar de que em um canal saturado os requisitos mínimos para uma chamada de qualidade dificilmente serão obtidos, o fato de aumentar o tempo necessário para estabelecer um canal é uma limitação bastante grave quando compara-se o ZRTP com outros protocolos similares que conseguem realizar o mesmo trabalho apenas usando a sinalização existente no SIP.

4.4 Análise Comparativa dos Protocolos de Segurança VoIP Fim-a-Fim

Durante o decorrer do capítulo foram apresentadas várias arquiteturas que visavam estabelecer um canal protegido de forma a possibilitar que dois usuários, corretamente identificados, pudessem realizar uma conversa com a garantia de que a mesma não estivesse sendo monitorada.

Visando uma maior independência dos usuários e a possibilidade de que esse canal fosse criado em cenários onde os pares possuíssem uma relação de confiança independente das existentes em seus provedores de acesso, foi acrescentado que a segurança desse canal deveria ser auto-contida nos processos existentes nos pares envolvidos, não necessitando de qualquer funcionalidade existente na infra-estrutura VoIP.

Esse último requisito naturalmente excluía todas as soluções baseadas em túneis seguros (*VPN*) apresentadas na seção 4.1, pois sua implementação independia diretamente das relações de confiança mútua entre os usuários, mas sim, que seus provedores de acesso ao meio possuíssem tal relacionamento.

Quando é necessário garantir a segurança de um canal multimídia baseada em uma estrutura fim-a-fim apenas três alternativas atendem a esse requisito. A primeira delas é o protocolo *S/MIME*, porém, conforme comentado anteriormente, seu uso faz com que todo o cabeçalho SDP tenha que ser protegido. Essa característica impede que os terminais usuários de serviços NAT possam usá-la (4.1.1).

Além disso, o *S/MIME* é limitado ao fato do iniciador precisar possuir antecipadamente o certificado de seu par destino de forma a ser possível a cifragem do conteúdo a

ser protegido. Isso quando ocorre, faz com que sejam necessárias algumas rodadas a mais de sinalização, visando que seja possível a obtenção desse certificado. Isso além de inseguro (vulnerabilidade a ataques de MITM(4.3.1)), torna necessário o uso de mensagens especiais dos protocolos SIP/SDP, fugindo dessa forma a sinalização básica especificada pelo protocolo (INVITE).

A segunda solução para implementar esta arquitetura é através do *MIKEY*. Este protocolo disponibiliza uma diversidade de formas para realizar o transporte seguro do contexto criptográfico, porém todos eles possuem algum tipo de limitação.

No primeiro modo, o *MIKEY-PS*, é usado uma chave compartilhada pré-acordada para prover autenticação e sigilo do contexto criptográfico. Essa solução é bastante eficiente em cenários fim-a-fim com um pequeno número de usuários, pois, uma vez que os segredos são combinados par-a-par, o seu uso em um ambiente com grande número de usuários (n), faria com que fossem necessárias a negociação de uma infinidade de chaves de sessão $(n^2 - n)/2$.

O MIKEY também pode usar uma infra-estrutura de chaves públicas (*PKI*) para realizar esta negociação, possuindo dois modos de funcionamento: o *MIKEY-PK* e o *MIKEY-RSA*. Ambos os modos usam a chave pública de seu par para cifrar a chave negociada e a sua chave privada para assinar as mensagens negociadas pelo protocolo. No primeiro modo, *MIKEY-PK*, existe um problema similar ao encontrado no *S/MIME*, pois para realizar a tarefa de cifragem, o usuário precisa antecipadamente possuir o certificado de seu par. Esse problema é resolvido pelo modo *MIKEY-RSA-R*, que na mensagem de iniciação, ao invés de gerar o segredo simétrico a ser compartilhado pelos pares, envia apenas o seu certificado, fazendo com que essa tarefa seja realizada no terminal que responderá a mensagem inicial.

Apesar de o modo *MIKEY-RSA-R* ser bastante robusto, o uso de uma *PKI* pode ser muito oneroso em determinado cenários, como por exemplo o formado por usuários residenciais. O uso de *PKI* nesses cenários acarretaria que cada usuário existente tivesse que adquirir um certificado oferecido por uma autoridade certificadora. Além disso, a construção de *PKI's* que sejam universalmente aceitas ainda é um desafio a ser vencido, sem comentar a necessidade de políticas de validação para os certificados a serem usados.

Como alternativa ao uso de *PKI*, o *MIKEY* oferece dois modos de transporte baseados no algoritmo Diffie-Hellman (DH): os modos *MIKEY-DH* e *MIKEY-DHHMAC*. Apesar de garantir a confidencialidade do segredo compartilhado de forma independente de uma *PKI*, o modo *MIKEY-DH* possui a limitação de necessitar do uso de certificados para garantir uma proteção contra ataques de MITM. Para contornar esse problema, o *MIKEY-DHHMAC* protege o protocolo usando mensagens autenticadas por um segredo compartilhado anteriormente entre os pares, o que acarreta uma limitação de escalabilidade, similar a encontrada no *MIKEY-PS*.

Finalmente a última alternativa existente para o problema referenciado neste documento é o *ZRTP*, que ainda é uma minuta de especificação em estudo na IETF. Esse protocolo também usa o algoritmo Diffie-Hellman para prover a negociação segura do contexto, porém ao invés de usar certificados e chaves estáticas compartilhadas entre os pares para prover proteção contra ataques de MITM, usa duas novas propriedades: a autenticação baseada em *SAS* e o uso de chave continuada.

Conforme apresentado anteriormente, a primeira consiste em fazer com que os pares confiram os valores públicos Diffe-Hellman acordados (SAS), valores esses que devem ser apresentados aos usuários através de uma interface gráfica. No caso de um ataque de MITM esses valores serão diferentes entre os pares, fazendo com que seja denunciado a

presença de um espião no canal.

A segunda propriedade oferecida pelo ZRTP é chamada de continuidade da chave. Esse conceito consiste no fato da chave usada para o processo de cifragem não ser o valor secreto gerado pelo algoritmo DH, mas sim, a combinação desse valor com outras chaves compartilhadas entre os terminais. Essas chaves compartilhadas podem ser segredos DH negociados anteriormente, bem como chaves acordadas através de outros protocolos, como por exemplo o MIKEY.

Essa característica, apesar de ferir a propriedade de *perfect secrecy forward* (3.3.2) oferecida pelo Diffie-Hellman, aumenta a robustez do protocolo, pois mesmo sofrendo um ataque de MITM, o segredo que o atacante possui não é suficiente para conseguir monitorar o conteúdo do canal.

Apesar do ZRTP ser bastante seguro, o fato de seu funcionamento ser baseado em uma série de mensagens posteriores ao processo oferecido pelos protocolos de sinalização (SIP), deixa ele pouco atraente do ponto de vista de eficiência. Isso porque muitos usuários poderem desistir da chamada pelo atraso ocasionado pelo estabelecimento do segredo. Além disso, quando comparado ao MIKEY, o ZRTP necessita de muito mais rodadas para realizar o mesmo trabalho.

Além disso, o MIKEY consegue ser inserido dentro do funcionamento básico dos protocolos de sinalização existentes. Nesses protocolos ele é inserido em suas mensagens de oferta/aceite através da descrição de um único atributo, o que faz com que os desenvolvedores de soluções VoIP precisem alterar muito pouco os seus produtos para prover a segurança oferecida pelo protocolo.

No próximo capítulo será apresentada uma alternativa às soluções apresentadas até

o momento. Essa solução tem por finalidade resolver as vulnerabilidades e limitações existentes nos protocolos analisados na presente seção, oferecendo a comunidade uma arquitetura fim-a-fim que na qual seja possível implementar um canal seguro.

5 MIKEY-DHHMAC-SAS: Extensão ao MIKEY

No decorrer do presente documento algumas alternativas foram apresentadas com o propósito de construir um canal seguro entre dois usuários.

Apesar de todas essas soluções atenderem a esse propósito, quando se trata de segurança em um cenário fim-a-fim, nem todas as arquiteturas apresentadas solucionam o problema e, aquelas que o resolvem, são limitadas.

Quando comparados, o protocolo MIKEY desponta como o mais eficiente, pois, em todos os seus modos de funcionamento, ele consegue realizar a negociação segura do contexto criptográfico em uma única rodada.

Em um cenário fim-a-fim com muitos usuários, o modo MIKEY que apresenta uma maior flexibilidade e menor custo de implantação é o que usa o protocolo de negociação de chaves Diffie-Hellman (DH). Porém, conforme foi visto no capítulo anterior, as duas formas como o DH está implementado no MIKEY (MIKEY-DH e MIKEY-DHHMAC) apresentam sérias limitações, pois necessitam de algum segredo prévio para conseguir ser robusto à ataques de MITM.

De posse dessa questão, o presente capítulo apresentará uma nova extensão ao proto-

colo MIKEY, o MIKEY-DHHMAC-SAS. Este modo de funcionamento estende o funcionamento padrão do MIKEY-DH, resolvendo o problema da escalabilidade encontrado em (EUCHNER, 2006). Adicionalmente, ele adiciona as propriedades de chave continuada e da autenticação via SAS (ZIMMERMANN; JOHNSTON; CALLAS, 2006), aumentando ainda mais a robustez do protocolo original (ARKKO *et al.*, 2004).

Ao final do capítulo será apresentada uma implementação de referência do protocolo, visando, dessa forma, dar uma noção prática de como o MIKEY-DHHMAC-SAS funcionaria em um cenário real.

5.1 Conceitos sobre MIKEY-DHHMAC-SAS

O MIKEY-DHHMAC-SAS é basicamente o MIKEY-DHHMAC com proteções adicionais encontradas no protocolo ZRTP. Seu objetivo é complementar o MIKEY-DHHMAC, dando a escalabilidade necessária para que ele possa ser usado em ambientes heterogêneos e formado por um número muito grande de usuários.

Seu funcionamento ocorre em uma rodada, inserindo na mensagem de iniciação (*DHHMAC_SAS_I*) o valor público Diffie-Hellman (*DHi*) construído pelo iniciador. Na mensagem de resposta (*DHMAC_SAS_R*), são enviados, além do valor público calculado pelo destinatário (*DHr*), o enviado pelo iniciador (*DHi*), fazendo com que ambas as mensagens possuam um formato bem similar, conforme é apresentado na figura 5.1.

O grande diferencial deste novo modo em relação aos demais previstos no MIKEY é que ele oferece dois níveis de autenticação. O primeiro nível usa o cabeçalho *KEMAC* para prover a autenticação das mensagens de iniciação e de resposta. Assim como no MIKEY-DHHMAC, o conteúdo do *KEMAC* é usado apenas para autenticar o corpo da

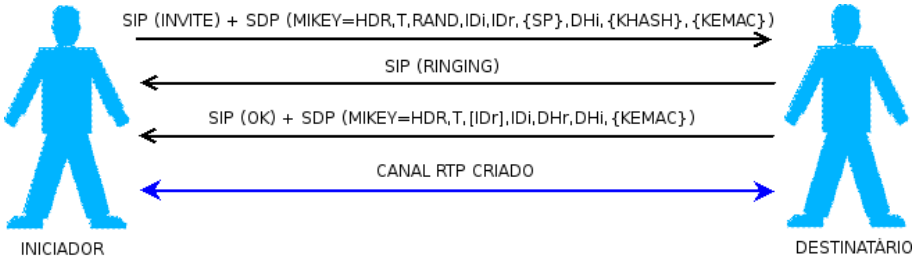

FIGURA 5.1 – Funcionamento do MIKEY no modo DHHMAC-SAS.

mensagem *MIKEY*, não sendo usados os recursos de cifragem existentes no mesmo.

Diferentemente do *MIKEY-DHHMAC*, este novo protocolo não usa chaves estáticas pré-agendadas para realizar a autenticação das mensagens e prover uma proteção contra ataques de *MITM*. Em seu lugar, o *MIKEY-DHHMAC-SAS* usa chaves dinâmicas que são baseadas em segredos anteriormente negociados pelos pares, podendo ser chaves acordadas através de outros protocolos ou por sessões DH antigas.

Para informar quais chaves serão usadas para realizar essa autenticação, o *MIKEY-DHHMAC-SAS* utiliza um novo cabeçalho *MIKEY*, o *KHASH* (*KeyHASH*). Esse novo cabeçalho transporta os últimos 32 bits das assinaturas geradas por três chaves compartilhadas entre os pares, escolhidas aleatoriamente, que serão utilizadas pelo protocolo para realizar a autenticação das mensagens. Esse cabeçalho é gerado de acordo com a fórmula apresentada na expressão 5.1.

$$KHASH = hash(s0)_{32}||hash(s1)_{32}||hash(s2)_{32} \tag{5.1}$$

O terminal remoto, após analisar a mensagem de iniciação, recupera o valor do KHASH e, comparando com as chaves que ele ossui localmente, encontra as chaves usadas pelo iniciador para criar o referido cabeçalho. Dessa forma, os usuários conseguem trocar de uma forma segura as chaves que foram usadas no processo de autenticação das

mensagens MIKEY. A chave *Kh* é definida por 5.2

$$Kh = hash(s0||s1||s2) \tag{5.2}$$

Em 5.2 o algoritmo *hash* usado é o mesmo definido para gerar a expressão KHASH, sendo este um algoritmo de conhecimento público.

Um comentário adicional sobre *KHASH* é que esse cabeçalho apenas existe na mensagem de iniciação. Isso é feito para evitar que um ataque do tipo *MITM*, impedindo que um atacante troque as chaves ofertadas pelo iniciador, substituindo-as por outras que ele tem acesso.

Uma vez que todas as chaves acordadas em sessões passadas são armazenadas localmente por uma quantidade de tempo definida de forma isolada pelo terminal, é possível que o terminal destino não possua alguma das chaves oferecidas pelo iniciador. Nesse caso, o terminal chamado deve responder a mensagem de iniciação (*DHHMAC_SAS_I*) sem inserir o cabeçalho *KEMAC*, que é opcional, demonstrando dessa forma que ele não possui as chaves oferecidas pelo iniciador para autenticar a mensagem MIKEY.

Conforme comentado, o uso da autenticação baseada no *KEMAC* é opcional, pois, além do motivo apresentado anteriormente, esse cabeçalho não pode ser calculada na primeira conferência segura negociada entre os pares. Destaque-se que a garantia contra ataques de *MITM* e a correta identificação dos pares é realizada através de um segundo nível de proteção baseado no *Short Authentication String* (*SAS*).

Assim como apresentado em (ZIMMERMANN; JOHNSTON; CALLAS, 2006), um SAS (ver 5.3) consiste de uma assinatura gerada através de uma função *HMAC* usando como entrada os valores públicos Diffie-Hellmann acordados na sessão, sendo a função

HMAC a mesma especificada no cabeçalho *KEMAC*.

$$SAS = hash(Dhi||DHr||"ShortStringAuthentication") \tag{5.3}$$

Assim como em (ZIMMERMANN; JOHNSTON; CALLAS, 2006), seu uso depende de uma participação ativa do usuário, além de requerer do sistema algum tipo de interface com o usuário pois, antes de iniciar a discussão de um conteúdo sigiloso, eles devem conferir os valores SAS que receberam (JOSEFSSON, 2003). No caso de estarem sofrendo um ataque de MITM, esses valores não coincidirão, fazendo com que o ataque seja anunciado e a conferência interrompida, conforme se pode-se ver na figura 4.23.

Para prover uma maior robustez contra ataques de MITM, ao invés de usar o segredo gerado pelo algoritmo DH ($DHKey$) para derivar as chaves usadas pelo protocolo de transporte multimídia (TGK), os terminais irão gerar um novo segredo derivado de sua combinação com aquele usado para autenticar as mensagens MIKEY (Kh), conforme apresentado na expressão 5.4.

$$TGK = hash(DHKey||Kh) \tag{5.4}$$

A propriedade acima dá ao MIKEY a característica de chave continuada, assim como o ZRTP, possibilitando que, mesmo sob um ataque de MITM, o canal seja preservado.

Diferentemente do ZRTP que possui uma certa preocupação de validar e identificar através do sistema quando uma SAS confere, uma vez que as chaves compartilhadas usadas no processo de cifragem são aleatoriamente escolhidas, o sistema pode armazenar automaticamente o segredo *DHKey*, assim que o canal for inicializado, não necessitando,

para isso, de nenhum tipo de sincronização.

5.2 Funcionamento do MIKEY-DHHMAC-SAS

Conforme apresentado na figura 5.1, o processo de comunicação MIKEY-DHHMAC-SAS é baseado em uma única rodada, onde o iniciador gera um valor público Diffie-Hellman (*DHi*) e o insere em uma mensagem inicial (*DHHMAC_SAS_I*), que deve ser confirmada e retribuída com a segunda parte pública *DHr* em uma mensagem de resposta (*DHHMAC_SAS_R*).

A primeira tarefa que o iniciador deve executar é a geração do seu segredo *DH(x)* e do seu valor público (*DHi*). Este último será transportado pela mensagem de iniciação. A regra de geração do segredo DH é a mesma apresentada na especificação original do protocolo MIKEY.

Após gerar o *DHi*, o usuário precisa definir os diversos parâmetros de uma mensagem de iniciação que se encontram especificados em (ARKKO *et al.*, 2004), com exceção do atributo KEMAC, que obedece as regras baseadas no *MIKEY-DHHMAC* e do *KHASH*, que é específico desse novo modo de transporte e foi apresentada na seção anterior.

De posse de todos os atributos necessários para compor a mensagem DHHMAC, o iniciador constrói a mensagem MIKEY, a insere no corpo de uma mensagem SDP (ARKKO *et al.*, 2006) e a envia ao seu par remoto.

Ao receber a mensagem de inicialização, o terminal destino deverá abrir o cabeçalho, identificar seu par e o modo de transporte em que o MIKEY está funcionando, que no caso será o *MIKEY-DHHMAC-SAS*.

Em seguida, irá localizar as assinaturas de todas as chaves compartilhadas com aquele usuário.

Se a autenticação das mensagens foi realizada pelo MIKEY através do cabeçalho *KEMAC* e, tendo sido possível recuperar as chaves utilizadas através de *KHASH*, será providenciada a conferência da etiqueta de autenticação existente na mesma. Caso ela não confira, deve ser gerada uma mensagem de erro, que será enviada como resposta à mensagem de iniciação.

Se a etiqueta de autenticação conferir, o terminal deverá gerar o segredo local (y) e o valor DH a ser compartilhado, nos mesmos moldes do ocorrido no terminal iniciador.

Em posse desses valores, o terminal poderá confeccionar a mensagem de resposta para enviá-la ao iniciador. Além disso, o terminal local deverá gerar o segredo compartilhado (TGK).

Realizada esta tarefa, o terminal chamado deverá inicializar o seu canal multimídia, enquanto aguarda o recebimento da mensagem de resposta pelo terminal iniciador.

Tendo recebido a mensagem de resposta, o iniciador deverá validá-la, através da verificação de sua autenticidade (caso exista) e, logo após, gerar a TGK que será compartilhada com seu par. Depois disso, ele deverá proceder à inicialização do canal multimídia local, possibilitando, desta forma, a realização segura da conferência.

Tendo inicializado o canal, e antes mesmo de ser possível usá-lo para trafegar informações sensíveis, os terminais devem efetuar a conferência do SAS. Opcionalmente, a aplicação poderá prover uma interface que possibilite ao usuário informar a validação da SAS.

A informação de validação da SAS pode ser usada para definir o período de tempo que

o segredo (*DHKey*) gerado para a sessão deverá permanecer armazenado no repositório seguro de chaves do sistema. No caso dessa informação não ser validada, o sistema não deverá armazenar a chave.

Validado o SAS, os terminais podem iniciar a troca de informações confidenciais entre si, pois todas as proteções oferecidas pelo protocolo estão ativas.

5.3 Analise de Segurança do MIKEY-DHHMAC-SAS

Uma vez que o *MIKEY-DHHMAC-SAS* usa o Diffie-Hellmann como algoritmo de negociação de chaves, uma especial atenção deve ser dada ao gerador de números aleatórios. Esse fato, apesar de não ser comentado em detalhe na especificação original do MIKEY, é de suma importância no atual modelo, pois, além do DH, operações aleatórias são bastante usuais no *MIKEY-DHHMAC-SAS*, como, por exemplo, na escolha das chaves a serem usadas no processo de autenticação *KEMAC*.

Para aumentar a segurança, esses números aleatórios podem ser gerados mediante o uso de algoritmos baseados em fenômenos físicos, tais como aqueles descritos em (ZIMMERMANN; JOHNSTON; CALLAS, 2006) e (MENEZES; OORSCHOT; VANSTONE, 1997).

É importante citar também que muitas operações do *MIKEY-DHHMAC-SAS* são baseadas em algoritmos de *hash*, sejam eles autenticados (*HMAC*) ou não. Apesar de o MIKEY padronizar como algoritmo de hash o *SHA-1* e o *HMAC-SHA-1*, recomenda-se o uso de algoritmos mais robustos como o *SHA-2* ou o *HMAC-SHA-2*, usando chaves de 256 bits pelo menos. Isso se deve à existência de ataques conhecidos sobre a família 1 do SHA.

Apesar de existirem ataques para as funcionalidades herdadas do ZRTP (ROBIN; SCHWARTZ, 2006), eles são bastante teóricos e somente ocorrem em cenários muito específicos e raros dentro da tecnologia VoIP. Tais cenários não serão abordados neste trabalho, fazendo com que se considere a tecnologia segura até o presente momento.

Para atender a necessidade de confirmação existente no ZRTP, o armazenamento das chaves é realizado de forma local e independente, não sendo esta uma exigência obrigatória. Isso é possível pelo fato de a chave usada para realizar o processo de cifragem simétrica ser baseada apenas em alguns segredos escolhidos aleatoriamente entre os valores armazenados localmente. A escolha dos valores a serem usados é anunciada de forma protegida através do uso de mensagens *HMAC*, o que torna necessário o uso de algoritmos de hash robustos.

Apesar da vantagem de não sincronismo apresentada pelo armazenamento local de chaves, existe um problema quando um usuário utiliza mais de um terminal para realizar suas chamadas VoIP seguras. Nesse caso ele possuirá um conjunto de chaves diferentes em cada equipamento utilizado. Para solucionar essa questão é possível o uso de tokens seguros para armazenar as chaves negociadas. Nesse caso o usuário, mesmo usando diversos equipamentos, possuirá um único repositório de chaves.

Porém essa solução estrapola o escopo de uma especificação de um protocolo, sendo mais uma questão de projeto. Conforme apresentado em 5.1, o terminal que recebe uma requisição MIKEY, quando não possui uma das chaves selecionadas pelo iniciador para realizar a proteção do conteúdo da mensagem, deve excluir o cabeçalho *KHASH* em sua resposta, o que fará com que seja obrigatória a validação do *SAS* pelos pares para garantir que um ataque de MITM não esteja ocorrendo.

5.4 Implementação de Referência

Para possibilitar a análise do comportamento real do protocolo proposto, foi desenvolvido um cliente VoIP que implementa o MIKEY-DHHMAC-SAS para a negociação do contexto criptográfico.

O presente cliente não tem por finalidade mostrar a eficiência do protocolo desenvolvido nesta pesquisa, mas sim o de comprovar na prática os conceitos abordados em sua construção.

Para permitir a verificação de que o cliente está criptografando o canal de voz, os protocolos SRTP e RTP foram implementados de modo a possibilitar o funcionamento do cliente tanto no modo protegido (SRTP) como no modo desprotegido (RTP).

Uma vez que o objetivo do modelo não é o de ser um cliente VoIP comercial e, considerando que os testes seriam realizados em ambiente controlado para fins acadêmicos, o cliente não possui os métodos SIP REGISTER, que são destinados para a autenticação dos clientes em uma rede SIP.

Outro método que não foi implementado pelos mesmos motivos acima foram aqueles os relacionados aos protocolos RTCP e SRTCP, pois, em ambientes controlados como o que ocorreu os experimentos, não existem anomalias que possam comprometer a qualidade da voz.

Para desenvolver o cliente foi utilizada a linguagem Java (JAVA, 2007), pelos motivos apresentados abaixo:

- Conhecimento do autor sobre a linguagem;
- Diversidade de *frameworks* opensource existentes e disponíveis para uso;

- Possibilidade de migração do código para ambientes celulares com mínimas modificações; e

- Diversidade de literatura dando suporte ao desenvolvimento.

5.4.1 Questões de Projeto

Uma vez escolhida a linguagem Java, foram selecionados os *frameworks* existentes que poderiam simplificar e agilizar o desenvolvimento do cliente.

Para implementar a criptografia, foi utilizada a Application Programming Interface (API) *Java Encription Extensions (JCE)* implementada pela Bouncy Castle (BOUNCY-CASTLE, 2007). Apesar de ser possível o uso da implementação de referência da Sun Microsystem (JAVA, 2007), elas não possibilitam o uso de chaves maiores que 128 bits para o AES, além de não possuir uma implementação de referência para ser usada em dispositivos móveis, fazendo com que o reuso do código gerado em aplicações práticas ficasse bastante prejudicado.

Apesar de a JCE ser bastante complexa, a funcionalidade que mais apresentou dificuldade de ser desenvolvida foi a que codificava e decodificava os dados multimídia (voz). Inicialmente foi tentado utilizar o framework JMF (Java Media Framework) (JMF, 2007), que é uma API de alto nível e que oferece ao desenvolvedor uma abstração da complexidade envolvida no processo de codificação multimídia. Todavia, a complexidade para extender suas classes fizeram com que a JMF fosse deixada de lado.

Em seu lugar foi usado outra API desenvolvida pela SUN, o JavaSound (JAVA, 2007). Diferentemente do JMF, o JavaSound implementa uma API de baixo nível, deixando os desenvolvedores manipular inclusive o processo de codificação da mídia. Devido a

esse fato, ele pode utilizar os vetores de bits gerados após a codificação, simplificando a manipulação e construção dos pacotes RTP e SRTP. Essa simplicidade teve um custo, pois foi necessário reconfigurar o ambiente de digitalização do som, o que não era necessário no JMF.

A última API utilizada na construção do cliente VoIP foi o Jain SIP (JAIN, 2007). O Jain SIP é um dos produtos da família de especificações Java para desenvolvimento de redes multimídia futuras (*Next Generation Communications*). Esse framework permite a construção de qualquer entidade SIP existente nas especificações da IETF e possui uma implementação de referência para ambientes desktop, servidores e móvel.

5.4.2 Modelo de Classe

O modelo de classe desenvolvido para implementar as funcionalidades desejadas na tese é apresentado na figura 5.2.

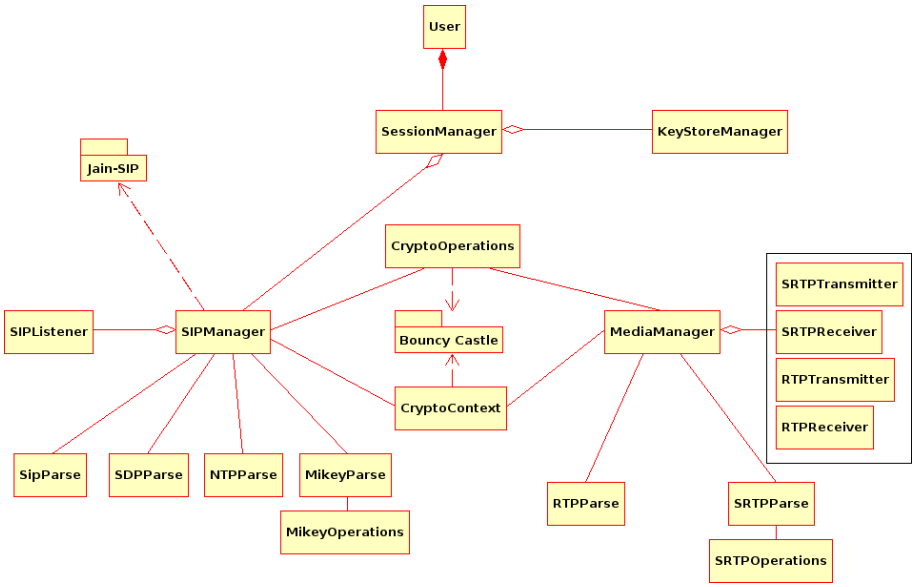

FIGURA 5.2 – Funcionamento do MIKEY no modo DHHMAC-SAS.

A classe mais importante desse modelo é a **User**, pois é a responsável por inicializar todas as demais classes do sistema, bem como tratar o armazenamento e recuperação das

informações necessários para que o usuário interaja na rede VoIP.

Outra classe de importância vital é **SessionManager**. Essa classe inicializa e controla o funcionamento das classes responsáveis pela gerência das sessões de sinalização (**SIPManager**) e multimídia (**MediaManager**). Outra tarefa de **SessionManager** é gerenciar o uso da **KeyStoreManager**, que é a responsável por armazenar e recuperar as chaves de criptografia utilizadas pelo usuário.

Conforme dito, o gerenciamento da sessão é realizado pela classe **SIPManager**. Para seu funcionamento, **SIPManager** precisa inicializar um listener (SIPListener), bem como algumas classes que realizarão a análise (*parse*) das mensagens SIP confeccionadas ou recebidas pelo sistema.

O **SIPListener** tem por responsabilidade criar um processo que fica aguardando indefinidamente mensagens SIP endereçadas ao usuário. Também é ele que, após o devido tratamento realizado por **SIPManager**, envia as mensagens confeccionadas localmente para os diversos pares remotos.

Os *parses* gerenciados por **SIPManager** são: **SIPParse**, **SDPParse**, **NTPParse** e **MikeyParse**. Todos esses parses tem por finalidade interpretar as mensagens recebidas e colocá-las em um formato que permita o **SIPManager** realizar as suas operações.

Um parse especial é **MikeyParse**. É ele que realiza a construção do cabeçalho MIKEY inserido no corpo do SDP, bem como sua interpretação. Uma vez que existem uma série operação específicas a ser realizada na construção e interpretação desse tipo de mensagem, **MikeyParse** necessita das funcionalidades de uma classe auxiliar, **MIKEYOperations**, que implementa as operações necessárias para o funcionamento dos processos definidos pelo protocolo MIKEY.

Para gerenciar a sessão multimídia é usado **MediaManager**. Diferentemente de SIP-Manager, ela possui listeners diferentes para transmissão e recepção, bem como específicos para os protocolos RTP (**RTPTransmitter** e **RTPReceiver**) e SRTP (**SRTPTransmitter** e **SRTPReceiver**). Além disso, ela possui classes especificas de parse para os dois protocolos (**SRTPParse** e **RTPParse**).

A realização das operações criptográficas existentes no protocolo MIKEY e SRTP é realizada através das classes **CryptoOperations** e **CryptoContext**. A primeira trata das operações criptográficas, ou seja, àquelas relacionadas ao processo de cifragem, geração de hashs, entre outras. Já **CryptoContext** é usado para armazenar e recuperar o contexto criptográfico, ou seja, às informações do usuário necessárias para a construção de um canal seguro usando o SRTP.

5.4.3 Funcionamento do Cliente

Para usar o cliente, basta possuir o *Java Runtime Environment* (*JRE*) (JRE, 2007) instalado em sua máquina na versão 5.0 ou superior.

Teoricamente o sistema pode ser usado em qualquer sistema operacional que possua a máquina virtual do Java. Ressalte-se, todavia, que a ferramenta apenas foi testada nos sistemas Microsoft Windows XP (MICROSOFT, 2007) e Ubuntu (UBUNTU, 2007).

Para utilizar o cliente basta, através de um *shell* (Windows ou Linux), digitar o seguinte comando: *$ java -jar camaleao.jar*. Após isso o sistema será iniciado e será apresentado a tela 5.3.

Sendo apresentada a tela inicial do sistema para usuário, ele deverá selecionar a criação de uma nova conta. Essa tarefa fará com que seja criada a estrutura de diretório usado pelo

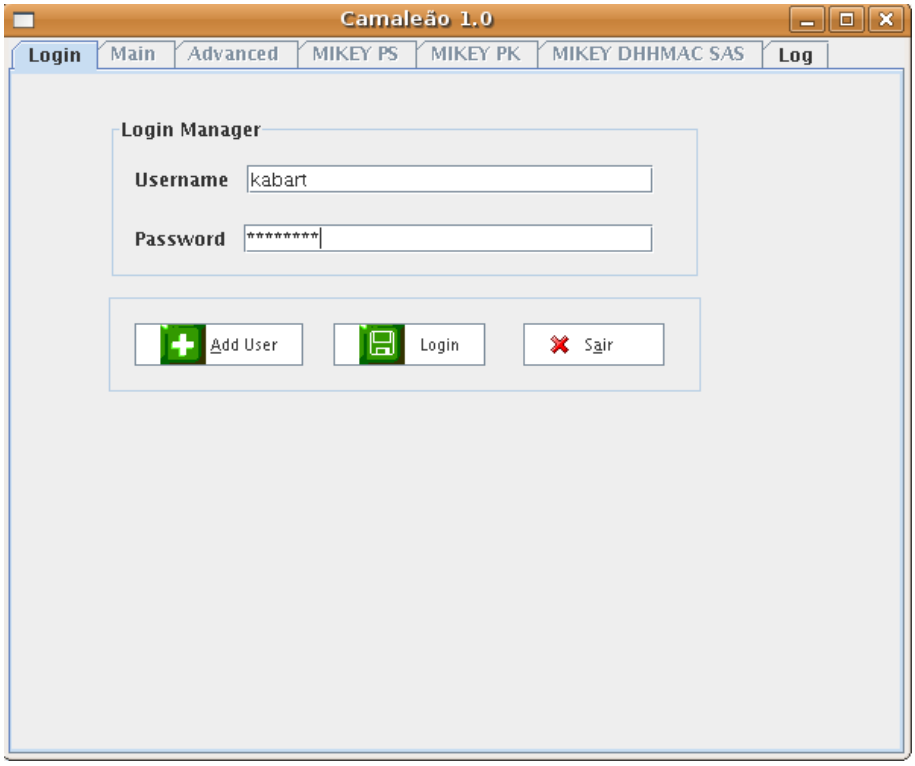

FIGURA 5.3 – Tela Inicial do Cliente VoIP.

usuário, bem como um repositório seguro, a (*keystore*), que será utilizado para armazenar as chaves de sessão acordadas e que serão utilizadas pelo MIKEY DHHMAC-SAS. Esse repositório é protegido pela senha inserida no campo *password* e é implementado usando o padrão do Java para keystore: o **JCEKS**.

Caso o usuário já possua uma conta no sistema, ele deve pressionar o botão *login*, que checará no sistema se o usuário existe e se a senha inserida confere com a criada no sistema.

Sendo bem sucedido o processo de autenticação, o botão *login* mudará de cor e de rótulo, informando que o usuário foi corretamente autenticado no sistema. Após isso, considerando que é a primeira vez que o usuário interage com o sistema, ele deverá pressionar a guia rotulada como *Advanced* (ver figura 5.4), pois sem essas configurações avançadas, não é possível realizar uma chamada.

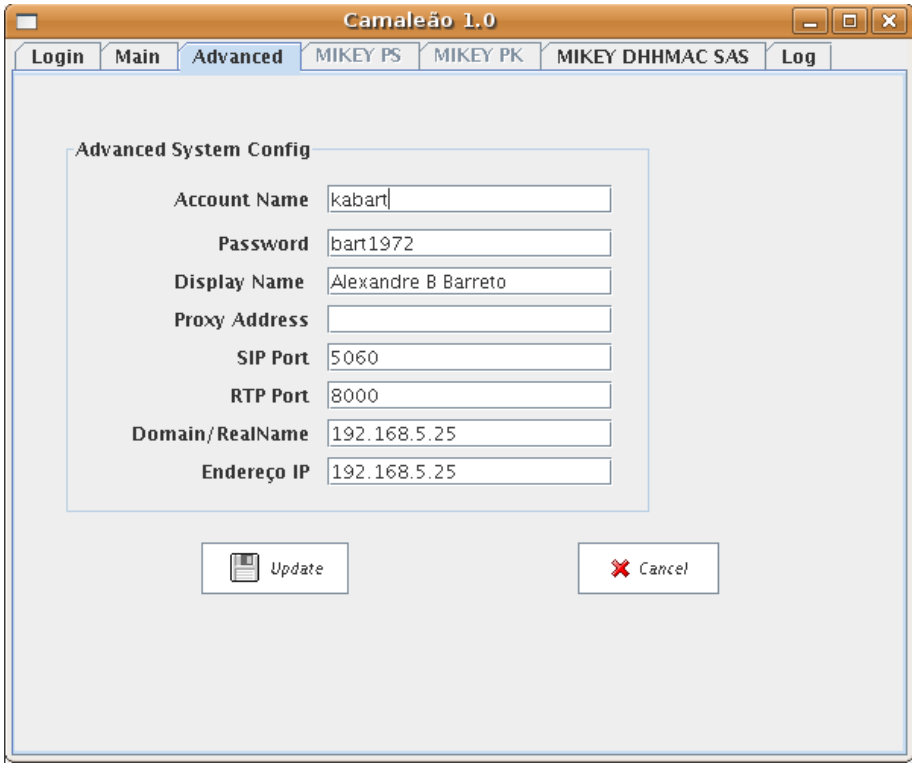

FIGURA 5.4 – Tela de Configurações Avançadas.

Uma vez pressionada a guia supra-citada, será aberto para o usuário uma tela de configuração básica do sistema, local onde ele deverá informar quais são os seus parâmetros para construção do canal multimídia, bem como o funcionamento padrão do cliente SIP local.

Após concluída a configuração avançada do ambiente, o usuário deverá presssionar a guia principal (*Main*), para ser possível realizar uma chamada VoIP. Conforme apresentado na figura 5.5, essa tela é dividida em seis áreas.

A primeira área (*Call Peer*) é aquela onde o usuário informa seu parceiro remoto que ele deseja contatar. Deve-se comentar que no campo *Peer Address* o usuário deverá digitar a URL do usuário desejado.

Na segunda área (Proxy Status) é apresentado o status do sistema. Essa área informa se foi bem sucedido o processo de autenticação do usuário em seu *proxy*. Como no pro-

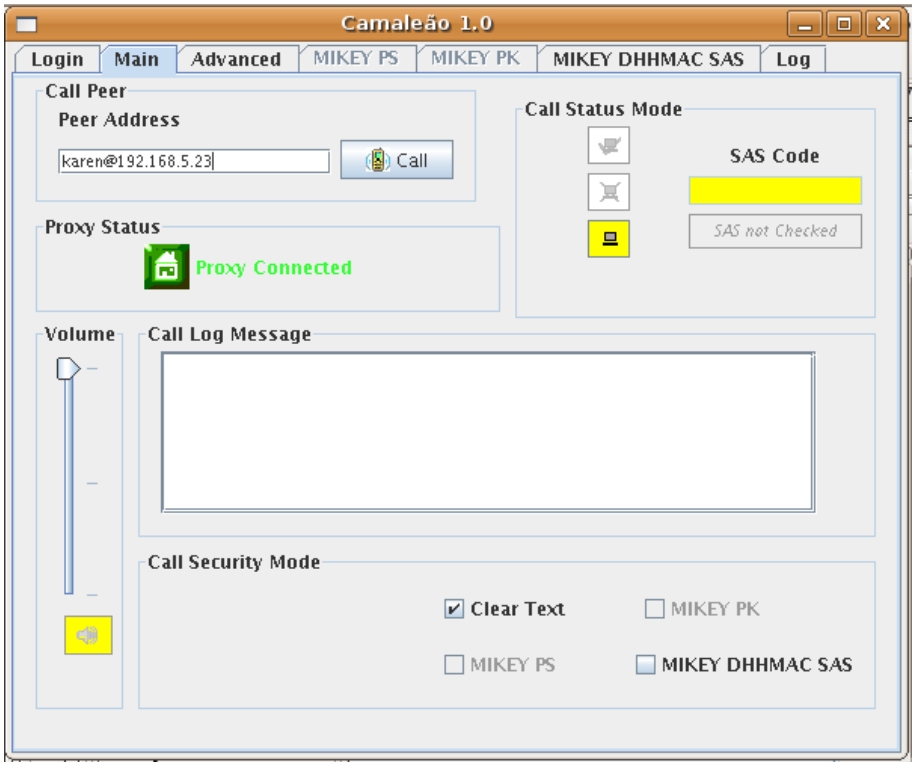

FIGURA 5.5 – Tela Principal do Sistema.

tótipo não foram desenvolvidos os métodos de autenticação, essa área sempre aparecerá *Proxy Connected*.

A área de volume permite que o usuário aumente ou diminua o som do áudio recebido e transmitido de seu telefone, porém não foi implementada.

A área de Log (*Call Log Message*) serve para informar ao usuário as diversas mensagens recebidas e enviadas pelo seu terminal, o que também não foi implementado.

Uma das áreas mais importantes do sistema é a *Call Status Mode*. Nesse local o usuário é informado sobre o estado da chamada, no que diz respeito a sua segurança. Quando o sistema não conseguir detectar se o ambiente está protegido ou não, os botões amarelos estarão ativos. Esse é o caso quando o sistema é inicializado. Caso o sistema consiga detectar que a chamada está comprometida, o botão vermelho é ativado. Por fim, caso a chamada esteja segura, é habilitado o botão verde.

Além dessa informação, é nessa guia que o usuário verifica o SAS gerado, bem como informa ao sistema se essa informação foi validada.

A última guia existente é a *Call Security Mode*, que informa ao sistema em que modo a chamada irá ser enviada ao outro par.

Para ser possível realizar uma chamada segura, o usuário deverá preencher o endereço do par remoto na guia *Peer Address*, pressionar o botão *MIKEY DHHMAC SAS* na guia *Call Security Mode* e pressionar o botão *Call*.

Uma vez realizada essa tarefa, as mensagens entre os pares serão negociadas e apresentadas na guia *Call Log Message* e, ao abrir o canal seguro, a guia *Call Status Mode* é modificada de acordo com o status da segurança da transação.

Realizada a abertura da conferência, o botão *Call* mudará de cor e deverá ser pressionado quando um dos usuários for finalizar a chamada.

6 Conclusão

O fato de não existir uma solução universalmente aceita para prover segurança a uma conferência baseada na arquitetura de telefonia IP é um fator limitante para seu crescimento e sua aceitação em ambientes onde requisitos de segurança sejam um fator preponderante.

Essa constatação foi o maior motivador para o desenvolvimento da presente pesquisa pois, enquanto essa pendência não for resolvida, dificilmente as indústrias incorporarão essa funcionalidade em larga escala em seus produtos.

Apesar disso, alguns segmentos da indústria adotaram arquiteturas baseadas em túneis para prover segurança a uma chamada VoIP, pois, além de ser uma tecnologia muito conhecida, a maior parte das empresas já as usam cotidianamente para prover segurança a sua rede tradicional de dados.

Porém, conforme pode-se ver na presente tese, o desenvolvimento de um padrão fim-a-fim, ou seja, onde os pares não dependem de nenhum recurso da infra-estrutura VoIP existente para compartilharem um segredo, é bastante desejável. Isso porque nem todos os cenários podem ser implementados por uma tecnologia de túnel, além do que essas tecnologias podem ter um custo alto para ser usada em ambientes muito vastos e heterogêneos.

Dentre as arquiteturas estudadas neste documento, pode-se perceber que todas elas apresentavam algum problema e limitações de uso. Entretanto, o MIKEY demonstrou possuir a melhor estrutura para prover esse serviço, pois ele é auto-contido nos protocolos de sinalização pré-existentes (SIP), não necessitando de aumento de banda para seu funcionamento.

O padrão apresentado neste documento, o *MIKEY DHHMAC-SAS*, é uma evolução natural do MIKEY, resolvendo problemas encontrados em sua especificação, além de inserir novas funcionalidades encontradas em protocolos que ainda estão em estudo na IETF.

Esse novo padrão tem como maior contribuição o fato de eliminar a necessidade de uso de uma infra-estrutura de chave pública, sem afetar a escalabilidade da arquitetura. Isso é muito importante em cenários onde o custo de usar uma infra-estrutura de chaves públicas é proibitivo, como por exemplo o uso em usuários residenciais da Internet.

A adoção em larga escala desse novo modo de transporte, possibilita que sejam alcançados esses e outros cenários onde os protocolos atuais são extremamente limitados, tornando possível a adoção de um padrão universal de negociação de chaves para telefonia IP, assim como já ocorre no seu transporte (SRTP/SRTCP).

6.1 Contribuições

Dentre as contribuições mais relevantes da presente tese cita-se:

- Desenvolvimento de um protótipo funcional que implementa o MIKEY-DHHMAC-SAS, tornando possível a percepção de seu funcionamento real; e

- Proposta de um novo padrão aberto (IETF) para a negociação de chaves em chamadas VoIP.

6.2 Trabalhos Futuros

A partir dos conhecimentos obtidos pretende-se dar continuidade à pesquisa através dos seguintes itens, dentre outros:

- Analisar a eficiência do modo de transporte proposto, verificando-a em relação aos demais modos existentes no MIKEY, bem como o ZRTP;
- Verificar a funcionalidade do modo MIKEY DHHMAC-SAS, usando algum método formal de análise a segurança;
- Submeter ao IETF esse modo de transporte, visando que o mesmo torne-se um padrão aberto a ser adotado na Internet;
- Adaptação do protocolo para uso em redes celulares e redes convencionais comutadas de telefonia;
- Analisar o desempenho do algoritmo em ambientes móveis (VoIP Mobile), verificando o impacto das operações de geração de números aleatórios em dispositivos de baixo poder computacional; e
- Dar continuidade ao protótipo desenvolvido, implementando as operações não implementadas, adaptando-o para ser usado em dispositivos celulares.

Referências Bibliográficas

ANDREASEN, F.; BAUGHER, M.; WING, D. **RFC 4568:Session Description Protocol (SDP) Security Description for Media Streams.** [S.l.]: Internet Engineering Task Force, 2004.

ANDREOLI, A. V. **IP Security**. 2000. Disponível em: <http://www.cert-rs.tche.br/docs_html/ipsec.html>. Acesso em: 03 mar. 2005.

ANWAR, Z.; LATHIA, M.; PAI, N.; TUCKER, M. Voip security: Media security overview. 2006. Disponível em: <http://www.cs.uiuc.edu/class/sp06/cs598cag/slides-/563.13.4%20Media%20Security%20Overview.ppt>. Acesso em: 16 ago. 2006.

ARKKO, J.; CARRARA, E.; LINDHOLM, F.; NORRMAN, M. N. K. **RFC 3830: MIKEY: Multimedia Internet KEYing.** [S.l.]: Internet Engineering Task Force, 2004.

ARKKO, J.; LINDHOLM, F.; NASLUND, M.; NORRMAN, K.; CARRARA, E. **RFC 4567: Key Management Extensions for Session Description Protocol (SDP) and Real Time Streaming Protocol (RTSP).** [S.l.]: Internet Engineering Task Force, 2006.

AVAYA. **What is the difference between IP Telephony and VoIP?** 2006. Disponível em: <http://www.avaya.com/gcm/master-usa/en-us/tasks/learn/facts-/iptelephony/qa1/iptelephony.htm>. Acesso em: 10 jul. 2006.

BAUGHER, M.; MCGREW, D.; NASLUND, M.; CARRARA, E.; NORRMAN, K. **RFC 3711: The Secure Real-time Transport Protocol (SRTP).** [S.l.]: Internet Engineering Task Force, 2004.

BILLIEN, J. **Key Agreement for Secure Voice Over IP.** Dissertação (Mestrado) — Kungl Tekniska *Hö*gskolan., Stockolm - Swedeen., 2003.

BOUNCYCASTLE. **Bouncy Castle - Java OpenSource API Cryptography.** 2007. Disponível em: <http://www.bouncycastle.org/>. Acesso em: 07 mar. 2007.

BUCHMANN, J. **Introdução a Criptografia.** [S.l.]: Berckeley, 2002.

CARUSO, C.; STEFFEN, F. **Segurança em Informática e de Informações.** [S.l.]: Editora Senac, 1999.

CHEN, E. **A Tour Through Zfone**. 2006. Disponível em: <http://voipsa.org/blog/2006/06/19/a-tour-through-zfone>. Acesso em: 07 jun. 2006.

COHEN, D. **RFC 741: Specifications for the Network Voice Protocol (NVP).** [S.l.]: Internet Engineering Task Force, 1977.

DIERKS, T. **RFC 4346: The TLS Protocol.** [S.l.]: Internet Engineering Task Force, 2006.

EUCHNER, M. **RFC 4650: HMAC-Authenticated Diffie-Hellman for Multimedia Internet KEYing (MIKEY).** [S.l.]: Internet Engineering Task Force, 2006.

HANDLEY, M.; JACOBSON, V.; PERKINS, C. **RFC 4566: Session Description Protocol (SDP).** [S.l.]: Internet Engineering Task Force, 2006.

IGNJATIC, D.; DONDETI, L.; AUDET, F.; LIN, P. **RFC 4738: MIKEY-RSA-R: An Additional Mode of Key Distribution in Multimedia Internet KEYing (MIKEY).** [S.l.]: Internet Engineering Task Force, 2006.

ISO/IEC. Iso/iec 13335-1: Management of information and communications technology security. In: . [S.l.]: International Organization for Standardization, 2004.

ISO/IEC. Iso/iec 17799: Código de prática para gestão da segurança de informações. In: . [S.l.]: International Organization for Standardization, 2005.

ITU-T. One-way transmission time: Recomendation g.114. In: . [S.l.]: International Telecommunication Union, 1996.

JAIN. **Jain SIP.** 2007. Disponível em: <http://java.sun.com/products/jain/news.html>. Acesso em: 07 mar. 2007.

JAVA. **Java Technology.** 2007. Disponível em: <http://www.java.sun.com>. Acesso em: 07 mar. 2007.

JENNINGS, C. **Example Call Flows Using SIP Security Mechanisms.** [S.l.]: Internet Engineering Task Force, 2004.

JMF. **Java Media Framework.** 2007. Disponível em: <http://java.sun.com/products/java-media/jmf/index.jsp>. Acesso em: 07 mar. 2007.

JOSEFSSON, S. **RFC 3548: The Base16, Base32 and Base64 Data Encodings.** [S.l.]: Internet Engineering Task Force, 2003.

JRE. **Java Runtime Environment.** 2007. Disponível em: <http://java.sun.com/javase/downloads/index.jsp>. Acesso em: 07 mar. 2007.

KENT, S.; SEO, K. **RFC 4301: Security Architecture for the Internet Protocol.** [S.l.]: Internet Engineering Task Force, 2005.

KUHN, D. R.; WALSH, T. J.; FRIES, S. **Security Considerations for Voice Over IP Systems**. [S.l.], 2005.

LEWIS, M. C. E. **Configuring Cisco Voice over Ip.** [S.l.]: Syngress; 1 edition, 2000.

LIPMAA, H.; ROGAWAY, P.; WAGNER, D. **CTR-Mode Encryption.** [S.l.]: National Institute of Standards and Technology (NIST), 2006.

MENEZES, A.; OORSCHOT, P. C. van; VANSTONE, S. A. **Handbook of Applied Cryptography.** [S.l.]: CRC Press, 1997.

MICROSOFT. **Windows XP.** 2007. Disponível em: <http://www.microsoft.com>. Acesso em: 07 mar. 2007.

MOREIRA, A. **Comutação/ e Transferência de Dados**. 2007. Disponível em: <http://www.dei.isep.ipp.pt/~andre/documentos/comutacao.html>. Acesso em: 03 mar. 2007.

NIST. **NIST Brief Comments on Recent Cryptanalytic Attacks on Secure Hashing Functions andthe Continued Security Provided by SHA-1H**. [S.l.], 2004. Disponível em: <http://csrc.nist.gov/hash_standards_comments.pdf>. Acesso em: 27 set. 2006.

OHTA, M. Overload control in a sip signalling network. **Transactions on Engineering, Computing and Technology**, v. 12, 2006.

OLSSON, A. **Entendendo Telecomunicações 2**. [S.l.]: Editora Erica, 2000.

PERKINS, C. **RTP: Audio and Video for the Internet.** [S.l.]: Addison Wesley, 2003.

RAMSDELL, B. **RFC 2633: S/MIME Version 3 Message Specification.** [S.l.]: Internet Engineering Task Force, 1999.

RANSOME, J. F.; RITTINGHOUSE, J. W. **VoIP Security**. [S.l.]: Elvesier Digital Press, 2005.

ROBIN, J.; SCHWARTZ, A. **Analysis of ZRTP**. [S.l.], 2006.

ROSENBERG, J.; SCHULZRINNE, H.; CAMARILLO, G.; JOHNSTON, A.; J.PETERSON; SPARKS, R.; HANDLEY, M.; SCHOOLER, E. **RFC 3261: Session Initiation Protocol (SIP).** [S.l.]: Internet Engineering Task Force, 2002.

SCHULZRINNE, H.; CASNER, S.; FREDERICK, R.; JACOBSON, V. **RFC 3550: RTP - A Transport Protocol for Real-Time Applications.** [S.l.]: Internet Engineering Task Force, 2003.

SOARES, L. F. G.; LEMOS, G.; COLCHER, S. **Redes de Computadores: Das LANs, MANs e WANs às Redes ATM.** [S.l.]: Editora Campus, 1995.

STINSON, D. R. **Cryptography: Theory and Practice, 2nd Edition.** [S.l.]: Chapman Hall/CRC, 2002.

STREDICKE, C. Securing voip media communication. Berlim, 2006.

TANENBAUM, A. **Redes de Computadores.** [S.l.]: Editora Campus, 2003.

TELEGEOGRAPHY. **Brazil and Nigeria Fastest-Growing VoIP Destinations.** 2005. Disponível em: <http://www.telegeography.com/press/releases/2005-12-15.php>. Acesso em: 07 mar. 2005.

TRAPPE, W.; WASHINGTON, L. **Introduction to Cryptography with Coding Theory.** [S.l.]: Prentice Hall, 2001.

UBUNTU. **Ubuntu: Linux for Human beings.** 2007. Disponível em: <http://www.ubuntu.com>. Acesso em: 07 mar. 2007.

WIKIPEDIA. **Diffie-Hellman key exchange.** 2007. Disponível em: <http://en.wikipedia.org/wiki/Diffie-Hellman_key_exchange>. Acesso em: 28 abr. 2007.

WIKIPEDIA. **Hash function.** 2007. Disponível em: <http://en.wikipedia.org/wiki/Hash_function>. Acesso em: 28 abr. 2007.

WIKIPEDIA. **Real-time Transport Protocol.** 2007. Disponível em: <http://en.wikipedia.org/wiki/Real-time_Transport_Protocol>. Acesso em: 28 abr. 2007.

ZIMMERMANN, P.; JOHNSTON, A.; CALLAS, J. **ZRTP: Extensions to RTP for Diffie-Hellman Key Agreement for SRTP.** [S.l.]: Internet Engineering Task Force, 2006.

FOLHA DE REGISTRO DO DOCUMENTO

1. CLASSIFICAÇÃO/TIPO	2. DATA	3. DOCUMENTO Nº	4. Nº DE PÁGINAS
TM	03 de setembro de 2007	CTA/ITA-IEC/TM-014/2007	155

5. TÍTULO E SUBTÍTULO:

Uma Arquitetura de Telefonia IP para Proteção da Mídia Fim-a-Fim

6. AUTOR(ES):

Alexandre de Barros Barreto

7. INSTITUIÇÃO(ÕES)/ÓRGÃO(S) INTERNO(S)/DIVISÃO(ÕES):

Instituto Tecnológico de Aeronáutica. Divisão de Ciência da Computação– ITA/IEC

8. PALAVRAS-CHAVE SUGERIDAS PELO AUTOR:

Telefonia IP; Negociação de Chaves; Criptografia

9. PALAVRAS-CHAVE RESULTANTES DE INDEXAÇÃO:

Telefonia; Criptografia de chave pública; Segurança; Protocolos de comunicação; Comunicação por voz; Telecomunicações; Comunicações; Engenharia eletrônica

10. APRESENTAÇÃO: (**X**) **Nacional** ( ) **Internacional**

ITA, São José dos Campos, 2007, 155 páginas

11. RESUMO:

O uso da tecnologia de voz sobre IP (VoIP) vem se tornando cada vez mais comum no mundo todo. No entanto, a implantação dessa tecnologia associada à necessidade de garantia de sigilo do canal compatível a sua precursora, a telefonia comutada, ainda é um desafio. Apesar de haver certo consenso em como prover o transporte seguro do dado multimídia transportado em uma conferência, ainda existe discussões de como efetuar a negociação de forma protegida das chaves que permitirão a criação do canal criptografado. Quando se deseja criar esse canal de forma independente de qualquer infra-estrutura de segurança existente, tais como nos cenários fim-a-fim, onde apenas os terminais envolvidos devem ser os responsáveis por essa proteção, as alternativas se resumem a apenas três: o uso dos protocolos S/MIME, MIKEY ou ZRTP. Apesar desses frameworks resolverem satisfatoriamente o problema, eles possuem limitações que os impedem de atender todos os cenários de implementação existentes. A presente dissertação propõe um novo modo de transporte para o MIKEY, o MIKEY-DHHMAC-SAS, que mantém as características principais do protocolo, mas adiciona outras existentes no ZRTP, solucionando assim as limitações dos protocolos existentes, podendo ser usado em qualquer conferência segura fim-a-fim.

12. GRAU DE SIGILO:

(**X**) **OSTENSIVO** ( ) **RESERVADO** ( ) **CONFIDENCIAL** ( ) **SECRETO**