FERNANDO TEUBL FERREIRA

**TÉCNICAS DE GERENCIAMENTO DE CHAVES COMPARTILHADAS
EM GRUPOS MULTICAST**

São Paulo

2007

FERNANDO TEUBL FERREIRA

**TÉCNICAS DE GERENCIAMENTO DE CHAVES COMPARTILHADAS
EM GRUPOS MULTICAST**

**Dissertação apresentada à Escola
Politécnica da Universidade de São
Paulo para a obtenção do título de
Mestre em Engenharia Elétrica**

**São Paulo
2007**

FERNANDO TEUBL FERREIRA

**TÉCNICAS DE GERENCIAMENTO DE CHAVES COMPARTILHADAS
EM GRUPOS MULTICAST**

**Dissertação apresentada à Escola
Politécnica da Universidade de São
Paulo para a obtenção do título de
Mestre em Engenharia Elétrica**

**Área de Concentração:
Engenharia Elétrica
Sistemas Eletrônicos**

**Orientador:
Prof.
Sergio Takeo Kofuji**

**São Paulo
2007**

Este exemplar foi revisado e alterado em relação à versão original, sob responsabilidade única do autor e com a anuência de seu orientador.

São Paulo, de fevereiro de 2007.

Assinatura do autor _____

Assinatura do orientador _____

FICHA CATALOGRÁFICA

Ferreira, Fernando Teubl

Técnicas de gerenciamento de chaves compartilhadas em grupos *Multicast* / F.T. Ferreira. -- ed.rev. -- São Paulo, 2007. 160 p.

Dissertação (Mestrado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Sistemas Eletrônicos.

1.Segurança em grupos de usuários utilizando redes *Multicast* I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Sistemas Eletrônicos II.t.

A meus pais, familiares, amigos e especialmente a Carolina de Melo Gagliato.

AGRADECIMENTOS

Agradeço a meu orientador, Sergio Takeo Kofuji, por toda dedicação, paciência e por todos os ensinamentos, além de toda a equipe do PAD — Grupo de Sistemas Pervasivos e de Alto Desempenho — tanto em virtude da ajuda direta quanto do auxílio indireto para a conclusão desta pesquisa.

Gostaria de agradecer em especial a Bruno Barberi Gnecco que não apenas me apoiou no início desta jornada, mas também colaborou durante toda a elaboração desta dissertação.

Também deixo os meus agradecimentos para Hilton Garcia Fernandes por toda a compreensão, ajuda e apoio. Foi fundamental para a conclusão deste trabalho, ainda, o auxílio da equipe do Núcleo de Tecnologia Sem Fio, bem assim da Caverna Digital, ambas do Laboratório de Sistemas Integráveis da Escola Politécnica da Universidade de São Paulo que, de uma forma ou de outra, colaboraram para esta pesquisa.

Agradeço, enfim, a todos os demais amigos que embora não tenham ajudado diretamente nesta pesquisa, foram essenciais para minha formação acadêmica e profissional.

RESUMO

Com a popularização da rede global, a Internet, as aplicações colaborativas ganharam destaque, sendo imprescindíveis nas mais diversas atividades pessoais e comerciais. Os avanços tecnológicos modernos trouxeram novas demandas de aplicações, com a inclusão de diversas funcionalidades em ambientes cooperativos como, por exemplo, a distribuição de dados multimídia sobre redes de comunicação. Entretanto, quando estas ferramentas são aplicadas em ambientes coletivos com muitos usuários, o uso das mesmas é deteriorado pelas limitações da rede. Protocolos *Multicast* possibilitam o uso destas aplicações colaborativas em razão de proporcionarem a redução do uso da rede para atividades coletivas, possibilitando a interação com dezenas, centenas ou milhares de usuários simultaneamente. Na medida em que as ferramentas colaborativas ganham espaço entre os usuários, surge também a necessidade do emprego de segurança entre grupos de usuários. Os grupos devem ser capazes de estabelecer comunicações *Multicast* seguras em que apenas os membros autorizados sejam hábeis a acessar os conteúdos veiculados pelo grupo. São exemplos de aplicativos que exigem *Multicast* seguro: videoconferências confidenciais, sincronismo de tabelas financeiras entre matriz e filiais, distribuição de vídeo e áudio para um grupo de assinantes, dentre inúmeras outras utilizações. A proteção do conteúdo do grupo é alcançada por meio de criptografia e as chaves devem ser atualizadas quando do ingresso de novo usuário ou na hipótese de desistência de algum membro do grupo. As técnicas de gerenciamento de chaves devem ser eficientes, tanto no aspecto de segurança, quanto no que pertine ao desempenho. Devem, ainda, possibilitar a sua utilização em grupos com quantidades massivas de usuários. Os objetivos do presente trabalho são, em suma, estudar os esquemas de gerenciamento de chaves de grupo, propor uma nova técnica cujo foco seja a minimização do uso dos recursos de rede em ambientes limitados, simular os modelos avaliados pelo simulador de rede NS-2 e analisar os impactos destes esquemas em aplicações colaborativas. Para tanto, desenvolveu-se um módulo para o NS-2 que permitiu ao simulador prover suporte ao gerenciamento de chaves, e, ainda, construíram-se dois aplicativos auxiliares para a geração de cenários e análise de resultados em simulações NS-2.

Palavras-chave: *Multicast*. Segurança. Gerenciamento de Chaves de Grupo.

ABSTRACT

With the popularization of the Internet, collaborative applications have been gaining marketshare, becoming intrinsic to a wide range of personal and commercial activities. Modern technological improvement brought new demands for these applications, such as the inclusion of new features in cooperative environments, such as the distribution of multimedia data over communication networks. However, when these tools are applied to collective environments with many users, their usability is hindered by the network limits. Multicast protocols allow the use of these collaborative applications, since they reduce the necessary network bandwidth, allowing the interaction with tens, hundreds or thousands of users simultaneously. As collaborative tools become more popular among users, there comes also the problem of security in user groups. The groups must be able to establish secure multicast communications in which only the authorized members can access the content distributed to the group. Examples of applications which demand secure multicast are confidential videoconferences, synchronization of financial tables between the headquarter and filials, distribution of video and audio to a group of users, among many others. The protection of group contents is achieved by cryptography and the keys must be updated whenever a new users joins or leaves the group. The techniques for key management must be efficient, in terms of security and performance, and also be passible of use in groups with massive amounts of users. The goals of this work are: to study the group key management systems, to propose a new technique whose focus is to minimize the use of network resources in limited environments, to simulate the models evaluated by the network simulator NS-2 and to analyze the impacts of these systems in collaborative applications. For that, a module for NS-2 has been developed that allows the simulator to provide support to key management and, moreover, two auxiliar applications were written to generate the scenarions and analyse the simulations returned by NS-2.

Keywords: *Multicast*. Security. Group Key Management.

Lista de Figuras

2.1	Transmissão <i>Unicast</i> vs. <i>Multicast</i>	9
2.2	Exemplo de cenário <i>Multicast</i>	10
2.3	Relação entre o número de usuários e a banda utilizada para o envio de dados em redes <i>Unicast</i> e <i>Multicast</i>	11
2.4	Formato da mensagem IGMP	15
2.5	<i>Hamming Code</i>	19
2.6	Exemplo de redundância para recuperação de dados	20
2.7	Diagrama Caso de Uso	23
2.8	Exemplo de chave compartilhada em transmissão <i>Multicast</i>	24
2.9	Exemplo de confidencialidade temporal	25
2.10	Esboço de estrutura de pacote para distribuição de chaves	26
2.11	Requisitos de análise para esquemas de gerenciamento de chaves	28
2.12	Controle de grupo	30
2.13	Controle de sub-grupo	31
2.14	Controle por membros	32
2.15	Esquema escalável vs. não-escalável	33
2.16	Agrupamento periódico de eventos	35
2.17	Agrupamento por lote de evento	36
2.18	Agrupamento misto	37
3.1	Transmissão <i>Multicast</i> um-para-muitos (1-n)	39
3.2	Transmissão <i>Multicast</i> muitos-para-muitos (n-n)	40
3.3	Exemplo de cenário <i>Multicast</i> com autenticação	43

3.4	Exemplo de autenticação progressiva	47
4.1	Estrutura de árvore de chave (CTKM)	51
4.2	Estrutura de árvore de chave após o abandono do Membro 9 (CTKM)	52
4.3	Exemplo de Árvore de Distribuição lolus	61
4.4	Mensagem de União (lolus)	62
4.5	Mensagem de Abandono (lolus)	63
4.6	Fluxo de envio do lolus	64
4.7	Exemplo de Árvore de Distribuição Segura (DEP)	66
4.8	Comunicação do membro com o SGM	68
5.1	Triângulo de Pascal	74
6.1	Aplicativo NAM: visualizador gráfico de simulações NS-2	90
6.2	Aplicativo Xgraph	91
6.3	Fluxo de simulação	94
6.4	Geração de uma rede <i>mesh</i>	95
6.5	Exemplo de uma topologia Mista com 60 nós	96
6.6	Estrutura do código NS-2 após a inclusão do MSec	99
6.7	Fluxograma do analisador de eventos	102
7.1	Simulação 1: Distribuição de usuários e taxa de <i>join</i> e <i>leave</i>	106
7.2	Simulação 1: Informações enviadas pelo servidor de dados	107
7.3	Simulação 1: Utilização geral da rede	108
7.4	Simulação 2: <i>Overhead</i> do servidor ocasionado pelo gerenciamento de chaves	110
7.5	Simulação 2: <i>Overhead</i> geral ocasionado pelo gerenciamento de chaves	112
7.6	Simulação 3: Distribuição de usuários e taxa de <i>join</i> e <i>leave</i>	114
7.7	Simulação 3: Fluxo de dados de gerenciamento recebido pelo Membro 1	115
7.8	Simulação 3: Utilização da rede pelo gerenciador de chaves	116
7.9	Simulação 3: Acumulado do uso médio de rede para gerenciamento de chaves do sistema	116
7.10	Simulação 4: Distribuição de usuários e taxa de <i>join</i> e <i>leave</i>	118

7.11	Simulação 4: lolus vs. DEP, com 4, 6 e 8 subgrupos	119
7.12	Simulação 5: Comparação entre os esquemas lolus, DEP, CTKM, EBS e EBCK	120
A.1	Formato MAC	126
A.2	Mapeamento <i>Multicast</i> no MAC	126
B.1	Problemas em roteamento <i>Multicast</i>	129
B.2	Roteamento <i>Multicast</i>	131
B.3	<i>Multicast Tunneling</i>	132
B.4	<i>Core Route</i>	134
B.5	Reestruturação dos RP	135

Lista de Tabelas

1.1	Exemplos de serviços <i>Multicast</i> e suas aplicabilidades em ambientes seguros	3
2.1	Tipos de mensagens IGMP	16
2.2	Esquemas de gerenciamento de chaves	34
4.1	Conjuntos de chaves administrativas do EBS(10,3,2)	58
4.2	Anotações e nomenclaturas para descrever o protocolo DEP	65
4.3	Componentes de um certificado DEP	66
4.4	Tabela comparativa (Iolus, DEP, CTKM, EBS e EBCK)	71
5.1	Número de chaves e quantidade total de usuários	75
5.2	Exemplo de distribuição de chaves EBCK	78
5.3	Exemplo de distribuição de chaves após a inclusão de novas chaves K_7 e K_8	86
7.1	Descrição da Simulação 1	105
7.2	Descrição da Simulação 2 (4 e 5)	110
7.3	Descrição da Simulação 3	113

Lista de Abreviaturas e Siglas

ACL	<i>Access Control List</i>	62
ASM	<i>Any Source Multicast</i>	38
CBR	<i>Constant Bit Rate</i>	97
CBT	<i>Core Based Trees</i>	133
CD	<i>Compact Disc</i>	19
CKMSS	<i>Centralized Key Management with Secret Sharing</i>	50
CRC	<i>Cyclic Redundancy Check</i>	16, 20
CSCW	<i>Computer Supported Cooperative Work</i>	1
CTKM	<i>Centralized Tree-Based Key Management</i>	34, 50
DCT	<i>Discrete Cosine Transform</i>	48
DEP	<i>Dual Encryption Protocol</i>	64
DoS	<i>Denial Of Service</i>	5
DR	<i>Designated Router</i>	19
DVMRP	<i>Distance Vector Multicast Routing Protocol</i>	134
EBCK	<i>Exclusion-Based Combinatorial Key</i>	5, 72
GPRS	<i>General Packet Radio Service</i>	3
GSA	<i>Group Security Association</i>	42
IANA	<i>Internet Assigned Names Authority</i>	125

ICMP	<i>Internet Control Message Protocol</i>	14, 15
IGMP	<i>Internet Group Management Protocol</i>	15
KDC	<i>Key Distribution Centre</i>	26
LAN	<i>Local Area Network</i>	125
MAC	<i>Message Authentication Codes</i>	21, 47
MAC	<i>Media Access Control</i>	125
MLD	<i>Multicast Listener Discovery</i>	15
MSC	<i>Minimum Set Cover</i>	83
MTBF	<i>Mean Time Between Failures</i>	64
MTU	<i>Maximum Transmission Unit</i>	92
NIC	<i>Network Interface Card</i>	125
PDA	<i>Personal Digital Assistant</i>	3
PIM	<i>Protocol-Independent Multicast</i>	133
QoS	<i>Quality Of Service</i>	29
RAID	<i>Redundant Array of Independent Disks</i>	19
RIP	<i>Routing Information Protocol</i>	133
RPF	<i>Reverse path forwarding</i>	130
SSM	<i>Source-Specific Multicast</i>	38
TCL	<i>Tool Command Language</i>	88
TESLA	<i>Timed Efficient Stream Loss-Tolerant Authentication</i>	47
TTL	<i>Time to Live</i>	14, 130

Sumário

RESUMO	i
ABSTRACT	ii
LISTA DE FIGURAS	iii
LISTA DE TABELAS	vi
LISTA DE ABREVIATURAS E SIGLAS	vii
 Capítulo 1: Introdução	 1
1.1 Objetivos	5
1.2 Contribuições obtidas	7
 Capítulo 2: Revisão da Literatura	 8
2.1 <i>Multicast</i>	8
2.1.1 Definição	8
2.1.2 Propriedades	8
2.1.3 Vantagens	9
2.1.3.1 Redução de banda	10
2.1.3.2 Gerenciamento do grupo	11
2.1.3.3 Desempenho	13
2.1.4 Especificação	13
2.1.5 Gerenciamento de grupos <i>Multicast</i>	14

2.1.5.1	IGMP	15
2.1.5.2	MLD	16
2.1.6	Propriedades desejáveis em protocolos <i>Multicast</i>	17
2.1.7	<i>Multicast</i> Confiável	18
2.2	Segurança em <i>Multicast</i>	21
2.2.1	Requisitos de segurança	22
2.2.2	Privacidade em grupos <i>Multicast</i>	23
2.2.3	Confidencialidade Temporal	24
2.2.4	Renovação de chaves	25
2.3	Esquemas de gerenciamento de chaves compartilhadas	27
2.3.1	Requisitos de análise	27
2.3.2	Controle de grupo, sub-grupo e por membro	30
2.3.2.1	Controle de grupo	30
2.3.2.2	Controle de sub-grupo	30
2.3.2.3	Controle por membro	31
2.3.3	Classificação	32
2.3.4	Estudo sobre esquemas de gerenciamento de chaves na literatura	34
2.3.5	Agrupamento de eventos	35
2.3.5.1	Periódico (<i>periodic</i>)	35
2.3.5.2	Lote (<i>batch</i>)	36
2.3.5.3	Misto	37

Capítulo 3: Cenários, aplicações e políticas de segurança 38

3.1	Cenários <i>Multicast</i> seguros	38
3.1.1	Comunicação um-para-muitos (1-n)	39
3.1.2	Comunicação muitos-para-muitos (n-n)	40
3.1.3	Diferenças entre cenários um-para-muitos e muitos-para-muitos	41
3.2	Autenticação em redes <i>Multicast</i>	42
3.2.1	Autenticação de usuário	42

3.2.1.1	Autenticação Centralizada	42
3.2.1.2	Autenticação Distribuída	43
3.2.2	Autenticação de dados	44
3.2.2.1	Autenticação de grupo	44
3.2.2.2	Autenticação de origem	45
3.2.3	Autenticação Progressiva	46
3.3	Política de Segurança e Desempenho	47
3.3.1	Algoritmo de Criptografia	48
3.3.2	Formas de Criptografia	48
3.3.3	Autenticação	48
3.3.4	Gerenciamento de Chaves	49

Capítulo 4: Esquemas de gerenciamento de chaves 50

4.1	CTKM	50
4.1.1	Operações de gerenciamento	51
4.1.2	Processo de abandono (<i>leave</i>)	52
4.1.3	Processo de União (<i>join</i>)	53
4.1.4	Renovação de chave (<i>rekey</i>)	55
4.2	EBS	55
4.2.1	Definição	55
4.2.2	Processo de abandono (<i>leave</i>)	57
4.2.3	Construção do sistema EBS	58
4.2.4	Processo de união (<i>join</i>)	58
4.3	lolus	59
4.3.1	Características	59
4.3.2	Funcionamento	60
4.3.3	Visão Operacional	62
4.3.3.1	Inicialização	62
4.3.3.2	Processo de união (<i>join</i>)	62

4.3.3.3	Saída de usuário (<i>leave</i>)	63
4.3.4	Transmissão de dados	63
4.4	DEP	64
4.4.1	Estrutura	66
4.4.2	Inicialização do sistema	67
4.4.3	Entrada de usuários	67
4.4.4	Comunicação Segura	69
4.4.5	Saída de usuário	69
4.4.6	Atualização de chaves	70
4.5	Resumo comparativo	71

Capítulo 5: EBCK: Uma proposta para gerenciamento de chaves 72

5.1	EBCK – <i>Exclusion-Based Combinatorial Key</i>	72
5.2	Definição	73
5.2.1	Relação entre chaves de exclusão e usuários	75
5.3	Escalabilidade	75
5.4	Manutenção do grupo	77
5.4.1	Autenticação	77
5.4.2	Processo de união	78
5.4.3	Múltiplas uniões	80
5.4.4	Rechaveamento	80
5.4.5	Abandono	81
5.4.6	Múltiplos abandonos	81
5.4.7	Crescimento dinâmico de chaves	85
5.4.8	Privilégios de usuários	86

Capítulo 6: Ferramentas e métodos de Simulação 88

6.1	Simulador de rede NS-2	88
-----	----------------------------------	----

6.1.1	Arquivo de configuração TCL	89
6.1.2	Visualização da simulação	89
6.1.3	Informações geradas	90
6.1.4	Geração de todos os eventos	92
6.2	Ferramentas desenvolvidas	93
6.2.1	Geração de cenários TCL	94
6.2.1.1	Tipos de topologia	95
6.2.1.2	Entrada e Saída de membros	96
6.2.1.3	Tipo de aplicação	97
6.2.1.4	Fluxos de dados na rede	98
6.2.1.5	Tipos de transmissão em um grupo <i>Multicast</i>	98
6.2.2	Agente <i>Multicast</i> seguro	99
6.2.3	Analisador de eventos	101

Capítulo 7: Simulações 104

7.1	Simulação 1: Protocolos escaláveis vs. Protocolos não-escaláveis em cenários 1–n	104
7.1.1	Descrição	104
7.1.2	Resultados e Análises	106
7.2	Simulação 2: Protocolos escaláveis por grupo em cenários 1–n	109
7.2.1	Descrição	109
7.2.2	Resultados e Análises	109
7.3	Simulação 3: Protocolos escaláveis por grupo em cenários n–n	112
7.3.1	Descrição	113
7.3.2	Resultados e Análises	114
7.4	Simulação 4: Protocolos escaláveis com sub-grupos	117
7.4.1	Descrição	117
7.4.2	Resultados e Análises	117
7.5	Simulação 5: Comparação entre esquemas de gerenciamento	118
7.5.1	Descrição	120

7.5.2	Resultados e Análises	120
Capítulo 8:	Conclusão	122
8.1	Trabalhos futuros	124
Apêndice A:	Multicast em TCP/IP	125
A.1	<i>Multicast</i> na Camada 2	125
A.2	<i>Multicast</i> na Camada 3 (IP)	127
Apêndice B:	Roteamento Multicast	128
B.1	Roteamento de pacotes <i>Multicast</i>	128
B.1.1	Formas de roteamento	128
B.1.2	Dificuldades de roteamento	129
B.1.2.1	RPF	130
B.1.3	Árvore <i>Multicast</i> (<i>Multicast Tree</i>)	130
B.1.4	Túnel <i>Multicast</i> (<i>Multicast Tunneling</i>)	131
B.2	Protocolos <i>Multicast</i>	132
B.2.1	DVMRP	132
B.2.2	CBT	133
B.2.3	PIM	133
B.2.3.1	PIM-DM (<i>Dense Mode</i>)	134
B.2.3.2	PIM-SM (<i>Sparse Mode</i>)	134
ÍNDICE REMISSIVO		143

1 INTRODUÇÃO

Sistemas Cooperativos ou Sistemas Colaborativos são sistemas que fornecem suporte computacional a um grupo de entidades que trabalha conjuntamente para um mesmo fim, sem que esteja necessariamente em uma mesma localidade ou intervalo de tempo. Trabalho Cooperativo Suportado por Computador (CSCW) constitui ferramenta designada para auxiliar a implantação e o desenvolvimento de trabalhos cooperativos.

Software colaborativo, conhecido como *groupware*, consubstancia ferramenta que auxilia o trabalho coletivo. Define-se *groupware* como: sistema baseado em computador que auxilia grupos de pessoas envolvidas em tarefas comuns (ou objetivos) e que provê interface para um ambiente compartilhado (Ellis; Wainer, 1994).

O processo de colaboração necessita de interoperabilidade e, desta forma, a comunicação entre entidades colaboradoras é a base para qualquer sistema cooperativo. Os *groupware* usualmente utilizam a rede Internet para interconectar todas as entidades colaboradoras.

A Internet possibilitou a interconexão entre redes, permitindo aos usuários comunicarem-se livremente sem restrição de localidade, possibilitando o desenvolvimento, implementação e utilização de inúmeras ferramentas colaborativas. Não apenas pessoas beneficiam-se desta grandiosa rede, usufruindo das ferramentas cooperativas (*groupware*), mas também empresas e grandes corporações encontram soluções para maximizar sua produtividade e lucro. São exemplos de ferramentas cooperativas: Mensagem Instantânea, Email, Wiki, Videoconferência, Bate-Papo, Fórum, entre muitos outros serviços encontrados na Internet.

Como exemplo, a empresa de tênis Nike®¹ beneficia-se da Internet, ferramenta que aumenta substancialmente suas vendas. Com efeito, através de sua página na Internet, os consumidores são capazes de personalizar os tênis vendidos pela empresa, podendo selecionar diversos tipos de modelo, cor, cadarço, acabamento, e até adicionar figuras personalizadas, sendo, ainda, viável visualizar o modelo gerado no *browser*. Após a confirmação da compra do modelo, a Nike® — que não possui fábricas próprias — sincroniza todas as indústrias terceirizadas para incluir em sua linha de produção o modelo especificado, processo que atinge desde a seleção da matéria prima até a confecção e entrega do modelo. Em um prazo médio de três semanas, o consumidor recebe em sua casa o modelo adquirido. Esta prática, além de agradar os consumidores, permite a minimização dos estoques das fábricas e, conseqüentemente, minimiza o capital parado de todas as empresas envolvidas no processo de fabricação.

¹ Nike® – Empresa Americana produtora de Tênis esportivo. Disponível em: www.nike.com

Além da troca de dados entre pessoas e empresas, as aplicações multimídia também vêm crescendo de forma acentuada através dos anos, possibilitando que um grupo de usuários em diversas localidades do mundo realize uma conferência de vídeo e áudio com apenas terminais de acesso convencionais, ou seja, um computador pessoal conectado à rede global, Internet.

Aplicativos colaborativos como Skype®² oferecem aos seus usuários recursos multimídia interativos de alta qualidade. O software Skype® pode ser instalado em sistemas operacionais Windows, Linux, Mac OS, Pocket PC ou até embarcado em celulares, possibilitando a conversação entre duas ou mais pessoas ao mesmo tempo.

Quando grupos de usuários desejam trocar informações entre si, diversas conexões ponto-a-ponto devem ser estabelecidas, interconectando os usuários uns aos outros, ou seja, todos os usuários devem estabelecer conexões, seja com um servidor, seja uns com os outros.

Em situações nas quais uma aplicação colaborativa interage com um número massivo de pessoas temos duas hipóteses. A primeira em que cada usuário deve transmitir as informações para o servidor, e este encaminhará estas informações para cada um dos demais usuários individualmente. Na segunda hipótese, há o próprio envio direto pelo usuário da informação para cada um dos membros do grupo.

O uso de conexões ponto-a-ponto em aplicações coletivas não tem comportamento escalável, pois a necessidade de estabelecer múltiplas conexões ponto-a-ponto entre todos os usuários influencia diretamente o uso da rede, ou seja, a cada usuário que entra no grupo, uma nova conexão (ou mais) deve ser estabelecida, gerando um excesso de tráfego redundante na rede, o que pode inviabilizar a aplicação com grande número de usuários.

Os protocolos *Multicast* possibilitam a troca de informações coletivas através de um único endereço de rede capaz de abranger todo um grupo de usuários. Em outras palavras, um endereço *Multicast* permite que uma determinada aplicação enxergue o grupo não como um conjunto de conexões ponto-a-ponto, mas sim como uma única conexão coletiva, proporcionando uma significativa diminuição dos recursos de rede por inibir o reenvio constante de dados para cada um dos membros do grupo. A Seção 2.1 irá fundamentar o conceito *Multicast*, explorando suas características e propriedades.

²Skype® – Empresa que explora em seu software a transmissão de voz sobre a rede IP (VoIP). Disponível em: www.skype.com

Tabela 1.1: Exemplos de serviços *Multicast* e suas aplicabilidades em ambientes seguros

Serviço <i>Multicast</i>	Aplicabilidade em ambientes seguros
Televisão Digital	Proteção da programação (<i>Pay-per-view</i>)
Conferência virtual	Conversações reservadas
Jogos e rádios <i>on-line</i>	Canais privados e seguros
Atualização de Software	Proteção e integridade dos dados
Sistemas financeiros	Atualização de contabilidade entre matriz-filial
Troca de informações entre roteadores	Confidencialidade nas trocas de tabelas de roteamento

A utilização do *Multicast* não apenas diminui o uso de rede, mas também viabiliza o emprego de aplicações colaborativas em dispositivos com capacidade de banda limitada, tal como PDAs³ ou *smartphones* que usualmente utilizam redes GPRS⁴ como forma de conexão.

Grande quantidade de aplicações *Unicast* necessita de segurança. Acesso a dados bancários, envio ou recebimento de informações sigilosas, conexões em zonas restritas de sites, entre outros, são alguns exemplos de aplicações que exigem segurança e privacidade entre as partes.

Em um cenário *Multicast*, diversas aplicações também necessitam de segurança. A Tabela 1.1 arrola alguns exemplos de serviços *Multicast* existentes e suas possíveis extensões, caso exista privacidade dos dados veiculados pelo grupo.

Quando há necessidade de proteção de conteúdo, o aplicativo deve cifrar todas as informações transmitidas de modo que apenas os destinatários que conhecem o segredo possam decodificá-las.

Ao contrário de conexões *Unicast* em que as chaves são facilmente negociadas durante o estabelecimento da conexão segura (veja IPSec, Thayer et al. (1998)), em um grupo *Multicast* tal não ocorre, ao menos diretamente.

Ao estudar modelos de segurança aplicados a grupos *Multicast*, por envolver um conjunto de usuários, surgem diversas peculiaridades não existentes no modelo de segurança *Unicast*. Algumas destas singularidades serão apresentadas no decorrer desta dissertação.

³PDA – Pequeno computador portátil que oferece ferramentas para trabalhos rotineiros de escritório e pessoal.

⁴GPRS – Serviço de transmissão de dados móvel disponibilizado para usuários GSM (*Global System for Mobile*). Informações adicionais disponíveis em <http://www.gsmworld.com/technology/gprs>

Qualquer ambiente seguro, seja *Unicast* ou *Multicast*, deve prover premissas básicas de segurança: Confidencialidade, Integridade, Autenticidade e Irretratabilidade. Tratando-se de grupos *Multicast*, o Controle de Acesso também deve ser empregado no modelo de segurança de grupo. Faz-se necessário conceituar, ainda que de forma sucinta, cada uma das premissas de segurança:

- **Confidencialidade:** O acesso aos dados por uma terceira pessoa não autorizada não pode ser permitido. O conteúdo é de acesso exclusivo do grupo e utiliza-se criptografia para alcançar a confidencialidade coletiva em que todas as informações são cifradas. O acesso às informações é restrito apenas para os membros conhecedores do segredo.
- **Integridade:** A mensagem não pode ser alterada durante a transmissão, de modo que o receptor não possa detectar esta modificação. Costumeiramente, utiliza-se função *hash* (ex.: MD5 (Rivest, 1992) e SHA-1 (Eastlake; Jones, 2001)) para detectar quaisquer modificações da mensagem durante o percurso entre o remetente e o(s) destinatário(s).
- **Autenticidade:** Faz-se necessário, em um sistema seguro, o emprego de autenticação para assegurar que o remetente seja legítimo, ou seja, o destinatário deve ser capaz de verificar que o autor da mensagem é realmente o usuário autêntico e não uma terceira pessoa mal intencionada. Existem diversas técnicas para atingir a autenticação. Via de regra, utilizam-se chaves assimétricas (ex.: RSA, Rivest et al. (1978)) em que o destinatário obtém de uma certificadora digital confiável a chave pública do remetente, utilizando-a para verificar a assinatura digital do remetente — o único conhecedor da chave privada — e certificando, por conseguinte, a autoria da mensagem.
- **Irretratabilidade:** O sistema deve oferecer um mecanismo para garantir que as partes envolvidas em uma transmissão de dados não possam retratar estas informações futuramente, ou seja, deve ser possível certificar, tanto pelo remetente, quanto pelo(s) destinatário(s), que a mensagem foi entregue ou enviada corretamente.
- **Controle de Acesso:** O Controle de Acesso deve garantir que apenas os participantes autorizados possam acessar os recursos exclusivos dos membros deste grupo. A título exemplificativo, uma terceira pessoa não deve ser capaz de entrar em um grupo *Multicast* caso não seja explicitamente habilitada. Geralmente, uma lista de usuários permitidos é incluída nos servidores confiáveis do sistema.

Além das premissas relativas de segurança citadas, deseja-se também, em uma conexão *Multicast* segura, um mecanismo de proteção contra ataques tipo DoS (*Denial Of Service*). Isso porque uma terceira pessoa mal intencionada pode atacar um servidor com requisições de união (*join*), sobrecarregando-o de forma a não ser capaz de aceitar outras requisições de usuários genuínos, o que torna o sistema indisponível durante o ataque.

Em suma, este trabalho apresenta técnicas para gerenciamento de grupos *Multicast* a fim de prover privacidade aos mesmos.

1.1 Objetivos

A presente dissertação propõe estudar, simular e analisar esquemas de distribuição e gerenciamento de chaves compartilhadas em grupos *Multicast*.

Além disso, propõe-se um novo esquema de gerenciamento de chaves — o EBCK — no Capítulo 5.

A dissertação é composta por:

1. **Estudo de ambientes:** Estudar e analisar os tipos de aplicação, cenários, topologias e políticas de segurança em grupos *Multicast* seguros;
2. **Estudo de técnicas de gerenciamento de chaves compartilhadas:** Estudar e analisar esquemas de distribuição e gerenciamento de chaves compartilhadas para proteção de conteúdo em grupos *Multicast*;
3. **Proposta de um novo esquema de segurança:** Especificação de um novo esquema para gerenciamento de chaves centralizadas para a inclusão de privacidade em grupos *Multicast*;
4. **Simulação e Análises:** Com a utilização de simulador de tráfego de rede, obter-se-ão características de comportamento para cada esquema de gerenciamento de chaves estudado.

Defronte da composição da pesquisa exposta, a dissertação subdivide-se em oito capítulos. Apresenta-se, a seguir, a disposição dos demais capítulos que compõem esta dissertação e seus respectivos resumos.

- **Capítulo 2 — Estudo bibliográfico:** O Capítulo 2 apresentará revisão da literatura, apontando as principais características do *Multicast*. Introduzir-se-á o conceito de segurança em grupos, descrever-se-ão e classificar-se-ão as técnicas de gerenciamento de chaves compartilhadas.
- **Capítulo 3 — Cenários, aplicações e políticas de segurança:** Serão abordados neste capítulo alguns tipos usuais de cenários e aplicações *Multicast* e as suas conseqüências nas políticas de segurança. Também serão abordados os tipos e modos de autenticação de grupo. Far-se-á, ainda, uma breve análise correlacionando política de segurança e desempenho.
- **Capítulo 4 — Esquemas de gerenciamento de chaves:** Este capítulo descreverá alguns dos esquemas de gerenciamento de chaves compartilhadas para grupos *Multicast* citados na literatura. Os esquemas estudados no Capítulo 4 serão simulados e analisados no Capítulo 7.
- **Capítulo 5 — EBCK: Uma proposta para gerenciamento de chaves:** O Capítulo 5 apresentará as especificações de um esquema para gerenciamento de chaves chamado EBCK. O esquema proposto será comparado com os esquemas apresentados no Capítulo 4 através das simulações e análises realizadas no Capítulo 7.
- **Capítulo 6 — Ferramenta e métodos de Simulação:** O Capítulo 6 apresentará a ferramenta de simulação NS-2, utilizada nesta dissertação, bem como os métodos e as ferramentas auxiliares desenvolvidas para a realização das simulações propostas.
- **Capítulo 7 — Simulações:** Serão simulados, em diversos ambientes, os esquemas de segurança apresentados no Capítulo 4, bem como o EBCK, proposto no Capítulo 5. Todas as simulações serão especificadas, apresentadas e analisadas neste mesmo capítulo.
- **Capítulo 8 — Conclusão:** O último capítulo concluirá a dissertação.

1.2 Contribuições obtidas

As contribuições trazidas por esta dissertação são as seguintes:

1. **Proposta:** Especificação de um novo esquema de gerenciamento de chaves — o EBCK — que administra as chaves do grupo de forma eficiente, tendo como principal qualidade o baixo uso de rede para realização do gerenciamento. O modelo EBCK aperfeiçoa o esquema EBS, apresentado por Morales et al. (2003);
2. **Análise:** A pesquisa apresenta análises e comparações dos esquemas lolus (Mittra, 1997), DEP (Dondeti et al., 1999d), CTKM (Wong et al., 1997), EBS (Morales et al., 2003) e EBCK (proposta). As análises obtidas podem ser utilizadas em futuros trabalhos envolvendo outros esquemas de gerenciamento;
3. **Ferramentas de simulação:** Foram desenvolvidas ferramentas de geração de cenários e extração de resultados para NS-2. Desenvolveu-se, ainda, um módulo para o NS-2 que permite simulações de aplicativos seguros utilizando esquema de gerenciamento de chaves.

2 REVISÃO DA LITERATURA

O capítulo atual define *Multicast*, explorando conceitos essenciais para seu funcionamento, além de expor algumas de suas características, vantagens e dificuldades. Na segunda parte deste mesmo capítulo, apresentar-se-á uma revisão do conceito de segurança em grupos *Multicast*, além de estudo e classificação das técnicas de gerenciamento de chaves compartilhadas.

2.1 *Multicast*

2.1.1 Definição

Internet Protocol Multicast é um mecanismo capaz de transportar um pacote proveniente de uma fonte para um ou mais destinatários sem a necessidade do reenvio redundante da mesma informação compartilhada para cada um dos destinatários que compõem um grupo de usuários.

2.1.2 Propriedades

A Figura 2.1 apresenta a síntese principal do comportamento *Unicast* e *Multicast* de forma simplificada, em um cenário onde haja um servidor de dados transmitindo informações para seis destinatários através de um roteador intermediário.

O uso do *Multicast* permite que o mesmo pacote não seja repetidamente enviado para cada um dos destinatários. Um único pacote de informação enviado a um endereço *Multicast* atinge todos os usuários deste grupo. Como contraste, o *Unicast* deve promover a redundância para distribuição dos dados coletivos, porquanto em uma conexão *Unicast* só é possível estabelecer conexões ponto-a-ponto.

Figura 2.1: Transmissão *Unicast* vs. *Multicast*

Estendendo o exemplo apresentado na Figura 2.1 em um cenário mais completo e complexo, a Figura 2.2 apresenta um novo exemplo de um cenário de transmissão *Multicast*, porém, proporcionando uma visão sistêmica de seu funcionamento. No exemplo, existe um servidor transmitindo dados para um grupo composto por seis clientes em uma infra-estrutura com cinco redes conectadas por quatro roteadores.

Conforme apresentado na Figura 2.1, um pacote *Multicast* é capaz de ser duplicado pelos roteadores intermediários que compõem a infra-estrutura da rede para dois ou mais segmentos da mesma, atingindo, assim, todos os membros do grupo com apenas um único pacote de origem (ver Figura 2.2).

Desta característica do *Multicast*, compreendem-se os benefícios de sua utilização em aplicações colaborativas. Efetivamente, o *Multicast* reduz a banda necessária para o envio dos dados compartilhados e, desta forma, diminui ou elimina a redundância de pacotes, fazendo com que os dados veiculados circulem por um maior tempo possível pela rede antes de serem duplicados.

2.1.3 Vantagens

O uso de protocolos *Multicast* em aplicações cooperativas pode oferecer diversos benefícios e vantagens para a distribuição do conteúdo compartilhado para grupos de usuários. Os

Figura 2.2: Exemplo de cenário *Multicast*

itens a seguir abordam — porém não todas — as vantagens obtidas quando a aplicação colaborativa opta por utilizar endereço *Multicast*.

2.1.3.1 Redução de banda

A significativa redução do uso dos recursos de rede para a distribuição dos dados compartilhados do grupo é a característica mais importante do endereço *Multicast*.

Como apresentado na Seção 2.1.2, o protocolo ponto-a-ponto *Unicast* promove a redundância da informação quando o conteúdo transmitido é comum a todos os membros de um grupo. Esta redundância não ocorre em redes *Multicast*.

A Figura 2.3 apresenta um gráfico que relaciona o número de membros do grupo com a banda utilizada pelo servidor de dados. O gráfico é teórico e despreza qualquer dado adicional dos protocolos de roteamento, troca de informações entre servidores ou qualquer outra informação adicional além dos próprios dados transmitidos.

Em conexões *Unicast*, o aumento da banda necessária para a distribuição da informação coletiva para um grupo de usuários de tamanho n é definido por $O(n)$, ou seja, a inclusão de novos usuários é linearmente proporcional ao volume de dados transmitidos pelo servidor.

Figura 2.3: Relação entre o número de usuários e a banda utilizada para o envio de dados em redes *Unicast* e *Multicast*

No caso do *Multicast*, a relação entre o número de usuários e o volume de dados transmitidos pelo servidor é definida por $O(1)$, ou seja, o aumento no número de membros no grupo não afeta o volume de dados enviados pelo servidor para atender todos os usuários, pois, para qualquer número de usuários no grupo, é necessário enviar apenas uma única informação para um endereço *Multicast*.

Destarte, a demanda necessária para a veiculação de uma mesma informação para um grupo de usuários *Multicast* segue uma proporção constante, enquanto em um mesmo modelo *Unicast* tem-se progressão linear (Williamson, 1999, pp. 8–9).

Em virtude do número de usuários não proporcionar uma significativa redução de desempenho, o *Multicast* é considerado um esquema escalável.

2.1.3.2 Gerenciamento do grupo

O controle e o gerenciamento dos membros de um grupo são mais fáceis quando se utiliza *Multicast*, pois o endereço IP *Multicast* é único e fixo, independente do número de usuários ativos.

Ilustrando a asserção anterior, supõe-se uma aplicação de videoconferência composta por dezenas ou centenas de usuários, onde cada membro envia e recebe dados uns dos outros. A aplicação não possui servidor centralizado e qualquer informação enviada por um usuário

deve atingir todos os demais usuários do grupo.

No modelo *Unicast*, cada usuário deverá manter e gerenciar uma lista de participantes ativos e transmitir individualmente as informações para cada um dos membros desta lista. Em caso de desistência ou inclusão de um usuário, o membro deverá notificar todos os demais usuários de sua ação, obrigando todos os usuários a atualizar as suas tabelas de membros ativos. Espera-se, desta forma, dificuldades no gerenciamento e no sincronismo do grupo, pois cada membro deve manter sua própria tabela de usuários e cada tabela deve estar em harmonia com as demais.

Mais ainda, caso a aplicação necessite de algum mecanismo de controle de grupo e restrição de envio ou recebimento de mensagens, cada membro deverá conhecer quais usuários do grupo podem (ou não) enviar e receber mensagens. Cada membro deve adequar-se às novas regras e condições em suas tabelas de controle, tornando o gerenciamento do grupo mais complexo e propiciando erros de inconsistência ou falhas de segurança no sistema.

Por outro lado, o IP *Multicast* permite a inclusão ou exclusão de membros no grupo de forma transparente para a aplicação, e apenas os roteadores que compõem a rede têm o consentimento e controle dos eventos de inclusão, exclusão, permissão e manutenção dos usuários do grupo. Estes roteadores deverão ter, por sua vez, tabelas e rotas relacionadas ao grupo constantemente atualizadas. As aplicação devem apenas ocupar-se com o envio para um único endereço IP *Multicast*.

Porém, para incorporar mecanismos de segurança em um grupo, enquanto que no modelo *Unicast* basta o estabelecimento de conexões ponto-a-ponto seguras para restringir o acesso ao conteúdo por terceiros não autorizados, o *Multicast* provê um mecanismo mais sofisticado para garantir a segurança do grupo, geralmente com a utilização de segredo compartilhado em que apenas os membros que conhecem este segredo podem criptografar ou decriptar os conteúdos trafegados.

Os problemas aumentam quando usuários entram e saem do grupo durante a utilização do serviço e precisam renovar de forma eficiente a chave compartilhada do grupo.

Esta dissertação apresenta estudo dos esquemas de gerenciamento de chaves compartilhadas utilizando comunicações *Multicast*.

2.1.3.3 Desempenho

Tomando-se por base o exemplo da aplicação multimídia que realiza conferência digital, no caso do *Unicast*, cada usuário deve percorrer toda a lista de membros ativos e verificar suas permissões de envio e recebimento e enviar para cada um destes membros a informação coletiva desejada. São necessários taxa de processamento e uso de memória para percorrer toda a lista de usuários. É preciso ainda, acessar repetidamente os recursos de entrada e saída do Sistema Operacional, geralmente muito lento. Considerando que todo este processo deve ser executado para cada pacote, problemas de desempenho podem surgir.

Por outro lado, ao utilizar *Multicast*, o servidor ou os membros não precisam fazer nenhum tipo de verificação: basta enviar um único pacote de dados para o endereço IP *Multicast* do grupo, eliminando tempo de processamento e acesso à memória para percorrer a lista de usuários do grupo e, ainda, diminuindo o acesso ao sistema de entrada e saída do Sistema Operacional para o envio das mensagens.

Destarte, as relações $O(n)$ para *Unicast* e $O(1)$ para *Multicast*, são válidas também sob o ponto de vista de desempenho computacional.

Por fim, além da redução da utilização de recursos computacionais por cada membro, os usuários *Multicast* não precisam enviar pacotes redundantes pela sua interface de rede, fazendo com que os roteadores necessitem de menos processamento e memória para manusear os dados da rede (Williamson, 1999, p. 10).

2.1.4 Especificação

A seção atual traz algumas especificações *Multicast* fundamentais para a compreensão e entendimento do *Multicast*. Embora o estudo do *Multicast* não seja o foco desta dissertação, as suas definições e propriedades são fundamentais para a compreensão dos esquemas de segurança apresentados nesta pesquisa, foco deste trabalho.

A definição atual do *Multicast* segue certas especificações. Nos itens abaixo, apresentam-se algumas das principais características *Multicast*.

- Em IPv4 (versão 4 do *Internet Protocol*) — padrão que prevalece na Internet atualmente — o endereço IP *Multicast* é definido em RFC 1112 (Deering, 1989) e os seus

endereços são representados pela classe D, cuja faixa de endereçamento varia entre 224.0.0.0 e 239.255.255.255.

- O *Multicast* utiliza o UDP como mecanismo de transporte (Williamson, 1999, pp. 13–15), pois caso fosse TCP, os destinatários (possivelmente dezenas de milhares) enviariam pacotes de controle do tipo ACK (*acknowledged*) para informar que os pacotes sejam recebidos (ou não) e, desta forma, o servidor receberia um enorme fluxo de pacotes tipo ACK, comprometendo significativamente o seu desempenho. Além disso, se um único usuário não recebesse um determinado pacote, o servidor seria obrigado a cessar todo o fluxo e repetir toda a seqüência perdida para todos os membros *Multicast*, o que também comprometeria o seu desempenho.
- Não há mensagens de controle (pacotes do tipo ICMP). A única exceção é a mensagem de controle do tipo *Echo* e *Echo Reply*, utilizada para o controle do *Multicast* (Moy, 1998, p. 174). Como já exposto anteriormente, um único pacote *Multicast* pode ser replicado para diversos segmentos da rede e se algum destes segmentos não for encontrado ou o TTL¹ (*Time to Live*) do pacote expirar, vários pacotes tipo ICMP seriam enviados de volta para o servidor para cada datagrama enviado, comprometendo de forma significativa o desempenho do *Multicast*.

Existem diversas outras propriedades e especificações do *Multicast* não citadas, mas para o escopo deste trabalho são suficientes as informações supra expostas.

Para informações adicionais sobre o modo através do qual o *Multicast* atua nas camadas de enlace e IP, vide o Apêndice A.

2.1.5 Gerenciamento de grupos *Multicast*

Uma das propriedades mais importantes do *Multicast* é o sistema de gerenciamento de grupo. É importante, para a geração de modelos seguros eficientes, o entendimento do modo pelo qual os usuários requisitam ou cessam as suas atividades em grupo *Multicast*.

¹TTL é o número máximo de máquinas que os pacotes podem percorrer numa rede de computadores antes de serem descartados. Qualquer roteador está programado para decrementar uma unidade do TTL aos pacotes veiculados. Esta é uma forma de evitar que os pacotes permaneçam na rede por tempo infinito.

Figura 2.4: Formato da mensagem IGMP

Os protocolos de gerenciamento *Multicast* têm por objetivos os seguintes, rol não exaustivo:

1. Permitir a um membro entrar ou deixar o grupo a qualquer momento;
2. Estabelecer padrões de informações de controle entre membros e roteadores;
3. Instituir níveis de privilégios sobre o uso do grupo;
4. Fornecer mecanismos de controle de acesso.

Em IPv4 e IPv6, os protocolo de gerenciamento são:

- **IPv4:** IGMP (atualmente na versão 3) (Deering, 1989).
- **IPv6:** MLD (atualmente na versão 2) (Deering et al., 1999).

2.1.5.1 IGMP

Como o ICMP, o protocolo IGMP utiliza o UDP para veicular a informação de controle (Doyle; Carroll, 2001, p. 411). O pacote IGMP é formado por oito octetos fixos. A Figura 2.4 apresenta a estrutura de uma mensagem IGMP.

Tabela 2.1: Tipos de mensagens IGMP

Type	Group Address	Meaning
0x11	<i>Unused (Zero)</i>	<i>General membership query</i>
0x11	<i>Used</i>	<i>Specific group membership query</i>
0x16	<i>Used</i>	<i>Membership report</i>
0x17	<i>Used</i>	<i>Leave group</i>
0x12	<i>Used</i>	<i>Membership report (version 1)</i>

O campo *Type* da estrutura IGMP apresentado na Figura 2.4 permite a identificação do tipo de pacote, o qual pode conter valores de acordo com a Tabela 2.1.

O próximo campo do IGMP, o *Resp Time*, cuida da manutenção do grupo verificando se todos os membros do segmento desta rede ainda estão ativos. O *Resp Time* é a máxima tolerância — medida em dezenas de segundos — que um roteador aguardará antes de deixar de encaminhar pacotes *Multicast* para este segmento de rede.

Quando membros do grupo recebem a mensagem IGMP, cada membro inicia uma contagem aleatória entre 0 e *Resp Time* (geralmente 10s). Ao término da contagem, o membro envia uma mensagem para o roteador confirmando a presença de pelo menos um membro ativo neste segmento de rede. Quando há mais de um membro neste segmento, o primeiro que enviar a resposta fará com que os outros membros cessem a contagem, visto que agora não é mais necessário comprovar a presença de usuários ativos neste segmento de rede.

Por fim, o campo *Checksum* da mensagem IGMP é responsável por garantir a integridade dos dados veiculados, verificando se o conteúdo foi corrompido através da comparação do valor CRC (*Cyclic Redundancy Check*) da mensagem com o valor CRC gerado.

Para a manutenção de lista de usuários ativos entre os roteadores da rede, estes deverão manter tabelas dos membros ativos em cada segmento. Quando a contagem de membros de um trecho de rede chegar a zero, o roteador deve avisar os demais roteadores para cessar o envio de pacotes para aquele segmento.

2.1.5.2 MLD

O MLD é a versão do IGMP para redes IPv6. O protocolo MLD é uma parte do protocolo ICMPv6, não um protocolo independente como o IGMP. O estudo mais aprofundado deste

protocolo diverge do escopo desta dissertação e, desta forma, apresentar-se-á apenas um breve comentário de suas versões.

- **MLD v1:** Não contém filtragem de fonte e todos os membros podem transmitir para este grupo.
- **MLD v2:** Permite aos *hosts* receber ou bloquear tráfego de determinadas fontes.

2.1.6 Propriedades desejáveis em protocolos *Multicast*

Esta seção veiculará alguns atributos desejáveis do *Multicast*. Trata-se de importante seção na medida em que permite a compreensão dos requisitos e limitações impostos aos grupos *Multicast*, os quais podem influenciar nos esquemas de gerenciamento de chaves de grupos.

- A entrada e a saída de um membro em um grupo *Multicast* devem ser realizadas dinamicamente, ou seja, o servidor de dados ou qualquer outro membro em estado de transmissão não podem cessar o fluxo de envio quando qualquer outro membro entrar ou sair do grupo.
- O roteamento dos dados deve ser estabelecido dinamicamente, ou seja, se um usuário entra ou sai do grupo *Multicast*, as informações necessárias para a veiculação dos dados *Multicast* devem ser estabelecidas sem comprometer o fluxo de transmissão atual de dados para os demais usuários.
- Cada membro do grupo pode estar em qualquer local da rede Internet, ou seja, não podem haver restrições quanto a sua localidade como, por exemplo, a obrigatoriedade de que pertença a uma mesma rede local ou a um mesmo sistema autônomo, exceto se tal condição for explicitamente definida pelo administrador da rede.
- Capacidade de estabelecer privilégios de participação.
- O *Multicast* deve permitir a autenticação, tanto pelo usuário, quanto pelo servidor.
- O *Multicast* deve permitir a autenticação dos dados de origem (autoria da mensagem).
- O *Multicast* deve fornecer ferramentas de segurança e privacidade para o grupo, assim como proteção do conteúdo veiculado.

- Deve-se possibilitar a existência do *Multicast* confiável: uma aplicação pode requerer a garantia de que todos os membros do grupo receberam todos os pacotes corretamente e em seqüência correta, similarmente ao protocolo TCP em conexões *Unicast*.

Saliente-se a extrema importância e necessidade, em quase todos os esquemas de gerenciamento de chaves de grupo, do *Multicast* confiável. Para gerenciar as chaves dos grupos, o administrador dos mesmos deve gerar mensagens de controle e enviá-las periodicamente para o grupo. Caso algum dos usuários não receba corretamente as informações, pode perder o direito de acessar os dados do grupo.

O uso de *Multicast* confiável em esquemas de gerenciamento de chave de grupo será melhor compreendido no Capítulo 4. A seção 2.1.7 apresentará os problemas e algumas metodologias para atingir a confiabilidade em conexões *Multicast*.

2.1.7 *Multicast* Confiável

Uma das mais importantes funcionalidades aplicadas à rede *Multicast* necessária ou desejável pelos esquemas de gerenciamentos de chaves compartilhadas é o *Multicast* confiável ou *Multicast Reliability* (Wittmann; Zitterbart, 2000, p. 29).

O estudo de técnicas para prover o *Multicast* confiável não corresponde ao escopo desta pesquisa. Entretanto, breve apresentação de algumas metodologias aplicadas afigura-se importante em razão de permitir o estudo do grau de robustez necessário durante a inclusão dos esquemas de gerenciamento de chaves compartilhadas em grupos seguros.

A quase totalidade dos protocolos de *Multicast* confiável baseia-se em uma (ou um conjunto) das seguintes práticas (Chandra et al., 2001):

- Rotular cada pacote *Multicast* com uma seqüência única de forma que cada membro possa ordenar e identificar a existência do pacote extraviado.
- Adotar uma política de NACK, *non-acknowledgement*: um membro apenas se manifesta quando não recebe um datagrama, caso contrário haveria uma explosão de *acknowledgement*.

TABELA DE POSIÇÃO 001 – R1 010 – R2 011 – D1 100 – R3 101 – D2 110 – D3 111 – D4	Hamming Code Gerando pacotes Rn: $R1 = D1 \text{ xor } D2 \text{ xor } D4$ $R2 = D1 \text{ xor } D3 \text{ xor } D4$ $R3 = D2 \text{ xor } D3 \text{ xor } D4$ D = Pacote de Dados R = Pacote redundante	Obtendo N tal que: $N = \text{bit1 bit2 bit3}$ Onde: $\text{bit1} = R1 \text{ xor } D1 \text{ xor } D2 \text{ xor } D4$ $\text{bit2} = R2 \text{ xor } D1 \text{ xor } D3 \text{ xor } D4$ $\text{bit3} = R3 \text{ xor } D2 \text{ xor } D3 \text{ xor } D4$ Se N for igual a 0, então Recebimento correto Senão Erro na posição N																								
EXEMPLO <table> <tr> <td>Valores:</td><td>Gerando erro:</td><td>Verificação:</td></tr> <tr> <td>D1 = 0</td><td>D1 = 1 (!)</td><td>B1: $R1 \text{ xor } D1 \text{ xor } D2 \text{ xor } D4 = 0$</td></tr> <tr> <td>D2 = 1</td><td>D2 = 0</td><td>B2: $R2 \text{ xor } D1 \text{ xor } D3 \text{ xor } D4 = 1$</td></tr> <tr> <td>D3 = 0</td><td>D3 = 0</td><td>B3: $R3 \text{ xor } D2 \text{ xor } D3 \text{ xor } D4 = 1$</td></tr> <tr> <td>D4 = 1</td><td>D4 = 1</td><td></td></tr> <tr> <td>R1 = 0</td><td>R1 = 0</td><td>N: B1 B2 B3 = 011 = D1</td></tr> <tr> <td>R2 = 1</td><td>R2 = 1</td><td></td></tr> <tr> <td>R3 = 0</td><td>R3 = 0</td><td>Correção: D1 = 0</td></tr> </table>			Valores:	Gerando erro:	Verificação:	D1 = 0	D1 = 1 (!)	B1: $R1 \text{ xor } D1 \text{ xor } D2 \text{ xor } D4 = 0$	D2 = 1	D2 = 0	B2: $R2 \text{ xor } D1 \text{ xor } D3 \text{ xor } D4 = 1$	D3 = 0	D3 = 0	B3: $R3 \text{ xor } D2 \text{ xor } D3 \text{ xor } D4 = 1$	D4 = 1	D4 = 1		R1 = 0	R1 = 0	N: B1 B2 B3 = 011 = D1	R2 = 1	R2 = 1		R3 = 0	R3 = 0	Correção: D1 = 0
Valores:	Gerando erro:	Verificação:																								
D1 = 0	D1 = 1 (!)	B1: $R1 \text{ xor } D1 \text{ xor } D2 \text{ xor } D4 = 0$																								
D2 = 1	D2 = 0	B2: $R2 \text{ xor } D1 \text{ xor } D3 \text{ xor } D4 = 1$																								
D3 = 0	D3 = 0	B3: $R3 \text{ xor } D2 \text{ xor } D3 \text{ xor } D4 = 1$																								
D4 = 1	D4 = 1																									
R1 = 0	R1 = 0	N: B1 B2 B3 = 011 = D1																								
R2 = 1	R2 = 1																									
R3 = 0	R3 = 0	Correção: D1 = 0																								

Figura 2.5: *Hamming Code*

- Usar roteadores intermediários como pontos de reconhecimento DR (*Designated Router*) para o “cacheamento” dos pacotes. Na hipótese de algum usuário não receber algum pacote, deverá notificar o DR mais próximo e receberá deste DR o pacote faltante. O servidor de dados só será requisitado se um dos roteadores intermediários DR não receber algum determinado pacote. Uma hierarquia pode ser formada, figurando no topo o servidor de dados e, nas folhas, os membros.
- Promover uma pequena redundância de informação para reduzir ou eliminar a retransmissão, similarmente utilizada na correção de erro (*error-correction*) de CD de áudio ou em sistemas RAID (*Redundant Array of Independent Disks*).

A Figura 2.5 demonstra uma das metodologias de inclusão de redundância para detecção e recuperação de pacotes conhecida como algoritmo de *Hamming Code* (Gitlin et al., 1992, p. 181):

Supõe-se, pela Figura 2.5, que cada pacote contém um único *bit*. Para enviar quatro pacotes (D1, D2, D3 e D4) são necessárias três informações redundantes (R1, R2 e R3). Constroem-se os pacotes Rn com a utilização do operador *xor*, de acordo com a figura, e estes sete pacotes são enviados para o grupo.

Como exemplo, caso haja um erro no pacote D1 (o erro poderia ocorrer em qualquer um,

Inclusão de Redundância	
Pacote de dados 1:	01 00011011
Pacote de dados 2:	02 01101101
Pacote de dados 3:	03 11010000
Pacote de paridade 4: $(1 \oplus 2 \oplus 3)$	04 10100110
Para reconstruir o pacote 3 extraviado, aplica-se a operação <i>xor</i> nos pacotes 1, 2 e 4:	
	03' 11010000

Figura 2.6: Exemplo de redundância para recuperação de dados

incluindo os pacotes redundantes), a verificação de N obtém um valor diferente de zero, indicando que há um erro no pacote 011, conforme indicado na tabela. Desta forma, o erro não foi apenas detectado, mas o pacote defeituoso foi identificado e corrigido.

No caso do *Multicast*, em razão da utilização de pacotes UDP, os quais aplicam CRC para checar a sua integridade, podem ser adotadas metodologias mais simplificadas como, por exemplo, um simples pacote adicional redundante, contendo o valor de paridade. Um pacote UDP chega a seu destino integralmente ou, então, não chega, sendo inviável chegar com valores corrompidos. Assim, não há necessidade de detecção e identificação de erro.

A Figura 2.6 ilustra o uso de paridade. Ao enviar três pacotes de dados com um *Byte* cada, um quarto pacote de paridade é gerado pela operação *xor* entre os três pacotes anteriores. Estes quatro pacotes, então, são rotulados e enviados normalmente para todos os membros.

Caso um membro não receba algum dos pacotes, por exemplo, o pacote 03, o mesmo pode ser reconstituído aplicando novamente a operação *xor* dos demais pacotes 1, 2 e 4 (este último de paridade). Desta forma, com a inclusão de uma pequena informação adicional, o sistema torna-se mais robusto contra perda ou extravio de pacotes e, desta forma, poupa o servidor de reenviar constantemente os pacotes perdidos.

A relação entre os pacotes de dados e a quantidade de pacotes de paridade pode ser facilmente estabelecida e balanceada. Saliente-se que em ambientes menos hostis são utilizados pacotes redundantes (paridade) em quantidade inferior comparativamente a ambientes mais hostis. Esta relação é usualmente estabelecida dinamicamente, através da ponderação da

quantidade de pacotes de paridade com requisições de retransmissões.

Para maiores informações sobre o modo pelo qual o *Multicast* é transmitido pela rede, os problemas que podem ser ocasionados e, ainda, sobre alguns protocolos *Multicast*, vide o Apêndice B.

2.2 Segurança em *Multicast*

Ao estudar segurança em redes *Multicast*, primeiramente é necessário compreender os principais assuntos a serem considerados durante a construção ou emprego de um modelo *Multicast* seguro.

Hardjono; Tsudik (2000) identificou e classificou quatro tópicos essenciais a serem abordados em qualquer ambiente *Multicast* seguro:

1. **Autenticação de fonte de dados *Multicast*:** Em um grupo seguro, quaisquer dados enviados para o grupo devem prover autenticidade dos autores. Fontes suspeitas podem enviar dados falsos, confundindo ou desorientando uma transmissão de dados, mesmo se seu conteúdo estiver cifrado e for desconhecido por um usuário malicioso. A autenticação, em geral, também deve prover integridade para garantir que os dados não sejam alterados no percurso. Geralmente, o uso de assinaturas digitais MAC ou HMAC (Krawczyk et al., 1997) possibilita a autenticidade e integridade simultaneamente. A Seção 3.2 abordará com mais detalhes as formas de autenticação;
2. **Políticas de segurança *Multicast*:** A definição e o emprego das políticas de segurança que coordenam o mecanismo de segurança em grupos *Multicast* são fatores fundamentais para a segurança de todo o sistema. Hardjono; Tsudik (2000) dividiram a segurança do grupo em política de administração de grupo e política aplicada à segurança. A primeira, é responsável pelo gerenciamento dos membros, enquanto que a política aplicada à segurança é responsável pela proteção dos dados do grupo;
3. **Confidencialidade dos dados *Multicast*:** Em uma comunicação *Multicast* segura é necessário confidencializar os dados trafegados, ou seja, não deve ser permitido o acesso aos dados veiculados por usuários não habilitados. Para prover confidencialidade de grupo é necessário utilizar criptografias. Ademais, uma ou mais chaves devem

ser conhecidas por todo o grupo. O remetente deve enviar as informações criptografando os dados com esta chave, de modo a possibilitar que apenas os membros do grupo possam decodificá-las;

4. **Gerenciamento de chaves em grupo *Multicast*:** A confidencialidade dos dados é realizada por meio de criptografia. Porém, as chaves simétricas ou assimétricas para criptografar as informações devem ser distribuídas e gerenciadas de forma coerente e segura. O papel de um gerenciador de chaves de grupo é gerar, administrar e atualizar estas chaves compartilhadas. Quando há uma ocorrência de união (*join*) ou desistência (*leave*), novas chaves devem ser geradas e atualizadas e, de acordo com políticas de segurança, deve existir uma renovação incondicional temporal das chaves do grupo.

A presente dissertação foca-se no gerenciamento de chaves compartilhadas.

2.2.1 Requisitos de segurança

O *Multicast* IPv4 não provê mecanismos de controle de acesso aos dados veiculados pelos membros (Ballardie, 1996) e, mais ainda, qualquer *host* com *Multicast* habilitado pode enviar uma requisição IGMP e unir-se ao grupo *Multicast* (Fenner, 1997). Não há controle de acesso ou autenticação neste processo (Hardjono; Tsudik, 2000).

Devido à ausência de controle de acesso, a construção de um sistema *Multicast* deve ser realizada através de um esquema de segurança estendido em conjunto com os recursos oferecidos pelo IGMP (no caso IPv4) e com os protocolos de roteamento (ex. DVMRP, PIM-DM, outros). Desta forma, obtém-se a segurança do grupo.

Para estudar esquemas de segurança *Multicast*, é necessário entender as suas hipóteses de uso. A Figura 2.7 apresenta um diagrama Caso de Uso, descrevendo os eventos que cada membro e, bem assim o servidor de autenticação, devem realizar. Estes eventos são fundamentais para a elaboração de um esquema de gerenciamento de chaves compartilhadas.

Figura 2.7: Diagrama Caso de Uso

2.2.2 Privacidade em grupos *Multicast*

Para alcançar a privacidade dos conteúdos que circulam pela rede, um grupo *Multicast* deve proteger o conteúdo destes dados, criptografando-os.

O método mais simples para proteger os dados é gerar uma chave aleatória simétrica e distribuí-la para todos os membros do grupo. Assim, todas as informações são codificadas utilizando-se algum algoritmo de criptografia (Schneier, 1996), de forma que as informações apenas possam ser decodificadas e lidas pelos demais membros do grupo que conhecem o segredo.

A Figura 2.8 ilustra um cenário com um grupo *Multicast* utilizando uma chave compartilhada para proteger o conteúdo do grupo.

O cenário apresentado na Figura 2.8 exemplifica a troca de informações de um grupo através da utilização de uma chave compartilhada. Apenas os membros legítimos do grupo têm a posse desta chave e, então, o servidor de dados codifica todo o conteúdo e o envia pela árvore *Multicast*. Apenas os membros que conhecem a chave compartilhada são capazes de decodificar os datagramas e obter os conteúdos plenos.

A chave deve ser constantemente trocada para manter a privacidade do grupo (Schneier, 1996) e a nova chave deve ser independente de todas as anteriores.

Figura 2.8: Exemplo de chave compartilhada em transmissão *Multicast*

Adicionalmente, deve existir um mecanismo de renovação de chave de grupo para prover a proteção temporal dos dados (*Backward Confidentiality* e *Forward Confidentiality*) do grupo (Wallner et al., 1999; Kim et al., 2000). A seção seguinte define o conceito da confidencialidade temporal.

2.2.3 Confidencialidade Temporal

A confidencialidade temporal dos dados pode ser definida a partir da consideração de dois casos:

1. **Entrada de usuário:** Quando um novo usuário entra no grupo, não deve ter acesso aos pacotes veiculados até a sua entrada. Uma nova chave deve ser gerada e distribuída para o grupo no momento da entrada deste novo usuário, sendo que o mesmo não tem acesso à chave antiga, apenas à nova chave do grupo;

Figura 2.9: Exemplo de confidencialidade temporal

2. **Saída de usuário:** Quando um usuário sai do grupo, este não deve mais ter acesso aos pacotes veiculados. Uma nova chave de grupo deve ser gerada e distribuída para todo o restante do grupo.

Exemplificando a confidencialidade temporal, a Figura 2.9² demonstra três usuários entrando e saindo do grupo ao longo do tempo.

No tempo t_i , Alice ingressa no grupo e uma nova chave K_1 é gerada, substituindo a antiga chave K_0 . A chave K_1 é distribuída para todos os membros, incluindo Alice. Note-se que Alice não tem a chave K_0 e, portanto, não pode decodificar os dados enviados até o instante t_i . No caso, Alice poderia armazenar todo o fluxo de dados e depois decodificá-lo. No tempo t_{i+1} , Bob ingressa no grupo e o mesmo processo de renovação de chaves ($K_1 \rightarrow K_2$) ocorre.

No tempo t_{i+2} , Alice deixa o grupo. Uma nova chave K_3 é gerada, mas Alice não terá acesso a esta nova chave. Note que Alice obteve as chaves K_1 e K_2 somente, podendo ter acesso apenas aos pacotes veiculados entre t_i e t_{i+2} , o período exato em que Alice permaneceu no grupo. Finalmente, no instante t_{i+3} Carol ingressa no grupo, sofrendo o mesmo processo de Alice e Bob quando estes ingressaram no grupo.

2.2.4 Renovação de chaves

O gerenciador de chaves tem como função administrar todas as chaves do grupo, atualizando as mesmas entre todos os usuários pertencentes ao *Multicast*.

²Baseada na Figura 1 do artigo Mittra (1997)

Modelo de pacote:

Cabeçalho	$(S)_{U_1}$	$(S)_{U_2}$	$(S)_{U_3}$	...	$(S)_{U_n}$
-----------	-------------	-------------	-------------	-----	-------------

Figura 2.10: Esboço de estrutura de pacote para distribuição de chaves

Para renovar uma chave compartilhada de um grupo, no caso mais simplório, cada usuário deverá possuir uma chave pessoal K_{usr_n} . Esta chave é gerada e enviada no momento em que o usuário requisita a sua entrada no grupo.

O servidor mantém uma lista de todas as chaves dos usuários. Nenhum usuário conhece a chave pessoal de outro, apenas o gerenciador de chaves tem a posse de todas as chaves dos membros.

O servidor, então, deve enviar um pacote com a nova chave compartilhada S do grupo, o KDC (*Key Distribution Centre*), e codificar a partir da chave individual de cada membro para garantir que apenas os usuários listados obtenham a posse da nova chave. Este modelo é conhecido como esquema plano, ou *Flat Schemes* (Rafaeli; Hutchison, 2003).

Neste esquema, é necessário um pacote de tamanho $k \times n$ em que k é uma constante, geralmente o tamanho da chave de criptografia, e n , o número de usuários atualmente no grupo. A Figura 2.10 apresenta uma estrutura de pacote para transmitir uma chave compartilhada S para todos os usuários U do grupo.

Conforme se observa, a solução apresentada não é escalável, pois o tamanho necessário para o gerenciamento de chaves é proporcional ao número de usuários, ou seja, há um crescimento de ordem $O(n)$, o que pode tornar inviável aplicações com dezenas de milhares de usuários.

Como exemplo, se utilizarmos uma criptografia fraca como a DES, com apenas 56 *bits* — em geral, os algoritmos modernos como o AES têm entre 128 a 256 *bits* — em um cenário com apenas mil usuários, cada pacote de renovação de chaves terá $k \times n = 56bits \times 1k = 56kb$, ou seja, seria necessário transmitir 7 kB por renovação de chaves (desconsiderando-se quaisquer cabeçalhos neste pacote).

Se, por exemplo, um usuário permanece no grupo em média por 1 hora, significa que cada usuário necessita de duas renovações (ao entrar e ao sair) a cada 3600 segundos ou uma renovação a cada 1800 segundos. Considerando-se a existência de mil usuários (em média), é necessária uma renovação a cada $1800s/1000 = 1,8s$. Sabendo-se que cada renovação tem 56 kb, torna-se necessária uma rede com banda de $\frac{56kb}{1,8s} = 36\frac{kb}{s}$, ou

seja, é necessária uma conexão com banda de $36 \frac{\text{kb}}{\text{s}}$ apenas para realizar o gerenciamento de chaves. O problema se agrava, pois haveria rajadas de requisições em determinados períodos, enquanto outros períodos permaneceriam ociosos. Em grupos maiores, a prática do modelo citado é inviável.

O gerenciamento das chaves de grupo deve ser realizado por um método efetivo e escalável, ou seja, a quantidade de usuários não deve impactar significativamente o comportamento de todo o sistema. O *Multicast*, como visto na Seção 2.1.3, tem um comportamento escalável e não seria conveniente incorporar um esquema de segurança não escalável para gerenciar as chaves do grupo.

Algumas técnicas de gerenciamento de chaves manipulam os dados de forma a reduzir o uso necessário da banda em cada renovação de chaves. Chega-se, pois, a um crescimento de apenas $O(\log(n))$ em relação ao número de usuários ativos no grupo, ou seja, viabiliza-se o uso para gerenciamento de grupos extremamente grandes.

O objetivo desta dissertação é, dentre outros, estudar e analisar o comportamento destas técnicas de gerenciamento de chaves escalares.

2.3 Esquemas de gerenciamento de chaves compartilhadas

A aplicação de chaves compartilhadas em grupos *Multicast* possibilitou a inclusão da confidencialidade dos dados transmitidos.

O gerenciamento de chaves em grupos *Multicast* pode ocasionar um tráfego excessivo de rede. Um gerenciador de chaves deve ser capaz de renovar as chaves do grupo utilizando menor banda de rede possível e, ao mesmo tempo, prover uma lógica confiável e robusta de segurança.

2.3.1 Requisitos de análise

Qualquer esquema de gerenciamento de chaves *Multicast* pode ser avaliado dentro dos critérios apresentados na Figura 2.11 (Challal; Seba, 2005).

Figura 2.11: Requisitos de análise para esquemas de gerenciamento de chaves

Os elementos apresentados na Figura 2.11 são resumidos a seguir:

- **Requisitos de Segurança**

- **Confidencialidade Temporal:** A cada entrada e saída, devem ser realizados procedimentos para garantir que um novo usuário não acesse as informações veiculadas até a sua entrada e, bem assim, devem haver procedimentos para que um usuário que deixe o grupo não seja mais capaz de acessar o conteúdo trafegado pelo mesmo.
- **Prevenção contra conspiração:** Caso dois ou mais membros se reúnam contra o gerenciador de chaves, este deve prover meios para banir o grupo conspirador sem impactar significativamente o desempenho do sistema.
- **Independência de chaves:** Caso alguma chave do grupo seja revelada para terceiros, esta não pode comprometer nenhuma das demais chaves do sistema.
- **Confiabilidade mínima:** O gerenciador de chaves deve confiar na menor quantidade de *hosts* possível. Noutras palavras, esquemas de segurança que utilizam diversos *hosts* para o auxílio do gerenciamento de chaves são mais propícios a falhas de segurança e confiabilidade.

- **Requisitos de QoS (*Quality Of Service*)**

- **Baixo consumo de banda:** O sistema deve reduzir ao máximo o uso de banda para a realização do gerenciamento das chaves de grupo.
- **1 afeta n:** Quando um usuário ingressa ou deixa o sistema, outros podem ser afetados. Caso todo o sistema seja afetado, o impacto deve ser o menor possível. Por exemplo, quando um membro é excluído do sistema, todos os usuários devem renovar a chave do grupo.
- **Atraso:** A renovação de chaves não pode atrasar o fluxo de dados, ou seja, após um evento, o grupo não deve perder transitoriamente o acesso aos dados.
- **Robustez:** Caso pacotes de controle sejam perdidos, as conseqüências devem ser mínimas e solucionáveis para os membros que não receberam os dados constantes dos pacotes perdidos.
- **Agrupamento de eventos:** Para atender múltiplos eventos (exemplo: requisição *join* ou *leave*), o esquema de segurança deve ser capaz de tratar todas as requisições de uma só vez e não uma de cada vez sequencialmente.

- **Requisitos do gerenciador de chaves**

- **Baixa necessidade de armazenamento:** O gerenciador de chaves deve administrar o menor número de chaves possível.
- **Baixo uso computacional:** O gerenciador de chaves não deve ser complexo a ponto de prejudicar o sistema por excesso de processamento para o gerenciamento das chaves.

- **Requisitos dos membros**

- **Baixa necessidade de armazenamento:** Os membros devem administrar o menor número de chaves possível.
- **Baixo uso computacional:** Os membros devem utilizar o mínimo de recursos computacionais para processar os dados de gerenciamento de chaves.

Figura 2.12: Controle de grupo

2.3.2 Controle de grupo, sub-grupo e por membro

Quando são avaliados os esquemas de distribuição e gerenciamento de chaves, afigura-se importante assimilar o seu modo de operação para compreender onde e de que forma podem ser aplicados, além de suas limitações de uso.

2.3.2.1 Controle de grupo

O esquema de gerenciamento e manutenção das chaves do grupo concentra-se em apenas um agente controlador. Todos os demais membros são tão-somente usuários. Quando há um modelo de controle de grupo, não há necessidade de eleger agentes confiáveis e, deste modo, o sistema, via de regra, torna-se mais robusto.

Na Figura 2.12, existem um Gerenciador de Chaves (GC) e seis Membros ($M_{1..6}$). Todo o processo de renovação e controle de chaves é centralizado pelo GC e não existe nenhum outro agente colaborador em todo o grupo.

2.3.2.2 Controle de sub-grupo

Um esquema que utiliza controle de sub-grupo é baseado no fato de que existem um ou mais nós confiáveis espalhados hierarquicamente no sistema. Cada agente é responsável por gerenciar os membros de seu sub-grupo.

Figura 2.13: Controle de sub-grupo

Na Figura 2.13, existem um Gerenciador de Chaves (GC), três Gerenciadores de sub-grupo (SG) e nove membros. Individualmente, cada um dos três grupos é controlado por seus respectivos SGs, os quais, por sua vez, são controlados pelo GC do grupo.

O controle de sub-grupo apresenta como desvantagem sobre o controle centralizado do grupo a necessidade de estabelecimento de *hosts* confiáveis (confiabilidade). Caso um dos *hosts* não seja devidamente escolhido, é possível que o mesmo esteja indisponível em certos momentos. Tal processo faz com que o sistema seja reorganizado, desperdiçando, destarte, recursos de rede.

Ainda, os esquemas baseados em sub-grupos não podem ser estendidos para outros ambientes, como, por exemplo, redes de sensores: cada nó de uma rede de sensores não tem capacidade computacional para atuar como sub-gerente de sub-grupos.

Porém, como vantagem principal, há uma significativa diminuição de banda no sistema em razão dos controladores de sub-grupo manterem os usuários reunidos em grupos menores. Desta forma, a saída e a entrada de usuários afeta apenas um segmento do sistema (1 não afeta n).

2.3.2.3 Controle por membro

Um sistema controlado por seus membros não possui nenhuma entidade específica para gerenciar o grupo. Cada membro controla os demais.

Neste modelo, a política de segurança é demasiadamente questionável e a sua implementação é muito complexa, porquanto não existe nenhum *host* confiável de fato no sistema para garantir a segurança.

Figura 2.14: Controle por membros

Na Figura 2.14 não existe nenhum Gerenciador de Chaves (GC). Os próprios membros são responsáveis por manter a segurança do grupo.

2.3.3 Classificação

Na literatura, um esquema de gerenciamento de chaves pode ser classificado em três tipos (Eskicioglu, 2002):

1. Escalabilidade

- **Esquemas escaláveis:** Um esquema escalável tem a sua curva de crescimento em relação ao número de usuários em uma ordem de $O(\log(n))$.
- **Esquemas não-escaláveis:** Um esquema não-escalável tem a sua curva de crescimento em relação ao número de usuários em uma ordem de $O(n)$ (L. R. Dondeti; Samal, 1999).

A Figura 2.15 apresenta gráfico teórico comparando os esquemas escaláveis e não escaláveis. Os modelos escaláveis têm, tipicamente, crescimento na ordem de $O(\log(n))$, enquanto que os não escaláveis têm a sua taxa de crescimento na ordem de $O(n)$.

Figura 2.15: Esquema escalável vs. não-escalável

2. Tipo de esquema de atualização (Bruschi; Rosti, 2000)

- **Esquema plano (*Flat Schemes*):** Modelo mais simples para renovação de chave (já apresentado anteriormente). Trata-se de modelo limitado, porquanto não escalável.
- **Esquemas agrupados (*Clustered Schemes*):** Similar ao esquema plano, porém, o grupo é dividido em sub-grupos em que n servidores confiáveis devem gerenciar os seus conjuntos de membros. Desta forma, o modelo se torna escalável, pois na medida em que o grupo cresce, novos servidores confiáveis são incorporados com seus respectivos conjuntos de usuários. Um exemplo deste modelo é o Iolus (Mittra, 1997).
- **Esquemas baseados em árvores (*Tree-Based Schemes*):** As chaves são organizadas em árvores em que a raiz é a chave do grupo e as folhas, as chaves individuais dos membros. Os nós são chaves que criptografam chaves (Caronni et al., 1998).
- **Sistema baseado em exclusão:** Gerencia chaves de grupo por exceção de chaves combinadas, com base em conjuntos de sub-chaves. Um exemplo é o esquema EBS (Morales et al., 2003).

3. Esquemas Centralizados, Esquemas de sub-grupos distribuídos e esquemas distribuídos (Rafaeli, 2000)

- **Esquemas Centralizados (Controle Centralizado do grupo):** Uma simples entidade controla todo o grupo responsável pelo gerenciamento de todas as chaves do grupo.

Tabela 2.2: Esquemas de gerenciamento de chaves

Tipo	Escalável	Não-escalável
Controle de grupo	Wong et al. (1997)	Chiou; Chen (1989)
	Caronni et al. (1998)	Gong; Shacham (1994)
	Balenson et al. (1999)	Harney; Muckenhirn (1997)
	Canetti et al. (1999)	Dunigan; Cao (1997)
	Chang et al. (1999)	Blundo et al. (1998)
	Wallner et al. (1999)	Poovendran et al. (1998)
	Waldvogel et al. (1999)	Chu et al. (1999)
	Banerjee; Bhattacharjee (2001)	Wallner et al. (1999)
	Eskicioglu; Eskicioglu (2002)	Scheikl et al. (2002)
Controle de sub-grupo	Mitra (1997)	
	Dondeti et al. (1999d)	
	Molva; Pannetrat (1999)	Ballardie (1996)
	Setia et al. (2000)	Briscoe (1999)
Controle de Membro	Hardjono et al. (2000)	
	Dondeti et al. (1999b)	Boyd (1997)
	Perrig (1999)	Steiner et al. (1997)
	Rodeh et al. (2000)	Becker; Willie (1998)
	Kim et al. (2000)	

- **Esquemas de sub-grupos distribuídos (Controle de sub-grupos):** Existe uma hierarquia de entidades gerenciadoras em que cada uma das entidades é responsável pelo gerenciamento da chave do seu sub-grupo.
- **Esquema distribuído (Controle de membros):** Não há nenhuma entidade controladora definida. Os próprios membros devem gerar as chaves e realizar o gerenciamento e o controle do grupo.

2.3.4 Estudo sobre esquemas de gerenciamento de chaves na literatura

A Tabela 2.2 lista alguns esquemas propostos na literatura classificados em seis categorias (Eskicioglu, 2003).

Os esquemas DEP (Dondeti et al., 1999d), Iolus (Mitra, 1997) e CTKM (Wong et al., 1997) têm sido comparados em experimentos baseados em análises como uso de rede e custo de criptografia dos usuários (Dondeti et al., 1999c).

Figura 2.16: Agrupamento periódico de eventos

2.3.5 Agrupamento de eventos

Em cenários com milhares de usuários, a constante entrada e saída de usuários pode comprometer o desempenho da rede, mesmo se utilizados algoritmos escaláveis $O(\log(n))$. Desta forma, são encontradas na literatura propostas que visam minimizar o envio contínuo de informações de controle para a renovação das chaves (Zhu et al., 2003).

Um esquema pode ser capaz de tratar n eventos em apenas uma única mensagem, otimizando, pois, as informações de cada evento de tamanho s , ou seja, se as mensagens fossem enviadas sequencialmente, teriam um tamanho total de $n \times s$, para uma mensagem de tamanho k , onde $s \leq k \leq n \times s$. Assim, espera-se uma redução de banda para o gerenciamento de chaves.

Existem duas formas básicas de agrupamento de eventos: agrupamentos periódicos ou Agrupamentos por lote. Podem-se utilizar esquemas de agrupamentos mistos, conforme se verá a seguir.

2.3.5.1 Periódico (*periodic*)

O envio é periódico, ou seja, ocorre em determinados intervalos de tempo, independentemente da entrada ou saída de usuários. A Figura 2.16 exemplifica o modelo de agrupamento por período. No exemplo, cada mensagem de controle acumula todos os eventos em um intervalo de $2 \times t$ e envia uma única informação contendo todos os eventos neste período.

Figura 2.17: Agrupamento por lote de evento

O agrupamento periódico tem como desvantagem a possibilidade de má aglomeração dos eventos. Na hipótese de agrupamento dos instantes t_0 , t_1 e t_2 , caso dois eventos ocorram em $t_1 - \Delta$ e $t_1 + \Delta$, onde Δ é um valor pequeno, duas mensagens com apenas um evento cada serão utilizadas, mesmo se houver dois eventos muito próximos.

2.3.5.2 Lote (*batch*)

Em um esquema de agrupamento por lote, acumulam-se n requisições e apenas depois de um determinado volume de eventos a mensagem é enviada.

A Figura 2.17 exemplifica o modelo de agrupamento. No exemplo, cada mensagem de controle é enviada após três requisições de entrada ou abandono.

O agrupamento por lote, tem como desvantagem a possibilidade de um usuário U aguardar muito tempo até a sua inclusão no grupo. Caso haja a requisição de um usuário U_1 , em que o próximo evento só irá ocorrer pelo usuário U_2 depois de um tempo t , considerando-se que o tempo t seja grande, o usuário U_1 deverá aguardar expressivo período de tempo antes de conseguir adquirir as chaves do grupo.

Figura 2.18: Agrupamento misto

2.3.5.3 Misto

Além do agrupamento periódico e por lote, um esquema de gerenciamento pode utilizar um agrupamento misto, ou seja, um agrupamento que não seja nem puramente periódico, nem, tampouco, puramente por lote.

A Figura 2.18 ilustra um agrupamento misto. No exemplo, quando ocorre um evento, o gerenciador espera a primeira ocorrência de dois eventos, alternativamente: (a) acumulação de três eventos ou (b) espera de um período t .

O agrupamento misto pode reunir as vantagens do agrupamento periódico e por lote, sem herdar os problemas encontrados nestes dois modelos.

3 CENÁRIOS, APLICAÇÕES E POLÍTICAS DE SEGURANÇA

Embora o conceito de *Multicast* seja o mesmo em quaisquer tipos de aplicações em que ele é utilizado, diferentes cenários podem apresentar características comportamentais completamente divergentes e, desta forma, afigura-se importante classificar e estudar os tipos de aplicações *Multicast* para compreender o tipo de política de segurança a ser utilizada.

Se uma política de segurança não se adequar ao ambiente, poderão surgir falhas de segurança, além de haver o comprometimento do desempenho global do sistema.

Neste capítulo analisar-se-ão os tipos de aplicações *Multicast* as quais podem ser um-para-muitos ou muitos-para-muitos, acentuando-se as diferenças entre elas e as implicações surgidas com o emprego de segurança.

Também serão estudados os tipos de autenticação e a sua importância durante a especificação da política de segurança em aplicações *Multicast*.

3.1 Cenários *Multicast* seguros

Uma conexão *Unicast* é uma comunicação ponto-a-ponto entre dois *hosts*, ou seja, entre um remetente e um destinatário. Define-se uma comunicação *Unicast* como comunicação um-para-um (1-1): “um envia para um”.

Quando se utiliza uma conexão *Multicast*, esta poderá se classificar em função da forma de sua transmissão. As formas de transmissão podem ser (Edwards et al., 2002, p. 20):

- Comunicação um-para-muitos (“1-n”, *one-to-many*, *non-interactive* ou SSM(*Source Specific Multicast*)).
- Comunicação muitos-para-muitos (“n-n”, *many-to-many*, *interactive* ou ASM(*Any Source Multicast*)).

Figura 3.1: Transmissão *Multicast* um-para-muitos (1-n)

3.1.1 Comunicação um-para-muitos (1-n)

A comunicação um-para-muitos (1-n) é tipicamente uma aplicação na qual um servidor dedicado de dados, vídeos ou áudios envia o conteúdo para um grupo *Multicast* de até centenas de milhares de usuários, ou seja, uma única fonte é responsável por todo o tráfego do grupo.

A Figura 3.1 apresenta exemplo de uma comunicação um-para-muitos em um cenário composto por um servidor de dados transmitindo informações para n usuários. A flecha grande representa o fluxo dos dados entre o servidor e os membros, sendo que os *hosts* com borda grossa são os membros do grupo *Multicast* e os nós R são os roteadores do sistema.

Uma aplicação típica com transmissão um-para-muitos é a WebTV. Nesta, existe um servidor de imagens e múltiplos usuários assistindo ao vídeo. Não existe nenhuma transmissão por parte dos membros, apenas recepção.

Figura 3.2: Transmissão *Multicast* muitos-para-muitos (n-n)

3.1.2 Comunicação muitos-para-muitos (n-n)

A transmissão muitos-para-muitos (n-n) é a comunicação através da qual todos os membros podem receber e transmitir as informações concomitantemente, não há um servidor de dados dedicado e o tráfego da rede é formado pelo montante da transmissão de cada membro.

A Figura 3.2 exemplifica um cenário muitos-para-muitos. Cada usuário representado com borda grossa pertence ao grupo *Multicast* e pode tanto enviar, quanto receber dados.

Em um cenário muitos-para-muitos podem também ser estabelecidos privilégios para cada membro. Por exemplo, um Membro M_1 é capaz de enviar e receber dados, enquanto que um outro Membro M_2 pode apenas receber dados.

Um exemplo típico de cenário n-n é o sistema utilizado em videoconferências: diversos usuários transmitem e recebem informações multimídia concorrentemente.

3.1.3 Diferenças entre cenários um-para-muitos e muitos-para-muitos

Geralmente, uma aplicação 1-n é utilizada por aplicações com transmissão massiva de dados e contém muito mais usuários do que uma aplicação n-n. Tome-se como exemplo uma aplicação WebTV (1-n) e uma videoconferência (n-n). A WebTV pode conter centenas de milhares de telespectadores, enquanto que a videoconferência possui tipicamente algumas dezenas de usuários.

Embora os cenários 1-n e n-n sejam similares no aspecto de coletividade da informação, as construções diferem sob diversos aspectos. Tipicamente, no primeiro cenário (1-n) há o problema de excesso de membros, daí porque a distribuição e o gerenciamento das chaves de criptografia tornam-se problemas no consumo de banda. Em contrapartida, o segundo cenário (n-n) apresenta geralmente número modesto de usuários, porém, a fonte de transmissão — antes única — agora é compartilhada por todos os membros. A complexidade para realizar o roteamento dos datagramas devido às possíveis combinações de caminhos para cada uma das fontes torna-se substancialmente significativa. Tipicamente, cada roteador deverá conter o par de informação fonte-grupo para encaminhar cada datagrama.

Além da complexidade de roteamento, cuidados com a segurança devem ser relevados em virtude do roteamento conter diversas fontes.

Ainda, em cenários 1-n só existe uma fonte e, desta forma, a autenticidade dos dados é facilmente obtida por meio do uso de chaves assimétricas. Já em um cenário n-n, pode ser necessária a autenticação de origem, de acordo com a política de segurança do grupo. Trata-se de processo que exige a incorporação de novos módulos para atingir não só a proteção dos dados (privacidade), mas também a autenticidade da autoria das informações veiculadas.

A Seção 3.2 abordará as políticas de segurança que exigem autenticação a partir do estudo e da classificação dos casos mais comuns em grupos *Multicast* seguros.

Para maiores informações sobre roteamento *Multicast*, ver Apêndice B.

3.2 Autenticação em redes *Multicast*

Uma das premissas de segurança é a autenticidade. Embora não seja o foco desta dissertação o estudo dos mecanismos de autenticação em grupos *Multicast*, este tópico explora o funcionamento de autenticação em conexões *Multicast* seguras.

Primeiramente, é necessário explorar os níveis de autenticação. A autenticação pode ser necessária tanto no que concerne aos usuários, quanto no que pertine aos dados (Hardjono; Dondeti, 2003).

3.2.1 Autenticação de usuário

A autenticidade do usuário é realizada no momento em que um usuário ingressa no sistema. Neste momento, deve haver a comprovação de sua legitimidade perante o grupo e, caso seja genuíno, inicia-se processo de união ao grupo *Multicast*.

Uma política de segurança pode estabelecer diversos tipos e níveis de autenticação de usuários, a qual pode ser classificada como autenticação centralizada ou distribuída.

3.2.1.1 Autenticação Centralizada

Em um processo de autenticação centralizada existe um servidor de autenticação, por exemplo, o GSA (*Group Security Association*) (Hardjono; Dondeti, 2003, pp. 82–85) cuja função é autenticar e autorizar um membro dentro de um grupo e prover seus respectivos privilégios.

Como exemplo de sistema de autenticação centralizada em uma aplicação 1–n, supõe-se um cenário conforme apresentado na Figura 3.3. Quando um membro deseja ingressar no grupo *Multicast*, inicia-se comunicação ponto-a-ponto segura com o servidor de dados (destacado com borda dupla).

O membro obtém um certificado digital do servidor de dados e, a partir deste certificado, acessa o servidor de autenticação (destacado em borda pontilhada). Também através de uma conexão ponto-a-ponto segura, verifica-se a autenticidade do servidor de dados.

Figura 3.3: Exemplo de cenário *Multicast* com autenticação

Supondo-se que haja a confirmação da legitimidade, o servidor de dados também obtém o certificado digital do novo membro e verifica a sua autenticidade no servidor de autenticação, novamente a partir de comunicação ponto-a-ponto segura.

Enfim, supondo-se que todas as partes sejam entidades legítimas, procede-se da seguinte forma: o servidor envia os dados de renovação de chave para o grupo e, então, envia a nova chave do grupo para o novo membro através de uma conexão segura.

3.2.1.2 Autenticação Distribuída

Em uma autenticação distribuída não há um servidor de autenticação. A autenticação do grupo é realizada pelos próprios membros que compõem este grupo. Trata-se de modelo utilizado geralmente em redes tipo *ad hoc*.

O processo de autenticação distribuída apresenta diversas falhas de segurança por não haver nenhuma entidade confiável.

3.2.2 Autenticação de dados

De acordo com a política de segurança estabelecida por aplicação do usuário, não é necessária apenas a privacidade do grupo, mas também faz-se imprescindível a autenticidade dos dados transmitidos. Neste caso, cada datagrama deve prover uma assinatura de origem, ou seja, o remetente da informação deve provar a sua autoria para que todos os destinatários possam verificar a legitimidade dos dados.

Em um grupo seguro com autenticação de dados, os mesmos devem ser privados, íntegros e, ademais, autênticos.

Finalmente, incumbe ressaltar a acerca da autenticação de dados. Numa política de segurança podem ser estabelecidos diversos níveis de autenticação de dados, que pode ser classificada como autenticação de grupo e autenticação de origem (Hardjono; Dondeti, 2003, p. 45).

3.2.2.1 Autenticação de grupo

A autenticação de grupo é utilizada quando uma aplicação precisa apenas verificar que a informação veiculada originou-se de algum membro do grupo (Canetti et al., 1999). Neste caso, não é necessário autenticar o membro que enviou o dado.

A autenticação de grupo é facilmente alcançada, pois em um grupo *Multicast* seguro existe uma chave compartilhada entre todos os membros, os quais conhecem, com exclusividade, o segredo. As informações enviadas, então, podem ser simplesmente criptografadas com esta chave de grupo a partir da utilização de algum modo de operação que suporte privacidade, integridade e autenticidade ao mesmo tempo. É o caso, por exemplo, do HMAC (Krawczyk et al., 1997).

Note-se que a autenticação de grupo só garante que as mensagens transmitidas sejam legítimas, mas não é possível verificar quais dos membros deste grupo enviaram, de fato, determinadas mensagens. Não há, por conseguinte, possibilidade de se aferir a autenticidade da origem.

3.2.2.2 Autenticação de origem

A realização da autenticação de origem é necessária quando a política de segurança do grupo exige que cada remetente comprove a autoria de todas as mensagens enviadas. Não basta que os remetentes pertençam ao grupo, eles também devem autenticar os seus dados, possibilitando que todos os destinatários possam verificar a legitimidade da informação.

Este tipo de autenticação exige uma elaboração mais sofisticada, comparativamente à autenticação de grupo.

A autenticação tratada nesta sub-seção pode ser aplicada de duas formas, quais sejam, autenticação por pacote e autenticação por bloco.

Autenticação por pacote

A autenticação por pacote consubstancia forma de prover a autenticação de origem através da assinatura digital de cada pacote enviado.

É preciso ressaltar que este modelo é exaustivo ou, até mesmo, computacionalmente inviável devido ao tempo de processamento para gerar e verificar cada assinatura digital presente em cada pacote, uma vez que a assinatura digital utiliza chaves assimétricas, computacionalmente muito lentas.

Autenticação por bloco

Em uma autenticação por bloco todo o conteúdo gerado pelo remetente é submetido a uma função *hash*. O valor desta função deve ser enviado após a transmissão de todo o conteúdo e o remetente assina digitalmente apenas esta última mensagem contendo o valor do *hash* das mensagens anteriores. Todos os destinatários devem fazer o caminho inverso: computar o valor *hash* e comparar ao valor declarado e assinado pelo remetente. A função *hash* é tipicamente 1000 vezes mais rápida que uma criptografia assimétrica (Hardjono; Dondeti, 2003, p. 56).

A autenticação por blocos apresenta diversos problemas, não obstante reduza drasticamente o processamento computacional para a verificação da assinatura digital.

Dentre os principais problemas, o *Multicast* utiliza UDP como mecanismo de transporte e, desta forma, não há garantia de que todas as informações tenham sido remetidas. Ademais, mesmo que tenham sido remetidas, não há garantia de ordem. Ainda, um membro malicioso pode enviar um pacote falso apenas para os demais destinatários computarem e invalidarem a autenticação (Hardjono; Dondeti, 2003, pp. 49–50).

Existem diversas propostas para autenticação de blocos de dados *Multicast* na literatura, todas formuladas para melhorar a performance e robustez da autenticação por bloco. Exemplos destas propostas são o *Star Hashing* e o *Tree Hashing* (Merkle, 1989).

3.2.3 Autenticação Progressiva

Autenticação progressiva é uma técnica baseada em cascata de *hash* (*hash chaining* (Genaro; Rohatgi, 1997)). Estes algoritmos têm como objetivo reduzir a quantidade de verificações de autenticidade, fazendo com que um datagrama $t - 3$ autentique o datagrama $t - 2$, o qual, por seu turno, autentica o datagrama $t - 1$ e assim por diante.

A Figura 3.4 apresenta um exemplo simples de autenticação progressiva. O processo inicia-se com a geração de um valor aleatório K_t . A partir do K_t , aplica-se uma função “f” *one-way* (comumente por convenção prática, usa-se funções *hash*), gerando o valor K_{t-1} . Novamente, aplica-se uma função “f” em K_{t-1} , gerando o K_{t-2} . Tal processo é repetido n vezes. Na figura ora em análise demonstrou-se a realização de seis passos.

Pela Figura 3.4, anexa-se K_{t-6} na primeira mensagem e o remetente assina a mensagem digitalmente. Os próximos pacotes terão anexados os valores K_{t-5} , K_{t-4} , K_{t-3} , K_{t-2} , K_{t-1} e K_t , respectivamente. Desta forma, a próxima mensagem é autenticada pela mensagem anterior, pois só é possível gerar o valor *hash* da mensagem anterior a partir do conhecimento do valor do *hash* da mensagem atual.

O modelo de Autenticação Progressiva, ou *Hash Chaininig*, é muito eficiente, pois somente há uma única operação de criptografia assimétrica (para computar a assinatura digital da primeira mensagem). Conforme já mencionado anteriormente, o cálculo de uma função *hash* é tipicamente 1000 vezes mais rápido do que um algoritmo de chaves públicas (Hardjono; Dondeti, 2003, p. 56) e o *overhead* incorporado em cada mensagem é mínimo, uma vez

Figura 3.4: Exemplo de autenticação progressiva

que o tamanho de um valor de *hash* usualmente não ultrapassa 20 *Bytes* (referente ao SHA-1 (Eastlake; Jones, 2001) de 160 *bits*).

Outra importante vantagem é a tolerância do sistema a falhas: a perda de um pacote não compromete a verificação da autenticação dos outros pacotes recebidos, pois o valor esperado pelo pacote faltante é obtido pela função *hash* do próximo pacote.

Na literatura, diversos esquemas eficientes utilizam variações deste mecanismo, como o EMSS (Perrig, 2000a), *Augmented Authentication Chain* (Golle; Nodadugu, 2001), Piggy-backing (Miner; Staddon, 2002) e TESLA (Perrig, 2000b), este último baseado em MAC¹ (*Message Authentication Codes*).

3.3 Política de Segurança e Desempenho

A Política de Segurança deve ser adequada às necessidades do grupo, caso contrário, podem haver problemas com desempenho. Os itens a seguir apontam os principais problemas ocasionados por uma política de segurança inadequada à aplicação.

¹O MAC é utilizado para autenticar uma mensagem. O algoritmo MAC tem como entrada uma chave secreta e uma mensagem de comprimento arbitrário e, como saída, o valor MAC. O MAC provê tanto a integridade, quanto a autenticidade da informação.

3.3.1 Algoritmo de Criptografia

Dependendo do ambiente, a complexidade dos algoritmos utilizados para a criptografia deve ser mensurada a partir da ponderação entre a segurança e as limitações de cada um dos membros que compõem o grupo.

Como exemplo, se uma aplicação atinge dispositivos portáteis de baixo poder de processamento como, por exemplo, PDAs e *SetTopBox*, os algoritmos utilizados não devem ser computacionalmente complexos, caso contrário, a taxa de processamento pode tornar-se o gargalo do sistema.

3.3.2 Formas de Criptografia

Além da escolha de um algoritmo de criptografia eficiente para o ambiente, é necessário compreender a política de segurança do grupo e o conteúdo trafegado.

Uma forma de reduzir computacionalmente o uso da criptografia em aplicações multimídia é o uso de criptografia parcial. Caso a aplicação seja destinada ao envio de vídeos, a criptografia pode ser aplicada apenas em porções específicas do conteúdo, não em todo o dado.

Caso uma aplicação envie vídeos compactados com MPEG-2, poderá criptografar apenas blocos fundamentais para a reconstrução da imagem ou áudio, como as tabelas de quantização e blocos de DCT (*Discrete Cosine Transform*). Sem estas partes, nenhum usuário é capaz de visualizar a transmissão, mesmo conhecendo partes do conteúdo não criptografado.

3.3.3 Autenticação

Como visto na Seção 3.2 existem diversos níveis de autenticação. Caso a aplicação não exija alto nível de segurança, podem ser utilizados níveis mais moderados de autenticação como, por exemplo, autenticação de grupo somente.

Mesmo se considerada a necessidade de autenticação de origem, devem ser escolhidos esquemas que melhor se adequem ao ambiente. Por exemplo, um grupo que utilize conexão *Multicast* em uma rede hostil deve utilizar mecanismo de autenticação robusto e tolerante a falhas para evitar o reenvio de informações relativas à autenticação e, por outro lado, se a aplicação utiliza *Multicast* confiável, um mecanismo de autenticação menos robusto pode ser escolhido para diminuir o *overhead* do sistema.

3.3.4 Gerenciamento de Chaves

O esquema de gerenciamento de chaves deve adequar-se ao grupo. Como exemplo, caso o grupo seja grande, espalhado por todo o mundo e com centenas de milhares de usuários, o gerenciador de chaves deve ser mais eficiente com a utilização de modelos de controle de sub-grupo (visto na Seção 2.3.2) devido à capacidade de redução do grupo em pequenos sub-grupos, além da não apresentação da característica “1 afeta n”.

Por outro lado, em ambientes como redes de sensores, a divisão de sub-grupos pode sobrecarregar os nós intermediários confiáveis devido ao baixo poder de processamento dos sensores. Nestes casos, é preferível um gerenciador de chaves baseado em controle de grupo. Fatores como falta de confiabilidade nos *hosts* também sugerem uma melhor eficiência em esquemas de gerenciamento baseados em controle de grupo.

4 ESQUEMAS DE GERENCIAMENTO DE CHAVES

O Capítulo 2 definiu *Multicast*, fazendo uma introdução quanto às técnicas de gerenciamento de chaves de grupo, classificando-as.

Este capítulo apresentará estudo de alguns esquemas de gerenciamento de chaves *Multicast* conhecidos na literatura. Serão analisados quatro esquemas, sendo dois de controle de grupo e dois de controle de sub-grupo (ver Seção 2.3.2).

Os esquemas avaliados neste capítulo serão:

- **Gerenciador centralizado de Grupo:** CTKM (Wong et al., 1997) e EBS (Morales et al., 2003)
- **Gerenciador por controle de Sub-Grupo:** lolus (Mittra, 1997) e DEP (Dondeti et al., 1999d)

A escolha do CTKM e do EBS justifica-se em razão destes representarem os dois paradigmas quanto ao gerenciamento de chaves centralizadas. O lolus e o DEP, embora tenham sido propostos em 1997 e 1999, respectivamente, são os protocolos de controle de sub-grupo mais citados na literatura, razão pela qual também serão explorados.

Ao final deste capítulo, na Seção 4.5, apresentar-se-á tabela comparativa entre os esquemas avaliados (Tabela 4.4).

4.1 CTKM

O modelo de hierarquia de árvore CTKM (Wong et al., 1997) cria e armazena uma coleção de chaves em uma árvore hierárquica de forma a possibilitar a redução da banda necessária nas operações *join*, *leave* e *rekey*.

O CTKM é a base dos principais esquemas de gerenciamento de chaves encontrados na literatura. CKMSS (Eskicioglu et al., 2003) consubstancia exemplo de algoritmo baseado no CTKM, porém com algumas melhorias. Esta dissertação estudará apenas o CTKM.

Figura 4.1: Estrutura de árvore de chave (CTKM)

4.1.1 Operações de gerenciamento

Para a compreensão e entendimento do esquema de gerenciamento baseado em árvores de chaves, especificamente o CTKM, supõe-se um cenário com nove membros (M1..M9). Considera-se, no exemplo, que cada membro já foi autenticado e já tem a posse de suas respectivas chaves.

Gera-se uma estrutura de hierarquia de chaves em que cada membro figura como uma folha desta árvore. A Figura 4.1 exemplifica a estrutura.

O conjunto de chaves que compõe o sistema é: {K1-9, K123, K456, K789, K1, K2, K3, K4, K5, K6, K7, K8, K9}. O conhecimento das chaves para cada membro segue da raiz até a folha onde o membro se encontra, percorrendo todas as ramificações. Como exemplo, o Membro 5 tem a posse das chaves K1-9, K456 e K5. O gerenciador de chaves é o único conhecedor de todas as chaves do sistema.

A única chave comum para todos os usuários é a chave raiz, no exemplo, chamada de K1-9, a qual será compartilhada por todo o grupo e utilizada para criptografar todos os dados veiculados.

Supondo-se que $(A)_B$ representa o conteúdo A criptografado com a chave B, as próximas seções apresentam o processo realizado em cada união (*join*) e em cada abandono (*leave*) de usuários, além do método para realizar a renovação de chave (*rekey*).

Figura 4.2: Estrutura de árvore de chave após o abandono do Membro 9 (CTKM)

4.1.2 Processo de abandono (*leave*)

Prosseguindo com o esquema apresentado na Figura 4.1, quando o Membro 9 deixa o grupo, deve perder o direito de acesso ao conteúdo dos dados transmitidos posteriormente a sua saída para manter a confidencialidade temporal do grupo e, desta forma, a chave compartilhada (no exemplo, a K1-9) deve ser renovada entre os demais membros remanescentes do grupo. Destarte, o gerenciador de chaves deve enviar para o grupo a nova chave K1-8 de forma que apenas os usuários habilitados sejam capazes de decodificar esta informação e obter a nova chave gerada para o grupo. O Membro 9 deve ser incapaz de decodificá-la. Também é preciso trocar todas as demais chaves que o Membro 9 compartilha com os membros de seu sub-grupo. No caso, a chave K789 é compartilhada com os membros 7 e 8 e, portanto, deve ser trocada para K78. A chave K9 será simplesmente descartada, uma vez que se trata da chave pessoal do Membro 9.

A nova árvore de chaves mantida pelo gerenciador é apresentada na Figura 4.2. De acordo com a figura, observa-se que a chave K789 foi substituída pela chave K78 e a chave do grupo K1-9 foi substituída pela chave K1-8.

Após o gerador de chaves atualizar todas as chaves conhecidas e compartilhadas pelo Membro 9, deve informar os demais usuários acerca das novas chaves compartilhadas pelo grupo. Para isso, o gerenciador deve percorrer todo o caminho da raiz (chave K1-9) até o último nó com ramificação (chave K789), enviando cada uma destas novas chaves criptografadas com a chave filha deste nó.

No exemplo, o pacote com as informações necessárias para a renovação das chaves do grupo é assim representado:

- $((K1-8)_{K123}, (K1-8)_{K456}, (K1-8)_{K78}, (K78)_{K7}, (K78)_{K8})$

Verificando-se o manuseamento desta mensagem entre os membros, estudam-se três casos: a decodificação pelo Membro 8, pelo Membro 2 e, finalmente, pelo Membro 9 (que deverá ser excluído).

- Membro 8 (tem as chaves K1-9, K789 e K8):

- $(K1-8)_{K123}$, $(K1-8)_{K456}$, $(K1-8)_{K78}$, $(K78)_{K7}$, $(K78)_{K8}$
- $(K1-8)_{K123}$, $(K1-8)_{K456}$, $(K1-8)_{K78}$, $(K78)_{K7}$, K78
- $(K1-8)_{K123}$, $(K1-8)_{K456}$, K1-8, $(K78)_{K7}$, K78
- K1-8, K78

O Membro 8 tem a chave K8 e, desta forma, consegue decodificar a nova chave K78. Com a chave K78, por sua vez, é possível decodificar a nova chave do grupo K1-8.

- Membro 2 (tem as chaves K1-9, K123 e K2):

- $(K1-8)_{K123}$, $(K1-8)_{K456}$, $(K1-8)_{K78}$, $(K78)_{K7}$, $(K78)_{K8}$
- K1-8, $(K1-8)_{K456}$, $(K1-8)_{K78}$, $(K78)_{K7}$, $(K78)_{K8}$
- K1-8

O Membro 2 obteve somente a chave do grupo K1-8, pois não pertence ao sub-grupo da chave K78 (antiga K789).

- Membro 9 (tem as chaves K1-9, K789 e K9):

- $(K1-8)_{K123}$, $(K1-8)_{K456}$, $(K1-8)_{K78}$, $(K78)_{K7}$, $(K78)_{K8}$

O Membro 9 não consegue decodificar nenhuma das informações e, desta forma, deixa de ter acesso à nova chave K1-8, perdendo, por conseguinte, o acesso aos dados do grupo.

4.1.3 Processo de União (*join*)

Seguindo o mesmo exemplo apresentado pela Figura 4.2, supõe-se que o Membro 9 queira retornar ao grupo. Se autenticado com sucesso, o gerenciador do grupo gera uma chave aleatória K9 e a compartilha com o novo Membro 9 através de uma conexão ponto-a-ponto segura imediatamente após o processo de autenticação.

Após a autenticação do membro e a geração de uma chave K_9 conhecida pelo Membro 9 e pelo gerenciador de chaves, este deve gerar as novas chaves K_{1-9} e K_{789} do grupo, substituindo as antigas K_{1-8} e K_{78} , respectivamente. Este processo é necessário para manter a confidencialidade temporal do grupo.

A composição da nova árvore de chaves é novamente apresentada pela Figura 4.1. Importante ressaltar que as chaves K_{1-9} e K_{789} , no processo de união da seção anterior, não têm nenhum vínculo com as chaves K_{1-8} e K_{78} apresentadas nesta seção. Apenas mantêm-se os mesmos nomes para facilitar a explicação.

Diferentemente do processo de abandono, não há necessidade de criptografar as novas chaves com as chaves filhas, basta a atualização das novas chaves criptografando-as com suas chaves antigas. Também é necessário informar as novas chaves K_{1-9} e K_{789} para o novo Membro 9, uma vez que este não conhece as chaves antigas K_{1-8} e K_{78} .

O pacote necessário para realizar o processo de União, no exemplo, é o seguinte:

- $((K_{1-9})_{K_{1-8}}, (K_{789})_{K_{78}}, (K_{1-9}, K_{789})_{K_9})$

Exemplifica-se o manuseio deste pacote pelos membros através do exame de dois casos: a decodificação pelo Membro 8 e a decodificação pelo Membro 9.

- Membro 8 (tem as chaves K_{1-8} , K_{78} e K_8):
 - $(K_{1-9})_{K_{1-8}}, (K_{789})_{K_{78}}, (K_{1-9}, K_{789})_{K_9}$
 - $K_{1-9}, K_{789}, (K_{1-9}, K_{789})_{K_9}$
 - K_{1-9}, K_{789}

O Membro 8, em razão de conhecer as antigas chaves K_{1-8} e K_{78} , consegue obter as novas chaves K_{1-9} e K_{789} .

- Membro 9 (tem a K_9):
 - $(K_{1-9})_{K_{1-8}}, (K_{789})_{K_{78}}, (K_{1-9}, K_{789})_{K_9}$
 - $(K_{1-9})_{K_{1-8}}, (K_{789})_{K_{78}}, K_{1-9}, K_{789}$
 - K_{1-9}, K_{789}

O Membro 9 não conhece as antigas chaves K_{1-8} e K_{78} , mas consegue obter as novas chaves do grupo por ter ciência da chave K_9 .

4.1.4 Renovação de chave (*rekey*)

Para atualizar a chave do grupo periodicamente basta a seguinte informação:

- $((K1-9')_{K123}, (K1-9')_{K456}, (K1-9')_{K789})$

Todos os membros do grupo têm as chaves K123 ou K456 ou K789, portanto, todos os membros conseguem obter a nova chave K1-9' do grupo.

4.2 EBS

Exclusion Basis System (EBS), ou Sistema Baseado em Exclusão, foi proposto por Linda Morales no 36º Congresso Internacional do Hawaii (HICSS), em 2003 (Morales et al., 2003).

O EBS foi o primeiro esquema de gerenciamento de chaves a utilizar combinações exclusivas de chaves para realizar a renovação de chaves de grupo de forma escalável.

Foram realizados diversos estudos e elaborados aperfeiçoamentos para o EBS (Eltoweissy et al., 2004; Samuel T. Redwine; Madison, 2004), assim como algumas variações, como o CKDS (Moharrum et al., 2004). Nenhuma destas variações será abordada nesta dissertação.

No Capítulo 5 será proposto um esquema chamado EBCK, baseado no esquema EBS.

4.2.1 Definição

Morales et al. (2003) definiu k , n e m como inteiros positivos, onde $1 < k$ e $m < n$. Um Sistema Baseado em Exclusão com dimensões (n, k, m) — denotado por $EBS(n, k, m)$ — é uma coleção de Γ sub-conjuntos de $[1, n] = \{1, 2, \dots, n\}$, de forma que para qualquer inteiro $t \in [1, n]$ seguem as seguintes propriedades:

1. t está no máximo em k sub-conjuntos de Γ ;

2. Há exatamente m sub-grupos, digamos $A_1, A_2, \dots, A_m$ em Γ , tal que $\bigcup_{i=1}^m A_i$ é $[1, n] - \{t\}$ (Noutras palavras, cada elemento t é excluído por uma união de exatamente m sub-conjuntos de Γ).

Como exemplo, supomos um sistema EBS(8,3,2) com a seguinte coleção de sub-conjuntos:

- $\Gamma = \{A_1 = \{5, 6, 7, 8\}, A_2 = \{2, 3, 4, 8\}, A_3 = \{1, 3, 4, 6, 7\}, A_4 = \{1, 2, 4, 5, 7\}, A_5 = \{1, 2, 3, 5, 6, 8\}\}$.

É possível, facilmente, verificar que cada inteiro $t \in [1, 8]$ está exatamente em três ($k = 3$) sub-grupos de Γ e cada inteiro t é excluído pela união de exatamente dois ($m = 2$) sub-conjuntos de Γ , como ilustrado abaixo:

- $[1, 8] - \{1\} = A_1 \cup A_2$
- $[1, 8] - \{2\} = A_1 \cup A_3$
- $[1, 8] - \{3\} = A_1 \cup A_4$
- $[1, 8] - \{4\} = A_1 \cup A_5$
- $[1, 8] - \{5\} = A_2 \cup A_3$
- $[1, 8] - \{6\} = A_2 \cup A_4$
- $[1, 8] - \{7\} = A_2 \cup A_5$
- $[1, 8] - \{8\} = A_3 \cup A_4$

Um Sistema Baseado em Exclusão com dimensões (n, k, m) representa uma situação em um grupo seguro com n usuários numerados de 1 a n , onde o servidor de chaves armazena chaves distintas para cada sub-conjunto em Γ . Se um sub-conjunto A_i esta em Γ , então, a chave A_i é conhecida por todos os usuários cujo número t apareça no sub-conjunto A_i .

Para cada $t \in [1, n]$ há m sub-conjuntos em Γ cuja união é $[1, n] - t$. Como exemplo, em um EBS(8,3,2), os usuários 5, 6, 7 e 8 (e nenhum outro) conhecem a chave A_1 .

4.2.2 Processo de abandono (*leave*)

Seguindo o exemplo EBS(8,3,2) apresentado anteriormente, sabemos que o usuário U_1 detém as chaves A_3 , A_4 e A_5 (pois apenas estes conjuntos contêm o valor $t = 1$). Desta forma, tão-somente estas chaves precisam ser trocadas. Sabendo-se que $[1, 8] - 1 = A_1 \in A_2$, então, duas mensagens criptografadas pelas chaves A_1 e A_2 são suficientes para distribuir as novas chave para os demais usuários autorizados.

Cada mensagem será criptografada com as chaves A_1 e A_2 e deverá ter as seguintes informações:

1. A nova chave S' do grupo
2. A nova chave A'_3 , criptografada com a antiga chave A_3
3. A nova chave A'_4 , criptografada com a antiga chave A_4
4. A nova chave A'_5 , criptografada com a antiga chave A_4

Em outras palavras, as duas mensagens enviadas para o grupo para excluir o Membro 1 são:

- **Servidor** $\rightarrow$ **Grupo**: $(S', (A'_3)_{A_3}, (A'_4)_{A_4}, (A'_5)_{A_5})_{A_1}$
- **Servidor** $\rightarrow$ **Grupo**: $(S', (A'_3)_{A_3}, (A'_4)_{A_4}, (A'_5)_{A_5})_{A_2}$

As duas mensagens permitem que todo o restante do grupo obtenha as novas chaves A'_3 , A'_4 e A'_5 , com exceção do usuário U_1 (dissidente), o qual não possui nenhuma das chaves A_1 e A_2 para a decodificação dos pacotes.

O EBS permite evitar o uso de dupla criptação (chave criptografando chave) para redução de banda. Para tanto, utiliza uma função de caminho unilateral (*one-way*) como, por exemplo, uma função *hash*, fazendo com que a geração de uma nova chave seja obtida por $A'_i = f(S, A_i)$, conforme apresentado por Chang et al. (1999).

Observa-se, por fim, que para um sistema EBS(23,11,12) são necessárias 12 mensagens para a renovação das chaves ou exclusão de membros.

Tabela 4.1: Conjuntos de chaves administrativas do EBS(10,3,2)

	N1	N2	N3	N4	N5	N6	N7	N8	N9	N10
A1	1	1	1	1	1	1	0	0	0	0
A2	1	1	1	0	0	0	1	1	1	0
A3	1	0	0	1	1	0	1	1	0	1
A4	0	1	0	1	0	1	1	0	1	1
A5	0	0	1	0	1	1	0	1	1	1

4.2.3 Construção do sistema EBS

Para qualquer k e m , seja $Canonical(k, m)$ a série canônica de todos os $\binom{k+m}{k}$ formas possíveis de sub-grupos com k elementos, o número n de usuários deve obedecer à expressão $\binom{k+m}{k} \geq n$, caso contrário, o sistema EBS deve ser redimensionado.

Como exemplo, supõe-se um sistema EBS(10,3,2). A Tabela 4.1 apresenta a distribuição das chaves de sistema para cada usuário (N1..N10). Na matriz, “1” significa que o usuário N_i , na correspondente coluna, tem a chave A_i da correspondente linha e “0”, caso contrário. Como definido no sistema EBS, cada usuário tem exatamente 3 chaves ($k = 3$), sendo necessárias 2 mensagens ($m = 2$) para excluí-lo, existindo $k + m = 5$ chaves A_i .

4.2.4 Processo de união (join)

Durante o processo de inclusão de um novo membro, este inicializa uma conexão segura ponto-a-ponto com o servidor de chaves, o qual deve associar uma identificação t para o usuário. Conforme a Tabela 4.1, supomos que existam 9 membros (N1..N9).

O gerenciador de chaves associa o novo membro $t = N10$. Nenhuma chave A_i precisa ser alterada, mas a chave de sessão S deve ser renovada para manter a confidencialidade temporal do grupo. Então, o gerenciador de chaves envia para o membro N10, através de uma conexão segura, as chaves A_3 , A_4 , A_5 , bem assim a nova chave de sessão S' .

Para distribuir a nova chave S' , a seguinte mensagem deve ser transmitida ao grupo:

- **Servidor** $\rightarrow$ **Grupo**: $(S')_S$

Todos os membros que conhecem a chave S (no caso, $N1..N9$) obtêm a nova chave S' do grupo, enquanto o novo Membro $N10$ recebe a chave S' pelo canal ponto-a-ponto seguro, junto com a A_i .

4.3 lolus

O protocolo lolus foi proposto por Mittra (1997) em um artigo publicado em ACM SIGCOMM, França. O lolus representou um dos mecanismos pioneiros escaláveis de gerenciamento de chaves para grupos *Multicast*.

4.3.1 Características

Mittra propôs um protocolo com as seguintes características:

1. **Escalabilidade:** Permite o uso do lolus em grandes grupos *Multicast*, com entradas e saídas dinâmicas de usuários;
2. **Robustez:** O *framework* deve ser apto a suportar interrupções (maliciosas ou acidentais) de segmentos da rede, minimizando o quanto possível as consequências desta ruptura;
3. **Flexibilidade:** O protocolo deve ser capaz de se adaptar a diversos modelos de segurança;
4. **Tecnologia:** O protocolo deve suportar diversos algoritmos de criptografias;
5. **Protocolo de Rede:** O *framework* deve ser independente do protocolo de rede utilizado (DVMRP, CBT, PIM, etc);
6. **Camada de protocolo:** Deve ser possível o emprego do protocolo em diversas camadas de rede.

As características supra expostas são estendidas para quase todas as propostas de gerenciamento de chave de grupo escaláveis presentes na literatura.

4.3.2 Funcionamento

A idéia do lolus é construir uma Árvore de Segurança (*Secure Distribution Tree*) composta por pequenos sub-grupos de usuários organizados em hierarquia, dividindo, assim, um grande grupo em diversos sub-grupos menores *Multicast*.

A escalabilidade é alcançada em virtude de cada sub-grupo ser relativamente independente, ou seja, cada sub-grupo tem a sua própria unidade *Multicast* e pode ser criado utilizando qualquer protocolo *Multicast*, como, por exemplo, o DVMRP ou PIM.

Cada sub-grupo tem a sua própria chave (K_{SGRP_i}), não uma chave global K_{GRP} . Assim, quando um membro ingressa ou sai do grupo, somente afeta o sub-grupo local. Como resultado, apenas a chave do sub-grupo K_{SGRP_i} precisa ser trocada. Desta forma, atinge-se a escalabilidade.

Esta arquitetura também beneficia a característica “1-afeta-n”, ou seja, a entrada ou saída de um usuário não proporciona atualização em todos os demais membros do grupo, o que é comum em sistemas centralizados com controle de grupo.

lolus adiciona dois tipos de entidades que administram e conectam os diversos sub-grupos. Estas entidades são:

- **Controlador de Segurança do Grupo (GSC – *Group Security Controller*):** Administra o sub-grupo raiz e todos os sub-grupos associados diretamente a este. O GSC é o responsável pela segurança do grupo como um todo.
- **Intermediador de segurança do grupo (GSI – *Group Security Intermediaries*):** Existe um intermediador de segurança do grupo por sub-grupo, o qual é responsável pela administração e gerenciamento dos membros deste sub-grupo. Genericamente, ele é chamado de Agente de Segurança de Grupo (GSA – *Group Security Agents*).

Figura 4.3: Exemplo de Árvore de Distribuição Iolus

A Figura 4.3¹ demonstra um exemplo desta arquitetura.

Os GSIs formam a hierarquia dos sub-grupos e o GSC mantém o controle de todos os sub-grupos. Todos os GSIs são servidores confiáveis e especiais autorizados a atuar pelo GSC ou por seus respectivos GSIs pais.

Os GSIs formam uma ponte entre os sub-grupos por receber os dados *Multicast* dos GSI pais ou GSC e retransmiti-los aos seus membros e aos GSI filhos (se houver).

Embora pareça que cada GSI precise decriptar e criptografar novamente os dados em cada transição de sub-grupo em razão de existirem diferentes chaves K_{SGRP_i} para cada GSI, demonstrar-se-á, nas próximas seções, que não é necessária a constante criptografia entre sub-grupos.

¹Figura retirada do artigo original (Mittra, 1997).

Modelo de pacote:

Figura 4.4: Mensagem de União (lolus)

4.3.3 Visão Operacional

4.3.3.1 Inicialização

A inicialização do lolus começa com a inicialização do GSC. Uma lista de controle de acesso ACL (*Access Control List*) é transmitida para o GSC e utilizada para estabelecer os membros aptos a acessar o grupo.

Uma vez inicializados, os servidores GSI e os outros membros iniciam o processo de união.

4.3.3.2 Processo de união (*join*)

Para unir o grupo *Multicast*, o membro localiza o GSA mais próximo e requisita a sua entrada (*join*). O GSA checa o ACL para estabelecer se o usuário tem ou não permissão para entrar no grupo.

Pressupondo-se que sua entrada é aprovada, gera-se uma chave secreta para o novo membro ($K_{GSA-MBR}$) a fim de que seja provida comunicação segura em *Multicast* entre o GSA e o novo membro.

Subseqüentemente, o GSA altera a chave de seu sub-grupo de K_{SGRP_i} para K'_{SGRP_i} e faz com que a nova chave seja conhecida por todos os membros do sub-grupo, incluído o novo membro. Para esta tarefa, o GSA envia uma mensagem contendo a nova chave K'_{SGRP} criptografada com a antiga chave K_{SGRP} , além de uma outra mensagem com a nova chave K'_{SGRP} criptografada com a chave do novo membro $K_{GSA-MBR}$. Esta última mensagem é enviada por um outro canal ponto-a-ponto. A Figura 4.4 apresenta a estrutura do pacote para união de membros.

Modelo de pacote:

Cabeçalho	$(K'_{GRP})_{K_{GSA-MBR_1}}$	$(K'_{GRP})_{K_{GSA-MBR_2}}$	...	$(K'_{GRP})_{K_{GSA-MBR_n}}$
-----------	------------------------------	------------------------------	-----	------------------------------

Figura 4.5: Mensagem de Abandono (lolus)

4.3.3.3 Saída de usuário (*leave*)

Quando um membro deixa o grupo, o GSA precisa desabilitar este usuário para que o mesmo não tenha mais acesso aos futuros dados do grupo. Para renovar a chave, o GSA deve gerar e enviar para o grupo uma nova chave K'_{SGRP} criptografada individualmente com cada uma das chaves dos seus respectivos membros $K_{GSA-MBR_i}$, com exceção do membro que deixa o grupo. A Figura 4.5 exibe o formato deste pacote.

Embora dentro de um sub-grupo o gerenciamento de chaves não seja escalar (*Flat Scheme*), a escalabilidade é mantida por não existir nenhum sub-grupo significativamente grande. Novos sub-grupos são gerados conforme haja o crescimento do sistema.

4.3.4 Transmissão de dados

Um servidor não deve enviar um dado *Multicast* criptografado com a chave do seu sub-grupo K_{SGRP} , pois, se assim fosse, apenas este sub-grupo poderia decodificar a informação.

A forma mais simples de contornar este problema é fazer com que todos os SGIs decriptem e criptografem novamente para o seu sub-grupo cada pacote transmitido a partir da utilização da chave K_{SGRP_i} de seu sub-grupo. Entretanto, este método não é usual por promover repetidas decodificações e codificações dos dados, desperdiçando, por conseguinte, processamento e memória nos SGIs.

Para minimizar a carga de processamento de criptografia, o servidor de dados gera uma chave aleatória K e criptografa todo o conteúdo com esta chave aleatória K , a qual deve ser criptografada com a chave de seu sub-grupo K_{SGRP} e anexada aos dados transmitidos. Desta forma, o processo de criptografia e decriptografia repercute unicamente na chave aleatória K e não em todo o dado.

Figura 4.6: Fluxo de envio do lolus

A Figura 4.6² demonstra o fluxo de envio de um servidor de dados no esquema lolus, onde “S” é o servidor de dados.

4.4 DEP

Dual Encryption Protocol (DEP) foi proposto por Dondeti et al. (1999d,a) e concebido para prover segurança em grupos com transmissão um-para-muitos (1-n) em redes *Multicast*. O DEP, assim como o lolus, utiliza o controle de sub-grupos *Multicast* para prover a escalabilidade.

Cada sub-grupo é administrado por um gerente de sub-grupo, chamado de SGM (*Subgroup Manager*). O protocolo assume que cada SGM é um roteador ou *host* estável, ou seja, espera-se deste SGM um alto valor MTBF (*Mean Time Between Failures*).

O protocolo DEP distingue dois tipos de usuários: Membros e Participantes. Usuários Membros são aqueles que pertencem ao grupo *Multicast* e têm acesso aos dados veiculados pelo grupo, enquanto que os participantes são os SGMs que auxiliam na administração dos

²Figura retirada do artigo original (Mitra, 1997).

Tabela 4.2: Anotações e nomenclaturas para descrever o protocolo DEP

SGM	Gerenciador de sub-grupo (<i>Subgroup Manager</i>)
LS_i	Chave do sub-grupo local (<i>Local Subgroup</i>)
KEK_a	Chave de criptografar chaves do SGM _a (<i>Key Encrypting Key</i>)
DEK	Chave de criptografar dados (<i>Data Encryption Key</i>)
KU_x	Chave pública do usuário X (<i>Public-Key</i>)
KR_x	Chave privada do usuário X (<i>Private-Key</i>)
EP	Chave de criptografia pública do grupo
ES	Chave de criptografia privada do grupo
HV	Valor <i>hash</i> (<i>Hash Value</i>)

sub-grupos, porém, não têm acesso ao conteúdo das informações transmitidas. Com esta distinção, DEP permite que os SGMs manuseiem os sub-grupos *Multicast* sem ter acesso à informação.

O esquema provê um mecanismo de criptação dupla que utiliza duas chaves auxiliares. A primeira, é a chave de topo de nível (*Top Level Key*) utilizada pelo remetente para proteger as informações das chaves criptadas dos participantes (SGMs). Esta chave é chamada de KEK (*Key Encrypting Key*). A segunda chave é chamada de chave de sub-grupo local (*Local Subgroup Keys*), usada pelos SGM para criptografar as informações criptografadas dos membros do seu correspondente sub-grupo. Esta chave é chamada de LS (*Local Subgroup*). Ambas as chaves são do tipo chave pública, ou seja, utilizam criptografia assimétrica.

Tanto o remetente (*sender*) quanto os SGMs são responsáveis pelo controle de acesso do grupo. O remetente administra as chaves de criptografia dos dados. Cada SGM controla os membros de seu sub-grupo e também é responsável pela distribuição e gerenciamento das chaves do seu sub-grupo.

Dondeti et al. (1999d) utilizou as anotações apresentadas na Tabela 4.2 para descrever o algoritmo DEP. Referidas anotações serão utilizadas nas demais seções.

Figura 4.7: Exemplo de Árvore de Distribuição Segura (DEP)

Tabela 4.3: Componentes de um certificado DEP

Informação de autenticação	Autenticação da informação
Nome do <i>host</i>	Nome do grupo <i>Multicast</i>
Identificador do <i>host</i>	Identificador do grupo <i>Multicast</i>
Chave pública do <i>host</i>	Duração da autorização (tempo inicial e final)

4.4.1 Estrutura

A Figura 4.7³ ilustra exemplo de uma árvore de distribuição de chaves. Na figura, SGM₁ é um participante e não tem acesso aos dados do grupo, enquanto que o SGM₂ e o SGM₃ são membros, ou seja, têm acesso ao conteúdo do grupo.

Para grandes grupos, a lista de controle de acesso pode ser grande ou, ainda, ser modificada ao longo do tempo. O protocolo requer a obtenção, por todos os membros, de autorização com alguma certificadora e nesta autorização deve constar o tempo de permanência deste membro no grupo *Multicast*. Tanto o remetente quanto os SGMs devem verificar as permissões do novo membro.

A Tabela 4.3 apresenta os principais campos que compõem o certificado.

³Retirada da Figura 1 do artigo Dondeti et al. (1998).

4.4.2 Inicialização do sistema

Considera-se que todos os nós da árvore de grupo (Figura 4.7) com ramificações (não folhas), incluindo o remetente (*sender*), são gerenciadores de sub-grupo. Cada SGM é responsável pela geração de uma chave secreta que devem compartilhada com todos os membros do seu sub-grupo.

No exemplo apresentado na Figura 4.7, o SGM_1 deve compartilhar a chave do sub-grupo LS_1 com todos os membros filhos, ou seja, com o SGM_m e o A.

O remetente deve gerar uma chave de topo de nível KEK e uma chave do seu sub-grupo local LS. As chaves KEKs são usadas para cifrar as chaves DEK dos participantes. Uma chave KEK é gerada para cada um dos SGMs e distribuída para os membros pelo remetente.

No exemplo (Figura 4.7), o remetente (*sender*) gera três chaves KEKs: KEK_1 , KEK_2 e KEK_3 , as quais são distribuídas para os seus descendentes SGM_1 , SGM_2 e SGM_3 , respectivamente.

Todos os membros e participantes do grupo devem conhecer os seus ancestrais (nós pais) e cada SGM deve propagar esta informação para todos os seus futuros descendentes.

4.4.3 Entrada de usuários

Quando um novo *host* H entra no grupo *Multicast* seguro, este *host* envia uma mensagem para o grupo contendo o seu certificado e aguarda a primeira resposta positiva. Um SGM capaz de atender à requisição responde esta mensagem, verificando o certificado e decidindo se a requisição será aprovada ou negada.

Supondo-se que a requisição seja aceita, o *host* H acata a primeira resposta válida e a une ao sub-grupo do SGM que atendeu a sua requisição.

Mantendo o exemplo da Figura 4.7, se o SGM_m foi o primeiro a responder à requisição de H, o SGM_m deverá enviar a sua identidade, bem assim a lista das identidades de seus ancestrais, no exemplo, o SGM_m e o SGM_1 , seus ancestrais. Como o SGM_m atendeu a sua requisição, o *host* H entra no sub-grupo SGM_m . A Figura 4.8(a)⁴ representa este processo.

⁴Retirada do artigo Dondeti et al. (1998).

Figura 4.8: Comunicação do membro com o SGM

Seguidamente, o *host* H , conhecendo os seus ancestrais, envia para o remetente (*source*) o seu certificado, que responde com o certificado de autorização contendo a nova chave pública do grupo EP' , bem assim a chave KEK_1 (sub-grupo do SGM_1). Toda a mensagem é criptografada com a chave pública do usuário H ($(EP)_{KU_H}$), de forma que apenas o membro H possa acessar o conteúdo desta mensagem. A Figura 4.8(b) demonstra o processo.

Após todo este processo, o *host* H precisa conhecer a chave do grupo local LS_m . O SGM_m , para a consecução deste desiderato, envia para H a chave LS_m criptografada com a chave pública do usuário h ($(LS_m)_{KU_H}$)

A nova chave do grupo EP' é criptografada com a chave antiga do grupo EP ($(EP')_{EP}$) para que todo o restante do grupo tenha acesso à nova chave do grupo.

Finalmente, o gerenciador SGM do sub-grupo adiciona o *host* H na lista de membros e muda a sua chave de sub-grupo, criptando-a com a chave pública do *host* H e enviando-a (Figura 4.8(c)).

Separadamente, o SGM envia a nova chave do grupo LS' criptografada com a antiga chave LS , permitindo a atualização da chave para todos os demais usuários.

4.4.4 Comunicação Segura

Para o envio da informação, o remetente (*sender*) gera uma chave DEK (chave de criptografia de dados) e a envia para o grupo da seguinte forma:

- $((\text{DEK}, \text{HV})_{ES_{KEK_1}})_{ES_{LSs}}, ((\text{DEK}, \text{HV})_{ES_{KEK_2}})_{ES_{LSs}}, ((\text{DEK}, \text{HV})_{ES_{KEK_3}})_{ES_{LSs}}$

Onde:

- LSs é a chave do sub-grupo raiz (topo de nível)
- HV é o valor *hash* da chave DEK

Deste modo, cada um dos nós filhos do remetente consegue decriptar a primeira parte da mensagem, pois todos têm a chave ES_{LSs} . Cada um deles, por sua vez, cripta o termo referente ao seu sub-grupo com a chave LS de seu sub-grupo.

Como exemplo, o SGM_1 irá repassar a seguinte mensagem para o seu sub-grupo:

- $((\text{DEK}, \text{HV})_{ES_{KEK_1}})_{ES_{LS_1}}$

Todos os membros com a chave KEK de seus respectivos sub-grupos conseguem obter a chave de dados DEK. Cada membro deve comparar o *hash* da chave DEK com o HV para verificar a integridade.

Observe-se que os SGM são membros, não apenas participantes, de forma que possuem a chave KEK, razão pela qual conseguem obter também a chave DEK do grupo e acessar os dados trafegados pelo remetente.

4.4.5 Saída de usuário

Quando um membro do grupo *Multicast* sai do mesmo, seja por vontade própria, seja em virtude de ter expirado o seu certificado, ele não deve ser capaz de acessar os dados do grupo. Para isso, o correspondente SGM do seu sub-grupo muda a chave do grupo local, cifrando a nova chave com a chave pública de cada um dos membros remanescentes (*Flat Scheme*).

Assim, todos os membros deste sub-grupo têm a nova chave LS de seu sub-grupo, com exceção do usuário que foi excluído do grupo.

Revisando o exemplo apresentado na Figura 4.7, se o *host* K deixa o grupo, o administrador SGM_n muda a chave de seu sub-grupo enviando a nova chave LSs criptografada com as chaves públicas dos *hosts* I e J. A chave KEK não precisa ser alterada. A seguinte mensagem deve ser enviada para o grupo:

- $((LS')_{KU_I}, (LS')_{KU_J})$

Quando o usuário que deixa o grupo é um SGM, seja ele participante ou membro, realiza-se o mesmo processo, com a diferença de que todos os membros descendentes devem ser notificados, assim como os SGMs filhos. Cada usuário notificado deve, então, entrar em um outro SGM.

4.4.6 Atualização de chaves

O remetente e os SGMs devem renovar periodicamente as suas chaves para precaverem-se de ataques do tipo *eavesdropping*⁵. Para mudar a chave do grupo, o SGM segue o mesmo procedimento descrito na Seção 4.4.5, com a diferença de que não irá excluir nenhum membro durante o processo de criptação da chave LS com as chaves públicas de cada usuário.

⁵Eavesdropper – Usuário malicioso que intercepta informações de um outro usuário sem que este saiba. O termo se origina da situação consistente na espera, atrás da porta, por uma pessoa que escuta a conversa entre dois ou mais interlocutores, os quais não têm ciência de que estão sendo ouvidos

4.5 Resumo comparativo

Como breve resumo comparativo entre os quatro esquemas estudados neste capítulo, mais o modelo proposto EBCK que será apresentando no Capítulo 5, a Tabela 4.4 lista as características para cada um destes esquemas.

Tabela 4.4: Tabela comparativa (Iolus, DEP, CTKM, EBS e EBCK)

	Iolus	DEP	CTKM	EBS	EBCK
Esquema escalável	Sim	Sim	Sim	Sim	Sim
Tipo de controle	sub-grupo	sub-grupo	grupo	grupo	grupo
Uso de nós intermediários	sim (confiável)	sim (não confiável)	não	não	não
Escalabilidade em <i>join</i>	$O(1)$	$O(1)$	$O(\log(n))$	$O(1)$	$O(1)$
Escalabilidade em <i>leave</i>	$O(1)^*$	$O(1)^*$	$O(\log(n))$	$O(\log(n))$	$O(\log(n))$
Escalabilidade em <i>rekey</i>	$O(1)$	$O(1)$	$O(\log(n))$	$O(\log(n))$	$O(1)$
Suporte a agrupamento	Sim**	Sim**	Não	Não	Sim
1 afeta n	Não	Não	Sim	Sim	Sim

* Por ser baseado em controle de sub-grupo, a escalabilidade global é $O(1)$, porém, dentro do sub-grupo é $O(n)$.

** Embora não especificado pelos autores, todo esquema baseado em *Flat Scheme* tem suporte a agrupamento.

Note-se que as comparações mostradas na tabela entre os esquemas de gerenciamento de chaves são teóricas. No Capítulo 7 demonstram-se análises comparativas através de simulações destes esquemas de gerenciamento.

5 EBCK: UMA PROPOSTA PARA GERENCIAMENTO DE CHAVES

O Capítulo 4 apresentou quatro esquemas de gerenciamento de chaves (CTKM, EBS, Iolus e DEP) que permitem a inclusão de um mecanismo escalável de segurança em grupos *Multicast*. Este capítulo 5º apresentará uma proposta de gerenciamento de chaves elaborada a partir das pesquisas e análises realizadas para a elaboração desta dissertação, o esquema EBCK. Este capítulo introduzirá o conceito utilizado para atingir a escalabilidade e descreverá as funcionalidades do EBCK, seus benefícios e desvantagens.

No Capítulo 7, o EBCK será simulado e analisado conjuntamente com os quatro esquemas apresentados no capítulo anterior.

5.1 EBCK – *Exclusion-Based Combinatorial Key*

Propõe-se, neste trabalho, um algoritmo de gerenciamento centralizado de chaves baseado no mesmo conceito do esquema EBS (Morales et al., 2003) chamado *Exclusion-Based Combinatorial Key*. O EBCK foi concebido para realizar o gerenciamento centralizado de chaves compartilhadas visando à minimização do uso dos recursos de rede, assim como a minimização do processamento e uso de memória para o gerenciamento das chaves por parte dos usuários (membros do grupo).

Com a otimização do uso dos recursos de rede e atividades computacionais e por ser o EBCK um algoritmo controlador de grupo centralizado (não exige nós adicionais confiáveis), o mesmo tem por escopo, dentre outros, permitir a sua implementação de forma eficiente em ambientes de baixa capacidade, em especial, redes de sensores.

Diversos algoritmos como o EBS e seus derivados, por exemplo, o CKDS (Moharrum et al., 2004), vêm sendo empregados em redes de sensores (Eltoweissy et al., 2006; Chorzempa, 2006; RuiYing et al., 2005; Moharrum; Eltoweissy, 2005).

Redes de sensores são caracterizadas por possuírem limitação de memória e processamento, assim como capacidade de comunicação reduzida. O EBCK adequa-se às limitações impostas por este ambiente.

5.2 Definição

O EBCK é um gerenciador de chaves baseado em exclusão de chaves combinadas. O gerenciador EBCK incorpora uma variação da lógica de exclusão mútua proposta por Samuel T. Redwine; Madison (2004), porém adaptada e aperfeiçoada com a utilização da “Cobertura Mínima” para diminuição do uso da rede.

O EBCK também realiza otimizações utilizando funções *one-way* para minimizar o uso de banda e reduzir o processamento. O uso de funções *one-way* e a exclusão mútua de usuários já foram apresentados em Chang et al. (1999), porém, em esquemas baseados em árvores, como o algoritmo CTKM.

O sistema EBCK provê a segurança do grupo através de uma chave compartilhada S conhecida por todos os membros $\in U$. Esta chave pode ser simétrica ou assimétrica, dependendo da política de segurança e do modo de transmissão:

- **Simétrica:** Usa-se uma chave simétrica S quando se deseja permitir que todos os usuários enviem e recebam mensagens, ou seja, um esquema muitos-para-muitos. Porém, caso seja necessária a autenticação de origem, faz-se imprescindível a inclusão de algum mecanismo adicional de autenticidade, conforme visto na Seção 3.2. O EBCK não especifica nenhum método de autenticação de origem.
- **Assimétrica:** Usa-se uma chave assimétrica S quando se deseja realizar uma comunicação um-para-muitos cuja chave privada S_{Pr} seja mantida em segredo pelo remetente (servidor de dados). Desta forma, todos os membros que têm a posse da chave pública S_{Pb} podem acessar as informações, mas nenhum membro é capaz de enviar a mensagem apresentando-se como falso servidor de dados em razão de desconhecer a sua chave privada S_{Pr} . Cumulativamente, em razão da chave assimétrica ser tipicamente mais lenta que uma chave simétrica, recomenda-se criptografar os dados utilizando uma chave simétrica aleatória K , a qual deve ser criptografada com a chave privada S_{Pr} $((K)_{S_{Pr}})$ e enviada juntamente com os dados. Assim, os membros precisam apenas decriptar a chave K com a chave pública S_{Pb} , utilizando esta chave simétrica K para decriptar o restante da informação.

Além da chave compartilhada S , define-se um grupo de chaves de exclusão K com k elementos: $\{K_1, K_2, K_3, \dots, K_k\}$. Cada usuário $U_j \in U$ possui um sub-conjunto único de chaves $C_i \subset K$, cujo tamanho é c . Todos os conjuntos C de cada usuário devem ser exclusivos, ou seja, existe apenas uma combinação C_i para cada membro.

$k = 1$						1													
$k = 2$						1		1											
$k = 3$					1		2		1										
$k = 4$				1		3		3		1									
$k = 5$			1		4		6		4		1								
$k = 6$		1		5		10		10		5		1							
$k = 7$		1		6		15		20		15		6		1					
$k = 8$	1		7		21		35		35		21		7		1				

(a) Modelo Clássico

	$\frac{k}{2}-4$	$\frac{k}{2}-3$	$\frac{k}{2}-2$	$\frac{k}{2}-1$	$\frac{k}{2}$	$\frac{k}{2}+1$	$\frac{k}{2}+2$	$\frac{k}{2}+3$	$\frac{k}{2}+4$	$\frac{k}{2}+5$
$k=2$					1	1				
$k=4$				1	3	3	1			
$k=6$			1	5	10	10	5	1		
$k=8$		1	7	21	35	35	21	7	1	
$k=10$	1	45	120	210	252	252	210	120	45	1

(b) Modelo Reorganizado ($c = \frac{k}{2}$)

Figura 5.1: Triângulo de Pascal

Tendo k como quantidade total de chaves de exclusão e c , a quantidade de chaves administradas por cada usuário, define-se:

- O número total de chaves de exclusão k deve ser múltiplo de dois (quantidade par).
- O número de chaves de exclusão mantida por cada usuário (c) deve ser exatamente a metade da quantidade total de chaves combinadas k , ou seja, $c = \frac{k}{2}$.

Justifica-se a relação entre c e k em razão de ser a melhor proporção apta a gerar maior número de combinações possíveis para qualquer conjunto K com k chaves.

Verifica-se a assertiva anterior através da análise do Triângulo de Pascal¹ apresentado na Figura 5.1(a).

Pela Figura 5.1(b), observa-se que a maior quantidade possível de combinações ocorre quando $c = \frac{k}{2}$.

Como será demonstrado nas próximas seções, o gerenciador de chaves EBCK precisa de uma informação de tamanho $h + z \times c$ para excluir um membro, onde h é o tamanho do cabeçalho da mensagem e z é o tamanho do valor *hash*. Assim, a reação é ótima por permitir o maior número de usuários com o menor tamanho necessário para o gerenciamento das chaves.

¹O Triângulo de Pascal é um triângulo numérico infinito formado pelo Binômio de Newton $\binom{k}{c}$.

Tabela 5.1: Número de chaves e quantidade total de usuários

Número de chaves	Número total de usuários
6	20
8	70
10	252
12	924
16	12.870
20	184.756
26	10.400.600
32	601.080.390

5.2.1 Relação entre chaves de exclusão e usuários

Definida a relação entre a quantidade total de chaves de exclusão k e o número de chaves c mantida por cada membro, a seguinte fórmula é válida para obter o número total de usuários para um conjunto K de tamanho k :

$$\binom{k}{c} = \frac{k!}{c!(k-c)!} \rightarrow \binom{k}{c = \frac{k}{2}} = \frac{k!}{(k/2)!(k/2)!} = \frac{k!}{((k/2)!)^2}$$

Como exemplo, um sistema EBCK com 6 chaves de exclusão pode gerenciar até $\binom{6}{3} = \frac{6!}{3!(6-3)!} = \frac{720}{6 \times 6} = 20$ usuários.

A Tabela 5.1 correlaciona a quantidade total de chaves de exclusão com a quantidade total de usuários suportados pelo sistema.

5.3 Escalabilidade

Para verificar a escalabilidade do EBCK, ou seja, estabelecer uma relação entre o número de usuários e o volume necessário de informações para o gerenciamento das chaves, é preciso relacionar o número de usuários com o tamanho necessário das informações de renovação de chaves.

Considerando que o volume necessário do EBCK para o gerenciamento de chaves é proporcional ao c (o que será demonstrado nas próximas seções) e o número máximo de usuários n é definido como $n = \frac{(c \times 2)!}{(c!)^2}$, deseja-se encontrar uma função f em que seja válida a seguinte relação: $s = O(f(n))$, onde s é o volume de dados necessários para o gerenciamento.

Um crescimento de ordem $O(\log(n))$ significa que para atender n usuários, é necessário o envio de uma informação de tamanho $\log(n)$. Se o crescimento EBCK em relação ao crescimento logaritmo for constante, maior que zero e não infinito, para n tendendo ao infinito, pode-se dizer que o EBCK é definido também como $O(\log(n))$.

Definindo $l = \ln(n)$, onde $n = \frac{(c \times 2)!}{(c!)^2} = \text{número de usuários no modelo EBCK}$, a função $\frac{l}{c}$ representa a relação entre o crescimento de uma função logarítmica sobre o crescimento do EBCK relativo ao volume de dados necessários para o gerenciamento.

Caso o limite abaixo seja constante e maior que zero, o EBCK pode ser representado por $O(\log(n))$:

$$0 < \lim_{c \rightarrow \infty} \left(\frac{l}{c} \right) < \infty$$

Para uma informação de tamanho c em EBCK, é necessária uma mensagem de tamanho l em um sistema logarítmico. l é obtido por $\ln(n)$, onde n é o número de usuários. Para relacionar l com c , ambos devem atender ao mesmo número de usuários. Conhecendo a relação $l = \ln(n)$, é possível obter o n através de c , utilizando a fórmula $n = \frac{(c \times 2)!}{(c!)^2}$. Assim, é possível estabelecer a relação entre uma função logarítmica e o crescimento EBCK apenas dependente de c :

$$\begin{aligned} \frac{l}{c} &= \frac{\ln(n)}{c} = \frac{\ln\left(\frac{(c \times 2)!}{(c!)^2}\right)}{c} \\ &= \frac{\ln((c \times 2)!) - \ln((c!)^2)}{c} = \frac{\ln((c \times 2)!) - 2 \times \ln(c!)}{c} \end{aligned}$$

Para valores grandes, é verdadeira a seguinte expressão (função de Stirling):

$$\ln(n!) \approx n \cdot \ln(n) - n$$

Como $c \rightarrow \infty$, é possível utilizar a aproximação de Stirling:

$$\begin{aligned} \frac{\ln((c \times 2)!) - 2 \times \ln(c!)}{c} &= \frac{(2 \cdot c \cdot \ln(2 \cdot c) - 2 \cdot c) - 2 \times (c \cdot \ln(c) - c)}{c} \\ &= 2 \cdot \ln(2 \cdot c) - 2 - 2 \cdot \ln(c) + 2 = 2 \cdot \ln(2 \cdot c) - 2 \cdot \ln(c) = 2 \times \ln\left(\frac{2 \cdot c}{c}\right) = \ln(4) \end{aligned}$$

Portanto:

$$\lim_{c \rightarrow \infty} \frac{l}{c} = \ln(4) \approx 1,386$$

Resta, pois, demonstrada: a escalabilidade do EBCK é de ordem $O(\log(n))$.

5.4 Manutenção do grupo

Para representar o processo realizado pelo EBCK com o intuito de adicionar ou excluir membros ou renovar chaves compartilhadas, será utilizado exemplo EBCK com um número total de chaves de exclusão K de $k = 6$. Supõe-se uma distribuição de chaves de exclusão $C_i \subset K$ entre os usuários U_j dispostos de acordo com a Tabela 5.2.

O valor “1” significa que o usuário U_j desta linha tem a posse da chave de exclusão K_i de sua respectiva coluna, enquanto o valor “0” significa a ausência de chave combinada por este usuário. Por exemplo, o usuário U_3 tem a posse das chaves K_1 , K_4 e K_5 .

Observa-se que cada usuário tem exatamente três chaves ($c = \frac{6[-k]}{2} = 3$) e cada um possui um conjunto exclusivo C_i de chaves K_i .

5.4.1 Autenticação

O EBCK não emprega nenhum mecanismo específico de autenticação em sua definição. Qualquer aplicação que utilize o esquema EBCK para o gerenciamento de chaves deverá especificar e incorporar o seu próprio sistema de autenticação.

Tabela 5.2: Exemplo de distribuição de chaves EBCK

	K_1	K_2	K_3	K_4	K_5	K_6
U_1	1	0	1	1	0	0
U_2	1	0	1	0	0	1
U_3	1	0	0	1	1	0
U_4	0	1	1	0	1	0
U_5	0	1	0	1	1	0
U_6	0	0	1	0	1	1
U_7	0	1	0	1	0	1
U_8	1	1	0	0	0	1

Considera-se, nas próximas seções deste capítulo, que todos os membros, ao ingressar no processo de união, já estejam previamente autorizados e o servidor de chaves já conheça as chaves públicas individuais de todos os usuários.

5.4.2 Processo de união

Durante o processo de união de um usuário U_j , o gerenciador estabelece um conjunto $C_j \subset K$ onde $C_j \cap C = \emptyset$, ou seja, C_j contém um conjunto de chaves único dos demais membros existentes.

Supondo que existam apenas sete membros no grupo ($M_1..M_7$), quando o usuário U_8 entrar no grupo, o servidor deve selecionar um conjunto exclusivo de chaves para ser associado ao usuário.

Seguindo o exemplo da Tabela 5.2, o gerenciador de chaves elege um conjunto de chaves K_i para o novo usuário U_8 . No caso, o gerenciador seleciona as chaves K_1 , K_2 e K_6 .

O gerenciador de chaves gera um valor aleatório chamado de *Salt*. O *Salt* será utilizado para renovar todas as chaves do sistema, incluindo a chave compartilhada S . Para gerar as novas chaves, utiliza-se uma função *one-way*.

Em EBCK, adota-se uma função *hash* e o algoritmo utilizado (exemplo: MD5 (Rivest, 1992) ou SHA-1 (Eastlake; Jones, 2001)) depende da política de segurança estabelecida para o grupo.

As novas chaves serão formadas da seguinte forma:

- $K' = h(K, Salt)$

Onde:

- **K**: Antiga chave compartilhada
- **K'**: Chave nova gerada pela função h
- **Salt**: Valor aleatório
- **função** $h(a, b)$: função *hash* do valor a concatenado com o valor b

O gerenciador de chaves, então, renova todas as chaves S' e $[K'_1..K'_6]$ e envia para o usuário U_8 as chaves S' , K'_1 , K'_2 e K'_6 criptografadas com a chave pública do novo membro U_8 , de forma que apenas o usuário U_8 obtenha estas informações.

Neste momento, o usuário U_8 tem as novas chaves do grupo, mas o restante do grupo ainda possui apenas as chaves não atualizadas.

Após, o servidor envia para o grupo o valor *Salt*. Este valor circula pelo grupo abertamente, uma vez que não há nenhum vínculo com qualquer outra chave existente.

Desta forma, as únicas mensagens necessárias no processo de união de um novo membro são:

- **Servidor** $\rightarrow$ **U_8** : $(K'_1, K'_2, K'_6, S')_{K_{U_8}}$ (K_{U_8} : chave pública do usuário 8)
- **Servidor** $\rightarrow$ **Grupo**: $(Salt)$

Todos os demais usuários devem atualizar as suas chaves $[K_1..K_6]$ realizando o mesmo processo: $K' = h(K, Salt)$. Desta forma, todos os usuários, incluindo o novo usuário U_8 , têm a nova chave do grupo S' .

Além da redução de banda, a utilização de uma função *one-way* para renovação da chave permite que nenhum usuário seja capaz de descobrir as chaves selecionadas pelo gerenciador para o usuário U_8 . Tais informações, possivelmente, abrirão brechas na segurança.

Para inclusão de usuários, as informações necessárias para a adição deste usuário são conferidas pela ordem $O(1)$.

5.4.3 Múltiplas uniões

Como visto na Seção 2.3.5, uma forma de reduzir o uso de rede no processo de entrada e saída de usuários é o agrupamento de eventos. O EBCK permite um fácil agrupamento de requisições de entrada, podendo o agrupamento ser periódico ou por lote, dependendo da política de segurança empregada no grupo.

Para adicionar múltiplos usuários ao grupo, realiza-se o mesmo processo de união simples para cada novo usuário. A diferença é que o servidor irá gerar um valor *Salt*, atualizar as chaves de sistema K e S apenas uma vez e distribuí-las para os usuários individualmente. Após, enviará um único pacote para o grupo contendo o valor *Salt*.

Caso dois usuários, U_9 e U_{10} , ingressem no grupo, será necessário o envio das seguintes mensagens:

- **Servidor** $\rightarrow U_9$: $(K'_2, K'_3, K'_4, S')_{K_{U_9}}$ (K_{U_9} : chave pública do usuário 9)
- **Servidor** $\rightarrow U_{10}$: $(K'_2, K'_3, K'_6, S')_{K_{U_{10}}}$ ($K_{U_{10}}$: chave pública do usuário 10)
- **Servidor** $\rightarrow$ **Grupo**: $(Salt)$

Assim, o EBCK reduz uma mensagem com o valor *Salt* para cada usuário adicional.

5.4.4 Rechaveamento

A renovação de chaves no esquema EBCK é extremamente simples e eficiente, porquanto basta enviar para o grupo a seguinte mensagem:

- **Servidor** $\rightarrow$ **Grupo**: $(Salt)$

Todos os usuários devem atualizar todas as suas chaves, tanto as chaves de exclusão (K_i), quanto a compartilhada S .

Para renovação de chaves, o volume das informações é dado por $O(1)$.

5.4.5 Abandono

Para excluir um usuário U_i é necessário enviar uma informação contendo o valor do *Salt* criptografado com as chaves $\neg C_i$ deste usuário, ou seja, como a combinação de chaves C_i deste usuário é exclusiva, então, todos os demais usuários n terão $C_n \cap \neg C_i \neq \emptyset$. Destarte, todos os demais membros terão pelo menos uma chave $K_j \in \neg C$ capaz de decodificar ao menos um termo contendo o valor *Salt*, exceto o usuário U_i .

Pelo mesmo exemplo apresentado na Tabela 5.2, supomos que o usuário U_6 seja excluído do grupo (voluntariamente ou por expulsão). Este usuário tem a posse das seguintes chaves combinadas: K_3 , K_5 e K_6 .

O servidor de chaves deve gerar e enviar um novo *Salt* para todo o grupo, porém, o usuário U_6 não deve ser capaz de ter acesso ao valor do *Salt* (caso contrário, este usuário poderia renovar as chaves e continuar no grupo).

Sabendo que, no exemplo, cada usuário tem exatamente três chaves exclusivas, se o servidor enviar para o grupo o valor do *Salt* criptografado exatamente com as chaves que o usuário U_6 não conhece, todos os demais usuários terão pelo menos uma destas chaves, podendo obter o valor do *Salt*.

No exemplo, o gerenciador de chaves deve formar e enviar a seguinte mensagem para o grupo:

- **Servidor** $\rightarrow$ **Grupo**: ($(Salt)_{K_1}$, $(Salt)_{K_2}$, $(Salt)_{K_4}$)

Como o usuário U_6 é o único que não tem as chaves K_1 , K_2 e K_4 , este membro nada pode fazer, sendo incapaz de obter o valor *Salt* e computar as novas chaves.

O U_3 , por exemplo, possui as seguintes chaves de exclusão: K_1 , K_4 e K_5 . Assim, pode usar a chave K_1 para decriptar o valor *Salt* do primeiro termo da mensagem acima e, assim, atualizar as suas chaves S' , K'_1 , K'_4 e K'_5 .

5.4.6 Múltiplos abandonos

Utiliza-se, neste protocolo, uma extensão do conceito proposto por Samuel T. Redwine; Madison (2004) — proposto originalmente por Chang et al. (1999) — para promover a capacidade de exclusão múltipla de usuários.

A exclusão múltipla de usuários utilizada no EBCK é incapaz de excluir um grupo de membros conspiradores (*collusion*), sendo utilizada apenas para reduzir a banda necessária para a renovação de chaves.

Quando dois ou mais usuários desejam sair do grupo, supõe-se que dois processos de abandono (*leave*) devam ocorrer. Porém, é possível, na maioria dos casos, otimizar estas duas mensagens em apenas uma, removendo redundâncias existentes entre estas duas mensagens.

O primeiro passo da exclusão múltipla é fazer um rearranjo das chaves de exclusão. Como exemplo, tomando como base a Tabela 5.2, deseja-se remover dois usuários, o U_1 e o U_2 , e sabe-se que o usuário U_1 tem a posse das chaves K_1 , K_3 e K_4 e o usuário U_2 tem posse das chaves K_1 , K_3 e K_6 . Para excluir os usuários U_1 e U_2 , respectivamente, precisa-se enviar duas informações ao grupo, conforme visto na exclusão simples de usuário:

- **Servidor** $\rightarrow$ **Grupo**: ($(Salt)_{K_2}$, $(Salt)_{K_5}$, $(Salt)_{K_6}$)
- **Servidor** $\rightarrow$ **Grupo**: ($(Salt)_{K_2}$, $(Salt)_{K_4}$, $(Salt)_{K_5}$)

Isto significa que para excluir os usuários U_1 e U_2 , os demais usuários devem ter as chaves K_2 , K_5 e K_6 para o usuário U_1 e as chaves K_2 , K_4 e K_5 para o usuário U_2 . Formaliza-se, então, os dois termos da seguinte forma: $(\bar{K}_2 \vee \bar{K}_5 \vee \bar{K}_6) \wedge (\bar{K}_2 \vee \bar{K}_4 \vee \bar{K}_5)$, sendo que $\bar{K}_i$ significa que é preciso conhecer a chave K_i , ou seja, quaisquer usuários que atendam esta condição são capazes de obter o valor *Salt*.

Utilizando lógica booleana, aplica-se a distributiva na equação anterior:

$$(\bar{K}_2 \vee \bar{K}_5 \vee \bar{K}_6) \wedge (\bar{K}_2 \vee \bar{K}_4 \vee \bar{K}_5) = (\bar{K}_2 \wedge \bar{K}_2) \vee (\bar{K}_2 \wedge \bar{K}_4) \vee (\bar{K}_2 \wedge \bar{K}_5) \vee (\bar{K}_5 \wedge \bar{K}_2) \vee (\bar{K}_5 \wedge \bar{K}_4) \vee (\bar{K}_5 \wedge \bar{K}_5) \vee (\bar{K}_6 \wedge \bar{K}_2) \vee (\bar{K}_6 \wedge \bar{K}_4) \vee (\bar{K}_6 \wedge \bar{K}_5)$$

Podemos simplificar os termos acima, hipótese em que teremos:

$$\begin{aligned} &= (\bar{K}_2) \vee (\bar{K}_2 \wedge \bar{K}_4) \vee (\bar{K}_2 \wedge \bar{K}_5) \vee (\bar{K}_5 \wedge \bar{K}_4) \vee (\bar{K}_5) \vee (\bar{K}_6 \wedge \bar{K}_2) \vee (\bar{K}_6 \wedge \bar{K}_4) \vee (\bar{K}_6 \wedge \bar{K}_5) \\ &= (\bar{K}_2) \vee (\bar{K}_5) \vee (\bar{K}_4 \wedge \bar{K}_6) \end{aligned}$$

Isto significa que quaisquer usuários cujo arranjo de chaves adeque-se à expressão supra explicitada conseguem obter o valor *Salt* e atualizar suas chaves.

Como o protocolo EBCK preza pela máxima redução do uso da rede, verifica-se a redundância deste conjunto de termos para as realidades presentes do grupo. A tabela a seguir separa cada um dos três termos e lista os usuários que se adequam a cada um destes termos.

Termos:	$(\bar{K}_2)$	$(\bar{K}_5)$	$(\bar{K}_4 \wedge \bar{K}_6)$
ID dos usuários	4, 5, 7 e 8	3, 4, 5 e 6	7
Quantidade	4	4	1

Obviamente, os usuários U_1 e U_2 não se enquadram em nenhum destes grupos, pois são exatamente os usuários que devem ser excluídos do grupo.

O próximo passo é tentar descobrir o menor conjunto de termos que atendem a todos os usuários. Consideramos, então, uma coleção C de um sub-conjunto finito S . No exemplo, temos:

$$S = \{3, 4, 5, 6, 7, 8\}$$

$$C = \{\{4, 5, 7, 8\}, \{3, 4, 5, 6, 7\}, 7\}$$

Procura-se um sub-conjunto $C' \subseteq C$, de forma que todos os elementos de S pertençam pelo menos a um dos sub-conjuntos de C' .

Este problema é conhecido como “Cobertura Mínima”, ou *Minimum Set Cover* (MSC). A solução deste problema é um NP-Difícil (Karp, 1972).

Uma forma para resolver este problema — adotado no EBCK — é o Algoritmo Guloso (*Greedy Algorithm*). O Algoritmo Guloso gera um sub-grupo C' utilizando a seguinte regra:

Inicializa o conjunto C' : $C' \leftarrow \{\}$.

Enquanto C' não contiver todos os elementos de S , faça:

$X \leftarrow$ termo de C tal que seja máximo em $\neg(S \cap C')$;

$C' \leftarrow C' \cup X$;

Exemplificando a regra na situação exposta, o primeiro termo ($\{K_4, K_6\}$) engloba o maior número de elementos (4 usuários) e os adiciona ao conjunto C' . Seguidamente, toma o segundo termo que envolve mais elementos (usuários) remanescentes, comparativamente ao terceiro termo (K_5). Estes dois termos já englobam todos os usuários, razão pela qual finaliza-se o algoritmo.

Destarte, o pacote necessário para excluir os usuários U_1 e U_2 é:

- **Servidor** $\rightarrow$ **Grupo**: $((Salt)_{K_2}, (Salt)_{K_5})$

Todos os usuários do grupo conseguem atualizar as chaves, com exceção dos usuários U_1 e U_2 .

É possível utilizar o múltiplo abandono para tantos membros quanto necessário. O exemplo a seguir apresenta o mesmo processo, porém com três membros.

Deseja-se remover os usuários 3, 4 e 5. Para tanto, três expressões são formadas:

- $U_3: (\bar{K}_2 \vee \bar{K}_3 \vee \bar{K}_6)$
- $U_4: (\bar{K}_1 \vee \bar{K}_4 \vee \bar{K}_6)$
- $U_5: (\bar{K}_1 \vee \bar{K}_3 \vee \bar{K}_6)$

Usando o mesmo conceito com o exemplo de dois usuários, temos:

$$\begin{aligned}
 & (\bar{K}_2 \vee \bar{K}_3 \vee \bar{K}_6) \wedge (\bar{K}_1 \vee \bar{K}_4 \vee \bar{K}_6) \wedge (\bar{K}_1 \vee \bar{K}_3 \vee \bar{K}_6) \\
 &= (((\bar{K}_2 \wedge \bar{K}_1) \vee (\bar{K}_2 \wedge \bar{K}_4) \vee (\bar{K}_2 \wedge \bar{K}_6) \vee (\bar{K}_3 \wedge \bar{K}_1) \vee (\bar{K}_3 \wedge \bar{K}_4) \vee (\bar{K}_3 \wedge \bar{K}_6) \vee (\bar{K}_6 \wedge \bar{K}_1) \vee (\bar{K}_6 \wedge \bar{K}_4) \vee (\bar{K}_6 \wedge \bar{K}_6)) \wedge (\bar{K}_1 \vee \bar{K}_3 \vee \bar{K}_6)) \\
 &= (((\bar{K}_1 \wedge \bar{K}_2) \vee (\bar{K}_1 \wedge \bar{K}_3) \vee (\bar{K}_2 \wedge \bar{K}_4) \vee (\bar{K}_3 \wedge \bar{K}_4) \vee (\bar{K}_6)) \wedge (\bar{K}_1 \vee \bar{K}_3 \vee \bar{K}_6)) \\
 &= ((\bar{K}_1 \wedge \bar{K}_2) \vee (\bar{K}_1 \wedge \bar{K}_3) \vee (\bar{K}_1 \wedge \bar{K}_2 \wedge \bar{K}_4) \vee (\bar{K}_1 \wedge \bar{K}_3 \wedge \bar{K}_4) \vee (\bar{K}_1 \wedge \bar{K}_6) \vee (\bar{K}_1 \wedge \bar{K}_2 \wedge \bar{K}_3) \vee (\bar{K}_1 \wedge \bar{K}_3) \vee (\bar{K}_2 \wedge \bar{K}_3 \wedge \bar{K}_4) \vee (\bar{K}_3 \wedge \bar{K}_4) \vee (\bar{K}_3 \wedge \bar{K}_6) \vee (\bar{K}_1 \wedge \bar{K}_2 \wedge \bar{K}_6) \vee (\bar{K}_1 \wedge \bar{K}_3 \wedge \bar{K}_6) \vee (\bar{K}_2 \wedge \bar{K}_4 \wedge \bar{K}_6) \vee (\bar{K}_3 \wedge \bar{K}_4 \wedge \bar{K}_6) \vee (\bar{K}_6)) \\
 &= ((\bar{K}_1 \wedge \bar{K}_2) \vee (\bar{K}_1 \wedge \bar{K}_3) \vee (\bar{K}_1 \wedge \bar{K}_6) \vee (\bar{K}_1 \wedge \bar{K}_2 \wedge \bar{K}_4) \vee (\bar{K}_1 \wedge \bar{K}_3 \wedge \bar{K}_4) \vee (\bar{K}_1 \wedge \bar{K}_2 \wedge \bar{K}_3) \vee (\bar{K}_2 \wedge \bar{K}_3 \wedge \bar{K}_4) \vee (\bar{K}_3 \wedge \bar{K}_4) \vee (\bar{K}_3 \wedge \bar{K}_6) \vee (\bar{K}_1 \wedge \bar{K}_2 \wedge \bar{K}_6) \vee (\bar{K}_1 \wedge \bar{K}_3 \wedge \bar{K}_6) \vee (\bar{K}_2 \wedge \bar{K}_4 \wedge \bar{K}_6) \vee (\bar{K}_3 \wedge \bar{K}_4 \wedge \bar{K}_6) \vee (\bar{K}_6)) \\
 &= ((\bar{K}_1 \wedge \bar{K}_2) \vee (\bar{K}_1 \wedge \bar{K}_3) \vee (\bar{K}_3 \wedge \bar{K}_4) \vee (\bar{K}_6))
 \end{aligned}$$

Após a simplificação, gera-se novamente a tabela com os respectivos termos:

Termos:	$(\bar{K}_1 \wedge \bar{K}_2)$	$(\bar{K}_1 \wedge \bar{K}_3)$	$(\bar{K}_3 \wedge \bar{K}_4)$	$(\bar{K}_6)$
ID dos usuários	8	1 e 2	1	2, 6, 7 e 8
Quantidade	1	2	1	4

Com a utilização do Algoritmo Guloso, chegamos ao conjunto $\{(\bar{K}_6), (\bar{K}_1 \wedge \bar{K}_3)\}$. Assim, torna-se necessária a geração da seguinte mensagem para excluir os usuários 3, 4 e 5:

- Servidor $\rightarrow$ Grupo: $((Salt)_{K_6}, (Salt)_{K_1 \oplus K_3})$

Note-se, novamente, que nenhum dos usuários 3, 4 e 5 consegue obter o valor *Salt*. Estão, por conseguinte, excluídos do grupo.

5.4.7 Crescimento dinâmico de chaves

O EBCK também permite o crescimento dinâmico de chaves do sistema, ou seja, mais chaves podem ser adicionadas durante o envio de dados caso o número de usuários alcance o limite máximo possível com as quantidades atuais de chaves combinadas k do grupo.

Esta característica permite a inicialização do grupo com um conjunto modesto de chaves de exclusão, de forma a minimizar o tamanho dos dados de controle e reduzir o uso de processamento e memória. Conforme haja o crescimento do grupo, novas chaves combinadas serão adicionadas, aumentando a capacidade máxima de usuários do grupo.

Como exemplo, caso um sistema comece com 10 chaves, poderá atender 252 usuários. Quando o usuário número 250 ingressa no grupo, o sistema adiciona mais duas chaves combinadas, totalizando 12 chaves e suportando até 924 usuários. O processo pode ser repetido inúmeras vezes.

A inclusão é realizada a partir da adição de um par de chaves em que uma delas, criptografada com a chave compartilhada S , seja distribuída para todos os usuários do grupo. Desta forma, todos terão uma das duas chaves adicionadas e os novos usuários poderão utilizar a segunda chave para gerar novas combinações exclusivas.

Exemplificando, supõe-se um sistema com 6 chaves e 8 usuários, conforme apresentado na Tabela 5.2. O Gerenciador de Chaves gera duas chaves aleatórias, K_7 e K_8 , e o servidor envia para o grupo (composto por oito membros) a seguinte mensagem:

- Servidor $\rightarrow$ Grupo: $(K_7)_S$

Observa-se que o desempenho é constante, ou seja, $O(1)$ (independe do número de usuários).

Todos os 8 usuários terão esta nova chave K_7 , de forma que um novo usuário U_9 possa utilizar a chave K_8 adicionada. A Tabela 5.3 apresenta o estado do grupo após a inclusão da nova chave. Note que um usuário U_9 foi adicionado com as mesmas chaves do usuário U_8 , porém, para este foi concedida a chave K_8 .

Tabela 5.3: Exemplo de distribuição de chaves após a inclusão de novas chaves K_7 e K_8

	K_1	K_2	K_3	K_4	K_5	K_6	$+K_7$	$+K_8$
U_1	1	0	1	1	0	0	1	0
U_2	1	0	1	0	0	1	1	0
U_3	1	0	0	1	1	0	1	0
U_4	0	1	1	0	1	0	1	0
U_5	0	1	0	1	1	0	1	0
U_6	0	0	1	0	1	1	1	0
U_7	0	1	0	1	0	1	1	0
U_8	1	1	0	0	0	1	1	0
$+U_9$	1	1	0	0	0	1	0	1

5.4.8 Privilégios de usuários

O modelo EBCK especifica uma forma de estabelecer privilégios pra grupos de membros. Para definir privilégios entre usuários, classifica-se uma chave ou um conjunto de chaves K_i que representa um grupo de usuários diferenciados.

Na forma mais simples, pode-se afirmar que os usuários que têm a chave K_1 podem enviar dados, enquanto que os demais só podem receber. No caso, o gerenciador de chaves envia a seguinte mensagem para o grupo:

- **Servidor** $\rightarrow$ **Grupo**: $(S_{pb}, (S_{pr})_{K_1})_S$

Qualquer usuário consegue a chave S pública do grupo, mas os usuários que têm a chave K_1 também obtêm a chave S privada.

Em alguns casos, quando há poucos usuários pertencentes a uma classe de privilégios, pode ser utilizada a combinação de duas chaves e apenas os usuários que conhecem estas chaves podem obter ou exercer determinadas funções.

Caso contrário, metade da quantidade total dos usuários seria reservada para usuários com este privilégio, desperdiçando vagas de usuários que não têm tais privilégios e diminuindo as combinações possíveis para atender todos estes usuários.

No caso de utilização de chaves duplas, por exemplo, as chaves K_1 e K_2 , o servidor deve enviar a seguinte mensagem para o grupo:

- **Servidor** $\rightarrow$ **Grupo**: $(S_{pb}, (S_{pr})_{K_1 \oplus K_2})_S$

Desta forma, apenas os usuários que conhecem as chaves K_1 e K_2 conseguem obter a chave S privada do grupo.

6 FERRAMENTAS E MÉTODOS DE SIMULAÇÃO

Para estudar os esquemas de segurança analisados no Capítulo 4, assim como o esquema proposto nesta dissertação apresentado no Capítulo 5, utilizou-se um simulador de redes para reproduzir o impacto ocasionado pelo gerenciamento e manutenção de chaves por cada um dos esquemas.

O simulador utilizado nesta dissertação, foi o NS-2 (NS-2), é um simulador gratuito e de código aberto amplamente utilizado em experimentos acadêmicos e comerciais em redes computacionais.

Este capítulo apresenta o simulador NS-2 e suas propriedades, assim como as modificações nele realizadas para atender às necessidades da pesquisa, bem como as ferramentas desenvolvidas para a geração de cenário e extração de dados.

6.1 Simulador de rede NS-2

NS-2 é um simulador de eventos discreto concebido para gerar e analisar redes computacionais. O NS-2 provê suporte para simulações TCP/UDP, roteamentos, protocolos *Multicast*, dentre diversas outras características, a partir da utilização de redes cabeadas e redes sem fio. É possível utilizar o NS-2 tanto na plataforma Linux quanto no Windows, neste último utilizando o aplicativo Cygwin.

A geração dos cenários é realizada através de um arquivo do tipo TCL que descreve todo o ambiente a ser analisado e estabelece informações como topologia, quantidade de *host*, configuração de cada *link*, execução de aplicativos, dentre diversas outras configurações.

O software NS-2 gera arquivos de saída contendo determinadas informações da simulação, de acordo com a especificação do arquivo de entrada TCL. Tipicamente, dois arquivos são gerados: um de extensão NAM e outro de extensão TR.

6.1.1 Arquivo de configuração TCL

Para qualquer simulação, o NS-2 exige um arquivo de entrada tipo TCL com todas as características e eventos do cenário a ser simulado. Um exemplo minimalista é apresentado a seguir.

exemplo_simulacao.tcl

```

set ns [new Simulator]

# Abrindo arquivo nam
set nf [open out.nam w]
$ns namtrace-all $nf

#Funcao de encerramento
proc finish {} {
    # Fechar arquivo nam
    global ns nf
    $ns flush-trace
    close $nf
    exit 0
}

# Criando dois host (n0 e n1)
set n0 [$ns node]
set n1 [$ns node]

# Estabelecendo link entre n0 e n1
$ns duplex-link $n0 $n1 2Mb 10ms DropTail

# Gerando agente TCP
set tcp [new Agent/TCP]
$ns attach-agent $n0 $tcp
set sink [new Agent/TCPSink]
$ns attach-agent $n1 $sink
$ns connect $tcp $sink

# Gerando aplicacao FTP
set cbr [new Application/Traffic/CBR]
$cbr attach-agent $tcp
$cbr set packetSize_ 1000
$cbr set interval_ 0.01

# Agendamento de eventos
$ns at 0.5 "$cbr start"
$ns at 4.5 "$cbr stop"

$ns at 5.0 "finish"

# Iniciar simulacao
$ns run

```

Neste exemplo, existem apenas dois *hosts* interconectados por um *link* de 2Mb, com 10ms de *delay*. A simulação tem 5 segundos de duração e do instante 0,5s até 4,5s existe uma aplicação CBR (*Constant Bit Rate*) que envia pacotes de 1000 *Bytes* a cada 0,01 segundo do *host* *n0* ao *n1*.

6.1.2 Visualização da simulação

O arquivo NAM contém toda a seqüência de eventos calculados pela simulação. Com o aplicativo “nam” — também incluído no pacote NS-2 — é possível visualizar graficamente todas as ações simuladas, mostrando a localização de cada nó, os *links* de conexão, a veiculação dos pacotes, dentre outras informações.

Figura 6.1: Aplicativo NAM: visualizador gráfico de simulações NS-2

A Figura 6.1 demonstra o exemplo de visualização de uma simulação. No caso, existem apenas oito nós, e as flechas sobre os *links* representam o tráfego dos dados. O NAM permite também navegar sobre o tempo e visualizar o percurso de um determinado pacote ou destacar o fluxo de uma aplicação específica para análises.

6.1.3 Informações geradas

O NS-2 pode gerar arquivos de extensão TR que contêm informações sobre determinadas ocorrências durante o cálculo da simulação especificado no arquivo TCL. As informações contidas nos arquivos TR devem ser explicitamente definidas dentro do arquivo TCL.

Geralmente, o formato do arquivo é uma tabela com dois campos: tempo e valor requisitado. Por exemplo, uma simulação pode requisitar a quantidade de dados enviados por um determinado nó (em *Bytes*), durante os 5 segundos de simulação. Um possível resultado seria o apresentado a seguir:

	out.tr
1	1500
2	3500
3	4200
4	3200
5	2000

O arquivo pode ser visualizado graficamente com o aplicativo Xgraph, também incluído no pacote NS-2, o qual gera gráficos em função do tempo das informações solicitadas (no exemplo, o tráfego de um determinado nó).

A Figura 6.2 apresenta o exemplo de uso do Xgraph, mostrando o fluxo de saída de três *hosts* distintos em função do tempo.

Figura 6.2: Aplicativo Xgraph

6.1.4 Geração de todos os eventos

É possível gerar um arquivo definido como *all-tracer*, declarando no código TCL a linha `$ns trace-all [open <nome_arquivo> w]`. Este arquivo contém toda as informações calculadas pelo NS-2, incluindo empilhamento de pacotes, descarte de pacotes e outras informações não visualizadas no NAM.

Neste arquivo *all-tracer*, cada linha representa um evento gerado durante o processamento da simulação. No caso das análises realizadas nesta dissertação, utilizaram-se redes cabeadas e, desta forma, cada linha (evento) tem a seguinte estrutura:

```
<event> <t> <from> <to> <pkt> <size> <flag> <fid> <src> <dst> <seq> <uid>
```

Onde:

- <event> Tipo de evento. Pode ser "+"(empilha no *buffer*), "-" (desempilha do *buffer*), "r" (*release*(entregue)), "d" (*drop*), "h" (*hop*), "v" (apenas comentário), dentre outros menos usuais.
- <t> Tempo exato do evento.
- <from> Identificação do nó que está enviando o pacote.
- <to> Identificação do nó que receberá o pacote.
- <pkg> Tipo do pacote ("cbr", "tcp", "udp", "prune", "graft", etc).
- <size> Tamanho do pacote (já fragmentado, caso seja maior do que o MTU (*Maximum Transmission Unit*)).
- <flag> Sinalizações adicionais do pacote. Geralmente não utilizado.
- <fid> Identificação única de fluxo. Muito importante para identificar aplicações.
- <src> Nó de origem do pacote.
- <dst> Destinatário final do pacote.
- <seq> Seqüência do pacote.
- <uid> Identificação única do pacote, muito usual para seguir o fluxo de um pacote específico.

Tipicamente, cada pacote gera três eventos: empilhamento (+), desempilhamento (-) e entrega (r).

Abaixo, segue simples exemplo de eventos ocorridos durante o envio de uma mensagem:

trace_all.out										
+	1	0	2	cbr	210	-----	0	0.0	3.1	0 0
-	1	0	2	cbr	210	-----	0	0.0	3.1	0 0
r	1.00234	0	2	cbr	210	-----	0	0.0	3.1	0 0

Toda a análise desta dissertação tem como base o arquivo *trace-all* que deve ser utilizado conjuntamente com a ferramenta de extração de dados descrita na Seção 6.2.3 para se obter todas as informações necessárias utilizadas no Capítulo 7.

6.2 Ferramentas desenvolvidas

Para a avaliação dos esquemas de segurança presentes neste trabalho, faz-se necessário o desenvolvimento de três módulos:

1. Gerador de cenário TCL;
2. Agente *Multicast* seguro;
3. Analisador de eventos.

As simulações realizadas no Capítulo 7 utilizaram a metodologia apresentada pela Figura 6.3.

Todas as simulações começam pelo gerador de cenário. O simulador NS-2 lê o arquivo TCL criado pelo gerador de cenários e, então, gera um arquivo do tipo NAM e outro *trace-all*. O arquivo NAM pode ser utilizado para visualização da simulação. O analisador de eventos obtém as informações *trace-all* da simulação e gera arquivos contendo determinadas informações especificadas e, então, é possível a geração de gráficos da simulação corrente.

Figura 6.3: Fluxo de simulação

As seções a seguir irão descrever os módulos desenvolvidos, a partir dos estudos realizados para a elaboração desta dissertação.

6.2.1 Geração de cenários TCL

A geração do cenário a partir da utilização de um arquivo TCL pode ocasionar um grande esforço para os usuários, o que representa desperdício de muito tempo de trabalho ou até mesmo torna o experimento inviável para cenários que exigem centenas de *hosts*.

Embora já existam diversos geradores de topologia¹ e cenário² para o NS-2, com o escopo de gerar simulações precisas para os experimentos peculiares deste trabalho, desenvolveu-se um gerador de cenário em linguagem C. Trata-se de um gerador capaz de criar grandes cenários de acordo com as características estabelecidas para uma determinada simulação.

Os tópicos a seguir apresentam algumas das principais características suportadas pelo gerador de cenários.

¹Geradores de Topologias para simulador NS-2, assunto disponível em <http://www.isi.edu/nsnam/ns/ns-topogen.html>

²Gerador de Cenários para simulador NS-2, assunto disponível em <http://www.isi.edu/nsnam/ns/ns-scengeneration.html>

Figura 6.4: Geração de uma rede *mesh*

6.2.1.1 Tipos de topologia

A partir do gerador de cenários implementaram-se quatro tipos de topologia:

- **Estrela:** Uma estrutura de rede tipo estrela é formada por um nó central conectado diretamente com todos os demais nós. Nenhuma simulação foi realizada com esta topologia neste trabalho.
- **Árvore:** A topologia tipo árvore é uma rede que segue hierarquia de conexão. No topo desta hierarquia, encontra-se a raiz e em cada sub-nível, o número de nós aumenta exponencialmente. Nenhuma simulação foi realizada com esta topologia neste trabalho.
- **Mesh:** Uma topologia *mesh* representa geralmente ambientes bem conectados sem seguir necessariamente uma geometria específica. Os nós estabelecem conexões uns relativamente aos outros de acordo com a aproximação ou disponibilidade. O gerador de cenário produz uma rede através de rede em anel e adiciona n conexões aleatórias igualmente distribuídas entre os nós. A Figura 6.4 demonstra este processo.
- **Genérica ou Mista:** Neste trabalho, chama-se topologia genérica a topologia que tenta aproximar características de uma rede real, a Internet. A construção desta topologia segue três etapas:
 1. Constrói-se uma malha tipo *mesh* com n nós;
 2. Em cada um destes nós, gera-se uma árvore de largura l e profundidade p , de acordo com a especificação fornecida;
 3. Enfim, todos os demais nós são interconectados com os nós desta árvore.

Figura 6.5: Exemplo de uma topologia Mista com 60 nós

A topologia mista é a mais utilizada nas simulações apresentadas no Capítulo 7. A Figura 6.5 apresenta exemplo desta topologia.

6.2.1.2 Entrada e Saída de membros

Ao estabelecer um grupo *Multicast*, usuários devem unir-se aos roteadores utilizando protocolos IGMP (Deering, 1989) (para IPv4) ou MDL (Deering et al., 1999) (para IPv6). O gerador de cenário desenvolvido suporta dois tipos de grupos: o estático e o dinâmico.

- **Estático:** Entende-se por grupo de usuários estático aquele em que todos os membros do grupo são dispostos para receber os dados logo no início da simulação. Durante toda a simulação, nenhuma entrada ou saída de membros ocorre, o que evita o restabelecimento de chaves e rotas de encaminhamento.
- **Dinâmico:** Quando gerado um cenário com grupos dinâmicos de membros, a simulação inicia-se e termina sem nenhum usuário no grupo. No entanto, no decorrer de

simulação, usuários entram (*join*) e saem (*leave*) aleatoriamente, em uma distribuição linear, o que força constantes restabelecimentos de chaves e rotas de encaminhamento. Este tipo de simulação será freqüentemente utilizada neste trabalho, uma vez que o objetivo da pesquisa é exatamente a análise e o tráfego ocasionado por entradas e saídas de usuários em um grupo *Multicast* seguro.

6.2.1.3 Tipo de aplicação

A ferramenta de geração de cenários e ambientes é capaz de gerar fluxos de aplicações com dois tipos de transmissão: Variada ou Constante.

- **Taxa de Transmissão Variada:** A taxa de transmissão variada representa aplicações que utilizam a rede de forma aleatória ou indeterminada. Fatores como explosão de envio (*burst*) podem ser encontrados neste tipo de transmissão em que, repentinamente, uma alta taxa de dados gerada pela aplicação é enviada para a rede e, no instante seguinte, a aplicação volta a utilizar pouco recurso da rede. O NS-2 utiliza uma distribuição exponencial para gerar este fluxo.

Este tipo de aplicação pode prejudicar as análises de desempenho de simulações quando se tem por escopo somente avaliar um tráfego específico, o que ocorre nas simulações realizadas nesta dissertação no Capítulo 7, cujo foco é o tráfego de gerenciamento de chaves.

- **Taxa de Transmissão Constante:** Um aplicativo com taxa de transmissão constante (CBR) consubstancia aplicação caracterizada por enviar um volume de dados constante para a rede. Como exemplo, uma distribuição de vídeo ou som é geralmente classificada como aplicativo com taxa de envio constante, pois o uso da rede é previsível em qualquer instante no curso da execução deste aplicativo.

Para as análises realizadas no Capítulo 7, estas espécies de aplicações favorecem a qualidade das medições por não ocasionarem fluxos imprevistos de dados, o que acarreta melhor análise do impacto do tráfego de gerenciamento de chaves na rede.

6.2.1.4 Fluxos de dados na rede

Ao gerar um cenário NS-2 para avaliar o fluxo de uma determinada aplicação, esta pode ser analisada por si só em uma rede dedicada ou, de outro lado, ser analisada em conjunto com outras aplicações concorrentes em uma rede compartilhada.

- **Rede dedicada:** Uma rede dedicada é aquela na qual toda a infra-estrutura é reservada somente para uma determinada aplicação, ou seja, não há nenhum outro fluxo de qualquer outra aplicação durante toda a simulação.

Tem-se como vantagem, na utilização de rede dedicada, a melhor discriminação de fatores aleatórios de comportamento de rede, tornando mais claros os valores gerados por esta aplicação durante a simulação. Porém, caso seja desejável estudar o comportamento de toda a infra-estrutura de uma rede corporativa, a utilização de rede dedicada é insuficiente e inadequada.

- **Rede compartilhada:** Uma rede compartilhada contém duas ou mais aplicações (fluxos de dados) executadas simultaneamente em diferentes pontos desta rede. Aplicações iniciam-se e encerram-se aleatoriamente, podendo ter distribuições constantes, exponenciais ou algumas outras pré-definidas.

Embora este modelo geralmente seja mais compatível com a realidade, a agregação de aplicativos em paralelo pode prejudicar a qualidade das medições de uma aplicação específica em razão de adicionar mais valores aleatórios ou imprevistos. As simulações propostas no Capítulo 7 evitam este tipo de ambiente.

6.2.1.5 Tipos de transmissão em um grupo *Multicast*

Como visto na Seção 3.1, um grupo *Multicast* pode ser classificado em dois tipos: comunicação um-para-muitos e comunicação muitos-para-muitos.

- **um-para-muitos:** O cenário um-para-muitos é caracterizado por haver apenas um remetente enviando dados para múltiplos destinatários.
- **muitos-para-muitos:** Um cenário muitos-para-muitos contém diversos membros que enviam e recebem dados comutativamente.

Figura 6.6: Estrutura do código NS-2 após a inclusão do MSec

6.2.2 Agente *Multicast* seguro

Ao simular diversos esquemas de distribuição e gerenciamento de chaves, além da aplicação, o simulador também deve considerar o tráfego ocasionado pelo gerenciamento de chaves. O cenário que descreve todo o ambiente deve fornecer especificações sobre o esquema de gerenciamento de chaves utilizado e o NS-2 deve ser capaz de interpretar e gerar os fluxos provenientes deste gerenciamento.

Desenvolveu-se, neste trabalho, uma pequena extensão do NS-2, chamada de MSec. O MSec é um agente agregado ao código do NS-2 que simula o comportamento de um esquema de gerenciamento de chaves durante a execução NS-2.

O NS-2, em uma visão geral, é escrito em parte em C++ e em parte em OTcl. O MSec foi implementado em C++, com uma interface em OTcl, o que permite ao cenário TCL interagir com o módulo MSec adicionado. A Figura 6.6 apresenta visão geral do NS-2, bem assim o modo pelo qual o módulo MSec foi adicionado ao código NS-2.

Após a recompilação do código NS-2, é possível criar um agente MSec dentro do arquivo TCL (descrição do cenário). Ao incorporar o Agente MSec, é possível configurar características do gerenciador de chaves do grupo dentro do próprio arquivo TCL. Para isso, uma simulação *Multicast* que utiliza gerenciamento de chaves de grupo deverá incluir em seu código TCL os seguintes comandos:

exemplo_simulacao_com_msec.tcl

```
...
set my_agent [new Agent/MSec]
my_agent set param_1 value_1
my_agent set param_2 value_2
my_agent set param_3 value_3
...
my_agent set param_n value_n
...
```

O código exposto cria um Agente MSec e atribui diversos parâmetros contendo as características do gerenciador de chaves, onde “param_x” corresponde à identificação do parâmetro, e “value_x”, ao valor que deve ser atribuindo para este parâmetro.

Todos estes parâmetros serão repassados para o módulo MSec e incorporados ao código NS-2, o qual deverá interpretar e comportar-se de acordo com as características descritas.

O Agente MSec suporta os seguintes atributos:

- **size_key_**: Tamanho de cada chave de criptografia (em *Bytes*).
- **header_**: Tamanho do cabeçalho.
- **len_join_**: Quantidade de chaves a ser transmitida durante uma requisição *join*.
- **factor_join_**: Fator de influência entre a quantidade de chaves e o número de usuários durante uma requisição *join*.
- **overhead_key_join_**: Cabeçalho adicional para identificação de cada chave durante uma requisição *join* (em *Byte*).
- **len_leave_**: Quantidade de chaves a ser transmitida durante uma requisição *leave*.
- **factor_leave_**: Fator de influência entre a quantidade de chaves e o número de usuários durante uma requisição *leave*.
- **overhead_key_leave_**: Cabeçalho adicional para identificação de cada chave durante uma requisição *leave* (em *Byte*).

- **len_rekey_**: Quantidade de chaves a ser transmitida durante uma requisição *rekey*.
- **factor_rekey_**: Fator de influência entre a quantidade de chaves e o número de usuários durante uma requisição *rekey*.
- **overhead_key_rekey_**: Cabeçalho adicional para identificação de cada chave durante uma requisição *rekey* (em *Byte*).
- **time_rekey_**: Tempo entre cada renovação de chaves, em segundos (0 para desabilitar a função de chaveamento periódico).

Espera-se, com os atributos supra expostos, promover um comportamento muito próximo ao comportamento real de um determinado esquema de gerenciamento de chaves durante a simulação.

O gerador de cenários desenvolvido provê suporte ao agente MSec NS-2, gerando, automaticamente, todos os parâmetros necessários de acordo com o protocolo de gerenciamento declarado.

Como limitação, o MSec não simula o processo de autenticação e sempre supõe que não haja perda de pacotes entre o servidor de chaves e seus membros. Saliente-se, novamente, que quase todos os esquemas de segurança escaláveis devem ter seus dados de controle transmitidos por meio de *Multicast* confiável. Espera-se, com isto, uma limitação que não afete significativamente a qualidade das simulações.

6.2.3 Analisador de eventos

O analisador de eventos consubstancia a terceira ferramenta desenvolvida neste trabalho e tem como objetivo extrair do arquivo *trace-all* — descrito na Seção 6.1.4 — todos os eventos e medições desejados.

O processo por meio do qual o analisador de eventos realiza as análises do arquivo *trace-all* é descrito no fluxograma apresentado na Figura 6.7.

O analisador de eventos obtém as análises desejadas através de um arquivo de configuração. Este arquivo de configuração contém, em cada linha, a descrição de uma determinada análise. Desta forma, com apenas um arquivo de configuração é possível extrair tantas análises quantas necessárias.

Figura 6.7: Fluxograma do analisador de eventos

Cada linha, ou análise, é composta pelos seguintes parâmetros:

<nome_arquivo> <lista_nos> <tipo_grafico> <modo_trans> <evento> <fluxo>

Onde:

- **<nome_arquivo>**: Nome do arquivo de saída. Tipicamente, já adiciona um caminho relativo (ex.: "sim/out.tr").
- **<lista_nos>**: Seleciona quais nós deverão ser processados. "[1]" significa que apenas o nó 1 será analisado, [1..3,6] significa que os nós 1, 2, 3 e 6 serão analisados e [*] indica que todos os nós serão analisados conjuntamente.
- **<tipo_grafico>**: O gráfico pode ser: (a) "avg", uma média entre os nós; (b) "sum", a soma de todos os nós; (c) "total", a soma acumulativa de todos os nós.
- **<modo_trans>**: Pode ser: (a) "recv", apenas dados recebidos; (b) "send", apenas dados enviados; (c) "both", dados enviados e recebidos por cada nó listado.
- **<evento>**: "r" para dados enviados com sucesso, "d" para dados perdidos, ou "*" para calcular todos os eventos.

- **<fluxo>**: Tipo de fluxo analisado. Podem haver: o fluxo MSec, o fluxo do Aplicativo ou o fluxo do Protocolo de roteamento *Multicast*. Pode haver, também, uma combinação entre fluxos, ou “*” para computar todos os fluxos de dados.

A primeira linha do arquivo de configuração deve conter informações sobre tempo e intervalo de análise e deve ter o formato “<start> <step> <end>”, onde “start” é o tempo inicial da análise (tipicamente 0), “step” é o intervalo das amostras e “end”, o tempo final da análise (tipicamente o mesmo tempo da simulação).

Um possível arquivo de configuração do analisador de eventos para um cenário com 600 nós, 200 membros e um servidor é:

```

info.cfg

0 0.5 60

# Analises:
out/source      [0]      total  send  r  <*>
out/src_key     [0]      avg    send  r  <msec>
out/src_app     [0]      avg    send  r  <cbr>
out/members     [1..200]  avg    recv  r  <*>
out/members_key [1..200]  avg    recv  r  <msec>
out/all         [*]      total  send  r  <*>
out/all_key     [*]      total  send  r  <msec>
out/all_prot    [*]      sum    send  r  <prune><graft>

```

O analisador de eventos gera, para cada análise, arquivo com uma tabela contendo dois campos: tempo de simulação e valor requisitado. Este arquivo é compatível com o valor *trace* apresentado na Seção 6.1.3 e pode ser aberto tanto pelo aplicativo Xgraph, quanto por uma planilha eletrônica, por exemplo, Microsoft Excel®.

7 SIMULAÇÕES

Neste capítulo será apresentado o resultado das análises das simulações dos esquemas lolus, DEP, CTKM, EBS, apresentados no Capítulo 4. Apresentar-se-á, ainda, o resultado da análise do esquema EBCK proposto no Capítulo 5.

As simulações foram realizadas utilizando o simulador de redes NS-2 (NS-2) com a inclusão do agente seguro MSec. Utilizaram-se, ainda, ferramentas auxiliares para geração de cenários e análise de dados, conforme apresentado do Capítulo 6.

A próxima seção avaliará a importância do uso de um esquema escalar para gerenciamento de chaves de grupo, apresentando os benefícios em relação a um esquema não escalar. As demais seções apresentarão simulações entre os esquemas de gerenciamento. Cada seção será composta pela descrição do cenário e seus resultados obtidos.

7.1 Simulação 1: Protocolos escaláveis vs. Protocolos não-escaláveis em cenários 1–n

Antes de avaliar as características dos protocolos de gerenciamento de chaves apresentados nesta dissertação, a Simulação 1 demonstrará o impacto de um protocolo escalável em relação a um protocolo não-escalável. A comparação evidencia-se importante para a compreensão da importância dos estudos e dos benefícios proporcionados pela utilização de esquemas de gerenciamento de chaves adequados e eficientes.

Para realizar esta simulação, avaliaram-se dois protocolos genéricos cujos comportamentos seguem a ordem de $O(n)$ e $O(\log(n))$ para um modelo não-escalar e escalar, respectivamente, onde n representa o número de membros do grupo.

7.1.1 Descrição

Consideram-se dois protocolos. O primeiro, um protocolo genérico caracterizado simplesmente como “escalável” e um segundo, também genérico, caracterizado como “não-escalá-

Tabela 7.1: Descrição da Simulação 1

Modo de transmissão	Multicast Confiável (sem perda)
Quantidade total de <i>hosts</i>	600
Número total de membros	400
Número de servidores dedicados	1
Característica de cenário	um-para-muitos
Taxa de envio da aplicação	CBR a $256 \frac{\text{kb}}{\text{s}}$ (1,6 kB a cada 0,05 s)
Tempo de simulação	60 segundos
Protocolo de roteamento <i>Multicast</i>	DVMRP
Tipo de topologia	Mista
Capacidade em cada <i>link</i>	10Mb
Atraso em cada <i>link</i> (<i>delay</i>)	10ms
Entrada e Saída de usuários	Dinâmica
Tamanho da chave de criptografia	128 <i>bits</i> (16 <i>Bytes</i>)
Período para cada evento de renovação	Não há renovação periódica

vel". Trata-se de protocolos designados como genéricos porque não demonstram nenhuma peculiaridade, apenas seguem um comportamento $O(n)$ e $O(\log(n))$ para as operações de entrada e saída de usuários.

Como definição, o tamanho dos cabeçalhos destes dois esquemas é de 32 *Bytes*. Os protocolos utilizam chaves simétricas com tamanho de 128 *bits* (ou 16 *Bytes*). O cenário é composto por 600 *hosts*, sendo 400, membros do grupo, e um deles, servidor dedicado de dados (cenário 1-n). Este servidor envia dados com uma taxa constante de $256 \frac{\text{kb}}{\text{s}}$ ($32 \frac{\text{kB}}{\text{s}}$).

Em cada entrada(*join*) ou saída(*leave*) de usuário, considerou-se:

- **não-escalável:** Cada operação (*join* e *leave*) terá o tamanho de $32 + n \times 16$, onde n varia de acordo com o número de membros no instante do envio do evento, ou seja, quanto mais membros estiverem atualmente no grupo, maior o volume de dados necessário para o gerenciamento das chaves (proporcional).
- **escalável:** Cada operação (*join* e *leave*) terá o tamanho de $32 + \log(400) = 176$ *Bytes*, considerando-se que há no máximo 400 usuários e são necessárias $\log_2^{400} \approx 8.6 \rightarrow 9$ chaves enviadas por operação. Considera-se que não há relação entre o número de membros atualmente no grupo e o tamanho necessário para o gerenciamento de chaves, ou seja, independentemente de quantos membros haja atualmente no grupo, o tamanho para veicular as informações de gerenciamento serão sempre constantes.

A Tabela 7.1 apresenta o resumo da Simulação 1.

Figura 7.1: Simulação 1: Distribuição de usuários e taxa de *join* e *leave*

7.1.2 Resultados e Análises

A Figura 7.1 apresenta a quantidade de membros no grupo em função do tempo, designado pela graduação esquerda do gráfico. No mesmo gráfico, também se apresenta o fluxo de entradas e saídas de usuários do grupo composto pela soma de requisições *join* e *leave* por segundo, utilizando a escala da direita do gráfico.

O sistema inicializa e finaliza sem qualquer membro no grupo. Os membros entram e saem em determinados períodos, aleatoriamente, e embora 400 membros entrem e saiam (uma única vez) do grupo, o máximo número de usuários é de 197 membros no instante $t = 30s$.

A Figura 7.2 apresenta o fluxo de saída do servidor. O fluxo foi classificado em dois tipos: tráfego da aplicação e tráfego do gerenciador de chaves.

Em ambas as simulações (escalável e não-escalável), o tráfego do aplicativo foi de $32 \frac{kB}{s}$, com uma taxa de envio constante, conforme especificado. O *Multicast* é um protocolo escalável e, desta forma, o número de membros não repercute na taxa de envio de dados da aplicação.

Figura 7.2: Simulação 1: Informações enviadas pelo servidor de dados

Considerando-se que 400 usuários entraram e saíram do sistema, temos na simulação 800 eventos *join* e *leave*. No caso do esquema escalável, analiticamente temos uma média de $\frac{800 \times 176 \text{ Bytes}}{60 \text{ Segundos}} = 2346 \frac{\text{Bytes}}{\text{Segundo}}$. Esta foi a média de dados de controle utilizada para o esquema escalável.

Já no esquema não-escalável, pelo gráfico de usuários apresentado na Figura 7.1, temos para o instante $t = 23s$ 8 requisições *join* e 12 *leave*, com 185 membros ativos. Estas requisições geram um pico de $(32 + 185 \times 16) \times (8 + 12) = 59840 \text{ Bytes}$, quase 60 kB apenas para o gerenciamento de chaves, ultrapassando quase duas vezes o próprio fluxo da aplicação que tem a taxa de $32 \frac{\text{kB}}{s}$.

O esquema não-escalável gerou um tráfego médio de $27514,21 \frac{\text{kB}}{s}$ apenas para realizar o gerenciamento das chaves, ou seja, no cenário proposto pela Simulação 1, o esquema escalável foi $\frac{27514,21}{2346} \approx 12\times$ mais eficiente do que o esquema não-escalável. Observa-se, ainda, que diversos protocolos escaláveis têm a taxa de *join* representada por $O(1)$, como o caso do EBCK, tornando a diferença mais significativa.

Além da análise do ponto de vista do servidor, a Figura 7.3 apresenta o uso médio de toda a rede, ou seja, a banda média utilizada, em $\frac{\text{Bytes}}{\text{segundo}}$, de cada *host* do sistema. Esta análise engloba tanto o servidor e os membros, quanto os demais usuários que compõem o sistema.

Figura 7.3: Simulação 1: Utilização geral da rede

Observa-se no gráfico da Figura 7.3 que o uso da rede é influenciado pelo número de membros atualmente no grupo. É fácil deduzir que, quanto maior o grupo, mais a informação deve se propagar pela rede, ocasionando um maior tráfego.

A partir da análise da Figura 7.3, notamos novamente a diferença entre o esquema escalável e o não-escalável. No esquema não-escalável, observa-se que a alta taxa de gerenciamento de chaves propaga-se por toda a rede, ocasionando um tráfego excessivo não apenas para o servidor, mas também para os membros e todos os demais *hosts*.

Outra consideração importante a ser feita é que no caso do esquema escalável a informação trafegada ao longo da simulação é próxima de uma constante, ou seja, é proporcional à quantidade de requisições *join* e *leave*. Esta característica faz com que o sistema seja mais previsível, ao contrário do esquema não-escalável que apresenta diversos picos e torna o sistema suscetível a constantes falhas.

7.2 Simulação 2: Protocolos escaláveis por grupo em cenários 1–n

A simulação 2 apresenta o resultado de simulações com os seguintes esquemas: CTKM, EBS e EBCK. Estes esquemas são protocolos escaláveis que realizam o controle de grupo, conforme estudado na Seção 2.3.2. Desta forma, os protocolos IOLUS e DEP que realizam o controle de sub-grupos não foram incluídos nesta simulação.

O cenário utilizado nesta simulação é compatível com a simulação 1 (escalável vs. não-escalável). A distribuição das requisições *join* e *leave* é a mesma apresentada na simulação 1, conforme Figura 7.1.

A simulação 2 tem como objetivo estudar e analisar o uso de rede ocasionado pelo gerenciamento de controle e, adicionalmente, analisar o comportamento geral da rede.

7.2.1 Descrição

Supõe-se que não há perda de pacotes de controle e existe renovação de chaves periodicamente, com intervalo de $2s$. A simulação terá 60 segundos, com 400 membros entrando e saindo do grupo em uma topologia mista formada por 600 *hosts* e, ainda, com um servidor de dados enviando pacotes em uma taxa constante de $256 \frac{kb}{s}$.

A Tabela 7.2 apresenta o resumo da simulação 2.

7.2.2 Resultados e Análises

A Figura 7.4 apresenta o volume de dados enviados pelo servidor para o gerenciamento de chaves. Desconsiderou-se, neste gráfico, o uso ocasionado pela aplicação, uma vez que se espera uma taxa constante de $32 \frac{kB}{s}$ para cada simulação, conforme especificado na Tabela 7.2.

Tabela 7.2: Descrição da Simulação 2 (4 e 5)

Modo de transmissão	Multicast Confiável (sem perda)
Quantidade total de <i>hosts</i>	600
Número total de membros	400
Número de servidores dedicados	1
Característica de cenário	um-para-muitos
Taxa de envio da aplicação	CBR a $256 \frac{kb}{s}$ (1,6 kB a cada 0,05 s)
Tempo de simulação	60 segundos
Protocolo de roteamento <i>Multicast</i>	DVMRP
Tipo de topologia	Mista
Capacidade em cada <i>link</i>	10Mb
Atraso em cada <i>link</i> (<i>delay</i>)	10ms
Entrada e Saída de usuários	Dinâmica
Tamanho da chave de criptografia	128 bits (16 Bytes)
Período para cada evento de renovação	2 segundos

Figura 7.4: Simulação 2: *Overhead* do servidor ocasionado pelo gerenciamento de chaves

Para cada um dos três esquemas analisados, a média dos dados enviados pelo servidor para o gerenciamento de chaves foi:

Esquema	<i>Join</i>	<i>Leave</i>	<i>Rekey</i>	Total
CTKM	1293 $\frac{\text{KB}}{\text{s}}$	1293 $\frac{\text{KB}}{\text{s}}$	32 $\frac{\text{KB}}{\text{s}}$	2618 $\frac{\text{KB}}{\text{s}}$
EBS	933 $\frac{\text{KB}}{\text{s}}$	1386 $\frac{\text{KB}}{\text{s}}$	43 $\frac{\text{KB}}{\text{s}}$	2363 $\frac{\text{KB}}{\text{s}}$
EBCK	320 $\frac{\text{KB}}{\text{s}}$	933 $\frac{\text{KB}}{\text{s}}$	24 $\frac{\text{KB}}{\text{s}}$	1277 $\frac{\text{KB}}{\text{s}}$

O uso necessário para cada evento justifica o comportamento do gráfico apresentado na Figura 7.4 em que o CTKM possui desempenho pior que o EBS no início, onde há mais eventos *join*, o que se inverte no final da simulação, quando o CTKM mostra-se mais eficiente que o EBS para as requisições *leave*.

Importante observar que o esquema CTKM, conforme apresentando em (Wong et al., 1997), tem a requisição *join* representada por $O(\log(n))$, não $O(1)$, como os sucessivos esquemas baseados em árvores.

No esquema EBS, também não se considerou o uso de funções *one-way* (Chang et al., 1999) citado por (Morales et al., 2003). O uso de funções *one-way* pode proporcionar uma redução de banda e foi descartado nas simulações, pois Morales apenas possibilitou a sua utilização, mas não especificou o seu uso em seu artigo.

O gráfico apresentado na Figura 7.5 apresenta o impacto dos esquemas de segurança em toda a rede. O gráfico apresenta a taxa média de pacotes recebidos pelos *hosts*, sendo membros ou não, em função do tempo. A graduação esquerda do gráfico refere-se à média, em *Bytes*, de recebimento dos esquemas CTKM, EBS e EBCK, enquanto que a graduação da direita refere-se à quantidade de usuários do sistema. As colunas apresentadas na função de quantidade de usuários representam o volume de *join* (coluna superior) e *leave* (coluna inferior), também utilizando a graduação direita do gráfico.

Conforme já mencionado na simulação 1, o volume de dados na rede é proporcional ao números de membros, pois os dados precisam se propagar por mais *hosts* para atingir todos os membros do grupo.

Interessante observar (Figura 7.5) o impacto da rede como um todo ocasionado pelos eventos (a soma da coluna superior (*join*) e inferior (*leave*)). Como exemplo, no instante $t = 35$ houve apenas 4 requisições *join* e 3 *leave*, o que pouco impactou a rede, enquanto que no $t = 40$ houve 8 requisições *join* e 12 *leave*, ocasionando um pico de utilização da rede.

Figura 7.5: Simulação 2: *Overhead* geral ocasionado pelo gerenciamento de chaves

Por este gráfico, compreende-se a importância de um esquema de gerenciamento de chaves que possibilite suporte a agrupamento de eventos, conforme visto na Seção 2.3.5. Nenhuma das simulações realizadas neste trabalho utilizou agrupamento de eventos.

7.3 Simulação 3: Protocolos escaláveis por grupo em cenários $n-n$

As simulações 1 e 2 apresentaram cenários $1-n$, ou seja, existem diversos membros e um servidor de dados. Nas simulações 1 e 2, os membros não têm permissão para enviar nenhuma informação, o que faz com que todo o tráfego seja gerado exclusivamente pelo servidor, com exceção dos eventos de requisição de *join* e *leave* e dos dados para o controle de roteamento *Multicast*. Nesta simulação, estudar-se-ão cenários em que múltiplos membros enviam e recebem dados simultaneamente.

Tabela 7.3: Descrição da Simulação 3

Modo de transmissão	Multicast Confiável (sem perda)
Quantidade total de <i>hosts</i>	100
Número total de membros	60
Característica de cenário	muitos-para-muitos
Taxa de envio da aplicação	CBR a $64 \frac{\text{kb}}{\text{s}}$ (0,4 kB a cada 0,05 s)
Tempo de simulação	60 segundos
Protocolo de roteamento <i>Multicast</i>	DVMRP
Tipo de topologia	Mista
Capacidade em cada <i>link</i>	10Mb
Atraso em cada <i>link</i> (<i>delay</i>)	10ms
Entrada e Saída de usuários	Dinâmica
Tamanho da chave de criptografia	128 bits (16 <i>Bytes</i>)
Período para cada evento de renovação	2 segundos
Tamanho da assinatura do membro por pacote	16 <i>Bytes</i>

7.3.1 Descrição

A simulação 3 apresenta um cenário típico de videoconferência e cada membro deve prover privacidade e autenticidade para os demais membros. Conforme exposto na Seção 3.2.2, pode haver autenticação de grupo ou autenticação de origem. A simulação 3 trata da autenticação de origem.

Como o objetivo da pesquisa não é avaliar os modelos de autenticação, considerou-se um modelo de autenticação genérico que realiza uma assinatura de 16 *Bytes* em cada pacote enviado por cada membro.

Cada membro enviará dados em uma taxa constante, pois o foco da pesquisa é apenas o gerenciamento de chaves e uma distribuição exponencial poderia prejudicar as medições. O cenário é composto por 100 *hosts* em uma topologia mista, com 60 membros entrando e saindo aleatoriamente do grupo, em um período de 60 segundos. A simulação inicia-se e encerra-se sem membros no grupo.

A Tabela 7.3 apresenta o resumo da Simulação 3.

Figura 7.6: Simulação 3: Distribuição de usuários e taxa de *join* e *leave*

7.3.2 Resultados e Análises

Na simulação, a entrada, a saída e a quantidade de usuários em função do tempo são apresentadas na Figura 7.6. Sessenta usuários entraram e saíram durante a simulação, atingindo, no instante $t = 29$, o número de 32 membros simultâneos.

A primeira análise é apresentada na Figura 7.7. O gráfico apresenta o comportamento de um membro m que entrou no grupo no instante $t_i = 8s$ e saiu no instante $t_f = 31s$. A taxa de envio da aplicação foi constante para todos os esquemas, ou seja, de $8,32 \frac{kB}{s}$, conforme anunciado na Tabela 7.3: $\frac{400B+16B}{0.05s} = 8320 \frac{B}{s} \approx 66,5 \frac{kb}{s}$.

A Figura 7.7 apresenta o fluxo de pacotes de gerenciamento de chaves recebido durante a permanência do usuário no grupo. A apresentação deste gráfico auxiliará a compreensão do impacto de cada processo *join*, *leave* e *rekey* para cada um dos membros do grupo. Note-se que no instante $t_1 = 13$ e $t_2 = 25$ não houve nenhum evento (Figura 7.6) e, desta forma, o membro m não recebeu nenhuma informação de renovação de chaves do servidor de chaves.

O gráfico da Figura 7.8 expõe o volume dos dados de controle de chaves gerado pelo

Figura 7.7: Simulação 3: Fluxo de dados de gerenciamento recebido pelo Membro 1

gerenciador de chaves. A graduação da esquerda representa o volume, em *Bytes*, do tráfego dos pacotes de gerenciamento em toda a rede, enquanto a graduação da direita apresenta o número de membros atualmente no grupo, sendo que as colunas superiores representam a quantidade de *join* e, as inferiores, a quantidade de *leave*.

O volume de dados de controle é proporcional ao tamanho do grupo e à quantidade de eventos. No instante $t = 34$, observamos um pico ocasionado pelo excesso de eventos. No caso, há 27 membros, duas requisições de *join* e quatro de *leave*. Isto significa que existem seis eventos enviados para 27 usuários, ou seja, o volume de dados presente na rede neste instante foi de 9391, 9113 e 5367 *Bytes* para os esquemas CTKM, EBS e EBCK, respectivamente.

Por fim, a Figura 7.9 apresenta o fluxo total (cumulativo) dos dados de controle durante toda a simulação. Novamente, temos que a graduação da esquerda representa o volume, em *Bytes*, da soma do uso da rede pelos esquemas de segurança, e a graduação da direita mostra o número de membros no grupo.

Figura 7.8: Simulação 3: Utilização da rede pelo gerenciador de chaves

Figura 7.9: Simulação 3: Acumulado do uso médio de rede para gerenciamento de chaves do sistema

A utilização total pode ser assim especificada:

Esquema	<i>Join</i>	<i>Leave</i>	<i>Rekey</i>	Total
CTKM	113688 (49,4%)	113688 (49,4%)	2813 (1,2%)	230191
EBS	83972 (39,5%)	124743 (58,7%)	3870 (1,8%)	212586
EBCK	28765 (25%)	83869 (73,1%)	2157 (1,9%)	114793

7.4 Simulação 4: Protocolos escaláveis com sub-grupos

A simulação 4 apresenta os esquemas escaláveis lolus e DEP que realizam controle de sub-grupos.

7.4.1 Descrição

O cenário utilizado para a realização da simulação 4 é similar ao utilizado na simulação 2 e, desta forma, as informações do cenário apresentadas na Tabela 7.2 são válidas para esta simulação.

7.4.2 Resultados e Análises

A Figura 7.10 apresenta a taxa de eventos *join* e *leave* no grupo, assim como o número total de usuários. Novamente, a graduação da esquerda refere-se ao número de usuários e a graduação da direita refere-se ao número de eventos (*join* e *leave*).

Para cada um dos esquemas estudados realizaram-se três simulações. Como os protocolos lolus e DEP são baseados em controle de sub-grupo, considerou-se, para cada simulação, o uso de 4, 6 e 8 sub-grupos, ou seja, dividiram-se os membros em 4 grupos de 150 *hosts* e 100 membros, 6 grupos de 100 *hosts* e 75 membros e, finalmente, 8 grupos de 75 *hosts* e 50 membros, respectivamente.

Figura 7.10: Simulação 4: Distribuição de usuários e taxa de *join* e *leave*

A Figura 7.11 apresenta o resultado destas seis simulações (3 para cada esquema). Quanto maior o número de sub-grupos, espera-se uma menor quantidade de dados para gerenciamento. Esta assertiva tem como base o fato de que quanto menor o número de usuários por sub-grupo, menor o tamanho do pacote necessário para o controle dos membros. Entretanto, importa notar que quando se diminui muito o número de membros por sub-grupo, começa a haver prejuízo para a escalabilidade do *Multicast*.

7.5 Simulação 5: Comparação entre esquemas de gerenciamento

A simulação 5 tem como objetivo comparar todos os esquemas de gerenciamento apresentados nos Capítulos 4 e 5.

Figura 7.11: Simulação 4: IOLUS vs. DEP, com 4, 6 e 8 subgrupos

Figura 7.12: Simulação 5: Comparação entre os esquemas IOLUS, DEP, CTKM, EBS e EBCK

7.5.1 Descrição

O cenário utilizado para esta simulação é o mesmo utilizado nas simulações 2 e 4, resumido na Tabela 7.2. O cenário 1-n foi escolhido para comparação em razão do DEP não suportar cenários n-n.

7.5.2 Resultados e Análises

A Figura 7.12 demonstra a comparação entre todos os esquemas em um único gráfico. A base de comparação foi a taxa de envio de dados de controle pelo servidor. Demais análises, embora tenham sua importância, estão diretamente relacionadas com o fluxo dos dados de gerenciamento encaminhado pelo servidor.

A primeira análise do gráfico lastreia-se na eficiência dos mecanismos baseados em controle de sub-grupo. A eficiência surge em virtude das informações transmitidas por estes sub-grupos não se propagarem para outros sub-grupos. Porém, uma aplicação pode ter dificuldades em utilizar os esquemas baseados em controle do sub-grupo devido a seleções dos *hosts* confiáveis. Também é preciso asseverar que o uso de controle de sub-grupos em redes de sensores ou redes *ad hoc* torna-se inviável.

Uma segunda análise aponta que, em esquemas baseados em controle de grupo, como o CTKM, EBS e EBCK, o uso da rede para o controle de chaves é mais independente em relação ao número de membros atualmente no grupo, enquanto que no lolus e no DEP, nota-se um crescimento da rede na medida em que o número total de membros aumenta. Este fato ocorre em razão do lolus e do DEP usarem um *Flat Schemes* para o gerenciamento dos membros de seus sub-grupos. Quanto maior a quantidade de membros em cada sub-grupo, maior a informação necessária para gerenciar a chave compartilhada.

8 CONCLUSÃO

A utilização de conexões *Multicast* pode proporcionar drástica redução do uso da rede em aplicações nas quais uma determinada informação deva atingir um grupo de usuários. A principal característica do *Multicast* é permitir a construção de grupos de usuários de forma transparente e escalar. Transparente, porque a aplicação não deve se preocupar com os usuários do grupo, e escalar, pois a adição de novos usuários não significa um aumento proporcional na utilização da rede.

Os protocolos IPv4 e IPv6 *Multicast*, o IGMP e MLD, respectivamente, não oferecem nenhum mecanismo de controle de acesso ao conteúdo, ou seja, não realizam nenhum tipo de criptografia para a proteção do conteúdo dos dados transmitidos pelo grupo.

Porém, inúmeras aplicações necessitam de algum mecanismo de segurança para proteger os conteúdos trafegados pela rede como, por exemplo: (a) sistema de WebTV via Internet para assinantes; (b) atualização de software pagos; (c) atualizações financeiras entre uma matriz e suas filiais; (d) videoconferências privadas, dentre muitas outras aplicações.

Desta forma, é necessário agregar algum mecanismo de segurança na aplicação para prover a proteção do grupo, ou seja, prover privacidade, irretratabilidade e integridade ao conteúdo do grupo.

Para a consecução da privacidade, utiliza-se criptografia. Dados cifrados só devem ser decodificados por usuários que conhecem o segredo, porquanto os usuários, membros de um grupo de usuários, devem trocar informações entre si sem que outros usuários não autorizados consigam lograr acesso ao conteúdo destas informações.

Em grupos *Multicast*, podem haver freqüentes requisições de entradas e saídas de usuários ao longo do tempo e, desta forma, a chave compartilhada pelo grupo deve ser modificada para garantir que os usuários que saiam do grupo deixem de ter acesso às informações do mesmo e, por outro lado, os usuários que ingressem, não tenham acesso aos dados enviados antes de sua entrada. Trata-se da confidencialidade temporal.

Com a movimentação de usuários no grupo, é necessário um mecanismo para atualização de chaves, o que possibilita que a cada evento de união ou abandono de algum membro ao grupo a chave compartilhada para o acesso dos conteúdos seja trocada.

A cada renovação de chaves de grupo, o usuário que acaba de entrar não pode ter acesso à chave antiga e o usuário que acaba de sair do grupo não pode ter acesso à nova chave, mas todos os demais usuários obtêm a nova chave, caso contrário, eles também perderiam o acesso ao conteúdo do grupo.

Porém, como visto nesta pesquisa, a inclusão de um sistema de gerenciamento de chaves compartilhadas pode proporcionar significativo aumento no uso da rede, devido ao tráfego de informações de controle.

Para a realização de gerenciamento de chaves, uma simples abordagem é enviar a nova chave criptografada com a chave pessoal de cada usuário para o grupo, porém, o tamanho da mensagem é proporcional à quantidade de usuários do grupo, o que torna o método inviável na hipótese de existir expressivo número de usuários no grupo.

Para atingir a escalabilidade do mecanismo de gerenciamento de chaves compartilhadas, diversas pesquisas propostas na literatura permitem a união e a exclusão de grupos sem que haja um aumento proporcional entre o número de usuários atualmente no grupo e a utilização da rede, o que permite obter relação $O(\log(n))$, onde n é o número de usuários do grupo *Multicast*.

Nesta pesquisa, foram analisados os protocolos lolus, DEP, CTKM e EBS. Trata-se de alguns protocolos escaláveis para gerenciamento de chaves de grupo freqüentemente encontrados na literatura.

Além disso, apresentou-se novo esquema de gerenciamento de chaves de grupo, o EBCK (*Exclusion-Based Combinatorial Key*). O EBCK tem como principal característica a máxima redução do uso dos recursos de rede em cada renovação de chaves compartilhadas.

O uso do EBCK pode proporcionar excelente desempenho em grupos *Multicast* com grande número de usuários em que geralmente a banda limita ou inviabiliza o uso de segurança.

Redes de sensores também podem ser beneficiadas com o uso do EBCK, em razão do mesmo utilizar pouco recurso de rede, não depender de nós intermediários para o gerenciamento e, ainda, prover uma redução no uso de processamento e memória.

A presente dissertação também apresentou como contribuição a comparação entre os protocolos lolus, DEP, CTKM, EBS e o esquema proposto EBCK, disponibilizando gráficos e análises entre estes protocolos para futuros trabalhos e pesquisas.

Além disso, apresentaram-se ferramentas para geração de cenários, além de analisador de eventos para NS-2. Houve, também, a disponibilização do agente seguro NS-2 para simulação de esquemas de gerenciamento de chaves.

Os quatro módulos criados e desenvolvidos nesta dissertação — EBCK, gerador de cenários TCL, agente *Multicast* seguro (Msec) e analisador de eventos — permitem a usuários menos

versados quanto ao mecanismo de funcionamento do NS-2, a realização de simulações de redes *Multicast* e *Multicast* seguro e, especialmente, possibilitam o seu uso por outros pesquisadores que desejam estender as funcionalidades dos módulos desenvolvidos para novas pesquisas e outros protocolos de gerenciamento.

8.1 Trabalhos futuros

A presente dissertação possibilita a realização dos seguintes trabalhos futuros:

1. O esquema EBCK pode ser aperfeiçoado ou adaptado para outros ambientes;
2. As ferramentas desenvolvidas para simulação em NS-2 podem ser reutilizadas em futuros trabalhos. Ademais, podem-se realizar adaptações e aperfeiçoamentos para estender o uso das ferramentas desenvolvidas nesta dissertação;
3. Em razão das ferramentas auxiliares NS-2 serem de fácil e simples utilização, é possível gerar uma página Web capaz de simular cenários e ambientes configurados por seus usuários, gerando diversas análises sobre a simulação realizada remotamente.

A MULTICAST EM TCP/IP

A.1 *Multicast* na Camada 2

Geralmente, o NIC (*Network Interface Card*) de uma LAN (*Local Area Network*) recebe apenas o pacote cujo campo de destinatário na camada de enlace contenha o seu endereço MAC (*Media Access Control*) ou pacote do tipo *Broadcast* (todos os seis octetos que compõem o endereço MAC em “1”).

Utilizando-se *Multicast*, o NIC deve ter a capacidade de possibilitar a “entrada” dos pacotes *Multicast* dos grupos a que pertence (Wittmann; Zitterbart, 2000, p. 45).

A Figura A.1 apresenta o campo de seis *Bytes*, ou 48 *bits*, que compõem o endereço MAC, segundo o padrão IEEE 802.3.

O *bit* assinalado na Figura A.1 indica para o NIC que o pacote consubstancia endereço *Multicast* ou *Broadcast*, fazendo com que o pacote suba para as camadas superiores.

Para endereçamento MAC em pacotes *Multicast*, o IANA (IANA) alocou um endereçamento especial em que os primeiros 25 *bits* possuem o valor “01-00-5E-7”, em hexadecimal. Desta forma, cria-se uma faixa de endereços *Ethernet* MAC válidos de 01-00-5E-00-00-00 a 01-00-5E-7F-FF-FF, representada em hexadecimal.

Os últimos 23 *bits* são preenchidos com os próprios 23 *bits* menos significativos do endereço IP *Multicast*. A Figura A.2 apresenta exemplo de conversão do endereço IP *Multicast* para o endereço MAC.

Pela Figura A.2, nota-se que durante a conversão há perda de cinco *bits*, ou seja, cada endereço único MAC pode conter 32 (2^5) endereços IPs *Multicast*. Este fato apenas retrata que no caso de haver dois ou mais IPs *Multicast* que produzem um mesmo endereço MAC em uma mesma rede, todos os membros de ambos os grupos terão que processar os pacotes um do outro para descartá-los posteriormente se estes não pertencerem ao grupo.

Observa-se, por fim, que nem todas as placas de rede têm suporte *Multicast*. Nestes casos, os pacotes *Multicast* neste segmento de rede deverão conter um endereçamento MAC do tipo *Broadcast* para que o pacote alcance todos os membros. Isto repercute no processamento adicional de todos os computadores locais não pertencentes ao grupo *Multicast*. Ressalte-se que este processo adicional é ocasionado pela necessidade de processamento de todos os pacotes em camadas superiores à camada enlace.

Figura A.1: Formato MAC

Figura A.2: Mapeamento *Multicast* no MAC

A.2 *Multicast* na Camada 3 (IP)

Este tópico demonstra como representar o endereço IP *Multicast* em IPv4 e IPv6 (Wittmann; Zitterbart, 2000, p. 47). O endereço IP em *Multicast* possui faixa reservada de utilização (Edwards et al., 2002, p. 21):

- **Em IPv4 (Classe D):** Os quatro *bits* iniciais contêm os valores “1110”, fornecendo uma faixa de endereço de 224.0.0.0 à 239.255.255.255.
- **Em IPv6:** Os oito *bits* iniciais contêm todos os *bits* em 1 (0xFF).

Nota-se, pela definição acima, que uma das principais diferenças entre o IPv4 e o IPv6 é a capacidade de endereçamento: 32 *bits* em IPv4 (ex.: 225.13.133.200) e 128 *bits* em IPv6 (ex.: 68E6:8C68:0000:0000:0021:1180:0285:F23F).

B ROTEAMENTO MULTICAST

B.1 Roteamento de pacotes *Multicast*

Os protocolos de gerenciamento (IGMP para IPv4 e MLD para IPv6) descrevem o modo pelo qual os usuários interagem com o roteador, porém eles não especificam como os roteadores trocam informações relativas à localização dos membros do grupo entre si e não determinam como os roteadores devem adiantar os pacotes *Multicast* de forma a garantir que a mensagem chegará a todos os membros *Multicast* (Dekker, 2000, p. 93).

Nos próximos tópicos, apresentar-se-ão as armadilhas a serem evitadas pelos protocolos, além de uma visão geral do modo pelo qual se realiza o adiantamento *Multicast*.

B.1.1 Formas de roteamento

O endereço IP *Multicast* é mais do que um simples destinatário (Williamson, 2000, p. 5): representa um grupo de usuários. Além disso, qualquer usuário pode entrar ou sair do grupo a qualquer momento, forçando constante manutenção das tabelas de rotas em cada roteador que compõe a rede.

Encontram-se arrolados abaixo alguns dos problemas comuns em conexões *Multicast*.

- Se um remetente envia um pacote *Multicast* para um grupo de usuários, sabe-se que o próprio remetente faz parte deste grupo e cuidados deverão ser implementados para que este pacote não retorne indevidamente ao remetente.
- Sabendo-se que os pacotes *Multicast* podem ser duplicados ao longo da rede, o protocolo de roteamento deve inibir situações nas quais pacotes duplicados circulem em um mesmo segmento duas vezes, fazendo com que dois ou mais pacotes idênticos trafeguem em uma mesma sub-rede.
- Os pacotes *Multicast* podem entrar em laço, ou seja, fazer um percurso circular. O problema é conhecido também em modelos *Unicast*, porém, em razão do *Multicast* ser

Figura B.1: Problemas em roteamento *Multicast*

capaz de duplicar pacotes, as possibilidades de ocorrerem laços são consideravelmente maiores do que em *Unicast*.

- Para cada remetente, diferentes caminhos devem ser adotados. É preciso notar que a utilização de simples caminho inverso de uma rota previamente estabelecida por um outro usuário do grupo *Multicast* geralmente significa um aumento de tráfego desnecessário na rede.

Alguns dos problemas supra expostos serão detalhados nas próximas seções.

B.1.2 Dificuldades de roteamento

Um dos problemas mais difíceis enfrentados para o roteamento de um pacote *Multicast* é o laço. Caso os protocolos *Multicast* não suportem mecanismos de detecção de laços, um tráfego excessivo pode ser gerado e comprometer severamente o desempenho do sistema.

A Figura B.1 apresenta exemplo de laço e de duplicamento de dados.

No cenário apresentado pela Figura B.1, observa-se que não apenas os roteadores devem encaminhar os pacotes de uma forma correta, mas também evitar duplicações e laços. No exemplo, o roteador R4, em razão de saber que havia usuários na rede 2, permitiu o encaminhamento do pacote para a rede 5, duplicando a informação. Conseqüentemente, o roteador R3 encaminhou para a rede 4 o mesmo pacote duas vezes.

B.1.2.1 RPF

RPF (*Reverse path forwarding*) é uma técnica para prevenir que o datagrama viaje em repetidos laços até o TTL (*Time to Live*) expirar. O RPF consiste em enviar um datagrama pela interface de rede para identificação de laços.

Porém, embora o RPF previna laços na rede, ele não previne a duplicação do pacote em uma mesma sub-rede.

B.1.3 Árvore *Multicast* (*Multicast Tree*)

Para realizar o roteamento de pacotes *Multicast*, muitos pesquisadores baseiam-se na teoria dos grafos para descrever o caminho de uma fonte a todos os membros (Arya et al., 2005).

A Figura B.2 apresenta um cenário composto por sete roteadores, oito *hosts* e seis membros.

Cada árvore define o caminho de sua fonte a todos os seus membros, ou seja, para o computador A enviar um pacote *Multicast* para os usuários B, C, D, E e F, utiliza-se uma distribuição em forma de árvore para realizar a entrega.

Para cada remetente, devem haver diferentes árvores. No mesmo exemplo (B.2), supomos que o usuário F envie um pacote *Multicast* ao grupo (A, B, C, D e E). As tabelas de roteamento deverão estabelecer outro comportamento, pois caso simplesmente sigam o fluxo inverso da árvore de envio do usuário A, haverá uso desnecessário de banda devido à localidade de F em relação aos destinatários.

Figura B.2: Roteamento *Multicast*

Um algoritmo de roteamento *Multicast* deverá estabelecer diferentes árvores para cada remetente e, assim, cada roteador deverá conter uma tabela de roteamento com a lista dos remetentes (origem) e suas rotas.

O problema surge devido ao tamanho da tabela de roteamento mantida por cada roteador da rede. Igualmente, em protocolos *Unicast*, uma primeira otimização é armazenar apenas o sufixo de rede das fontes e destinatários em vez do endereço IP e, desta forma, é possível agrupar usuários de mesma rede que apresentam um mesmo comportamento.

B.1.4 Túnel *Multicast* (*Multicast Tunneling*)

Um dos principais problemas na utilização do *Multicast* é que poucas redes suportam os datagramas *Multicast*. Porém, para permitir o seu uso em trechos não habilitados, cria-se um túnel entre dois roteadores com *Multicast* ativo através de outros roteadores sem este serviço habilitado.

O protocolo UMTF provê esta capacidade (Rahman, 2002, p. 400) e é amplamente utilizado em MBONE (*Multicast Backbone*). O MBONE representa uma grande rede *Multicast* e consiste em rede de roteadores *Multicast* interconectados ponto-a-ponto através de *Multicast Tunneling*.

O conceito é simples: o pacote *Multicast*, no limite de uma rede sem suporte a *Multicast*, é encapsulado dentro de um datagrama *Unicast* UDP e enviado ao próximo roteador com

Figura B.3: *Multicast Tunneling*

o serviço *Multicast* ativo. Após atravessar a rede em *Unicast*, o datagrama *Multicast* é extraído do pacote *Unicast* e, assim, continua normalmente o seu procedimento de entrega. A Figura B.3 apresenta um cenário que exemplifica o empacotamento *Multicast* em pacote *Unicast* para tornar possível o seu uso em uma rede puramente *Unicast*.

B.2 Protocolos *Multicast*

Nesta seção serão apresentados resumidamente alguns protocolos *Multicast*.

B.2.1 DVMRP

Distance Vector Multicast Routing Protocol (DVMRP), definido em Waitzman et al. (1988), é um dos primeiros e mais conhecidos protocolos de roteamento *Multicast*. É

usado para troca de informações entre roteadores e grupos. Ademais, usa uma variação do protocolo RIP (Hedrick, 1988) e gera tabelas de rotas *Multicast* baseadas em distância entre roteadores.

O DVMRP utiliza o IGMP para trocar informações com outros roteadores.

B.2.2 CBT

Diferente do DVMRP, o CBT, definido em Ballardie (1997), evita o envio de dados de controle desnecessários, ou seja, não envia datagramas por uma interface até um ou mais *hosts* requisitarem explicitamente a sua inclusão ao grupo. Trata-se de *demand-driven*: quando um *host* usa o IGMP para unir-se a um particular grupo, o roteador local deve informar a outros roteadores para que estes comecem a receber datagramas.

O CBT elege um roteador por região, dinamicamente (*discovery mechanism*). O roteador eleito é conhecido como *Core Router* (CR).

Os demais roteadores devem conhecer o *Core Router* de sua região. A Figura B.4 apresenta exemplo de cenário CBT, onde os CR (delimitados por uma linha pontilhada) estão indicados por uma flecha.

B.2.3 PIM

PIM (Wittmann; Zitterbart, 2000, p. 105) consiste em dois protocolos independentes. Esta bipartição surge em função do trabalho diferente de cada protocolo em cada ambiente distinto.

- **PIM - Dense Mode:** Pequenas áreas (Adams, 2005).
- **PIM - Sparse Mode:** Grande áreas (Estrin et al., 1998).

Figura B.4: *Core Route*

B.2.3.1 PIM-DM (*Dense Mode*)

Provê baixo atraso para redes que têm grande banda.

É otimizado para garantir a entrega dos pacotes em vez de diminuir o *overhead*.

Além disso, utiliza o conceito *broadcast-and-prune* (similar a DVMRP). O PIM-DM inicia-se com o uso de RPF para espalhar os datagramas para todos os grupos e somente cessa sua atividade quando recebe uma explícita mensagem de outro roteador.

B.2.3.2 PIM-SM (*Sparse Mode*)

Conserva a preocupação com a banda. É semelhante ao CBT (*Demand-Driven*): os *Core Router* do CBT são chamados de *Rendezvous Point* (RP) no PIM-SD.

A principal diferença entre o CBT e o PIM-SM é que o PIM-SM é capaz de otimizar a conexão através da reconfiguração dinâmica.

Cada roteador mantém alguns roteadores RP potenciais que podem ser selecionados a

Figura B.5: Reestruturação dos RP

qualquer momento. Se um RP torna-se inalcançável ou algum outro roteador torna-se mais eficaz para a entrega dos pacotes *Multicast* em um determinado grupo, o PIM-SM seleciona outro roteador RP para a reconstrução da árvore de distribuição.

A Figura B.5 apresenta exemplo de reconfiguração dos RPs. No cenário, o roteador R7 é um RP e, em um instante seguinte, o sistema estabelece como RP o roteador R1, otimizando a entrega da informação do servidor ao membro.

Referências Bibliográficas

- ADAMS, A. Protocol independent multicast - dense mode. **RFC 3973**, Janeiro 2005.
- ARYA, VIJAY; TURLETTI, THIERRY; KALYANARAMAN, SHIVKUMAR. Encoding of multicast trees. In **4th International IFIP-TC6 Networking Conference**, Canada, Maio 2005.
- BALENSON, DAVID; MCGREW, DAVID; SHERMAN, ALAN. Key management for large dynamic groups: One-way function tree and amortized initialization. In **IRTF SMUG**, Fevereiro 1999.
- BALLARDIE, A. Scalable multicast key distribution. **RFC 1949**, Maio 1996.
- BALLARDIE, A. Core based trees multicast routing. **RFC 2189**, Setembro 1997.
- BANERFEE, S.; BHATTACHARJEE, B. Scalable secure group communication over ip multicast. In **International Conference on Network Protocols**, Novembro 2001.
- BECKER, C.; WILLIE, U. Communication complexity of group key distribution. In **5th ACM Conference on Computer and Communications Security**, Novembro 1998.
- BLUNDO, C.; SANTIS, A. DE; HERZBERG, A.; VACCARO, U.; YUNG, M. Perfectly-secure key distribution for dynamic conferences. In **Information and Computation**, pp. 1–23, Outubro 1998.
- BOYD, C. On key agreement and conference key agreement. In **Information Security and Privacy, Second Australasian Conference, ACISP'97**, pp. 294–302, Julho 1997.
- BRISCOE, B. Marks: Multicast key management using arbitrary revealed key sequences. In **First International Workshop on Networked Group Communication**, Novembro 1999.

- BRUSCHI, DANILO; ROSTI, EMILIA. Secure multicast in wireless network of mobile hosts: Protocols and issues. In **ACM Baltzer MONET Journal**, 2000. Special Issue on Multipoint Communication in Wireless Networks.
- CANETTI, R.; GARAY, J.; ITKIS, G.; MICCIANCIO, D.; NAOR, M.; PINKAS, BENNY. Multicast security: A taxonomy and some efficient constructions. In **Proceedings of IEEE INFOCOM**, volume 2, pp. 708–716, Março 1999.
- CARONNI, GERMANDO; WALDVOGEL, MARCEL; SUN, DAN; PLATTNER, BERNHARD. Efficient security for large and dynamic multicast groups. In **Seventh Workshop on Enabling Technologies (WET ICE'98)**. IEEE Computer Society Press, 1998.
- CHALLAL, YACINE; SEBA, HAMIDA. Group key management protocols: A novel taxonomy. volume 2, 2005.
- CHANDRA, R.; RAMASUBRAMANIAN, V.; BIRMAN, K. P. Anonymous gossip: Improving multicast reliability in mobile ad-hoc networks. In **International Conference On Distributed Computing Systems**, 2001.
- CHANG, I.; ENGEL, R.; KANDLUR, D.; PENDAKARIS, D.; SAHA, D. Key management for secure internet multicast using boolean function minimization techniques. In **Proceedings of IEEE INFOCOM**, volume 2, Março 1999.
- CHIOU, G. H.; CHEN, W. T. Secure broadcast using the secure lock. In **IEEE Transactions on Software Engineering**, pp. 929–934, Agosto 1989.
- CHORZEMPA, MICHAEL WILLIAM. Key management for wireless sensor networks in hostile environments. Master's thesis, Virginia Polytechnic Institute and State University, 2006.
- CHU, H.; QIAO, L.; NAHRSTEDT, K. A secure multicast protocol with copyright protection. In **Proceedings of IS&T/SPIE Symposium on Eletronic Imaging: Science and Technology**, Janeiro 1999.
- DEERING, S.; FENNER, W.; HABERMAN, B. Multicast listener discovery for ipv6. **RFC 2710**, Outubro 1999.
- DEERING, S. E. Host extensions for ip multicasting. **RFC 1112**, Agosto 1989.
- DEKKER, MARCEL. **Compressed Video Over Networks**. Ming-Ting Sun e Amy R. Reibman, 2000.

- DONDETI, L.; MUKHERJEE, S.; SAMAL, A. Scalable secure one-to-many group communication using dual encryption. In **Computer Communication Journal**, 1999a.
- DONDETI, L. R.; MUKHERJEE, S.; SAMAL, A. A distributed group key management scheme for secure many-to-many communications. Technical report, Department of Computer Science, University of Maryland, 1999b.
- DONDETI, L. R.; MUKHERJEE, S.; SAMUEL, A. Comparison of hierarchical key distribution schemes. In **Proceedings fo IEEE Globecom Global Internet Symposium**, Rio de Janeiro, Brasil, Dezembro 1999c.
- DONDETI, LAKSHMINATH; MUKHERJEE, SARIT; SAMAL, ASHOK. A dual encryption protocol for scalable secure multicasting. Technical Report UNL-CSE-1998-001, 1998. Disponível em citeseer.ist.psu.edu/dondeti98dual.html.
- DONDETI, LAKSHMINATH R.; MUKHERJEE, SARIT; SAMAL, ASHOK. A dual encryption protocol for scalable secure multicasting. In **Fourth IEEE Symposium on Computers and Communications**, Julho 1999d.
- DOYLE, JEFF; CARROLL, JENNIFER DEHAVEN. **Routing TCP/IP**. Cisco Press, 2001.
- DUNIGAN, TOM; CAO, CATHY. Group key management. Technical report, Julho 1997. Disponível em <http://www.csm.ornl.gov/~dunigan/gkmp.ps>.
- EASTLAKE, D.; JONES, P. Us secure hash algorithm 1 (sha1). **RFC 3174**, Setembro 2001.
- EDWARDS, BRIAN; GUILIANO, LEONARD; WRIGHT, BRIAN. **Interdomain Multicast Routing**. Addison-Wesley, 2002.
- ELLIS, CLARENCE; WAINER, JACQUES. Computer supported cooperative work. In **Acm Conference On Computer Supported Cooperative Work**, pp. 78–88, 1994.
- ELTOWEISSY, MOHAMED; HEYDARI, M. HOSSAIN; MORALES, LINDA; SUDBOURGH, I. HAL. Combinatorial optimization of group key management. In **Journal of Network and Systems Management**, volume 12, pp. 30–50, 2004.
- ELTOWEISSY, MOHAMED; MOHARRUM, MOHAMMED; MUKKAMALA., RAVI. Dynamic key management in sensor networks. In **Communications Magazine, IEEE**, volume 44, pp. 122–130, Abril 2006.
- ESKICIOGLU, A. M.; ESKICIOGLU, M. R. Multicast security using key graphs and secret sharing. In **IEEE International Conference on Network 2002**, pp. 228–241, Agosto 2002.

- ESKICIOGLU, AHMET M. Multimedia security in group communications: Recent progress in wired and wireless networks. In **Proceedings of the IASTED International Conference on Communications and Computer Networks**, pp. 125–133. Cambridge, Novembro 2002.
- ESKICIOGLU, AHMET M. Multimedia security in group communications: Recent progress in key management, authentication, and watermarking. In **ACM Multimedia Systems Journal**, volume 1, pp. 239–248, Setembro 2003.
- ESKICIOGLU, AHMET M.; DEXTER, SCOTT; DELP, EDWARD J. Protection of multicast scalable video by secret sharing: simulation results. In **Security and Watermarking of Multimedia Contents V**, volume 5020, pp. 505–515, 2003.
- ESTRIN, D.; FARINACCI, D.; HELMY, A.; THALER, D.; DEERING, S.; HANDLEY, M.; JACOBSON, V.; LIU, C.; SHARMA, P.; WEI, L. Protocol independent multicast-sparse mode. **RFC 2362**, Junho 1998.
- FENNER, W. Internet group management protocol v2. **RFC 2236**, 1997.
- GENNARO, R.; ROHATGI, P. How to sign digital streams. In SPRINGER-VERLAG, editor, **Advances in Cryptology – CRYPTO**, pp. 180–197, Agosto 1997.
- GITLIN, RICHARD D.; HAYES, JEREMIAH F.; WEINSTEIN, STEPHEN B. **Data Communication Principles**. Springer, 1992.
- GOLLE, P.; NODADUGU, N. Authenticating streamed data in the presence of random packet loss. In **NDSS - Network and Distributed System Security Symposium**, pp. 13–22, Fevereiro 2001.
- GONG, L.; SHACHAM, N. Elements of trusted multicasting. In **Proceedings of the IEEE International Conference on Network Protocols**, pp. 23–30, Outubro 1994.
- HARDJONO, T.; CAIN, B.; DORASWAMY, N. A framework for group key management for multicast security. In **IETF Internet draft**, Agosto 2000.
- HARDJONO, T.; TSUDIK, G. Ip multicast security: Issues and directions. In **Annales de Telecom**, pp. 324–334. Addison Wesley, Julho/Agosto 2000.
- HARDJONO, THOMAS; DONDETI, LAKSHMINATH R. **Multicast and Group Security**. Artech House, 2003.
- HARNEY, H.; MUCKENHIRN, C. Group key management protocol (gkmp). **RFC 2094**, Julho 1997.

- HEDRICK, C. Routing information protocol. **RFC 1058**, Junho 1988.
- IANA. Internet assigned names authority. Disponível em <http://www.iana.org/assignments/multicast-addresses>. Acesso em 14 mai. 2006.
- KARP, RICHARD M. Reducibility among combinatorial problem. In **Complexity of Computer Computations, Proc. Sympos. IBM**, pp. 85–103, 1972.
- KIM, Y.; PERRIG, A.; TSUDIK, G. Simple and fault-tolerant key agreement for dynamic collaborative groups. In **7th ACM Conference on Computer and Communications Security**, Novembro 2000.
- KRAWCZYK, M.; BELLARE, M.; CANETTIR. . Hmac: Keyed-hashing for message authentication. **RFC 2104**, Fevereiro 1997.
- L. R. DONDETI, S. MUKHERJEE; SAMAL, A. Survey and comparsion of secure group communication protocols. Technical report, University of Nebraska-Lincoln, Junho 1999.
- MERKLE, R. C. A certified digital assignature. In **Advances in Cryptology**, pp. 218–238, Agosto 1989. Springer-Verlag.
- MINER, S.; STADDON, J. Graph-based authentication of digital streams. In **IEEE Symposium on Security and Privacy**, pp. 234–246, Maio 2002.
- MITTRA, S. Iolus: A framework for scalable secure multicasting. In **Proceeding of the ACM SIGCOMM 97**, pp. 277–288, Setembro 1997.
- MOHARRUM, M.; MUKKAMALA, R.; ELTOWEISSY, M. Ckds: An efficient combinatorial key distribution scheme for wireless ad-hoc networks. In **Performance, Computing, and Communications, 2004 IEEE International**, 2004.
- MOHARRUM, MOHAMMED A.; ELTOWEISSY, MOHAMED. A study of static versus dynamic keying schemes in sensor networks. In **PE-WASUN '05: Proceedings of the 2nd ACM international workshop on Performance evaluation of wireless ad hoc, sensor, and ubiquitous networks**, New York, NY, USA, 2005.
- MOLVA, R.; PANNETRAT, A. Scalable multicast security in dynamic groups. In **6th ACM Conference on Computer and Communications Security**, pp. 101–112, Novembro 1999.
- MORALES, LINDA; SUDBOROUGH, I. HAL; ELTOWEISSY, MOHAMED; HEYDARI, M. HOSSAIN. Combinatorial optimization of multicast key management. In **Proceedings of the 36th Hawaii International Conference on System Sciences (HICSS'03)**, 2003.

- MOY, JOHN T. **Ospf: Anatomy of an Internet Routing Protocol**. Addison-Wesley Professional, 2 de Fevereiro 1998.
- NS-2. Network simulator. Disponível em <http://www.isi.edu/nsnam/ns/>. Acesso em 06 jul. 2006.
- PERRIG, A. Efficient collaborative key management protocols for secure autonomous group communication. In **International Workshop on Cryptographic Techniques and E-Commerce (CryptTEC'99)**, pp. 192–202, 1999.
- PERRIG, A. Efficient authentication and signing of multicast streams over lossy channels. In **IEEE Symposium on Security and Privacy**, pp. 56–73, Maio 2000a.
- PERRIG, A. Tesla: Multicast source authentication transform. In **IRTF**, Novembro 2000b.
- POOVENDRAN, R.; AHMED, S.; CORSON, S.; BARAS, J. A scalable extension of group key management protocol. Technical report, Institute for Systems Research, 1998.
- RAFAELI, S. A decentralized architecture for group key management. In **Computing Department**. Lancaster University, Setembro 2000.
- RAFAELI, SANDRO; HUTCHISON, DAVID. A survey of key management for secure group communication. In **ACM Computing Surveys (CSUR)**, volume 35, pp. 309–329. ACM Press, Setembro 2003.
- RAHMAN, SYED MAHBUBUR. **Multimedia Networking**. Idea Group Inc (IGI), Janeiro 2002.
- RIVEST, R. The md5 message-digest algorithm. **RFC 1321**, Abril 1992.
- RIVEST, R.; SHAMIR, A.; ADLEMAN, L. A method for obtaining digital signatures and public-key cryptosystems. In **Communications of the ACM**, volume 21, pp. 120–126, 1978.
- RODEH, O.; BIRMAN, K.; DOVEL, D. Optimized group rekey for group communication system. In **Network and Distributed System Security Symposium**, Fevereiro 2000.
- RUIYING, DU; HUIJUAN, TU; SONG, WEN. An efficient key management scheme for secure sensor networks. In **Parallel and Distributed Computing, Applications and Technologies PDCAT**, 2005.
- SAMUEL T. REDWINE, JR.; MADISON, JAMES. A logic for the exclusion basis system. In **37th Annual Hawaii International Conference on System Sciences (HICSS'04)**, 2004.

- SCHEIKL, O.; LANE, J.; BOYER, R.; ELTOWEISSY, M. Multilevel secure multicast: The rethinking of secure locks. In **Proceeding of the 2002 ICPP Workshop on Trusted Computer Paradigms**, Agosto 2002.
- SCHNEIER, B. **Applied Cryptography Second Edition: protocols, algorithms, and source code in C**. Wiley, 1996.
- SETIA, S.; KOUSSIH, S.; JAJODIA, S. Kronos: A scalable group re-keying approach for secure multicast. In **IEEE Symposium on Security and Privacy**, Maio 2000.
- STEINER, M.; TSUDIK, G.; WAIDNER, M. Cliques: A new approach to group key agreement. Technical report, IBM Research, Dezembro 1997.
- THAYER, R.; DORASWAMY, N.; GLENN, R. Ip security. **RFC 2411**, 1998.
- WAITZMAN, D.; PARTRIDGE, C.; DEERING, S. Distance vector multicast routing protocol. **RFC 1075**, Novembro 1988.
- WALDVOGEL, M.; CARONNI, G.; SUN, D.; WEILER, N.; PLATTNER, B. The versakey framework: Versatile group key management. In **JSAC Special Issue on Middleware**, pp. 1614–1631, Agosto 1999.
- WALLNER, D.; HARDER, E.; AGEE, R. Key management for multicast: Issues and architectures. **RFC 2627**, Junho 1999.
- WILLIAMSON, BEAU. **Developing IP Multicast Networks**, volume 1. Cisco Pres, 19 de Outubro 1999.
- WILLIAMSON, BEAU. **Developing IP Multicast Networks**, volume 1. Cisco Press, 2000.
- WITTMANN, RALPH; ZITTERBART, MARTINA. **Multicast Communication: Protocols, Programming, & Applications**. Morgan Kaufmann, 12 de Maio 2000.
- WONG, C. K.; GOUDA, M. G.; LAM, S. S. Secure group communications using key graphs. Technical report, Department of Computer Sciences, The University of Texas at Austin, Julho 1997.
- ZHU, SENCUN; SETIA, SANJEEV; JAJODIA, SUSHIL. Performance optimizations for group key management schemes for secure multicast. In **23rd IEEE International Conference on Distributed Computing Systems (ICDCS 2003)**, Maio 2003.

Índice Remissivo

Esquema de Gerenciamento de Chaves

- CTKM, 34, 50, 104
- DEP, 34, 50, 64, 104
- EBCK, 5
- EBS, 50, 55, 104
- lolus, 34, 50, 59, 104

Algoritmo Guloso, 83

CKMSS, 50

Cobertura Mínima, 73, 83

CRC, 20

Criptografia

- AES, 26
- DES, 26
- Hash, 45–47, 74, 78
- HMAC, 21, 44
- MAC, 21, 47
- MD5, 78
- RSA, 4
- SHA-1, 47, 78

CSCW, 1

CTKM, 34, 50, 104

- Entrada de usuário, 53
- Estrutura de Chaves, 51
- Exclusão de usuário, 52
- Renovação de Chaves, 55

DEP, 34, 50, 64, 104

- Entrada de usuários, 67
- Exclusão de usuários, 69

EBCK, 5, 72, 104

EBS, 50, 55, 104

Entrada de usuários, 58

Exclusão de usuários, 57

Matriz de chaves, 58

Uso de funções *one-way*, 57

GPRS, 3

Groupware, 1

Bate-Papo, 1

Email, 1

Fórum, 1

Mensagem Instantânea, 1

Skype, 2

Videoconferência, 1

WebTV, 39, 41

Wiki, 1

GSA, 42

Hamming Code, 19

Hash, 4

MD5, 4

SHA-1, 4

ICMP, 14, 15

IGMP, 15, 22

Check Sum, 16

Resp Time, 16

Type, 16

lolus, 34, 50, 59, 104

Entrada de usuários, 62

Exclusão de usuários, 63

IPSec, 3

- Mensagem
 - ACK, 14
 - TTL, 14
- MLD, 15, 16
 - Versão 1, 17
 - Versão 2, 17
- Multicast
 - Designated Router, 19
 - Echo, 14
 - Echo Reply, 14
 - Faixa de Endereçamento, 14
 - GSA, 42
 - ICMP, 14, 15
 - IGMP, 15, 22
 - MLD, 15, 16
 - NACK, 18
 - TTL, 14
- NS2, 104
- PDA, 3, 48
- Protocolo
 - CBT, 59
 - DVMRP, 22, 59
 - ICMP, 14, 15
 - IGMP, 15, 22
 - MLD, 15, 16
 - PIM, 59
 - PIM-DM, 22
 - TCP, 14, 18
 - TESLA, 47
 - UDP, 14, 15, 20, 46
- RAID, 19
- Segurança
 - Autenticidade, 4
 - Certificadora Digital, 4
 - Confidencialidade, 4
 - Controle de Acesso, 4
 - Denial Of Service, 5
 - Eavesdropping, 70
 - GSA, 42
 - Integridade, 4
 - IPSec, 3
 - Irretratabilidade, 4
 - KDC, 26
- Simulador
 - NS2, 104
- Smartphone, 3