
ESMP: UM PROTOCOLO DE SEGURANÇA MULTICAST PARA UMA ARQUITETURA DE INTERNET DO FUTURO

Juliano Coelho Gonçalves de Melo



UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE COMPUTAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Uberlândia
2019

Juliano Coelho Gonçalves de Melo

**ESMP: UM PROTOCOLO DE SEGURANÇA
MULTICAST PARA UMA ARQUITETURA
DE INTERNET DO FUTURO**

Dissertação de mestrado apresentada ao Programa de Pós-graduação da Faculdade de Computação da Universidade Federal de Uberlândia como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

Área de concentração: Ciência da Computação

Orientador: Prof. Flávio de Oliveira Silva, Ph.D.

Uberlândia

2019

Dados Internacionais de Catalogação na Publicação (CIP)
Sistema de Bibliotecas da UFU, MG, Brasil.

M528e Melo, Juliano Coelho Gonçalves de, 1979-
2019 ESMP [recurso eletrônico] : um protocolo de segurança multicast
para uma arquitetura de Internet do Futuro / Juliano Coelho Gonçalves
de Melo. - 2019.

Orientador: Flávio de Oliveira Silva.

Dissertação (mestrado) - Universidade Federal de Uberlândia,
Programa de Pós-Graduação em Ciência da Computação.

Modo de acesso: Internet.

Disponível em: <http://dx.doi.org/10.14393/ufu.di.2019.342>

Inclui bibliografia.

Inclui ilustrações.

1. Computação. 2. Internet - Sistemas de segurança. 3. Redes
definidas por software. 4. Multicasting (Redes de computadores) -
Medidas de segurança. I. Silva, Flávio de Oliveira, 1970- (Orient.) II.
Universidade Federal de Uberlândia. Programa de Pós-Graduação em
Ciência da Computação. III. Título.

CDU: 681.3

Maria Salete de Freitas Pinheiro - CRB6/1262

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE COMPUTAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Os abaixo assinados, por meio deste, certificam que leram e recomendam para a Faculdade de Computação a aceitação da dissertação intitulada **"ESMP: UM PROTOCOLO DE SEGURANÇA MULTICAST PARA UMA ARQUITETURA DE INTERNET DO FUTURO"** por **Juliano Coelho Gonçalves de Melo** como parte dos requisitos exigidos para a obtenção do título de **Mestre em Ciência da Computação**.

Uberlândia, 21 de fevereiro de 2018

Orientador: _____

Prof. Dr. Flávio de Oliveira Silva
Universidade Federal de Uberlândia

Banca Examinadora:

Prof. Dr. Cesar Augusto Cavalheiro Marcondes
Universidade Federal de São Carlos

Prof. Dr. Rodrigo Sanches Miani
Universidade Federal de Uberlândia

*Pensar é o trabalho mais difícil que existe.
Talvez por isso tão poucos se dediquem a ele.*

Agradecimentos

Agradeço à minha família, em especial minha mãe Rosângela, minha avó Archidamia, meu avô Borgico e à Jane (futura esposa) pelo amor incondicional em todos os caminhos que resolvi percorrer no âmbito pessoal, profissional e acadêmico. No fim aprendemos que família e conhecimento são as maiores riquezas.

Agradeço aos colegas do grupo MEHAR, pela troca de experiências e aprendizado; um agradecimento especial ao meu amigo Pedro Frosi Rosa, cuja paciência nas discussões sem fim é infinita; obrigado pelos ensinamentos técnicos e pelos conselhos sábios que me deu no decorrer dessa jornada.

Não poderia deixar de fora um grande colega de profissão, Natal Vieira de Souza Neto. Sua generosidade profissional é ímpar e devo deixar registrado que aprendi muito, e tenho certeza que continuarei aprendendo com você.

Agradeço aos docentes e técnicos da Faculdade de Computação pelo ensinamento e suportes prestados nos últimos anos. É uma honra poder dizer que, após esses anos, faço parte de uma comunidade tão importante.

Deixo um forte abraço para Sônia e Erisvaldo, integrantes da secretaria de pós-graduação da Faculdade de Computação; sempre se mostraram gentis, competentes e prontos para resolver quaisquer problemas; e foram muitos. São dois grandes amigos que fiz e levarei por toda a vida.

Por fim gostaria de deixar um agradecimento especial ao meu amigo e orientador Flávio de Oliveira Silva, cuja perseverança e espírito de trabalho nos influencia a ser profissionais cada vez melhores. Dentro de qualquer instituição de ensino, esses valores são fundamentais.

*“A imaginação é mais importante que a ciência, porque a ciência é limitada, ao passo
que a imaginação abrange o mundo inteiro”.
(Albert Einstein)*

Resumo

A Internet mostrou-se incapaz de responder aos novos requisitos (QoS, mobilidade, multicasting, segurança, etc.) demandados pelo surgimento de novas aplicações, dispositivos e serviços na rede mundial de computadores. Essas limitações levaram pesquisadores do mundo inteiro a pensar em novas arquiteturas de rede. Essas arquiteturas são chamadas de arquiteturas de Internet do Futuro e sua principal função é suprir às demandas das aplicações atuais e futuras. O Brasil possui algumas iniciativas e uma delas é a Entity Title Architecture (ETArch). Dentre seus principais objetivos, podemos citar a capacidade de fazer comunicação multicast de forma intrínseca e fazer uma aproximação semântica entre as suas camadas, de tal forma que os requisitos das entidades comunicantes (aplicações, sensores, etc.) sejam considerados pelas camadas intermediárias no estabelecimento da comunicação.

No que tange à essas deficiências, a segurança é pré-requisito para a implantação de qualquer arquitetura. Por outro lado, multicasting é essencial à proliferação de aplicações de mídias digitais, jogos multiplayer, etc. A motivação desse trabalho está na intenção de resolver esses dois requisitos simultaneamente. O objetivo é construir a especificação de um protocolo de segurança multicast (ESMP) que transforme um ambiente de comunicação multicast em uma rede de confiança. Entende-se por rede de confiança, um ambiente onde as entidades possam confiar uma nas outras a fim de realizar uma comunicação segura. Esse objetivo passa pela criação de vários serviços/mecanismos de segurança, tais como confidencialidade, integridade, gerenciamento de chaves, disponibilidade e autenticação.

Essa especificação foi aplicada à arquitetura ETArch. Essa escolha deveu-se às suas características de oferecer multicasting de forma intrínseca, de ser altamente flexível quanto às necessidades das aplicações e de ter uma visão muito próxima da abstração proposta pelas Redes Definidas por Software. Como hipótese, assumiu-se que o ambiente de comunicação seguro das informações deve ser definido antes mesmo que os dados comecem a ser trafegados, ou seja, a proteção das informações transmitidas no plano de dados se dará quando as operações necessárias para o estabelecimento do ambiente seguro de comuni-

cação multicast já tiverem sido realizadas pelo plano de controle. As Redes Definidas por Software e tecnologias como OpenFlow viabilizam essa hipótese.

Neste trabalho, foi definida a especificação de segurança multicast do protocolo ESMP, e também, foram demonstrados através de métodos de análise e avaliação os serviços/mecanismos de segurança propostos. Alguns resultados são obtidos: o ESMP consegue atenuar grande parte dos ataques modelados através do método de análise e avaliação, tais como ataques de espionagem, ataques de modificação de mensagens, ataques de reflexão, ataques de mascaramento, etc.; o ESMP consegue oferecer serviços e mecanismos de segurança que competem com os principais protocolos de segurança da arquitetura legada e com o MobilityFirst; os serviços/mecanismos de segurança do ESMP suportam o contexto de comunicação multicast.

Palavras-chave: Segurança multicast. Arquitetura de Internet do Futuro. Redes Definidas por Software.

Abstract

The internet has shown itself incapable of responding to the new requirements (QoS, mobility, multicasting, security, etc.) demanded by the emergence of new applications, devices, and services in the global computer network. These limitations have led researchers around the world to think of new network architectures. These architectures are called Future Internet Architectures, and their primary function is to meet the demands of current and future applications. Brazil has some initiatives, and one of them is the Entity Title Architecture (ETArch). Among its main objectives, we can mention the ability to make multicast communication intrinsically and to make a semantic approximation between its layers, in such a way that the intermediary layers consider the requirements of the communicating entities (applications, sensors, etc.) in the establishment of communication.

About these deficiencies, security is a prerequisite for the deployment of any architecture. On the other hand, multicasting is essential to the proliferation of digital media applications, multiplayer games, etc. The motivation for this work is intended to solve these two requirements simultaneously. The goal is to build a specification for a multicast security protocol (ESMP) that transforms a multicast communication environment into a trusted network, where entities can trust one another to make secure communication. This goal involves the creation of various security services/mechanisms, such as confidentiality, integrity, key management, availability, and authentication.

This specification was applied to the ETArch architecture. This choice was due to its characteristics to offer intrinsic multicasting, being highly flexible regarding the needs of applications and having a very close view of the abstraction proposed by Software Defined Networks. We assumed that the environment of secure communication of information must be defined even before the data transmission, which means, the protection of the information transmitted in the data plan will be given when the control plan has already performed the operations necessary for the establishment of the secure multicast communication environment. Software Defined Networks and technologies like OpenFlow make this hypothesis viable.

In this work, the ESMP multicast security specification was defined, and the proposed security services/mechanisms were also demonstrated through analysis and evaluation methods. Once the security environment is established, the communications made in the control/data plan are protected from imminent attacks on the network. Some results are obtained: ESMP can mitigate much of the attacks modeled by the method of analysis and evaluation, such as snooping attacks, message modification attacks, reflection attacks, masquerading attacks, etc .; ESMP can provide security services and mechanisms that compete with the major security protocols of the legacy architecture and with MobilityFirst; the ESMP security services/mechanisms support the multicast communication context.

Keywords: multicasting security. Future Internet Security. Software Defined Networking.

Lista de ilustrações

Figura 1 – Modos de utilização do IPSec	46
Figura 2 – Arquitetura de rede definida por software (SDN)	50
Figura 3 – OpenFlow	51
Figura 4 – Arquitetura ETArch	59
Figura 5 – Representação do DTS	61
Figura 6 – Exemplo de topologia da ETArch	62
Figura 7 – Organização do DTS em escala mundial	66
Figura 8 – Arquitetura da integração do ODTONE com o DTSA	70
Figura 9 – Operações genéricas do ESMP	81
Figura 10 – Conexão peer to peer com utilização de unicast múltiplo	82
Figura 11 – Comparação da escala de chaves de uma conexão <i>multicast</i> ETArch e uma conexão <i>unicast</i> múltiplo do TCP/IP	83
Figura 12 – Distribuição de chaves dinâmica da arquitetura ETArch	86
Figura 13 – Operação de encapsulamento do protocolo ESMP	93
Figura 14 – Formato da primitiva de segurança do ESMP	95
Figura 15 – Diagrama de sequência do <i>handshake</i> de sessão	98
Figura 16 – Diagrama de sequência do handshake de conexão	104
Figura 17 – Operação de hash do protocolo ESMP	109
Figura 18 – Envio de primitivas ESMP para a rede mundial de comunicação ETArch	114

Lista de tabelas

Tabela 1	– Mecanismos de mitigação de ataques do protocolo SSL/TLS	44
Tabela 2	– Mecanismos de mitigação de ataques do protocolo IPSec	47
Tabela 3	– Mecanismos de mitigação de ataques da arquitetura <i>MobilityFirst</i> . . .	55
Tabela 4	– Sequência de passos na solicitação de um <i>attach</i>	72
Tabela 5	– Mecanismos de mitigação de ataques da arquitetura ETArch (status quo)	76
Tabela 6	– Tipo de mensagens do protocolo ESMP	95
Tabela 7	– Mecanismos de mitigação de ataques do ESMP na ETArch	121
Tabela 8	– Comparação do ESMP com outros protocolos e arquiteturas	123

Lista de siglas

ACL Access Control List

AES Advanced Encryption Standard

ALM Application Layer Multicast

ARPANET Advanced Research Projects Agency Network

AS Autonomous System

CA Certification Authority

CBC Cipher Block Chaining

CDC Centro de Distribuição de Chaves

CERT Centro de Estudos para Resposta e Tratamento de Incidentes em Computadores

CEF Caixa Econômica Federal

DES Data Encryption Standard

DFD Data Flow Diagrams

DoS Denial of Service

DTS Domain Title Service

DTSA DTS Agent

DTSCP Domain Title Service Control Protocol

EDOBRA ExtenDing and Deploying Ofelia in BRAzil

EM EntityManager

ESMP Entity Security Multicast Protocol

ETArch Entity Title Architecture

ETCP Entity Title Control Protocol

FIA Future Internet Architecture

FP7 Seventh Framework Programme

GLL Generic Link Layer

GNRS Global Name Resolution Service

GNS Global Name Service

GUID Globally unique identifiers

HMAC Hashed Message Authentication Code

HRN Human-Readable-Names

HTTP Hypertext Transfer Protocol

IDS Intrusion Detection System

IEEE Institute of Electrical and Electronics Engineers

IPS Intrusion Prevention System

IPSec IP Security Protocol

ISP Internet Service Provider

ITU International Telecommunication Union

ITU-T ITU Telecommunication Standardization Sector

IXP Internet Exchange Point

JAIN SLEE Java APIs for Integrated Networks - Service Logic Execution Environment

LAN Local Area Network

LSA Link-State Advertisements

LSP Label Switch Path

MAC¹ Message Authentication Code

MAC² Machine Access Control

MBONE Multicast BackBone

MD5 Message-Digest-5

MDTSA Master DTS Agent

MICS Media Independent Command Service

MID Multicast Identifier

MIES Media Independent Event Service

MIIS Media Independent Information Service

MIH Media Independent Handover

MIHF MIH Function

MM MobilityManager

NA Network Address

NAS Name Assignment Services

NE Network Element

NIST National Institute of Standards and Technology

NSF National Science Foundation

OFELIA OpenFlow in Europe: Linking Infrastructure and Applications

OSI Open Systems Interconnection

OSPF Open Shortest Path First

PoP Point of Presence

QM QoSManager

QoE Quality of Experience

QoS Quality of Service

RA Resource Adaptor

RFC Request for Comments

RINA Recursive InterNetwork Architecture

RSA Rivest-Shamir-Adleman

SBB Service Building Blocks

SDN Software Defined Networking

SHA Secure Hash Algorithm

SIP Session Initiation Protocol

SLA Service Level Agreements

SM SecurityManager

SMART The Support of Mobile Sessions with High Transport Demand

SSL Security Sockets Layer

TCP Transmission Control Protocol

TLS Transport Layer Security

UFU Universidade Federal de Uberlândia

UML Unified Modeling Language

WAN Wide Area Network

WM WorkspaceManager

VPN Virtual Private Networks

XIA eXpressive Internet Architecture

XML eXtensible Markup Language

Sumário

1	INTRODUÇÃO	25
1.1	Motivação	27
1.2	Objetivos e Desafios da Pesquisa	28
1.3	Hipótese	29
1.4	Contribuições	29
1.5	Organização da Dissertação	30
2	FUNDAMENTAÇÃO TEÓRICA	33
2.1	Aspectos Conceituais de Segurança	33
2.1.1	Confidencialidade	35
2.1.2	Integridade	35
2.1.3	Disponibilidade	36
2.1.4	Autenticação	36
2.1.5	Política de Segurança	37
2.2	Limitações da arquitetura TCP/IP	38
2.3	Principais protocolos de segurança da arquitetura TCP/IP	42
2.3.1	SSL/TLS	42
2.3.2	IPSec	45
2.4	Redes Definidas por Software	49
2.4.1	OpenFlow	50
2.5	Proposta para Internet do Futuro	52
2.5.1	MobilityFirst	52
3	VISÃO GERAL DA ARQUITETURA ETARCH E SEUS CON- CEITOS PRINCIPAIS	57
3.1	Camadas ETArch	58
3.2	Entidade	59
3.3	Título	60

3.4	Domain Title Service	60
3.5	Workspace	61
3.6	DTSA	63
3.6.1	MDTSA	63
3.7	Workspace de dados	64
3.8	Workspace de controle	65
3.9	Implementação atual da ETArch	68
3.9.1	Implementação do DTSA no EDOBRA	68
3.9.2	Visão geral do ETCP/DTSCP	73
3.10	Segurança na ETArch: status quo	75
4	PROPOSTA DE UM PROTOCOLO DE SEGURANÇA <i>MULTICAST</i> PARA UMA ARQUITETURA DE INTERNET DO FUTURO	79
4.1	Sintetização de chaves	81
4.2	Gerenciamento de chaves	84
4.3	Entity Security Multicast Protocol - ESMP	88
4.3.1	Modelagem das políticas de segurança	90
4.3.2	Encapsulamento e Desencapsulamento	93
4.3.3	Operação de <i>handshake</i>	96
4.4	Comunicação no plano de dados	107
4.5	Operação de hash do protocolo ESMP	108
5	ANÁLISE E DISCUSSÃO DOS RESULTADOS	111
5.1	Método para a análise e avaliação de segurança do ESMP	112
5.2	Análise e avaliação de segurança do ESMP	114
5.2.1	Definição das metas de segurança do ESMP	114
5.2.2	Especificação das ameaças/ataques que inibem as metas de segurança do ESMP	115
5.2.3	Análise das soluções de segurança do ESMP e da arquitetura ETArch	117
5.2.4	Combinação dos mecanismos de segurança do ESMP e da ETArch com as ameaças mapeadas pelo processo de análise	117
5.2.5	Segurança multicast na ETArch	122
5.3	Avaliação dos resultados	122
6	CONCLUSÃO	125
6.1	Principais Contribuições	127
6.2	Trabalhos Futuros	128
6.3	Contribuições em Produção Bibliográfica	129
	REFERÊNCIAS	131

Introdução

Os alicerces conceituais que sustentam a rede mundial de computadores, hoje representada pela arquitetura TCP/IP, foram concebidos na década de sessenta. Baseados em algumas iniciativas da época de criar alternativas de comunicação, surge a proposta de comutação de pacotes (LEINER et al., 2009), base da ideia central do que veio a ser a ARPANET (MARILL; ROBERTS, 1966); primeira rede de computadores por comutação de pacotes e ancestral direta da Internet.

A Internet, desde então, passou por uma evolução inimaginável nessas últimas cinco décadas. Impulsionada pelo avanço de processamento de hardware previsto pela Lei de Moore (MOORE, 2006), houve ampliação da largura de banda e da capacidade de vazão dos enlaces a baixo custo. A malha de elementos de rede estendeu-se pelo mundo inteiro e o número de *hosts* aumentou drasticamente. Novas tecnologias de transmissão foram criadas, tanto "com fio" quanto "sem fio" e foram introduzidos vários protocolos na camada física.

Esse contexto evolutivo criou o ambiente necessário para o desenvolvimento de novas aplicações e serviços que são executados por dispositivos variados; como *laptops*, *tablets*, computadores de escritório, sensores, cartões inteligentes, *smartphones*, entre outros. É importante salientar que o aparecimento dessas novas aplicações, acrescentada à essa evolução tecnológica, trazem basicamente duas implicações: diversificação das tecnologias de transmissão (GSM, 3G, 4G, 5G, Wi-Fi, Ethernet, etc.) e modificação dos padrões de tráfego das redes de computadores (CISCO, 2017). Esses fatos impõem sobre a Internet a necessidade de suportar novos requisitos, tais como *Quality of Service* (QoS), *Quality of Experience* (QoE), mobilidade, segurança, *multicast*, etc.

A arquitetura TCP/IP mostrou-se com dificuldade de responder às novas demandas devido à inexpressividade semântica entre as camadas da pilha de protocolos; à sobrecarga do endereçamento IP (identificador e localizador); e à estrutura rígida das camadas intermediárias, que não permitem a inclusão de novos serviços sem aumentar a complexidade da arquitetura (PEREIRA, 2012).

Essas limitações da Internet levaram pesquisadores do mundo inteiro a pensar em

novos modelos que fomentassem o desenvolvimento de novas arquiteturas. Essas arquiteturas são chamadas de arquiteturas de Internet do Futuro e sua principal função é atender aos requisitos das novas aplicações que surgiram decorrentes da evolução da Internet.

Nesse contexto, vários projetos foram financiados com o objetivo de construir arquiteturas que pudessem se tornar a rede do futuro. O projeto NSF FIA (*Future Internet Architecture*) (NSF, 2010) e o programa FP7 (*Seventh Framework Programme*) são exemplos de financiamentos que fomentaram essa iniciativa (SILVA, 2013). Como exemplo de projetos, podemos citar: MobilityFirst (VENKATARAMANI et al., 2014); *Recursive InterNetwork Architecture* (RINA) (TROUVA et al., 2011) (VRIJDEERS et al., 2014); *eXpressive Internet Architecture* (XIA) (HAN et al., 2012) (NAYLOR et al., 2014); *Entity Title Architecture* (ETArch) (SILVA, 2013); etc.

Todas as arquiteturas citadas tem como objetivo solucionar os gargalos do TCP/IP. No caso da ETArch, foi concebida através de duas principais abordagens teóricas: é uma arquitetura *clean slate* (ROBERTS, 2009), extremamente flexível, capaz de suportar as diversas modificações de requisitos no decorrer do tempo; e, é uma arquitetura baseada nos conceitos de *Software Defined Network* (SDN) (COX et al., 2017), capaz de separar o plano de dados do plano de controle, e desviar o gerenciamento de rede para o software, o que abre grandes possibilidades para as redes atuais e futuras. Dentre os vários gargalos, a ETArch possui a preocupação inicial de oferecer suporte às comunicações *multicast* e realizar uma aproximação semântica entre a camada de aplicação e a camada intermediária para atender os requisitos das aplicações atuais e futuras de forma transparente (SILVA, 2013). No decorrer do seu desenvolvimento, outras preocupações se mostraram iminentes: mobilidade (SILVA, 2013), Qos/QoE (LEMA, 2014) e segurança (MELO et al., 2017a).

Este trabalho apresenta preocupação com dois gargalos da arquitetura legada: *multicasting* e segurança. A primeira preocupação refere-se à proliferação de aplicações de mídias digitais, teleconferências, jogos *multiplayer*, etc. Essa realidade levou a arquitetura TCP/IP a criar uma extensão *multicast* para o IP (DEERING; CHERITON, 1990), porém, não se apresentou eficaz aos seus propósitos de atender os requisitos dessas aplicações. A inutilização desses serviços pelos ISPs corroboram esse fato (HOSSEINI et al., 2007). A segunda preocupação refere-se ao fato de que a arquitetura implantada na Internet, têm necessariamente, que fornecer segurança para as aplicações conforme suas necessidades. É preocupação desse trabalho oferecer mecanismos/serviços de segurança distintos para requisitos distintos de aplicação. Entende-se que essa flexibilidade é importante para que a arquitetura consiga oferecer suporte aos requisitos das aplicações atuais e futuras.

O protocolo *Entity Security Multicast Protocol* (ESMP), proposto nesse trabalho, faz a junção desse dois requisitos para fornecer comunicação *multicast* segura às entidades de rede. A proposta é construir uma rede de confiança, onde as entidades participantes do contexto de comunicação confiem uma nas outras para a realização de uma comuni-

cação segura. Esse propósito passa pela elaboração de vários serviços e mecanismos de segurança.

A especificação do ESMP foi aplicada à arquitetura ETArch. Essa escolha deveu-se principalmente à capacidade da arquitetura em fornecer suporte intrínseco de comunicação *multicast* e de fazer uma aproximação semântica entre as camadas da pilha dos protocolos, de tal forma que as camadas intermediárias reconhecem facilmente os requisitos da aplicação. Outro recurso da ETArch que comprometeu na escolha dessa arquitetura é a facilidade de incluir serviços novos nas camadas intermediárias sem alterar a complexidade da arquitetura.

1.1 Motivação

Na arquitetura Internet tradicional, a comunicação *multicast* é realizada de duas formas: através de uma extensão *multicast* para o IP, denominado IP Multicast (DEERING; CHERITON, 1990); ou através da *Application Layer Multicast* (ALM) (HOSSEINI et al., 2007). No caso do IP Multicast, a implantação desse protocolo não se mostra eficiente devido a problemas técnicos, administrativos e de negócios (HOSSEINI et al., 2007). No entanto, o que ocorreu, de um modo geral, é que o tráfego *multicast* passou a ser tratado pelas ALMs. O grande problema dessa abordagem é que, como de praxe, as demandas de aplicações que deveriam ser resolvidas de modo transparente pelas camadas intermediárias da arquitetura legada são desenvolvidas pelas aplicações. Nesse caso em especial, a desvantagem da ALM é não alcançar a efetividade e eficiência das árvores de comutação do IP Multicast (HOSSEINI et al., 2007).

Um outro problema da Internet tradicional está relacionado à demanda de segurança dessas novas aplicações. Podemos começar pela granularidade oferecida quanto às entidades que participam de uma comunicação na rede. Os serviços de segurança são oferecidos em várias camadas da arquitetura, e por conta disso, há uma sobreposição de funções na pilha de protocolos (KUROSE; ROSS, 2013). Isso provoca um *overhead* de comunicação e uma disfuncionalidade nos princípios filosóficos do modelo de camadas (PEREIRA, 2012). Ainda assim, não consegue prover segurança para usuários, conteúdos, serviços, etc.

Ainda que arquiteturas de Internet do Futuro são motivadas pelos vários gargalos do TCP/IP, elas ainda se mostram com dificuldade de solucionar os dois requisitos citados acima, quais sejam: *multicasting* e segurança. No caso do MobilityFirst (VENKATARAMANI et al., 2014), a comunicação *multicast* não é intrínseca. Na realidade, ela se dá através da realização de cópias de primitivas na origem (*unicast* múltiplo) (FOROUZAN; FEGAN, 2009), e esse processo está detalhado na subseção 2.5.1. Se há dificuldade em oferecer comunicação *multicast* de forma que um endereço represente um subconjunto de entidades, os desafios de se introduzir segurança nessa comunicação é ainda maior (HARDJONO; TSUDIK, 2000).

A motivação desse trabalho está na intenção de resolver esses dois requisitos simultaneamente, ou seja, a proposta é criar um ambiente de comunicação *multicast* seguro. Além disso, é importante que se tenha flexibilidade de escolha dos serviços de segurança oferecidos, ou seja, é importante que entidades comunicantes distintas possuam serviços de segurança distintos. Essa flexibilidade oferece diferentes requisitos para diferentes demandas. Outra motivação é que a segurança e o *multicasting* sejam gerenciados por camadas intermediárias da pilha de protocolos e se mostrem totalmente transparente às entidades comunicantes. A facilidade de introduzir mais serviços de segurança ou a facilidade de modificar serviços de segurança também está presente nas motivações dessa proposta.

Quanto à ETArch, apesar de ter iniciado uma solução em segurança de rede (MELO et al., 2017a), ainda não consegue cumprir objetivos básicos de segurança, tais como confidencialidade, autenticação, disponibilidade e integridade. No decorrer desse trabalho, é apresentado às deficiências de cada uma dessas metas.

No que tange à essas deficiências, o cerne motivacional é elevar o nível de segurança da arquitetura dentro do que se espera de uma arquitetura de Internet do Futuro. Um dos pré-requisitos da implantação de qualquer arquitetura em um ambiente de produção é a análise e avaliação de serviços e mecanismos de segurança que ela oferece. Portanto, é de crucial interesse de qualquer arquitetura que ela tenha um nível de segurança satisfatório.

Em linhas gerais, o que se quer é embutir segurança na conexão *multicast* em uma arquitetura de Internet do Futuro sem os gargalos da arquitetura legada, que atrapalham e são obstáculos para as novas formas de tráfego da rede mundial de computadores. Em consequência, queremos elevar o nível da ETArch em termos de segurança através de uma nova especificação de protocolo para a arquitetura, que tem como meta a implantação de um ambiente seguro de comunicação *multicast* para suas entidades.

1.2 Objetivos e Desafios da Pesquisa

O objetivo geral desse trabalho é construir a especificação de um novo protocolo de segurança *multicast* para uma arquitetura de Internet do Futuro. O protocolo denomina-se *Entity Security Multicast Protocol* (ESMP) e tem a função de transformar um ambiente distribuído de comunicação em uma rede de confiança, onde todas as entidades possam confiar uma nas outras a fim de realizar uma comunicação segura. Para alcançar o objetivo geral citado, propõe-se abaixo alguns objetivos específicos.

- Criar a especificação dos serviços/mecanismos de segurança do ESMP, tais como autenticação, confidencialidade, integridade, disponibilidade e gerenciamento de chaves. Essa especificação, tem, obrigatoriamente, que satisfazer os seguintes requisitos:

- oferecer suporte às conexões *multicast*;

- ser transparente à camada de aplicação;
 - oferecer diferentes serviços/mecanismos de segurança para diferentes entidades comunicantes (sensores, aplicações de banco; jogos *multiplayer*; etc.);
 - sintetizar o número de disseminação de chaves quando comparada com as comunicações *peer-to-peer* da arquitetura legada;
 - elevar o nível de segurança da ETArch (status quo), de tal forma que a arquitetura possa ser utilizada em ambientes de produção.
- Demonstrar as metas do ESMP através de métodos de análise e avaliação de segurança.
- Analisar os resultados obtidos e realizar uma análise comparativa entre o ESMP na ETArch e todos os protocolos e arquiteturas de Internet do Futuro descritas no trabalho, inclusive a ETArch (status quo).

Os desafios de pesquisa estão relacionados à segurança de comunicação *multicast*. Os serviços/mecanismos de segurança de uma comunicação *multicast* tais como gerenciamento de chaves, distribuição de chaves públicas, autenticação mútua das entidades, confidencialidade, e outros; podem ser distintos de uma comunicação comum *peer-to-peer*. Um exemplo é com relação ao caso específico da autenticação. Em uma comunicação *peer-to-peer* fala-se em autenticação de origem. No caso de uma comunicação *multicast*, existe o conceito de autenticação de grupo (HARDJONO; TSUDIK, 2000).

1.3 Hipótese

Acredita-se que com a especificação do ESMP na ETArch, é possível transformar o ambiente distribuído de rede ETArch em uma rede de confiança, onde as entidades possam confiar uma nas outras a fim de estabelecer uma comunicação segura *multicast* entre os participantes pertencentes de um determinado contexto de comunicação.

Como a ETArch se trata de uma arquitetura que utiliza os conceitos SDN, acredita-se ser possível que essa rede de confiança seja elaborada pelo plano de controle. Essa rede visa garantir a proteção da comunicação entre entidades, executando serviços e mecanismos de segurança nas primitivas de controle e de dados transmitidas.

1.4 Contribuições

A especificação de segurança proposta contribui para qualquer arquitetura SDN, inclusive a ETArch. Os serviços/mecanismos de segurança, em SDN, são construídos em uma peça de software externa, denominado controlador. Dessa forma, toda arquitetura que separa o plano de controle do plano de dados pode utilizar esse trabalho como referência.

As demais contribuições do trabalho estão relacionadas à ETArch, que até a conclusão desse projeto, mostrava-se ineficiente na proteção contra ameaças ou ataques iminentes na rede (seção 3.10). Esse trabalho propõe a especificação de serviços/mecanismos de segurança para a ETArch, possibilitando assim a sua evolução e a possibilidade de trabalhos futuros feitos nessa área.

1.5 Organização da Dissertação

O restante desse trabalho segue organizado da seguinte forma:

- o capítulo 2 discute aspectos conceituais de segurança e as limitações da arquitetura TCP/IP em termos de comunicação *multicast* e segurança, requisitos que são o foco da proposta desse trabalho. Depois dessa primeira discussão, é apresentado os principais protocolos de segurança dessa arquitetura e faz-se uma análise da eficiência dos seus serviços/mecanismos quanto aos ataques de rede. Logo após, é introduzido os conceitos de SDN e OpenFlow, além de ser apresentado uma pesquisa relacionada à segurança em redes definidas por software. Por fim, é abordado uma proposta de arquitetura para Internet do Futuro, que também passa por um processo de análise dos seus mecanismos/serviços de segurança;
- o capítulo 3 apresenta a visão geral da arquitetura ETArch, escolhida como arquitetura de Internet do Futuro onde será feita as aplicações das funcionalidades da especificação do protocolo de segurança proposto nesse trabalho. O capítulo descreve os principais conceitos, os componentes, as camadas, os protocolos, os fluxos das primitivas e as implementações da arquitetura. Outro ponto importante é a análise de segurança que se faz da ETArch (status quo). Esse resultado é importante para fundamentar a construção da especificação do ESMP;
- o capítulo 4 descreve a especificação de um novo protocolo *multicast* de segurança de rede, o ESMP; detalhando as metas, os serviços/mecanismos de segurança e as operações desse protocolo. O detalhamento das trocas de primitivas do ESMP com a ETArch no plano de dados/controle também é explanado nesse capítulo.
- o capítulo 5 demonstra através de um método de análise e avaliação de segurança as metas do protocolo ESMP. Nesse capítulo é feita a descrição do método de análise e avaliação utilizado; a execução e aplicação do método de análise e avaliação nos mecanismos de segurança do ESMP; e por fim, é apresentado o resultado do método aplicado. Além disso, o capítulo apresenta uma avaliação comparativa entre o ESMP e outros protocolos/arquiteturas de Internet do Futuro apresentados no decorrer desse trabalho;

- o capítulo 6 conclui este trabalho, apresentando as considerações e contribuições. O trabalho aqui apresentado não é um ponto final, mas um ponto de partida e, portanto, são indicadas algumas sugestões de trabalhos futuros.

Fundamentação Teórica

O presente capítulo, inicialmente, discutirá aspectos conceituais e limitações da arquitetura Internet em realizar certos tipos de comunicações de forma segura. Depois dessa discussão, apresentaremos os principais protocolos de segurança da arquitetura TCP/IP. Logo após, abordaremos os aspectos conceituais de *Software Defined Network* (SDN) (COX et al., 2017) e o protocolo OpenFlow (MCKEOWN et al., 2008). Por fim, abordaremos uma proposta de arquitetura para Internet do Futuro com foco em trabalhos de segurança.

2.1 Aspectos Conceituais de Segurança

O termo segurança de redes de computadores consiste em um conjunto de medidas capazes de manter uma comunicação segura. Essa é uma definição abrangente que traz várias possibilidades. As propriedades básicas de uma rede de comunicação segura são: confidencialidade, integridade, disponibilidade, autenticação e não repúdio (DOULIGERIS; SERPANOS, 2007). Em alguns manuais e referências, *National Institute of Standards on Technology* (NIST) (GUTTMAN; ROBACK, 1995) e X.800.ITU-T (REC, 1991), soma-se a essas propriedades o controle de acesso; a capacidade de manter registros de todas as atividades do sistema; e, a segurança operacional, como *firewalls* e sistemas de detecção de invasão.

Na Internet, o estabelecimento de comunicação segura entre entidades comunicantes, além de atender os requisitos clássicos de segurança descritos acima, precisam trocar mensagens de controle e de dados (algo semelhante a comunicação do TCP). A preparação dessa comunicação não é tão simples, pois os mecanismos que permitem que essa comunicação ocorra de forma segura são complexos e tem que levar em conta aspectos bastante sutis. Esses mecanismos, em suma, são responsáveis pelo desvio, prevenção, detecção, e correção de uma gama grande de ataques e/ou ameaças que podem comprometer os sistemas finais em níveis distintos de gravidade. Em FIPS (2004), é possível verificar, que um impacto alto de quebra de sigilo nos sistemas finais comunicantes pode provocar

grandes perdas financeiras, lesões graves com risco de morte ou até perda de vida. Este trabalho levará em consideração esse níveis de impactos no momento das definições de políticas de segurança.

Uma maneira útil de classificar os ataques se encontram tanto em X.800 ITU-T (REC, 1991), quanto na RFC.4949 (SHIREY, 2007). Entre os ataques passivos, pode-se citar o vazamento de conteúdo de mensagem e a análise de tráfego. Por outro lado, entende-se como ataque ativo o mascaramento/disfarce (*masquerading*), a modificação de mensagens (*modification attack*), repetição/repassa (*replaying attack*), negação de serviços (*Denial of Service - DoS*), dentre outros. O Centro de Estudos para Resposta e Tratamento de Incidentes em Computadores (CERT) (BR, 2018) possui um lista de incidentes reportados, representados por diversos gráficos estatísticos.

Alguns termos que serão adotados nesse trabalho merecem ser esclarecidos. Será utilizado a definição de ameaça e ataque do RFC.4949 (SHIREY, 2007). Ameaça é considerada como sendo uma chance de violação de segurança que existe quando há uma circunstância, capacidade, ação ou evento que poderia quebrar a sistema de segurança e causar danos. Uma ameaça é uma possibilidade de explorar uma vulnerabilidade qualquer. Ataque é um ato inteligente (deliberado) que viola os serviços e a política de segurança de um sistema vulnerável, ou seja, um sistema onde existia uma ameaça. É inevitável, que no decorrer do trabalho, haja uma relação unívoca entre os serviços/mecanismos de segurança e as ameaças/ataques que porventura possam existir por conta de vulnerabilidades, ou seja, os serviços e/ou mecanismos de segurança propostos têm que possuir a capacidade de proteger uma ou mais vulnerabilidades do sistema. Essa relação auxiliará no estudo teórico necessário para desenvolver o módulo de segurança da ETArch.

A RFC.4949 (SHIREY, 2007) e X.800.ITU-T (REC, 1991) define serviço de segurança, como sendo, por exemplo, um serviço de comunicação ou de processamento que é fornecido ao sistema para dar um tipo específico de proteção aos recursos desse sistema. Como exemplo, podemos citar o serviço de segurança entregue pelo *Transport Layer Security* (TLS) à camada de transporte para garantir proteção das informações trafegadas pela rede pública. Mecanismo de segurança seria a forma como o serviço é implementado. Como exemplo, podemos citar novamente o TLS. Sumariamente, essa comunicação se inicia através de um *handshake*. Nessa fase, faz-se por exemplo, as escolhas dos algoritmos criptográficos, algoritmos de *Message Authentication Code* (MAC¹), as trocas que envolvem o certificado e o *nonce* do servidor, etc. Toda essa descrição anterior de como os serviços são oferecidos pelo *TLS* denomina-se mecanismos de segurança, ou seja, os serviços oferecidos são implementados pelos mecanismos de segurança.

Para avaliar as necessidades de segurança atuais, precisa-se de um meio sistemático para definir os requisitos (serviços) e caracterizar as técnicas (mecanismos) para satisfazê-los.

Para fornecer uma rede protegida, uma arquitetura de internet tem que levar em

consideração ataques mal-intencionados, danos não intencionais, e as ameaças existentes no sistema que possam comprometer a comunicação. Algumas metas importantes desse trabalho são: confidencialidade, integridade, disponibilidade e autenticação.

Outra meta importante desse trabalho está relacionada ao fornecimento de serviços de segurança de acordo com as demandas das aplicações. A política de segurança é o mecanismo que possibilita essa flexibilidade na oferta desses serviços.

2.1.1 Confidencialidade

Confidencialidade (REC, 1991) é a proteção dos dados contra divulgação não autorizada, ou seja, apenas entidades comunicantes autorizadas podem ver o conteúdo da mensagem. Para qualquer outra entidade não autorizada, o conteúdo da mensagem deve ser inútil. Dessa forma, essa mensagem deve estar cifrada.

Esse serviço de segurança protege contra ataques passivos (STALLINGS, 2014). Um dos ataques resolvidos tem relação ao vazamento de conteúdo. O invasor consegue interceptar a mensagem (*snooping attack*), mas não consegue lê-la. Outro benefício adquirido com esse serviço está relacionado ao ataque de análise de tráfego. Nesse ataque, o invasor intercepta a mensagem para extrair informações de padrões de tráfego. Quanto maior o montante de primitivas analisadas, mais efetivas são as deduções adquiridas. Essa proteção exige que o invasor não consiga observar origem e destino, frequência, tamanho e/ou outras características de tráfego. O algoritmo de criptografia utilizado é predominante na falta de padronização dessas primitivas (STALLINGS, 2014).

2.1.2 Integridade

Integridade (REC, 1991) é um serviço de comunicação que garante que a mensagem recebida é exatamente a mesma que foi enviada por uma entidade autorizada, ou seja, essa mensagem não contém nenhuma alteração de qualquer espécie, e também, não se trata de um repasse.

O serviço de integridade relaciona-se à proteção contra modificação de mensagens, negação de serviço, e repúdio. Ataque de modificação de mensagens envolve exclusão, inserção, alteração de conteúdos da mensagem (DULANEY, 2009). Mecanismos que utilizem funções *hash* e algoritmos MAC¹ são suficientes para que o receptor possa verificar a mensagem.

Negação de serviço impede que a infra-estrutura de comunicação (ou qualquer serviço, hardware, software, subconjunto de enlaces, *switches*) tenha a sua capacidade gerencial normal das suas funcionalidades prejudicada. Pode-se ter ataque destinado a um *host*, a um serviço de auditoria de segurança, ou até mesmo uma rede inteira pode ser perturbada com tráfego com rajadas de pacotes (MIRKOVIC et al., 2005) (STALLINGS, 2014). Pode-se associar também a integridade como sendo uma das contramedidas que podem

ser tomadas para atenuar o ataque de negação de serviço (STALLINGS, 2014). Mecanismos de *hash/nonces*/chaves simétricas compartilhadas podem auxiliar as entidades comunicantes a detectarem falsas e/ou repetidas primitivas. Nesse contexto, o hospedeiro pode recusar rajadas de primitivas que não são legítimas em um determinado contexto de comunicação. Esse ataque é um dos mais difíceis de serem combatidos, porém, pode-se considerar essa recusa de processamento de primitivas ilegítimas como sendo uma das contramedidas de segurança que conseguem atenuá-lo.

Repúdio é um processo no qual as entidades envolvidas na comunicação não podem provar que uma transação ocorreu entre elas (DULANEY, 2009). Mecanismos que envolvem funções *Hash/MAC*¹/assinatura digital podem ser utilizados para conter esse tipo de ataque. De toda forma, o auxílio de um terceiro confiável se torna necessário.

2.1.3 Disponibilidade

Disponibilidade (REC, 1991) (SHIREY, 2007) é a capacidade de um sistema ou de um recurso de sistema ser acessível e utilizável sob demanda por uma entidade autorizada.

O ataque de negação de serviço é uma ameaça iminente contra a disponibilidade. Na subseção 2.1.2, já mencionamos alguns mecanismos que podem ser utilizados contra esse ataque. As contramedidas para restringir essa ameaça são bastante diversas, algumas envolvem teoria de sistemas distribuídos (tolerância a falhas), *self-management* (tolerância a falhas), outras alguns tipos de segurança operacional, como colocar equipamentos de *firewall*, sistemas de detecção de invasão (IDSs) e sistema de prevenção de invasão (IPSs) (KUROSE; ROSS, 2013), ou até mesmo evitar através de um controle de fluxo ou mecanismo de alocação de banda (KHONDOKER et al., 2014).

Como podemos ver, é um problema amplo, que trafega em contramedidas que envolvem várias áreas da ciência da computação. (REC, 1991) trata a disponibilidade como uma propriedade a ser tratada em vários serviços de segurança. Ela depende do gerenciamento e controle apropriado dos recursos do sistema, dessa forma, também depende do controle de acesso e outros serviços de segurança (STALLINGS, 2014).

Nesse contexto, onde há diversas contramedidas existentes em vários setores distintos da ciência da computação, esse trabalho preocupa-se em analisar apenas os serviços de segurança da especificação do protocolo ESMP e os recursos da arquitetura ETArch (arquitetura escolhida para aplicação das funcionalidades do ESMP) que possam atenuar esse ataque. A análise não tem a pretensão de solucionar as várias maneiras de ataques do DoS, mas apenas mitigar alguns casos específicos.

2.1.4 Autenticação

A autenticação é um serviço de segurança que garante a autenticidade da comunicação. Há algumas formas distintas de se garantir autenticidade. No caso de haver uma comu-

nicação em que só uma mensagem é enviada, o serviço de autenticação tem que garantir para a entidade de destino que a mensagem é autêntica, e isso se resume na garantia de que a mensagem tenha sido enviada pelo remetente registrado no cabeçalho da primitiva (STALLINGS, 2014). Um exemplo dessa situação é o *Open Shortest Path First* (OSPF), que precisa verificar a autenticidade das primitivas de controle que divulgam os estados de enlace (KUROSE; ROSS, 2013).

Outra situação seria a interação existente entre duas entidades comunicantes. Primeiramente, o serviço tem que garantir a autenticidade mútua, ou seja, cada uma das entidades têm que autenticar a outra para que a comunicação estabeleça um laço de confiança entre as partes. Nesse contexto, cada uma é realmente quem diz ser. Em um segundo momento, o serviço tem que garantir a proteção contra alguns ataques, como *man-in-the-middle* (CIAMPA, 2009), ataque por reflexão (*reflection attack*) (DONG; CHEN, 2012), disfarce (*masquerading*) (HAMID; GIANLUIGI; LILBURN, 2010), e ataque de repetição (*replaying attack*) (STALLINGS, 2014).

No ataque *man-in-the-middle*, o invasor intercepta a mensagem e pode apenas observar as informações ou modificá-la para prejudicar a comunicação das entidades envolvidas. O ataque por reflexão acontece quando um invasor mantém uma comunicação com uma das entidades que participam da comunicação como se ele (invasor) fosse uma entidade autorizada. O disfarce acontece quando o invasor finge ser uma das entidades que participam da comunicação. O ataque de repetição ocorre quando o invasor intercepta uma mensagem válida da comunicação entre entidades autorizadas, e mais tarde, usa essas mensagens para conseguir comunicação em outras sessões posteriores.

2.1.5 Política de Segurança

Política de segurança é um conjunto de ações/medidas necessárias que visam conter as ameaças e vulnerabilidades de um sistema de computação ou de uma rede de computadores. A implementação e manutenção dessas políticas tem a função de governar vários serviços/mecanismos de segurança que são cruciais para proteger, por exemplo, uma comunicação (HARDJONO; TSUDIK, 2000).

Podemos citar como exemplo de serviços/mecanismos de segurança (STALLINGS, 2014): configuração do algoritmo de criptografia (*Advanced Encryption Standard* - AES / *Data Encryption Standard* - DES); configuração do algoritmo assimétrico (RSA/CURVA ELIPTICA); configuração da função de *hash* (*Message Digest 5* - MD5 / *Secure Hash Algorithm* - SHA); configuração da política de disseminação de chaves compartilhadas (simétrica/assimétrica/híbrida); configuração da troca de chaves públicas (autoridade de chave pública/certificado); configuração da autoridade certificadora (*Certification Authority* - CA) aceita na comunicação (Caixa Econômica Federal - CEF / Banco do Brasil, etc.); configuração do tipo de certificado (X.509.ITU-T (ITU-T, 2001), etc.); e outros.

No caso do *IP Security Protocol* (IPSec) (STALLINGS, 2014), as políticas de segurança governam as Associações de Segurança (AS). O protocolo é flexível às necessidades do usuário graças a esse recurso, porém, é válido lembrar que a comunicação de segurança realizado é *peer-to-peer*.

No caso de comunicação *multicast*, há alguns desafios associados à política de segurança, tais como (HARDJONO; TSUDIK, 2000): gerenciamento das entidades quanto à participação nos grupos (autenticação/inclusão/exclusão/controle de acesso dessas entidades nos grupos); políticas para lidar com erros decorrentes específicos dos mecanismos de segurança, como por exemplo, erros em servidores de chaves; distribuição de chaves à fim de conseguir confidencialidade das informações; e outros.

A seguir, tratamos das limitações da arquitetura legada e focamos a discussão em dois requisitos: *multicasting* e segurança. A escolha desses requisitos tem relação direta com a motivação desse trabalho, que é criar uma especificação de protocolo que visa garantir segurança em comunicações *multicast*.

2.2 Limitações da arquitetura TCP/IP

A arquitetura TCP/IP mostrou-se incapaz de responder aos novos requisitos (QoS, QoE, mobilidade, *multicasting*, segurança, etc) demandados pelo surgimento de novas aplicações e serviços na rede mundial de computadores (PEREIRA, 2012) (SILVA, 2013) (KHONDOKER et al., 2014). Em parte, o efeito limitador dessa arquitetura concentra-se basicamente em duas características: a sobrecarga que existe no endereçamento lógico IP (identificador e localizador); e a estrutura rígida e inflexível das camadas intermediárias (PEREIRA, 2012) (SILVA, 2013) (VENKATARAMANI et al., 2014). Nessa seção, focaremos nas limitações que dizem respeito aos encaminhamentos *multicast* e segurança.

Antes de adentrar no assunto das limitações de *multicasting* da arquitetura legada, alguns conceitos se tornam necessários: (i) *unicasting* ou comunicação *unicast* estabelece uma relação biunívoca entre um endereço e uma entidade do conjunto. Na prática, existe um única origem e um único destino. O roteador recebe uma primitiva e a encaminha por apenas uma de suas interfaces (FOROUZAN; FEGAN, 2009); (ii) *multicasting* ou comunicação *multicast* estabelece uma relação unívoca entre um endereço e um subconjunto de entidades do conjunto. Na prática, existe uma única origem e um grupo de destinos. O roteador recebe uma primitiva e a encaminha por várias de suas interfaces. (FOROUZAN; FEGAN, 2009); (iii) *broadcast* estabelece uma relação unívoca entre um endereço e todas as entidades do conjunto. Na prática, o roteador recebe uma primitiva e a encaminha por todas as suas interfaces (FOROUZAN; FEGAN, 2009); (iv) *unicast* múltiplo é uma maneira de enviar uma primitiva para vários receptores. A ideia central é fazer a replicação dessas primitivas na origem e enviá-las para "N" destinos. O software de email é um bom exemplo desse método. Ao enviar uma mensagem para vários

emails distintos, ele cria "N" réplicas, e consequentemente executa "N" envios *unicast* (FOROUZAN; FEGAN, 2009). A redundância de pacotes criadas na rede não é eficiente, principalmente quando a mensagem é enviada para um grupo muito grande.

Mesmo que o protocolo IP tenha alcançado sucesso e seja amplamente utilizado, ele é um dos grandes responsáveis pelas limitações citadas. Seu crescimento deveu-se principalmente à proliferação de aplicativos *unicasting*, tais como correio eletrônico e transferência confiável de arquivos. No entanto, a diversificação de aplicações de mídias digitais, teleconferências, bancos de dados distribuídos, jogos *multiplayer* e outros, culminou na proposta de uma extensão *multicast* para o IP, denominado IP Multicast (DEERING; CHERITON, 1990).

Em suma, o IP Multicast (DEERING; CHERITON, 1990) transmite apenas uma cópia dos dados e os elementos de rede (NE) apropriados (roteadores, *switches*, e outros) geram cópias de envio aos receptores. Após décadas de pesquisa sobre as diversas questões do IP Multicast, como gerenciamento de grupo, segurança, escalabilidade, QoS, a implantação generalizada desse protocolo na rede tem sido perseguida por questões técnicas, administrativas e de negócios (HOSSEINI et al., 2007). Número limitado de endereços *multicast*, limitações de segurança, inundações de primitivas de controle para monitoramento e gerenciamento de grupos de um roteador *multicast* representam algumas das críticas recebidas. O controle de manutenção dos grupos *multicast*, em especial, é um entrave na implementação desse protocolo em ambientes de *datacenters*, *Points of Presence* (POPs), *Internet Exchange Points* (IXPs), entre outros. A falta de oferta de serviços de comunicação *multicast* em provedores de serviços de Internet locais (ISPs) corroboram esse fato (HOSSEINI et al., 2007).

Apesar do IPV6 atenuar alguns problemas do IPV4, tais como a limitação de endereços (extensão do endereço para 128 bits), QoS (adição de prioridade e rótulos de fluxos no cabeçalho), ajustes ou possíveis manutenções no roteamento devido ao QoS (adição de cabeçalhos de extensão para roteamentos livres ou restritos), fragmentação (fragmentação só na origem), *multicasting* (alocação provisória ou permanente de endereços em interfaces de rede - uma interface pode pertencer a vários grupos) (HINDEN; DEERING, 2006) (FOROUZAN; FEGAN, 2009), e outros; o *multicast* baseado no IPV6 apresenta desafios em relação a segurança (DAVIES; KRISHNAN; SAVOLA, 2007), podendo potencializar ataques; além disso, *multicast* em cenários de mobilidade, apresentam uma série de desafios (ROMDHANI et al., 2004), potencializado pela junção desses dois requisitos.

Com relação à pressão por serviços *multicast*, um *Internet Service Provider* (ISP) pode estar disposto a oferecer um serviço melhor com soluções em infraestrutura por um preço ou para atender Acordo de Nível de Serviço (SLA). Essas novas infraestruturas podem contextualizar o surgimento de aplicações que suportam IP Multicast (AGGARWAL, 2014) (CICIC; BRYHNI, 2000). Por outro lado, houve uma grande proliferação de aplicações que tratam a questão da comunicação *multicast* sem nenhuma espécie de suporte

da rede, denominadas *Application Layer Multicast* (ALM) (HOSSEINI et al., 2007). No primeiro caso, podemos citar as *Virtual Private Networks* (VPNs) (ROSEN; REKHTER, 2006) (AGGARWAL; KAMITE; FANG, 2007) baseadas em MPLS com serviços *multicast* (AGGARWAL, 2014), e refletores Unicast/Multicast (EL-SAYED; ROCA; MATHY, 2003) (CICIC; BRYHNI, 2000); no segundo caso podemos citar alguns serviços de colaboração de mídia baseados em nuvem (rede sobreposta), como o Google Hangouts, Skype e Whatsapp (XAVIER et al., 2016).

No caso de algumas iniciativas, como por exemplo a utilização de refletores *unicast-multicast*, há necessariamente a inclusão de um sistema central de multicast (MBONE - *Multicast BackBone*). MBONE é uma rede de sobreposição virtual executada sobre os recursos *unicasts* da rede IP (FOROUZAN; FEGAN, 2009), ou seja, pode ser entendida como um grupo de roteadores *multicast* que comutam seus pacotes utilizando-se para isso roteadores *unicast*. Para isso, são criados encapsulamentos ou tunelamentos lógicos que interligam diversas ilhas *multicast*. O Refletor unicast-multicast (CICIC; BRYHNI, 2000) é um aplicativo que permite às entidades compartilhar grupos e sessões MBone. Essa solução poderia ser utilizada para distribuição de vídeos, onde um conteúdo é distribuído para vários clientes. A qualidade do refletor é sua capacidade de controle e facilidade de uso, porém, existe um desafio com relação a escalabilidade (CICIC; BRYHNI, 2000). Além disso, o MBone, apesar de diminuir a quantidade de dados requeridos para uma transmissão multiponto, utiliza-se de mecanismos da rede subjacente (*unicast*) e tem limitações quanto ao controle de acesso às árvores *multicast* (SILVA, 2013).

Existem alguns pontos fundamentais para a relutância de fornecimento/adoção de serviços IP Multicast (DEERING; CHERITON, 1990) pelas ISPs (EL-SAYED; ROCA; MATHY, 2003) (HOSSEINI et al., 2007) (FOROUZAN; FEGAN, 2009): (i) o *multicasting* quebra o modelo de precificação usual, aquele no qual apenas o fluxo de entrada é cobrado; (ii) a questão de segurança dos protocolos do IP Multicast. Um exemplo é o *Internet Group Management Protocol* (IGMP), que em diversas soluções *multicast* entrega a mensagem para *hosts* que não pertencem ao grupo. Outros pontos são a facilidade de ataques de inundação (*flooding attacks*) via *multicasting*, recepção não autorizada de dados de uma sessão *multicasting*, dificuldade de configuração de *firewalls*, entre outros; (iii) questões técnicas de mapeamento entre endereços IP e MAC²; (iv) complexidade de gestão e monitoramento pelos elementos de rede (*switches*, roteadores, etc); (v) aumento do processamento no núcleo da rede; (vi) granularidade mínima de endereçamento oferecida pela camada de rede. Independentemente do protocolo, não permite que se especifique um endereço ou elabore um mecanismo de segurança para usuários, conteúdos, etc.; (vii) os roteadores com capacidade IP Multicast tem que ser instalados em todos os níveis de rede (roteadores de *backbone*, núcleo e borda) para que o serviço funcione e esteja amplamente disponível. Certamente, apresenta um custo substancial para a empresa.

Em contrapartida, a abordagem do IP Multicast faz com que o emissor envie apenas

uma primitiva para a rede, e os elementos de rede (NE) transmitem essa primitiva apenas uma vez por enlace (link). O resultado é que cada NE/receptor recebe apenas uma primitiva, pois a árvore gerada pelo método IP Multicast não admite redundâncias. No caso da ALM, os nós do grafo da aplicação são os *hosts*, e portanto, necessariamente, haverá duplicações. Nesse quesito, o IP Multicast possui um método mais eficiente que qualquer ALM (HOSSEINI et al., 2007).

No entanto, o que ocorreu, de um modo geral, é que o tráfego passou a ser tratado pela ALM, e os sistemas finais começaram a ser os protagonistas do controle da comunicação *multicast* na rede mundial de computadores. Afinal de contas, as desvantagens da ALM, como atrasos de comutação mais longos e uso de rede de maneira menos eficiente quando comparado à eficiência das árvores de comutação do IP Multicast são equilibrados com suas vantagens: implantação imediata em nível mundial; manutenção e/ou adição de novos serviços de segurança nos algoritmos de comutação de pacotes *multicast*; eliminação da sobrecarga de controle nos elementos de rede do núcleo/borda com relação à manutenção dos grupos *multicast*. No caso do ALM, esse controle é feito nos próprios *hosts* (sistemas finais) (HOSSEINI et al., 2007). Um ponto negativo para as ALMs é que a topologia física da rede subjacente é completamente oculta. A capacidade da ALM em gerar árvores de comutação que tenha uma aproximação topológica com a camada subjacente melhora o seu desempenho de entrega de primitivas e pode ser utilizado como uma métrica para avaliar a performance dessas aplicações.

Segundo (HARDJONO; TSUDIK, 2000), não é o objetivo do IP Multicast fornecer todos os requisitos de segurança que envolvem a complexidade de uma comunicação *multicast*. Em vez disso, esse modelo permite que serviços e mecanismos adicionais sejam construídos sobre ele. Essa dissociação acaba sendo vantajosa em alguns aspectos: permite que modelos e arquiteturas de segurança sejam implantados sem afetar a árvore de distribuição *multicast*, e do ponto de vista do aplicativo, cada qual implementa as suas funcionalidades de segurança de acordo com suas necessidades. Esse fato destoa um pouco das novas arquiteturas de Internet do futuro, que se preocupam em oferecer camadas intermediárias de rede flexíveis a tal ponto que as aplicações atuais e futuras pudessem utilizá-las de acordo com suas necessidades. Enquanto o IP Multicast deixa o trabalho para as aplicações por existir várias limitações de arquitetura, as arquiteturas de Internet do futuro tentam oferecer as demandas das aplicações de forma transparente nas camadas intermediárias.

Outro problema de segurança é a granularidade proporcionada pelas camadas da arquitetura legada. Serviços de segurança são oferecidos em várias camadas e a razão disso é simples: mesmo que a camada de rede cifre os dados de datagramas, e autentique endereços IP no nível de rede, ela não consegue prover segurança no nível de usuário, conteúdo, serviços, etc. Outro aspecto importante é a complexidade estrutural que a arquitetura apresenta para inclusão ou modificação de serviços em suas camadas intermediárias. Um

exemplo é o *Secure Sockets Layer* (SSL), que é um protocolo de aplicação que fornece serviços de transporte. No caso do *IP Security Protocol* (IPSec), é um protocolo que proporciona os mesmos problemas de complexidade para a arquitetura. É um protocolo que oferece serviços de rede e é encapsulado por um protocolo de rede (IP). Dependendo da situação, pode ser encapsulado duas vezes (tunelamento) (STALLINGS, 2014). Essas sobreposições de funcionalidades provocam um *overhead* de comunicação quanto à multiplexação/demultiplexação de primitivas, aumenta a complexidade da arquitetura e causa uma disfuncionalidade nos princípios filosóficos do modelo de camadas. Além disso, são protocolos rígidos, ou seja, não se consegue implantar ou modificar serviços desses protocolos trivialmente.

A seguir, são apresentados os principais protocolos de segurança da Internet. O objetivo é descrever seus principais serviços com o intuito de realizar uma análise/avaliação de segurança quanto à capacidade de atenuar ameaças e/ou ataques iminentes na rede. No Cap. 5 é feita uma avaliação comparativa entre esses protocolos e a especificação do ESMP proposta nesse trabalho.

2.3 Principais protocolos de segurança da arquitetura TCP/IP

Nessa seção, verificamos como os principais protocolos de segurança da arquitetura TCP/IP fornecem a segurança de rede, ou seja, analisamos as características desses protocolos que protegem o tráfego das primitivas de dados/controle entre duas entidades comunicantes quaisquer. Para isso, analisamos brevemente o funcionamento e os serviços oferecidos por esses protocolos, tais como autenticidade, confidencialidade, integridade e disponibilidade.

2.3.1 SSL/TLS

SSL/TLS são protocolos que contém a mesma padronização de segurança na Internet. O TLS é um protocolo derivado do SSL e está presente nos principais navegadores da web. Suas diferenças não pertencem ao escopo desse trabalho, e essa seção tem a finalidade de discorrer brevemente sobre as suas principais características. Apesar da explanação abaixo sempre se referir ao protocolo SSL, é importante salientar que todas as características de funcionamento apresentadas também servem para o TLS.

O SSL é um dos protocolos mais utilizados da rede de computadores, até porque auxilia na segurança de uma das aplicações mais utilizadas da Internet, o *Hypertext Transfer Protocol* (HTTP). O SSL é considerado um protocolo de aplicação que oferece serviços de transporte. É composto por basicamente quatro protocolos: protocolo de handshake (*Handshake Protocol*), protocolo de especificação de mudança de cifra (*Change Cipher*

Spec protocol), protocolo de alerta (*Alert Protocol*) e o protocolo de registro SSL (*SSL Record Protocol*) (STALLINGS, 2014). Os três primeiros protocolos sobrepõem o último, ou seja, todas as *Protocol Data Units* (PDUs) dos primeiros três protocolos sempre são processados ou encapsulados pelo último.

O procedimento do protocolo de registro SSL passa basicamente por cinco passos (FOROUZAN; FEGAN, 2009) (STALLINGS, 2014): fragmentação da informação em blocos; Compactação de cada um desses blocos (opcional); adição de um Message Authentication Code (MAC¹); encriptação do conteúdo gerado até o momento; anexação de um cabeçalho SSL. Esse procedimento oferece basicamente dois serviços: confidencialidade e integridade de mensagens.

O protocolo de especificação de cifra consiste em apenas 1 byte e tem a função de ativar os serviços de segurança negociados na fase de handshake. Após trocarem essa mensagem, as duas entidades comunicantes podem começar a utilizar os serviços (FOROUZAN; FEGAN, 2009) (STALLINGS, 2014).

O protocolo de alerta tem 2 bytes e transmitem mensagens de alerta, ou de erro, ou de possíveis tentativas de ataque. O primeiro byte refere-se à gravidade do problema (Advertência ou Fatal) e o segundo byte refere-se à descrição do problema. Algumas mensagens graves, como por exemplo, "*bad_record_mac*" (um MAC¹ incorreto foi recebido) pode indicar ataque e por conta disso, significará, o término imediato da conexão. Cada mensagem é passível de um tipo de ação por parte do protocolo (FOROUZAN; FEGAN, 2009) (STALLINGS, 2014).

Dito isso, esse protocolo fornece alguns serviços interessantes para duas entidades que se comunicam na WEB, quais sejam: (i) autenticação das entidades participantes; (ii) estabelecimento de chaves compartilhadas entre as entidades; (iii) integridade da mensagem através da utilização de MAC¹s; (iv) prevenção de ataques de replicação (utilização de *nonces*); (v) confidencialidade das primitivas trocadas; e outros.

Outro ponto importante é que as comunicações de segurança que utilizam os serviços oferecidos pelo SSL possuem características *peer-to-peer* (STALLINGS, 2014). Portanto, esse protocolo não provisiona segurança para comunicações multicast. Outra questão a ser mencionada é que esse protocolo, apesar de pertencer a camada de aplicação, é rígido e não oferece capacidade de atender requisitos de aplicações distribuídas que necessitam de conexões *one-to-many/many-to-many* por dois motivos: o primeiro é que não tem a natureza de conexões *one-to-many/many-to-many* (STALLINGS, 2014); o segundo é que não permite inclusão ou alteração dos serviços necessários sem alterar a complexidade da estrutura da Internet (PEREIRA, 2012).

Quanto à mitigação de possíveis ataques à arquitetura TCP/IP, o SSL oferece contramedidas que são detalhadas a seguir.

SSL é robusto contra o ataque de espionagem (*snooping attack*) porque ele possui várias configurações distintas na fase de handshake que permite diferentes mecanismos de

Tabela 1 – Mecanismos de mitigação de ataques do protocolo SSL/TLS

Metas de Segurança	Ataques de Segurança	Mitigação
Confidencialidade	Espionagem (<i>Snooping</i>)	v
	Análise de tráfego (<i>Traffic analysis</i>)	x
Integridade	Modificação (<i>Modification</i>)	v
	Repúdio (<i>Repudiation</i>)	x
Disponibilidade	Negação de serviço (<i>Denial of service</i>) DoS	x
Autenticação	<i>Man-in-the-middle</i>	v
	Ataque por reflexão (<i>reflection attack</i>)	v
	Disfarce (<i>Masquerading</i>)	v
	Repetição (<i>Replaying</i>)	v
Segurança multicast	Apenas os ataques mitigados pelo protocolo	x

Adaptada de (KHONDOKER et al., 2014).

criptação de mensagem (STALLINGS, 2014).

Com relação a ataques de modificação de mensagens (*modification attack*); o SSL possui funções de MAC¹, que são aplicadas aos dados da aplicação, e portanto, o receptor pode checar a integridade das informações da primitiva. (STALLINGS, 2014)

Para ataques man-in-the-middle e reflexão (*reflection attack*); o SSL possui vulnerabilidades que podem ser exploradas. Como na primeira fase do *handshake*, as primitivas ainda não estão protegidas, um atacante pode aproveitar desse ameaça para executar qualquer um desses dois ataques. Essa interceptação não é fácil de ser realizada, e o atacante, necessariamente, tem que possuir um certificado válido de uma das unidades certificadoras aceitas pela comunicação. Apesar dessa ameaça sutil na fase de *handshake*, pode-se dizer que depois que é estabelecida a conexão, esses dois ataques são atenuados devido à capacidade de autenticação mútua entre as entidades envolvidas. O disfarce (*masquerading attack*) não é um problema para o SSL, pois possui mecanismos de autenticação mútua. (STALLINGS, 2014)

Quanto à ataques de negação de serviço (DoS); o SSL não possui mecanismos para atenuá-lo. Uma das formas de mitigar esse ataque é modificar o caminho de roteamento provocada pelo congestionamento do ataque e/ou controlar alocação de banda (KHONDOKER et al., 2014). O SSL não possui nenhum desses recursos.

O ataque de repúdio poderia ser evitado, caso não houvesse à ameaça dos ataques *man-in-the-middle* e reflexão (*reflection attack*). No caso do *man-in-the-middle*, uma terceira entidade pode conhecer a chave simétrica compartilhada entre os dois participantes na comunicação *peer-to-peer*. A forma de evitar esse ataque é que os participantes da comunicação conheçam a chave compartilhada antes mesmo do primeiro contato.

Quanto aos ataques de análise de tráfego (*traffic analysis attack*), a forma de atenuá-lo é ter um mecanismo para ocultar a identidade dos usuários (KHONDOKER et al., 2014) (STALLINGS, 2014). O SSL não possui esse mecanismo, pois está exposto no cabeçalho do pacote IP a origem e o destino da comunicação.

A repetição (*replaying attack*) é atenuada, pois no ato da verificação de integridade, analisa-se a sequência dos *nonces* das duas entidades envolvidas na comunicação na operação dos cálculos de funções de *hash* (STALLINGS, 2014).

Quanto à segurança *multicast*, o SSL não oferece suporte nem tão pouco segurança para essa natureza de comunicação.

A Tab. 1 possui um resumo dos mecanismos de mitigação de ataques por metas de segurança do protocolo SSL/TLS. Essa tabela se junta às informações de protocolos de outras arquiteturas para a realização de uma comparação final (no Cap. 5) entre o que está descrito nessa seção (SSL/TLS) e o que está sendo proposto no Cap. 4 (protocolo de comunicação *multicast* para arquiteturas de Internet do Futuro).

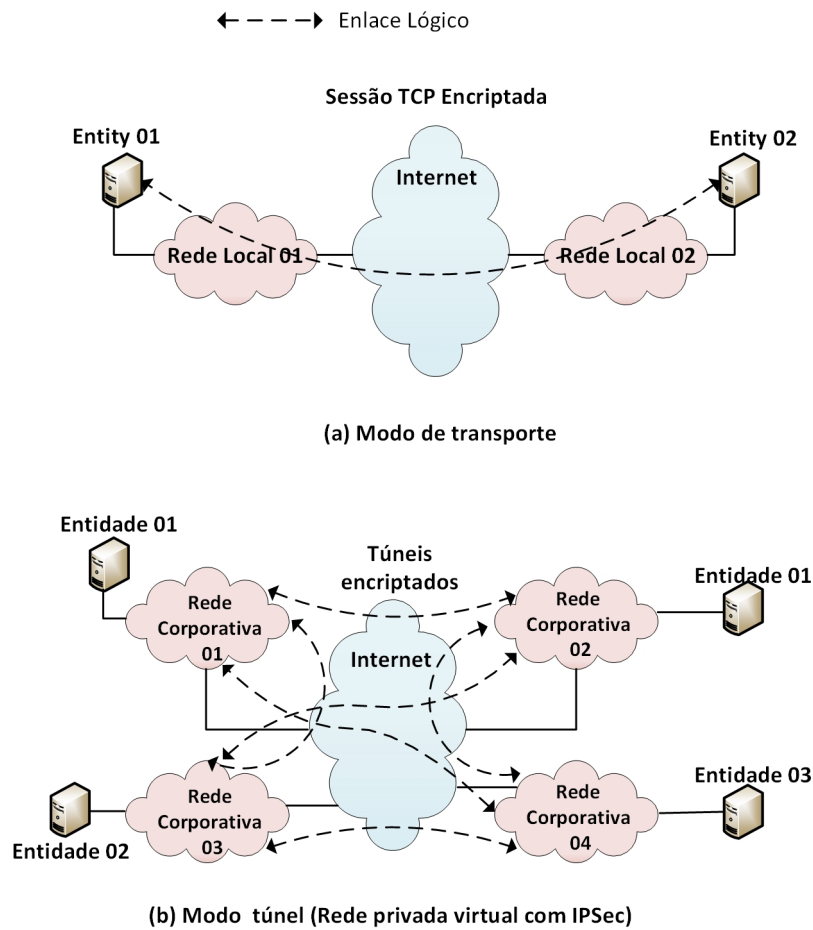
2.3.2 IPSec

O IPSec (FRANKEL; KRISHNAN, 2011) é uma coleção de protocolos definidos pela *Internet Engineering Task Force* (IETF) para fornecer segurança em nível de rede.

Esse protocolo oferece alguns benefícios (STALLINGS, 2014): o IPSec está abaixo da camada de transporte, e por conta disso, é transparente para as aplicações; o IPSec pode ser implantado em roteadores ou *firewalls*, e nesse caso em particular, a rede local onde a entidade está localizada não precisa gerar *overhead* de processamento com relação a mecanismos de segurança; o IPSec fornece uma estrutura (políticas de segurança (SP - *Security Police*) / associação de segurança (SA - *Security Association*)) para que as entidades escolham os mecanismos de encriptação, *hashing*, autenticação, e outros; o IPSec, através das SPs/SAs, consegue fazer uma discriminação no tratamento dos tráfegos de chegada/saída da rede local.

Dito isso, alguns procedimentos são importantes para o funcionamento desse protocolo (STALLINGS, 2014): (i) gerenciamento de chaves; (ii) estabelecimento de uma conexão lógica SA baseada em políticas de segurança aplicada a cada pacote IP; (iii) definição do

Figura 1 – Modos de utilização do IPSec



Adaptada de (STALLINGS, 2014).

modo de uso utilizado (Transporte/Túnel); (iv) encapsulamento do *payload* das primitivas conforme protocolo configurado (AH - *Authentication Header* / ESP - *Encapsulating Security Payload*) na política de segurança.

O protocolo *Internet Key Exchange* (IKE) (STALLINGS, 2014) é responsável pelas primeiras operações do protocolo IPSec. Essas operações são conhecidas como trocas iniciais, e são responsáveis pela negociação de algoritmos criptográficos, *nonces*, valores Diffie-Hellman para cálculo do par de chaves pública/privada, etc. Nesse momento é criada uma conexão lógica SA especial denominada IKE SA. Ela define os parâmetros para um canal seguro, e a partir desse momento, as entidades envolvidas se autenticam e montam a primeira SA IPSec.

O SA IPSec (FOROUZAN; FEGAN, 2009) é uma conexão lógica orientada através de uma política de segurança negociada na fase de gerenciamento de chaves. A política de segurança (STALLINGS, 2014) é armazenada em um banco de dados que possui entradas/seletores que apontam para um ou mais SAs. Os SAs possuem informações importantes para o funcionamento geral do protocolo IPSec tais como: identificador

Tabela 2 – Mecanismos de mitigação de ataques do protocolo IPSec

Metas de Segurança	Ataques de Segurança	Mitigação
Confidencialidade	Espionagem (<i>Snooping</i>)	v
	Análise de tráfego (<i>Traffic analysis</i>)	v
Integridade	Modificação (<i>Modification</i>)	v
	Repúdio (<i>Repudiation</i>)	v
Disponibilidade	Negação de serviço (<i>Denial of service</i>) DoS	x
Autenticação	<i>Man-in-the-middle</i>	v
	Ataque por reflexão (<i>reflection attack</i>)	v
	Disfarce (<i>Masquerading</i>)	v
	Repetição (<i>Replaying</i>)	v
Segurança <i>multicast</i>	Apenas os ataques mitigados pelo protocolo	x

Adaptada de (KHONDOKER et al., 2014).

do SA; protocolo utilizado para a autenticação e/ou confidencialidade das informações (AH/ESP); modo do protocolo IPSec(Túnel/Transporte); algoritmos de encriptação do ESP; algoritmos de autenticação do AH; valores iniciais; *nonces*; etc.

Existem dois modos de utilização do protocolos ESP/AH (STALLINGS, 2014). O modo transporte é utilizado na comunicação direta entre duas entidades. A proteção do modo transporte se estende ao *payload* de um pacote IP. Nesse contexto, o *payload* representa as primitivas e as informações (cabeçalhos/dados) da camada de transporte, ou seja, a autenticação e/ou a encriptação realizada pelos protocolos ESP/AH são direcionadas às informações (cabeçalhos/dados) que vêm após o cabeçalho IP. O Modo túnel oferece proteção ao pacote IP. Nesse contexto, o *payload* do IPSec é todas as informações (cabeçalhos/dados) do pacote IP. Para isso, é criado um outro pacote IP externo (diferente do IP interno) que direcionará as informações até o destino. Esses novos IPs geralmente referem-se aos *gateways* (roteadores/*firewalls*) de segurança das redes locais envolvidas na comunicação. A ocultação das informações (cabeçalhos/dados) dos pacotes IPs no modo

de tunelamento dificulta a análise de tráfego por um possível atacante (STALLINGS, 2014). A Fig. 1 exibe o funcionamento desses dois modos de utilização.

É importante mencionar que o encapsulamento se modifica de acordo com o modo de utilização executado. Pela Fig.1(a), percebe-se que a comunicação acontece diretamente entre as duas entidades. Na Fig.1(b), a comunicação acontece entre os *gateways* da rede corporativa. É importante mencionar que a Rede Corporativa 01 faz três cópias da mesma primitiva para que o objetivo do envio para as outras redes exibidas na figura seja alcançado. Como vimos na seção 2.2, o *unicast* múltiplo pode sobrecarregar a rede na sua origem e não é um mecanismo de envio eficiente. Esse mecanismo é utilizado nessa situação porque o IPSec é um protocolo *peer-to-peer*, e carrega em si o peso das limitações da arquitetura legada.

Quanto aos serviços oferecidos, podemos citar confidencialidade, integridade, autenticação e rejeição de pacotes replicados (representa uma solução parcial para resolução do DoS). No mais, esse protocolo permite uma grande flexibilidade com as definições das políticas de segurança, e com as associações das SAs; porém, se trata de um protocolo rígido e não está apto a grandes modificações de forma fácil e transparente. Isso se deve a estrutura rígida da arquitetura ao qual esse protocolo pertence.

Quanto à mitigação de possíveis ataques à arquitetura TCP/IP, o IPSec oferece contramedidas que são detalhadas a seguir.

IPSec é robusto contra o ataque de espionagem (*snooping attack*) porque possui distintos mecanismos de encriptação configurados através dos SAs. O protocolo responsável pela encriptação de mensagem é o ESP. (STALLINGS, 2014).

Com relação a ataques de modificação de mensagens (*modification attack*); o IPSec possui diversas maneiras de verificar a autenticação da mensagem entre as entidades participantes da comunicação. O processo de autenticação verifica, consequentemente, a integridade de mensagem; portanto, o receptor pode checar se houve modificação das informações recebidas. (STALLINGS, 2014).

Para ataques *man-in-the-middle* e reflexão (*reflection attack*); o IPSec exige suporte para gerenciamento de chaves manual ou automatizado (STALLINGS, 2014). O primeiro garante que os sistemas finais possuam chaves compartilhadas antes mesmo do primeiro contato. Dessa forma, esses dois ataques podem ser atenuados pelo protocolo. Quanto ao disfarce (*masquerading attack*); não é um problema para o IPSec, já que possui mecanismos de autenticação mútua (STALLINGS, 2014).

Quanto à ataques de negação de serviço (DoS); o IPSec não possui mecanismos para atenuá-lo. Uma das formas de mitigar esse ataque é modificar o caminho de roteamento provocada pelo congestionamento do ataque e/ou controlar alocação de banda (KHON-DOKER et al., 2014). O IPSec não possui nenhum desses recursos.

O ataque de repúdio pode ser evitado com utilização de comunicação através de par de chaves pública/privada. O IKE é uma melhoria do algoritmo de troca de chave Diffie-

Hellman (STALLINGS, 2014) e é utilizado para atenuar essa espécie de ataque.

Quanto aos ataques de análise de tráfego (*traffic analysis attack*), uma forma de atenuá-lo é ter um mecanismo para ocultar a identidade dos usuários (KHONDOKER et al., 2014) (STALLINGS, 2014). O IPSec possui esse mecanismo no modo de tunelamento, pois a encriptação se estende por todo o pacote IP (STALLINGS, 2014); consequentemente, a origem e o destino dos *hosts* que participam da comunicação são escondidos de um possível atacante.

A repetição (*replaying attack*) é atenuada, pois no ato da verificação de integridade, analisa-se a sequência dos *nonces* das duas entidades envolvidas na comunicação (STALLINGS, 2014).

Quanto à segurança *multicast*, o IPSec não oferece suporte nem tão pouco segurança para essa natureza de comunicação. No caso da utilização do modo túnel, a Fig. 1 mostra que não existe nesse protocolo a comunicação *multicast* intrínseca, e sim, uma cópia do pacote para cada receptor (*unicast* múltiplo).

A Tab. 2 possui um resumo dos mecanismos de mitigação de ataques por metas de segurança do protocolo IPSec. Essa tabela se junta às informações de protocolos de outras arquiteturas para a realização de uma comparação final (no Cap. 5) entre o que está descrito nessa seção (trabalhos relacionados) e o que está sendo proposto no Cap. 4 (protocolo de comunicação *multicast* para arquiteturas de Internet do Futuro).

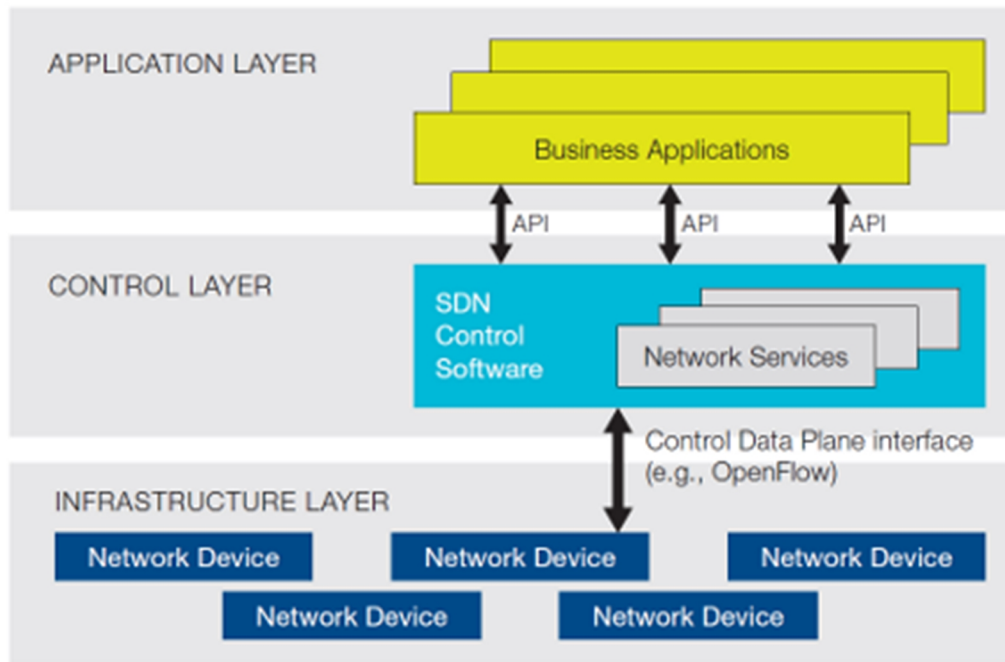
A seguir é apresentado o conceito de Redes Definidas por Software. A importância dessa apresentação deve-se ao fato de que a ETArch (arquitetura onde será aplicada as funcionalidades da especificação do protocolo ESMP) abstrai os conceitos principais do SDN. Nessa mesma sessão é apresentado o protocolo OpenFlow (protocolo muito utilizado em arquiteturas SDN). A ETArch utiliza a versão 1.0 desse protocolo.

2.4 Redes Definidas por Software

Open Networking Foundation (ONF) (FOUNDATION, 2011) é uma organização sem fins lucrativos que visa promover *Software Defined Networking* (SDN) (COX et al., 2017) (KREUTZ et al., 2015) e criar padronizações de protocolos e tecnologias relacionadas. SDN representa uma oportunidade de repensar a rede quanto à forma de gerenciamento e operação. Um dos principais objetivos desse paradigma é criar uma rede ágil e flexível que suporte mudanças rápidas com base nas necessidades e demandas de empresas e aplicações.

Uma das características mais importantes do conceito de SDN é a noção de separação entre o plano de controle e o plano de dados. A ideia é reunir todo o plano de controle segregado e colocá-lo em uma instância de software externa logicamente centralizada. Esta nova noção introduz uma infraestrutura de rede que consiste em três camadas: camada de infra-estrutura, camada do controlador e camada de aplicação.

Figura 2 – Arquitetura de rede definida por software (SDN)



Fonte: (FOUNDATION, 2012).

Alguns controladores SDN, como o *Open Network Operation System* (ONOS) (FOUNDATION, 2014), possuem abstrações bem definidas para realizar a comunicação entre as camadas descritas acima, denominadas *API Southbound* e *API Northbound*. A primeira compreende os serviços dos protocolos que determinam como o controlador gerencia o *switch*, e a segunda, são os serviços que o controlador oferece às aplicações, para que essas possam participar da administração da rede. Quanto maior a abstração oferecida, mais simples se torna o desenvolvimento dessas aplicações. OpenFlow (MCKEOWN et al., 2008), é um dos protocolos que materializam a *API Southbound*.

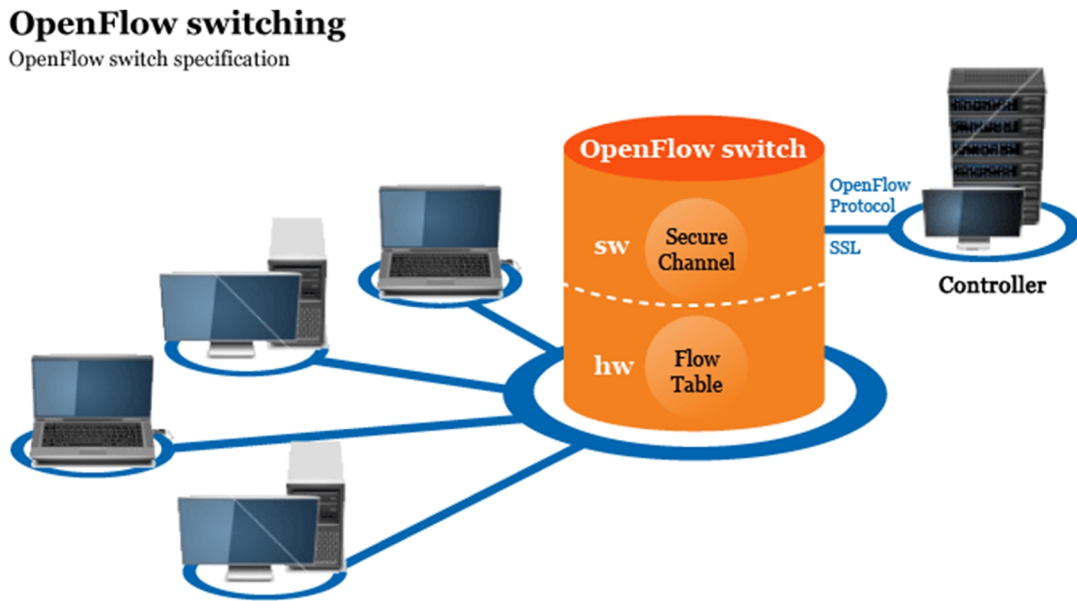
A arquitetura SDN aplica um controle *top-down* de rede logicamente centralizado. Simplificadamente, os controladores gerenciam o controle de rede, de forma centralizada, através de suas aplicações. Esse modelo impulsiona alguns benefícios, tais como a síntese e flexibilização do gerenciamento de rede, como também a eficiência de reação à eventos dinâmicos e/ou inesperados.

2.4.1 OpenFlow

A tecnologia OpenFlow (KERNER, 2012) (MCKEOWN et al., 2008) pode ser considerada uma implementação de SDN, consolidada e utilizada por diversos pesquisadores e fabricantes (FOUNDATION, 2011).

O termo OpenFlow se refere ao *switch* OpenFlow, ao protocolo OpenFlow e ao controlador OpenFlow, os quais possuem papéis diferentes. Um *switch* OpenFlow é uma

Figura 3 – OpenFlow



Fonte: (JIN, 2016).

construção SDN feita para permitir a separação do plano de controle do plano de dados. Nesse tipo de *switch*, o controle das primitivas de dados é feito por uma peça de software externa logicamente centralizada, conhecida como controlador. A comunicação entre o *switch* e o controlador se dá através de um canal de comunicação seguro SSL/TLS e é feita pelo protocolo OpenFlow.

A Fig. 3 representa os elementos da Tecnologia OpenFlow. O *switch* OpenFlow estabelece uma associação segura para comunicação com o controlador, que contém tabelas de fluxos *Flow Tables*. Um *frame* recebido pelo *switch* é condicionado à análise na tabela de fluxos. Caso exista um fluxo para aquele *frame*, ele é encaminhado de acordo com o fluxo; caso contrário, ele é enviado ao controlador, que decidirá seu destino.

A comunicação do *switch* OpenFlow com o controlador é feita por meio do protocolo OpenFlow (KONTESIDOU; ZARIFIS, 2009), que pode utilizar uma associação segura SSL/TLS (LARA; KOLASANI; RAMAMURTHY, 2014). Pode-se dizer que o primeiro *frame* de cada *stream* é enviado ao controlador, que estabelecerá o respectivo fluxo (se houver) a ser utilizado por todos os demais *frames*.

Quando há uma regra de *switching* definida, o controlador configura o *switch*, que insere a regra em sua tabela de fluxos (i.e., cria um fluxo no *switch*), e, deste modo, os demais *frames* dessa *stream* são chaveados de acordo com a regra recém configurada e não mais são encaminhados ao controlador OpenFlow.

Na referência original do OpenFlow, o procedimento determinado pelo controlador e armazenado nos *switches* é chamado de Ações. Um exemplo de ação poderia estar relacionado ao endereço MAC² do *frame*, isto é, todos os *frames* com determinado MAC² são destinados a uma porta específica do *switch*.

Na versão 1.0 do OpenFlow, usada pela ETArch, os possíveis parâmetros a serem usados na criação de ações são: Porta de entrada, Endereço Ethernet de Origem, Endereço Ethernet de Destino, Ethernet *Type*¹, VLAN id, VLAN *priority*, Endereço IP de Origem, Endereço IP de Destino, *Protocol*², ToS (*Type of Service*), Portas³ de Origem e de Destino (KONTESIDOU; ZARIFIS, 2009). A ETArch faz uso das ações OpenFlow, baseadas no endereço Ethernet, para implementar o conceito de *workspace*, que será apresentado no Capítulo 3.

A seguir é apresentada uma proposta de arquitetura de Internet do Futuro. O objetivo é focar a discussão nos requisitos de segurança e multicasting. O intuito é avaliar essas características e averiguar a capacidade da arquitetura em atenuar ataques/ameaças iminentes na rede. No Cap. 5 é feita uma avaliação comparativa entre essa arquitetura e a especificação do ESMP proposta nesse trabalho.

2.5 Proposta para Internet do Futuro

O principal papel de uma arquitetura de Internet do Futuro é atender os requisitos de novas aplicações que surgiram decorrentes da evolução da Internet nas últimas décadas. A principal motivação é que a arquitetura legada não consegue oferecer serviços que garantam mobilidade, comunicação *multicast*, QoS, segurança, dentre outros; sem aumentar a complexidade da sua estrutura.

Duas abordagens são possíveis (REXFORD; DOVROLIS, 2010): evolucionária e *clean slate*. Enquanto nesta, o novo projeto ignora a arquitetura legada e começa a nova estrutura "a partir do zero"; naquela, a nova arquitetura evolui "a partir da atual".

A seguir é apresentada a descrição de um desses projetos (MobilityFirst) ressaltando alguns aspectos centrais de sua arquitetura de rede. A opção por uma arquitetura *clean slate* tem haver com o fato de que a arquitetura de Internet do Futuro escolhida para apresentar a especificação do protocolo proposto nesse trabalho é a ETArch, que possui em comum a visão de uma nova pilha de protocolos, bem como novas abordagens para identificação e localização; ambas características de ruptura com a arquitetura legada.

2.5.1 MobilityFirst

O projeto MobilityFirst (VENKATARAMANI et al., 2014) argumenta que a mobilidade e confiabilidade são objetivos centrais da arquitetura. Confiabilidade, nesse contexto, significa que a rede deve ser resiliente à presença de um pequeno número de terminais maliciosos. Mobilidade é a preocupação de se ter movimentação de dispositivos (*hosts*, servidores, celulares, e outros) sem interrupções de serviços.

¹ Este campo do *frame* Ethernet é frequentemente referenciado como *EtherType*

² Refere-se ao campo *Protocol* do cabeçalho do Protocolo IP

³ Refere-se ao campo *Port* da interface abstrata *Sockets* utilizada pelos protocolos TCP/UDP

Para atingir esses objetivos, a MobilityFirst aprimora a mobilidade ao separar claramente nomes ou identificadores de endereços ou locais de rede e aprimora a segurança, representando ambos de maneira intrinsecamente verificável, contando com um serviço de nomes globalmente distribuído (*Global Name Service* - GNS) que vincula nome à localização (VENKATARAMANI et al., 2014). Tanto os nomes (identificadores) quanto os endereços (localizações) são definidos como identificadores globais únicos (*Globally unique identifiers* - GUIDs). Os GUIDs são derivadas de chaves públicas e são robustos aos sequestros ou falsificações. (VENKATARAMANI et al., 2014).

Essa comunicação baseada em GUID, auxiliado pelo GNS, é o ponto fundamental da arquitetura, e a torna flexível para lidar com uma variedade de *endpoints*, tais como dispositivos, usuários, serviços, conteúdos, descritores textuais; e, primitivas, independentemente de localização, lhe permite suportar comunicação *device-to-device*, *device-to-service*, *centric-content*, *anycast*, *multicast* e outros (VENKATARAMANI et al., 2014).

O GUID é autocertificado, ou seja, qualquer dispositivo de rede pode autenticar outro dispositivo. Um GUID é derivado de um *hash* unidirecional sem a necessidade de uma certificação de terceiros. Essa autenticação entre as entidades ocorre através de um desafio (*nonce*). Um pré-requisito é que as entidades precisam possuir um par de chaves pública/privada para realizar o processo de autenticação entre si. (VENKATARAMANI et al., 2014)

Um endereço de rede (*Network Address* - NA) é um identificador autocertificado. Ele representa o endereço da localização de um dispositivo. No entanto, pode-se ter vários dispositivos alocados na mesma localização. Fazendo uma analogia, o NA representa o endereço um sistema autônomo (*Autonomous System* - AS).

O GNS permite a comunicação *peer-to-peer*, mapeando identificadores (GUIDs ou *Human-Readable-Names* (HRN)) para um NA e um conjunto de atributos. O GNS possui um serviço de atribuição de nomes (*Name Assignment Services* - NAS) para resolver nomes legíveis para o GUID, e um serviço de resolução de nomes (*Global Name Resolution Service* - GNRS) para resolver um GUID para o conjunto de seus atributos (VENKATARAMANI et al., 2014) e para o NA (KHONDOKER et al., 2014).

Para que uma comunicação *peer-to-peer* se concretize, uma entidade obtém o NA correspondente do GUID antes mesmo de enviar a primeira primitiva. A tupla [GUID, NA] é o identificador de destino roteável para esse tipo de comunicação. Esse roteamento é realizado em duas etapas. Primeiramente, o protocolo de roteamento inter-rede é responsável pela entrega da primitiva no NA. Logo após, dentro do NA, o protocolo intra-rede é responsável pela entrega da primitiva para o GUID registrado no seu cabeçalho (VENKATARAMANI et al., 2014).

Na MobilityFirst, outro recurso extremamente importante é a comunicação com reconhecimento de contexto através de descrições de atributos registradas no GNS e vinculadas ao GUID. Essas descrições permitem a configuração de primitivas distintas para cada

contexto. (VENKATARAMANI et al., 2014).

Multicast representa uma instância de entrega sensível ao contexto. O GUID da comunicação *multicast* é o *Multicast Identifier* (MID), que possui o mesmo esquema de formato e autocertificação do GUID. A diferença é que o MID é vinculado a vários GUIDs, que representam as entidades inscritas no grupo *multicast*. Cada membro do grupo se inscreve no MID, informando seu respectivo NA. O serviço de resolução de nomes retorna todos os NAs vinculados a pelo menos um GUID que está no MID. O remetente envia os dados endereçados para $[MID_1, NA_1]$, $[MID_2, NA_2]$, $[MID_3, NA_3]$... $[MID_n, NA_n]$; para cada retorno de NAs. Depois que cada uma dessas mensagens chegam aos seus respectivos NAs, uma cópia é feita para cada membro que está em MID_1 , MID_2 , MID_3 , MID_n (VENKATARAMANI et al., 2014).

Percebe-se que a comunicação *multicast* da MobilityFirst realiza cópias de primitivas na origem por duas vezes. A primeira, quando as mensagens $[MID_1, NA_1]$, $[MID_2, NA_2]$, $[MID_3, NA_3]$... $[MID_n, NA_n]$ são enviadas para cada um dos NAs enumerados. A segunda, quando é feita uma cópia para cada membro de MID_1 , MID_2 , MID_3 ... MID_n na NA correspondente. Dessa forma, a comunicação *multicast* não é intrínseca, pois não existe um endereço que compõe todo o subconjunto de entidades. No entanto, a comunicação é realizada similarmente a uma comunicação de *unicast* múltiplo do TCP/IP.

Um dos principais *endpoints* da MobilityFirst é o conteúdo, que é nomeado usando GUIDs de autocertificação. Diferentemente de um GUID de dispositivo, o GUID de conteúdo é calculado como um *hash* unidirecional do próprio conteúdo, permitindo que qualquer entidade receptora verifique a integridade da mensagem. A segunda diferença é que o GNS não armazena o estado de todos os conteúdos, pois colocaria uma carga muito alta em qualquer provedor GNS; em vez disso o endereço de conteúdo roteável para esse tipo de contexto é o $[PID, CID]$, em que CID é o GUID de conteúdo e o PID é o GUID do editor. O PID poderia estar atrelado a um provedor GNS, uma rede de terminal ou até mesmo um roteador (VENKATARAMANI et al., 2014).

Quanto à segurança, os mecanismos da MobilityFirst podem ser vistos a seguir (KHONDOKER et al., 2014).

1. O identificador de conteúdo é um GUID. GUID é um resultado de *hashing* do conteúdo e pode ser usado como chave pública para o mecanismo de encriptação.
2. Permite atualizações frequentes de roteamento usando a função de GNRS.
3. Usa um protocolo integrado que permite nomes de chaves públicas autocertificáveis.

As contramedidas da arquitetura MobilityFirst à possíveis ataques são detalhadas a seguir.

Tabela 3 – Mecanismos de mitigação de ataques da arquitetura *MobilityFirst*

Metas de Segurança	Ataques de Segurança	Mitigação
Confidencialidade	Espionagem (<i>Snooping</i>)	v
	Análise de tráfego (<i>Traffic analysis</i>)	x
Integridade	Modificação (<i>Modification</i>)	v
	Repúdio (<i>Repudiation</i>)	v
Disponibilidade	Negação de serviço (<i>Denial of service</i>) DoS	v
Autenticação	<i>Man-in-the-middle</i>	v
	Ataque por reflexão (<i>reflection attack</i>)	v
	Disfarce (<i>Masquerading</i>)	v
	Repetição (<i>Replaying</i>)	x
Segurança <i>multicast</i>	Apenas os ataques mitigados pela arquitetura	x

Adaptada de (KHONDOKER et al., 2014).

MobilityFirst é robusto contra o ataque de espionagem (*snooping attack*) porque ele possui um mecanismo de encriptação da mensagem que utiliza o GUID do conteúdo como chave pública (KHONDOKER et al., 2014) (VENKATARAMANI et al., 2014).

Com relação à ataques de modificação de mensagens (*modification attack*); a MobilityFirst atribui um único GUID para cada conteúdo. GUID é um hash de conteúdo, portanto, o receptor pode checar a integridade de conteúdo (KHONDOKER et al., 2014).

Para ataques *man-in-the-middle*, reflexão (*reflection attack*) e disfarce (*masquerading attack*); a MobilityFirst utiliza um protocolo que autentica o usuário através de um desafio bilateral simples. Todos os dispositivos podem autenticar-se sem ajuda de terceiros (KHONDOKER et al., 2014) (VENKATARAMANI et al., 2014).

Quanto à ataques de negação de serviço (DoS); a MobilityFirst consegue atenuá-lo, utilizando para isso a função do GNRS para atualizar a rota das primitivas (KHONDOKER et al., 2014). Um congestionamento de rede provocado por esse ataque pode ser evitado com a modificação das rotas percorridas pelo conteúdo.

O ataque de repúdio pode ser evitado, pois o GUID é autocertificado e pode ser utilizado como chave pública (VENKATARAMANI et al., 2014). Parte-se do princípio que as trocas entre dispositivos se dá através da utilização do par de chaves pública/privada. Dessa forma, esse ataque é atenuado pela MobilityFirst (KHONDOKER et al., 2014).

Quanto aos ataques de análise de tráfego (*traffic analysis attack*) e repetição (*replaying attack*); esses a MobilityFirst não consegue atenuá-los nem tão pouco evitá-los pelos seguintes motivos:

1. não possui um mecanismo que permita comunicação anônima (KHONDOKER et al., 2014), ou seja, no caso geral, a tupla [NA, GUID] é o endereço de destino das mensagens enviadas; portanto, conhece-se a identidade dos usuários (GUID) e o NA (por exemplo, ISP) que vão receber as mensagens (VENKATARAMANI et al., 2014). Por conta disso, a análise de tráfego não pode ser evitada;
2. não possui um mecanismo para vincular as mensagens às sessões (KHONDOKER et al., 2014); dessa forma, não podem fazer um controle de sequenciação dessas mensagens.

A Tab. 3 possui um resumo dos mecanismos de mitigação de ataques por metas de segurança da arquitetura MobilityFirst. Essa tabela se junta às informações de protocolos de outras arquiteturas para a realização de uma comparação final (no Cap. 5) entre o que está descrito nessa seção (trabalhos relacionados) e o que está sendo proposto no Cap. 4 (protocolo de segurança *multicast* para arquiteturas de Internet do Futuro).

Visão geral da arquitetura ETArch e seus conceitos principais

As várias limitações da tecnologia legada, algumas delas especificadas na seção 2.2, levaram pesquisadores a pensar em novos modelos que fomentassem o desenvolvimento de novas arquiteturas. Um desses modelos nasceu em 2011 e foi denominado Modelo de Títulos (PEREIRA, 2012).

Os principais objetivos desse modelo são: criar uma nova estratégia de endereçamento horizontal, que resolva o problema de ambiguidade de identificação e localização no endereçamento da arquitetura atual, além de eliminar a hierarquia dos segmentos de rede e sub-rede que ocorre no endereçamento IP pelo uso de máscaras; atenuar a necessidade de vários endereços, sendo que nesse modelo é utilizado apenas um endereço unificado para comunicação entre as entidades; suporte semântico das necessidades de comunicação por uma camada de serviço; realizar injeção de requisitos novos nas camadas intermediárias sem que aumente a complexidade do modelo. Todos esses objetivos refletem em soluções de alguns gargalos da arquitetura TCP/IP, alguns já discutidos na seção 2.2.

O Modelo de Títulos deu origem à arquitetura ETArch (SILVA, 2013). A proposta é que sejam resolvidos, naturalmente, pelas camadas inferiores, problemas que atualmente são solucionadas pela camada de aplicação da arquitetura legada. Para isto, a arquitetura tem de materializar os propósitos do Modelo de Títulos. Discutiremos adiante como a ETArch se propõe a oferecer requisitos da Internet do Futuro, tais como: *multicast*, mobilidade, segurança, QoS, QoE, dentre outros.

Mobilidade não está presente desde o começo da Internet. As inovações da ETArch referente ao endereçamento horizontal e aos aspectos de identificação e localização viabilizam o serviço *multicast* e a questão da mobilidade, já que não será preciso modificar a identificação do dispositivo em movimento. O *workspace* (SILVA, 2013) é essencialmente um barramento lógico, que se constitui em um grupo *multicast* e é naturalmente suportado pela arquitetura.

As limitações de segurança da arquitetura legada já foram discutidas na seção 2.2.

A ETArch, no entanto, apesar de ter iniciado uma solução no âmbito de segurança de rede (MELO et al., 2017a), ainda não consegue cumprir os requisitos necessários de autenticação, confidencialidade, integridade e controle de acesso. A função desse trabalho é elaborar um protocolo de segurança *multicast* que cumpra esses requisitos e construa um ambiente confiável de comunicação.

Outro requisito importante a ser considerado é o QoS (*Quality of Service*) e o QoE (*Quality of Experience*). Ambos os conceitos são fortemente interligados, principalmente para aplicações de fluxo contínuo de mídia e aplicações de mídia que executam em tempo-real, como é o caso de aplicações VoIP. Para a voz, atrasos menores que 150 ms não são percebidos pelo ouvido humano, atrasos entre 150 a 400 ms podem ser aceitáveis, e atrasos que ultrapassem 400 ms comprometerão o entendimento das frases (KUROSE; ROSS, 2013). Nesse caso, uma garantia de atraso através de mecanismos de QoS, implica em uma boa experiência (QoE) de uso da aplicação por parte do usuário.

A ETArch apresenta mecanismos que suportam recursos avançados de controle de rede, com o objetivo de habilitar aplicativos móveis multimídia com garantia de QoS ao longo do tempo (SILVA, 2013). É importante salientar que os requisitos fazem parte de um *workspace*, que naturalmente é *multicast*. A integração de QoS com uma arquitetura naturalmente *multicast* é bastante promissora.

O roteamento na ETArch não é realizado entre dois *hosts*, como a arquitetura legada; é um roteamento orientado a *workspace*, naturalmente *multicast*. A rota a ser definida considera os requisitos de todas as entidades que compõem ou estão anexadas a esse *workspace*, de tal forma que os caminhos de comunicação resguardecam a qualidade de serviço demandada pelas entidades comunicantes.

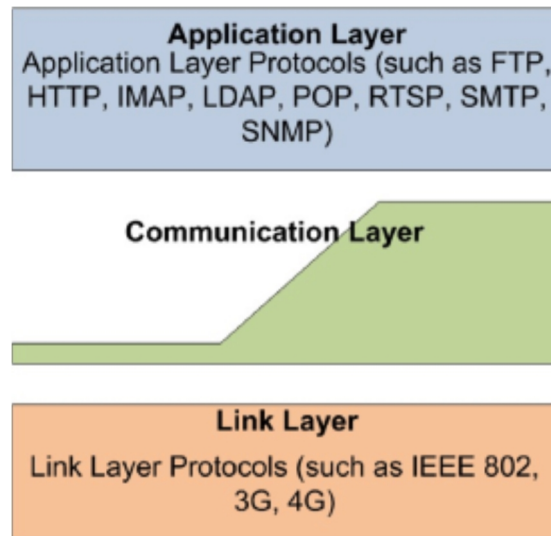
O cálculo da rota é feito pelo controlador SDN (*Software Defined Networking*) da ETArch (*DTS Agent - DTSA*) (NETO et al., 2016a), e é executado antes mesmo que a entidade de comunicação receba o conteúdo requisitado, ou seja, o comportamento do controlador no caso do algoritmo de roteamento é pró-ativo. Dessa forma, algumas vantagens podem ser citadas: o algoritmo é flexível; não haverá custo de roteamento por parte dos elementos de rede, pois sua função depois de calculada a rota, resume-se apenas no encaminhamento das primitivas.

As seções subsequentes tratarão de forma detalhada os aspectos conceituais da ETArch.

3.1 Camadas ETArch

O padrão arquitetural em camadas (*Layer*) é adotado como ponto de partida da especificação da arquitetura ETArch, também adotado pelo Modelo de Referência OSI (*Open Systems Interconnection*) (TANENBAUM et al., 2003), que é muito eficiente para garantir o desacoplamento de problemas ortogonais, o que contribui para a alta coesão dos protocolos que a compõe.

Figura 4 – Arquitetura ETArch



Fonte: (SILVA, 2013).

A ETArch, apesar da abordagem *clean slate*, apenas reexamina as camadas de rede e transporte da arquitetura legada. A camada física e de aplicação, que sofreram maiores evoluções com o crescimento em escala da Internet, são preservadas. Na arquitetura TCP/IP, a pilha de comunicação da arquitetura Internet depende das camadas de transporte (TCP/UDP/SCTP) e rede (IP). Na arquitetura ETArch, essas camadas são substituídas pela camada de comunicação que possui altura flexível.

O formato não usual da camada de comunicação indica a flexibilidade da arquitetura em se adaptar aos requisitos da aplicação. Aplicações com mais requisitos utilizam mais módulos e funções de rede, enquanto aplicações sem nenhum requisito, estabelecem comunicação "quase" direta com a camada física.

3.2 Entidade

Entidade é o elemento fundamental da ETArch. Pode ser definido como qualquer coisa que tenha capacidade de se comunicar, como um *host*, *smartphone*, *Network Element* (NE), etc. Uma entidade possui ao menos um título, e um conjunto de requisitos. A diferença de concepção em relação ao modelo tradicional está na forma de definição dos requisitos (PEREIRA, 2012).

Um requisito é uma demanda que a entidade necessita. Antes de iniciar a comunicação, uma entidade passa às camadas inferiores os requisitos que precisa para que a experiência do usuário seja satisfatória. A rede deve ser capaz de atender os requisitos dessa comunicação.

Desse ponto de vista, a arquitetura ETArch pode satisfazer diferentes conjuntos de requisitos propostos por diferentes visões de Internet do futuro (SILVA, 2013): centradas

em serviços (*Service-Centric*) (MARTÍNEZ et al., 2012); centrada em conteúdo (*Content-Centric*) (JACOBSON et al., 2009); dispositivos em geral (IoT - *Internet of Things*) (GUBBI et al., 2013); e ainda pode suportar a visão atual baseadas em *hosts*.

3.3 Título

Título é definido como um identificador único, não ambíguo e independente da topologia da rede subjacente (PEREIRA, 2012). Uma entidade tem uma relação unívoca com títulos, ou seja, uma entidade possui um ou mais títulos, dependendo da necessidade. Na prática, qualquer entidade comunicante possui pelo menos um nome válido, e diferentemente da arquitetura legada, esse identificador independe da sua localização, facilitando sua mobilidade. O título tem papel fundamental na implementação do endereçamento horizontal.

O *workspace*, que será apresentado adiante, é um barramento lógico de comunicação *multicast*. Em termos práticos, representa um conjunto de entidades que participam de uma comunicação. Ele também é representado por um título. Quando uma entidade deseja participar dessa comunicação, ela requisita esta admissão através do título do *workspace*. A rede é a responsável pela configuração dos elementos de rede para executar essa admissão.

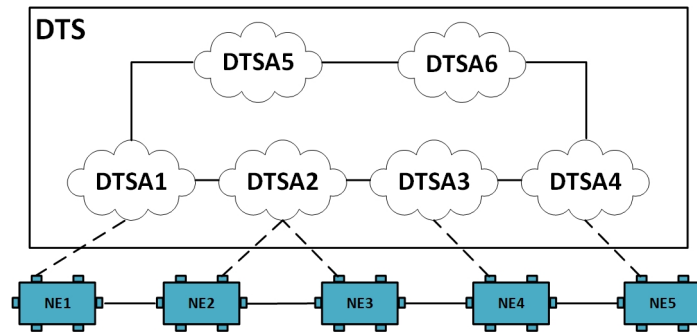
Esse tipo de visão de funcionamento da arquitetura desacopla a comunicação estabelecida ou o conteúdo disponibilizado de qualquer tipo de localização. Não é preciso da localização das entidades do grupo para que a rede execute a admissão de uma nova entidade (solicitadora do recurso) nessa comunicação. Nessa arquitetura, as primitivas de controle carregam o nome do *workspace* ao invés do endereço de origem e destino. A justificativa é que o crescimento do conteúdo da rede legada proporciona problemas de escalabilidade, agravada ainda mais pela comunicação *peer-to-peer*.

3.4 Domain Title Service

O *Domain Title Service* (DTS) representa o conjunto universo de todo o plano de controle existente na arquitetura ETArch. Dentro desse conjunto, há vários agentes, denominados DTSA's (*Domain Title Service Agent*), que viabilizam a divisão da rede em subconjuntos gerenciáveis. Cada um desses subconjuntos pode ser visto como um domínio distinto da rede, cada qual gerenciado por agentes do DTS. O DTS, como conjunto universo ou domínio maior, tem o papel de gerenciar todos os subdomínios, realizando dessa forma, a orquestração total das funcionalidades de controle.

Dentre essas funções, podemos citar: resolução de títulos de *workspace* para viabilizar a comunicação *multicast*; atribuição de títulos para as entidades comunicantes; gerenciamento do ciclo de vida de todas as entidades pertencentes ao DTS; cálculo da melhor rota

Figura 5 – Representação do DTS



Adaptada de (SILVA, 2013).

para a extensão do *workspace*; oferecer qualidade de serviço para as entidades comunicantes. O DTS é um ambiente distribuído que gerencia todas as funcionalidades de controle e torna possível a realização de comunicação entre entidades na arquitetura ETArch.

Na figura 5, representativa do conjunto universo DTS, os seus agentes (DTSA1 a DTSA6) fazem parte do que denomina-se plano de controle e os NEs (NE1 a NE5) fazem parte do que denomina-se plano de dados. O plano de controle é a inteligência que configura os elementos de rede a fim de preparar a comunicação de dados *multicast*, ou seja, gerenciar o ciclo de vida do *workspace*, que é uma das funções do DTS citadas anteriormente, prepara o ambiente para o encaminhamento de dados.

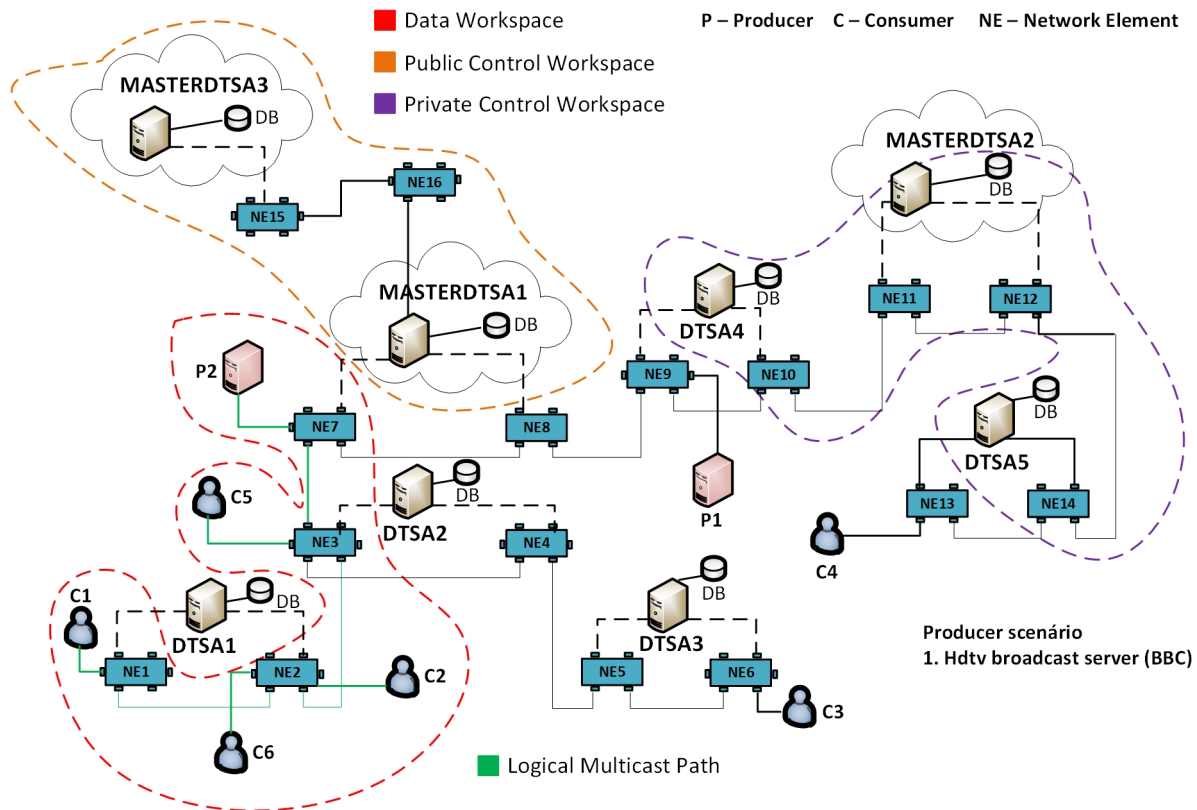
3.5 Workspace

O *workspace* (SILVA et al., 2013) é um barramento lógico, independente de topologia, na qual entidades podem se ligar (*attach*) para participar de um domínio de comunicação *multicast*. O *workspace* pode representar um chat, uma transmissão HDTV feita pela BBC (British Broadcasting Corporation), o serviço de um site de leilão eletrônico, o serviço de um site de *E-Commerce*, etc. O *workspace*, nesse contexto, pode representar uma "coisa oferecida qualquer".

Cada uma dessas representações possui requisitos mínimos para que o usuário tenha uma experiência satisfatória. É responsabilidade do *workspace* manter a qualidade de serviço da "coisa representada", ou seja, todas as entidades pertencentes ao grupo de comunicação receberão a qualidade de serviço necessária para que o usuário possa ter um QoE satisfatório.

Na figura 6, podemos visualizar, para fins didáticos, um exemplo de topologia da arquitetura ETArch. O DTS, não representado na figura, é o conjunto universo que detém três domínios maiores, controlados pelo MASTERDTSA1, MASTERDTSA2 e MASTERDTSA3. O MASTERDTSA1, por sua vez, possui 3 subdomínios, compostos pelo DTSA1, DTSA2, e DTSA3. O MASTERDTSA2 possui dois subdomínios, compostos por

Figura 6 – Exemplo de topologia da ETArch



DTSA4 e DTSA5. O MASTERDTSA3 não possui subdomínio algum.

Para exemplificarmos o DTS em uma topologia real, ele é o espaço formado por todos esses domínios/subdomínios e é responsável por gerenciá-los corretamente a fim de manter a comunicação de todas as entidades envolvidas. Consegue-se essa tarefa, gerenciando o ciclo de vida das entidades, *workspaces*, etc. que estejam envolvidos nessa estrutura. Na presença de alguma falha na topologia, ou na modificação das métricas dos enlaces, o DTS é capaz de reconstituir o *workspace* a fim de manter a comunicação e a qualidade de serviço demandada pelas entidades envolvidas. Para auxiliá-lo, ele possui dois agentes que serão explanados nas seções subsequentes: o DTSA e o *Master DTS Agent* (MDTSA). Esses dois agentes mantêm o gerenciamento dos três *workspaces* representados na figura: o *workspace* de dados, tracejada em vermelho, o *workspace* de controle público, tracejada em laranja, e o *workspace* de controle privado, tracejado em roxo.

O termo 'Público' na nomeação do *workspace* não quer dizer acesso irrestrito, mas tem analogia com as redes Carriers em telecomunicações, onde esta característica é empregada no suporte de interações de controle inter-domínios.

O *workspace* é fundamental na implantação dos conceitos de endereçamento horizontal. Quando uma entidade quer se anexar ao *workspace*, ela requisita o *'attach'* através de uma primitiva de controle, utilizando para isso o título do *workspace*, que é o endereço de destino das primitivas durante a comunicação. Se o processo de solicitação for realizado com sucesso, a entidade começa a receber os serviços disponibilizados pelo *workspace*.

Todos os consumidores do *workspace* de dados da Fig. 6 (C1, C2, C5, C6) fizeram esse processo, e depois de aceito, começaram a receber os serviços disponibilizados pelo produtor P2, que pode ser por exemplo, uma transmissão de vídeo HDTV disponibilizado por alguns dos servidores da BBC.

3.6 DTSA

O DTSA é um controlador ETArch SDN. Atualmente, se trata de um superconjunto das funcionalidades disponíveis no controlador introduzido pelo OpenFlow (MCKEOWN et al., 2008). Esforços atuais já estão sendo feitos na elaboração de um *switch* e controlador ETArch independente de protocolo OpenFlow.

As funções do DTS, discutidas na seção 3.4, são distribuídas para seus agentes: DTSA e MDTSA. Esses dois agentes materializam todas as funções do DTS, ou seja, desempenham todas as operações que mantêm a comunicação dentro desse espaço distribuído.

Um DTSA tem informações e é responsável pelo gerenciamento da comunicação de entidades que pertencem ao seu domínio. Na figura 6, C1, C2, C6, NE1, NE2 são entidades que pertencem ao domínio do DTSA1. Por este motivo, o DTSA1 conhece toda a topologia de seu próprio domínio, todos os *workspaces* criados por suas próprias entidades ou estendidos de outros domínios. O *workspace* de dados da Fig. 6 é criado no domínio do MASTERDTSA1, que também está desempenhando, nesse caso, o papel de DTSA. Esse *workspace* está presente em outros dois domínios: o domínio do DTSA2 e o domínio do DTSA1. Dessa forma, tanto o DTSA1, quanto o DTSA2 possuem em seus registros informações desse *workspace* de dados criado por P2. Dizemos que essas informações são de um *workspace* estendido, pois não foi criado nem pelo DTSA1, nem pelo DTSA2, e sim, foi estendido do domínio pertencente ao MDTSA1.

Um exemplo concreto de funcionalidade de um DTSA é o cálculo de rotas para extensão do *workspace*. Quando uma entidade quer entrar na comunicação, por exemplo, a entidade C2, da Fig. 6, ela requisita essa operação através de uma primitiva de controle. O DTSA1 recebe essa primitiva de controle, registra essa entidade em seu repositório, e logo depois, estende o *workspace* até C2, como mostrado. Nesse caso, o DTSA1 já detinha a informação desse *workspace* em seus registros, pois já o havia estendido anteriormente para C1 e C6. Dessa forma, ele mesmo (DTSA1) estende o *workspace* através de um algoritmo de roteamento intra-domínio. Caso o DTSA1 não conhecesse o *workspace* de dados solicitado por C2, o processo de extensão seria outro, e sua execução seria desempenhada pelo algoritmo de roteamento inter-domínio.

3.6.1 MDTSA

O DTSA, como visto, possui as informações necessárias para gerenciar a comunicação do seu domínio, ou seja, das entidades que controla. Na Fig. 6, podemos observar que o

DTSA4 gerencia o subdomínio composto por NE9, NE10 e P1; e que o DTSA5 gerencia o subdomínio composto por NE13, NE14 e C4. No entanto, esse conjunto de DTSA's ou subdomínios pertencem a um domínio maior, que é controlado por um tipo especial de agente: o MASTERDTSA2.

Poderíamos considerar, por exemplo, que o MASTERDTSA2 representa uma universidade e que cada um dos subdomínios, DTSA4 e DTSA5, representa um departamento: Física e Matemática, respectivamente. A primeira função desse MDTSA seria conhecer toda a topologia do domínio, a fim de conseguir gerenciar a comunicação entre vários subdomínios diferentes. No caso da Fig. 6, imagine que o departamento de física quer se comunicar com o departamento de Matemática. A visão geral de todos os departamentos da universidade através de um grafo auxilia essa comunicação. Como segunda função, o MDTSA tem de externalizar solicitações de controle internas feitas pelos seus DTSA's e internalizar funções de controle externas. Imagine que o departamento de matemática queira fornecer um curso a distância, e que os alunos se encontrem em diversas cidades distintas, ou, que exista um aluno na universidade que esteja participando de um curso a distância ministrada na universidade de Cambridge.

Tecnicamente, para que os departamentos se comuniquem ou para que os alunos consigam assistir seus respectivos cursos, como mencionado acima, haverá a extensão do *workspace* em domínios distintos, executados em dois contextos diferentes: o primeiro é a extensão do *workspace* de dados dentro do domínio do MDTSA, ou seja, entre domínios DTSA's que pertencem a um único domínio MDTSA; o segundo, refere-se à extensão do *workspace* entre domínios MDTSA's distintos. Essa extensão acontece através dos chamados *workspaces* de controle e serão detalhados em uma subseção subsequente. O primeiro contexto pode ser exemplificado na Fig. 6, através do *workspace* de controle privado (tracejado em linha roxa), onde o DTSA4 pode se comunicar com o DTSA5, ou qualquer um desses DTSA's podem se comunicar com o MASTERDTSA2. O segundo contexto pode ser exemplificado através do *workspace* de controle público (tracejado em linha laranja), onde há a comunicação do MASTERDTSA1 e MASTERDTSA3.

O MDTSA é um superconjunto do DTSA, ou seja, um MDTSA pode desempenhar função de DTSA, mas o contrário não se consuma. Esses dois agentes desempenham apenas funções de controle e preparam a rede para o plano de dados. Considerando que o plano de controle tem característica de tráfego especiais e como tal, precisa de uma qualidade de serviço prioritária, a configuração dos chamados *workspaces* de controle é estabelecida na inicialização da rede.

3.7 Workspace de dados

Na arquitetura ETArch, as entidades não estão interessadas em *hosts*, ou aplicações, ou conteúdos. As entidades se interessam por um conjunto abstrato de propriedades, que

denominamos de "coisa oferecida". Na Fig. 6, podemos verificar que P2, C1, C2, C5, C6, NE1, NE2, NE3 e NE7 são entidades que pertencem ao *workspace* de dados (tracejado em vermelho). Todos os consumidores estão utilizando a "coisa oferecida", nesse caso, pelo produtor P2. Nada impede que o cenário fosse outro, e que produtores agissem como consumidores e vice-versa, como é o caso de um chat.

Na criação de um *workspace*, o plano de controle configura os NEs para que a primitiva enviada chegue em todas as entidades comunicantes. Na Fig. 6, percebe-se a criação do caminho lógico *multicast* (em verde) que chega em todos os consumidores envolvidos. Inicialmente, só esse caminho é configurado, o que torna a comunicação *multicast* uma propriedade intrínseca do *workspace*.

No cenário da Fig. 6, onde P2 distribui informações HDTV, só esse caminho lógico *multicast* (em verde) é suficiente para a distribuição da mídia, que será recebida por todas as entidades do grupo.

3.8 Workspace de controle

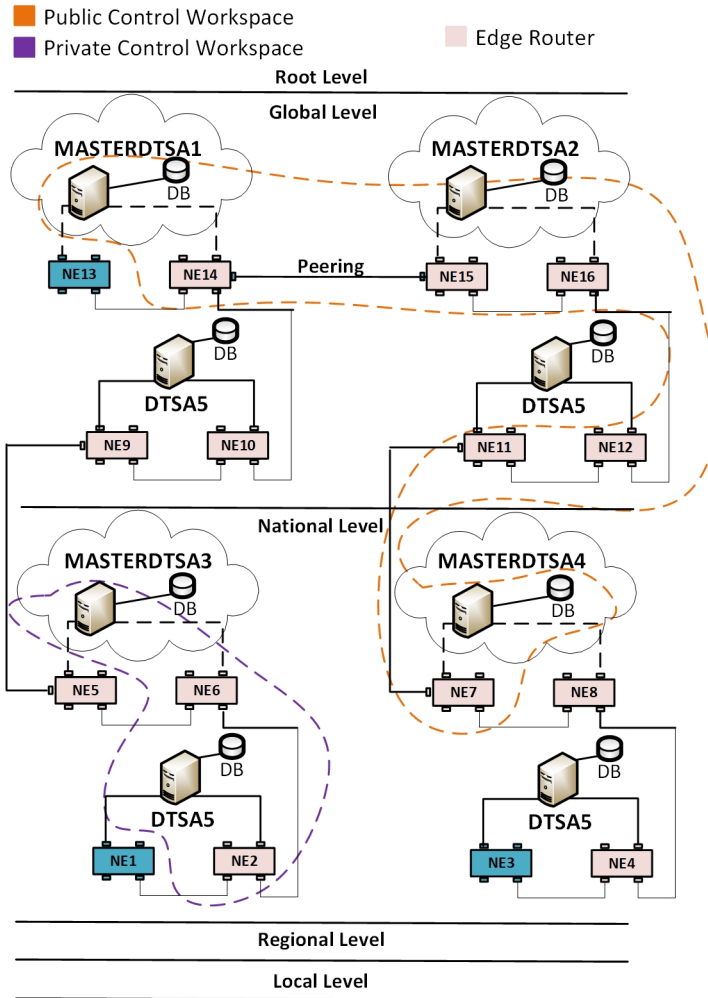
O *workspace* de controle possui todas as propriedades de *workspaces* já apresentadas em 3.5, porém, são exclusivas para as comunicações de controle da rede. Essas comunicações são o que torna possível o gerenciamento do ciclo de vida de *workspaces*, entidades, etc. Sem o plano de controle, não seria possível a comunicação entre as entidades. Todas as funcionalidades necessárias são regidas por dois protocolos: ETCP (*Entity Title Control Protocol*) e DTSCP (*Domain Title Service Control Protocol*) (SILVA, 2013).

O *workspace* de controle é utilizado para comunicação inter-domínio. Como exemplo, se uma entidade solicita a busca por um *workspace* de dados que não foi criada por uma entidade do seu próprio domínio e, portanto, não é conhecido por seu DTSA, haverá três possibilidades de comunicação no plano de controle, quais sejam: i) entre DTSA's; ii) entre um DTSA e seu respectivo MDTSAs; e, iii) entre MDTSAs. É bom frisar que a comunicação inicial entre entidade e DTSA, quando há o pedido pelo *workspace* de dados, é feito pelo plano de controle, e a comunicação é entre entidade e DTSA.

Resumidamente, a entidade primeiramente se registra no DTSA. Toda entidade comunicante, obrigatoriamente, tem de se registrar no DTSA do seu domínio. Logo após, ela solicita participar de uma comunicação, através de uma primitiva de controle WORKSPACE_ATTACH (Protocolo ETCP), carregando nessa primitiva o título do *workspace* que deseja participar.

Como esse DTSA não possui informações do *workspace* solicitado, ele pedirá auxílio ao seu MDTSAs, através da primitiva WORKSPACE_LOOKUP (Protocolo DTSCP). Tomando como exemplo, se a entidade solicitadora é a C1 da Fig. 6, e o DTSA1 não reconhece o *workspace* requerido, então ele pedirá auxílio para o MASTERDTSA1, através do NE2 (elemento de borda que levará a solicitação de controle apropriada para outros

Figura 7 – Organização do DTS em escala mundial



domínios), utilizando para isso o *workspace* de controle privado (tracejado em roxo) na Fig. 6. Apesar desse *workspace* não estar desenhado no domínio MASTERD TSA1, ele existe e é semelhante ao desenhado no domínio do MASTERD TSA2. Nota-se que esse *workspace* é responsável pela comunicação de (i) e (ii), descritas anteriormente.

Caso o MASTERD TSA1 também não reconheça tal *workspace*, ele solicitará auxílio para os MD TSAs vizinhos. No caso desse exemplo, para as entidades MASTERD TSA2 e MASTERD TSA3. Essa comunicação será feita através do *workspace* de controle público (tracejado em laranja) na Fig. 6. Nessa figura, só está desenhado o *workspace* de controle público que comunica o MASTERD TSA1 e MASTERD TSA3, porém, existe um *workspace* de controle público nessa topologia, que conecta MASTERD TSA1 e MASTERD TSA2; esse *workspace* não foi desenhado na figura por conta de simplificá-la, a fim de torná-la mais didática. Nota-se que esse *workspace* é responsável pela comunicação de (iii), descrita anteriormente.

Nesse ponto, fica claro que esses dois *workspaces* de controle são importantíssimos para a comunicação de dados em nível global, inter-domínios. Dentre as inúmeras funções que essa comunicação realiza, uma delas é a extensão do *workspace* de dados para domínios

diferentes, ou seja, na prática, é a forma do servidor de vídeo distribuir *streaming* entre ISPs distintos em nível global.

Importante mencionar que o MDTSA conhece toda a topologia do seu domínio, tem o registro de todos os DTSA's, *workspaces* e NEs sob seu controle. O DTSA é um subdomínio e conhece apenas a topologia, *workspaces* e NEs desse subespaço. As comunicações de controle são necessárias porque a informação está distribuída no espaço do DTS, ou seja, nenhuma entidade conhece todas as informações da estrutura topológica da arquitetura ETArch em escala mundial.

Na prática, esses conhecimentos parciais de DTSA's e MDTSA's é feito na inicialização da rede, através de arquivos que contêm uma especificação *eXtensible Markup Language* (XML). Basicamente, cada DTSA deve conhecer seus vizinhos e a forma à qual se conectam. Assim como acontece no protocolo BGP, um DTSA tem conhecimento sobre os NEs de borda do seu domínio.

A Fig. 7 mostra o DTS como sendo um sistema distribuído em nível mundial e como os *workspaces* de controle auxiliam nessa comunicação. O *workspace* de controle privado (tracejado em roxo) faz a comunicação entre o DTSA5 e o MASTERDTSA3. O *workspace* de controle público (tracejado em laranja) realiza a comunicação entre dois ou mais níveis da organização DTS. Na figura referida, o *workspace* público está fazendo a comunicação entre o MASTERDTSA4, presente no nível "*National Level*" e o MASTERDTSA2, presente no nível "*Global Level*". Esse *workspace* também faz o *Peering* entre MASTERDTSA1 e MASTERDTSA2. Se esses dois MDTSA's representassem dois ISPs de nível 1, poderíamos dizer que essa troca de informações tende a alcançar o escopo global das informações existentes nessa rede de escala mundial, ou seja, essa troca de informações entre domínios dá acesso a todos os *workspaces* criados no planeta. Uma entidade poderia solicitar WORKSPACE_ATTACH (Protocolo ETCP) a qualquer *workspace* pertencente a qualquer domínio dessa rede. É uma abordagem semelhante àquela adotada pela rede legada ao que se refere aos Tiers, onde uma rede pode chegar a qualquer outra rede da Internet.

A arquitetura ETArch organizou sua estrutura em 4 níveis: nível local, com escopo para empresas, residências, etc.; nível regional, que abrange vários níveis locais, como os ISPs da Internet; nível nacional, que abrange diversos níveis regionais; e nível raiz ou global, que seria os ISPs de nível 1 da Internet. Os *tiers* "*Regional Level*" e "*Local Level*" estão apenas indicados na Fig. 7, mas apresentam a mesma lógica de estrutura das demais apresentadas na figura.

Quando o DTSA não possuir informações sobre determinado *workspace*, significa que esse *workspace* não está presente nesse subdomínio. Nesse caso, o DTSA deverá requisitar o *workspace* de dados ao seu MDTSA. Essa requisição é feita através de um *workspace* de controle privado. Se o MDTSA do nível local também não possui esta informação, ele deverá requisitar o *workspace* de dados ao nível regional. Essa requisição é feita através de um *workspace* de controle público. Caso o *workspace* de dados especificado não esteja no

nível regional, o MDTSA deve requisitar o *workspace* de dados ao nível nacional, e assim a pesquisa continua, até chegar no *Root Level*, que pode devolver uma resposta positiva, e estender o *workspace* até a entidade solicitante, ou devolver uma resposta negativa, que afirma que tal *workspace* solicitado não existe na estrutura de redes mundial ETArch.

Essas operações de controle detalhadas acima serão importantíssimas para a execução do algoritmo de roteamento orientado a *workspace*, que estende o *workspace* até a entidade solicitante, para que ela comece a fazer parte da comunicação. Esse roteamento, como já dito, independe da localização física das entidades comunicantes.

3.9 Implementação atual da ETArch

Em 2012, o projeto "OpenFlow in Europe: Linking Infrastructure and Applications" (OFELIA) (SILVA, 2013) publicou uma chamada aberta à comunidade científica visando agregar novos parceiros ao projeto. A partir de alguns resultados já existentes da arquitetura ETArch, foi submetida uma proposta chamada "Extending and Deploying Ofelia in BRAzil" (EDOBRA) (GUIMARAES; CORUJO; AGUIAR, 2012), que pode ser dividida em duas partes: implantação de uma ilha na Faculdade de Computação da UFU (Universidade Federal de Uberlândia); e, desenvolvimento e experimentação da arquitetura ETArch no ambiente físico disponibilizado.

A criação da ilha Ofelia no Brasil foi executada com sucesso. Essa ilha contém servidores para controle, servidores para criação de máquinas virtuais, *switches OpenFlow*, *NETFPGAs* e fibra óptica diretamente conectada à Europa.

Todas as implementações da ETArch, desde então, em termos de mobilidade (SILVA, 2013), *multicast* (GONÇALVES et al., 2014) (SILVA, 2013), QoS e QoE (LEMA, 2014), roteamento (NETO et al., 2016b) e segurança (MELO et al., 2017b) foram implementados utilizando o DTSA do projeto EDOBRA.

3.9.1 Implementação do DTSA no EDOBRA

A principal entidade da arquitetura é o DTSA (Controlador SDN estendido), e, portanto, sua implementação é fundamental para a materialização da ETArch. Todas as funcionalidades do DTSA são disponibilizadas através dos protocolos ETCP e DTSCP (protocolos de controle da arquitetura). Algumas de suas funcionalidades, indispensáveis para o entendimento desse trabalho, serão descritas em seções subjacentes.

O desenvolvimento se deu em linguagem de programação Java, utilizando o protocolo OpenFlow através do controlador FloodLight (SWITCH, 2012) e do conceito de JAIN SLEE (*Java APIs for Integrated Networks - Service Logic Execution Environment*) (MENDONÇA, 2009).

A extensão do FloodLight consiste na criação de um novo módulo que instancia o DTSA. O DTSA será composto de vários serviços com alta coesão e baixo acoplamento, a fim de atender todas as funcionalidades oferecidas pelo DTS.

Quanto ao JAIN SLEE, permite a criação de aplicações que possuam requisitos como alta vazão, baixa latência, escalabilidade e disponibilidade (FEMMINELLA et al., 2011), e quando juntada aos serviços do DTSA, adiciona-se segurança, QoS, QoE, *multicasting* e mobilidade na comunicação das entidades; todos esses requisitos são fundamentais em um ambiente de telecomunicações.

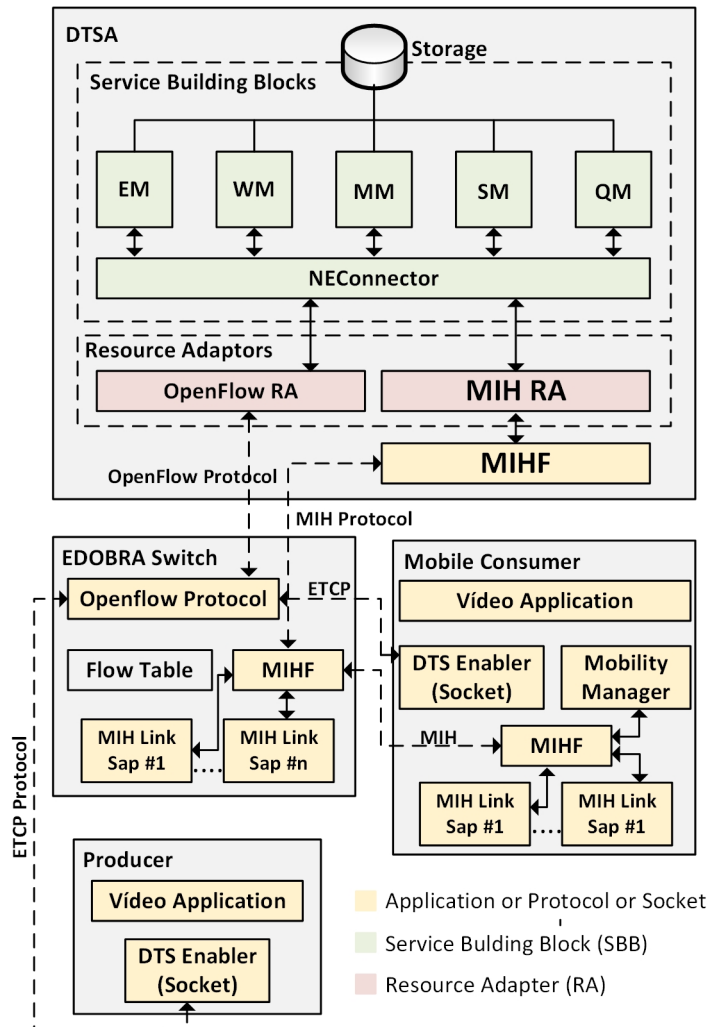
Dois conceitos são essenciais para entender o modelo de componentes JAIN SLEE: SBB (*Service Building Blocks*) e RA (*Resource Adaptor*). SBB contém a aplicação, a lógica do serviço. Os SBBs possuem um ou mais filhos e são organizados em um grafo de componentes. Ele pode capturar os serviços de um RA (*Resource Adapter*) e lançá-lo a outro SBB, para que receba um tratamento específico. Um RA adapta recursos externos para que sejam utilizados em conjunto com o JAIN SLEE. Na prática, ele abstrai protocolos de rede, transformando suas primitivas em serviços. Esses serviços, serão enviados às SBBs para devido tratamento.

Um dos motivos da escolha do JAIN SLEE é que o Mobicents (SLEE, 2009) já é uma realidade no mundo das operadoras de telecomunicações. Uma operadora pode ter diversos RAs para diferentes protocolos de rede, como por exemplo, RA para HTTP (*Hypertext Transfer Protocol*) (FIELDING et al., 1999), RA para SIP (*Session Initiation Protocol*) (ROSENBERG et al., 2004) e assim para quaisquer outros protocolos. Dessa forma, a operadora pode aproveitar os RAs disponíveis na comunidade e utilizá-los, tendo apenas o trabalho de construir SBBs específicos para suas regras de negócio.

Quanto à especificação do MIH (*Media Independent Handover*), foi elaborada por um grupo de trabalho denominado IEEE 802.21, e seu objetivo principal era a criação de uma *Generic Link Layer* (GLL) (GROUP et al., 2009) que solucionasse o problema do handover vertical de forma transparente para o usuário. Entende-se por handover vertical a mudança de ponto de acesso entre redes sem fio de diferentes padrões e tecnologias realizada por uma entidade móvel.

A camada de abstração GLL foi posta entre as camadas de enlace e de rede. As diversas informações recebidas da rede (eventos de enlace) são passadas para as camadas superiores através da GLL. Depois de analisar essas informações, os módulos do DTSA (SBBs), representados no primeiro retângulo tracejado da Fig. 8, conseguem executar comandos de enlace através da abstração dessa mesma camada. Essa camada faz a interface ou a ligação de enlaces de diferentes tecnologias com as funcionalidades do DTSA. O IEEE 802.21 (GROUP et al., 2009) define o *MIH Function* (MIHF) para prover as funcionalidades da GLL, que oferece três serviços básicos: *Media Independent Event Service* (MIES); *Media Independent Command Service* (MICS) e *Media Independent Information Service* (MIIS). Não é escopo desse trabalho detalhar esses serviços, porém, em (SILVA, 2013)

Figura 8 – Arquitetura da integração do ODTONE com o DTSA



encontra-se a descrição detalhada do processo de *handover* realizado pela ETArch e como as funcionalidades do MIH são utilizadas pela arquitetura.

Caso a entidade móvel possua várias interfaces de diferentes tecnologias sem fio, a escolha da mudança de tecnologia, bem como da mudança de ponto de acesso passa por vários fatores, quais sejam: análise do limite da cobertura dos pontos de acesso; análise das bandas disponíveis em diferentes pontos de acesso; consumo de energia das interfaces da entidade quando conectados aos pontos de acesso disponíveis; e outros. Se o dispositivo apresenta, por exemplo, baixa carga de bateria, a entidade fará o *handover* utilizando a tecnologia de interface que consuma menos energia. Nesse contexto, há a preocupação com a qualidade de serviço disponibilizada para a entidade comunicante segundo critérios pré-estabelecidos.

A Fig. 8 mostra o DTSA desenvolvido no projeto EDOBRA. Foram desenvolvidos dois RAs: OpenFlow RA e MIH RA. O RA do OpenFlow abstrai o protocolo OpenFlow, e o objetivo principal é fazer a comunicação da camada física subjacente da rede com os componentes do DTSA executados dentro do JAIN SLEE. Por outro lado, o RA do

MIH abstrai o ODTONE (RUI et al., 2011), que é uma implementação de código aberto da especificação IEEE 802.21. O principal objetivo desse RA é fazer a comunicação entre enlaces físicos de redes sem fio e os componentes da ETArch executados dentro do JAIN SLEE. Essa manobra possibilitará o *handover* (vertical ou horizontal) otimizado de entidades ligadas à um determinado *workspace*.

Essa mesma figura apresenta os SBBs que implementam as funcionalidades do DTSA.

O SBB NEconnector é uma interface ou camada genérica de comunicação entre a rede e o DTSA. Atualmente, os protocolos que se comunicam com o NEConnector é o MIH e o OpenFlow. No futuro, novos protocolos podem ser criados, bem como o OpenFlow e MIH podem ser substituídos. Em ambos os casos, os SBBs acima do NEConnector podem ser reaproveitados graças a essa interface de comunicação, que desacopla os SBBs dos serviços de enlace feitos pelos protocolos MIH e OpenFlow.

Os SBBs, situados acima do NEConnector, são responsáveis pelas funcionalidades do DTS, já citadas na seção 3.4. Essas funcionalidades são exclusivamente de controle e são responsáveis por preparar as comunicações de entidades, que são realizadas através dos *workspaces* de dados criados no espaço DTS. Todas essas funcionalidades são realizadas através dos protocolos ETCP e DTSCP. Cada um desses protocolos possui suas primitivas e APIs (*Application Programming Interface*). O ETCP oferece serviços que envolvem interações entre entidades e DTSAs, enquanto o DTSCP é responsável por interações entre DTSAs, ou entre MDTSAs, ou entre DTSAs e MDTSAs. Adiante, será detalhado alguns serviços desses protocolos que são fundamentais para o o entendimento desse trabalho. Em (SILVA, 2013) existe a descrição das APIs e primitivas de forma detalhada para cada um desses protocolos.

O SBB EntityManager (EM) recebe as primitivas responsáveis pelo registro e pela remoção de registro de uma determinada entidade. Antes de uma entidade pertencer a um determinado *workspace*, para consumir ou produzir serviços, ela precisa se registrar no seu respectivo DTSA. Essas primitivas receberão o tratamento adequado, e de acordo com a situação, haverá a manutenção do *storage* (banco de dados do DTSA), acrescentando ou deletando o registro de determinada entidade.

O SBB WorspaceManager (WM) recebe todas as primitivas relacionadas aos *workspaces*: criar ou excluir o *workspace*; inserir ou retirar uma entidade do *workspace*; alterar as características do *workspace*; dentre outras. O storage do DTSA também é manipulado por essas operações.

O SBB MobilityManager (MM) é responsável pela manutenção das entidades da arquitetura ETArch no processo de otimização do *handover* desempenhado pelo ODTONE. O MIH RA receberá a primitiva dos eventos de enlace subjacentes e lançará ao NEConnector, que então lançará ao MobilityManager, que é o responsável pelo tratamento da primitiva. O MM se comunicará com outros DTSAs, pois a mobilidade do dispositivo causa, necessariamente, a extensão do *workspace* dessa entidade até seu novo ponto de

Tabela 4 – Sequência de passos na solicitação de um *attach*

01	Entidade solicita um WORKSPACE_ATTACH, enviando o título do <i>workspace</i>
02	WORKSPACE_ATTACH chega até o NE ao qual a entidade está fisicamente ligada
03	NE encapsula WORKSPACE_ATTACH na primitiva PACKET_IN e lança para o controlador
04	PACKET_IN chega no controlador FloodLight (lib dentro do DTSA)
05	Floodlight encaminha o PACKET_IN para o OpenFlow RA
06	OpenFlow RA transforma o PACKET_IN em serviço e lança
07	Serviço PACKET_IN é capturado pelo SBB NEConnector
08	NEConnector faz análise do serviço PACKET_IN, recuperando a primitiva WORKSPACE_ATTACH
09	NEConnector lança o serviço WORKSPACE_ATTACH-ind
10	WorkspaceManager captura serviço WORKSPACE_ATTACH-ind
11	WorkspaceManager recupera informações sobre o <i>workspace</i> requisitado no seu <i>storage</i>
12	WorkspaceManager faz um update no storage acrescentando a entidade solicitante no <i>workspace</i>
13	WorkspaceManager lança serviço WORKSPACE_ATTACH-resp contendo as informações da resposta
14	NEConnector captura serviço WORKSPACE_ATTACH-resp
15	NEConnector solicita ao FloodLight que configure regras, nos <i>switches</i> OpenFlow, necessárias para que o <i>workspace</i> seja estendido até a entidade solicitante
16	FloodLight insere as regras na primitiva OpenFlow FLOW_MOD e a envia ao NE (<i>switches</i> OpenFlow)
17	NEConnector solicita ao FloodLight que responda para a entidade
18	FloodLight encapsula a resposta na primitiva OpenFlow PACKET_OUT e a envia ao NE da entidade solicitante
19	NE envia a resposta para a entidade

Fonte: (NETO et al., 2016a).

acesso. As manutenções necessárias ao *workspace* da entidade móvel são responsabilidade desse SBB. Esse processo é descrito detalhadamente em (SILVA, 2013).

O SBB SecurityManager (SM) (MELO et al., 2017b) é uma tentativa inicial de introduzir na ETArch autenticação e controle de acesso aos recursos oferecidos pelos protocolos de controle. O presente capítulo (seção 3.10) mostra as limitações dessas implementações iniciais de segurança e cria uma nova proposta de segurança de rede que possibilita que as aplicações possam solicitar serviços seguros conforme suas necessidades. A proposta engloba a construção de uma rede de confiança, onde cada entidade comunicante possa confiar nas outras entidades comunicantes que fazem parte do mesmo grupo ou contexto

de comunicação. Novos mecanismos de segurança que proporcionam autenticidade, confidencialidade, integridade, disponibilidade e gerenciamento de chaves fornecem os recursos necessários para esse fim. A entidade (aplicação, sensor, etc.) pode escolher os serviços de segurança que garantem o seu bom funcionamento através de um identificador de política de segurança. Todos esses mecanismos são detalhados no capítulo 4 e compõem o objeto da proposta desse trabalho, que é a elaboração de um novo protocolo de segurança para uma arquitetura de Internet do Futuro (ETArch é a arquitetura escolhida, onde será aplicada as funcionalidades da especificação do ESMP). O SBB SM será totalmente remodelado com o objetivo principal de cumprir os requisitos básicos/necessários de segurança que garantam todas as características (autenticidade, confidencialidade, etc.) citadas acima.

O QoS Manager (QM) possui um conjunto de funcionalidades que implementa o "*The Support of Mobile Sessions with High Transport Demand*" (SMART) (LEMA, 2014). É uma proposta parecida com o MPLS (ROSEN; VISWANATHAN; CALLON, 2001), porém, a técnica de roteamento será regida por *workspaces* e não mais por LSPs (*Label Switch Path*). O SMART apresenta como principal objetivo aprimorar a rede ETArch com novos mecanismos que suportam recursos avançados de controle a fim de garantir QoS para uma gama de aplicações sensíveis a características de métrica de rede, como largura de banda, tempo de resposta dos enlaces, atrasos de processamento e transmissão dos elementos de rede.

3.9.2 Visão geral do ETCP/DTSCP

Os protocolos de controle da arquitetura são responsáveis pelas funcionalidades oferecidas pelo DTS. Dois protocolos foram desenvolvidos para essa finalidade: ETCP e DTSCP. O primeiro é responsável pela comunicação de controle entre entidade e DTSA. O segundo, por sua vez, é responsável pela comunicação existente ente DTSA. É válido ressaltar que um MD TSA é um DTSA com algumas funcionalidades adicionais. Alguns serviços do protocolo ETCP são mostrados a seguir:

1. ENTITY_REGISTER: serviço pelo qual a entidade requisita seu registro no DTSA correspondente. Para tal, é passado como parâmetro o título e uma lista de requisitos necessários para que a entidade tenha uma comunicação satisfatória.
2. WORKSPACE_CREATE: serviço utilizado para criar um novo *workspace*. Uma entidade registrada, caso queira oferecer um serviço qualquer, tem que criar um *workspace* de comunicação para oferecê-lo. É o primeiro passo para que haja comunicação no plano de dados. Para tal, é passado como parâmetro o título do *workspace*, o título da entidade criadora desse *workspace* e as capacidades de comunicação que o *workspace* pode oferecer.

3. `WORKSPACE_ATTACH`: serviço pelo qual uma entidade registrada requisita sua participação em um certo grupo de comunicação (*workspace*). Para tal, é passado como parâmetro o título do *workspace* e o título da entidade que está requisitando essa participação.

Podemos descrever, brevemente, o funcionamento desse protocolo (ETCP) através da ilustração da Fig. 6. P2 é um produtor, ou seja, ele produz serviço para consumo. No caso, ele está oferecendo transmissão HDTV para todas as entidades que participam do *workspace*. Alguns passos foram necessários para que ele conseguisse desempenhar essa tarefa: (i) P2 registrou-se no seu DTSA correspondente, que conforme figura, é representado pelo MASTERDTSA1. Para tal, foi utilizado o serviço `ENTITY_REGISTER`; (ii) P2, após registrado, solicitou para o MASTERDTSA1 a criação de um *workspace* de dados (tracejado em vermelho) através do serviço `WORKSPACE_CREATE`. Logo após, esse *workspace* é configurado apenas em NE7, pois ainda não existe nenhuma entidade associada/atachada. Nesse momento, P2 já pode oferecer a transmissão HDTV para os participantes dessa comunicação; (iii) C1 requisita a sua associação no *workspace* criado por P2. Nesse momento, C1 passa a receber a transmissão HDTV produzida em P2. Logo após, C2, C5 e C6 realizam essa mesma operação. Nesse momento, a transmissão representada na Fig. 6 está sendo consumida por todas as entidades associadas ao *workspace* criado por P2.

A sequência de troca de primitivas dos serviços de controle do protocolo ETCP citados acima são similares, e portanto, detalhamos apenas um deles: `WORKSPACE_ATTACH`. O detalhamento desse serviço pode ser visto na Tab. 4. Em (SILVA, 2013) é descrito de forma detalhada as primitivas e serviços do protocolo ETCP.

Quanto aos serviços do DTSCP, alguns são mostrados a seguir:

1. `WORKSPACE_LOOKUP`: serviço pelo qual o DTSA/MDTSA tem que pesquisar um determinado *workspace* não conhecido por essas entidades. Essa pesquisa é crucial no momento em que uma entidade requisita a sua associação a um *workspace* não conhecido pelo DTSA/MDTSA correspondente. Se a solicitação do `WORKSPACE_LOOKUP` for positiva, o DTS, através dos seus agentes executará a extensão do *workspace* até a entidade solicitadora dessa comunicação. Esse serviço é crucial para qualquer tipo de procedimento inter-domínio.
2. `DTSA_REGISTER`: serviço responsável pela solicitação de registro de um DTSA em seu MDTSA correspondente. Para esse fim, é passado como parâmetro o título da entidade a ser registrada e a lista que representa os DTSA vizinhos ligados ao DTSA que está solicitando o registro. Essa lista tem o papel de encaminhar a primitiva de controle até o MDTSA através de NEs de borda, que possuem a função de realizar a comunicação de primitivas inter-domínio.

3. DTSA_MESSAGE: Serviço que permite trocas genéricas de primitivas entre DT-SAs. Para tal, um dos parâmetros passados é a primitiva a ser trocada.

O protocolo DTSCP pode ser visto de forma detalhada em (SILVA, 2013).

3.10 Segurança na ETArch: status quo

Esta seção apresenta algumas considerações sobre o status quo de segurança na ETArch. Como já mencionado na subseção 3.9.1, o SBB SM da Fig. 8 é responsável pela introdução de mecanismos de autenticação e de controle de acesso na arquitetura ETArch (MELO et al., 2017a).

A autenticação na ETArch (MELO et al., 2017a) pressupõe, inicialmente, que tanto o DTSA quanto as entidades registradas devem possuir uma par de chaves pública/privada e certificados válidos. O DTSA, nesse contexto, age como uma autoridade certificadora, e é o responsável pela confecção desses certificados. No caso do seu próprio, ele realiza uma autoassinatura digital desse documento. No caso dos certificados das entidades, todos eles são assinados digitalmente por seus respectivos DTSAs.

O procedimento da autenticação (MELO et al., 2017a) passa pela modificação de uma primitiva do protocolo ETCP (SILVA, 2013): ENTITY_REGISTER; são acrescentados dois novos campos. O primeiro, *accessInformation*, é utilizado para armazenar informações que serão utilizadas no momento da autenticação; o segundo, *authenticationType*, é o tipo de autenticação que será associada à entidade no momento do registro. Um exemplo de conteúdo do primeiro campo seria o certificado da entidade, e um exemplo de conteúdo do segundo campo seria "Por certificado"; ou seja, a distribuição de chaves públicas se dá através de certificados digitais.

O processo de autenticação se dá quando a entidade requisita um registro e envia um ENTITY_REGISTER para o DTSA. O SBB SM então autentica a informação. O procedimento detalhado dessa operação de autenticação não está exposto em Melo et al. (2017a). Logo após a autenticação, o SBB SM solicita ao NEConnector que envie a primitiva para seu SBB correspondente, que nesse caso é o SBB EM. O protocolo segue seu itinerário normal e se a operação for realizada com sucesso, o registro da entidade é efetivado no DTSA e uma resposta positiva é enviada para a entidade solicitante.

É importante salientar que o papel do SBB SM é interceptar a primitiva de registro da entidade antes que ela chegue no SBB EM. Depois desse ponto, se a entidade for autenticada com sucesso, a primitiva segue seu itinerário normal (como explicado acima); caso não seja autenticada, uma resposta negativa é devolvida à entidade solicitante.

Quanto ao controle de acesso (MELO et al., 2017a), a modificação mais importante foi uma alteração realizada na primitiva WORKSPACE_CREATE; o campo *accessControlList* (*Access Control List* - ACL) foi adicionado. Se trata de uma lista de entidades que podem participar do contexto de comunicação (*workspace*) que está sendo criado.

Tabela 5 – Mecanismos de mitigação de ataques da arquitetura ETArch (status quo)

Metas de Segurança	Ataques de Segurança	Mitigação
Confidencialidade	Espionagem (<i>Snooping</i>)	x
	Análise de tráfego (<i>Traffic analysis</i>)	v
Integridade	Modificação (<i>Modification</i>)	x
	Repúdio (<i>Repudiation</i>)	x
Disponibilidade	Negação de serviço (<i>Denial of service</i>) DoS	v
Autenticação	<i>Man-in-the-middle</i>	x
	Ataque por reflexão (<i>reflection attack</i>)	x
	Disfarce (<i>Masquerading</i>)	x
	Repetição (<i>Replaying</i>)	x
Segurança <i>multicast</i>	Apenas os ataques mitigados pela arquitetura	v

Adaptada de (KHONDOKER et al., 2014).

Essa informação é armazenada no DTSA, ou seja, o título do *workspace* é associado à um conjunto de entidades que podem futuramente participar da comunicação. Além disso, a priori, só a entidade proprietária do *workspace* pode solicitar alguma modificação nessa lista de entidades ou em quaisquer outros recursos dessa comunicação.

Uma das maneiras de avaliar a segurança da ETArch no que diz respeito ao seu status quo passa por dois passos: analisar os mecanismos de segurança da arquitetura, descobrindo possíveis vulnerabilidades; e, combinar essa análise com modelagens de ataques (KLOTI; KOTRONIS; SMITH, 2013). Esse modelo de análise faz parte dos métodos de avaliação e será detalhado no Cap. 5.

Quanto à autenticação, ela só acontece no momento de registro da entidade. Ataques de disfarce (*masquerading*), ataques de repetição (*replaying attack*), ataques por reflexão (*reflection attack*) e ataques *man-in-the-middle* são facilmente aplicáveis devido à essa verificação exclusiva. Esses ataques foram listados por uma razão óbvia: são os ataques que um processo de autenticação deveria amenizar; ou de forma ideal, solucionar totalmente.

No entanto, não foi objetivo e nem preocupação da ETArch até esse momento tratar desse assunto nessa ótica de "vulnerabilidades versus mecanismos de segurança versus ataques". Nesse caso, pode-se falar que não se cumpre nessa arquitetura, até então, os requisitos necessários de um processo de autenticação.

Quanto ao controle de acesso, o primeiro passo passa pela autenticação da entidade que está solicitando um recurso qualquer da rede (STALLINGS, 2014). Essa autenticação habilita o servidor de políticas de acesso a inferir/deferir solicitações de recursos. A partir do momento que a operação de autenticação está comprometida, os mecanismos de controle de acesso estão comprometidos e extremamente vulneráveis aos ataques citados anteriormente.

As contramedidas da arquitetura ETArch (status quo) a possíveis ataques são detalhadas a seguir.

ETArch não possui contramedidas ao ataque de espionagem (*snooping attack*) porque não possui nenhum mecanismo de encriptação nos planos de controle/dados.

Com relação a ataques de modificação de mensagens (*modification attack*); a ETArch não possui mecanismos de verificação de integridade nos planos de dados/controle.

Para ataques *man-in-the-middle* e reflexão (*reflection attack*); a ETArch não possui um mecanismo para autenticação mútua das entidades participantes da comunicação, portanto, não consegue atenuar nenhum desses ataques. Quanto ao disfarce (*masquerading attack*); se trata de outro problema, pois o mecanismo de autenticação do remetente é verificado apenas no momento do registro. Uma Entidade 02 (sem ao menos estar registrada) pode mascarar uma primitiva de controle, como se fosse a Entidade 01 (registrada). Se esse pacote se tratar de um ENTITY_UNREGISTER, o atacante (Entidade 02) pode cancelar o registro da Entidade 01 ou de quaisquer outras entidades registradas. Esse fato acontece, pois a única primitiva de controle que atravessa um processo de autenticação do remetente é a WORKSPACE_REGISTER (MELO et al., 2017a).

Quanto ao ataque de negação de serviço (DoS); a ETArch consegue atenuá-lo por dois motivos: o primeiro refere-se à capacidade do DTSA em atualizar automaticamente as rotas das primitivas de um *workspace* em caso de congestionamento ou outras intempéries provocadas por esse ataque; o segundo refere-se à iniciativa do projeto SMART na ETArch, que consegue gerenciar a admissão das entidades no *workspace* através de controle de largura de banda, de tal forma que uma entidade só consiga participar da comunicação se não houver sobrecarregamento dos elementos de rede.

O ataque de repúdio não pode ser evitado, pois as primitivas de controle podem ser mascaradas, e nesse caso, a Entidade 02 pode enviar uma primitiva pela Entidade 01. Nesse contexto, a ETArch não possui mecanismos para reconhecer o remetente da primitiva enviada (mascarada) pela Entidade 02. O DTSA, nessa situação, reconheceria a primitiva enviada pela Entidade 02 como sendo uma primitiva enviada pela Entidade 01. A primitiva mascarada descrita acima poderia ser a ENTITY_UNREGISTER.

Quanto aos ataques de análise de tráfego (*traffic analysis attack*), uma forma de atenuá-lo é ter um mecanismo para ocultar a identidade dos usuários (KHONDOKER et al., 2014) (STALLINGS, 2014). A Etarch possui esse mecanismo, pois o endereçamento de destino das primitivas se dá através do título do *workspace*; portanto, os títulos das entidades envolvidas na comunicação são ocultadas de um possível atacante.

A repetição (*replaying attack*) de primitivas não são atenuadas; a ETArch não possui um mecanismo para vincular as mensagens às sessões; dessa forma, não podem fazer um controle de sequenciação dessas mensagens.

Quanto à segurança *multicast*, a ETArch oferece o suporte à comunicação, porém não oferece a segurança necessária quanto aos seus conceitos mais básicos; porém, os mecanismos mitigados pela arquitetura, como análise de tráfego (*traffic analysis*) e negação de serviço (DoS), são atendidos no contexto da comunicação *multicast*.

A Tab. 5 possui um resumo dos mecanismos de mitigação de ataques por metas de segurança da arquitetura ETArch (status quo). Essa tabela se junta às informações de protocolos de outras arquiteturas para a realização de uma comparação final (no Cap. 5) entre o que está descrito nessa seção e o que está sendo proposto na sessão 4.3 (protocolo de comunicação *multicast* para arquiteturas de Internet do Futuro).

No próximo capítulo, é apresentada a proposta de um protocolo *multicast* de segurança, cujas funcionalidades são aplicadas na arquitetura ETArch.

Proposta de um protocolo de segurança *multicast* para uma arquitetura de Internet do Futuro

É sabido que uma das motivações existenciais de arquiteturas de Internet do Futuro é a preocupação constante com as demandas das aplicações atuais/futuras, que de certa forma evidenciaram na prática as limitações da arquitetura legada. Já foi discutido na seção 2.2 que alguns requisitos, como comunicação/segurança *multicast*, ainda não foram resolvidos por essa arquitetura e uma das causas é o sobrepeso colocado sobre o endereçamento de rede IP (localização/identificação). Não se consegue, dessa forma, endereçar intrinsecamente um subgrupo de entidades no nível de rede nem tão pouco oferecer segurança para esse tipo de comunicação.

A proposta desse trabalho é estabelecer no plano de controle os mecanismos necessários para oferecer comunicação *multicast* com nível de segurança diversificado, ou seja, a política de segurança será determinada pelo contexto de comunicação. Aplicações de transferências bancárias, de chats, de leilões eletrônicos, de transmissões de fluxo contínuo de vídeo (por exemplo, futebol) ou de transmissões de reuniões estratégicas de governos estaduais terão demandas diferentes quanto ao nível de segurança e, portanto, terão políticas de segurança distintas.

A primeira preocupação para atingir o objetivo de uma comunicação *multicast* segura está relacionada com a distribuição das chaves. A comunicação tende a ser mais segura quando as partes envolvidas já compartilham uma chave antes mesmo de qualquer contato. Sendo assim, é possível que todas as primitivas trocadas entre as entidades já estejam cifradas. Essa forma ideal de distribuição física nem sempre é possível, e a entrega manual pode se tornar desfavorável na medida que uma grande quantidade de entidades necessitem participar dessa comunicação.

A utilização de um centro de distribuição de chaves é possível graças ao conceito SDN, que propõe a separação do plano de controle em uma peça externa de software,

que no caso da ETArch, materializa-se no DTSA. O protocolo *multicast* de segurança de entidade (*Entity Security multicast Protocol* - ESMP) que será proposto; parte dele está localizado no DTSA, no módulo SecurityManager (SBB SM) da Fig. 8, e parte dele está localizado na entidade, na camada "*Communication Layer*" da Fig. 4. Todas as decisões da comunicação de segurança realizadas no plano de dados serão tomadas anteriormente pelo plano de controle. O SBB SM, onde se localiza o protocolo ESMP no DTSA, executará as operações necessárias (*handshakes*) para que a comunicação no plano de dados seja feita de forma segura. A porção do protocolo que se localiza na entidade (na *Communication Layer*), além de auxiliar na comunicação de controle com o DTSA, executará os serviços/mecanismos de segurança nas informações transmitidas entre as entidades que compõem o *workspace* de dados. Na prática, a inteligência das políticas adotadas conforme requisitos da aplicação são realizadas pelo DTSA (plano de controle), enquanto as entidades apenas executam as políticas, adotadas anteriormente, no plano de dados.

Essa utilização de SDN traz outra vantagem intrínseca da ETArch: novos recursos de segurança podem ser implementados facilmente pela arquitetura no plano de controle. Para isso, basta atualizar a aplicação de segurança no DTSA e o módulo do ESMP nas entidades comunicantes. Os NEs de núcleo e de borda da rede permanecem intactos, sem necessitar de nenhuma espécie de atualização, o que facilita a inclusão de novos serviços/mecanismos de segurança com extrema facilidade de implementação, e sem alterar em nada a complexidade da arquitetura.

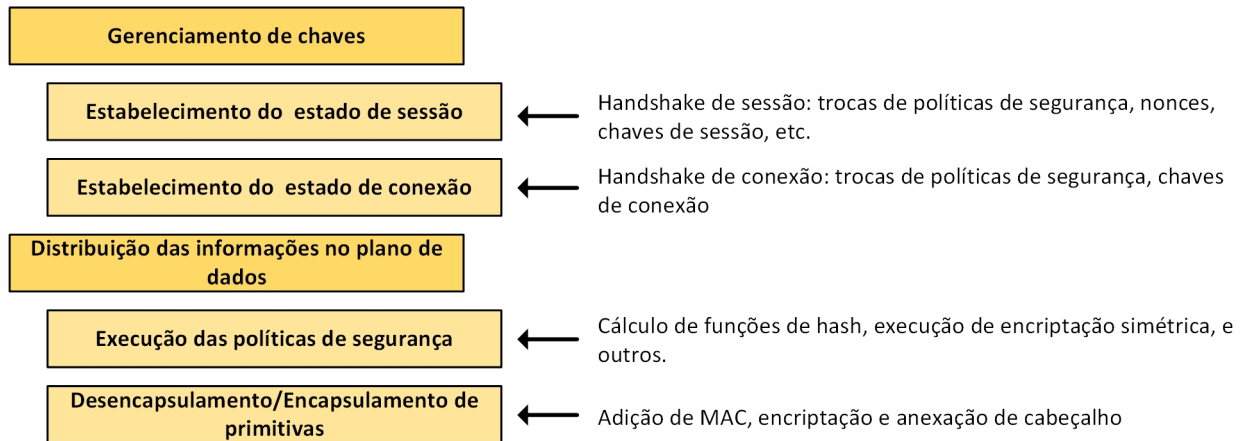
A proposta desse trabalho difere muito da proposta de segurança inicial da ETArch (seção 3.10). Primeiramente, é proposto um protocolo de segurança *multicast*, denominado *Entity Security multicast Protocol* (ESMP). Esse protocolo é totalmente transparente em relação às entidades comunicantes e às primitivas de controle da ETArch. Por esse motivo, não há modificações nos protocolos de controle da ETArch, como foi feito na proposta anterior (seção 3.10). ESMP também tem a capacidade de executar diferentes serviços de segurança para diferentes entidades comunicantes. Essa flexibilidade também não está disponível no status quo de segurança da ETArch.

Todas as operações do protocolo ESMP são detalhadas ainda nesse capítulo. No entanto, de forma geral, o estabelecimento de um ambiente de comunicação seguro no plano de dados/controle passa por um processo de gerenciamento de chaves, que engloba duas fases iniciais de negociações: estabelecimento de um estado de sessão através do que denominou-se *handshake* de sessão; estabelecimento de um estado de conexão através do que denominou-se *handshake* de conexão. Essas duas fases se realizam no plano de controle e são essenciais para que o ESMP consiga proteger as informações no plano de dados.

Dito isso, torna-se necessário explicar que a proteção no plano de dados se dá tanto para primitivas de controle quanto para primitivas de dados. Os serviços/mecanismos de

segurança são aplicados conforme política de segurança selecionada pela entidade comunicante. A proteção das informações passa por um encapsulamento do *payload* da camada superior (camada de aplicação) pelo ESMP no remetente. Quando essa primitiva chega no receptor, ela é desencapsulada por esse mesmo protocolo.

Figura 9 – Operações genéricas do ESMP



A Fig. 9 representa as operações genéricas do funcionamento do protocolo ESMP. O gerenciamento de chaves engloba as trocas de informações compartilhadas entre as entidades comunicantes. Exemplos dessas informações são *nonces*, chaves de sessão (chaves mestre), chaves de conexão (temporárias), etc. Essas trocas são realizadas pelo *handshake* de sessão e pelo *handshake* de conexão. Após essas negociações iniciais, as informações no plano de dados são protegidas através das políticas de segurança negociadas pelo plano de controle. Essa figura é genérica, porém, cada uma dessas operações são detalhadas em seções subsequentes.

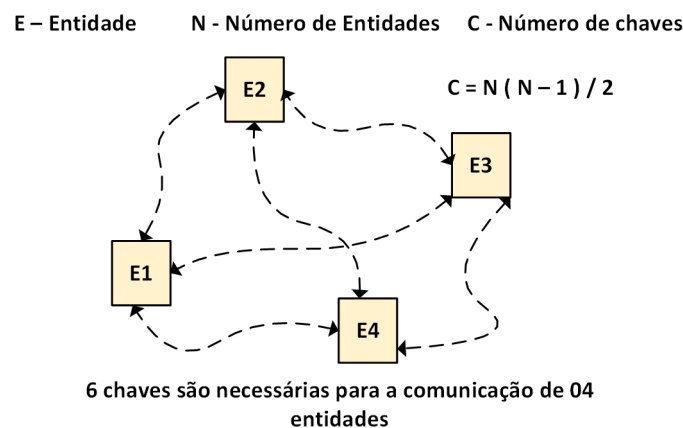
O restante deste capítulo segue assim: a seção 4.1 descreve como se dá a sintetização de chaves de uma comunicação segura multicast quando comparada às comunicações peer-to-peer (ALMs) que emulam multicasting na arquitetura legada; a seção 4.2 explana de forma genérica como se dá a distribuição de chaves (estabelecimento do estado de sessão e de conexão) no plano de controle e como se dá a transmissão das informações (primitiva de dados e de controle) após essas duas fases de negociação; a seção 4.3 explana de forma detalhada as operações e os conceitos do ESMP, quais sejam: atributos do estado de sessão/conexão (seção 4.3); modelagem das políticas de segurança (subseção 4.3.1); encapsulamento e desencapsulamento das primitivas (subseção 4.3.2); e as operações de *handshake* (subseção 4.3.3).

4.1 Sintetização de chaves

Uma conexão *peer-to-peer*, onde um conjunto de "N" entidades precisam estabelecer uma conexão segura, de tal forma que, cada entidade precisa se comunicar com todas

as outras entidades (Fig. 10), a escala de chaves distribuídas no nível de rede é de $N(N-1)/2$, pois cada par de entidades envolvida na comunicação precisa de pelo menos uma chave compartilhada. Uma das iniciativas desse trabalho é diminuir essa escala, e sintetizar esse número de disseminação de chaves em uma comunicação *multicast*. Os recursos da arquitetura ETArch acabam por tornar essa tarefa bastante simples, já que oferece conexão *multicast* intrínseca, e portanto, não utiliza o recurso de *unicast* múltiplo em seus terminais.

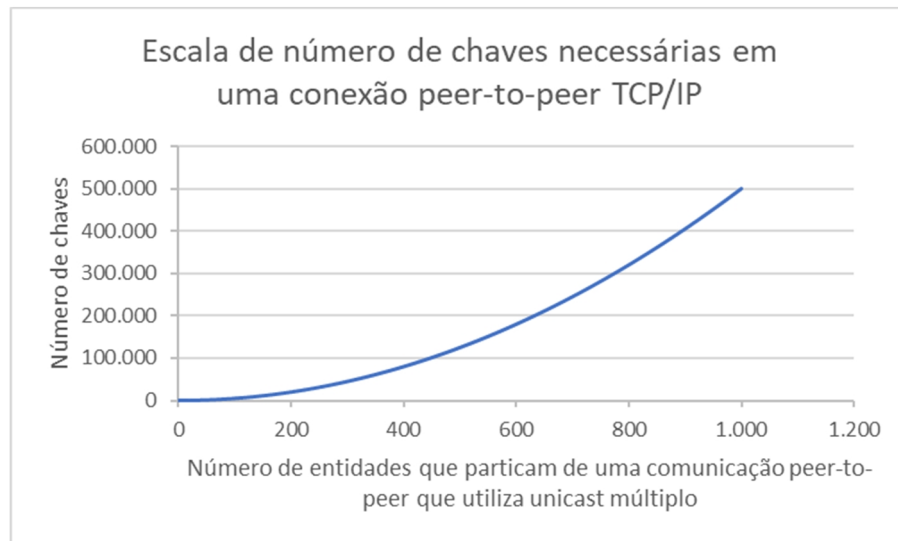
Figura 10 – Conexão peer to peer com utilização de unicast múltiplo



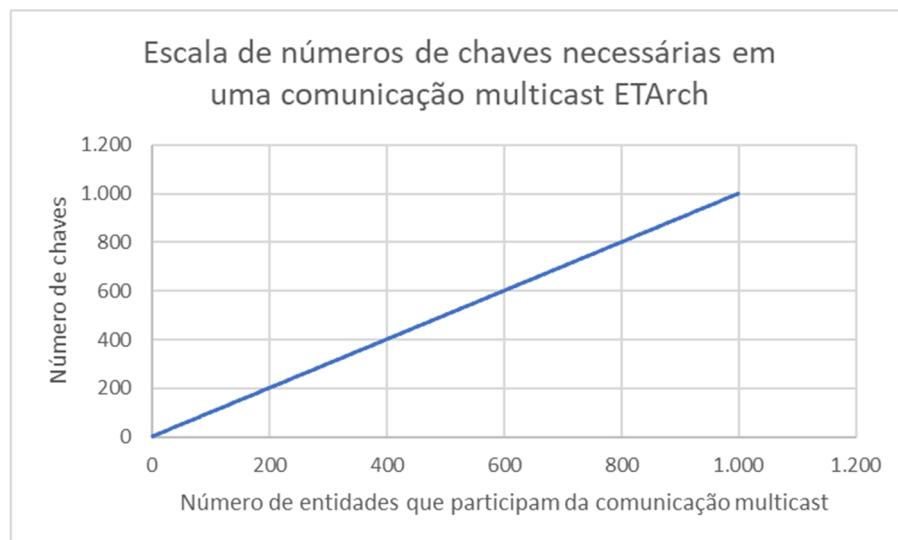
A escala calculada acima diz respeito ao número de chaves simétricas que têm que ser compartilhadas em conexões *peer-to-peer* no nível de rede da arquitetura TCP/IP, porém, se a comunicação estiver sendo protegida pela rede e pela aplicação (ALMs, ou aplicações *peer-to-peer*) simultaneamente, a escala de crescimento do número de chaves em relação às entidades comunicantes seria ainda mais insustentável quanto ao gerenciamento e manutenção dessa estrutura de distribuição. Supondo que uma rede de 1000 entidades admite 1000 aplicações (uma em cada entidade), essa proteção simultânea custaria aproximadamente um milhão (1.000.000) de chaves compartilhadas.

Para reduzir essa escala de chaves frente ao número de entidades comunicantes, utilizaremos um centro de distribuição de chaves. Esse papel será desempenhado pelo DTSA. Desse modo, cada entidade participante de um *workspace* de dados terá, necessariamente, uma chave exclusiva com esse DTSA. Para esse fim, haverá negociações iniciais entre DTSA e entidade a fim de criar um estado de sessão, onde um dos atributos de tal estado é uma chave de sessão que será denominada chave mestre. Essa chave será utilizada para quaisquer comunicações entre DTSA e entidade. Em outra fase de negociações, essa chave mestre será utilizada para negociar com o DTSA um estado de conexão, onde um dos atributos armazenados será uma chave de conexão (temporária), e esta por sua vez, será utilizada para trocar informações confidenciais no plano de dados. Nesse aspecto, o número da escala de chaves frente ao número de entidades em uma conexão *multicast* será $N+Q$ chaves, onde "N" é o número de entidades (cada entidade terá uma chave mestre para comunicação com o DTSA), e o "Q" representa o número de chaves adicionais

Figura 11 – Comparação da escala de chaves de uma conexão *multicast* ETArch e uma conexão *unicast* múltiplo do TCP/IP



(a) Escala de chaves necessárias para entidades que emulam multicast através de unicast múltiplo



(b) Escala de chaves necessárias para entidades que utilizam multicast sem replicação de primitivas na origem

de conexões lógicas pertencentes à entidade. Se a entidade possuir 1 *workspace*, contém 1 chave adicional, se possuir 2 *workspaces*, contém duas chaves adicionais, e assim por diante. O número de *workspaces* determina o número de conexões lógicas que a entidade oferece, e o número de conexões lógicas determina a quantidade de chaves temporárias. Essas chaves são utilizadas para as trocas de mensagens (no plano de dados) entre as entidades participantes de um determinado contexto de comunicação.

Desse modo, podemos analisar a escala de chaves da Fig.11(a) e Fig.11(b). A Fig.11(a) representa a escala de número de chaves necessárias em relação a uma conexão *peer-to-peer* feita na arquitetura legada, enquanto a Fig.11(b) representa essa mesma escala, com os mesmos números de entidades comunicantes, quando essa conexão é feita na arquitetura

ETArch. Pressupõe-se que nos dois casos, é oferecida segurança no nível de rede. No caso da ETArch, poderia ser uma aplicação (entidade) que oferece fluxo contínuo de vídeo com segurança no nível de rede. No caso da arquitetura TCP/IP, poderia se tratar de uma aplicação de correio eletrônico que dispara a mesma mensagem para 1000 usuários distintos.

A análise do gráfico indica que a aplicação de correio eletrônico da arquitetura TCP/IP necessita de quase meio milhão de chaves compartilhadas para distribuir a informação de forma segura para 1000 entidades distintas, enquanto a aplicação de distribuição de vídeo da ETArch necessita de um pouco mais de 1000 chaves para distribuir a informação para o mesmo número de receptores. Esse fato ocorre porque a distribuição intrinsecamente *multicast* não admite repetição de primitivas na origem, enquanto o *unicast* múltiplo precisa fazer conexões peer-to-peer para distribuir seu conteúdo.

4.2 Gerenciamento de chaves

A distribuição de chaves no ambiente ETArch é realizado pelo protocolo ESMP e leva em conta a política de segurança selecionada pela entidade comunicante, ou seja, o DTS pode executar diferentes maneiras de distribuição de chaves para diferentes entidades (aplicações, sensores, etc.) comunicantes. Essa seção explana de forma genérica como se dá a disseminação de chaves para que o restante da comunicação pós-distribuição possa ocorrer de forma segura. O detalhamento dessa operação é explicada à posteriori, quando analisamos as fases de funcionamento do protocolo proposto. Como a proposta é fornecer as chaves necessárias de forma dinâmica, essa técnica de distribuição inicial é extremamente importante e passa a ser a principal força desse sistema de segurança.

A escolha do serviço de distribuição de chaves pelas entidades (aplicações, sensores, etc.) depende dos impactos que essas entidades sofrem quando é realizada uma quebra de sigilo na comunicação das informações que transmite. Algumas dessas quebras podem provocar pequenas perdas financeiras, outras podem provocar grandes perdas financeiras, e por fim, algumas quebras podem ter consequências que signifiquem perda de vida ou lesões graves. Um exemplo é a aplicação bancária, que possui uma forma de distribuição diferente de uma aplicação de compra eletrônica online. No primeiro exemplo, há o risco de grande perda financeira e a troca de chaves é feita fisicamente, antes mesmo que a primeira transação financeira seja executada. No segundo exemplo, existe um risco maior de ataque, porque as trocas de chaves compartilhadas são feitas dinamicamente, sendo que as primeiras negociações do protocolo (TLS) são realizadas por primitivas que viajam pela rede sem nenhum tipo de cifragem. Esse fato acarreta uma vulnerabilidade que pode ser explorada por algum possível atacante.

Dito isso, algumas maneiras de distribuição tendem a ser mais seguras que outras. Geralmente, distribuições onde já existem chaves compartilhadas entre as partes devido

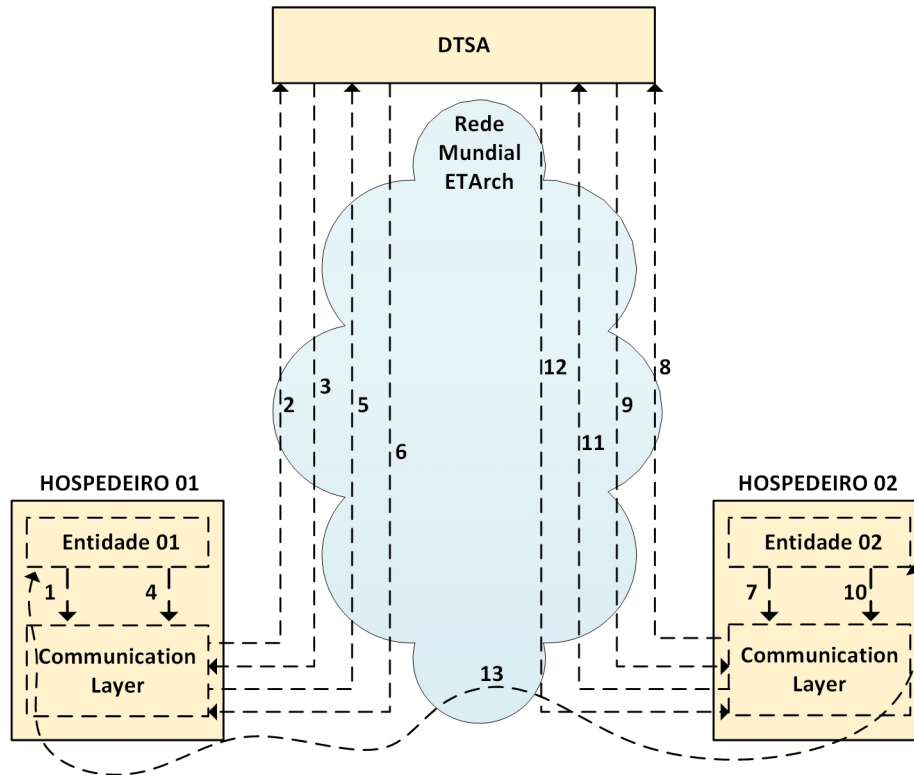
à entrega física ou manual realizada antes mesmo da primeira comunicação tendem a ser mais seguras, pois as primitivas iniciais dessa comunicação já estarão protegidas pelos mecanismos/serviços de segurança. Por outro lado, quando esta troca inicial de chaves ocorre de maneira totalmente dinâmica, as primeiras primitivas de negociação estarão desprotegidas de serviços de segurança tais como autenticação e confidencialidade. Nessa fase inicial podem haver vulnerabilidades a serem exploradas. A proposta é que o ESMP consiga fornecer várias maneiras de distribuição de acordo com as requisitos das entidades.

Essas várias formas de implementações de disseminação de chaves passam pela escolha da política de segurança que melhor se encaixa nos requisitos da entidade. Essa política de segurança será detalhada em seções posteriores. O importante é que quanto maior o nível de segurança dessa distribuição, maior será a pré-condição para que a entidade participe de determinado método de distribuição. Há três tipos básicos de distribuição de chaves, quais sejam (STALLINGS, 2014): (i) distribuição de chave simétrica usando encriptação simétrica; (ii) distribuição de chave simétrica usando encriptação assimétrica; (iii) um método híbrido de distribuição.

O pré-requisito para a utilização de (i) e (ii) é que já existam chaves compartilhadas antes mesmo da primeira comunicação. Por esse motivo, esses dois primeiros métodos tendem a ser mais seguros que o terceiro. No primeiro caso, o DTSA e a entidade já compartilham uma chave simétrica mestre (de sessão) antes do início da etapa de *handshake* do protocolo ESMP. No segundo caso, tanto DTSA quanto entidade têm que possuir um par de chaves pública/privada e certificados válidos; caso a política de segurança opte por utilizar certificados para a distribuição de chaves públicas entre os participantes da comunicação. No caso três, para os propósitos desse trabalho, o pré-requisito é que apenas o DTSA possua um cadastro em alguma autoridade certificadora; dessa forma, ele (DTSA) têm que possuir um certificado válido e um par de chaves pública/privada.

O protocolo ESMP prevê essas três formas de distribuição, mas esse trabalho manterá o foco na técnica híbrida. Esse método faz o uso de um centro de distribuição de chaves (CDC), que é materializado no DTSA. O DTSA compartilha chaves de sessão secretas no momento do registro das entidades. Nesse contexto, a entidade registrada compartilha uma chave de sessão (chave mestre) que tem a função principal de realizar comunicação segura entre a entidade e o DTSA. Entende-se por comunicação segura a utilização de serviços/mecanismos de segurança nas primitivas que serão trocadas após o compartilhamento da chave mestre. Em um outro momento, mais precisamente, quando a entidade produtora de serviços solicita a criação de um *workspace*, algumas outras chaves (chaves de conexão) são trocadas entre entidade e DTSA. Essas chaves de conexão são temporárias e seu tempo de ciclo de vida é o mesmo do *workspace*. Chaves de funções de *hash* e de encriptação são trocadas nesse momento. Essas chaves de conexão são vinculadas ao *workspace* e sua principal função é a aplicação de serviços/mecanismos de segurança no plano de dados da arquitetura.

Figura 12 – Distribuição de chaves dinâmica da arquitetura ETArch



Adaptada de (STALLINGS, 2014).

Esse método híbrido, de forma geral, utiliza encriptação assimétrica para distribuição de chaves simétricas no momento do registro das entidades. Porém, depois dessa primeira troca, todas as outras primitivas serão orientadas pelas chaves simétricas. Esse fato se deve à falta de desempenho de algoritmos de encriptação de chave pública (STALLINGS, 2014). Com essa hierarquia de três níveis de chaves (chave assimétrica, chave simétrica de sessão, chave simétrica de conexão), a encriptação de chave pública será utilizada apenas ocasionalmente.

Como dito, essas trocas de chaves são melhor explicadas nas seções seguintes, mais precisamente, quando é explanado a fase de *handshake* do protocolo ESMP (subseção 4.3.3). A Fig. 12 apresenta genericamente as operações que proporcionam as trocas de chaves dinâmicas da arquitetura ETArch.

1. Entidade 1 (aplicação 1) solicita o seu registro no DTSA.
2. O protocolo ESMP encapsula a primitiva `ENTITY_REGISTER` e a envia para o DTSA. A operação de *handshake* de sessão (detalhada na subseção 4.3.3.1) desmembra a operação 2 em várias etapas. Em uma dessas etapas, a Entidade 01 gera a chave de sessão e envia para o DTSA.
3. O DTSA passa as informações necessárias para o estabelecimento do estado de sessão da Entidade 1 (aplicação 1).

4. Entidade 1 (aplicação 1) solicita a criação de um *workspace* (de título w1) para o DTSA.
5. O protocolo ESMP encapsula a primitiva WORKSPACE_CREATE e a envia para o DTSA.
6. O DTSA passa as informações necessárias para o estabelecimento do estado de conexão do *workspace criado* (de título w1). Uma dessas informações é a chave de conexão. Nesse momento, a Entidade 01 e o DTSA estão criando o estado de conexão do *workspace* w1.
7. Entidade 2 (aplicação 2) solicita o seu registro no DTSA.
8. O protocolo ESMP encapsula a primitiva ENTITY_REGISTER e a envia para o DTSA. A operação de *handshake* de sessão (detalhada na subseção 4.3.3.1) desmembra a operação 2 em várias etapas. Em uma dessas etapas, a Entidade 02 gera a chave de sessão e envia para o DTSA.
9. O DTSA passa as informações necessárias para o estabelecimento do estado de sessão da Entidade 2 (aplicação 2).
10. Entidade 2 (aplicação 2) solicita o seu "*attach*" no *workspace* "w1".
11. O protocolo ESMP encapsula a primitiva WORKSPACE_ATTACH e a envia para o DTSA.
12. O DTSA passa as informações necessárias para o estabelecimento do estado de conexão do *workspace* (de título w1). Uma dessa informações é a chave de conexão. Nesse momento, o DTSA está distribuindo o estado de conexão do *workspace* w1 para a Entidade 02.
13. A comunicação entre a Entidade 1 (aplicação 1) e Entidade 2 (aplicação 2) ocorre no plano de dados mediante serviços/mecanismos de segurança requisitados pela Entidade 1 através da escolha de uma política de segurança. Por exemplo, a Entidade 1 pode ter requisitado confidencialidade, e nesse caso, o *payload* das primitivas serão encriptados.

Nos tópicos acima, não estão detalhadas as trocas de primitivas necessárias entre DTSA e entidades; e, não estão detalhadas a geração de eventos e trocas de primitivas que ocorrem entre as camadas do DTSA (ver Fig. 8). Todavia, a primitiva WORKSPACE_ATTACH está detalhada na Tab. 4; as outras primitivas de controle possuem funcionamento similar. Os detalhes dos protocolos (ETCP e DTSCP) de controle da ETArch se encontram em (SILVA, 2013).

O importante é salientar que todos os pacotes de controle encapsulados pelo ESMP na entidade, quando chegam no DTSA, são primeiramente desencapsulados pelo SBB SM. Depois que as trocas de chaves são realizadas, o protocolo de controle (por exemplo, ENTITY_ATTACH) segue seu itinerário normal, como descrito na Tab. 4. Caso algum erro se apresente na fase de execução do ESMP, uma mensagem de erro será devolvida para a entidade. Essas trocas de mensagens são expostas com mais detalhes quando explanamos o funcionamento do protocolo proposto. A função dessa seção é mostrar o gerenciamento de chaves de forma genérica, já que todo o processo será descrito pela operação de *handshake* do protocolo ESMP.

Em suma, o ESMP (no DTSA) intercepta as mensagens de controle que foram encapsuladas (na entidade) por seus cabeçalhos. Faz as operações de segurança necessárias, seguindo como diretriz a política de segurança escolhida pela entidade. Logo após o término das suas operações, a primitiva de controle encapsulada pelo ESMP segue seu fluxo normal de execução. A única diferença é que essas primitivas sempre estarão encapsuladas pelo protocolo ESMP no caminho entre entidade e DTSA.

4.3 Entity Security Multicast Protocol - ESMP

O ESMP é um protocolo de segurança *multicast* que oferece um conjunto de mecanismos/serviços que transformam o ambiente de comunicação em uma rede de confiança. Entende-se por rede de confiança um ambiente de comunicação onde as entidades comunicantes possam confiar uma nas outras. Para esse fim, o protocolo tem que possuir a capacidade de oferecer serviços de gerenciamento de chaves, autenticação, confidencialidade, integridade, disponibilidade, e outros. Esses serviços/mecanismos de segurança oferecidos são vinculados às necessidades reais das entidades comunicantes (sensores, aplicações, etc.). Nota-se que a capacidade de oferecer segurança *multicast* de acordo com cada entidade é um recurso bastante promissor. Soma-se a isso o conceito de entidade da arquitetura, que é "qualquer coisa" que tem capacidade de comunicação.

Desse ponto de vista, nota-se que o protocolo possui a capacidade de diversificar suas operações de segurança conforme política de segurança selecionada pela entidade. Diferentes operações de segurança pedem diferentes políticas de segurança. Uma aplicação bancária pode necessitar operar com distribuição de chaves simétricas usando encriptação assimétrica. Todavia, a comunicação de um sensor utilizará apenas recursos de distribuição de chaves simétricas usando encriptação simétrica, já que esse tipo de operação tende a consumir menos hardware do dispositivo.

Outra preocupação do ESMP é ser totalmente transparente para as entidades comunicantes. Por essa razão, esse protocolo pertence à camada "*Communication Layer*" da Fig. 4. A ideia é que o protocolo no nível de rede consiga resolver às demandas de segurança das aplicações. Para cumprir os requisitos de segurança, o protocolo encapsula as

primitivas de controle na entidade e desencapsula no DTSA, e vice-versa. Com relação ao plano de dados (trocas de primitivas de dados), essa operação ocorre entre as entidades participantes de um contexto de comunicação (*workspace*).

Alguns conceitos se tornam importantes para o conhecimento da especificação desse protocolo, quais sejam: estado de sessão; e, estado de conexão.

Um estado de sessão do protocolo ESMP é uma associação entre uma entidade e seu DTSA. Um conjunto de parâmetros de segurança são negociados na fase do primeiro *handshake* realizado pelo protocolo, mais precisamente, quando a entidade solicita o seu registro no DTSA. Se nessa solicitação, o parâmetro de política de segurança for acionado na primitiva ENTITY_REGISTER (campo *requirements*); o ESMP negociará os parâmetros através desse *handshake*. Um estado de sessão é definido pelos seguintes parâmetros:

1. identificador da sessão: identificador gerado pelo DTSA para identificar um estado de sessão ativo. Um estado de sessão é sempre vinculado à uma entidade;
2. chave mestre: chave compartilhada entre entidade e DTSA;
3. *nonce* do servidor: sequência aleatória de mensagens do servidor. Essa sequência é conferida apenas para mensagens ESMP do servidor que acontecem entre entidade e DTSA;
4. *nonce* da entidade: Sequência aleatória de mensagens da entidade. Essa sequência é conferida apenas para mensagens ESMP da entidade que acontecem entre entidade e DTSA;
5. versão: Trata-se da versão do protocolo ESMP que está sendo utilizado pela arquitetura;
6. vetor de inicialização: parâmetro de inicialização para protocolos de encriptação "*Cipher Block Chaining*" (CBC);
7. título da entidade: identificador da entidade;
8. certificado do DTSA: a sessão armazena a chave pública do DTSA. Essa informação será crucial para realizar a troca de chave mestre (primeira troca de chave) da entidade com o DTSA;
9. política de segurança: identificador da política de segurança da sessão.

Um estado de conexão ESMP é uma associação entre as entidades participantes de um *workspace*. A configuração dos parâmetros desse estado determina como é a segurança aplicada no plano de dados. Nesta especificação do ESMP, uma entidade possui apenas uma sessão, mas pode possuir "N" conexões. A conexão é criada no momento que a entidade requisita a criação de um *workspace* através da primitiva WORKSPACE_CREATE.

Se no campo "*capabilities*" dessa primitiva, o parâmetro de política de segurança estiver acionado; o ESMP negociará os parâmetros do estado de conexão na execução do segundo *handshake*. Um estado de conexão é definido pelos seguintes parâmetros:

1. identificador da conexão: identificador gerado pelo DTSA para identificar um estado de conexão ativo. Um estado de conexão é sempre vinculado à um *workspace*;
2. identificador da sessão: identificador que indica a sessão vinculada à essa conexão;
3. *nonce* da entidade: sequência aleatória de mensagens da entidade. Essa sequência está relacionada com as primitivas trocadas entre as entidades dessa conexão; porém, esse *nonce* pertence à sequência de primitivas enviadas pela entidade remetente;
4. chave de conexão: chave compartilhada entre as entidades dessa conexão;
5. título do *workspace*: identificador do *workspace*;
6. política de segurança: identificador da política de segurança da conexão.

As operações de encapsulamento/dencapsulamento, o formato das primitivas do protocolo, a modelagem da política de segurança e as operações de *handshake* são os temas das próximas sessões. O detalhamento dessas operações é essencial para o entendimento do funcionamento do ESMP.

4.3.1 Modelagem das políticas de segurança

Política de segurança, como descrito em 2.1.5, é um conjunto de contramedidas que atenuam às ameaças e ataques em um sistema. No caso da ETArch, a política de segurança tem o objetivo de governar serviços/mecanismos que possuem como função a proteção sistemática de comunicação no plano de controle e de dados.

Uma das características que tornam políticas de segurança atraentes para qualquer arquitetura é o fato da flexibilidade que se ganha nas operações de segurança. Duas entidades (aplicações) distintas podem consumir mecanismos/serviços de segurança distintos, com níveis de segurança diferentes. Um exemplo é quanto ao gerenciamento de chaves. Essa operação pode ser feita de várias maneiras diferentes, e o nível de segurança é proporcional aos pré-requisitos dessa operação.

Uma troca de chaves, cujos pré-requisitos é distribuir (fisicamente) pelo menos uma chave compartilhada entre as partes comunicantes antes do primeiro contato tende a possuir um nível de segurança maior do que uma troca que não possui esse pré-requisito inicial (trocas totalmente dinâmicas). Nesse aspecto, um aplicativo de transações bancárias pode possuir uma política de segurança diferente de um aplicativo que faz compras online. Essa política de segurança pode se diferenciar no atributo "gerenciamento de chaves". A primeira aplicação, geralmente, possui uma chave compartilhada antes mesmo da

primeira transação bancária. A segunda, geralmente, utiliza o protocolo TLS, que não possui esse pré-requisito, e portanto, possui um nível de segurança menor. A política de segurança é o mecanismo que dá à arquitetura essa flexibilidade.

Na prática, trata-se de um repositório que pode ser implementado como um banco de dados relacional, em memória, ou em bancos *NoSQL*. O fato é que a entidade que requisita um serviço de segurança, seleciona o identificador que representa uma política de segurança. Há uma política de segurança que cobre a comunicação da entidade com o DTSA, e há uma política de segurança que é responsável pela comunicação entre as entidades do *workspace*. A primeira pode ser denominada política de sessão, enquanto a segunda, política de conexão.

Essas políticas serão melhor detalhadas no decorrer desse capítulo. A função dessa seção é verificar quais são os atributos que constituem uma política de segurança. Abaixo, há uma descrição detalhada para cada um desses atributos. Alguns não possuem explicação devido à obviedade do item.

1. Gerenciamento de distribuição de chaves

1. Distribuição de chave simétrica utilizando chave simétrica. A pré-condição é que as entidades comunicantes já tenham trocado chaves simétricas compartilhadas antes mesmo da primeira comunicação. É um mecanismo altamente confiável, pois as primeiras trocas de mensagens (de controle ou de dados) entre as partes envolvidas já estão utilizando serviços/mecanismos de segurança, tais como integridade, autenticação, entre outros.
2. Distribuição de chave simétrica utilizando chave assimétrica. A pré-condição é que as entidades comunicantes já possuam pares de chaves pública/privada e certificado válidos. É um mecanismo altamente confiável, pois as primeiras trocas de mensagens (de controle ou de dados) entre as partes envolvidas já estão utilizando serviços/mecanismos de segurança, tais como integridade, autenticação, entre outros.
3. Distribuição híbrida. A pré-condição é que o CDC (DTSA) tenha um par de chaves pública/privada e um certificado válido. As primeiras negociações do protocolo de segurança correspondente são feitas através de primitivas que não possuem em seu formato nenhuma espécie de serviço de segurança aplicado. Desse modo, essas primeiras primitivas apresentam vulnerabilidade ou ameaças que podem ser explorados por um atacante. No caso da ETArch, existem alguns pré-requisitos que dificultam esse ataque, quais sejam: o atacante tem que possuir um certificado válido emitido através de uma autoridade certificadora que é aceita pelo protocolo ESMP. O TLS, que é um dos protocolos de segurança mais utilizados na internet, possui esquema de distribuição semelhante, e portanto, enfrenta esse mesmo problema.

2. Distribuição de chave pública

1. Autoridade de chave pública. A pré-condição é que o DTSA conheça todas as chaves públicas das entidades do seu domínio, e cada entidade, tem que conhecer a chave pública do seu DTSA. Nesse contexto, o DTSA se assemelha a um diretório de chaves públicas e tem a função de distribuí-las para as entidades conforme suas necessidades.
2. Certificado. Um certificado é um arquivo que consiste basicamente de uma chave pública, o identificador/nome do proprietário da chave e uma assinatura digital (do *hash* do conteúdo do certificado) de um terceiro (autoridade certificadora). A priori, este certificado pode ser obtido através de uma estrutura hierárquica de diretórios de chaves públicas ou pode ser enviado pela entidade comunicante (DTSA, nesse caso específico da ETArch).

3. Autoridade certificadora

1. Nenhum. Não utiliza certificados para distribuição de chave pública.
2. CEF. O certificado está assinado pela CEF.
3. Banco do Brasil. O certificado está assinado pelo Banco do Brasil.
4. Outras. Outras autoridades certificadoras reconhecidas pelo protocolo.

4. Padrão do certificado

1. Nenhum. Não utiliza certificados para distribuição de chave pública
2. X-509 ITU-T. Formato do certificado utilizado (ITU-T, 2001).

5. Função de hash

1. MD5. Referência pode ser encontrada em (STALLINGS, 2014).
2. SHA-1. Referência pode ser encontrada em (STALLINGS, 2014).
3. SHA-3. Referência pode ser encontrada em (STALLINGS, 2014).

6. Algoritmo de encriptação simétrico

1. Triple DES com duas chaves. Referência pode ser encontrada em (STALLINGS, 2014).
2. Triple DES com três chaves. Referência pode ser encontrada em (STALLINGS, 2014).
3. AES. Referência pode ser encontrada em (STALLINGS, 2014).

7. Algoritmo assimétrico

1. Nenhum. O item 1 do "Gerenciamento de distribuição de chaves" da modelagem de políticas de segurança não utiliza esse recurso.
2. RSA. Referência pode ser encontrada em (STALLINGS, 2014).
3. Diffie-Hellman. Referência pode ser encontrada em (STALLINGS, 2014).
4. Curva Elíptica. Referência pode ser encontrada em (STALLINGS, 2014).

8. Algoritmo de MAC¹

1. HMAC. Referência pode ser encontrada em (STALLINGS, 2014)

9. Gerador de números aleatórios

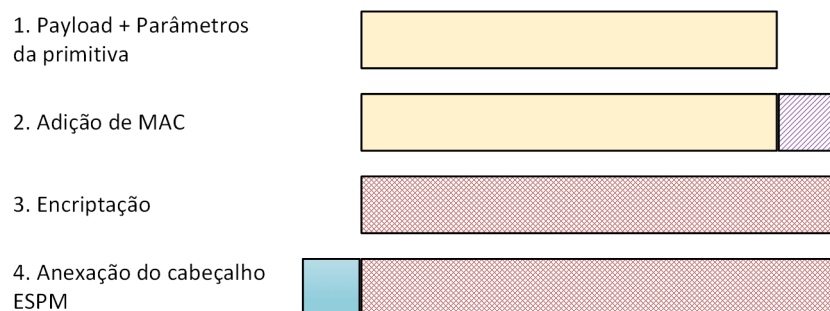
1. *True Random Number Generator* (TRNG). Referência pode ser encontrada em (STALLINGS, 2014).
2. *Pseudorandom Number Generator* (PRNG). Referência pode ser encontrada em (STALLINGS, 2014).
3. *Pseudorandom function* (PRF). Referência pode ser encontrada em (STALLINGS, 2014).

A distribuição híbrida, item 03 do atributo "Gerenciamento de distribuição de chaves", será o foco desse trabalho no que tange a explanação das operações de segurança realizadas pelo protocolo ESMP.

4.3.2 Encapsulamento e Desencapsulamento

A Fig. 13 apresenta a operação geral de encapsulamento do ESMP. Inicialmente, a entidade comunicante (aplicação, e outros) indica na sua lista de requisitos o identificador da política de segurança adequada e envia uma mensagem a ser transmitida. Essa mensagem, representada pelo primeiro item da figura, pode conter dados a serem transmitidos, como também pode se tratar de uma primitiva de controle da ETArch.

Figura 13 – Operação de encapsulamento do protocolo ESMP



No caso de ser uma primitiva de controle, duas situações são possíveis: (i) a primitiva de controle é enviada antes que as trocas de chaves compartilhadas sejam realizadas pela

operação de *handshake*; (ii) a primitiva de controle é enviada depois que a operação inicial de *handshake* já foi realizada.

No primeiro caso, os passos a serem executados referem-se aos itens 1 e 4 da Fig. 13. Como as chaves ainda não foram trocadas, as primeiras mensagens de negociação estão desprotegidas de serviços de segurança. Porém, é importante que o cabeçalho do ESMP seja anexado a mensagem. Desta forma, o DTSA reconhece a primitiva como sendo de segurança e a envia ao módulo SBB SM para devido tratamento. Essas primeiras primitivas, certamente, tratam-se das primitivas que realizam a operação de *handshake* de sessão (explicada posteriormente, ainda nessa seção).

No segundo caso, os passos a serem executados referem-se aos itens 1, 2, 3 e 4 da Fig. 13. Entende-se que as primitivas de controle necessitam de serviços de autenticação, integridade e confidencialidade.

Há uma terceira situação, onde as primitivas não fazem parte do plano de controle. Nesse caso, a comunicação ocorre no plano de dados, sem a participação do DTSA. Essas primitivas de dados estão protegidas por serviços/mecanismos de segurança segundo política de segurança selecionada pela entidade proprietária do *workspace*. Os passos referentes aos itens 1 à 4 da Fig. 13 são executados antes da transmissão das informações.

A tarefa de encapsular/desencapsular as primitivas em uma entidade (que não seja o DTSA) pertence à camada "*Communication Layer*" (ver Fig. 12), presente em todas as entidades comunicantes da arquitetura ETArch. No DTSA, essa tarefa é executada pelo SBB SM. Na prática, o NEConnector, ao perceber que a primitiva pertence ao protocolo ESMP, ele (NEConnector) a envia ao SBB SM. Esse bloco de serviço desencapsula a primitiva de controle ETArch, executa todas as operações necessárias para criar um ambiente de segurança de comunicação entre DTSA e entidade, e logo após, quando os procedimentos de segurança já houverem sido executados; a primitiva de controle da ETArch desencapsulada pelo ESMP segue seu fluxo normal de execução.

Em suma, o protocolo recebe uma mensagem a ser transmitida (da entidade/aplicação), aplica uma função de MAC¹ (*Hashed Message Authentication Code* - HMAC) (BELLARE; CANETTI; KRAWCZYK, 1996a) (BELLARE; CANETTI; KRAWCZYK, 1996b), encripta, acrescenta um cabeçalho e transmite a primitiva resultante através de um *workspace*. Os dados recebidos são desencapsulados (retirada do cabeçalho do protocolo ESMP), decryptados e verificados (através do código *hash*). Caso as verificações sejam aceitáveis, as informações do *payload* da primitiva são passadas para a camada superior. No caso de um ENTITY_REGISTER (primitiva enviada para o DTSA), elas são repassadas para o SBB EM (ver Fig. 8). No caso de primitivas de dados (primitivas enviadas para as outras entidades de um *workspace*), elas são repassadas para a entidade receptora (aplicação receptora). Na Fig. 12 (item 13), há a representação dessa troca de informações no plano de dados.

É importante salientar que a função de *hash* utiliza alguns parâmetros que são es-

Tabela 6 – Tipo de mensagens do protocolo ESMP

Tipo de mensagem	Parâmetros
SESSION_ENTITY_HELLO (ESMP_SEH)	versão do ESMP, política de segurança
SESSION_DTSA_HELLO (ESMP_SDH)	versão do ESMP, identificador da sessão, certificado
SESSION_ENTITY_INFORMATION_EXCHANGE (ESMP_SEIE)	nonce da entidade, chave mestre(de sessão), vetor de inicialização da entidade
SESSION_DTSA_INFORMATION_EXCHANGE (ESMP_SDIE)	nonce do servidor, <i>nonce</i> da entidade, chave mestre(de sessão), vetor de inicialização do servidor
SESSION_ENTITY_FINISHED (ESMP_SEF)	hash, status(do hash)
SESSION_DTSA_FINISHED (ESMP_SDE)	hash, status(do hash)
CONNECTION_ENTITY_INFORMATION_EXCHANGE (ESMP_CEIE)	identificador da sessão, título do <i>workspace</i>
CONNECTION_DTSA_INFORMATION_EXCHANGE (ESMP_CDIE)	identificador da conexão, chave de conexão(temporária), identificador da sessão, política de segurança
ESMP	status do ESMP

senciais para fazer a autenticação mútua das partes. Dentre eles, podemos citar: chaves secretas compartilhadas, número de sequência das mensagens, o título da entidade remetente, etc. Essa função *hash*, cujos parâmetros são constituídos essencialmente por informações compartilhadas entre as partes e informações que são exclusivas do emissor, tem a função de realizar a autenticação da comunicação de controle, verificar a integridade de primitivas, não permitir que primitivas replicadas sejam processadas pelas entidades receptoras, etc.

Figura 14 – Formato da primitiva de segurança do ESMP

8 bits	8 bits	48 bits	16 bits	> 0 bits
Tipo de mensagem	Tipo de conteúdo	Título de entidade de origem	Comprimento do payload	Conteúdo Parâmetros – Payload

A última etapa do processamento do protocolo ESMP (Fig. 13) é anexar um cabeçalho no início da primitiva resultante. O formato desse cabeçalho pode ser visto na Fig. 14. A especificação dos seus campos são detalhadas abaixo.

- Tipo de mensagem (8 bits): são os tipos de mensagens do protocolo. Cada mensagem preenche o campo de parâmetros de uma forma diferente. Isso acontece porque

a lista de parâmetros passados na primitiva depende do tipo de mensagem que está sendo enviado. Os parâmetros de cada mensagem podem ser vistos na Tab. 6.

- ❑ Tipo de conteúdo(8 bits): identificador do tipo de conteúdo encapsulado, ou seja, esse identificador representa o tipo de primitiva da ETArch que será encapsulada pelo ESMP. Um exemplo é o identificador "-1", que representa uma primitiva de dados da ETArch. Outro exemplo é o identificador "0", que identifica a primitiva ENTITY_REGISTER. Os identificadores das primitivas de controle da ETArch podem ser vistos em (SILVA, 2013).
- ❑ Entidade de origem(48 bits): título da entidade que está enviando a primitiva ESMP.
- ❑ Comprimento do *payload*(16 bits): representa o tamanho do *payload* da primitiva. Entende-se por *payload* as informações transmitidas pela entidade (aplicação). Esse comprimento é calculado em relação ao texto aberto (sem cifras).
- ❑ conteúdo(>0 bits): esse campo é dividido em duas partes. A primeira representa os parâmetros do tipo de mensagem escolhido. Esses parâmetros depende do tipo de mensagem da primitiva e podem ser vistos na Tab. 6. A segunda parte representa o *payload*, que é caracterizado por uma primitiva de controle ou uma primitiva de dados da arquitetura ETArch.

4.3.3 Operação de *handshake*

O protocolo ESMP, através do plano de controle, realiza todas as operações necessárias para que a comunicação no plano de dados aconteça conforme requisitos de segurança da entidade (aplicação). Para esse fim, as comunicações iniciais de controle também devem ser feitas de forma segura. A operação inicial que realiza a autenticação mútua entre entidade e DTSA e negocia os parâmetros necessários de sessão/conexão é denominada *handshake*. Antes mesmo que quaisquer dados sejam transmitidos, esse protocolo prepara o ambiente de comunicação para proteger os dados conforme política de segurança selecionada pela aplicação.

Na ETArch, dois *handshakes* são necessários e acontecem em momentos distintos. O primeiro, podemos denominá-lo "*handshake* de sessão". Dentre suas funções, podemos citar o estabelecimento de um estado de sessão entre entidade e DTSA e a realização de autenticação mútua entre essas duas entidades. O segundo, podemos denominá-lo "*handshake* de conexão". Dentre suas funções, podemos citar o estabelecimento de um estado de conexão entre entidade e DTSA. O DTSA utilizará as informações de conexão para distribuí-las às entidades que venham a participar de um *workspace* já criado. Dessa forma, essas novas entidades conseguirão ler as informações do plano de dados, pois um dos parâmetros de um estado de conexão é a chave secreta (temporária) que será utili-

zada para a decifração dessas primitivas. Essas duas operações estão em pauta e serão detalhadas nessa seção.

Os passos do "*handshake* de sessão" e do "*handshake* de conexão" que serão vistos nas próximas seções dizem respeito ao gerenciamento de distribuição híbrida de chaves citado na seção 4.3.1. A decisão por esse atributo diz respeito à utilização do protocolo TLS na Internet. Por certo, é um dos protocolos de segurança mais utilizados da arquitetura TCP/IP, e utiliza o mecanismo de gerenciamento de distribuição híbrida de chaves nas suas operações iniciais. Apresentar todas as configurações de políticas de segurança que possam ser objeto de requisição de uma entidade é inviável, logo, fica dito a justificativa da escolha de gerenciamento de chaves que será adotado nesse trabalho.

4.3.3.1 Handshake de sessão

O primeiro contato que a entidade que solicita segurança tem com o DTSA é determinado pela operação de "*handshake* de sessão". As trocas de mensagens tem objetivos pré-determinados: configuração de um estado de sessão entre entidade e DTSA; troca de chaves de sessão (chave mestre compartilhada entre DTSA e entidade); troca de *nonces*; negociação da versão do protocolo ESMP a ser utilizado; e outros.

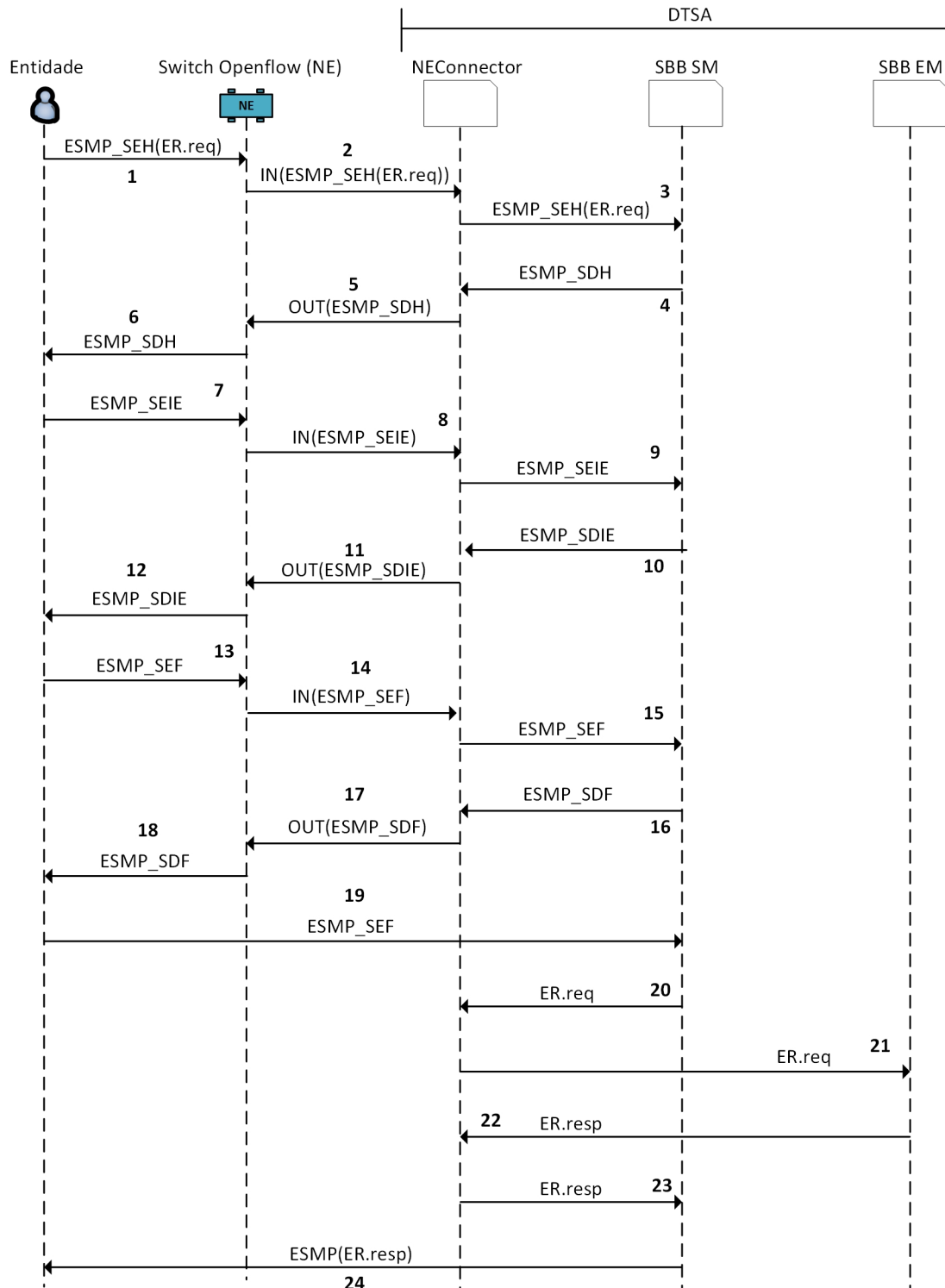
A operação desse *handshake* acontece quando a entidade solicita seu registro no DTSA (ENTITY_REGISTER) e aciona a política de segurança na sua lista de requisitos.

As trocas de primitivas necessárias com o DTSA para estabelecer uma comunicação segura conforme requisitos de segurança da aplicação passa pelas operações de *handshake* apresentadas na Fig. 15. Tanto os parâmetros quanto às siglas de cada primitiva apresentada se encontram na Tab. 6. Para fins de sintetização, esse diagrama leva em conta cenários de sucesso. Parte-se do pressuposto que todas as trocas de mensagens exibidas são realizadas com êxito. Cada passo dessa operação está descrito detalhadamente abaixo:

1. primitiva SESSION_ENTITY_HELLO (ESMP_SEH) encapsula a requisição ENTITY_REGISTER (ER.req) da entidade e é enviada ao NE. O protocolo ESMP foi ativado, pois a entidade (aplicação) setou a política de segurança na sua lista de requisitos da primitiva ER. Nesse primeiro momento, só as etapas 1 e 4 da operação de encapsulamento do ESMP (Fig. 13) são executadas. As etapas 2 e 3 necessitam de uma chave de sessão compartilhada. Essa primitiva inicial está desprotegida, porém, carrega em seus parâmetros informações básicas: versão do protocolo e identificador da política de segurança selecionada pela aplicação;
2. NE encapsula a primitiva do "item 1" em um OFPT_PACKET_IN e envia-a ao DTSA. A primitiva chega em NEConnector;
3. NEConnector desencapsula a primitiva ESMP_SEH(ER.req). Nesse momento, ao invés do NEConnector enviar o ER para o SBB EM (bloco de serviço responsável

Figura 15 – Diagrama de sequência do *handshake* de sessão

OUT – OFPT_PACKET_IN OUT – OFPT_PACKET_OUT ER – ENTITY_REGISTER ESMP – Primitivas de Segurança



pelo registro de entidades); ele (NEConnector) percebe que a primitiva que chega é uma das mensagens do protocolo ESMP. Nesse ponto, há uma interceptação do módulo de segurança no funcionamento normal do ER. Primeiro faz-se os mecanismos de segurança, depois a primitiva ER pode seguir seu itinerário de funcionamento

normal. Nessa perspectiva, o NEConnector envia a primitiva ESMP_SEH(ER.req) para o SBB SM. SBB SM desencapsula a primitiva ER.req e armazena-a (temporariamente). A desencapsulação pressupõe apenas a retirada do cabeçalho do protocolo ESMP. Nesse ponto, ainda não houve operações de *hash* e encriptação. O SBB SM começa a montar seu estado de sessão nesse momento. SBB SM gera um número de sessão e armazena os parâmetros "política de segurança" e "título da entidade", enviados pela primitiva ESMP_SEH. Se o DTSA deferir a versão do protocolo enviada, ele também gravará essa informação;

4. SBB SM envia a primitiva SESSION_DTSA_HELLO (ESMP_SDH) para o NEConnector. Essa primitiva carrega a versão do protocolo, identificador de sessão e certificado. Caso a versão do protocolo da primitiva ESMP_SEH (item 3) seja aprovada pelo DTSA, ele repete essa informação no ESMP_SDH para que a entidade possa posteriormente gravá-la em seu estado de sessão. O identificador da sessão é gerado no DTSA e enviado para a entidade. Quanto ao certificado; se trata da chave pública do DTSA assinado por terceiro. Essa informação é crucial para a troca de chave mestre, que acontecerá posteriormente. A autoridade certificadora que assinou esse documento digital está configurada na política de segurança escolhida;
5. NEConnector encapsula ESMP_SDH em um OFPT_PACKET_OUT e envia-a ao NE;
6. NE envia a primitiva ESMP_SDH para a entidade. Nesse ponto, a entidade começa a montar o seu estado de sessão com as informações que recebe. Neste momento, as informações são: identificador da sessão, versão do protocolo ESMP, identificador da política de segurança, título da entidade e certificado do DTSA;
7. primitiva SESSION_ENTITY_INFORMATION_EXCHANGE (ESMP_SEIE) é enviada ao NE. A entidade, nesse momento, gera um número sequencial de envio de mensagens (*nonce* da entidade), uma chave mestre (de sessão) e um vetor de inicialização para algoritmos de encriptação "*Cipher Block Chaining*" (CBC). Nesse momento, todas as etapas de encapsulamento do ESMP (Fig. 13) são realizadas. O campo conteúdo (parâmetros da primitiva/*payload*) é encriptado utilizando para isso a chave pública do DTSA (certificado) passada através da primitiva ESMP_SDH (item 4). O *hash* calculado nessa fase, utiliza-se dos campos conteúdo/cabeçalho (verificará na recepção apenas integridade); pois o estado de sessão ainda não possui dados compartilhados com o DTSA, como *nonces*, chaves simétricas, etc. É válido salientar que a operação de encriptação assimétrica e a função de *hash* inicial (que verifica apenas integridade) serão executadas exclusivamente nesse passo do *handshake*. Em todos os outros, a encriptação será simétrica (chave mestre) e o

- hash* será feito nos moldes do HMAC, onde informações que só os participantes conhecem entram como parte do processo (verificação de integridade e autenticação);
8. NE encapsula a primitiva do "item 7" em um OFPT_PACKET_IN e envia-a ao DTSA. A primitiva chega em NEConnector;
 9. NEConnector desencapsula a primitiva ESMP_SEIE e a envia para o SBB SM. SBB SM desencapsula os parâmetros da primitiva e armazena-as junto com as outras informações do estado de sessão. Como os dados estão protegidos, o DTSA utiliza sua chave privada para obter os parâmetros. Nesse momento, são armazenados *nonce* da entidade, chave mestre (de sessão) e vetor de inicialização (caso seja necessário);
 10. em contrapartida, o DTSA gera a sua própria sequência (*nonce* do DTSA); gera seu próprio vetor de inicialização (para algoritmos de encriptação CBC); e, envia novamente o *nonce* da entidade (só o DTSA poderia ter esse *nonce*) para que haja o começo da autenticação mútua. Essas informações são colocadas como parâmetros da primitiva SESSION_DTSA_INFORMATION_EXCHANGE (ESMP_SDIE) e enviadas para a NEConnector. Nesse momento, o campo conteúdo da primitiva (parâmetros/*payload*) passa por todas as etapas de encapsulamento do protocolo (Fig. 13), porém, a encriptação é feita utilizando a chave mestre e o *hash* é feito utilizando informações compartilhadas entre os participantes. Nesse caso, utilizam-se a própria chave mestre, e o *nonce* da entidade. Na medida que o estado de sessão vá sendo completado, essas informações compartilhadas tendem a crescer, e o *hash*; nesse contexto, realiza a verificação de integridade, a verificação de autenticação e a verificação de primitivas repetidas;
 11. NEConnector encapsula ESMP_SDIE em um OFPT_PACKET_OUT e envia-a ao NE;
 12. NE envia a primitiva ESMP_SDIE para a entidade. Nesse momento, a entidade decifra o campo "conteúdo" da primitiva (parâmetros/*payload*) através da chave mestre. A partir desse ponto, todos os pacotes serão protegidos através de encriptação simétrica, e todos os *hashs* serão realizados utilizando informações compartilhadas (entre entidade/DTSA). As informações *nonce* do servidor e vetor de inicialização do servidor serão adicionados ao estado de sessão já existente. A informação "*nonce* da entidade" será utilizada para autenticar o DTSA;
 13. primitiva SESSION_ENTITY_FINISHED (ESMP_SEF) é enviada ao NE. A entidade, nesse momento, gera um *hash* com todas as primitivas trocadas do item 01 até o item 12. Essa primitiva possui dois parâmetros: *hash*, e status da primitiva. O status tem o objetivo de confirmar uma transação positivamente (2), confirmar uma transação negativamente (1) e também pode carregar um conteúdo inexpressivo

- (NULO). Nesse momento, a primitiva enviada para o NE possui o *hash* calculado e status NULO;
14. NE encapsula a primitiva do "item 13" em um OFPT_PACKET_IN e envia-a ao DTSA. A primitiva chega em NEConnector;
 15. NEConnector desencapsula a primitiva ESMP_SEF e a envia para o SBB SM. SBB SM desencapsula os parâmetros da primitiva. Como os dados estão protegidos, o DTSA utiliza a chave mestre para obter os parâmetros. Nesse momento, O DTSA autentica a entidade comunicante através do *hash*, que contém alguns dados compartilhados (como nonce da entidade, chave mestre, e outros). A autenticação mútua que havia começado no item 12 (quando a entidade autentica o DTSA), acaba nesse item (quando o DTSA autentica a entidade). Juntamente com essa operação de autenticação mútua, o DTSA consegue autenticar todas as mensagens enviadas no processo de "*handshake* de sessão". Essa verificação de todas as mensagens garante que nenhuma delas foi modificada no decorrer da operação de *handshake*, nem mesmo aquelas desprotegidas que iniciaram o processo (ESMP_SEH e ESMP_SDH);
 16. caso a operação de autenticação das mensagens do "item 15" seja realizada com sucesso, a primitiva SESSION_DTSA_FINISHED (ESMP_SDF) é enviada com o parâmetro de "status_hash" configurado como 2 (autenticação verificada). O DTSA faz os seus próprios cálculos de *hash* do conjunto das mensagens enviadas do item 01 até o item 12 e coloca no parâmetro "*hash*" dessa primitiva. Esse cálculo é realizado independentemente do cálculo feito anteriormente pela entidade (item 13);
 17. NEConnector encapsula ESMP_SDF em um OFPT_PACKET_OUT e envia-a ao NE;
 18. NE envia a primitiva ESMP_SDF para a entidade. Nesse momento, a entidade decifra o campo "conteúdo" da primitiva (parâmetros/*payload*) através da chave mestre e verifica o campo "status_hash" enviado pelo DTSA. Se o seu conteúdo for 2 (autenticação verificada), a entidade analisa o *hash* (de todas as primitivas anteriores) enviado pelo DTSA;
 19. se a análise do *hash* da primitiva do "item 18" for realizada com sucesso, a entidade envia um SESSION_ENTITY_FINISHED (ESMP_SEF) para o SBB SM. Nota-se aqui que houve uma sintetização dessa comunicação, já que esse item está se comunicando diretamente com o SBB SM, e isso não é possível fisicamente. Esse "item 19" pode ser entendido como uma comunicação lógica entre esses dois "seres". O importante é que essa primitiva carrega no campo "hash" um valor nulo e no campo "status_hash" o valor 2, que indica o sucesso da verificação de *hash* calculada pelo DTSA e analisada pela entidade;

20. a primitiva ER.req, armazenada temporariamente (item 3), é enviada pelo SBB SM até o NEConnector;
21. NEConnector reconhece a primitiva ER.req e dispara um evento que é captado pelo SBB EM (responsável pelo registro de entidades);
22. SBB EM realiza o registro da entidade no repositório (esse repositório não foi exibido na Fig. 15) e envia uma primitiva de resposta ETArch (ER.resp) para o NEConnector;
23. mais uma vez o ESMP vai interceptar o itinerário normal de funcionamento de uma primitiva de controle ETArch (caso a mesma não estivesse utilizando mecanismos/serviços de segurança). Nesse ponto, o NEConnector reconhece a primitiva de resposta (ER.resp) e a envia para o SBB SM;
24. esse módulo de segurança (SBB SM) encapsula a primitiva ER.resp, seguindo a política de segurança da sessão, e envia essa primitiva para a entidade. Nota-se aqui novamente a sintetização de comunicação utilizada no item 19. Imaginemos que essa comunicação direta é lógica, pois fisicamente seria impossível esse envio. A entidade desencapsula a primitiva através da chave mestre, calcula os *hashs* necessários, e finalmente, tem em sua posse a resposta do registro da entidade solicitada no item 1. Se a resposta for positiva, o ambiente de comunicação de controle entre entidade e DTSA está seguro e pronto para utilização. Resta ainda montar esse ambiente para a comunicação entre entidades participantes de um *workspace* no plano de dados. Essa primitiva, ESMP, é de utilização geral e sua função é carregar dados ou informações que necessitam de segurança. O seu único parâmetro é um campo denominado "status do ESMP". Nesse caso, em específico, o campo é nulo e a única preocupação dessa primitiva é carregar (de forma segura) a resposta da requisição de registro no DTSA (ER.resp) feita pela entidade.

Depois de realizado o "*handshake* de sessão", todas as comunicações de controle entre entidades registradas e DTSA estão protegidas quanto à integridade das mensagens, autenticação mútua das partes envolvidas (entidades/DTSA), detecção de ataque de replicação de primitivas, e confidencialidade das principais informações de controle. A prova de conceito desses serviços/mecanismos de segurança é demonstrada no capítulo 5 através do método de avaliação proposto.

Depois de criado o estado de sessão, que é responsável pela orientação de segurança das comunicações de controle; temos que verificar o processo que cria o estado de conexão, que é responsável pela orientação de segurança das comunicações de dados.

4.3.3.2 Handshake de conexão

O *handshake* de conexão é responsável pela concretização do estado de conexão. Depois disso, a comunicação de dados pode ser feita de forma segura entre as entidades participantes de um *workspace*. Para isso, algumas operações são importantes: troca de chave de conexão (chave temporária/chave gerada por *workspace*); geração de um identificador de conexão; entre outras.

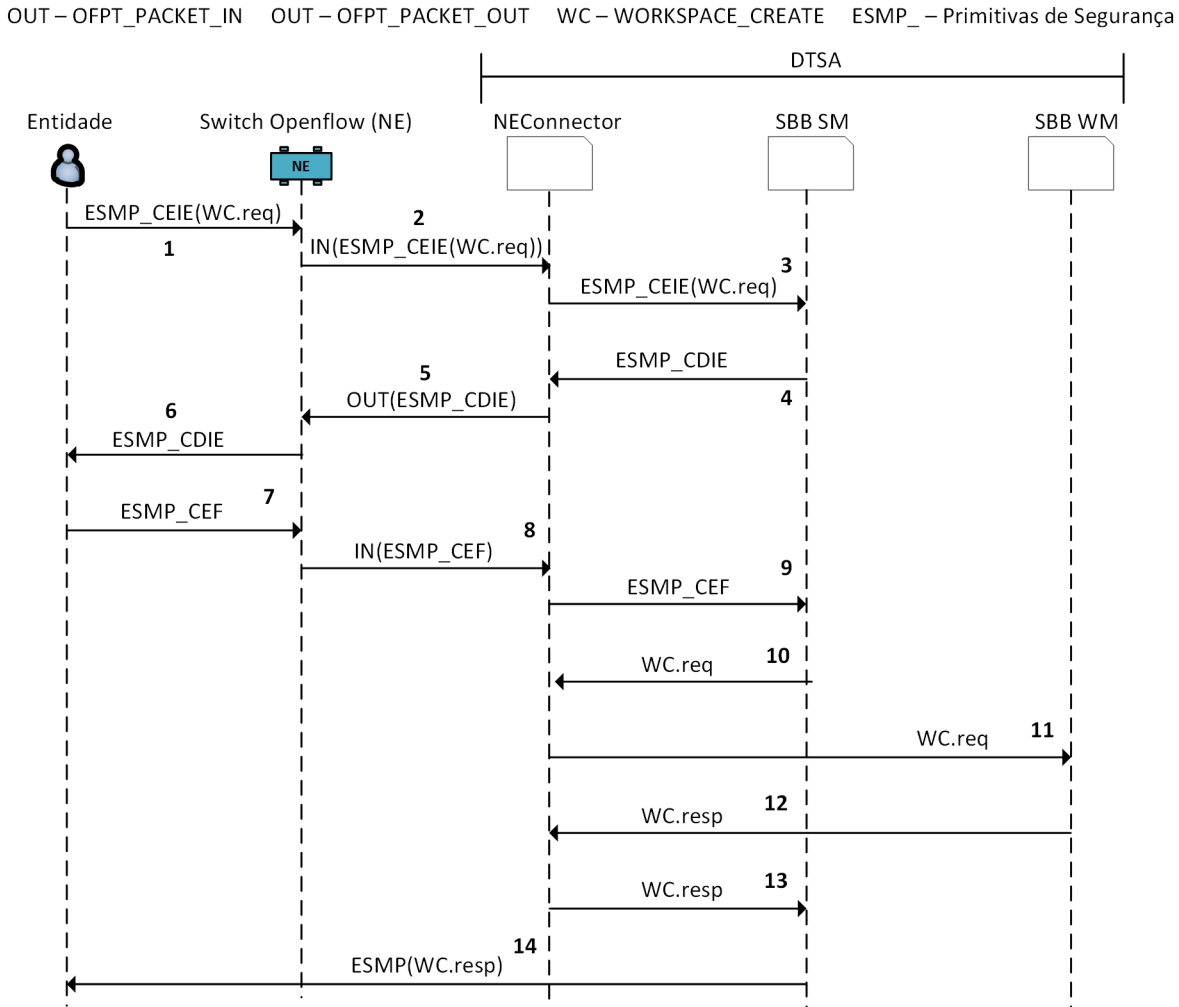
O processo de *handshake* de conexão é mais sintético que o de sessão, pois grande parte das operações que são de crucial importância para a elaboração de um ambiente seguro de comunicação já foram realizados.

A operação desse *handshake* acontece quando a entidade solicita a criação de um *workspace* (WORKSPACE_CREATE) e aciona a política de segurança na sua lista de requisitos.

O processo de "*handshake* de conexão" é apresentado na Fig. 16. Os parâmetros de cada primitiva utilizada podem ser vistos na Tab. 6. Para fins de sintetização, esse diagrama leva em conta cenários de sucesso. Parte-se do pressuposto que todas as trocas de mensagens exibidas são realizadas com êxito. Cada passo dessa operação está descrito detalhadamente abaixo:

1. primitiva CONNECTION_ENTITY_INFORMATION_EXCHANGE (ESMP_CEIE) encapsula a requisição WORKSPACE_CREATE (WC) da entidade e é enviada ao NE. O protocolo ESMP foi ativado, pois a aplicação indicou a política de segurança na sua lista de requisitos da primitiva WC. Diferentemente da primeira primitiva do "*handshake* de sessão", a ESMP_CEIE não está desprotegida. Como já existe, nesse momento, um estado de sessão configurado para a entidade corrente; o campo conteúdo utiliza-se da chave mestre para estabelecer contato com o DTSA. O DTSA já consegue nessa oportunidade: autenticar a entidade, analisar integridade e replicação de primitivas. Os parâmetros passados em ESMP_CEIE são o identificador da sessão e o título do *workspace*;
2. NE encapsula a primitiva do "item 1" em um OFPT_PACKET_IN e envia-a ao DTSA. A primitiva chega em NEConnector;
3. NEConnector desencapsula a primitiva ESMP_CEIE(WC.req). Nesse momento, ao invés do NEConnector enviar o WC para o SBB WM (bloco de serviço responsável pela criação do *workspace*); ele (NEConnector) percebe que a primitiva que chega é uma das mensagens do protocolo ESMP. Nesse ponto, há uma interceptação do módulo de segurança no funcionamento normal do WC. Primeiro faz-se os mecanismos de segurança, depois a primitiva WC pode seguir seu itinerário de funcionamento normal. Nessa perspectiva, o NEConnector envia a primitiva ESMP_CEIE(WC.req) para o SBB SM. SBB SM desencapsula a primitiva

Figura 16 – Diagrama de sequência do handshake de conexão



WC.req e a armazena (temporariamente). A desencapsulação pressupõe a retirada do cabeçalho do protocolo ESMP, a descriptação do campo "conteúdo" através da chave mestre (de sessão); e, as verificações necessárias de autenticação da entidade, integridade e replicação de primitivas. Essas verificações são feitas através de procedimentos realizados na função hash. Logo após, o DTSA começa a construção do estado de conexão através dos parâmetros enviados no "item 1". Primeiramente, ele gera um identificador de conexão, vincula esse identificador de conexão à sessão correspondente e registra o título do *workspace* à essa conexão. Cada estado de conexão pertence a um contexto de comunicação (*workspace*). Nesse momento, também é gerado uma chave de conexão (temporária), que será compartilhada por todas as entidades participantes desse *workspace*;

4. SBB SM envia a primitiva CONNECTION_DTSA_INFORMATION_EXCHANGE (ESMP_CDIE) para o NEConnector. Essa primitiva carrega o identificador da conexão, a chave de conexão (temporária) e o identificador da sessão; os dois primeiros parâmetros foram gerados pelo DTSA (item 3). O identificador da sessão é faculta-

tivo; será necessário apenas se a primitiva encapsulada tratar-se de um "WORKSPACE_ATTACH". O DTSA distribuirá esse estado de conexão (chave de conexão, identificador da conexão, e outros) para todas as entidades que solicitarem "*attach*" nesse *workspace*. Nesse momento, o DTSA está fazendo a montagem desse estado de conexão juntamente com a entidade proprietária dessa comunicação;

5. NEConnector encapsula ESMP_CDIE em um OFPT_PACKET_OUT e envia-a ao NE.
6. NE envia a primitiva ESMP_CDIE para a entidade. Nesse ponto, a entidade começa a montar o seu estado de conexão com as informações que recebe. Neste momento, as informações da conexão são o identificador da conexão, o identificador da sessão vinculado à essa conexão, o título do *workspace*, a chave de conexão e o identificador da política de segurança. A entidade gera um *nonce* (sequência de primitivas) que será utilizado no momento de oferta do serviço. Essa sequência é armazenada no estado de conexão, completando assim todos os seus parâmetros. Esse *nonce*, em especial, existirá apenas nos estados de conexão das entidades que ofertam algum tipo de serviço. Será utilizada na troca de primitivas de dados; sendo totalmente irrelevante para as trocas realizadas no âmbito de controle. Portanto, essa informação não faz parte dos registros de estados de conexão do DTSA. Uma entidade produtora de serviços que oferece transmissão contínua de vídeo, terá suas primitivas numeradas através desse parâmetro;
7. primitiva CONNECTION_ENTITY_FINISHED (ESMP_CEF) é enviada ao NE. O único parâmetro dessa primitiva é o status de finalização. Essa primitiva finaliza as trocas de informações para a construção de um estado de conexão na entidade e no DTSA. Esse status sinaliza se o "item 6" foi executado com sucesso, isto é, nesse caso, se o estado de conexão foi montado na entidade conforme o previsto;
8. NE encapsula a primitiva do "item 7" em um OFPT_PACKET_IN e envia-a ao DTSA. A primitiva chega em NEConnector;
9. NEConnector desencapsula a primitiva ESMP_CEF e a envia para o SBB SM. SBB SM desencapsula os parâmetros da primitiva e verifica se o estado de conexão foi montado na entidade conforme planejado. Se sim, o campo status de finalização conterá o número 2 (procedimento realizado com sucesso);
10. a primitiva WC.req, armazenada temporariamente (item 3), é enviada pelo SBB SM até o NEConnector;
11. NEConnector reconhece a primitiva WC.req e dispara um evento que é captado pelo SBB WM (responsável pela criação dos *workspaces*);

12. SBB WM realiza a criação do *workspace* e registra suas informações no repositório (esse repositório não foi exibido na Fig. 16). Logo após, envia uma primitiva de resposta ETArch (WC.resp) para o NEConnector;
13. mais uma vez o ESMP vai interceptar o itinerário normal de funcionamento de uma primitiva de controle ETArch (caso a mesma não estivesse utilizando mecanismos/serviços de segurança). Nesse ponto, o NEConnector reconhece a primitiva de resposta (WC.resp) e a envia para o SBB SM;
14. esse módulo de segurança (SBB SM) encapsula a primitiva WC.resp, seguindo a política de segurança da conexão, e envia essa primitiva para a entidade. Nota-se aqui que houve uma sintetização dessa comunicação, já que a primitiva ESMP(WC.resp) está sendo enviada diretamente para a entidade, e isso não é possível fisicamente. Essa comunicação pode ser entendida como uma comunicação lógica entre esses dois "seres". Essa primitiva, ESMP, é de utilização geral e sua função é carregar dados ou informações que necessitam de segurança. O seu único parâmetro é um campo denominado "status do ESMP". Nesse caso, em específico, o campo é nulo e a única preocupação dessa primitiva é carregar (de forma segura) a resposta da requisição de "criação do *workspace*" (WC.resp) feita pela entidade.

Depois de realizado o "*handshake* de conexão", a comunicação existente dentro do contexto de um *workspace* pode ser realizada de forma segura. Entende-se por forma segura, a oferta de serviços que o protocolo ESMP é capaz de oferecer ao plano de dados depois das duas operações de *handshake*, quais sejam: autenticação mútua entre as entidades que participam do *workspace*; confidencialidade das informações trocadas; integridade das primitivas de dados e, disponibilidade no que concerne aos mecanismos de segurança (detecção de ataques de replicação pelas entidades comunicantes).

Alguns dos conceitos mencionados acima já foram descritos nas fases de encaminhamento de primitivas explicadas nos dois *handshakes* propostos na ETArch. Para reforçar esses conceitos descritos, faremos uma prova de conceito desses serviços/mecanismos de segurança no Capítulo 5 através do método de avaliação proposto.

4.3.3.3 Handshake de conexão do WORKSPACE_ATTACH

O *handshake* de sessão é responsável pela construção de um estado de sessão entre entidade e DTSA. Esse estado orienta a comunicação de forma segura entre essas duas entidades. Esse processo se dá quando a entidade solicita um registro no seu DTSA através do protocolo de controle ENTITY_REGISTER (ER).

O *handshake* de conexão estabelece um estado de conexão na entidade e no DTSA. Na entidade, esse estado de conexão é responsável pela orientação de segurança no plano de dados. No DTSA, esse estado de conexão fica registrado para posterior distribuição

dos seus parâmetros. Esse processo se dá quando a entidade solicita a criação de um *workspace* ao seu DTSA através do protocolo de controle WORKSPACE_CREATE, e esse estado de conexão será consumido na comunicação do plano de dados.

Caso alguma entidade queira utilizar os serviços de um *workspace* já criado, ela (entidade); primeiramente tem que se registrar. O registro cria o estado de sessão explicado em 4.3.3.1. Nota-se que o estado de sessão é vinculado à entidade, e que cada entidade possui a sua própria negociação com o DTSA.

O segundo passo é a entidade requisitar o seu "attach" no *workspace*. Nesse momento, há algumas diferenças entre este *handshake* de conexão (do *attach*) e o realizado na sub-seção 4.3.3.2. A primeira diferença é que a primitiva de controle ETArch encapsulada pela primitiva ESMP_CEIE (Fig. 16) é o WORKSPACE_ATTACH (WA). A segunda diferença é que a entidade não conhece o identificador de sessão do *workspace* que está requisitando, pois esse estado de sessão foi criado pela entidade proprietária; portanto, o parâmetro correspondente a essa informação é NULO.

O DTSA também terá um comportamento diferente. No *handshake* da primitiva ER, o DTSA gera alguns parâmetros do estado de conexão, como chave de conexão (temporária), identificador da conexão, e outros. No *handshake* da primitiva WA, o DTSA apenas pesquisa o estado de conexão do *workspace* correspondente (que está registrado) e passa as informações para a primitiva ESMP_CDIE (Fig. 16). Todo o processo de troca de mensagens do *handshake* de conexão pode ser visto na Fig. 16. Essa seção, cita apenas as diferenças entre o *handshake* do protocolo ER e do protocolo WA.

No fim desse processo, a entidade possui o seu próprio estado de sessão (único para cada entidade) e o estado de conexão utilizado pelo *workspace* que ele requisitou. Esse estado de conexão não foi construído pela entidade que solicitou o "attach", e sim pelo proprietário do *workspace*. O DTSA apenas distribui um estado de conexão feito anteriormente por esse proprietário. Por esse motivo, a comunicação no plano de dados não pode utilizar parâmetros de sessão, e nem comunicação de controle entre ENTIDADE/DTSA devem utilizar parâmetros de conexão. No caso dessa entidade que fez o "attach", ela não conhece o estado de sessão vinculado a esse estado de conexão que está sendo distribuído pelo DTSA. Por esse motivo, a orientação de comunicação segura do plano de dados e do plano de controle devem ser totalmente independentes.

4.4 Comunicação no plano de dados

Antes que a comunicação segura de dados entre entidades participantes de um determinado contexto de comunicação (*workspace*) se concretize, é necessário que as operações de controle do protocolo ESMP já tenham sido executadas. Nesse momento, todas as trocas de informações necessárias para que o plano de dados possa desempenhar seguramente as suas transmissões foram realizadas.

A Fig. 12 exibe bem essa situação. O "item 13" representa a comunicação de dados da arquitetura ETArch. A transmissão das informações começa na entidade (Entidade 01, por exemplo) . A primitiva de dados é recebida pela "*Communication Layer*" e encapsulada pelo protocolo ESMP. Entende-se por encapsulamento o processo de encapsulamento apresentado na Fig. 13. Basicamente, o protocolo acrescenta um MAC¹, encripta, acrescenta o seu cabeçalho e envia as primitivas para todas as entidades pertencentes à comunicação.

Todas as informações necessárias para a realização da comunicação estão no estado de conexão da entidade. Nessa especificação do protocolo ESMP, a chave de conexão (temporária) é utilizada tanto nos algoritmos de encriptação simétrico quanto na geração de valores fixos de funções MAC¹. Esse valor fixo gerado é uma técnica de autenticação bem conhecida e denomina-se "soma de verificação criptográfica" ou MAC¹. Poder-se-ia ter uma chave distinta para cada uma dessas operações, mas essa decisão pertence mais à fase de implementação do que à fase de especificação.

É importante salientar que o tipo de mensagem (do protocolo ESMP) utilizado para carregar informações gerais que não possui um conjunto de parâmetros específicos ou uma funcionalidade determinística, como é o caso da maioria das primitivas do *handshake*, é o ESMP(status_ESMP). Conforme Tab. 6, essa primitiva carrega apenas o parâmetro opcional "status do ESMP". Entende-se por esse parâmetro uma resposta de sucesso ou fracasso na realização de determinadas operações. Como é opcional, pode-se encontrar o valor NULO atribuído a esse status. No transporte do plano de dados, esse parâmetro não tem grande importância; porém, foi colocado para precisões futuras.

Quando essa primitiva chega nas entidades de destino, o cabeçalho do protocolo ESMP é retirado, o campo conteúdo é decriptado, verificado (através das operações de hash), e entregue à entidade (Entidade 02, por exemplo) receptora. Todas essas operações são orientadas pela política de segurança registrada no estado de conexão de cada entidade que participa do *workspace*.

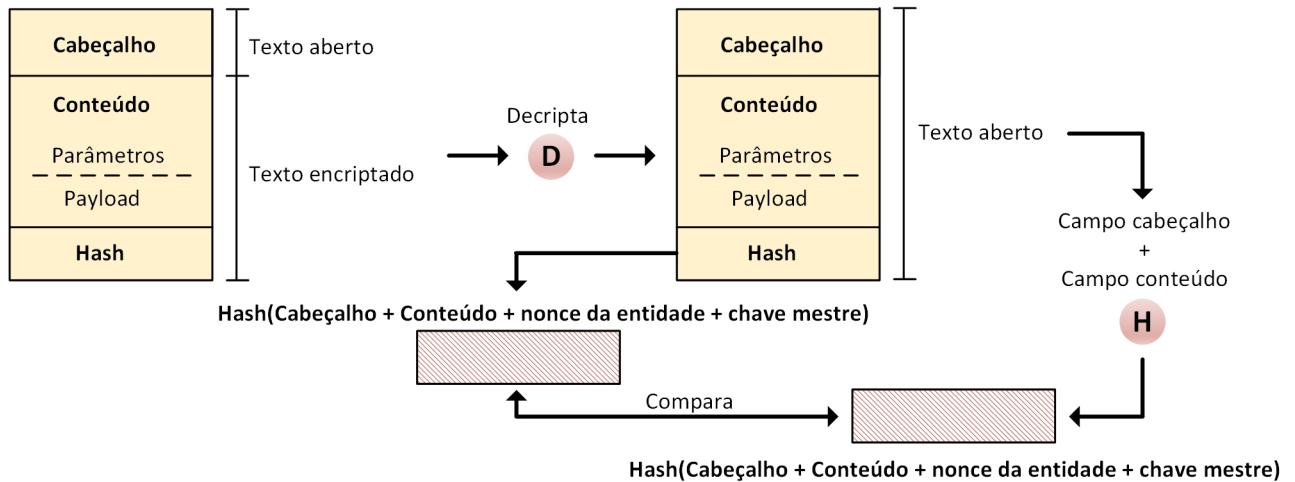
A Entidade 01 (Aplicação 01) e a Entidade 02 (Aplicação 02) utilizadas, nessa seção, como exemplos de entidades comunicantes podem ser vistas na Fig. 6. Vê-se que o encapsulamento/descapsulamento é realizada pela "*Communication Layer*" presentes nos hospedeiros dessas entidades.

4.5 Operação de hash do protocolo ESMP

A operação de *hash* da arquitetura ETArch se dá de acordo com a Fig. 17. Essa operação pode apresentar diferenças sutis dependendo da situação em que está sendo executada, porém, genericamente, segue o modelo operacional apresentado nessa figura.

Primeiramente, há a decifração da primitiva do protocolo ESMP. Conforme visto em 4.3.2, é encriptado na origem apenas o campo "conteúdo" e o campo "*hash*".

Figura 17 – Operação de hash do protocolo ESMP



Logo após, o DTSA aplica a função de *hash* na concatenação do campo conteúdo com o campo cabeçalho. Outras informações, como *nonces* e chaves compartilhadas, podem se juntar à concatenação desses dois campos. As informações que são passadas para a função de *hash* depende do momento do seu processamento. Algumas primitivas são enviadas sem cálculo de *hash*, pois no momento desse envio ainda não há nenhuma informação compartilhada entre as partes. Um exemplo dessa primitiva é ESMP_SEH (ver Fig. 15). Outras primitivas já executam o cálculo de *hash* com todas as informações de interesse. Um exemplo dessa primitiva é a ESMP_SDF (ver Fig. 15). Os parâmetros passados para a função de *hash* depende do quão completo está o estado de sessão e o estado de conexão do DTSA e das entidades, respectivamente.

É importante salientar que o "*nonce* da entidade" e a "chave mestre" já são informações registradas no DTSA, e por conta disso, essas informações não foram buscadas da primitiva que a entidade enviou, e sim, do estado de sessão do DTSA. O identificador da sessão está vinculado à entidade e por conta disso, essa busca é possível e fácil de efetivar.

A última operação é comparar o *hash* calculado pelo DTSA com o *hash* enviado na primitiva. Se os hashes forem iguais, pode-se tirar algumas conclusões: a entidade é realmente quem diz ser (autenticação da entidade verificada com sucesso); a primitiva não foi modificada no decorrer da comunicação (integridade da mensagem verificada com sucesso); e, o pacote não é repetido.

A autenticação da entidade é possível graças às informações compartilhadas (que só entidade/DTSA) conhecem no momento da verificação do *hash*. A integridade da mensagem é possível, pois se alguma porção dos campos "Cabeçalho", "Conteúdo" e "Hash" houvessem sido modificados, os hashes comparados seriam diferentes. Quanto à repetição de pacotes, é possível verificá-la, pois uma das informações compartilhadas é o *nonce* da entidade. Essa sequência é conhecida pelo DTSA, pois é um dos campos registrados no estado de sessão. O conhecimento dessa informação garante a rejeição de pacotes repetidos, que

podem estar sendo enviados por um ataque de replicação.

A seguir será demonstrado cada um dos serviços de segurança que o ESMP se propõe a resolver, quais sejam: integridade, autenticação, confidencialidade e disponibilidade. Esta análise será realizada por um método de avaliação que será detalhado no Cap. 5.

Análise e discussão dos Resultados

Este capítulo apresenta reflexões sobre a especificação do protocolo de segurança multicast (ESMP) para arquiteturas de Internet do Futuro proposta no capítulo 4. A escolha da aplicação das funcionalidades dessa especificação na arquitetura ETArch deveu-se às diversas vulnerabilidades existentes nos seus serviços/mecanismos de segurança (vistas em 3.10). Aliás, a Tab. 5 mostra que os serviços/mecanismos da ETArch (status quo) mitigam apenas 02 dos 09 ataques que são objetos de preocupação do ESMP. É válido salientar que essas mitigações, devido à critérios de avaliação de segurança desse trabalho, não dizem respeito diretamente aos mecanismos/serviços de segurança da ETArch (status quo). A análise de tráfego é mitigada devido ao ocultamento que a arquitetura ETArch (status quo) realiza do endereço de origem e destino dos hosts. A negação de serviço é mitigada devido à capacidade da arquitetura em calcular novas rotas de primitivas por conta de problemas de congestionamento e à capacidade de gerenciar controle de largura de banda. Nota-se que se na ETArch não houvesse o módulo SBB SM (responsável pela implementação inicial de mecanismos de autenticação e controle de acesso); ainda assim a ETArch atenuaria esses mesmos ataques devido à recursos da arquitetura. Esse fato torna a ETArch (status quo) uma arquitetura totalmente desprovida de contramedidas de segurança necessárias para confrontar os ataques clássicos abordados na tabela em questão.

O objetivo desse capítulo é realizar uma análise de segurança de alto nível, extensível e adaptável do ESMP quanto à sua meta principal de criar uma rede de confiança, onde as entidades comunicantes possam confiar uma nas outras a fim de estabelecer uma comunicação segura. Para esse fim, são utilizadas técnicas de modelagem de ameaças (HERNAN et al., 2006) que possui como principal objetivo expor as vulnerabilidades do sistema. Nessa primeira fase de análise, é feita uma modelagem dos principais componentes do ESMP e um estudo dos seus mecanismos/serviços de segurança.

Em uma segunda fase de análise, o objetivo é expor essas vulnerabilidades à ataques e fazer uma avaliação do comportamento dos mecanismos de segurança. Para esse fim, é utilizada técnicas de modelagem de ataques (SAINI; DUAN; PARUCHURI, 2008). Nesse

momento, há a descrição de como os ataques podem explorar as ameaças expostas na primeira fase de análise.

Essas duas técnicas (modelagem de ameaças/modelagem de ataques) já foram aplicados na análise e avaliação das soluções de segurança do SSL/TLS, do IPSec e da arquitetura MobilityFirst no Cap. 2; e da ETArch (status quo) na Seção 3.10. Na ocasião, essas análises e avaliações foram feitas baseadas na literatura do estado da arte, protótipos e demonstrações das soluções de segurança empregadas por esses protocolos/arquiteturas descritas nesses capítulos. A análise e avaliação desses protocolos/arquiteturas visavam o cumprimento das metas clássicas de segurança, quais sejam: confidencialidade, integridade, disponibilidade e autenticação. Essas mesmas metas são utilizadas nesse capítulo para avaliar e analisar o ESMP, e a análise têm como fundamento teórico a especificação apresentada no Cap. 4. Toda a análise e avaliação de segurança feita neste capítulo cobre a comunicação do plano de dados/plano de controle da arquitetura ETArch (ver Fig. 6).

Para a análise, fizemos algumas suposições sobre os recursos dos atacantes: não conseguem obter acesso ao canal de controle seguro que fornece conectividade entre um comutador OpenFlow e seu DTSA; não podem comprometer os sistemas finais (endpoints), que são materializadas por todas as entidades (DTSAs, aplicações, sensores, etc.) comunicantes da arquitetura ETArch; no entanto, possuem acesso a todas as informações do plano de dados, que é o montante de todas as primitivas de controle e de dados transmitidas pela rede. Essas suposições de recursos baseiam-se em três motivos: (i) assumimos que o ISP tomou medidas razoáveis para proteger o canal de comunicação entre comutador OpenFlow e DTSA (por exemplo, via TLS); (ii) assumimos que toda entidade está protegida contra vírus, worms, etc.; (iii) queríamos nos concentrar nos ataques que surgem devido às ameaças existentes na comunicação de rede.

O restante do capítulo segue assim: inicialmente será descrito o método de análise e avaliação de segurança utilizado para validar as hipóteses descritas na Seção 1.3; na Seção 5.2 é feita a análise e avaliação da especificação do protocolo ESMP em relação às metas de segurança do protocolo; por fim, a Seção 5.3 apresenta os resultados da avaliação do ESMP e uma tabela comparativa entre a proposta apresentada nesse trabalho (ESMP) e as arquiteturas/protocolos apresentados no Cap. 2 e na Seção 3.10.

5.1 Método para a análise e avaliação de segurança do ESMP

Para implantar uma arquitetura ou um protocolo na Internet, é necessário saber se eles atendem os requisitos de segurança. No caso da especificação do ESMP apresentada no Cap. 4; ela tem como principal objetivo criar uma rede de confiança, onde a comunicação têm que cumprir as metas clássicas de segurança: confidencialidade, disponibilidade, autenticação e integridade.

A segurança de um sistema pode ser analisada de duas maneiras: centrada no sistema e centrada no ataque (KLOTI; KOTRONIS; SMITH, 2013). A primeira utiliza uma técnica de modelagem de ameaças. A segunda utiliza uma técnica de modelagem de ataques à possíveis vulnerabilidades.

Esse trabalho utiliza uma abordagem híbrida, apropriada para analisar o ESMP a partir da perspectiva de modelar ameaças existentes após análise dos seus mecanismos de segurança. Em uma segunda fase, faz-se a modelagem de ataques às vulnerabilidades encontradas. Essas duas modelagens (vulnerabilidades/ataques) são contrapostas para validar se os mecanismos de segurança superam os ataques modelados. Essa abordagem combinada fornece uma visão melhor das vulnerabilidades das arquiteturas aos ataques do que seguir uma das metodologias (KLOTI; KOTRONIS; SMITH, 2013).

Na fase de análise ou da modelagem das ameaças existentes no sistema, um dos mecanismos utilizados é decompor o sistema em seus componentes relevantes e analisar cada componente quanto à suscetibilidade às ameaças (HERNAN et al., 2006). Pode-se citar como exemplos de componentes: sistemas finais (hosts, servidores, etc.); fluxo de dados (entre camadas da arquitetura ou pela Internet); armazenamento de dados (processo de armazenamento em repositórios); processos (cálculos ou programas executados pelo hospedeiro das entidades) (HERNAN et al., 2006). Há várias formas de representar os componentes e os fluxos entre eles. Pode-se utilizar diagramas de fluxo de dados (Data Flow Diagrams - DFDs), diagramas UML, e outros (HERNAN et al., 2006).

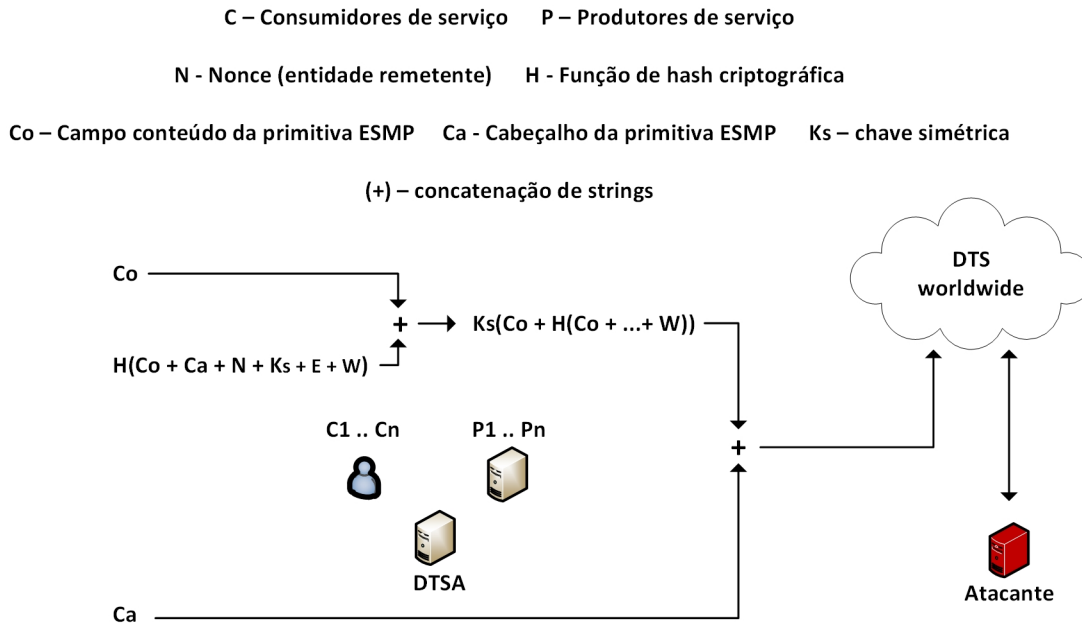
O ESMP se preocupa com a segurança da comunicação de rede, ou seja, em estabelecer uma rede de confiança onde entidades da ETArch possam se comunicar de forma segura, tanto no plano de controle quanto no plano de dados. Por conta disso, o componente de preocupação da análise do ESMP é o fluxo de dados, e a representação da troca de informações dessas comunicações é representada através de diagramas de sequência UML das Figs. 15 e 16. A Fig. 12, item 13, representa de forma genérica a comunicação no plano de dados.

Outra representação é importante para o entendimento da análise de segurança da sessão 5.2, que é a representação genérica das operações que são executadas na fase de encapsulamento das primitivas de controle/dados do protocolo ESMP. Essa representação (Fig. 18) auxilia no entendimento de como as informações trafegam na rede ETArch e como elas ajudam a conter os ataques que são descritos através da metodologia de modelagem de ataques ao sistema. Uma das metodologias conhecidas para modelar esses ataques é conhecida como árvores de ataques (SAINI; DUAN; PARUCHURI, 2008). Nesse trabalho, a modelagem desses ataques é descritiva.

Depois de detalhado os mecanismos que são utilizados nesse capítulo para análise e verificação de segurança do ESMP, segue abaixo a metodologia híbrida utilizada:

1. definir as metas de segurança do ESMP;

Figura 18 – Envio de primitivas ESMP para a rede mundial de comunicação ETArch



2. especificar as ameaças/ataques que inibem as metas de segurança do ESMP;
3. analisar as soluções de segurança do ESMP e os recursos da arquitetura ETArch;
4. combinar os itens 2 e 3 para avaliar a superação dos serviços/mecanismos de segurança em relação aos ataques.

5.2 Análise e avaliação de segurança do ESMP

Essa seção tem o objetivo de demonstrar as metas do protocolo ESMP através do método de análise e avaliação explanado em 5.1. A ideia é validar essas metas e demonstrar a melhoria de uma eventual implantação desse protocolo na arquitetura ETArch.

5.2.1 Definição das metas de segurança do ESMP

A meta geral do ESMP é criar uma rede de confiança. Entende-se por rede de confiança quando uma entidade pode confiar em outra entidade para a realização de uma comunicação segura. Em outras palavras, o objetivo geral é dar as pessoas liberdade para desfrutar dos recursos oferecidos pela rede de computadores sem medo de comprometer seus direitos e interesses.

Para alcançar esse objetivo geral, quatro metas de segurança são necessárias para a elaboração dessa rede de confiança, quais sejam: confidencialidade, integridade, disponibilidade e autenticação.

É meta do protocolo ESMP que todo serviço/mecanismo de segurança oferecido seja efetivo tanto no plano de controle quanto no plano de dados. No plano de controle,

as negociações de segurança serão entre entidade/DTSA. Cada uma dessas entidades montará uma sessão peer-to-peer com seus respectivos DTSA's. No caso do plano de dados, os mecanismos/serviços de segurança são aplicados, geralmente, em contextos de comunicação multicast.

5.2.2 Especificação das ameaças/ataques que inibem as metas de segurança do ESMP

A seguir, é feito um mapeamento das ameaças/ataques que inibem as metas de segurança do ESMP, e são descritas as contramedidas para mitigá-los. A relação entre ameaças, ataques e contramedidas são descritas abaixo por meta de segurança.

5.2.2.1 Ameaças/ataques que inibem a meta confidencialidade

Ataque de espionagem (Snooping) e ataque de análise de tráfego (traffic analysis attack) são duas possíveis ameaças contra a confidencialidade (STALLINGS, 2014) (KHONDOKER et al., 2014).

O objetivo do ataque de espionagem é capturar as primitivas das entidades comunicantes para a leitura de informações que possam trazer algum tipo de vantagem para o atacante. Esse ataque pode ser evitado com um mecanismo de encriptação de dados para proteger as primitivas (STALLINGS, 2014).

Quanto ao ataque de análise de tráfego, é mais sutil. O atacante extrai informações de padrões de tráfego. Ele poderia observar frequência e tamanho das mensagens, a identidade e a localização das entidades comunicantes, entre outros. Essas informações são úteis para descobrir a natureza da comunicação (STALLINGS, 2014). Uma forma de atenuar esse ataque é ocultar a identidade das entidades comunicantes (STALLINGS, 2014) (KHONDOKER et al., 2014).

5.2.2.2 Ameaças/ataques que inibem a meta integridade

Duas ameaças contra a integridade são os ataques de modificação (modification attack) e repúdio (repudiation) (KHONDOKER et al., 2014).

Ataque de modificação é quando uma mensagem original é alterada, ou seja, a mensagem que é enviada pelo remetente não é a mesma mensagem recebida pelo receptor (STALLINGS, 2014). Mecanismos de segurança que envolvem funções de MAC¹ e/ou assinatura digital conseguem verificar a integridade da mensagem.

Repúdio é um processo pelo qual as entidades não têm certeza se uma comunicação ocorreu entre elas. Ambas podem negar a autoria da transação efetuada (DULANEY, 2009). É inevitável que se tenha informações compartilhadas apenas entre as duas entidades comunicantes para que o repúdio seja atenuado e/ou seja utilizado o serviço de assinatura digital na comunicação. Desse modo, pode-se utilizar mecanismos de hash

com essas informações compartilhadas ou assinatura digital para atenuar essa ameaça. De toda forma, é necessário a presença de um terceiro para distribuir as informações compartilhadas de forma segura entre as entidades.

5.2.2.3 Ameaças/ataques que inibem a meta disponibilidade

O ataque de negação de serviço (DoS) é uma ameaça contra a disponibilidade. Um exemplo desse ataque é inundar a rede com rajadas de primitivas (MIRKOVIC et al., 2004). Esse ataque pode ser evitado com a modificação de controle do fluxo das primitivas e controle de alocação de banda (KHONDOKER et al., 2014).

5.2.2.4 Ameaças/ataques que inibem a meta autenticação

Os ataques que são ameaças contra a autenticação são: ataque man-in-the-middle, reflexão (reflection attack), mascaramento (masquerading) e repetição (replaying attack) (KHONDOKER et al., 2014).

O atacante main-in-the-midde se localiza entre o remetente e o receptor. Um exemplo de ataque é o atacante intermediar a comunicação entre o remetente e o receptor, até que ele (atacante) obtenha a chave compartilhada dessa comunicação. A partir daí, o atacante não interfere mais ativamente na comunicação; apenas espiona. Tanto remetente quanto receptor não estão cientes do problema, nem sequer sabem que existe um terceiro lendo as mensagens trocadas. (STALLINGS, 2014). A forma de mitigação desse ataque é realizar autenticação mútua através de assinatura digital (KHONDOKER et al., 2014) e/ou funções de MAC¹. O pré-requisito para solucioná-lo completamente é que as entidades comunicantes já tenham chaves compartilhadas antes mesmo da primeira comunicação.

O ataque de reflexão ocorre quando um atacante finge ser uma das entidades autorizadas. Esse ataque pode ser atenuado através da autenticação mútua entre as entidades feita através de assinaturas digitais (KHONDOKER et al., 2014) e/ou funções de MAC¹. O pré-requisito para solucioná-lo completamente é que as entidades comunicantes já tenham chaves compartilhadas antes mesmo da primeira comunicação.

No mascaramento, o atacante modifica a primitiva com o objetivo de se passar por um usuário autorizado para obter acesso aos serviços disponibilizados. (KHONDOKER et al., 2014). Esse ataque pode ser impedido através da verificação de autenticação da entidade remetente.

O ataque de repetição ocorre quando o atacante captura mensagens de comunicação e as utiliza mais tarde em outras sessões dessa mesma comunicação. Um gerador repetitivo dessas mensagens pode gerar, por exemplo, indisponibilidade de serviço no host atacado. Uma maneira de mitigar esse ataque é verificar a integridade da mensagem através de um nonce sequencial vinculado à sessão/conexão da comunicação (KHONDOKER et al., 2014).

5.2.3 **Análise das soluções de segurança do ESMP e da arquitetura ETArch**

A especificação completa do ESMP foi descrita no Cap. 4 e os recursos da arquitetura ETArch estão descritas no Cap. 3. Esses dois capítulos compõem o que existe de soluções de segurança em uma possível implantação do ESMP na ETArch. Essas soluções são discutidas na Subseção 5.2.4, quando combinarmos o conjunto dessas soluções de segurança com a modelagem de ameaças especificadas na Subseção 5.2.2.

O componente mais importante para a avaliação do ESMP é o fluxo de dados/controlado das primitivas desse protocolo, pois esse é o componente que é avaliado pela sessão seguinte. O diagrama de sequência UML das Figs. 15 e 16 detalham o fluxo das primitivas de controle nos componentes da arquitetura ETArch. A Fig. 12 (item 13) apresenta de forma mais genérica a transmissão de primitivas de dados pelos componentes da ETArch presentes nos hospedeiros das entidades comunicantes.

5.2.4 **Combinação dos mecanismos de segurança do ESMP e da ETArch com as ameaças mapeadas pelo processo de análise**

Essa subseção é a última fase do processo de análise e de avaliação dos mecanismos de segurança do ESMP e dos recursos da ETArch. Alguns recursos da ETArch intrínsecos da arquitetura podem ser vistos também como mecanismos de segurança, e nesse contexto, tais recursos adicionam-se aos mecanismos de segurança do ESMP com o objetivo de auxiliar na mitigação das ameaças modeladas na Subseção 5.2.2.

A avaliação de cada um desses recursos de segurança se dá no momento de contraporlos às ameaças que podem inibir as metas de segurança do ESMP. O objetivo é avaliar a superação dos mecanismos de segurança do ESMP e da ETArch em relação aos ataques iminentes na rede.

Ao todo, são nove os ataques/ameaças mapeados. O comportamento dos mecanismos de segurança para mitigar cada um desses ataques/ameaças é descrito abaixo.

5.2.4.1 **Ataque de espionagem (Snooping)**

O ESMP, a partir da etapa 07 do seu primeiro handshake (Fig. 15), possui encriptação em todas as suas primitivas de controle/dados. As etapas 07 a 09 são encriptadas a partir da chave pública do DTSA, e todas as outras primitivas (a partir da etapa 10) já começam a ser encriptadas através da chave simétrica registrada no estado de sessão entre entidade e DTSA. Quanto ao plano de dados, as primitivas são encriptadas através da chave simétrica registrada no estado de conexão das entidades participantes do workspace. Desse modo, a partir da etapa 10, a encriptação segue o padrão de execução da Fig.

18. É importante mencionar que as informações transportadas da etapa 01 até a etapa 06 não estão protegidas, porém, não são informações relevantes e que traz benefícios para um possível atacante. Dentre essas informações encontram-se: versão do protocolo, identificador da política de segurança, certificado público do DTSA, entre outras.

Nesse contexto descrito do plano de controle e de dados, pode-se dizer que o ESMP cumpre à meta de segurança de confidencialidade e consegue mitigar o ataque de espionagem.

5.2.4.2 Ataque de análise de tráfego (traffic analysis attack)

No caso da ETArch, o endereço de origem e destino é indireto, ou seja, não se trata do endereço das entidades comunicantes (origem e destino); e sim, do contexto de comunicação onde os serviços são oferecidos. É saliente mencionar, que mesmo se o atacante obtivesse o título das entidades, ainda assim não saberia sua localização por conta da separação que a ETArch realiza entre identificador e localização.

Nesse contexto descrito, tanto as primitivas do plano de controle quanto as primitivas do plano de dados são protegidas contra eventuais análises de padrão de tráfego por dois motivos: o primeiro é que as entidades participantes da comunicação não aparecem no cabeçalho das primitivas do ESMP e o segundo é que a localização dessas entidades não estão informadas em seu título. Uma terceira informação é importante: os títulos das entidades são codificados através de funções de hash, o que atrapalha a leitura feita pelo atacante na tentativa de descobrir o título das entidades envolvidas no contexto de comunicação.

5.2.4.3 Ataque de modificação (modification attack)

O ESMP possui verificação de integridade de mensagem através de funções de hash. A partir da etapa 07 da Fig. 15, todas as primitivas de controle e dados terão suas primitivas verificadas no que concerne à integridade dessas mensagens. Essa verificação se dá através do cálculo de funções MAC¹, que basicamente possui como parâmetro uma função de hash e dados compartilhados. O processo de verificação de integridade através de função de hash e o mecanismo de cálculo dessa função nas primitivas transmitidas na rede podem ser vistos nas Figs 17 e 18, respectivamente.

Da etapa 01 até a etapa 06 do primeiro handshake, as primitivas ainda não podem ser autenticadas devido à falta de informações compartilhadas entre entidade e DTSA. Contudo, o processo de handshake realiza em suas etapas finais a verificação de integridade de todo o histórico de mensagens trocadas (item 01 até item 12) e com isso, garante que nenhuma dessas mensagens do primeiro handshake foi modificada. Todas as próximas primitivas do plano de controle/plano de dados são verificadas através da função de hash.

Nesse contexto descrito, tanto as primitivas do plano de controle quanto as primitivas do plano de dados são protegidas contra eventuais ataques de modificação que podem ocorrer devido a interceptação das primitivas por um eventual atacante.

5.2.4.4 Repúdio (repudiation)

O ESMP não consegue atenuar a ameaça de repúdio através do gerenciamento de chaves híbrido (Seção 4.2). Esse gerenciamento serve de modelo para a especificação descrita no Cap. 4. No entanto, a política de segurança do protocolo ESMP sugere outras duas formas de gerenciamento de chaves: (i) distribuição de chave simétrica usando encriptação simétrica; (ii) distribuição de chave simétrica usando encriptação assimétrica.

Os modelos de gerenciamento (i) e (ii) possuem algumas semelhanças de distribuição; ambos possuem chaves compartilhadas antes mesmo da primeira comunicação. No caso (i), o DTSA tem necessariamente que ter uma chave mestre compartilhada com cada uma de suas entidades. No caso (ii), todas as entidades de comunicação, tem necessariamente, que possuir um par de chaves pública/privada e certificados validados por terceiros antes da primeira comunicação.

No caso (i), o repúdio poderia ser atenuado; caso as chaves simétricas fossem distribuídas par a par no plano de dados para a realização de uma comunicação multicast. No entanto, se assim fosse; a escala de chaves necessárias para essa comunicação seria inviável (ver Fig. 11(a)). Desse modo, o repúdio também não é resolvido por esse gerenciamento de chaves, pois uma das metas do ESMP é sintetizar a escala de chaves de uma comunicação multicast.

No caso (ii), todas as comunicações poderiam utilizar o recurso da assinatura digital. Nesse caso, haveria uma sobrecarga devido à ineficiência de desempenho de encriptação/desencriptação dos algoritmos assimétricos. Contudo, o repúdio seria resolvido para a arquitetura ETArch.

O fato é que esse trabalho não cria a especificação de (i) e (ii). Pela discussão anterior, percebe-se que (ii) soluciona o problema do repúdio em uma comunicação multicast, porém, essa especificação ainda não foi criada. Nesse contexto, a versão de especificação ESMP desse trabalho não mitiga o repúdio.

5.2.4.5 Ataque de negação de serviço (DoS)

A ETArch possui dois recursos de arquitetura que funcionam como mecanismos de segurança contra essa ameaça/ataque. O primeiro diz respeito ao roteamento orientado à workspace, que pode eventualmente atualizar as suas rotas devido às necessidades das entidades comunicantes. Um exemplo é modificar as rotas devido à necessidade de percorrer um caminho com maior eficiência energética (NETO et al., 2016a). Outro exemplo é a modificação automática e transparente das rotas devido à mobilidade das entidades que participam do contexto de comunicação (SILVA, 2013). Parte-se do pressuposto que

a ETArch pode mitigar esse ataque através desse recurso, no momento em que as entidades não estejam recebendo um serviço de forma adequada; devido, por exemplo, a um congestionamento de rede provocado pelo DoS.

O segundo é a capacidade do projeto SMART (LEMA, 2014) em reservar recursos de banda para o contexto de comunicação. Em outras palavras, só participa da comunicação entidades autorizadas pelo DTSA, ou seja, a entidade só participa da comunicação caso haja disponibilidade de recursos de banda, de tal forma que a associação de mais uma entidade na comunicação não sobrecarregue os NEs da arquitetura. Esse controle consegue mitigar congestionamento de rede por ataques DoS.

Esses dois recursos da arquitetura ETArch, não solucionam totalmente, mas conseguem mitigar ameaças/ataques dessa natureza.

5.2.4.6 Ataques man-in-the-middle e de reflexão (reflection attack)

Ataques man-in-the-middle e de reflexão (reflection attack) só podem ser totalmente solucionados se o processo de gerenciamento de chaves possuir como pré-requisito a troca de chaves compartilhadas antes mesmo da primeira comunicação de controle ou de dados. Nesse aspecto, o ESMP possui duas abordagens que satisfazem esse pré-requisito: distribuição de chaves simétricas utilizando chaves simétricas e distribuição de chaves simétricas usando encriptação assimétrica. No caso da abordagem híbrida explicada no Cap. 4 há alguns infortúnios que precisam ser mencionados.

Nas primeiras trocas (item 01 à item 06 - Fig. 15) do primeiro handshake de comunicação, as primitivas se encontram desprotegidas, e portanto, nesse momento, são passíveis de exploração por esses dois ataques. No entanto, o ESMP possui o pré-requisito de que o DTSA deve, necessariamente, possuir um certificado válido emitido por uma das autoridades certificadoras aceitas pelo protocolo. Dessa forma, o atacante para realizar um ataque man-in-the-middle e/ou de reflexão tem que possuir um certificado emitido por terceiros. Essa ameaça, apesar de existir, não é fácil de ser materializada por um eventual atacante. O TCP, que é um dos protocolos mais utilizados na Internet, também possui semelhante ameaça.

No entanto, a partir da etapa 07 começa-se a construção do estado de sessão, das trocas de informações compartilhadas e do processo de autenticação mútua. As próximas primitivas, tanto do plano de controle quanto do plano de dados, são verificadas quanto à integridade e à autenticidade pelas entidades comunicantes. Os mecanismos utilizados na abordagem híbrida são cálculos e verificações de hash, passando como parâmetros informações de autenticação, como o título do contexto de comunicação (workspace), o título das entidades remetentes, as chaves compartilhadas, etc.

Esses recursos do ESMP não solucionam completamente esses ataques, porém, conseguem atenuá-los de maneira bem concisa.

Tabela 7 – Mecanismos de mitigação de ataques do ESMP na ETArch

Metas de Segurança	Ataques de Segurança	Mitigação
Confidencialidade	Espionagem (Snooping)	v
	Análise de tráfego (Traffic analysis)	v
Integridade	Modificação (Modification)	v
	Repúdio (Repudiation)	x
Disponibilidade	Negação de serviço (Denial of service) DoS	v
Autenticação	Man-in-the-middle	v
	Ataque por reflexão (reflection attack)	v
	Disfarce (Masquerading)	v
	Repetição (Replaying)	v
Segurança multicast	Apenas os ataques mitigados pelo ESMP na ETArch	v

Adaptada de (KHONDOKER et al., 2014).

5.2.4.7 Mascaramento/disfarce (masquerading)

Este ataque é mitigado através de uma assinatura digital, ou através de qualquer mecanismo que autentique mutuamente às partes envolvidas. Esse recurso de autenticação mútua já foi explanado nessa seção. Consegue-se essa contramedida através de operações de hash com parâmetros compartilhados entre as partes. O processo de verificação e do cálculo de hash pode ser visto na Fig. 17 e a execução da operação nas primitivas transmitidas na rede pode ser vista na Fig. 18.

Esse recurso do ESMP de autenticar mutuamente às partes envolvidas no processo de comunicação atenua o ataque de mascaramento, tanto no plano de dados quanto no plano de controle.

5.2.4.8 Ataque de repetição (replaying attack)

O ataque de repetição é mitigado pelo ESMP devido ao mecanismo que permite verificar a sequenciação das primitivas que estão vinculadas ao estado de sessão (no caso

das primitivas de controle) e ao estado de conexão (no caso das primitivas de dados). Esse recurso evita com que as entidades envolvidas na comunicação aceitem primitivas repetidas como sendo legítimas.

5.2.5 Segurança multicast na ETArch

Quanto à segurança multicast, a ETArch oferece suporte à essa comunicação, e todos os mecanismos de segurança (Subseção 5.2.3) que atenuam as ameaças (Subseção 5.2.2) que comprometem às metas do ESMP (Subseção 5.2.1) são executados ou efetivados em workspaces de controle que possuem apenas duas entidades comunicantes (entidade/DTSA), mas também em workspaces de dados que possuem um contexto de comunicação multicast. A exigência de workspaces "peer-to-peer" na comunicação de controle é uma exigência do ESMP, não uma limitação da ETArch. No ESMP, cada entidade possui uma sessão exclusiva com seu DTSA, e uma medida de segurança pertinente é que essas primitivas de controle trocadas entre essas duas entidades não cheguem em entidades desconhecidas; que podem ser maliciosas.

A Tab. 7 possui um resumo dos mecanismos de mitigação de ataques por metas de segurança do protocolo ESMP na arquitetura ETArch. Essa tabela se junta às informações de outros protocolos e arquiteturas descritas no decorrer desse trabalho para a realização de uma comparação final que será apresentada na sessão 5.3.

5.3 Avaliação dos resultados

A metodologia para análise e avaliação do ESMP utilizada é uma abordagem híbrida (KLOTI; KOTRONIS; SMITH, 2013), que combina modelagem do próprio sistema (modelagem dos componentes do sistema) (HERNAN et al., 2006) e modelagem de ataques de um sistema (SAINI; DUAN; PARUCHURI, 2008).

A técnica é simples: descreve as metas de segurança; analisa os componentes do sistema (no nosso caso, fluxo de dados) e modela seus recursos; modela as ameaças iminentes do sistema; e, por fim, verifica se os recursos do sistema conseguem mitigar as ameaças modeladas. O resultado desse método de análise e verificação do ESMP na ETArch pode ser visto na Tab. 7.

As ameaças mitigadas pelo protocolo ESMP na ETArch são: ataque de espionagem (Snooping); análise de tráfego (analysis traffic); modificação de mensagens (modification attack); denial of service (DoS); main-in-the-middle; ataque de reflexão (reflection attack); ataque de disfarce/mascaramento (masquerading) e ataque de repetição (replaying attack). Tal como está sinalizado na tabela, todos esses ataques/ameaças são mitigados em um contexto de comunicação multicast.

No caso do repúdio, o ESMP não consegue mitigar essa ameaça; ou seja, o ESMP na ETArch não consegue afirmar que a transação foi feita por determinada entidade em

Tabela 8 – Comparação do ESMP com outros protocolos e arquiteturas

MF - MobilityFirst**ESQ** - ETArch Status Quo**ESMP** - ESMP na ETArch

Metas de Segurança	Ataques	TLS	IPSec	MF	ESQ	ESMP
Confidencialidade	Espionagem	v	v	v	x	v
	Análise de tráfego	x	v	x	v	v
Integridade	Modificação	v	v	v	x	v
	Repúdio	x	v	v	x	x
Disponibilidade	Negação de serviço DoS	x	x	v	v	v
Autenticação	Man-in-the-middle	v	v	v	x	v
	Reflexão	v	v	v	x	v
	Mascaramento	v	v	v	x	v
	Repetição	v	v	x	x	v
Segurança multicast	Apenas os ataques mitigados	x	x	x	v	v

Adaptada de (KHONDOKER et al., 2014).

um contexto de comunicação multicast no plano de dados. Para isso, é necessário uma abordagem de comunicação que utilize, por exemplo, assinatura digital. A especificação do ESMP prevê essa necessidade através do gerenciamento de chaves simétricas a partir de encriptação assimétrica na Seção 4.2; porém, esse método de gerenciamento não foi descrito nesse trabalho. A evolução dessa especificação pode mitigar essa ameaça de repúdio no futuro.

Por fim, a Tab. 8 mostra a comparação do ESMP na ETArch com todas as outras arquiteturas ou protocolos descritas nesse trabalho. A comparação é realizada com o objetivo de apresentar as contribuições do ESMP para a arquitetura ETArch.

O ESMP na ETArch se mostra à frente, em questões de segurança, de arquiteturas de Internet do Futuro que já são maduras no mercado, como por exemplo, o MobilityFirst.

No caso do IPSec, SSL/TLS e MobilityFirst, eles não apresentam mecanismos de segurança para comunicação multicast, sendo que esse é um dos objetivos fulcrais do ESMP por conta das demandas das aplicações atuais.

A evolução do ESMP na ETArch quando comparado com a ETArch (status quo) é significativa. A análise de tráfego e o DoS mitigados pela ETArch (status quo) não tem relação com os mecanismos de segurança do SBB SM dessa arquitetura. Nos dois casos, a mitigação acontece devido à recursos da própria ETArch (status quo). Devido a esse fato, parte-se do princípio que a ETArch (status quo) não possui nenhum mecanismo de segurança capaz de mitigar qualquer uma das ameaças modeladas pelo método de análise e avaliação.

Conclusão

O objetivo geral deste trabalho é criar uma especificação de um protocolo de segurança *multicast* para uma arquitetura de Internet do Futuro. O protocolo denominou-se *Entity Security Multicast Protocol* (ESMP) e tem como função principal a transformação de um ambiente de comunicação desprovido de contramedidas de segurança em uma rede confiável, onde entidades possam confiar uma nas outras a fim de realizar uma comunicação segura. Desse modo, a hipótese de segurança do trabalho não destoa dos aspectos conceituais de rede definida por software, nem da ETArch, e nem sequer das funções que a segurança assume na Internet.

O objetivo geral do trabalho é alcançado paulatinamente, na medida que cada objetivo específico da seção 1.2 é satisfeito.

O primeiro objetivo específico da seção 1.2 foi resolvido no decorrer do Cap. 4. A especificação de autenticação, confidencialidade, integridade e disponibilidade se dissolve nas operações de gerenciamento de chaves (seção 4.2), multiplexação/demultiplexação (subseção 4.3.2), *handshakes* (subseção 4.3.3), cálculo e verificação de *hash* (seção 4.4). Parte dessa dissolução encontra-se também na especificação de como as primitivas de dados são operadas pelo ESMP (seção 4.4). Como dito anteriormente, esse primeiro objetivo específico está espalhado pelas soluções da proposta descrita no Cap. 4, e a razão disso é que a especificação desses serviços de segurança passa pela criação de vários mecanismos de segurança que são necessários para desenvolvê-los.

Nota-se, na seção 1.2, que o primeiro objetivo está destrinchado em vários requisitos que são de extrema importância para o cumprimento das metas do ESMP.

O primeiro requisito é que os serviços de segurança forneçam suporte às comunicações *multicast*. Esse contexto de comunicação é intrínseco da arquitetura ETArch, e portanto, todos os serviços de segurança especificados pelo ESMP se adéquam à essa espécie de comunicação.

O segundo requisito é que os serviços de segurança sejam transparentes à camada de aplicação. No decorrer do Cap. 4, mencionamos que a entidade (aplicação) aciona os serviços de segurança ao setar o identificador da política de segurança nos campos "*requi-*

rements" ou *"capabilities"* (dependendo da primitiva). Esse requisito é alcançado por uma característica intrínseca da própria arquitetura, que fornece uma aproximação semântica entre as camadas. Nesse contexto, os serviços/mecanismos de segurança são executados de forma transparente para a aplicação (entidade comunicante).

O terceiro requisito diz respeito à oferecer diferentes serviços/mecanismos de segurança para diferentes entidades (sensores, aplicações de banco, etc.). Esse requisito é resolvido com a flexibilidade da política de segurança (seção 4.3.1), que permite que cada aplicação requirite serviços/mecanismos de segurança de acordo com suas necessidades.

O quarto requisito relaciona-se com a sintetização de número de chaves distribuídas na comunicação *multicast* quando comparada com a comunicação *peer-to-peer*. A introdução do Cap. 4 resolve essa questão, demonstrando que enquanto a comunicação *peer-to-peer* do TCP/IP tem um crescimento exponencial de chaves em relação ao número de entidades participantes da comunicação, o modo de distribuição de chaves do ESMP possui um crescimento linear. Os gráficos 11(a)(b) demonstram essa afirmação.

O quinto requisito é que a especificação do ESMP na ETArch eleva o nível de segurança da arquitetura em relação à ETArch (status quo). Em 3.10 é analisada a segurança dessa arquitetura no seu estado atual, e na seção 5.3 é demonstrada a evolução da ETArch quando utilizada com o protocolo ESMP.

Dito isso, encerra-se a discussão do primeiro objetivo específico, e chega-se à conclusão que o trabalho conseguiu cumprir as metas esperadas.

O segundo objetivo específico é resolvido pela seção 5.2. Utiliza-se um método de análise e avaliação híbrido em relação às técnicas de modelagem de ataque e modelagem de sistema. Esse método se mostrou eficaz na amostragem das vulnerabilidades do sistema e da resistência dos mecanismos de segurança em relação aos ataques/ameaças iminentes em uma rede de comunicação. Dessa forma, conseguiu-se demonstrar se os mecanismos/serviços de segurança do ESMP são suficientes para cumprir as metas do protocolo. A Tab. 7 exibe os resultados obtidos quanto à capacidade que os mecanismos/serviços de segurança do ESMP têm de mitigar ameaças/ataques conhecidos.

Dito isso, encerra-se a discussão do segundo objetivo específico, e chega-se à conclusão que o trabalho conseguiu cumprir as metas esperadas.

O terceiro requisito é fazer uma avaliação comparativa entre os resultados obtidos desse trabalho e algumas arquiteturas e protocolos de segurança que já são maduros no mercado. Foi escolhido para comparação os dois principais protocolos da arquitetura TCP/IP (IPsec/TLS), uma arquitetura de Internet do Futuro (MobilityFirst) e a própria arquitetura que serviu de protótipo do ESMP (ETArch (status quo)). O resultado dessa avaliação comparativa se encontra na Tab. 8.

Dito isso, encerra-se a discussão do terceiro objetivo específico, e chega-se à conclusão que o trabalho conseguiu cumprir as metas esperadas.

Com todos os objetivos específicos alcançados e as demonstrações realizadas através

do método de avaliação proposto, entende-se como cumprido o objetivo geral, que é a especificação de um ambiente de comunicação *multicast* seguro. Uma vez atingido os objetivos específicos e geral, as próximas seções tratam de destacar as principais contribuições deste trabalho e apresentar os trabalhos futuros para melhoria da hipótese atual, além daqueles que podem ser gerados a partir deste.

6.1 Principais Contribuições

O trabalho apresenta uma proposta de segurança para ser utilizada em ambientes SDN. A contribuição geral é a especificação de um novo protocolo de segurança *multicast* (ESMP), que visa garantir um ambiente de comunicação seguro dentro de um contexto *multicasting*. Essa especificação de segurança utiliza a ETArch como protótipo de arquitetura de Internet do Futuro, arquitetura esta que até então era desprovida de mecanismos/serviços de segurança eficientes quanto às ameaças/ataques iminentes na rede. A evolução dessa arquitetura foi demonstrada e faz parte das contribuições gerais da proposta desse trabalho.

A arquitetura ETArch oferece *multicasting* de forma intrínseca, o que auxiliou o ESMP em prover segurança nessa espécie de comunicação. Prover segurança *multicasting* não é uma tarefa fácil, nem para a arquitetura legada nem para arquiteturas de Internet do Futuro. A corroboração desse fato se encontra na seção 2.5.1; percebe-se claramente que o MobilityFirst possui um método de distribuição de primitivas para grupos, utilizando a reprodução na origem (*unicast* múltiplo). Faz-se essa uma característica especial dessa especificação e que provê mais uma contribuição para arquiteturas que queiram se aventurar em segurança para grupos de entidades.

A arquitetura ETArch oferece outra característica intrínseca muito importante: a aproximação semântica entre a camada de aplicação e a camada intermediária. Essa característica da arquitetura auxiliou o ESMP a distribuir serviços/mecanismos de segurança de acordo com às necessidades da aplicação (entidade comunicante). Essa possibilidade garante segurança para as aplicações atuais e futuras. Em um provável cenário de IoT, esse recurso é muito interessante, porque os diversos dispositivos da internet das coisas possuem capacidades de hardware diversos. Dessa forma, a política de segurança pode divergir para vários cenários diferentes da vida real.

Outra característica do ESMP que pode servir como referência para futuros protocolos de segurança é a capacidade de sintetizar o número de segredos compartilhados na operação de disseminação de chaves para as entidades comunicantes de rede que pertencem a um grupo *multicast*. Enquanto o crescimento de chaves em um conexão *peer-to-peer* é exponencial, o crescimento do ESMP é linear. Os gráficos 11(a)(b) mostram que enquanto é preciso de quase meio milhão de chaves para gerenciar um grupo *multicast* através de uma ALM, no caso do ESMP, é preciso de pouco mais de 1000 chaves. Esse fato ga-

rante viabilidade de grupos grandes na ETArch e reduz a complexidade de gerenciamento desses grupos; o que ainda é um dos grandes gargalos da questão de se ter comunicação *multicast* na arquitetura legada e em arquiteturas de Internet do Futuro. Essa capacidade de diminuir o número de chaves compartilhadas é mais uma característica que se junta às contribuições desse trabalho.

No mais, é importante salientar que qualquer projeto que utilize os conceitos de SDN, ou seja, que separa o plano de controle do plano de dados pode utilizar essa especificação como referência para um futuro projeto de arquitetura que se preocupa com os requisitos de segurança. Esses requisitos são fundamentais para que qualquer arquitetura seja implantada em um ambiente de produção.

6.2 Trabalhos Futuros

A especificação inicial do ESMP trata de várias questões que são importantes no âmbito de discussões relativas à segurança de arquiteturas de Internet. Contudo, há ainda espaço para melhorias dessa especificação no que envolve várias questões, quais sejam:

- ❑ a especificação do ESMP no Cap. 4 utiliza como base o gerenciamento de chaves híbrido (seção 4.2). É importante que se especifique as operações, o fluxo das primitivas, as funcionalidades, etc. dos outros métodos de gerenciamento propostos nessa mesma seção. Aliás, é necessário que se pense em outras formas de distribuição que cubra às necessidades da maioria dos aplicativos; já que esse trabalho de segurança estará sempre em aberto e é flexível para modificações no decorrer do tempo;
- ❑ este trabalho propôs serviços/mecanismos de segurança para um único domínio de DTSA. É importante que essa especificação se estenda a fim de buscar segurança em nível global. Para esse fim, a especificação tem que prever as operações do ESMP nas comunicações de controle da ETArch em ambientes inter-domínios;
- ❑ este trabalho não menciona a possibilidade de se ter entidades inteligentes distribuídas na rede com a finalidade de detectar padrões de fluxos maliciosos, a fim de que o ESMP possa com essas informações corrigir o problema encontrado. Esse tipo de mecanismo poderia auxiliar no combate aos ataques DoS, que dentre todos os ataques modelados nesse trabalho, talvez seja o mais difícil de solucionar;
- ❑ quanto mais frequentemente as chaves secretas compartilhadas forem trocadas, mais seguras elas são, pois o oponente tem menos texto cifrado para trabalhar; outra ameaça iminente é que uma entidade comunicante pode se tornar um futuro atacante quando ela se desconectar do grupo. Essas trocas de chaves periódicas não foram previstas pela especificação;

- ❑ acrescentar na especificação primitivas de alerta contra eventuais erros de comunicação do protocolo ou contra eventuais ataques que alguma entidade esteja sofrendo dentro do contexto de comunicação *multicast*; o ideal é que para cada alerta houvesse uma contramedida do protocolo para solucionar o problema detectado;
- ❑ implantação dessa especificação na ETArch. Seria importante testar essa especificação experimentalmente, através de aplicações de chat, aplicações de vídeo, etc.;
- ❑ pós implantação do ESMP, desenvolver novas aplicações para testar os mecanismos/serviços experimentalmente. Um exemplo que poderia ser dado é uma aplicação que repete incansavelmente a transmissão de uma primitiva da rede. Nesse contexto, seria interessante medir o tempo que a ETArch gasta na modificação dessa rota (no caso de congestionamento dos NEs) ou como ele age na questão do controle de banda. Outra métrica interessante seria a discussão de decaimento do processamento das entidades que estão recebendo esses pacotes. Apesar dessas entidades enxergarem que se trata de uma repetição, quando são muitos pacotes, analisar o seus comportamentos é algo relevante.

6.3 Contribuições em Produção Bibliográfica

Ao longo do trabalho, foram explorados temas relacionados às redes de computadores e telecomunicações. O artigo *Entity Title Architecture (ETArch): Future Internet multicast Architecture Based on Quality of Service, Mobility, and Security* foi submetido ao periódico IEEE Communications Magazine.

Referências

- AGGARWAL, R. Kamite, Y., Fang, L., Rekhter, Y., and C. Kodebo-niya, "Multicast in Virtual Private LAN Service (VPLS). [S.l.], 2014.
- AGGARWAL, R.; KAMITE, Y.; FANG, L. Multicast in vpls. **draft-raggarwa-l2vpn-vpls-mcast-01. txt**, Work in progress. Sunil Khandekar Alcatel North America, v. 701, 2007.
- BELLARE, M.; CANETTI, R.; KRAWCZYK, H. Keying hash functions for message authentication. In: SPRINGER. **Annual International Cryptology Conference**. 1996. p. 1–15. Disponível em: <https://doi.org/10.1007/3-540-68697-5_1>.
- _____. Message authentication using hash functions: The hmac construction. **RSA Laboratories' CryptoBytes**, v. 2, n. 1, p. 12–15, 1996.
- BR, N. de Informação e Coordenação do P. **Estatísticas dos Incidentes Reportados ao CERT.br**. 2018. <https://www.cert.br/stats/incidentes/>. Disponível em: <<https://www.cert.br/stats/incidentes/>>. Acesso em: 21 nov. 2018.
- CIAMPA, M. **Security Plus guide to network security fundamentals, 3th edition**. [S.l.]: Cengage Learning, 2009.
- CICIC, T.; BRYHNI, H. Multicast-unicast reflector. In: **proceedings of Protocols for Multimedia Communications (PROMS) conference**. [S.l.: s.n.], 2000. p. 60–69.
- CISCO. **Cisco Visual Networking Index: Forecast and Trends, 2017–2022**. 2017. Disponível em: <<https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-741490.pdf>>. Acesso em: 28 jan. 2019.
- COX, J. H. et al. Advancing Software-Defined Networks: A Survey. **IEEE Access**, v. 5, p. 25487–25526, 2017. Disponível em: <<https://doi.org/10.1109/ACCESS.2017.2762291>>.
- DAVIES, E.; KRISHNAN, S.; SAVOLA, P. **IPv6 transition/co-existence security considerations**. [S.l.], 2007.
- DEERING, S. E.; CHERITON, D. R. Multicast routing in datagram internetworks and extended lans. **ACM Transactions on Computer Systems (TOCS)**, ACM, v. 8, n. 2, p. 85–110, 1990.

- DONG, L.; CHEN, K. **Cryptographic protocol: security analysis based on trusted freshness**. Springer Science & Business Media, 2012. Disponível em: <<https://doi.org/10.1007/978-3-642-24073-7>>.
- DOULIGERIS, C.; SERPANOS, D. N. **Network security: current status and future directions**. John Wiley & Sons, 2007. Disponível em: <<https://doi.org/10.1002/0470099747>>.
- DULANEY, E. **CompTIA Security + Study Guide: SY0-201, 4th edition**. [S.l.]: John Wiley & Sons, 2009.
- EL-SAYED, A.; ROCA, V.; MATHY, L. A survey of proposals for an alternative group communication service. **IEEE network**, IEEE, v. 17, n. 1, p. 46–51, 2003.
- FEMMINELLA, M. et al. Implementation and performance analysis of advanced it services based on open source jain sleet. In: IEEE. **Local Computer Networks (LCN), 2011 IEEE 36th Conference on**. 2011. p. 746–753. Disponível em: <<https://doi.org/10.1109/LCN.2011.6115544>>.
- FIELDING, R. et al. Rfc 2616. **Hypertext Transfer Protocol–HTTP/1.1**, v. 2, n. 1, p. 2–2, 1999.
- FIPS, P. 199. **Standards for Security Categorization of Federal Information and Information Systems**, v. 2, 2004.
- FOROUZAN, B. A.; FEGAN, S. C. **Protocolo TCP/IP-3**. [S.l.]: AMGH Editora, 2009.
- FOUNDATION, O. N. **The ONF is an Operator Led Consortium Transforming Networks into Agile Platforms for Service Delivery**. 2011. Disponível em: <<https://www.opennetworking.org/>>. Acesso em: 03 fev. 2019.
- _____. **Software-Defined Networking: The New Norm for Networks**. 2012. Disponível em: <<https://www.opennetworking.org/>>. Acesso em: 27 fev. 2019.
- _____. **This wiki documents the current development version of ONOS (master)**. 2014. Disponível em: <<https://wiki.onosproject.org/display/ONOS/Wiki+Home/#app-switcher>>. Acesso em: 03 fev. 2019.
- FRANKEL, S.; KRISHNAN, S. Ip security (ipsec) and internet key exchange (ike) document roadmap. rfc 6071 (informational). IETF, n. 6071, fev. 2011. Disponível em: <<https://tools.ietf.org/html/rfc6071>>.
- GONÇALVES, M. A. et al. Etarch: projeto e desenvolvimento de uma arquitetura para o modelo de título com foco na agregação de tráfego multicast. Universidade Federal de Uberlândia, 2014.
- GROUP, I. . W. et al. Ieee standard for local and metropolitan area networks-part 21: Media independent handover. **IEEE Std 802.21-2008**, p. c1–c301, 2009.
- GUBBI, J. et al. Internet of things (iot): A vision, architectural elements, and future directions. **Future generation computer systems**, Elsevier, v. 29, n. 7, p. 1645–1660, 2013.

- GUIMARAES, C.; CORUJO, D.; AGUIAR, R. **ATNOG. EDOBRA (Extending and Deploying OFELIA in BRAzil)**. 2012. Disponível em: <<http://atnog.av.it.pt/content/ofelia-edobra>>. Acesso em: 12 jan. 2019.
- GUTTMAN, B.; ROBACK, E. A. **An introduction to computer security: the NIST handbook**. DIANE Publishing, 1995. Disponível em: <<https://doi.org/10.6028/NIST.SP.800-12>>.
- HAMID, J.; GIANLUIGI, M.; LILBURN, W. D. **Handbook of electronic security and digital forensics**. [S.l.]: World Scientific, 2010.
- HAN, D. et al. Xia: Efficient support for evolvable internetworking. In: USENIX. [S.l.], 2012.
- HARDJONO, T.; TSUDIK, G. Ip multicast security: Issues and directions. In: SPRINGER. **Annales des télécommunications**. [S.l.], 2000. v. 55, n. 7-8, p. 324–340.
- HERNAN, S. et al. **Uncover Security Design Flaws Using The STRIDE Approach**. 2006. Disponível em: <<https://adam.shostack.org/uncover.html>>. Acesso em: 28 jan. 2019.
- HINDEN, R.; DEERING, S. **IP version 6 addressing architecture**. [S.l.], 2006.
- HOSSEINI, M. et al. A survey of application-layer multicast protocols. **IEEE Communications Surveys and Tutorials**, v. 9, n. 1-4, p. 58–74, 2007. Disponível em: <<https://doi.org/10.1109/COMST.2007.4317616>>.
- ITU-T, D. I.-T. Recommendation x. 509 version4. **ITU-T Publications**, v. 5, 2001.
- JACOBSON, V. et al. Networking named content. In: ACM. **Proceedings of the 5th international conference on Emerging networking experiments and technologies**. 2009. p. 1–12. Disponível em: <<https://doi.org/10.1145/1658939.1658941>>.
- JIN, A. **OpenFlow Switch with Hardware Flow Table**. 2016. Disponível em: <<http://learn.linksprite.com/project/openflow-switch-with-hardware-flow-table/>>. Acesso em: 01 mar. 2019.
- KERNER, S. **OpenFlow Protocol 1.3.0 Approved**. 2012. [Http://www.enterprisenetworkingplanet.com/nethub/openflow-protocol-1.3.0-approved.html](http://www.enterprisenetworkingplanet.com/nethub/openflow-protocol-1.3.0-approved.html). Disponível em: <<http://www.enterprisenetworkingplanet.com/nethub/openflow-protocol-1.3.0-approved.html>>. Acesso em: 17 jun. 2012.
- KHONDOKER, R. et al. Security of selected future internet architectures: A survey. In: IEEE. **2014 Eighth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS)**. 2014. p. 433–440. Disponível em: <<https://doi.org/10.1109/IMIS.2014.62>>.
- KLOTI, R.; KOTRONIS, V.; SMITH, P. Openflow: A security analysis. In: IEEE. **Network Protocols (ICNP), 2013 21st IEEE International Conference on**. 2013. p. 1–6. Disponível em: <<https://doi.org/10.1109/ICNP.2013.6733671>>.

KONTESIDOU, G.; ZARIFIS, K. **Openflow Virtual Networking: A Flow-Based Network Virtualization Architecture**. Dissertação (Mestrado) — Royal Institute of Technology, Stockholm, Nov 2009.

KREUTZ, D. et al. Software-defined networking: A comprehensive survey. **Proceedings of the IEEE**, Ieee, v. 103, n. 1, p. 14–76, 2015. Disponível em: <<https://doi.org/10.1109/JPROC.2014.2371999>>.

KUROSE, J. F.; ROSS, K. W. **Redes de Computadores e a Internet: uma abordagem top-down**. 6ª edição. [S.l.]: São Paulo, Editora Pearson, 2013.

LARA, A.; KOLASANI, A.; RAMAMURTHY, B. Network innovation using openflow: A survey. **IEEE communications surveys & tutorials**, IEEE, v. 16, n. 1, p. 493–512, 2014.

LEINER, B. M. et al. A brief history of the internet. **ACM SIGCOMM Computer Communication Review**, ACM, v. 39, n. 5, p. 22–31, 2009. Disponível em: <<https://doi.org/10.1145/1629607.1629613>>.

LEMA, J. C. **Evolving Future Internet clean-slate Entity Title Architecture with quality-oriented control-plane extensions**. Dissertação (Mestrado) — Universidade Federal do Rio Grande do Norte, 2014.

MARILL, T.; ROBERTS, L. G. Toward a cooperative network of time-shared computers. In: ACM. **Proceedings of the November 7-10, 1966, fall joint computer conference**. 1966. p. 425–431. Disponível em: <<https://doi.org/10.1145/1464291.1464336>>.

MARTÍNEZ, A. et al. Toward a new addressing scheme for a service-centric internet. In: IEEE. **Communications (ICC), 2012 IEEE International Conference on**. 2012. p. 6463–6467. Disponível em: <<https://doi.org/10.1109/ICC.2012.6364737>>.

MCKEOWN, N. et al. Openflow: enabling innovation in campus networks. **ACM SIGCOMM Computer Communication Review**, ACM, v. 38, n. 2, p. 69–74, 2008. Disponível em: <<https://doi.org/10.1145/1355734.1355746>>.

MELO, P. H. A. D. d. et al. Mecanismos de autenticação e controle de acesso para uma arquitetura de internet do futuro. Universidade Federal de Uberlândia, 2017.

_____. Mecanismos de autenticação e controle de acesso para uma arquitetura de internet do futuro. Universidade Federal de Uberlândia, 2017.

MENDONÇA, J. **JAIN SLEE User Guide - Community Documentation**. 2009. Disponível em: <http://docs.jboss.org/mobicents/jain-slee/2.7.0.FINAL/container/user-guide/en-US/html_single/>. Acesso em: 13 jan. 2019.

MIRKOVIC, J. et al. Internet denial of service: Attack and defense mechanisms (radia perlman computer networking and security). Prentice Hall PTR, 2004.

_____. Internet denial of service: Attack and defense mechanisms (radia perlman computer networking and security). Prentice Hall PTR, 2005.

- MOORE, G. E. Cramming more components onto integrated circuits, reprinted from electronics, volume 38, number 8, april 19, 1965, pp. 114 ff. **IEEE Solid-State Circuits Society Newsletter**, IEEE, v. 11, n. 3, p. 33–35, 2006. Disponível em: <<https://doi.org/10.1109/N-SSC.2006.4785860>>.
- NAYLOR, D. et al. Xia: architecting a more trustworthy and evolvable internet. **ACM SIGCOMM Computer Communication Review**, ACM, v. 44, n. 3, p. 50–57, 2014. Disponível em: <<https://doi.org/10.1145/2656877.2656885>>.
- NETO, N. V. d. S. et al. Greenow: um algoritmo de roteamento orientado a workspace para uma arquitetura de internet do futuro. Universidade Federal de Uberlândia, 2016.
- _____. Greenow: um algoritmo de roteamento orientado a workspace para uma arquitetura de internet do futuro. Universidade Federal de Uberlândia, 2016.
- NSF. **FOUNDATION, N. S. NSF Future Internet Architecture Project. 2010.** 2010. Disponível em: <<http://www.nets-fia.net/>>. Acesso em: 22 jan. 2019.
- PEREIRA, J. H. d. S. **Modelo de título para a próxima geração de Internet.** Tese (Doutorado) — Universidade de São Paulo, 2012.
- REC, I. X. 800 security architecture for open systems interconnection for ccitt applications. **ITU-T (CCITT) Recommendation**, 1991.
- REXFORD, J.; DOVROLIS, C. Future internet architecture: clean-slate versus evolutionary research. **Communications of the ACM**, ACM, v. 53, n. 9, p. 36–40, 2010. Disponível em: <<https://doi.org/10.1145/1810891.1810906>>.
- ROBERTS, J. The clean-slate approach to future internet design: a survey of research initiatives. **annals of telecommunications-Annales des télécommunications**, Springer, v. 64, n. 5-6, p. 271–276, 2009.
- ROMDHANI, I. et al. Ip mobile multicast: Challenges and solutions. **IEEE Communications Surveys & Tutorials**, IEEE, v. 6, n. 1, 2004.
- ROSEN, E.; REKHTER, Y. Bgp/mps ip virtual private networks (vpns) rfc 4364. **The Internet Engineering Task Force (IETF®)**, 2006.
- ROSEN, E.; VISWANATHAN, A.; CALLON, R. Multiprotocol label switching architecture,” rfc 3031. Citeseer, 2001.
- ROSENBERG, J. et al. Rfc 3261: Sip session initiation protocol, june 2002. **Internet Engineering Task Force**, p. 1–269, 2004.
- RUI, A. et al. **ODTONE - An Open Source Multi-Platform - Implementation of the IEEE 802.21 Protocol.** 2011. Disponível em: <<http://atnog.github.io/ODTONE/>>. Acesso em: 13 jan. 2019.
- SAINI, V.; DUAN, Q.; PARUCHURI, V. Threat modeling using attack trees. **Journal of Computing Sciences in Colleges**, Consortium for Computing Sciences in Colleges, v. 23, n. 4, p. 124–131, 2008.
- SHIREY, R. Rfc 4949–internet security glossary.[sl]: Version, 2007. **Citado na**, IETF, n. 4949, p. 32, ago. 2007. Disponível em: <<https://tools.ietf.org/html/rfc4949>>.

SILVA, F. et al. Enabling network mobility by using iee 802.21 integrated with the entity title architecture. In: **Anais do IV Workshop de Pesquisa Experimental da Internet do Futuro-31o Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos**. [S.l.: s.n.], 2013. p. 24–34.

SILVA, F. d. O. **Endereçamento por título: uma forma de encaminhamento multicast para a próxima geração de redes de computadores**. Tese (Doutorado) — Universidade de São Paulo, 2013.

SLEE, M. J. **Introduction to Mobicents JAIN SLEE**. 2009. Disponível em: <<http://docs.jboss.org/mobicents/jain-slee/2.6.0.FINAL/container/user-guide/en-US/html/introduction.html>>. Acesso em: 13 jan. 2019.

STALLINGS, W. **Criptografia e segurança de redes**. 6ª edição. [S.l.]: São Paulo, Editora Pearson, 2014.

SWITCH, B. **Developing floodlight modules. Floodlight OpenFlow controller**. 2012. Disponível em: <<http://www.enterprisenetworkingplanet.com/nethub/openflow-protocol-1.3.0-approved.html>>. Acesso em: 13 jan. 2019.

TANENBAUM, A. S. et al. Computer networks, 4-th edition. ed: **Prentice Hall**, 2003.

TROUVA, E. et al. Is the internet an unfinished demo? meet rina! In: **TERENA Networking Conference**. [S.l.: s.n.], 2011. p. 12.

VENKATARAMANI, A. et al. Mobilityfirst: a mobility-centric and trustworthy internet architecture. **ACM SIGCOMM Computer Communication Review**, ACM, v. 44, n. 3, p. 74–80, 2014. Disponível em: <<https://doi.org/10.1145/2656877.2656888>>.

VRIJERS, S. et al. Experimental evaluation of a recursive internetwork architecture prototype. In: IEEE. **Global Communications Conference (GLOBECOM), 2014 IEEE**. 2014. p. 2017–2022. Disponível em: <<https://doi.org/10.1109/GLOCOM.2014.7037104>>.

XAVIER, R. et al. Resource allocation algorithms for multicast streaming in elastic cloud-based media collaboration services. In: IEEE. **Cloud Computing (CLOUD), 2016 IEEE 9th International Conference on**. 2016. p. 947–950. Disponível em: <<https://doi.org/10.1109/CLOUD.2016.0142>>.