DESIGN AND VALIDATION OF RECEIVER ACCESS CONTROL IN THE AUTOMATIC MULTICAST TUNNELING ENVIRONMENT

VEERA NAGASIVA TEJESWI MALLA

A THESIS
IN
THE DEPARTMENT
OF
COMPUTER SCIENCE AND SOFTWARE ENGINEERING

PRESENTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF MASTER OF COMPUTER SCIENCE
CONCORDIA UNIVERSITY
MONTRÉAL, QUÉBEC, CANADA

AUGUST 2014

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By : **Veera Nagasiva Tejeswi Malla**

Entitled : **Design and Validation of Receiver Access Control in the Automatic Multicast Tunneling Environment** and submitted in partial fulfilment of the requirements for the degree of

Master of Computer Science

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee :

_____ Chair
Dr. G. Butler

_____ Examiner
Dr. H. Harutyunyan

_____ Examiner
Dr. N. Narayanan

_____ Supervisor
Dr. J.W. Atwood

Approved by _____
Dr. V. Haarslev
Graduate Program Director

_____ 2014. _____
Dr. Amir Asif, Dean
Faculty of Engineering and Computer Science

ABSTRACT

Design and Validation of Receiver Access Control in the Automatic Multicast Tunneling

Veera Nagasiva Tejeswi Malla

Standard IP multicast offers scalable point-to-multipoint delivery, but no control over who may send and who may receive the data stream. Participant Access Control has been developed by Islam and Atwood, but only for multicast-enabled network regions. Automatic Multicast Tunneling has been developed by the Internet Engineering Task Force. It extends the range of multicast data distribution to unicast-only network regions, but provides no Participant Access Control.

We have designed the additional features that AMT must have, so that AMT has the necessary Participant Access Control at the receiver's end in the AMT environment. In addition, we have validated our design model using the AVISPA formal modeling tool, which confirms that the proposed design is secure.

Acknowledgments

I would like to express my deepest gratitude to my supervisor Prof. **John William Atwood**. He encouraged and guided me through his patience and enormous knowledge right from day one. His support helped me to understand and enhance my knowledge in the subject.

More than anyone else, I would like to thank my beloved parents, **Mr. Subbarayudu Malla** and **Mrs. Gowramma Marabadu.** It was their immeasurable love, support and encouragement that made this Master's degree possible. I would also like to thank my **esteemed** friends, sisters and their families for their love, support and confidence in my potentiality.

Veera Nagasiva Tejeswi Malla

Contents

List of Figures

List of Acronyms

AVISPA	Automated Validation of Internet Security Protocols and Applications
AAAS	Authentication, Authorization and Accounting Server
AAA	Authentication, Authorization and Accounting
AR	Access Router
AMT	Automatic Multicast Tunneling
EU	End User
EP	Enforcement Point
EAP	Extensible Authentication Protocol
EPQ	Enforcement Point in Querier
EPG	Enforcement Point in Gateway
EAP-FAST	Extensible Authentication Protocol-Flexible Authentication via Secure Tunneling
GSA	Group Security Association
GSAM	Group Security Association Management
GSPD	Group Security Policy Database

GPAD	Group Peer Authentication Database
HLPSL	High Level Protocol Specification Language
IKEv2	Internet Key Exchange version2
IETF	Internet Engineering Task Force
IGMP	Internet Group Management Protocol
IPsec	Internet Protocol Security
MSK	Master Session Key
MAC	Message Authentication Code
MLD	Multicast Listener Discovery
PANA	Protocol for Carrying Authentication for Network Access
PaC	PANA Client
PAA	PANA Authentication Agent
PAC	Participant Access Control
Q	Querier in Multicast Router
SA	Security Association
SAD	Security Association Database
SPD	Security Policy Database
PAD	Peer Authorization Database
SPI	Security Parameter Index
SIGMP	Secure Internet Group Management Protocol
UDP	User Datagram Protocol

Chapter 1

Introduction

Some applications require data to be delivered from a sender to multiple receivers. Examples of such applications include audio and video broadcasts, real-time delivery of stock quotes, and teleconferencing applications. A service where data is delivered from a sender to multiple receivers is called multipoint communication or **Multicast**. It provides an efficient way to support high bandwidth, one-to-many applications on a network. The IP-specific version of the multicast concept is called **IP Multicast**. In IP multicast, IGMP (Internet Group Management Protocol) is used to dynamically register individual hosts in a multicast group. Hosts identify group memberships by sending IGMP messages to their local multicast router, those routers listen to IGMP messages and periodically send out queries to discover which groups are active or inactive on a particular subnet [25]. One major problem in IP multicast is that even hosts without any permissions are able to join multicast groups, i.e., there is no

mechanism to prevent unauthorized users from accessing a multicast network. Consequently it became impossible for those service providers who tried to bill for multicast data usage. Content providers were not interested in paying extra money to transmit multicast streams to those unauthorized users. To overcome this problem and to make multicast more efficient, Participant Access Control (PAC) was introduced in [2]. PAC includes Sender Access Control [18], which authenticates and authorizes each sender and accounts for sender behavior by deploying AAA protocols and Receiver Access Control [19], which is a scalable, distributed and secure architecture where authorized end users can be authenticated before delivering any data, using Extensible Authentication Protocol (EAP) and the Protocol for Carrying Authentication for Network Access (PANA). Although PAC provides access control for IP multicast, yet it faces a few challenges. It needs all the components between the multicast source and the receiver(s) to support IP multicast technology, in other words hosts that are in a unicast-only-network cannot access the multicast transmissions. Unfortunately, many of the devices that could use IP multicast lack multicast connectivity to networks that carry traffic generated by multicast sources. The reasons for the lack of connectivity vary, but are primarily the result of service provider policies and network limitations.

To overcome this problem, a solution was proposed by the Network Group at the Internet Engineering Task Force (IETF) called **Automatic Multicast Tunneling (AMT)**. AMT is designed to provide a mechanism for a migration path from a unicast-enabled region to a fully multicast-enabled backbone without any explicit

tunnels between them. Without requiring any manual configuration, AMT allows the hosts to receive multicast traffic from the native multicast infrastructure. In AMT, multicast queries and reports are exchanged between a Gateway (close to the receiver) and a Relay (on the multicast enabled network) by encapsulating them in User Datagram Protocol (UDP) packets. Any IP multicast data packets that are pertinent to the receiver are also encapsulated by the Relay and sent to the Gateway for distribution to the receiver. The goal of AMT is to foster the deployment of native IP multicast by enabling a potentially large number of nodes to connect to an already-present multicast provider network. Though it has it own advantages, it provides no Receiver Access Control for multicast groups.

The problems with previous architectures are, IP multicast offers scalable point-to-multipoint delivery, but no Access Control in it. PAC offers Access Control, but it is limited to only native multicast environment. AMT extends multicast service to unicast region, but no Access Control. So, our goal is to combine all, i.e., in addition to the current features of AMT, we must add PAC features at receivers end.

As a solution we have proposed a design architecture that provides Receiver Access Control in AMT. The term Receiver Access Control used here means authentication, authorization and accounting (AAA) functionalities for receivers of a multicast group. In our architecture we implement Receiver Access Control features of IP multicast at the receiver's end in the AMT environment by considering all the security issues. We have also formally validated our model using the AVISPA tool.

In order to have a better understanding of the architecture, we will discuss more

about IP multicast and its challenges in Chapter 2. Chapter 3 explains about PAC. In Chapter 4 we will focus on AMT and its challenges. Chapter 5 explains Problem statement, goals and in Chapter 6 we will discuss about our model (proposed solution). Chapter 7 gives information about AVISPA tool and describes how we validate our model with it. Chapter 8 contains conclusion followed by future work.

Chapter 2

IP Multicast

2.1 Intoduction

There are classes of applications that require distribution of information to a defined set of users. IP multicast, an extension to IP, is required to properly address these communication needs. As the term implies, IP multicast has been developed to support efficient communication between a source and multiple remote destinations. IP multicast protocols and underlying technologies enable efficient distribution of data, voice, and video streams to a large population of users, ranging from hundreds to thousands to millions of users. One important design principle of IP multicast is to allow receiver-initiated attachment (joins) to information streams, thus supporting a distributed informatics model. A second important principle is the ability to support optimal pruning such that the distribution of the content is streamlined by pushing

replication as close to the receiver as possible. These principles enable bandwidth-efficient use of underlying network infrastructure [26].

2.2 List of Examples

Multicast applications include data casting, distribution of real-time financial data, entertainment digital television over an IP network, Internet radio, multipoint video conferencing, distance learning, streaming media applications, and corporate communications. Other applications include distributed interactive simulation, grid computing, and distributed video gaming.

2.2.1 Example: IPTV

As an example in the IPTV arena, with the current trend toward the delivery of High-Definition TV (HDTV) signals, each requiring data flows in the 12-Mbps range, and the consumers' desire for a large number of channels (200-300 being typical), there has to be an efficient mechanism of delivering a signal of 1-2 Gbps in aggregate to a large number of remote users. If a source had to deliver one Gbps of signal to, say, one million receivers by transmitting all of this bandwidth across the core network, it would require a petabyte-per-second network fabric, which is currently impossible.

On the contrary, if the source could send the 1 Gbps of traffic to, say, 50 remote distributions points (e.g., head ends), each of which then makes use of a local distribution network to reach 20,000 subscribers, the core network needs to support 50 Gbps

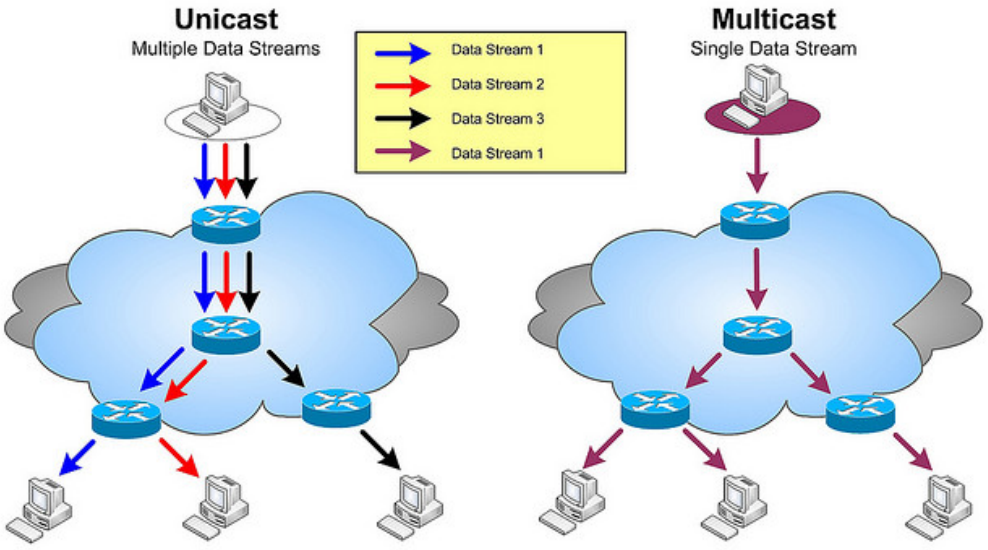

Figure 1: Multicast Example

only, which is possible with proper design. For these kind of reasons, IP multicast is seen as a bandwidth-conserving technology that optimizes traffic management by simultaneously delivering a stream of information to a large population of recipients, including corporate enterprise users and residential customers [26]. Figure 1 shows the difference between unicast and multicast.

2.3 Benefits of IP Multicast

2.3.1 Scalability

IP multicast technology enjoys intrinsic scalability, which is critical for many current applications. The server does not have to produce data packets for each end user. It just produces one packet and sends it out. The packet is then duplicated by

the network routers whenever it is needed. So, on the routing tree, the bandwidth is not wasted either. Moreover, the source is not aware of all the individual users session information any more. The end users are managed in a different way by new participants in the multicast distribution technology [30].

2.4 Multicast Protocols and Concepts

2.4.1 Internet Group Management Protocol (IGMP)

Definition

Multicast communication is based on the construct of a group of receivers (hosts) that have an interest in receiving a particular stream of information, be it voice, video, or data. Hosts that are desirous of receiving data intended for a particular group have to join the group using a group management protocol. Hosts should become explicit members of the group to receive the data stream, but such membership may be ephemeral and/or dynamic. Groups of IP hosts that have joined the group and wish to receive traffic sent to this specific group are identified by multicast addresses.

In IP multicast IGMP is used by host receivers to join or leave a multicast host group. It is used by IPv4-based receivers to report their IP multicast group membership to neighboring multicast routers, i.e., it defines the signaling communication occurring between receiving hosts and their local multicast router. Hosts establish group memberships by sending IGMP messages to their local multicast router. Multicast-enabled routers monitor for IGMP messages to maintain forwarding tables for the

various interfaces on the router. They also periodically send out queries to discover which groups are active or inactive on a given subnet [26].

IGMPv1

Host membership Report: : When a host wishes to join a multicast group, it sends an IGMP Host Membership Report message to the specific group address, regardless of whether there are already other hosts on its subnet that are host group members. Unlike a multicast router, a host does not keep track of the host group membership of other hosts on its subnet. Because a multicast router is listening in multicast promiscuous mode, it receives and processes IGMP Host Membership Report messages sent to any multicast address.

Host Membership Query: An IGMP v1 multicast router periodically sends an IGMP Host Membership Query message to 224.0.0.1 (the allhosts group) to refresh its knowledge of host members on the subnet. For each host group for which there are members on the subnet, one hosts group member responds with an IGMP Host Membership Report messages [39].

IGMPv2

Version 2 extends the functionality of IGMP while maintaining backward compatibility with IGMP v1. It adds two new message types, an IGMP v2 Host Membership Report and a Leave group message. It also adds a variation on the Host Membership Query called the Group-Specific Host Membership Query. Finally IGMPv2 defined

clearly how a multicast Querier (a router that send Queries) is elected if there are multiple multicast routers connected to a common network. In IGMPv1, all multicast routers were expected to send Queries. IGMPv2 stipulates that only the multicast router with the lowest IP address on the segment shall become the Querier and send Queries. Other routers are free to listen to the Replies (they have to do it anyway) but they do not send Queries themselves.

IGMP v2 Host Membership Report: The IGMP v2 Host Membership Report has the same function as the IGMP v1 Host Membership Report except that it is intended to be received by IGMP v2 routers.

Leave Group Message: The Leave Group message is used to reduce the time it takes for the multicast router to stop forwarding multicast traffic when there are no longer any members in the host group. If a host responds to the last IGMP query, it might be the last or only member of the host group. When this host leaves the group it sends an IGMP Leave Group message to 224.0.0.2 (the all routers group). Upon receipt of the Leave Group message, the router sends a series of group-specific queries for the host group. If no host responds to the group-specific queries, the router determines that there are no more members of that host group on that particular subnet and removes the entry from the IGMP interface group table.

IGMP Group-Specific Query: An IGMP Host Membership Query is sent to 224.0.0.1 (the all hosts group) to query for the group membership of all hosts on the subnet. IGMP v2 routers can also send a group-specific query, a query for a specific multicast group sent to the group address [15].

IGMPv3

Version 3 extends the functionality of IGMP while maintaining backward compatibility with IGMPv2. It provides support for a new approach called source specific multicast (SSM). Instead of specifying the group IP address only, IGMPv3 messages contain the multicast group IP address and the unicast IP address of the content source. This combination identifier is represented as (S, G) where S is the unicast IP address of the source and G is the group address [8].

IGMPv3 does not use leave group messages, as this functionality is provided through the source address filtering system [26].

2.4.2 Multicast Listener Discovery (MLD)

Multicast Listener Discovery enables management of subnet multicast membership for IPv6.

MLD v1

Its functionality is similar to IGMPv2.

MLD v2

Its functionality is similar to IGMPv3.

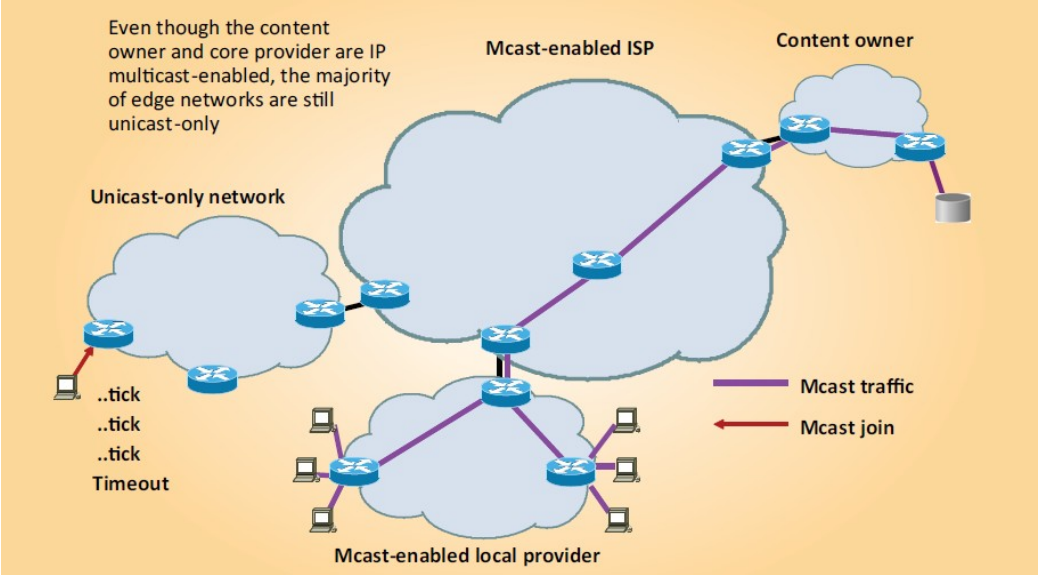

Figure 2: IP Multicast Challenge

2.5 Challenges in IP Multicast

- One of the major weaknesses of the existing multicast service model is the IGMP protocol, which fails to provide any sort of receiver access control (i.e., an unauthenticated user can send an IGMP request to join a multicast group).

- Unfortunately, the basic IGMP join message itself is unsecured, which means that IGMP messages are not encrypted and can be spoofed.

- It is not suitable for the end users who do not have native multicast connectivity. Only hosts that have native multicast connectivity will receive multicast data. As shown in Figure 2 the hosts that are in a unicast only network can neither send multicast-join nor they can receive multicast traffic from the source that is in a multicast-enabled network.

Chapter 3

Participant Access Control (PAC) in IP Multicast

3.1 Introduction

This chapter introduces the reference architecture for multicast access control that is used in our laboratory and the roles of its components. After, we explain what was added to this architecture in order to achieve Receiver Access Control in IP multicast.

3.2 Reference Architecture

A secure multicast architecture with sender and receiver access control has been proposed in [2] and has interacting components as illustrated in Figure 3.

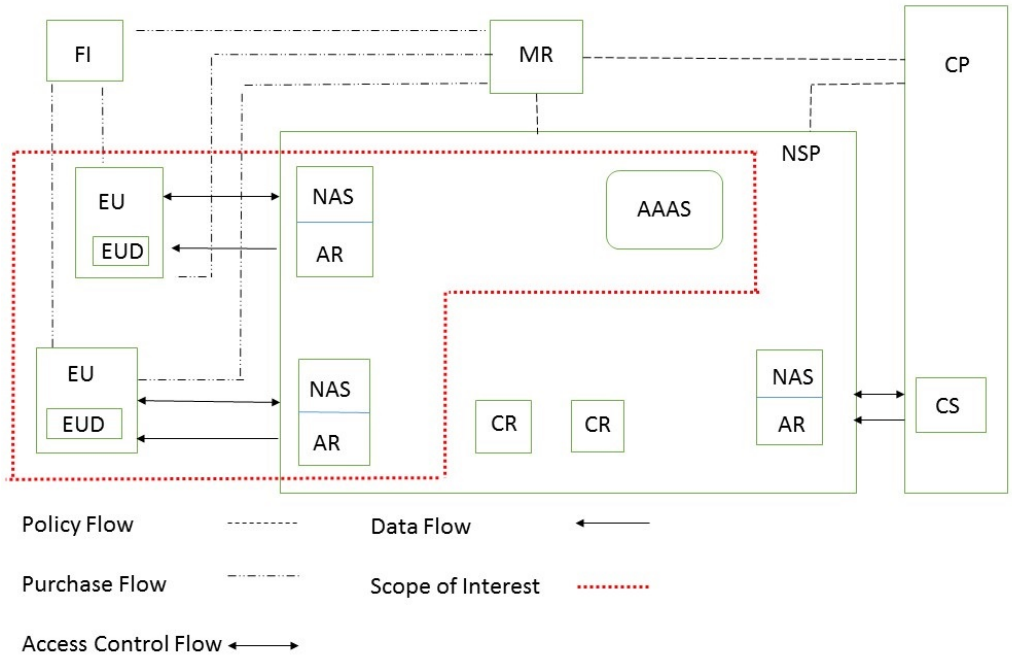

Figure 3: Reference Multicast Architecture

3.2.1 Components

A brief description of components is as follows:

End User (EU): A subscriber who wishes to receive multicast data.

End User Device (EUD): A device that is operated by an EU to receive multicast data.

Merchant (MR): The merchant is actually a web based participant who introduces and advertises the service to the public. It should be easily accessible for a customer so that he/she can visit the MR anytime and anywhere in the Internet to choose a service. The MR is responsible for receiving the order from potential EUs, checking their ability to pay through trusted communications with Financial Institutions and issuing the EU a permission ticket to allow him/her to join a multicast session in the

future.

Network Service Provider (NSP): The NSP makes use of its internal components for delivery and charging issues as well as authentication and authorization of end users. It receives multicast data from the source and once the end user presents his permission ticket to the NSP, it delivers the multicast data to authenticated and authorized end users. The NSP creates an account for each individual end user at the network edge for further accounting purposes.

CS: Content Server is the actual producer of physical data stream. It is managed by a Content Provider.

CP: Content Provider is the provider of server-side policies, who communicates with the Merchant (MR) and the Network Service Provider (NSP) to issue basic policies and distribute them to all the actors.

FI: Financial Institution is an outsider financial participant (e.g., a bank, a credit grantor, etc.) who has secure and trusted connections with the MR. The financial institution approves or disapproves the MR's request regarding the individual EU's ability to pay. Once the FI approves the ability to pay for a specific end user, the MR becomes assured that the EU can be charged at the end of the session for his consumed resources. That is, the FI will be responsible for eventually taking money from the user on the MR's behalf [30].

AAA Server (AAAS): A device for managing authentication, authorization and accounting within the NSP.

Access router (AR): A routing device within the NSP, close to the EUD, which is

responsible for adjudicating access rights to the network.

Network Access Server (NAS): The enforcement function for managing authentication, authorization and accounting within the NSP. Normally co-located with the AR.

Core Router (CR): A routing device within the NSP that does not have any EUD connected to it directly.

3.2.2 Information Flow

Moreover, four kinds of information flows are shown in Figure 3. They are as follows

Policy flow: Exchange of policy information.

Purchase flow: The transactions related to subscribing to and paying for a group session.

Access control flow: The presentation of authentication and authorization information. There are two categories: Receiver Access Control flow, which is between the NSP and receivers (e.g., EUs) and Sender Access Control flow, which is between the NSP and senders (e.g., CP).

Data flow: The delivery of the subscribed data stream, from the CS through the NSP to the EUD.

The term Access Control used here mean authentication, authorization and accounting (AAA) functionalities for both sender and receivers of a multicast group. However, we are limiting our focus to Receiver Access Control in this document. Therefore, in Figure 3, our interest focuses on the area inside the red (dotted) line

[24].

3.3 Underlying Protocols

Access control in IP multicast is achieved with different protocols, which are explained briefly in subsections below. Extensible Authentication Protocol (EAP) is used to authenticate EUs for network access. It runs between an EAP peer and an EAP server, where the EAP peer will request authentication and the EAP server will authenticate the peer. In some cases the EAP authenticator will act as a pass-through between EAP peer and EAP server (see Section 3.3.1). In such cases, Protocol for carrying Authentication for Network Access (PANA) is used to carry EAP messages between the EAP peer and the EAP authenticator (see Section 3.3.2), while the Diameter protocol is used to carry EAP messages between the EAP authenticator and the EAP server (see Section 3.3.3).

3.3.1 Extensible Authentication Protocol (EAP)

Definition

Extensible Authentication Protocol [1] is an authentication protocol or framework that supports multiple authentication methods.

Components

EAP peer: is an End User(EU) host that attempts to access a network and responds to the authenticator.

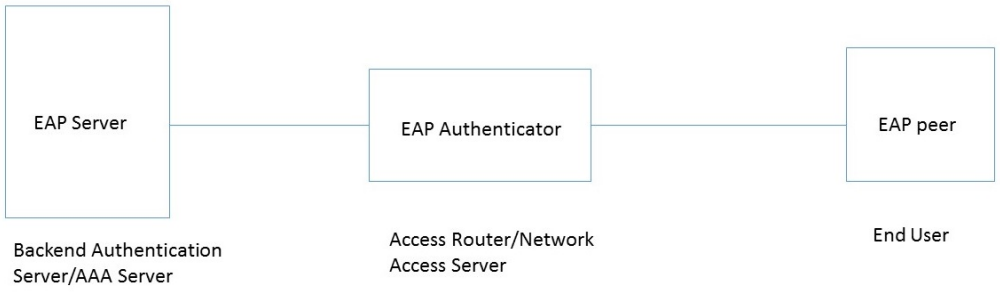

Figure 4: EAP Components

EAP authenticator: is an Access Router (co-located with Network Access Server NAS as shown in figure 4) that initiates EAP authentication. It can act as a pass-through (if necessary).

EAP authentication server: or backend authentication server is an entity that provides an authentication service by executing EAP methods for the authenticator [30].

Authentication in EAP

The EAP authentication exchange is shown in Figure 5:

1. The authenticator sends a Request to authenticate the peer. The Request has a type field to indicate what is being requested (e.g., Identity, MD5-challenge, etc.).

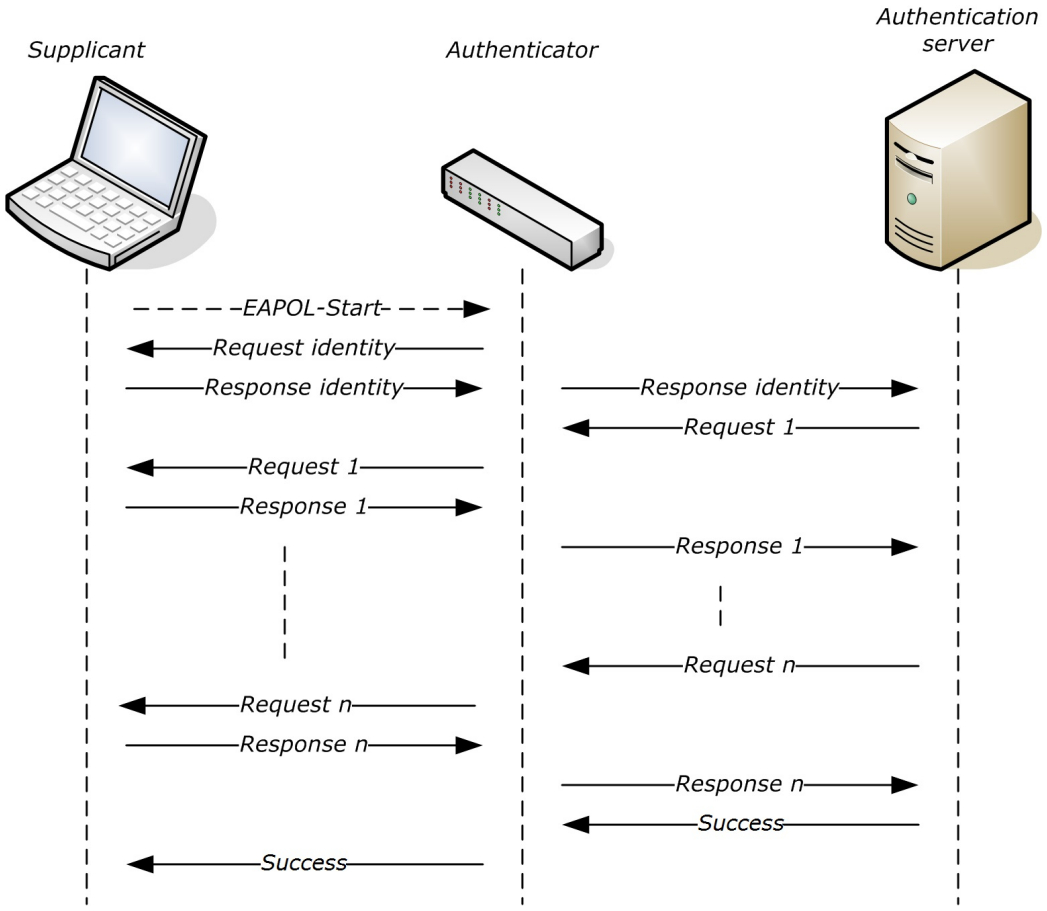

Figure 5: EAP Message Exchanges

2. The peer sends a Response packet in reply to a valid Request. As with the Request packet, the Response packet also contains a type field.

3. The authenticator sends an additional Request packet, and the peer replies with a Response. The sequence of Requests and Responses continues as long as needed. Since EAP is a *lock step* protocol, a new Request cannot be sent prior to receiving a valid Response to the initial Request.

4. The conversation continues until -

    - the authenticator determines that successful authentication has occurred,

in which case the authenticator must transmit an "EAP success" message.

- or in contrast, the authenticator cannot authenticate the peer, in which case the authenticator transmits an "EAP failure" message.

3.3.2 Protocol for Carrying Authentication for Network Access (PANA)

Definition

PANA [16] is a network-layer transport for Extensible Authentication Protocol (EAP) to enable network access authentication between clients and access networks. In EAP terms, PANA is a UDP-based EAP lower layer that runs between an EAP peer and an EAP authenticator. The general PANA framework is shown in figure 6.

Components

PANA client (PaC): is an entity of the protocol that resides in the access device (e.g., laptop). It is responsible for providing the credentials in order to prove its identity for network access authorization. The PaC and EAP peer are co-located.

PANA Authentication Agent (PAA): is a protocol entity in the network whose responsibility is to verify the credentials provided by the PaC and authorize network access to the device. The PAA and EAP authenticator are co-located in same device.

Enforcement Point (EP): is a node on the access network where pre-packet filtering is done on the inbound and outbound traffic of access devices. EP should prevent

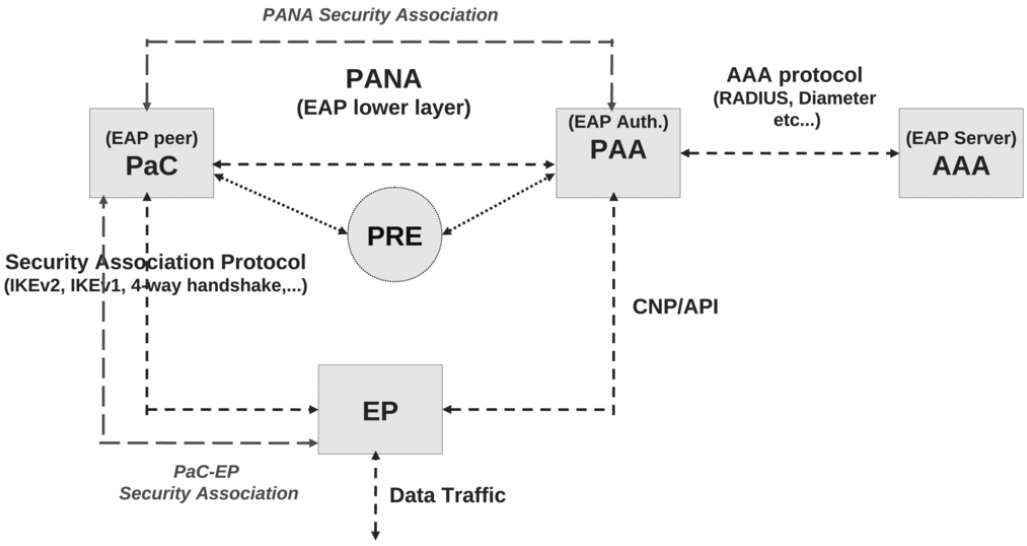

Figure 6: PANA Framework

data traffic from and to any unauthorized client, unless the data is either a PANA message or other allowed traffic types (e.g., ARP, DHCP, etc.). The EP and PAA may be co-located.

Authentication Server (AS): is a conventional back-end AAAS that terminates the EAP and the EAP methods.

Protocol Overview

PANA messages are sent between the PaC and PAA as part of a PANA session. A PANA session consists of four distinct phases:

1. **Authentication and Authorization phase:** This is the phase that initiates a new PANA session and executes EAP between the PaC and PAA. The PAA conveys the result of authentication and authorization to the PaC at the end of this phase.

2. **Access phase:** After successful authentication and authorization, the End User device gains access to the network. Now it can send and receive IP traffic through the EP.

3. **Re-authentication phase:** During the access phase, PAA may and PaC should initiate re-authentication if they want to update the PANA session lifetime before it expires. EAP is carried by PANA for re-authentication.

4. **Termination phase:** During this phase an explicit disconnect message is sent by either PaC or PAA to discontinue the access service at any time.

3.3.3 Diameter Protocol

The Diameter protocol [14] is intended to provide an Authentication, Authorization and Accounting (AAA) framework for applications such as network access or IP mobility. It evolved from and replaces the much less capable *Radius* protocol [32] that preceded it. It is used to carry EAP packets from an EAP Authenticator to the EAP Authentication Server (AAA Server).

3.3.4 IPsec

Internet Protocol Security (IPsec) is a protocol suite that provides an interoperable, high quality, cryptographically-based security service for IPv4 and IPv6 networks. It can be used to protect any traffic across an IP network. The protocol suite is composed of four main components.

1. Security protocols, which provide traffic security services, such as authentication and encryption. There are two variants: Authentication header (AH) [21] and Encapsulating Security Payload (ESP) [20].

2. The security architecture defined in [22] is based on the concept of a Security Association (SA). An SA is a simplex logical connection that affords security services. In IPsec, an SA is a network-level abstraction implemented through the use of AH or ESP. An arbitrary 32-bit value, called a Security Parameter Index (SPI), is used by the receiving end of the connection to identify the SA to which the incoming traffic should be bound. Access to IPsec SAs is managed using three conceptual databases, they are Security Association Database (SAD), Security Policy Database (SPD), Peer Authentication Database (PAD). [37] extends the security architecture to multicast communication. It defines the concept of a Group Security Association (GSA), which is used to protect the communications within the group. Access to IPsec GSAs is managed using three conceptual databases, they are Security Association Database (SAD), Group Security Policy Database (GSPD), Group Peer Authentication Database (GPAD). In IPsec for an incoming protected packet, the SPI contained in its IPsec header is used as an index into the SAD, to determine the parameters for decoding the packet contents.

   For an outgoing packet that is to be protected, destined to a particular peer address, and having a particular next protocol, the SPD (unicast peer address)

or the GSPD (multicast peer address) is consulted to determine the SPI that is specific to that peer and that protocol. The resulting SPI is used to locate the SAD entry, which in turn specifies the encoding to be used for the packet contents. If no matching entry is found in the SPD or GSPD, the PAD (unicast case) or the GPAD (multicast case) is consulted to determine which key management protocol is to be used to establish the SAs between participants or the GSAs among participants. The key management protocol is responsible for negotiating the parameters to be placed in the SPD (or GSPD) and the SAD, corresponding to the newly-established SA (or GSA).

3. Key Management Protocols are used to automatically manage the keys used in IPsec. IKEv2 is a component of IPsec used for performing mutual authentication and establishing and maintaining SAs between two devices. It establishes two SAs between the two devices, one in each direction. The SPI for each direction is chosen by the receiving device, thus ensuring that there is no conflict between the chosen SPI and any other SPI in use on that device. For multicast groups, Group Key Management for IPsec GSAs may be done by Group Domain of Interpretation (GDOI) [38] or by G-IKEv2 [33], which is based on IKEv2. In this document we use Group Security Association Management (GSAM) as group key management protocol (see Section 3.3.6) for our specific needs.

4. Cryptographic algorithms are used in security protocols and key management protocols for authentication and encryption .

3.3.5 SIGMP

Secure Internet Group Management Protocol (SIGMP) is an extension of IGMP, which runs among the EUs and the ARs. The EU implements the host portion of SIGMP while the AR implements the router portion of SIGMP. SIGMP queries and reports are each divided into two categories, Open Group Query (OGQ), Secure Group Query (SGQ), Open Group Report (OGR), Secure Group Report (SGR). OGQ and OGR are for open groups and SGQ and SGR are for secure groups. In SIGMP, queries and reports for open groups are delivered without any protection, but for secure groups they are protected by IPsec GSAs. GSAs are of two kinds: GSA-q and GSA-r. GSA-q is used to protect SGQ messages and GSA-r is used to protect the SGR messages.

3.3.6 GSAM

The Group Security Association Management (GSAM) protocol is used to manage the GSAs used in SIGMP (similar to IKEv2 in unicast). The network entities in GSAM are the same as those in SIGMP, including ARs and EUs. In GSAM, an AR (specifically, the Querier) plays the role of group controller / key server (GCKS). It accepts registrations from NQs and EUs that have been authorized at the application level and grants them group membership in the secure multicast groups that the EUs are authorized to join. The members of this set of EUs are called Group Members (GMs). The AR/Q creates and updates SPI, GSA-r and GSA-q for a secure group and distributes them to GMs in the secure group using secure tunnels. The Q, the

NQs (if any), and the GMs will update their local SADs and GSPDs according to the parameters of GSA-q and GSA-r to protect the SIGMP packets.

3.4 Receiver Access Control Interactions in Multicast Architecture

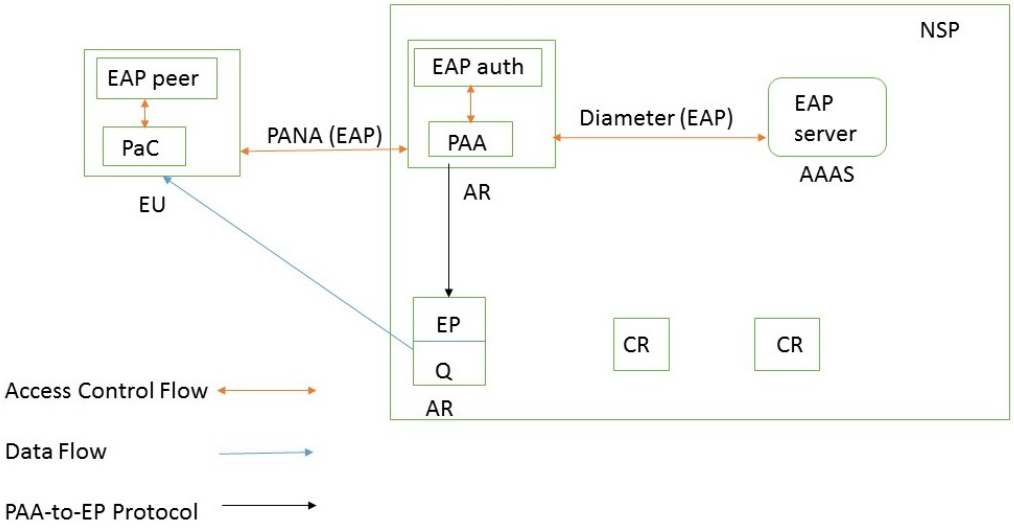

Figure 7: Receiver Access Control Architecture in IP Multicast

The receiver access control can be viewed at two levels: the application level and the network level. The work reported in [19] is focused on the application level and the work reported in [24] is focused on the network level.

3.4.1 Access Control at Application Level

EU will be issued a "token" by a valid merchant in order to receive data from his favorite multicast service. The token contains application layer information and is placed into an EAP packet (see figure 8). For Authentication, Authorization and Accounting purpose, the packet has to move from the EU to an EAP server. The AR (co-located with the NAS) forwards the packet as the EAP authenticator to the EAP server. After successful authentication EAP exports Master key (see 3.4.1). The EAP packet is carried through PANA framework between the EU and AR (as shown in figure 8).

As shown in Figure 7 the PANA client (PaC) will be on the EU. On the NSP side

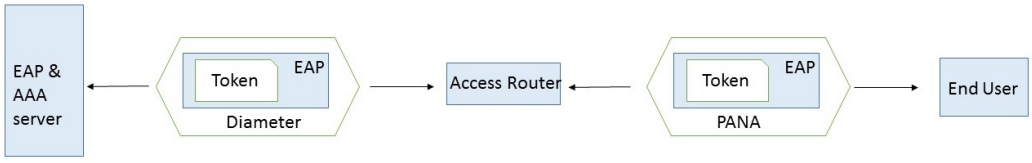

Figure 8: EAP Packet Carriers

of the network segment, there are two PANA-related functions: the PANA Authentication Agent (PAA) and one or more Enforcement Points (EPs). In the simple case (only one AR for the network segment), the PAA and the EP will be co-located with the AR. In more complex cases (more than one AR for a specific network segment), one AR will have both PAA and EP functions, and the rest will have only the EP function.

A PANA session consists of 5 different phases (actually four, but first phase is divided into two). In the following we have explained how the entities will behave in

our architecture during these phases:

Handshake phase: The PaC (EU), on receiving a request (ticket) from the upper layer to join a multicast group while no PANA session had been completed yet, initiates a PANA session by sending a PANA-Client-Initiation message to the PAA (access router).

Authentication and authorization phase: Immediately following the handshake phase, PANA will carry EAP messages between the EU and the AR. A AAA protocol (e.g., Diameter) will carry the EAP exchanges between the AR and AAAS, where the ticket is validated. On successful authentication, both the AAAS and the PaC derive the same Master session key (MSK), which is forwarded to the PAA by the AAAS .

Access phase: On receiving the MSK the PAA generates a separate 64 octet PaC-EP Master Key (PEMK) for each EP [29] and PaC also calculates the same key. One way of calculating this key is,

$$PEMK = prf + (MSK, \text{“}IETFPEMK\text{”}|SID|KID|EPID)$$

Here, prf+ is a pseudo-random function defined in [28]. “IETF PEMK” is the ASCII code representation, SID is a four-octet Session Identifier, KID is associated with the MSK and EPID is the identifier of the EP. This key is specific to the multicast group that the EU has joined at the application level, and will be used for authorization at the network layer.

Re-authentication phase: If PANA re-authentication is required, this phase will trigger a EAP re-authentication, which returns to the access phase upon successful re-authentication.

Termination phase: The receiver (e.g., on receiving multicast leave request from the upper layer) or the AR (e.g., the receivers authorized access has expired or been revoked) may choose to discontinue the access service at any time. At the end of this phase, the AR will gather accounting data for the receiver and communicate these data to the AAAS (using Diameter) for updating the receiver's account [19].

The EAP packet is then carried through AAA protocol (e.g., Diameter Protocol) between AR and EAP server (as shown in figure 8). Clearly, a AAA server must be co-located with the EAP server to receive the Diameter packets, extract the EAP material and forward them to the EAP server.

3.4.2 Access Control at the Network Level

When the EU makes his/her request for the network-level join and if the request is for an Open group, the operations are same as in IGMPv2. For the access control of secure groups the SIGMP (Secure IGMP) module will be invoked, to send the SGR (Secure Group Report) to the Q. The SGR will be passed to the IPsec module. If this is the first time that an SGR has been sent for this network-level group, there will be no corresponding entry in the SAD or the GSPD, so the GPAD will be examined. The GPAD entry corresponding to this new group will indicate that GSAM is the appropriate key management protocol. The IPsec module will therefore invoke GSAM to negotiate Security Association key pairs (GSA-r, GSA-q). It will use the PANA PEMK (previously stored in the GPAD) for its authentication. Once GSAM has completed its negotiations, the resulting parameters will be installed in the SAD and

the GSPD. (This will provide for outgoing and incoming messages, each with its own SPI and key for this group.) The IPsec module will then be able to send the SGR. Any subsequent SGR message will be sent using the parameters stored in the GSPD and the SAD (for more details see [24]).

Chapter 4

Automatic Multicast Tunneling

The following terminology is largely adapted from document [6].

4.1 Definition

Automatic Multicast Tunneling (AMT) allows multicast communication to take place between hosts, sites or applications that do not have native multicast access and one or more sources that have native multicast connectivity, to request and receive SSM and ASM traffic from a network that does provide multicast connectivity to that source. Without requiring any manual configuration, AMT allows the hosts to receive multicast traffic from the native multicast infrastructure. AMT operates with a pseudo interface where UDP-based encapsulation is done to overcome problems of multicast connectivity.

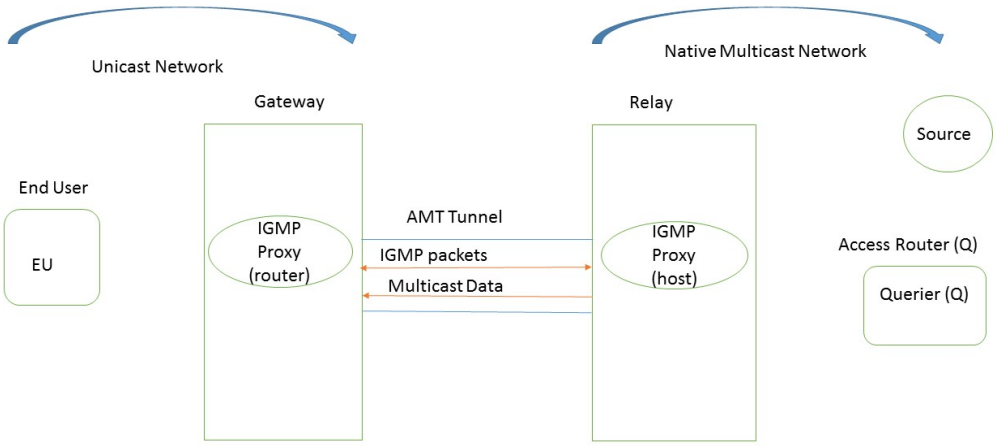

Figure 9: AMT Multicast Architecture

4.2 Components

As shown in Figure 9, the roles of components are as follows:

AMT Relay: The relay router terminates one end of tunnel and has one or more interfaces. The downstream interfaces of a relay serves gateways, i.e., the relay accepts encapsulated IGMP and MLD group membership messages from gateways, encapsulates and forwards the requested multicast traffic back to those gateways.

The upstream interfaces of a relay communicates with a native multicast infrastructure, i.e., the relay sends join and prune/leave requests towards multicast sources and accepts requested multicast traffic from those sources.

AMT Gateway: It is host or a router that terminates the other end of the tunnel. It does not have native multicast connectivity to the multicast backbone infrastructure and has one or more interfaces. The downstream side of a gateway serves one or

more multicast receivers, i.e., the gateway accepts group membership requests from receivers and forwards requested multicast traffic back to those receivers. The gateway functionality may be directly implemented in the host requesting the multicast service or within an application running on a host.

The upstream side of a gateway connects to a relay. A gateway sends encapsulated IGMP and MLD messages to a relay to indicate an interest in receiving specific multicast traffic.

AMT Pseudo-Interface: It is conceptually a network interface on Gateway and Relay, where the Gateway executes host portion and Relay executes the router portions of IGMP and MLD protocols. It is basically a virtual interface where AMT encapsulation and decapsulation occurs. Some implementations may treat it exactly as any other interface and others may treat it as a tunnel end-point.

4.3 IGMP Proxy

The IGMP proxy implementation in the Gateway runs AMT on an upstream interface and router-mode IGMP/MLD on downstream interfaces to provide host access to multicast traffic.

The IGMP proxy implemented in the Relay, runs AMT on a downstream interface and host-mode IGMP/MLD on a upstream interface. This "relay proxy"sends group membership reports to a local, multicast-enabled router to join and leave specific SSM or ASM groups. By using an IGMP proxy, the IGMP/MLD protocol behavior

in AMT, from the point of view of the EU, is exactly the same as in IP multicast.

4.4 AMT Architecture Operations

Initially, let us assume that the multicast-enabled ISP provides the AMT Relay service. As shown in figure 10, the hosts connected to the unicast-only network are acting as AMT Gateways.

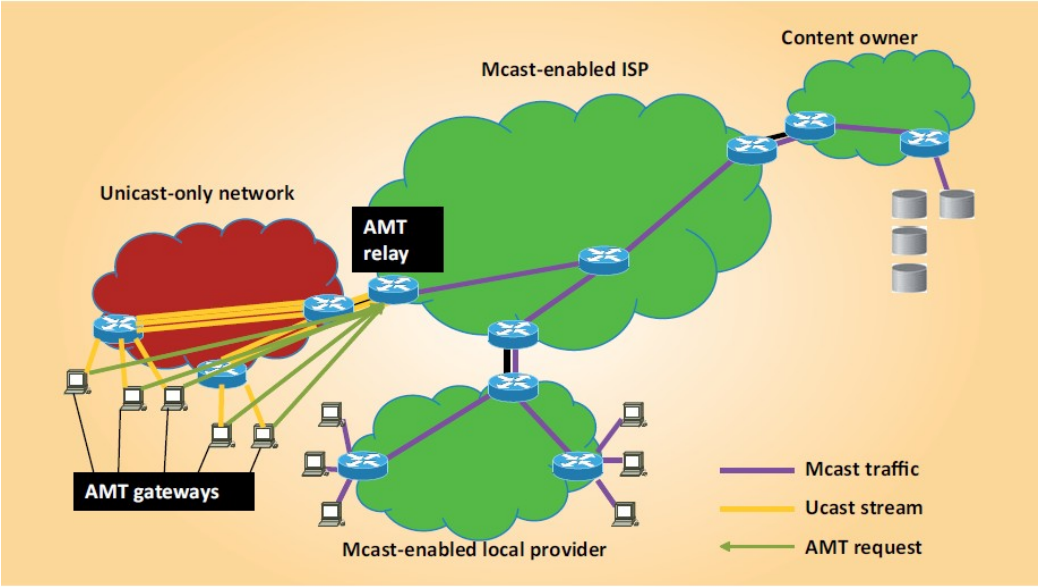

Figure 10: AMT

1. When host wants to join a multicast group, it sends a membership report to gateway thinking that it is an IGMP router (Querier).

2. Before sending the received report, the Gateway will send a Request message to the Relay to solicit a general-query response. The Relay responds by sending Membership Query message back to the gateway. The Membership Query message carries an encapsulated general query that is processed

by the IGMP or MLD protocol implementation on the Gateway to produce a membership/listener report. Each time the Gateway receives a Membership Query message it starts a timer whose expiration will trigger the start of a new Request. This timer-driven sequence is used to mimic the transmission of a periodic general query by an IGMP/MLD router. This query cycle may continue indefinitely once started by sending the initial Request message.

3. After receiving the general query from the Relay, the Gateway will send the membership report encapsulated to the Relay. Each report is encapsulated and sent to the Relay after the Gateway has successfully established communication with the Relay via a Request and Membership Query message exchange (for more details on membership update sequence refer to [6]).

4. The AMT Relay will decapsulate the IGMP messages and trigger an upstream PIM join towards the source.

5. Finally the requested multicast data is transferred from the multicast source to host/EU through the Relay and the Gateway.

4.5 AMT Benefits

4.5.1 Simplicity

To establish the AMT tunnel, the receiving network simply sends a request to the unicast address of the Relay. The rest of the tunnel establishment is done automatically

without the need for any additional configuration or overhead of manual monitoring.

4.5.2 Efficiency

AMT uses UDP encapsulation, providing different source UDP ports for the encapsulated streams of data, allowing transit routers to perform flow-based load balancing for more efficient link utilization.

4.5.3 Resiliency

If the Gateway uses an Anycast address to discover the Relay, it will automatically find the closest Relay. When the Relay becomes overloaded or unavailable the Gateway is routed to next closest Relay and the routing table is updated automatically [34].

4.6 AMT Challenges

- An intruder can easily impersonate an AMT Relay and Gateway.
- An intruder can learn the value of Message Authentication Code (MAC) successfully.
- AMT does not have Access Control over multicast groups.

Chapter 5

Problem Statement

5.1 Introduction

In this chapter, we will mention deficiencies of the previous multicast architectures and reach to the specific point that we wish to talk about later on as thesis contribution. Our goal is to find a design solution that can overcome those problems.

5.2 Deficiencies and Goals

Multicast architectures defined in Chapters 3,4 has few limitations and challenges. Participant Access Control (PAC) architecture in [19] [24] is compatible only in multicast enabled region. It requires that all the components between the multicast source and the receiver support IP multicast technology; in other words hosts that are in unicast-only-network cannot access multicast enabled network. Unfortunately, many

of the devices or applications that use IP multicast lack multicast connectivity to networks that carry traffic generated by multicast sources. The solution for the above problem is the architecture of AMT [6], in which unicast hosts are capable of receiving multicast data. However, AMT is incapable of differentiating between legitimate and non-legitimate users. Because IGMP is insecure, there is no control over the users who are trying to access the network without permission.

Therefore our goal is to add Receiver Access Control features to the current functionality features of AMT and see how those features fit in our model and how the security considerations are handled.

Usually in IP multicast, Receiver Access Control features are implemented between the client that wants access to the network and the server that authenticates and authorizes it. However, in the AMT environment we have a Gateway and a Relay additionally between them. During this process we have to analyze where all the RAC components will fit in the current model and how they communicate without causing any distraction to the security considerations. Also, what are the changes that we need to make in the Gateway and the Relay, so that the whole Receiver Access Control functionality in AMT environment reflects the same results and security as in IP multicast. Considering all these issues, we propose a design solution that provides Receiver Access Control in the AMT environment.

Chapter 6

Receiver Access Control in AMT

6.1 Proposed Solution

Here, in this section we are going to explain how a PANA framework is accommodated into the AMT environment to achieve *Receiver Access Control*. Figure 11 shows the AMT architecture with "Receiver Access Control". The whole design is based on the fact that all messages and data between EU and AR must pass through the AMT Tunnel, i.e., between gateway and relay.

6.1.1 PANA Proxy

Usually in IP multicast environment PANA communication is done directly between PaC and PAA. But in AMT we have additional components called the Gateway and the Relay between them. We want all the PANA messages to flow through the Gateway and the Relay. For this to happen, we explicitly do not want the PaC to

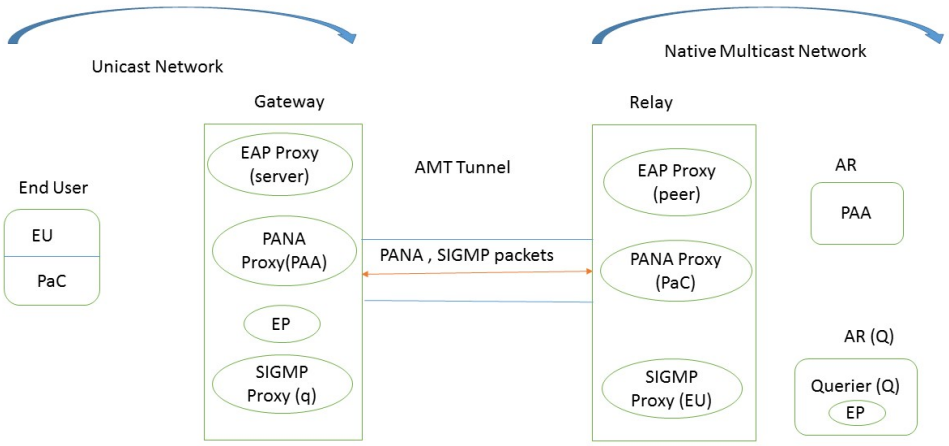

Figure 11: AMT Architecture with Receiver Access Control

know the "real address" of the PAA, because it might allow the messages to flow outside of AMT tunnel taking another route. Considering above facts we introduced "PANA Proxy" concept in AMT. In the unicast-only network, the PANA proxy is implemented in the Gateway and works as a PAA for the PaC, i.e., the PaC assumes that the Gateway is its PAA and starts exchanging messages with the Gateway. By default, the PaC discovers the PAA in the Gateway using the normal mechanism for PAA discovery as defined in [27]. In the native multicast network PANA proxy is implemented in the Relay and works as a PaC for the PAA, i.e., the PAA assumes that the Relay is its PaC and communicates with it. The idea of PANA proxy is taken from PANA Relay Element (PRE) [13], which enables PANA messaging between a PaC and PAA where the two nodes cannot reach each other by means of regular IP routing.

The proposed model also implements a PANA Client (PaC) inside an End User

(EU) host that is residing on the non-multicast enabled network (unicast-only network) looking for an IGMP join. A PANA Enforcement Point (EP) is implemented inside the AMT Gateway, so that it can transfer the necessary keys to the q (querier) in the Gateway. The PAA that consults a back-end Authentication Server (AS) for authentication and authorization of a PaC may co-located with Q or may be on other device that is residing on the multicast enabled network.

6.1.2 EAP Proxy

PANA carries the methods of the Extensible Authentication Protocol (EAP) that are responsible for authenticating the PaC, i.e., EU (they are co-located). After authentication is done, the EAP method exports a Master Session Key (MSK) to the PaC and the PAA. In AMT, the Gateway and the Relay act as a friendly Man-in -Middle. As a result they will know (or be able to construct) the MSK for protecting IGMP messages (see section 6.1.4). Considering this idea, we introduced an EAP Proxy concept in AMT. In the unicast-only network, the EAP proxy is implemented in the Gateway and functions as an EAP server for the EAP peer. In the native multicast network, the EAP proxy is implemented in the Relay and functions as an EAP peer for the EAP server co-located with the PAA. (Note: EAP peer and PaC are on same device and EAP authenticator and PAA are co-located).

6.1.3 SIGMP Proxy

In IP multicast, SIGMP runs between EUs and ARs. In AMT we have the Gateway and the Relay between and we want all the SIGMP packets to pass through them. For this to happen, we explicitly do not want the EU to know the *real address* of the AR or else the messages might take another route other than AMT tunnel. So considering the above facts, in the unicast network we have introduced a SIGMP proxy in the Gateway (as shown in figure 11), which makes EU to think that the Gateway is its AR/q. (q is querier in gateway, we are calling it as querier because in AMT there is only one AR and itself is querier). In the multicast network SIGMP proxy is in the Relay and acts as EU for the real Querier.

6.1.4 Receiver Access Control Interactions in AMT

As explained in Section 3.4, the Receiver Access Control (RAC) can be viewed at two levels: the application level and the network level.

Access Control at Application level

We know that a PANA session consists of five phases (see section 3.4.1). In the following we have explained how the PANA messages are exchanged during these phases in AMT using PANA proxy and EAP proxy.

1. Handshake Phase: The PaC, on receiving a request from the upper layer to join a multicast group, initiates a PANA session by sending a PCI message to the Gateway thinking it is the PAA. Gateway finds it as a PANA packet and

forwards it to the Relay. The Relay, having a PANA proxy acting as a PaC, forwards the packet to the actual PAA. The response goes back from actual PAA to PaC through the Relay and the Gateway.

2. Authentication and authorization phase: After the handshake phase, EAP packets carried by PANA will be exchanged between the PaC and the PAA. For better understanding, we took an example of EAP-FAST method [9], an efficient EAP method. This method has two phases, in which phase 1 is responsible for TLS handshake resulting in a secure tunnel between peer and server. As explained, EAP proxy acting as EAP server is in the Gateway and the EAP peer is in the EU. The secure tunnel is formed between the EU and the Gateway (say STunnel1), resulting in a fresh secret key between them. The same secure tunnel with another key is formed between the Relay and the PAA (say STunnel2) during phase 1. In phase 2, EAP method payloads carrying user credentials in PANA packets are transferred to the Gateway through STunnel1 and the Gateway, who shares the secret key with the EU during phase1, will decrypt and forward them to the Relay through the AMT Tunnel (assuming AMT tunnel is secured). Finally the Relay protects the payloads with keys obtained in formation of STunnel2 and forwards the EAP message to the PAA. The PAA verifies those credentials and authenticates EU and sends the results back.

   After a successful authentication, the PaC and PAA derive a *Master Session*

Key (MSK). As the Gateway and the Relay are part of PANA exchanges and acting as a friendly Man-in-Middle, they can compute the MSK as well. On receiving the MSK the PAA transfers MSK to EPQ (Enforcement point in Querier) using IPsec, with a key calculated in the normal way for two IPsec peers [40].

3. Access Phase: PaC and EPG (Enforcement point in Gateway), Relay and EPQ with acquired pre shared key (MSK) during authentication phase calculate the secret key called PEMK (section 3.4.1 shows how to calculate PEMK) respectively. As the EPs are on different devices they end up calculating different PEMKs, i.e., PEMK1 between the PaC and the Gateway, PEMK2 between the Relay and the actual Querier.

   With those PEMKs, they establish a two different IPsecGSAs between them for cryptographic protection of IGMP messages. Each IPsecGSA contains one GSA-r (Group Security Association for reports) and one GSA-q (Group Security Association for queries), for details see 6.1.4, 6.1.5. This phase is also used to test liveness of the PANA session.

4. Re-authentication and Termination phases are similar to that described in 3.4.1, except the fact that these PANA messages are exchanged through the AMT Tunnel.

Access Control at Network level

In SIGMP [24] some messages are protected by IPsec GSAs (Group Security Association). In this protocol all the operations for OGQ (Open Group Query) and OGR (Open Group Report) are retained from IGMPv3. However, for the access control of secure groups, a few operations are added in it. The following subsection will describe how SIGMP is fitted in to AMT.

EU Operations

Once the Authentication is done at application level, EU will make his/her request for the network-level join and will send an SIGMP report message, believing it is being sent to the real Q (In fact, it will be received by the SIGMP proxy in the Gateway). If this is the first time, when the report is sent to the IPsec (GSA) module, GSAM will be invoked to negotiate the cryptographic parameters (keys and SPIs see Section 6.1.5). The IPsec module will then be able to send the report protected by those secure parameters to the Gateway where the SIGMP proxy (q) is implemented. The q in gateway will forward the message to the Relay through a secured tunnel (assumed) and finally the Relay will forward it to the actual Q that accepts the request.

Q Operations

On receiving a secured report, Q will invoke IPsec module to decrypt it. Later it checks for multicast address and if the address indicates a secure group it performs an address consistency verification. In this verification it compares the multicast

address in the group record and destination address in the IP header. If verification succeeds, it will proceed to update the group memberships and refresh the timers, if not it will discard the report.

6.1.5 GSAM

Group Security Association Management Protocol (GSAM) manages IPsec GSAs in two phases. In phase1, mutual authentication of EU and Q is done to achieve the registration of an EU. In phase 2, Q creates and distributes SA pair (GSA-q, GSA-r), named GSAM-TEK-SA to protect SIGMP messages (for details see [24]) .

Usually in IP multicast environment GSAM negotiations are done between EU and real Q, but in AMT we must not let the EU to communicate directly with the actual Q. For that as explained earlier, we implement an SIGMP proxy, which acts as querier functionality (q) in the Gateway, so that EU starts mutual authentication with the Gateway (q) using the derived PANA secret key, i.e., PEMK1. After authentication is done the Gateway (q) creates and distributes GSAM-TEK-SA (SA pair) to EU. On the other side of AMT tunnel SIGMP proxy acting as EU in the Relay performs mutual authentication with the actual Querier (Q) using PEMK2 and receives an SA pair from Q.

6.2 Multiple Sessions in PANA

Basically PANA is designed to support a single session, which contains four phases: Authentication and Authorization Phase, Access Phase, Re-authentication Phase and Termination Phase. This is sufficient for its original intended use as a method for controlling access of a device to the network.

In our case, we are using PANA plus EAP to control access to (potentially) multiple multicast sessions. Therefore, we have to verify that PANA can support multiple sessions in order to make AMT an efficient design.

For that we have explored *Open Diameter-1.0.7*. The project provides the complete source code of Diameter and PANA and incomplete source code of EAP. For the EAP source code, only two authentication methods MD5 and archie are supported. Later, a new version of the project (1.0.8) is developed within our laboratory with another EAP authentication method called EAP-FAST. Open Diameter also provides three stable applications : *aaad, nasd* and *pacd* [23].

- aaad plays the role of the Diameter server and the EAP backend authentication server.
- nasd plays the role of the Diameter client, EAP pass-through server and PANA Agent (PAA)
- pacd plays the role of the PANA Client (PaC) and EAP client.

By tracing out the source code of Open Diameter project and research on the state machines of EAP and PANA, we concluded that PANA is capable of handling multiple

sessions simultaneously. Thus it is suitable for use in controlling access to multiple multicast sessions within AMT.

Chapter 7

AVISPA

7.1 Definition

To quicken the development of protocols and enhance their security, it is important to have appropriate tools that support the analysis of these protocols and help to find vulnerabilities in the early stages of development. Favorably, these tools should be entirely automated, robust, expressive, and easily usable, so that they can be integrated into the protocol development and standardization processes to improve the speed and quality of these processes. A number of (semi-)automated protocol analysis tools have been proposed, e.g.[3, 31, 35], which can analyze small and medium-scale protocols. However, scaling up to large scale Internet security protocols is a considerable challenge, both scientific and technological. This challenge was met by a team from several European universities and research organizations [4], who developed a formal modeling tool for validating the security properties of protocols, with

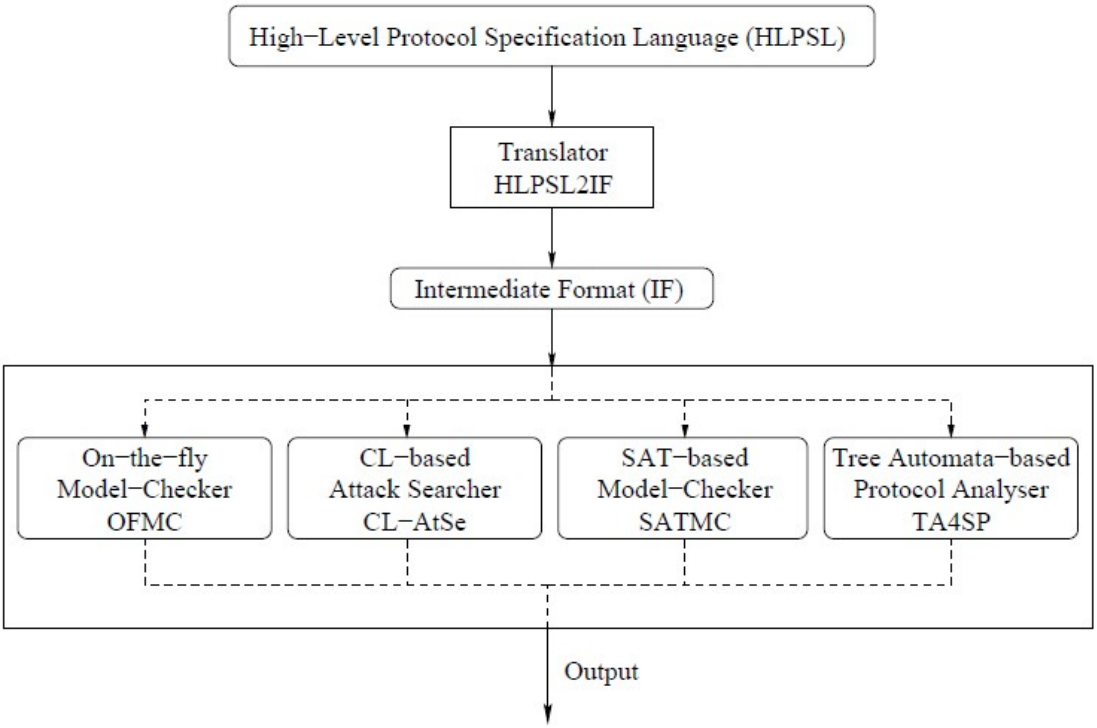

Figure 12: AVISPA Structure

an emphasis on the IETF protocols. It has been used by members of the team to validate the security properties of a significant number of IETF networking protocols [4]. This tool is called Automated Validation of Internet Security-sensitive Protocols and Applications (AVISPA). It is a "push-button" tool, in the sense that once the protocol under study has been modeled and the security goals stated, the rest of the process of validating the security properties is automatic.

The architecture of AVISPA is shown in figure 12. The first step in using the tool is to present the analyzed protocol in a special language called High Level Protocol Specification Language (HLPSL). We discuss the HLPSL language in more detail in the following section. The HLPSL presentation of the protocol is translated into

the lower level language called Intermediate Format (IF). This translation is performed by the translator called HLPSL2IF. This step is totally transparent to the user. The IF presentation of the protocol is used as an input to the four different back-ends: On-the-fly ModelChecker (OFMC), CL-based Attack Searcher (CL-AtSe), SAT-based Model-Checker (SATMC) and Tree-Automata-based Protocol Analyzer (TA4SP). These back-ends perform the analysis and output the results in a precisely defined output format stating whether there are problems in the protocol or not. Further explanation of the four back-ends is provided in Section 7.4.

7.2 High Level Protocol Specification Language

AVISPA uses High Level Protocol Specification Language (HLPSL) [4] to represent the analyzed protocols. In this section we take a closer look into the structure of HLPSL language according to the AVISPA tutorial [36]. In order to express the protocols in HLPSL language, it is easiest to translate the protocols first into A-B format, for instance:

```
A -> S: {Kab}_Kas

S -> B: {Kab}_Kbs
```

The notation above illustrates the Wide Mouth Frog (WMF) protocol [7], where endpoints A and B attempt to set up a secure session. First A generates a new session key Kab and encrypts it by using a key Kas and sends the encrypted key to the trusted server S. Kas is a key that is shared between A and S. S decrypts

the message, re-encrypts it by using a shared key Kbs and transmits the encrypted message to B. B can decrypt the message by using the shared secret Kbs and obtains the session key Kab. HLPSL language is a role-based language, which means that actions of each participant are defined in a separate module, called a basic role. In the case of the WMF example above, the basic roles are: Alice (A), Bob (B) and server (S). Basic roles describe what information the corresponding participant has initially (parameters), its initial state and how the state can change (transitions). To continue the WMF example, the role of Alice would be expressed in following way:

```
role alice(A,B,S : agent,
Kas : symmetric_key,
SND, RCV : channel (dy))
played_by A def=
local
State : nat,
Kab : symmetric_key
init State := 0
transitions
...
end role
```

The role indicates that agents A, B and S are participating in the protocol suite, A has a shared key Kas with the agent S and A uses channels SND (send) and RCV (receive) for communication. Currently, the only supported channel model for communication

in AVISPA is Dolev-Yao (dy). AVISPA's selection of this model is supported by the fact that this model can emulate the actions of an arbitrary adversary, and it is also very challenging because it gives advantage to the intruder as opposed to other models. Dolev-Yao is a very strong model because it assumes that the intruder can intercept every message in the channel and can build any message from the intercepted messages using for that infinite memory and processing capabilities. It is also based on perfect cryptography, which means that the intruder cannot decrypt a message M ciphered with a key K with another key K' different from K.

The section called local defines the local variables of Alice, which are State, which is described by a natural number (nat), and symmetric key Kab. Initial state of Alice is 0. The transition section describes received and sent messages and how they affect the state of the role. For instance the role server has following transition called step1:

```
step1.State = 0 /\ RCV({Kab'}_Kas) =|>
      State':= 2 /\ SND({Kab'}_Kbs)
```

The transition means that if the server's state is 0 and it receives a message from its RCV channel containing a key Kab' that is encrypted with a key Kas, the server changes its state to 2, encrypts the key Kab' with the Kbs and sends the encrypted key to the channel SND. In addition to basic roles the HLPSL language defines so called composition roles, which are used to combine several basic roles. Combining the basic roles means that the roles can execute in parallel. The composition roles define the actual protocol sessions. For instance, in the case of the WMF protocol there are three basic roles Alice, Bob and Server. The composition role, called session,

initiates one instance of each role and thus defines one protocol run. The composition role does not define transitions the way basic roles do, instead it initiates basic roles and defines channels used by the basic roles. The composition role is defined for instance in the following way:

```
role session(A,B,S :agent,
Kas,Kbs :symmetric_key) def=
local SA, RA, SB, RB SS, RS: channel (dy)
composition
alice (A, B, S, Kas, SA, RA)
/\bob (B, A, S, Kbs, SB, RB)
/\server(S, A, B, Kas, Kbs, SS, RS)
end role
```

Finally the HLPSL defines a top level role, called here as environment, which contains global variables and combines several sessions. This top level role can be used to define what information an intruder has and where the intruder can access the protocol. For example, the intruder may play a role of a legitimate user in a protocol run. The following role definition shows how a top level environment can be defined. The letter i in the definition indicates the intruder.

```
role environment()
def=
const  a, b, s : agent,
```

```
kas, kbs, kis : symmetric_key

intruder_knowledge = {a, b, s, kis}

composition

session(a,b,s,kas,kbs)

/\ session(a,i,s,kas,kis)

/\ session(i,b,s,kis,kbs)

end role
```

Every security protocol has some goals that it is supposed to meet. In order to write the protocol in HLPSL format, we must know these goals. The analysis is done against the defined security goals and the results indicate whether the protocol meets the goals or not. The security goals of the protocol are presented in an HLPSL language section called goals. Security goals are actually defined in transition sections of basic roles. The definitions of security goals in the transition section are called goal facts. The goals section simply describes which combinations of these goal facts indicate an attack [36]. Below there is an example of a goal fact. The notation means that Bob allows that the key K1 can be shared with Alice, but it must remain secret between the two. The second argument of the secret fact is called protocol id and it simply names the secret fact and distinguishes the different security goals from each other. [12]

```
role bob{

...

local
```

```
State : nat,

Nb,Na : text,

K1 : message

init

State := 1

transition

1. State = 1 /\ RCV({Na'}_K) =|>

State':= 3 /\ Nb' := new()

/\ SND({Nb'}_K)

/\ K1':= Hash(Na'.Nb')

/\ secret(K1',k1,{A,B})

...

end role
```

7.3 HLPSL2IF Translator

The HLPSL2IF translator translates HLPSL specification into an IF specification automatically.

- First it will check the number of conditions met by phrasing the HLPSL specification.
- Then, the role description in hierarchical structure is flattened and translated into step rules of IF.

- From the instantiation given in the HLPSL file, the initial state of IF is computed.
- Finally, the goals are computed as a state-based encoding of the properties given in the HLPSL file [17].

7.4 Back-ends

As Figure 12 shows, AVISPA integrates four different back-ends. Here the word back-end means an entity that inputs a sequence of IF language statements, does analysis and produces the analysis output. The four different back-ends used in AVISPA, OFMC, CL-AtSe, SATMC and TA4SP, are complementary rather than equivalent. Thus, the output of the back-ends may differ. All back-ends assume perfect cryptography, which means that an attacker cannot solve encryption without the knowledge of the whole key. Also, the transmission channel is assumed to be controlled by a Dolev-Yao attacker. This means that the attacker has basically full control over the channel [36].

7.4.1 The On-the-fly Model-Checker (OFMC)

OFMC [11] performs protocol classification and bounded validation by exploring the transition system described by an IF specification in a demand-driven way. OFMC implements a number of correct and complete symbolic techniques. It supports the

specification of algebraic properties of cryptographic operators, and typed and untyped protocol models.

7.4.2 The Constraint-Logic-based Attack Searcher (CL-AtSe)

CL-AtSe [11] applies constraint solving as in [10], with some powerful simplification heuristics and redundancy elimination techniques. CL-AtSe is built in a modular way and is open to extensions for handling algebraic properties of cryptographic operators. It supports type-flaw detection and handles associativity of message concatenation.

7.4.3 The SAT-based Model-Checker (SATMC)

SATMC [11] builds a propositional formula encoding a bounded unrolling of the transition relation specified by the IF, the initial state and the set of states representing a violation of the security properties. The propositional formula is then fed to a state-of-the-art SAT solver and any model found is translated back into an attack.

7.4.4 The Tree Automata based on Automatic Approximations for the Analysis of Security Protocols(TA4SP)

TA4SP [5] approximates the intruder knowledge by using regular tree languages and rewriting. For secrecy properties, TA4SP can show whether a protocol is flawed (by under-approximation) or whether it is safe for any number of sessions (by over-approximation).

7.5 Developing the HLPSL model

In this section we describe how we have transformed our model into HLPSL code to achieve the end results.

- Our model in HLPSL code has four basic roles. They are client, server, gateway, relay. The roles client and server serve as Pac and PAA respectively. As per our model, gateway and relay are acting as a friendly man-in-middle, they form SAs with client and server respectively and forward the EAP/PANA messages accordingly. The roles of the gateway and the relay is important because as we saw in Chapter 4, attacks are possible on both the gateway and the relay. So we consider all four roles as main actors in HLPSL.

- In the real world, there are large number of clients asking for a specific multicast application and they may request different multicast data streams as well. So, there is a need to distinguish all these clients and their requests. For that reason we have added constants such as *request-id* and *response-id*, which assign a random unique number for each request made by clients. We transferred these constants along with nonces of client and server in initial request messages.

- After few a initial messages, PANA starts carrying EAP method (EAP-FAST) for authentication. As EAP-FAST is already validated between two nodes in [30] and now we implement it among four nodes in our HLPSL code. As phase 1 in EAP-FAST results in shared key (SA) between two nodes, to make it simpler we introduced a shared key K1 between client, gateway and K2 between relay

and server. Client and gateway protects further data with K1 and relay and server with K2.

- After authentication all the four roles are able to calculate secret key (MSK). Using MSK and PANA nonces they calculate MAC (Message Authentication Code) value as well. Our goal is to maintain the secrecy of secret keys MSK, K1, K2. Section 7.7 describes more about goals. Derivation of secret key (MSK) and MAC is shown in figure 13 below.

```
% Calculation of Master Session key
Msk' := H(Nec'.Nes.Psk')

%Calculation of Message Authentication Code
 Mac' := INT(PRF(H(Nec'.Nes.Psk').Nps.Npc.Kid).Pmsg)
```

Figure 13: Derivation of Secret key and MAC

- After calculation of above mentioned keys, the results were passed to client from server.

- Session role defines executing of several basic roles in parallel. In our HLPSL code session role is composed of client, gateway, relay and server roles. Every role has two channels, send and receive, on which they send and receive messages. We should run these four roles in parallel for messages to pass through the AMT tunnel (See figure 14).

```
role session(
      C,G,R,S     :agent,
      K1,K2       :symmetric_key,
      H,PRF,INT :hash_func)
def=
local SC,RC,SG,RG,SR,RR,SS,RS :channel (dy)
composition
       client     (C,G,R,S,K1,H,PRF,INT,SC,RC)
    /\gateway(C,G,R,S,K1,H,PRF,INT,SG,RG)
    /\relay     (C,G,R,S,K2,H,PRF,INT,SR,RR)
    /\server    (C,G,R,S,K2,H,PRF,INT,SS,RS)
end role
```

Figure 14: Session Role

- In environment role, we can modify the number of parallel sessions and the knowledge of intruder. In our code intruder has given the knowledge of all the hash functions, agents and his own private key. First we executed a session without any intruder. In next step we executed session with client as intruder and then gateway, relay, server as intruders (See figure 15).

7.6 Validation of Our Model

In this section we describe how we modeled our architecture with HLPSL language to meet the AVISPA validation requirements.

- EAP is a versatile framework that facilitates multiple authentication mechanisms. It runs between a peer and a server. PANA is a network access authentication protocol that works as a link-layer (lower layer) protocol for transmitting

```
role environment()
def=
const c,g,r,s     : agent,
    kk1,kk2   : protocol_id,
    k1,k2,ki      : symmetric_key,
    h,prf,int : hash_func
intruder_knowledge = {c,g,r,s,h,prf,int,ki}

composition
        session(c,g,r,s,k1,k2,h,prf,int)
      /\session(i,g,r,s,ki,k2,h,prf,int)
      /\session(c,i,r,s,ki,k2,h,prf,int)
      /\session(c,g,i,s,k1,ki,h,prf,int)
      /\session(c,g,r,i,k1,ki,h,prf,int)
end role
```

Figure 15: Environment Role

EAP information. PANA carries EAP authentication methods (encapsulated inside EAP packets) between a peer node and a server in the access network.

- As described in chapter 5 End User (with EAP peer, Pac), AMT Gateway will reside on Unicast-only network and Server (with EAP server, PAA), AMT Relay will reside on Multicast-enabled network. Now we have to make sure that the same security considerations are met even if the EAP and PANA packets are passed through AMT Gateway and Relay.

- The HLPSL language is a role-based language, which means the action sequences of each participant are defined in a separate module, called a basic

role. Here, we define four general types of basic roles: the *role client*, *role server*, *role gateway* and *role relay* which describe the initial information they had and how their state can change.

- We defined global variables, local variables and constants for each basic role. Agents, hash functions, shared secret keys and channels are defined as global variables. Nonces, results are defined as local variables, which can be transferred between agents through the transmission channels.

- HLPSL local variables can be changed and transferred but cannot be shared. Yet, it is possible to share a constant value whenever we require roles to have pre-shared knowledge.

- In the *transition section* we defined set of transitions, each one represents the receipt of a message and sending of a reply message. We defined these transitions according to an order in which the messages are transferred in PANA, EAP protocols.

- Later we defined *composed roles*, which instantiate one or more basic roles by gluing them together so they execute together, usually in parallel (with interleaving semantics). A composed role instantiates one instance of each basic role and thus describes one whole protocol session [12].

- Finally, role environment, which is a top-level role, is defined. It contains global constants and a composition of one or more sessions, where the intruder may

play some roles as a legitimate user. We can define intruder's initial knowledge in a statement, which includes names of all the agents, his private key and hash functions, etc.

7.7 Security Goals

AVISPA allows us to define security goals [12] in HLPSL. It can be done by augmenting the transitions of the basic roles with so-called *goal facts.* Any kind of sensitive data that has to be exchanged needs AVISPA back-ends to track its confidentiality, integrity and safe transmission. Once we inform AVISPA about the goal facts, then we need to assign a meaning by describing them in the HLPSL goal section. This will clarify which combination of such facts indicates an attack.

The *witness* and *request* events are goal facts related to authentication of an agent. We used them to check whether or not an agent is right in believing that its intended peer that is actually present in current session, has reached a definite state in its transition section and agrees on a certain value, which usually is a fresh nonce. The fresh nonce is actually generated by each agent to authenticate the other one. HLPSL will support general security goals such as-

- **secrecy** - If we want to express a certain value should be kept secret between two or more agents, we use secrecy goal. It ensures the failure of an intruder to discover the message exchanged between two roles that is supposed to be kept secret.

- **authenticity** - it verifies a distinguishing identifier claimed by or for an agent, which may be a peer in a communication or the source of some data as a server. The verification is achieved presenting authentication information that collaborates the binding between the agent and the identifier.

7.7.1 Security Goals of our Model

In the goal section of our HLPSL code, we explicitly ask the AVISPA model checker to validate the secrecy of both the shared secret keys (K1, K2) and MSK, which ensures the intended security of further communications. Security goals are shown in figure 16

```
goal
%Secrecy of Shared Key between Client and Gateway
 secrecy_of kk1

%Secrecy of Shared Key between Relay and Server
  secrecy_of kk2

%Secrecy of Master Session Key
  secrecy_of s_msk
end goal
```

Figure 16: Security Goals

7.8 AVISPA Results

Considering the security goals mentioned in goal section of our HLPSL code, no attack has been found. Summary results of three AVISPA back-ends OFMC, CL-AtSe and SATMC appeared to be **safe.** This shows our model (Receiver Access Control in AMT) in reality is immune to all those potential attacks and threats.

Chapter 8

Conclusion and Future Work

In this thesis we have proposed a solution that provides receiver access control for multicast groups in the AMT environment. This solution allows only legitimate end users in a unicast-only-network to access networks and receive multicast data from multicast enabled sources. Later we explained how IGMP messages are secured with IPsec GSAs. We have also discussed how our model supports multiple sessions, by which an end user host can have multiple requests to different multicast groups and receive data simultaneously. Finally, we have modeled our solution by using the HLPSL description language. The validation result from AVISPA show our model is invulnerable to any kind of attacks.

Our model is based on the assumption that the AMT tunnel is secured. Currently, AMT tunnel has IGMP messages flowing through it; our design adds PANA, SIGMP packets to the tunnel. Future work for researchers might be taking measures to secure the Tunnel between Gateway and Relay, i.e., all the PANA, SIGMP messages and

multicast data going through AMT tunnel must be secured, so that no intruder can spoof the packets flowing through it.

Appendix A

HLPSL Source Code

```
role client(
C,G,R,S : agent,
K1      : symmetric_key,
H,PRF,INT : hash_func,
SND,RCV : channel(dy))

played_by C def=

  local
State : nat,
Npc,Nps,Nec,Nes : text,
Psk,Pmsg : text,
Result : text,
Kid : text,
Msk : hash(text.text.text),
Mac : hash(hash(hash(text.text.text).text.text.text).text)

  const s_msk,s_mac,kk1 : protocol_id,
request_id : text,
respond_id : text,
start_eap_fast : text
  init
State :=0

  transition
  1.  State = 0 /\ RCV(start) =|>
```

```
      State':=2 /\ SND(0)
  2.  State = 2 /\ RCV(Nps') =|>
      State':=4 /\ Npc' := new()
                /\ SND(Npc',request_id)
  3.  State = 4 /\ RCV(respond_id.S) =|>
      State':=6 /\ SND(start_eap_fast)
  4.  State = 6 /\ RCV({Nes'}_K1) =|>
      State':=8 /\ Nec' := new()
                /\ Psk' := new()
                /\ SND({Nec'}_K1,{Psk'}_K1)
                /\ Msk' := H(Nec'.Nes'.Psk')
  5.  State = 8 /\ RCV(Mac',Kid,Result) =|>
      State':=10/\ Mac' := INT(PRF(H(Nec.Nes.Psk).Nps.Npc.Kid).Pmsg)
                /\ SND(Mac', Kid)
                /\ secret(K1,kk1,{C,G})
                /\ secret(Msk,s_msk,{C,G,R,S})
                /\ secret(Mac',s_mac,{C,G,R,S})
end role
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
role gateway(
C,G,R,S : agent,
K1 : symmetric_key,
H,PRF,INT : hash_func,
SND,RCV : channel(dy))

played_by G def=

  local
State : nat,
Npc,Nps,Nec,Nes : text,
Psk,Pmsg : text,
Result : text,
Kid : text,
Msk : hash(text.text.text),
Mac : hash(hash(hash(text.text.text).text.text.text).text)

  const s_msk,s_mac,kk1 : protocol_id,
request_id : text,
respond_id : text,
start_eap_fast : text
  init
State :=10
```

```
  transition
  1.  State = 10 /\ RCV(0) =|>
      State':=12 /\ SND(0)
  2.  State = 12 /\ RCV(Nps') =|>
      State':=14 /\ SND(Nps')

  3.  State = 14 /\ RCV(Npc',request_id) =|>
      State':=16 /\ SND(Npc',request_id)
  4.  State = 16 /\ RCV(respond_id.S) =|>
      State':=18 /\ SND(respond_id.S)

  5.  State = 18 /\ RCV(start_eap_fast) =|>
      State':=20 /\ SND(start_eap_fast)
  6.  State = 20 /\ RCV(Nes') =|>
      State':=22 /\ SND({Nes'}_K1)
  6.  State = 22 /\ RCV({Nec'}_K1,{Psk'}_K1) =|>
      State':=24 /\ SND(Nec',Psk')
  7.  State = 24 /\ RCV(Mac',Kid,Result) =|>
      State':=26 /\ SND(Mac',Kid,Result)
  8.  State = 26 /\ RCV(Mac', Kid) =|>
      State':=28 /\ SND(Mac', Kid)
end role
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
role relay(
C,G,R,S : agent,
K2 : symmetric_key,
H,PRF,INT : hash_func,
SND,RCV : channel(dy))

played_by R def=

  local
State : nat,
Npc,Nps,Nec,Nes : text,
Psk,Pmsg : text,
Result : text,
Kid : text,
Msk : hash(text.text.text),
Mac : hash(hash(hash(text.text.text).text.text.text).text)

  const s_msk,s_mac,kk2 : protocol_id,
request_id : text,
respond_id : text,
```

```
start_eap_fast : text
  init
State :=11

  transition
  1.  State = 11 /\ RCV(0) =|>
      State':=13 /\ SND(0)
  2.  State = 13 /\ RCV(Nps') =|>
      State':=15 /\ SND(Nps')

  3.  State = 15 /\ RCV(Npc',request_id) =|>
      State':=17 /\ SND(Npc',request_id)
  4.  State = 17 /\ RCV(respond_id.S) =|>
      State':=19 /\ SND(respond_id.S)

  5.  State = 19 /\ RCV(start_eap_fast) =|>
      State':=21 /\ SND(start_eap_fast)
  6.  State = 21 /\ RCV({Nes'}_K2) =|>
      State':=23 /\ SND(Nes')
  6.  State = 23 /\ RCV(Nec',Psk') =|>
      State':=25 /\ SND({Nec'}_K2,{Psk'}_K2)
  7.  State = 25 /\ RCV(Mac',Kid,Result) =|>
      State':=27 /\ SND(Mac',Kid,Result)
  8.  State = 27 /\ RCV(Mac', Kid) =|>
      State':=29 /\ SND(Mac', Kid)
end role
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
role server(
C,G,R,S : agent,
K2 : symmetric_key,
H,PRF,INT : hash_func,
SND,RCV : channel(dy))

played_by S def=

  local
State : nat,
Npc,Nps,Nec,Nes : text,
Psk,Pmsg : text,
Result : text,
Kid : text,
Msk : hash(text.text.text),
Mac : hash(hash(hash(text.text.text).text.text.text).text)
```

```
  const s_msk,s_mac,kk2 : protocol_id,
request_id : text,
respond_id : text,
start_eap_fast : text
  init
State :=1

  transition
  1.  State = 1 /\ RCV(0) =|>
      State':=3 /\ Nps' := new()
                /\ SND(Nps')
  2.  State = 3 /\ RCV(Npc',request_id) =|>
      State':=5 /\ SND(respond_id.S)
  3.  State = 5 /\ RCV(start_eap_fast) =|>
      State':=7 /\ Nes' := new()
                /\ SND({Nes'}_K2)
  4.  State = 7 /\ RCV({Nec'}_K2,{Psk'}_K2) =|>
      State':=9 /\ Msk' := H(Nec'.Nes.Psk')
                /\ Mac' := INT(PRF(H(Nec'.Nes.Psk').Nps.Npc.Kid).Pmsg)
                /\ SND(Mac',Kid,Result)
                /\ secret(K2,kk2,{S,R})
end role
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
role session(
C,G,R,S  :agent,
        K1,K2     :symmetric_key,
        H,PRF,INT :hash_func)
def=
local SC,RC,SG,RG,SR,RR,SS,RS :channel (dy)
composition
      client (C,G,R,S,K1,H,PRF,INT,SC,RC)
      /\gateway(C,G,R,S,K1,H,PRF,INT,SG,RG)
      /\relay  (C,G,R,S,K2,H,PRF,INT,SR,RR)
      /\server (C,G,R,S,K2,H,PRF,INT,SS,RS)
end role
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
role environment()
def=
const c,g,r,s   : agent,
      kk1,kk2   : protocol_id,
      k1,k2,ki : symmetric_key,
      h,prf,int : hash_func
```

```
intruder_knowledge = {c,g,r,s,h,prf,int,ki}

composition
   session(i,g,r,s,ki,k2,h,prf,int)
%/\session(c,g,r,s,k1,k2,h,prf,int)
%/\session(c,i,r,s,ki,k2,h,prf,int)
  %/\session(c,g,i,s,k1,ki,h,prf,int)
  %/\session(c,g,r,i,k1,ki,h,prf,int)
end role
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
goal
   secrecy_of kk1
   secrecy_of kk2
   secrecy_of s_msk
end goal
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
environment()
```

Bibliography

[1] B. Aboba, L. Blunk, J. Vollbrecht, J. Carlson, and H.Levkowetz. Extensible authentication protocol (eap). Request for Comments 3748, Internet Engineering Task Force, http://tools.ietf.org/html/rfc3748, June 2004.

[2] J. William Atwood. An architecture for secure and accountable multicasting. *in 32nd IEEE Conference on Local Computer Networks (LCN 2007), PP.73-78*, October 2007.

[3] B. Blanchet. An efficient cryptographic protocol verifier based on prolog rules. *In Proc. CSFW01. IEEE Computer Society Press*, 2001.

[4] Y. Boichut and A. Armando. The avispa tool for the automated validation of internet security protocols and applications. Technical report, http://www.avispa-project.org/delivs/2.1/d2-1.pdf, August 2003.

[5] Y. Boichut, P-C Heam, and F. Oehl. Improvements on the genet and klay technique to automatically verify security protocols. *In Proc. Int. Ws. on Automated Verification of Infinite-State Systems (AVIS'2004), joint to ETAPS'04*, pages 1–11, April 2004.

[6] G. Bumgardner. Automatic multicast tunneling, internet draft. *IETF draft Work in progress, draft-ietf-mboned-auto-multicast*, April 2014.

[7] M. Burrows. Security protocols open repository, wide mouthed frog. 1989.

[8] B. Cain, S. Deering, I. Kouvelas, B. Fenner, and A. Thyagarajan. Internet group management protocol, version 3. Request for Comments 3376, Internet Engineering Task Force, http://tools.ietf.org/html/rfc3376, October 2002.

[9] N. Cam-Winget, D. McGrew, J. Salowey, and H. Zhou. The flexible authentication via secure tunneling extensible authentication protocol method (eap-fast). Request for Comments 4851, Internet Engineering Task Force (IETF), https://ietf.org/doc/rfc4851/, May 2007.

[10] Y. Chevalier and L. Vigneron. Automated unbounded verification of security protocols. Research Report 4369, INRIA, July 2002.

[11] L. Compagna and A. Armando. Sat-based model-checking for security protocols analysis. *International Journal of Information Security*, pages 7:3–32, January 2008.

[12] J. Cuellar and P. Hankes. A beginner's guide to modelling and analysing internet security protocols. Technical report, http://www.avispa-project.org/package/tutorial.pdf, June 2006.

[13] P. Duffy, A. Yegin, Y. Ohba Ed, S. Chakrabarti, and R. Cragie. Protocol for carrying authentication for network access (pana) relay element. Request for Comments 6345, Internet Engineering Task Force (IETF), https://ietf.org/doc/rfc6345/, August 2011.

[14] V. Fajardo, J. Arkko, J. Loughney, and G. Zorn. Diameter base protocol. Request for Comments 6733, Internet Engineering Task Force, http://tools.ietf.org/html/rfc6733, October 2012.

[15] W. Fenner. Internet group management protocol, version 2. Request for Comments 2236, Internet Engineering Task Force, http://tools.ietf.org/html/rfc2236, November 1997.

[16] D. Forsberg, Y. Ohba, B. Patil, H. Tschofenig, and A. Yegin. Protocol for carrying authentication for network access (pana). Request for Comments 5191, Internet Engineering Task Force, http://tools.ietf.org/html/rfc5191, May 2008.

[17] P-C. Heam, J. Santiago, and L. Vigano. The intermediate format. Technical report, http://www.avispa-project.org/delivs/2.3/d2-3.pdf, June 2003.

[18] S. Islam and J. William Atwood. Sender access control in ip multicast. *in Procedings 32nd Annual Conference on Local Computer Networks (LCN 2007), Dublin, Ireland*, October 2007.

[19] S. Islam and J. William Atwood. Multicast receiver access control using pana. *Proceedings of the 1st Taibah University International Conference on Computing and Information Technology (ICCTT 2012)*, March 2012.

[20] S. Kent. Ip encapsulating security payload (esp). Request for Comments 4303, Internet Engineering Task Force, http://tools.ietf.org/html/rfc4303, December 2005.

[21] S. Kent and R. Atkinson. Ip authentication header (ah). Request for Comments 4302, Internet Engineering Task Force, http://tools.ietf.org/html/rfc4302, December 2005.

[22] S. Kent and K. Seo. Security architecture for the internet protocol. Request for Comments 4301, Internet Engineering Task Force, http://tools.ietf.org/html/rfc4301, December 2005.

[23] B. Li. User guide for project diameter+pana+eap-fast. October 2013.

[24] B. Li and J. William Atwood. Receiver access control for ip multicast at the network level. *Submitted to Computer Networks*, August 2014.

[25] J. Liebeher and M. Zark. *Mastering Networks An Internet Lab Manual*. Third edition, August 2003.

[26] D. Minoli. *IP Multicast with Applications to IPTV and mobile DVB-H*. Hoboken, N.J, August 2008.

[27] L. Morand, A. Yegin, S. Kumar, and S. Madanapalli. Dhcp options for protocol for carrying authentication for network access (pana) authentication agents. Request for Comments 5192, Internet Engineering Task Force (IETF), https://ietf.org/doc/rfc5192/, May 2008.

[28] Y. Nir, C. Kaufman, P. Hoffman, and P. Eronen. Internet key exchange (ikev2) protocol. Request for Comments 5996, Internet Engineering Task Force, http://tools.ietf.org/html/rfc5996, September 2010.

[29] Y. Ohba and A. Yegin. Definition of master key between pana client and enforcement point. Request for Comments 5807, Internet Engineering Task Force, http://tools.ietf.org/html/rfc5807, March 2010.

[30] M. Parham. Validation of the security of participant control exchanges in secure multicast content delivery. Master's thesis, Department of Computer Science and Software Engineering, Concordia University, September 2011.

[31] L. C. Paulson. The inductive approach to verifying cryptographic protocols. *Journal of Computer Security, 6(1):85128*, 1998.

[32] C. Rigney, S. Willens, A. Rubens, and W. Simpson. Remote authentication dial in user service (radius). Request for Comments 2865, Internet Engineering Task Force, http://tools.ietf.org/html/rfc2865, June 2000.

[33] S. Rowles, A. Yeung, P. Tran, and Y. Nir. Group key management using ikev2, internet draft. *IETF draft Work in progress, draft-yeung-g-ikev2*, April 2013.

[34] A. Salem. Formal validation of security properties of amt's three-way handshake. Master's thesis, Department of Computer Science and Software Engineering, Concordia University, July 2011.

[35] D. Song. Athena: A new efficient automatic checker for security protocol analysis. *In Proc. CSFW99. IEEE Computer Society Press*, 1999.

[36] L. Vigano. Automated security protocol analysis with the avispa tool. *Electronic Notes in Theoretical Computer Science (ENTCS)*, pages 155:61–86, May 2006.

[37] B. Weis, G. Gross, and D. Ignjatic. Multicast extensions to the security architecture for the internet protocol. Request for Comments 5374, Internet Engineering Task Force, http://tools.ietf.org/html/rfc5374, November 2008.

[38] B. Weis, S. Rowles, and T. Hardjono. The group domain of interpretation. Request for Comments 6407, Internet Engineering Task Force, http://tools.ietf.org/html/rfc6407, October 2011.

[39] B. Williamson. *Developing IPMulticast Networks- The Definitive guide to designing and deploying Cisco IP multicast networks*. Cisco Press, January 2000.

[40] A. Yegin, Y. Ohba, R. Penno, and C. Wang. Protocol for carrying authentication for network access (pana) requirements. Request for Comments 4058, Internet Engineering Task Force, http://tools.ietf.org/html/rfc4058, May 2005.