# FORMAL VALIDATION OF SECURITY PROPERTIES OF AMT'S THREE-WAY HANDSHAKE

Ali Salem

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

April 2011

Concordia University

School of Graduate Studies

This is to certify that the thesis prepared

|  |  |
|---|---|
| By: | Ali Salem |
| Entitled: | Formal Validation of Security Properties of AMT's Three-way Handshake |

and submitted in partial fulfillment 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. D. Goswami

______________________ Examiner
Drs. L. Narayanan

______________________ Examiner
Dr. J. Opatrny

______________________ Supervisor
Dr. J.W. Atwood

Approved ______________________
Chair of Department or Graduate Program Director

______________________

Dr. Robin A. L. Drew, Dean

Faculty of Engineering and Computer Science

Date ______________________

# Abstract

Formal Validation of Security Properties of AMT's Three-way Handshake

Ali Salem

Multicasting is a technique for transmitting the same information to multiple receivers over IP networks. It is often deployed on streaming media applications over the Internet and private networks. The biggest problem multicast introduces today is that it is an "all or nothing" solution. Every element on the path between the source and the receivers (links, routers, firewalls) requires multicast protocols to be enabled. Furthermore, multicast has a conceptual business model, and therefore is not an easy case to make. These factors, embedded deep in technology, but ultimately shaped by economics, led to a lack of multicast deployment. To address this problem, the AMT (Automatic IP Multicast without explicit Tunnels) specification has been developed by the Network Working Group at the IETF. This specification is designed to provide a mechanism for a migration path to a fully multicast-enabled backbone. It allows multicast to reach unicast-only receivers without the need for any explicit tunnels between the receiver and the source. We have formally validated the three-way handshake in the AMT specification using AVISPA against two main security goals: secrecy and authentication. We have demonstrated that the authentication goal is not met: an attacker can masquerade as an AMT relay, and the AMT gateway (at the end user) cannot distinguish a valid relay from an invalid one. Another attack was also found where an intruder can disconnect or shutdown a valid session for a valid end-user using a replay attack.

# Acknowledgments

The author wishes to express his deepest gratitude to his supervisor Dr. J. William Atwood for his guidance, advice, criticism, encouragement and insight throughout the research. The author would also like to thank his mother Zohra Jaafar, his father Mohamad Salem and the rest of his family members for their endless support and patience. And last, he wishes to thank his friends for their encouragement, love, and support.

# Contents

# List of Figures

# Chapter 1

# Introduction

In IP multicast, applications send one copy of a packet and address it to a group of receivers (at the multicast address) that want to receive it rather than to a single receiver (for example, at a unicast address). Multicast depends on the network to forward the packets to only those networks and hosts that need to receive them, therefore controlling network traffic and reducing the amount of processing that hosts have to do. Multicast applications are not limited by domain boundaries but can be used throughout the entire Internet [21]

The biggest problem multicast presents today is an "all or nothing" solution [27]. Every link on the network, every router and firewall between source and receiver, requires multicast protocols to be enabled. Additionally, the business model for multicast is abstract and is not an easy case to make. Multicast is an infrastructure capability that enables other services. From a business perspective, multicast resembles DNS and BGP, which are vital infrastructure protocols that are generally not billed directly. Consequently, those service providers who tried to bill for Internet multicast found disappointing results. Content providers were not interested in paying extra to transmit multicast streams that could not be received by many end users, and networks with many end users were unwilling to pay extra to receive multicast content that did not exist. The result was a chicken-and-egg problem between content and audience.

To overcome this problem, a transition strategy was proposed by the Network Group at The Internet Engineering Task Force (IETF) called Automatic IP Multicast Without Explicit Tunnels (AMT) [31]. This involves setting up relays at peering points in multicast networks that can be reached from gateways installed on hosts connected to unicast networks. With AMT, multicast control messages are encapsulated in User Datagram Protocol (UDP) packets and sent to relays, which then transmit them natively toward the source. This results in moving the replication point closer to the user, and cuts down on traffic across the transit provider's network. Thus, the content owner can more easily see a profitable business model, by avoiding the linear costs of adding unicast subscribers. 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. AMT is an interim solution to help build scalable video and other multicast services during the transition to multicast-enabled local service providers.

Our work is to formally validate the AMT specification. All possible events of protocol communication are examined, and then all the resulting possibilities are explored systematically against two main security goals between concerned parties: secrecy and authentication. As no previous attempts have been publicly made to assess any sort of such validation of AMT, we believe that our work will be regarded as a useful initiative in this direction.

The organization of the thesis is as follows:

- Chapter 2 gives a brief background of IP Multicasting, and highlights a brief comparison with Unicast and Broadcast. We also look at the benefits multicasting provides to the content providers as well as the end users, in addition to the main obstacles challenging Internet Service Provider from deploying this technology in their infrastructure. We also provide an overview of the AMT specification, its applicability in today's world, its components, and how it can act an interim solution.

- Chapter 3 discusses a protocol analysis tool called AVISPA. This chapter describes the architecture of the tool and syntax for protocol specification language called HLPSL. It also gives an example of how a real protocol can be specified with the HLPSL language and how the output of the AVISPA Tool is analysed.
- Chapter 4 discusses the implementation of formal validation of AMT under AVISPA. The results of the validation are shown and explained.
- Chapter 5 summarizes the thesis, concludes with comments on the validation results of the AMT protocol and states some future work directions.

# Chapter 2

# Background

## 2.1 IP Multicast

### 2.1.1 Definition

The greater part of Internet traffic uses unicast data delivery [20]. In unicast, a client requests data and a server transmits the data directly to it. Each client requesting data receives its own copy of the stream from the server. As the number of clients increases, the cost of unicast delivery increases in a linear fashion accordingly. This in return requires the server to be powerful enough to transmit a duplicate stream to every interested client, and the links on the network must have adequate bandwidth to handle all the duplicate streams.

Standing out against unicast delivery, broadcast data delivery on the other hand allows a server to send a single stream to the network, which will be received by all end users on the network, regardless of their interest in the data. For example, a radio station broadcasts its signal to all radios within a given area regardless of the number of listeners. The benefit of this, and broadcasting traffic in general, is very clear for the owner of content - whether there is one interested listener or 1 million interested listeners, the cost to transmit remains the same. The disadvantage is that the traffic is sent to all listeners, whether they are interested or not [20].

By comparing unicast and broadcast side to side, especially when it comes to suitability of employment over the Internet, we can easily conclude that unicast data delivery is better suited than broadcast. Unicast delivery is intended for users who are explicitly requesting it, but not the rest. On the other hand, broadcast will deliver the content to all users on the network, and in the world of Internet, we could be tackling about millions and millions of users. Hence, unicast delivery works more efficiently over the Internet.

Between the two extremes of unicast and broadcast lies a third option: multicast. In multicast, the source transmits a single stream of data. Unlike unicast or broadcast, the network intelligently determines where that stream of content is desired and delivers the stream only to interested users. By delivering a single stream of data, a multicast source enjoys the same level of efficiency as broadcast with respect to cost of transmission remaining constant whether the number of receivers is one person or one million people. Additionally, by delivering that content only to interested users, multicast enjoys a similar level of efficiency to unicast with respect to traffic not being forwarded to uninterested users. Multicast enjoys the best of both worlds by grasping the benefits of unicast and broadcast without enduring their deficiencies.

In multicasting, senders send each data packet once and at most one copy of each packet flows through the physical links under normal conditions. For example, assume that a sender, S, wants to send a message to receivers R1 and R2, as shown in Figure 1. In case of unicast transmission, S should transmit the same data twice and the bandwidth usage between the sender and the intermediate node is doubled. In broadcasting, other receivers such as R3 will get the packets although they are not relevant to R3, causing unnecessary bandwidth consumption. But in multicasting, only a single copy of the message is transmitted from the sender and it is copied at the intermediate node to be sent to the multicast group. A multicast group can range in size from a few nodes to several thousands. In the example given in Figure 1, the multicast group consists of nodes R1 and R2.

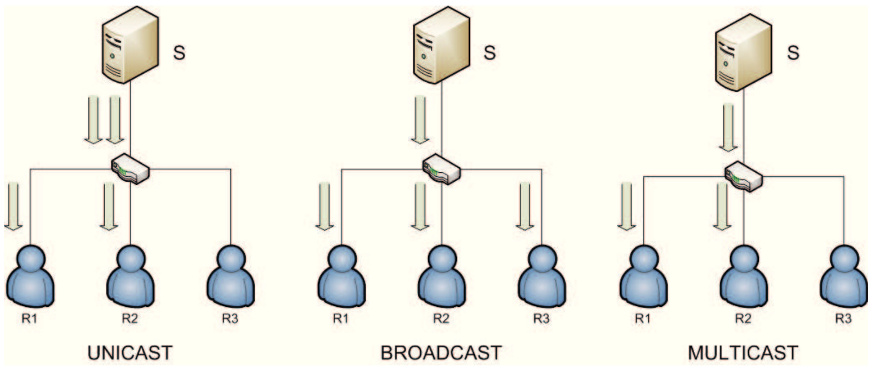

Figure 1: Unicast, broadcast and multicast transmissions

Some applications that enjoy the benefits of multicast today are:

- Multicast Reflector, at University of Oregon.
- IPTV at Cisco Systems.
- Products at Real Networks.
- Windows Media Technologies, Conference XP at Microsoft.
- MacTV at Apple.
- VideoCharger at IBM.

### 2.1.2 Challenges with Multicast

As much as multicast looks tempting in theory, it is unfortunately not deployed widely and actively in reality. The present problems with recognizing a multicast service are considered multidimensional [24], involving several different players:

- Users simply want to have access to the content and are not interested in the delivery method, e.g., unicast or multicast.
- Internet Service Providers want to be compensated for delivering extra services. There has typically been no way of easily monetizing these services. Consequently, turning multicast into a commercial service has been highly impeded.

- Content Providers love multicast because in reality they pay less money for delivering more.

Figure 2 demonstrates these problems in high-level terms.

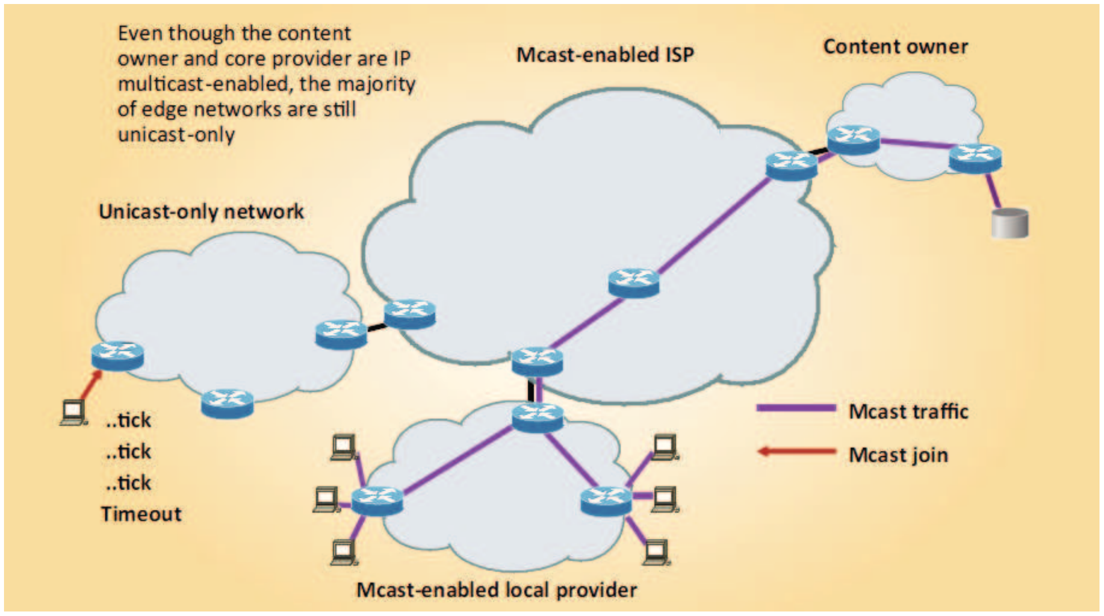
Even though the content owner and core provider are IP multicast-enabled, the majority of edge networks are still unicast-only
Mcast-enabled ISP
Content owner
Unicast-only network
..tick
..tick
..tick
Timeout
Mcast traffic
Mcast join
Mcast-enabled local provider

Figure 2: Content delivery in today's IP environment [20]

## 2.2 Automatic Multicast without Tunneling

### 2.2.1 History

The dynamics of content delivery over the Internet have tilted towards the wide scale use of unicast. Even a content owner with connection to a native multicast-enabled ISP is required to offer both unicast and multicast content. This has came about since receivers are classically connected to unicast only last-mile network environments. This results in high bandwidth costs per stream for the content owner and all service providers in the delivery chain (see Figure 3).

When hosts need to show interest in receiving multicast content, they send a message into the network to indicate that they are interested in a particular multicast

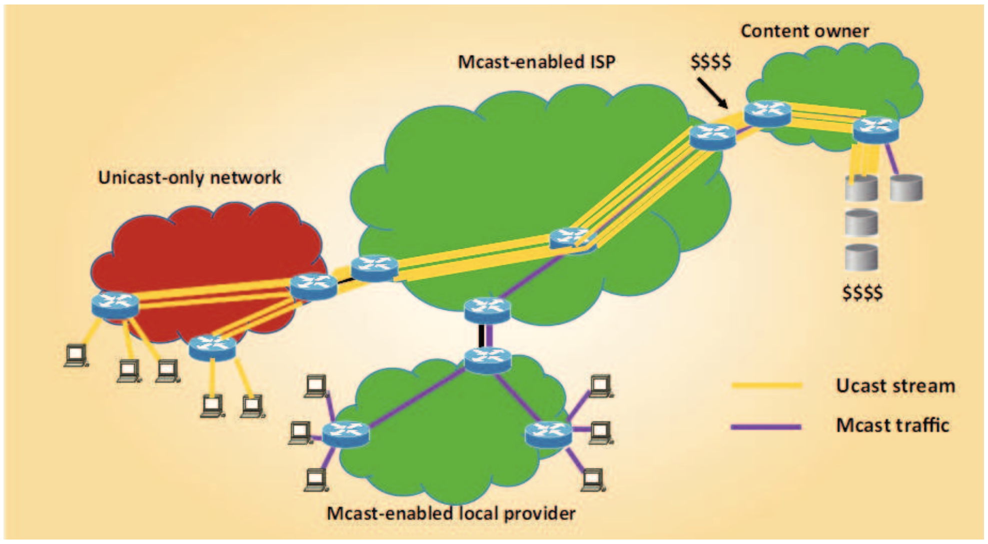

Figure 3: Unicast Only Last Mile Environment [20]

group. This message is called an IGMP membership report. This message is received by the first hop router next to the receiver. In a unicast-only network, the router will discard such message since it lacks multicast functionality. As a result, the receiver application in the host is compelled to send unicast requests instead for the same content. The output of this behavior was:

- Content owners made more money by reaching more end-users through unicast content delivery.
- The ISP made more money by allowing the content owner to transmit multiple copies of the stream.

Consequently, the following resulted:

1. People grew familiar with unicast data delivery (through the likes of Youtube and BBC iPlayer [24]).
2. Service providers no longer considered the requirement to deploy multicast, mostly due to lack of business justification.
3. Additional rise of unicast delivery models took place.

4. Multicast became more and more niche.

All of these factors, embedded deep within technology, but ultimately directed by economics and business justification, led to a lack of deployment of multicast technology. As a result, AMT [31] (Automatic IP Multicast without explicit Tunnels) has been developed.

This specification is intended to offer a migration path to a fully multicast-enabled backbone and allow multicast to reach receivers in networks with no multicast capabilities, without the requirement for any pre-configured tunnels between the receiver and the source. It provides the benefits of multicast where multicast is deployed. More details about the AMT specification are provided in the following section.

### 2.2.2 Definition

Automatic Multicast Tunneling (AMT) allows multicast communication to take place among remote multicast-enabled sites or hosts, attached to a network that has no native multicast support. Without requiring any manual configuration, AMT allows the hosts to exchange multicast traffic with the native multicast infrastructure. AMT operates with an encapsulation interface so that no future modification to applications is required, all protocols (not just UDP) are handled, and there is no extra overhead in core routers [31].

### 2.2.3 Components

*The following terminology is largely adapted from draft-ietf-mboned-auto-multicast-10* [31].

**AMT Site** is a multicast network (or host) with an attached / resident gateway served by an AMT Gateway. It could also be a standalone AMT Gateway.

**AMT Relay** is usually a multicast router configured to support transit routing between AMT Sites and the native multicast backbone infrastructure. The relay router has one or more interfaces connected to the native multicast infrastructure, zero or more interfaces connected to the non-multicast capable inter-network, and an AMT pseudo-interface. This device terminates one end of an AMT tunnel and encapsulates multicast packets into those tunnels. While usually a router, it may be a standalone server. Put more simply, an AMT Relay receives AMT Requests from an AMT Gateway.

**AMT Gateway** is a host, or site gateway router, supporting an AMT Pseudo-Interface. It does not have native multicast connectivity to the multicast backbone infrastructure. This device terminates the other end of an AMT tunnel and de-encapsulates multicast packets from those tunnels. Put more simply, an AMT Gateway sends AMT Requests to the AMT Relay. AMT Gateways are expected to be implemented in two ways:

- In a network device (home gateway, router).
- In a host (standalone software or built into an application).

**AMT Pseudo-Interface** is a point logically equivalent to an interface where AMT encapsulation (of multicast packets inside unicast packets) occurs. Some implementations may treat it exactly like any other interface and others may treat it like a tunnel end-point. In most (if not all) AMT implementations, the pseudo-interface will be a tunnel end-point.

Using these definitions, we can have a better understanding of how AMT works when we explain it next. Initially, let us assume that the multicast-enabled ISP provides the AMT Relay service (Figure 4).

In this diagram, the hosts connected to the unicast-only network are acting as AMT Gateways. Now let us consider the following scenario when some of these hosts

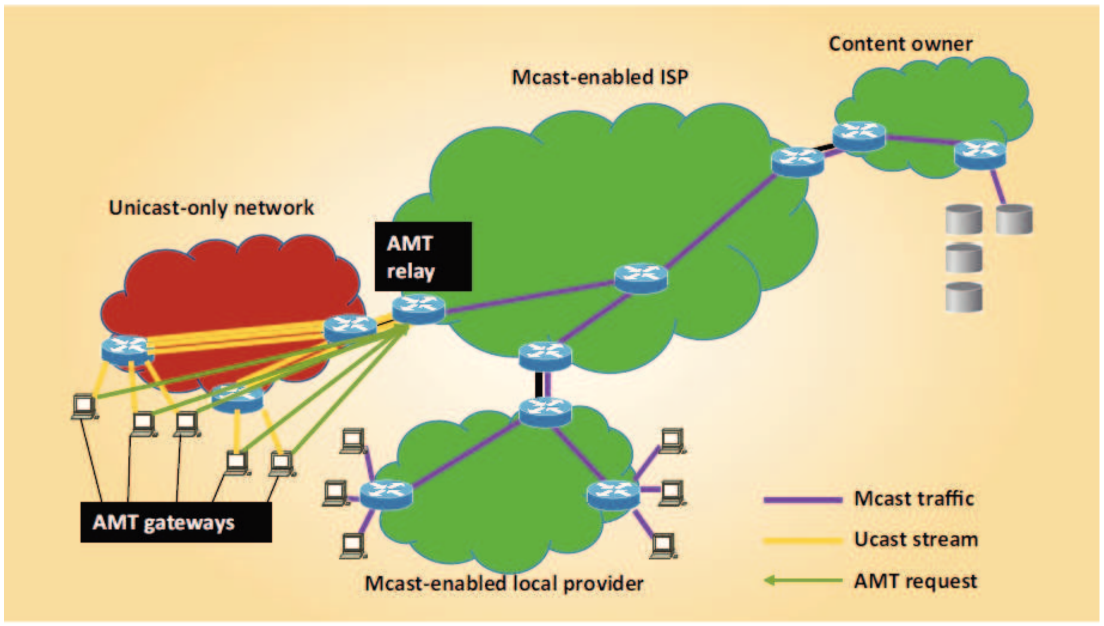

Figure 4: Multicast Enabled ISP Providing AMT Service [20]

want to request content:

- They first send an IGMP membership report to the first hop router;
- The AMT Gateway process running on the host will seize the IGMP report and trigger an AMT request towards an AMT Relay;
- An AMT tunnel will be created between the Relay and the Gateway through a 3-way handshake process (explained later in this section in more detail);
- Since an AMT tunnel is now created, the host will encapsulate the IGMP membership report into it and send it to the AMT Relay;
- The AMT Relay will decapsulate the IGMP message and trigger an upstream PIM join [19] towards the source;

A migration path for the unicast-only network now exists for it to become multicast-enabled. It could start by moving the relay into its network domain and creating a multicast peering with the upstream ISPs. Then, to further minimize the bandwidth load, it can gradually push multicast capabilities down through the network, into the first-hop routers, removing the need for the host-based AMT Gateways.

### 2.2.4 AMT Relay discovery

This section will explain how the AMT Gateway locates the AMT relay. According to [31], we need to assign an address to the Relay that is recognized throughout the Internet. In an IP network, one way of providing this function is via an Anycast Address. Ultimately, it is anticipated there will be an allocated IANA Anycast address for the AMT Anycast prefix [24]. Currently the prefix is provided by ISC [1] (via 154.17.0.0/16). Each ISP with an AMT Relay needs to promote this address as reachable throughout the Internet. To search and locate the unicast address of the nearest AMT Relay, the AMT Gateway sends a message called AMT Relay Discovery message destined to the AMT Anycast Address. The message is sent to the reserved UDP port 2268 and includes a special code (or Nonce), which is used to secure the set up of the tunnel (Figure 5).

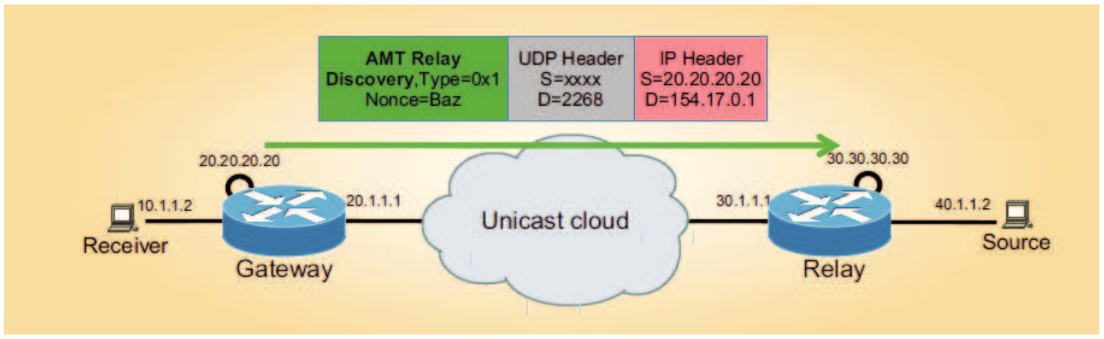

Figure 5: AMT Relay Discovery Message [20]

On receipt of an AMT Relay Discovery message, the Relay will respond to the Gateway with an AMT Relay Advertisement message, which includes the Relay's unique IP address. This unique IP address will then be used by the Gateway for communication with the Relay with any future AMT message (AMT Requests and AMT Membership Updates). Once again the reserved UDP port 2268 is used and the reply also contains the same Nonce that was originated by the Gateway. As a result, the Gateway knows that this reply was a reply to its Discovery message (Figure 6).

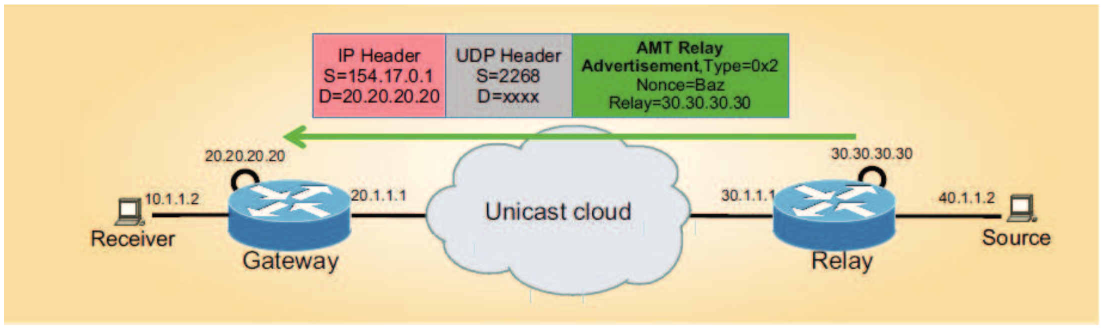

Figure 6: AMT Relay Advertisement Message [20]

On receipt of the AMT Relay Advertisement Message, the Gateway begins the "3 way handshake" by sending an AMT Request message to the Relay using the relay's unique IP address as the destination (again along with a new Nonce) (Figure 7).

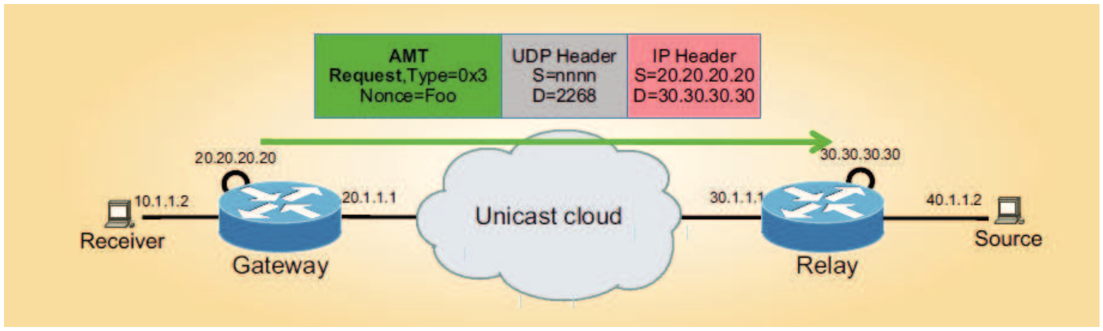

Figure 7: AMT Relay Request Message [20]

The Relay responds with an AMT Query that includes the new Nonce from the AMT Request, as well as an opaque security code (MAC) that it will expect in any future messages from the Gateway. The AMT query in fact encapsulates the underlying IGMP membership query and includes the Querier's Query Interval Code (QQIC), which specifies the Query Interval used by the querier (Figure 8).

To join any upstream sources, the Gateway responds with an AMT Membership Update that includes the opaque security code, the original nonce from the AMT Request, and an encapsulated IGMPv3 packet (Figure 9).

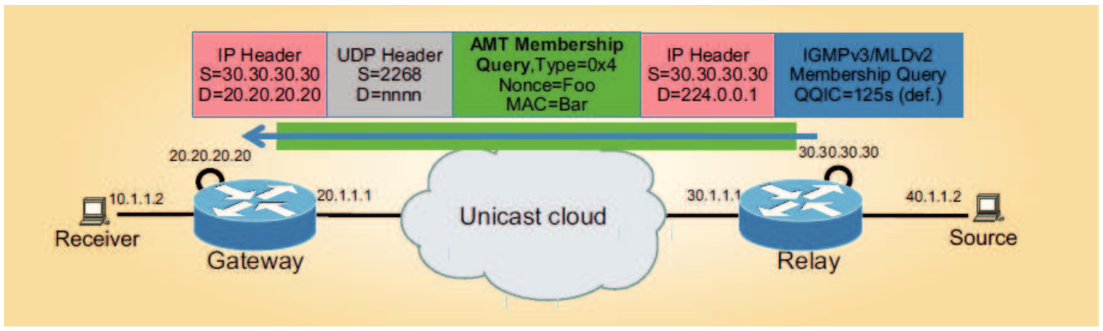

Figure 8: AMT Membership Query Message [20]

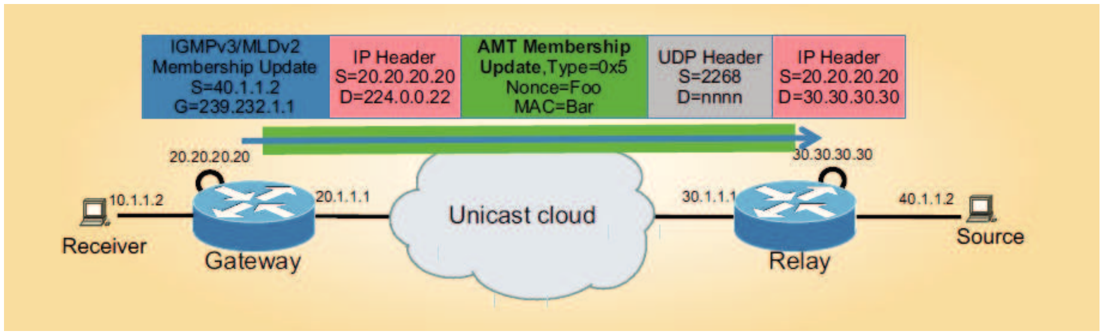

Figure 9: AMT Membership Update Message [20]

By validating the security code and Nonce, the Relay completes the tunnel set up and starts using it for multicast traffic. The Relay adds the appropriate pseudo/tunnel interface to the multicast route for that particular stream and begins duplicating and encapsulating packets to the Gateways (Figure 10).

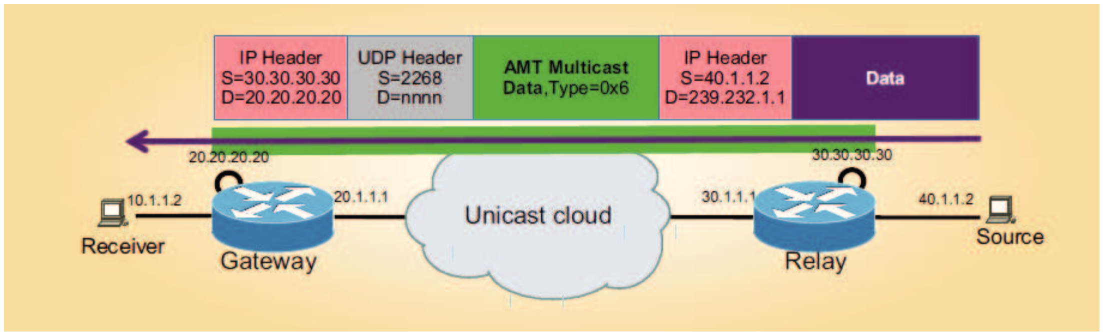

Figure 10: AMT Multicast Data Transfer [20]

Any further streams will use the same Request/Query/Update "3-way handshake" (but will not need to use the Discovery/Advertisement process since the tunnel will already have been established). If any Request does not receive a Query in response, the Gateway will then use the Discovery/Advertisement mechanism to find the next available Relay. Once the tunnel has been established, the communication is effectively identical to a normal router host IGMPv3 relationship. The Gateway (host) sends periodic AMT Membership Updates to refresh the state on the Relay (router), sending the appropriate update to leave the group when the traffic is no longer desired. Once the tunnel is no longer required by any more receivers it is maintained by the Gateway/Relay for a further time-out period. In that way a new receiver does not need to build a new tunnel if that receiver becomes active again shortly afterwards.

### 2.2.5 AMT Benefits

Using AMT, an ISP can benefit by deploying a single AMT relay to provide service to all its customers. Before that relay becomes overloaded, or if the tunnel traffic it generates causes an undue load on certain links in the network, the ISP can deploy additional AMT relays, both to distribute the server load and to lessen the network load. Other benefits offered by AMT to the IP broadcasting industry for delivering content are listed below.

**Simplicity** : To establish the AMT tunnel, the receiving network simply sends out an AMT Advertisement message to a well known Anycast Prefix. The rest of the tunnel establishment is done automatically without the need for any additional configuration or overhead of manual monitoring.

**Resiliency** : Since the Relay discovery uses an Anycast address, Gateways will automatically find the closest Relay. Should a Relay become unavailable or unreachable, the routing table will automatically update itself and use the next closest Relay.

**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.

## 2.3 Three-way Handshake

The term “three-way handshake” is usually associated with the TCP 3-way handshake process where it is used to establish and tear down socket reliable connections over the network. TCP handshake is required for both communicating parties to setup initial sequence numbers (required to allow the communication to be reliable) and ensure that they have both understood each other. UDP, which is employed by AMT, on the other hand, does not involve having a reliable connection between the communicating parties. According to the AMT specification [31], the 3-way handshake, which is used to establish the tunnel, is used only to help avoid spoofing and denial of service (DoS) attacks between the gateway and the relay for each multicast join or leave. However, our formal validation results helped prove that the 3-way handshake process is in fact prone to spoofing or impersonation by a third party.

The manner of establishing the tunnel in AMT is different from the ways it is being established in other security protocols, especially the ones modelled under AVISPA. The IKEv2 protocol [23], establishes a secure tunnel between its hosts and gateway using the DHCP-IPSec-tunnel, an exhange that is secured using IPsec, to guarantee authentication and integrity between participating parties. PEAP [6] and EAP-TTLS [28] make use of the handshake phase in TLS [28] to establish a secure tunnel where the identities of the communicating parties are exchanged and authenticated through this tunnel. TLS uses symmetric cryptography for privacy and a keyed message authentication code for message reliability.

The IKEv2-MAC protocol [23], a variant of IKEv2, uses an authentication method which involves exhanging the MAC of a pre-shared secrect that both partcipating nodes possess. As we will see later, the absence of proper MAC authentication in AMT leads to a security problem. In Mobile IP networks, a strong authentication scheme is used for security purposes. Integrity of tunneling and registration messages between a mobile node, foreign agent, and home agent [15] are protected with the help of a preshared 128-bit key. Mobile IP uses a tunneling protocol [15] to allow messages from the packet data network PDN to be directed to the mobile node's IP address.

By closley examining the validation results of the existing models of IKEv2, IKEv2-MAC, PEAP, and EAP-TTLS, and Mobile IP under AVISPA, and comparing them with the one of AMT, we notice major differences in the ways tunnels are set up and secured, in addition to the strong authentication schemes offered by these protocols. AMT's tunnel establishment and authentication scheme is not followed by any of the other protocols modelled under AVISPA. By modelling the AMT's three-way handshake process under AVISPA and comparing it with other modelled protocols, we were able to indicate that there is an absence of a proper authentication scheme between the participating AMT entities, Relay and Gateway, in addition to lack of proper security in the three-way handshake tunnel establishment.

# Chapter 3

# Design & Methodology

## 3.1 Formal Analysis

When a secure routing protocol is being developed, it is important to be able to prove that it fulfills the anticipated security properties. This step is a prerequisite to implementing the solution because it allows us to avoid future problems in the design level. It is usually carried out with mathematical calculations or using some specialized tools based on logic calculation such as BAN logic and GNY logic [22]. One of the most used techniques is formal analysis. Formal analysis is based on a mathematical modeling of the analyzed system [8]. Modeling the system implies modeling its components. Formal analysis is in fact a formal validation process. Primarily, the system that needs to be analyzed is first formalized (modeled) according to previously established assumptions. Afterwards, the anticipated properties of the system are also formalized. The actual analysis step follows, in which, based on the assumptions made and on the models of the system and of the properties, the validity of the properties is established. It is important to note that the results highly depend on the model under consideration.

Formal validation of security protocols is of great importance before they gain market or academic acceptance. Some standard and widely used security protocols for the Internet have been proved to suffer from critical design flaws that an attacker can exploit to overthrow their security. The reason is that their security goals were simply informally evaluated, creating potential attack paths as a result. Automated reasoning techniques are commonly used to evaluate the protocols in a formal way, increasing the assurance respecting the purported security. Automated or semi-automated tools were developed such as Casper/FDR, Murö, Athena and AVISPA [9][17][26][29][3].

We have chosen the model checker AVISPA tool for our formal analysis because it is powerful, easy to use and open source. It is a project sponsored by the European Union to validate the security goals of different protocols. Already 85% of the IETF protocols were proven by this tool, which has demonstrated its capabilities [22]. More information on AVISPA is provided in the next section.

## 3.2 AVISPA

To quicken the development of the protocols and enhance their security, it is important to have appropriate tools that support the analysis of the protocols and help to find the vulnerabilities in the early stages of development [32]. 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 [3]. A number of (semi-)automated protocol analysis tools have been proposed, e.g., [4][9][17], which can analyze small and medium-scale protocols such as those in the Clark/Jacob library [16]. However, scaling up to large scale Internet security protocols is a considerable challenge, both scientific and technological. As a result, a push-button tool for the Automated Validation of Internet Security-sensitive Protocols and Applications has been developed, the AVISPA Tool [32].

The architecture of AVISPA is shown in figure 11. The first step in using the tool is to present the analyzed protocol in a special language called High Level Protocol

Specification Language (HLPSL) [13]. We discuss the HLPSL language in more detail in the following section.

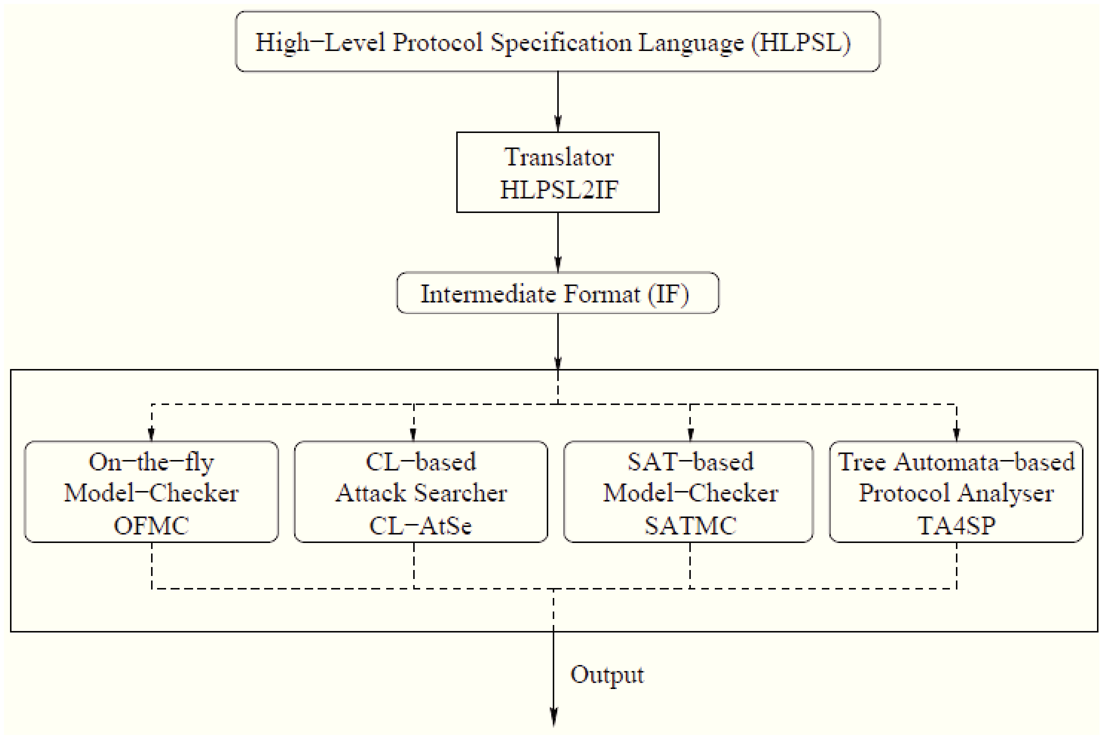
High–Level Protocol Specification Language (HLPSL)
Translator HLPSL2IF
Intermediate Format (IF)
On–the–fly Model–Checker OFMC
CL–based Attack Searcher CL–AtSe
SAT–based Model–Checker SATMC
Tree Automata–based Protocol Analyser TA4SP
Output

Figure 11: The Architecture of AVISPA[32]

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. IF presentation of the protocol is used as an input to the four different back-ends: On-the-fly Model-Checker (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 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 3.4.

## 3.3 High Level Protocol Specification Language

AVISPA uses High Level Protocol Specification Language (HLPSL) to present the analyzed protocols. In this section we take a closer look into the structure of HLPSL language according to the AVISPA tutorial [32]. 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 Wide Mouth Frog (WMF) protocol [2], 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 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 to 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) [18]. Support for other intruder models such as algebraic intruder model [11], message-based inspection model [7], and multi-agent based systems semantics [25], can be integrated with AVISPA's communication channel in the future [30]. AVISPA's selection of this model is supported by the fact that this model can emulate the actions of an arbitrary adversary [12], and it is also very challenging because it gives advantage to the intruder as opposed to other models [12]. 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 the 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 that 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 also so called composition roles that 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, that 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 [32].

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.

```
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
```

A goal section of the protocol definition can be as follows:

```
goal
  secrecy_of k1
  authentication_on bob_alice_nb
end goal
```

The first statement describes the goal fact above and the second statement describes another goal fact that was not included in the example. We do not show the syntax in the transition section for this security goal. However, this statement is used to indicate the authentication. Notation bob_alice_nb is simply used to name the corresponding goal facts in transition sections of basic roles.

## 3.4 Back-ends

As Figure 11 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 [32].

**The On-the-fly Model-Checker (OFMC)** [5] performs protocol falsification 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.

**The Constraint-Logic-based Attack Searcher (CL-AtSe)** [5] applies constraint solving as in [14], 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.

**The SAT-based Model-Checker (SATMC)** [5] 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.

**The TA4SP (Tree Automata based on Automatic Approximations for the Analysis of Security Protocols) back-end** [10] 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).

To validate the security properties of the AMT specification using AVISPA, we discuss three different scenarios in this section that are going to be implemented in HLPSL using the input file explained in the following section. In our scenario design, we use an incremental methodology. We start with minimal number of sessions, and minimal intruder knowledge. Note that, an intruder can impersonate any agent just by putting the variable 'i' instead of the agent and so, he can receive every message sent to the hacked agent. We make things more challenging by adding information to the hacker and adding more parallel sessions. Since the OFMC back-end of AVISPA finds one goal at a time, the number of times we need to run each scenario will be equal to the number of goals we have. With each run, we choose one goal and comment the rest out.

After defining the sessions, we define the environment composed of sessions, intruder knowledge and the security goals. The Intruder knowledge describes the communication and security parameters known to the hacker besides the channels, the hash functions, the agent identities and so on. By combining this knowledge with the message intercepted or received during the protocol validation, the intruder can build new messages and try to violate the security defined in the section "Goals" of the HLPSL environment code. The goals often wished are the data confidentiality (secrecy_of), authentication of agent and detection of replay attack (authentication_on, request, witness).

In our protocol, the goals we need to verify are as follows:

- The authentication of one agent to the other;
- The confidentiality or secrecy of some local parameters.

- The absence of one or more replay attacks.

# Chapter 4

# Implementation & Experiments

## 4.1 AMT Three-way Handshake HLPSL Specification

In this section, we will be examining the HLPSL input file used to validate the AMT specification.

### 4.1.1 Gateway HLSPL

We first define the role Gateway, its parameters, and its transitions. Note that since the Gateway is not supposed to be aware of the cryptographic hashing function used by the Relay to create the message authentication code (MAC), there is no need to define it in the role. The Gateway will simply echo the MAC back to the Relay upon receiving it in the AMT membership message.

```
role gateway(
  G, R    : agent,
  SND,RCV : channel(dy))

played_by G def=
local
    State : nat,
    Ng    : text,                          % Nonce created by Gateway
    MAC   : hash(text.agent.symmetric_key) % Message Authentication Code
                                           % (A crytographic hash created
                                           % by the relay using the IP of
                                           % the Gateway,source port,
                                           % Nonce, and a local secret
                                           % known only to the Relay
init
   State := 0

transition

1. State =  0 /\ RCV(start) =|>
   State':= 2 /\ Ng' := new()
              /\ SND(G.Ng')               % Send AMT Relay Discovery
2. State  =  2 /\ RCV(R.Ng) =|>           % Receive AMT Relay
                                          % Advertisement Message
                                          % Unicast IP of Relay
                                          % is now known to Gateway
   State' := 4 /\ Ng' := new()
              /\ SND(G.Ng')               % Send AMT Request Message
3. State  =  4 /\ RCV(R.Ng.MAC') =|>      % Receive AMT Query Message
                                          % includes Gateway
                                          % Nonce and MAC
   State' := 6 /\ SND(G.MAC'.Ng)          % Send AMT Membership
```

```
                                          %Update Message
                /\ request(G,R,gateway_relay_mac,MAC') % Security
                                                        % Goal: Gateway
                 % Authenticates
                 % Relay on
                 % MAC

end role
```

### 4.1.2 Relay HLSPL

```
role relay(

  G, R    : agent,
  Hash    : hash_func,
  SND,RCV : channel(dy))

played_by R def=

  local
    State : nat,
    Ng    : text,    % Nonce created by Gateway
    MAC   : message, % Message Authentication Code
                     % (A crytographic hash created by
                     % the relay using the IP of the Gateway,
                     % source port, Nonce, and a
                     % local secret known only to the Relay
    LS    : text     % A local secret known only to the Relay


  init
    State := 1


transition


1. State   = 1 /\ RCV(G.Ng') =|>          % Receive AMT Relay Discovery
                                          % Message
   State' := 3 /\ SND(R.Ng')              % Send AMT Relay Advertisement
                                          % Message
2. State   = 3 /\ RCV(G.Ng') =|>          % Receive AMT Request Message
   State' := 5 /\ LS' := new()            % Create Local Secret
               /\ MAC' := Hash(Ng'.G.LS') % Create MAC
               /\ SND(R.Ng'.MAC')         % Send AMT Query Messasge
               /\ secret(LS',s_ls,{R})    % Security Goal: Local secret
                                          % should only be
                                          % known to Relay
               /\ witness(R,G,gateway_relay_mac,MAC') % Security
                                                      % Goal: Gateway
                                                      % Authenticates
                                                      % Relay on
                                                      % MAC
               /\ secret(MAC',s_mac,{R,G}) % Security Goal: MAC
                                           % should only be
                                           % known to Relay & Gateway
3. State  =  5 /\ RCV(G.MAC.Ng) =|>        % Receive AMT Membership Update
   State' := 7
```

```
end role
```

### 4.1.3 Session Role HLPSL

To aid in executing several roles in parallel, we define the role session, which is basically a composition of both roles Gateway and Relay shown earlier.

```
role session(

  G,R   : agent,
  Hash  : hash_func)

def=

  local SND, RCV : channel (dy)

  composition

     gateway (G,R,SND,RCV)
  /\ relay   (G,R,Hash,SND,RCV)

end role
```

### 4.1.4 Environment Role HLPSL

Our scenarios are going to be similar in all parts of the HLPSL specification except the environment role. In this role, we can modify the number of parallel sessions and intruder knowledge.

```
role environment()
```

```
def=

const
    g,r                   : agent,
    relay_gateway_mac,
    relay_gateway_ls      : protocol_id,
    h                     : hash_func

intruder_knowledge = {g,r,h}

composition
    session(g,r,h)
 /\ session(g,i,h)
 /\ session(i,r,h)

end role
```

## 4.2 Problem Classification

Since the desired security properties are usually not mentioned or stated in the original documents explaining the protocol, the AVISPA team alongside some partners came up with a deliverable document that lists a set of selected problems that were the result of rigorous evaluation of approximately 80 protocols [30]. Having come out with 384 problems divided into 33 groups, such a range of problems can be used as a general framework for classifying problems found with the AMT protocol. Usually, all HLSPL protocol specifications submitted to the AVISPA project have to classify the security goals they are validating by referring to the groups mentioned in the deliverable. Before stating the list of groups of the AMT protocol, it would be useful and fair to quote the text mentioned under the security considerations section according to the recent AMT draft [31]:

> The anycast technique introduces a risk that a rogue router or a rogue AS could introduce a bogus route to the AMT Relay Anycast prefix, and thus divert the traffic. Network managers have to guarantee the integrity of their routing to the AMT Relay Anycast prefix in much the same way that they guarantee the integrity of all other routes.
>
> Within the native MBGP infrastructure, there is a risk that a rogue router or a rogue AS could inject a false route to the AMT Subnet Anycast Prefix, and thus divert joins and cause RPF failures of multicast traffic. As the AMT Subnet Anycast Prefix will be advertised by multiple entities, guaranteeing the integrity of this shared MBGP prefix is much more challenging than verifying the correctness of a regular unicast advertisement. To mitigate this threat, routing operators should configure the BGP sessions to filter out any more specific advertisements for the AMT Subnet Anycast Prefix.
>
> Gateways and relays will accept and decapsulate multicast traffic from any source from which regular unicast traffic is accepted. If this is for any reason felt to be a security risk, then additional source address based packet filtering MUST be applied:
>
> - To prevent a rogue sender (that can't do traditional spoofing because of e.g. access lists deployed by its ISP) from making use of AMT to send packets to an SSM tree, a relay that receives an encapsulated multicast packet MUST discard the multicast packet if the IP source address in the outer header does not match the source address that would be extracted using the rules of Section 7.2.
> - A gateway MUST discard encapsulated multicast packets if the

> source address in the outer header is not the address to which the encapsulated join message was sent. An AMT Gateway that receives an encapsulated IGMPv3/MLDv2 (S,G)-Join MUST discard the message if the IP destination address in the outer header does not match the source address that would be extracted using the rules of Section 7.2.

Although there is a mention of a consideration that a rogue router could introduce a bogus route to the AMT Relay Anycast prefix, there is still no guarantee for the Gateway whether it is addressing an authentic Relay or not. In addition to this, an intruder can still spoof the IP address of the Relay and impersonate it after the Gateway learns about the IP address of the Relay it is going to start the three-way handshake with. The list of suggested problems are mentioned below:

**(G1) Entity authentication (Peer Entity Authentication):** One party needs to be certain, through presentation of evidence and/or credentials, of the identity of a second party involved in a protocol run, and that the second party has actually participated during execution of the current run of the protocol.

**(G3) Replay Protection:** One party needs to be certain that an authenticated message was generated during this session, during a known recent time window, or had never been accepted before.

**(G12) Confidentiality (Secrecy):** One party needs to be certain that a particular data item is not made available to unauthorized individuals such as intruders.

### 4.2.1 Goals

Security goals are specified in HLPSL by augmenting the transitions of basic roles with so-called goal facts and by then assigning them a meaning by describing, in the

HLPSL goal section, what conditions, that is, what combination of such facts, indicate an attack. A simple example of this is secrecy, where the goal facts assert which values should be secret between whom, and the goal declaration in the goal section describes that any time the intruder learns a secret value, and it is not explicitly a secret between him and someone else, then it should be considered an attack. Now although the recent AMT draft[31] does not explicitly mention any security goals in terms of authentication or secrecy between Gateway and Relay, it is quite safe to assume that we would like the AMT Gateway to identify that it is communicating with a real AMT Relay, and vice versa, and that is after successfully conducting a three-way handshake between both. According to our HLPSL validation results, and given the details of the three-way handshake mentioned in the AMT draft [31], we have correctly identified one authentication problem where an intruder can present himself as a relay instead of the real one. Details of this attack will be explained later in this chapter. The security goals chosen for the three-way handshake to be validated against are:

- **Mutual authentication between Relay and Gateway through the use of the MAC** In this security goal, we validate whether an intruder could be able to impersonate a Relay or a Gateway in an AMT session. This maps directly to our problem classification G1, Entity authentication.
- **Secrecy of the MAC** In this security goal, we validate whether an intruder could introduce a threat to an existing valid session between an AMT Relay and Gateway after learning the value of the MAC. In other words, we try to look for a replay attack. This maps directly to our problem classifications G3 and G12, Replay Protection and Confidentiality.
- **Secrecy of the Local Secrect** In this security goal, we validate whether an intruder could introduce a threat to an existing valid session between an AMT Relay and Gateway after learning the value of the local secret parameter that

is part of the hashed value of the MAC. This maps directly to our problem classification G12, Confidentiality.

## 4.3 Scenarios

As already mentioned, we build our solution according to the following scenarios:

### 4.3.1 Scenario 1

In this scenario, we are going to compose one session only between Gateway and Relay, and limit the intruder knowledge only to the identities of Gateway and Relay. Figure 12 describes the typical protocol flow for this scenario. The following changes are also made to the HLPSL input file:

```
session(g,r,h)
intruder_knowledge = {g,r}
```

The purpose of this scenario is to model an attack, if found, without explicitly stating that the intruder is impersonating either a Gateway or a Relay, in which case he can easily replay attacks among different sessions.

### 4.3.2 Scenario 2

Similar to Scenario 1, except that we increase the intruder's knowledge with the cryptographic hashing function. As we show in the results later in this chapter, the additions in this scenario did not affect the results from the first scenario. Figure 12 also describes the typical protocol flow for this scenario. The following changes are also made to the HLPSL input file:

```
session(g,r,h)
intruder_knowledge = {g,r,h}
```

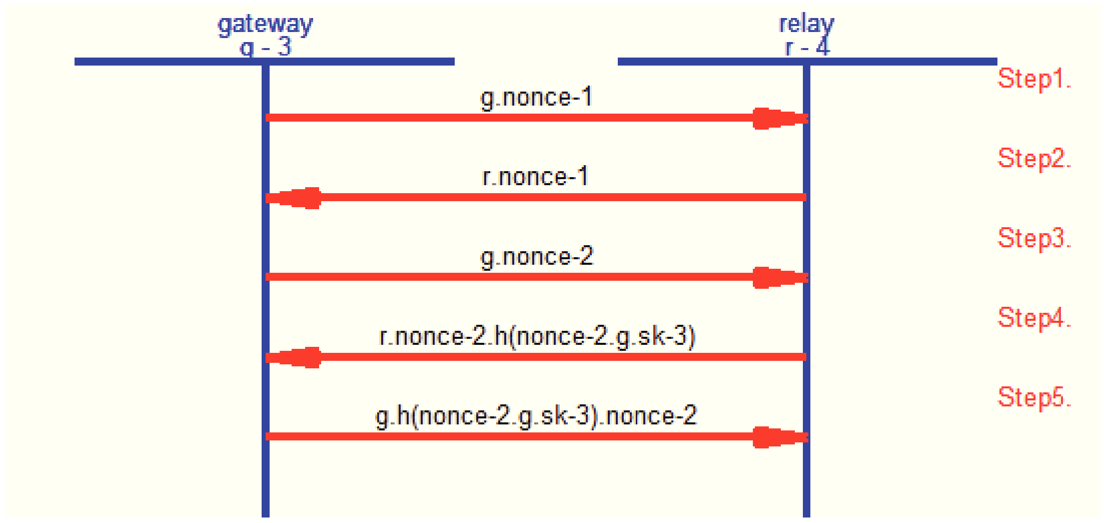

Figure 12: Scenario 1 - AMT Three-way Handshake

### 4.3.3 Scenario 3

In this scenario, we are going to compose four parallel sessions, two of which are composed of Relay and Gateway, the other two include the intruder impersonating a Gateway, and then a relay. Intruder knowledge will not be limited. Figure 13 shows the typical flow of the protocol in this scenario. The following is also reflected in the HLPSL input file:

```
   session(g,r,h)
/\ session(g,r,h)
/\ session(g,i,h)
/\ session(i,r,h)

intruder_knowledge = {g,r,h}
```

## 4.4 Validation Results

For running AVISPA validation, we used a personal computer running Windows 7, with 4 GB of RAM and an Intel DualCore 2.53 GHz processor. After completing the

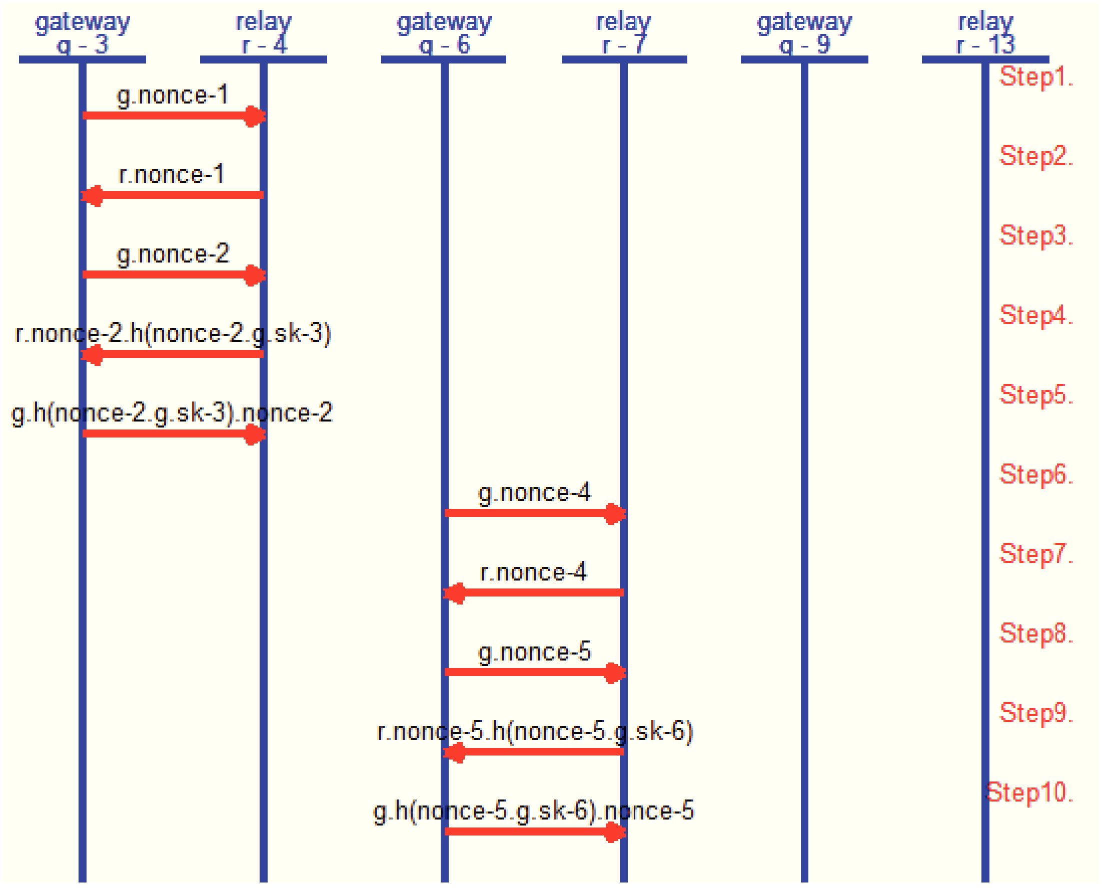

Figure 13: Scenario 3 - AMT Three-way Handshake

specification as described above in three scenarios, we divided the validation process into two steps. The first step was to validate the specification using the OFMC back-end tool of AVISPA, and the second step was to use the ATSE back-end. For both steps we prepared the same input file; the specification listed earlier. OFMC is the back-end tool of AVISPA that has the highest speed in detecting attacks. This is the reason why we started the validation with it. AVISPA's ATSE back-end tool provides a translation for the protocol specification into a set of constraints. In particular, each step of the protocol is modeled by using constraints on the adversary's knowledge. Then, these constraints are used to find attacks against the specified security objectives. Details of validation results for each goal are explained in the following subsections.

### 4.4.1 Authentication on MAC

According to [31], the produced MAC is used only for routability purposes by the respondent, and the originator does not need to know anything about it in terms of content or hashing algorithm. As a result, if any party impersonates itself as an authentic relay, the gateway has no way to figure out otherwise. All the scenarios produced the same results, an attack was found where the intruder pretended to be a relay and consequently fooled the Gateway. An attack trace from the third scenario is shown below. Figure 15 shows a visual protocol flow of scenario 3 where the intruder impersonates a relay. Note than since an intruder can easily impersonate a relay, he will also be able to easily learn unsecured parameters passed with the gateway.

```
i -> (g,3): start
(g,3) -> i: g.Ng(1)
i -> (g,3): r.Ng(1)
(g,3) -> i: g.Ng(2)
i -> (g,3): r.Ng(2).x1003
(g,3) -> i: g.x1003.Ng(2)
```

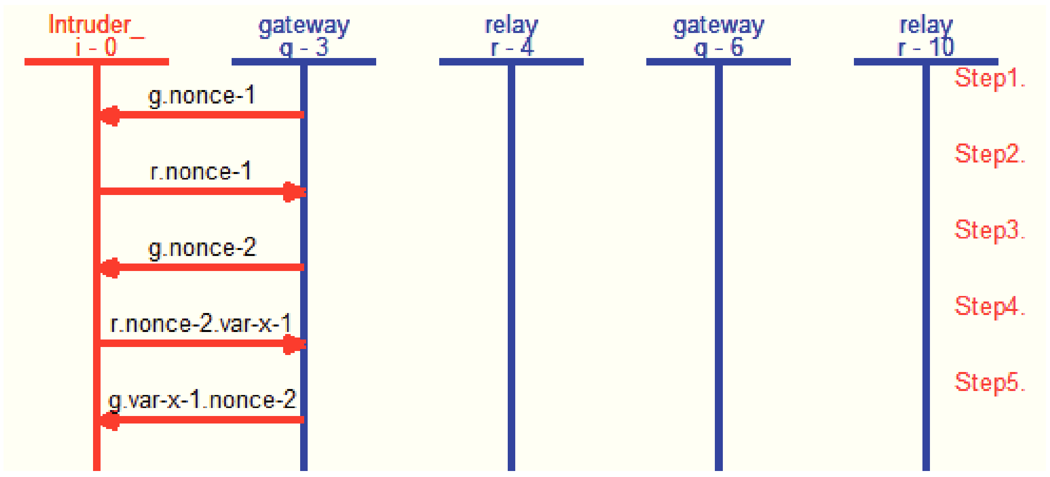

Figure 14: Intruder impersonating a relay

### 4.4.2 Secrecy of MAC

As explained earlier in section 2.2.5, the secrecy of the MAC (message authentication code) that is created by the AMT Relay is modeled by means of the goal predicate secret(T,id,A,B) standing for "the value of term T is a secret shared only between A and B" [30]. This secrecy property is going to be violated only when the intruder becomes aware of the value T that is considered as secret and that he is not allowed to know. (According to [30], if in a certain session the intruder plays the role of a honest agent that is allowed to know the secret value, then the intruder is allowed to know it and no attack is reported for this value.) The label id (of type protocol_id) is used to identify the goal. In our specification, the secret property is modeled using secret (MAC,secrecy_of s_mac,R,G). In all three scenarios, AVISPA found the protocol to be unsafe with regard to this goal, implying that the intruder was able to learn the value of the MAC. This gives rise to a scenario where an intruder may be able to make use of the MAC by sending a Membership update Leave/Done message to the Relay while spoofing the source IP of the Gateway. This can result in the Relay disconnecting the unicast stream to the Gateway.

A run by AVISPA produced the following attack trace, showing how the intruder was able to learn the value of the MAC in the first scenario. Figure 14 supports the attack trace by showing a visual run of the protocol and how the intruder was able to learn the value of the MAC.

```
ATTACK TRACE
i -> (r,3): g.x230
(r,3) -> i: r.x230
i -> (r,3): g.x239
(r,3) -> i: r.x239.h(x239.g.LS(2))
i -> (i,17): h(x239.g.LS(2))
i -> (i,17): h(x239.g.LS(2))
```

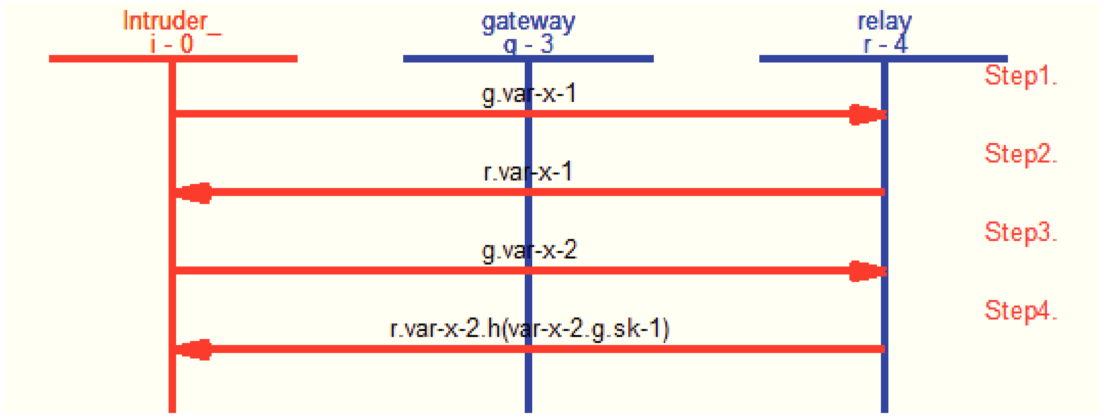

Figure 15: Intruder learning value of MAC

### 4.4.3 Secrecy of Local Secret

According to [31], the local secret never has to be shared with the other side, it is only used to verify return routability of the originator. In the light of this knowledge, the Gateway does not care what the local secret or MAC produced by the Relay is, since it simply echoes it back in the AMT membership update message. As explained in the previous subsection, an intruder cannot figure out the contents of the MAC, and thus will not be able to learn the value of the local secret. To further confirm this, the validatation results from all three scenarios showed the protocol as safe with respect to this goal for a bounded number of sessions.

## 4.5 AMT Demonstration

A demonstration of AMT's functionality was carried out in the beginning to help realize the specification in the real world. It is important to note that we did not validate our security goals through this experiment, but rather recognize the steps of the AMT process visually. Juniper Networks has developed a functioning AMT gateway and relay, which are available as a trial service [27]. They were downloaded and used to carry out the following experiment explained next. In this experiment, all used Cisco routers were initially configured with the PIM-SM multicast protocol. For building routing tables, OSPF routing protocol was employed. The topology used in this experiment is shown in figure 12. The VLC player was used to stream MPEG-4 video content on the streaming server, and also used on the end-user side to receive it. AMT Relay functionality is installed and initially disabled on the Streamer, while AMT Gateway functionality is installed and initially disabled on Sink2.

### 4.5.1 AMT Experiment

In the first scenario of this experiment, and while multicast functionality was enabled on all routers and no AMT functionality was enabled, the outcome was successfully streaming content from the streamer to the receivers (Sink1, Sink2). In the second scenario, we disabled the multicast functionality on R3, and hence, Sink2 was no longer able to receive multicast traffic. Keeping the multicast functionality disabled on R3, in the third scenario we enable the AMT functionality on the Streamer (AMT Relay), and the AMT functionality on Sink2 (AMT Gateway). After some duration taken by the Gateway to discover and set up the tunnel with the Relay, Sink2 was able to successfully receive multicast content from the Streamer.

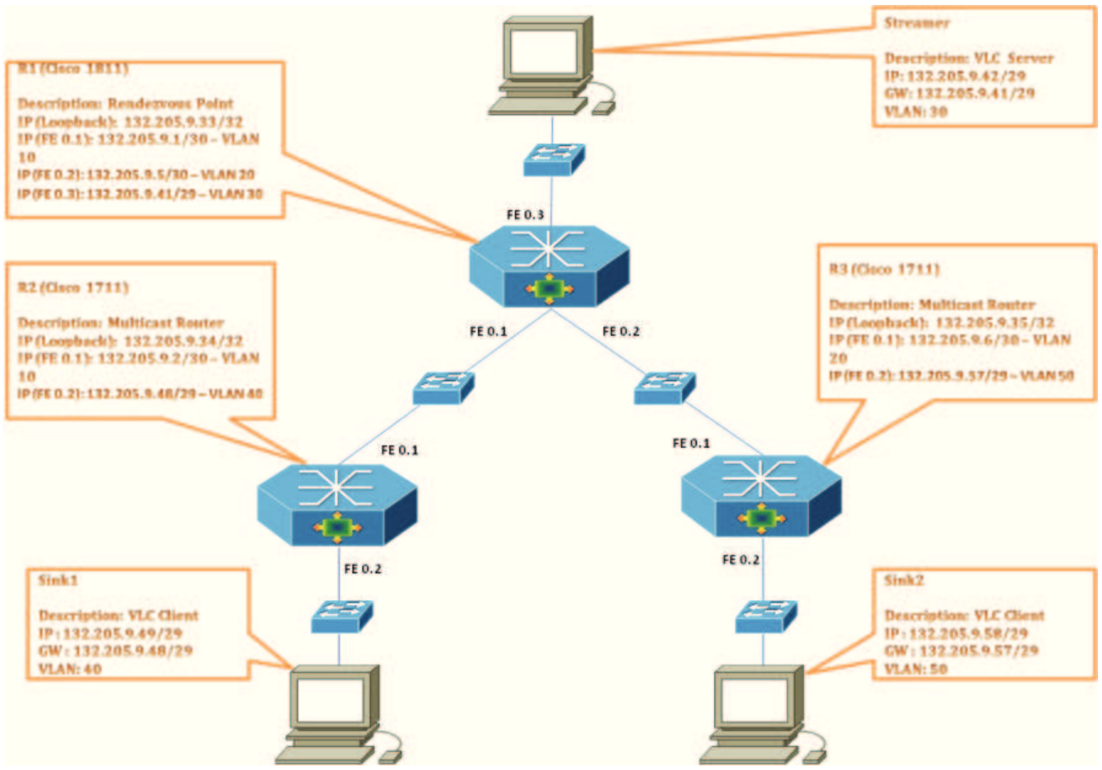
Streamer
Description: VLC Server
IP: 132.205.9.42/29
GW: 132.205.9.41/29
VLAN: 30
R1 (Cisco 1811)
Description: Rendezvous Point
IP (Loopback): 132.205.9.33/32
IP (FE 0.1): 132.205.9.1/30 – VLAN 10
IP (FE 0.2): 132.205.9.5/30 – VLAN 20
IP (FE 0.3): 132.205.9.41/29 – VLAN 30
FE 0.3
R2 (Cisco 1711)
Description: Multicast Router
IP (Loopback): 132.205.9.34/32
IP (FE 0.1): 132.205.9.2/30 – VLAN 10
IP (FE 0.2): 132.205.9.48/29 – VLAN 40
FE 0.1
FE 0.2
R3 (Cisco 1711)
Description: Multicast Router
IP (Loopback): 132.205.9.35/32
IP (FE 0.1): 132.205.9.6/30 – VLAN 20
IP (FE 0.2): 132.205.9.57/29 – VLAN 50
FE 0.1
FE 0.1
FE 0.2
FE 0.2
Sink1
Description: VLC Client
IP : 132.205.9.49/29
GW : 132.205.9.48/29
VLAN: 40
Sink2
Description: VLC Client
IP : 132.205.9.58/29
GW : 132.205.9.57/29
VLAN: 50

Figure 16: AMT Experiment 1

# Chapter 5

# Conclusions and Future Work

In our thesis we presented how the AVISPA formal validation tool can be used to validate the proposed security properties of the AMT protocol and locate problems. From our knowledge, and as stated earlier in this thesis, there were no previous attempts to validate the three-way handshake or any other part of the AMT protocol. So we consider that our work is an important step forward in using formal validation for everyone benefiting from the AMT protocol. The major advantage of our approach is the use of AVISPA toolkit and its HLPSL specification language. The reason is related to the models that this tool uses for the network and the intruder. The way these models are implemented assures that a validation process is performed for all the possible topologies that can exist, being given the number and the type of the roles specified. We consider this a major improvement of the formal validation of AMT.

One can observe that ATSE back-end is faster than OFMC. Nevertheless, both of the two tools should be used together, because they apply different algorithms. So if both report that the protocol is safe, the probability that the actual system accomplishes the security objectives is higher. We restricted the validation to a maximum of four sessions, there was no need to go higher when our goals were validated successfully.

The results we obtained for the three scenarios proved to be the same after running them under AVISPA. The results indicated that regardless of the differences we introduced to the number of sessions or intruder's knowledge, the attacks remained the same.

## 5.1 Future Work

One scenario that needs to be addressed well in the draft is the one where an AMT Gateway could be operating behind a Network Address Translation (NAT) server. A gateway operating behind a NAT can lead it to appear to the Relay as having different source ports with every connection. One possible outcome of this scenario is as follows:

- After Gateway learns unicast address of Relay, it begins three-way handshake and sends an AMT request message and uses a source port SP1;
- Relay replies with AMT Query message containing MAC, Gateway completes the three-way handshake by replying with a Membership Update message containing the MAC but uses a different source port SP2;
- When Relay attempts to verify the MAC based on the source IP and port of Gateway, it will get an invalid comparison with original MAC and discard the packet.

Examining the scenario explained above, we can conclude that the AMT Gateways may need to use a designated local port, or range of ports, for initiating connections with the Relay. The gateway must guarantee that it uses the same local port across its communication with the Relay, otherwise it will be required to send a teardown message each time the source port changes to reduce the earlier state created with the Relay.

With respect to the security goals discussed in subsections 4.4.1 and 4.4.2, we suggest encrypting the MAC as a step towards fulfilling the goals. The secret key could be available either dynamically through an intermediate step introduced between the discovery of the Relay and the 3-way handshake process where the Relay and the Gateway agree on the key, or having it communicated to them by a trusted third party in an earlier stage. In all cases, we believe that encrypting the MAC is important in terms of narrowing down opportunities for the intruder to grasp.

# Bibliography

[1] Internet systems consortium. `http://www.isc.org/`.

[2] Wide mouth frog protocol. `http://en.wikipedia.org/wiki/Wide_Mouth_Frog_protocol`.

[3] Alessandro Armando, David Basin, Yohan Boichut, Yannick Chevalier, Luca Compagna, Jorge Cuellar, Paul Hankes Drielsma, Pierre-Cyrille Heám, Jacopo Mantovani, Sebastian Mödersheim, David von Oheimb, Michaël Rusinowitch, Judson Santiago, Mathieu Turuani, Luca Viganò, and Laurent Vigneron. The AVISPA Tool for the Automated Validation of Internet Security Protocols and Applications. In Kousha Etessami and Sriram K. Rajamani, editors, *Proceedings of the 17th International Conference on Computer Aided Verification (CAV'05)*, volume 3576 of *LNCS*. Springer, 2005. Available at `http://www.avispa-project.org/publications.html`.

[4] Alessandro Armando, David A. Basin, Mehdi Bouallagui, Yannick Chevalier, Luca Compagna, Sebastian Mödersheim, Michaël Rusinowitch, Mathieu Turuani, Luca Viganò, and Laurent Vigneron. The aviss security protocol analysis tool. In *Proceedings of the 14th International Conference on Computer Aided Verification*, CAV '02, pages 349–353, London, UK, UK, 2002. Springer-Verlag.

[5] Alessandro Armando and Luca Compagna. Sat-based model-checking for security protocols analysis. *Int. J. Inf. Secur.*, 7:3–32, January 2008.

[6] Joe Salowey Hao Zhou Glen Zorn S. Josefsson Ashwin Palekar, Dan Simon. Protected EAP Protocol (PEAP) Version 2. Internet-Draft draft-josefsson-pppext-eap-tls-eap-10, Internet Engineering Task Force, October 2004. Work in progress.

[7] Stylianos Basagiannis, Panagiotis Katsaros, and Andrew Pombortsis. An intruder model with message inspection for model checking security protocols. *Computers Security*, 29(1):16–34, 2010.

[8] B. Blanchet and A. Chaudhuri. Automated formal analysis of a protocol for secure file sharing on untrusted storage. In *Security and Privacy, 2008. SP 2008. IEEE Symposium on*, pages 417 –431, May 2008.

[9] Bruno Blanchet. An Efficient Cryptographic Protocol Verifier Based on Prolog Rules. In *14th IEEE Computer Security Foundations Workshop (CSFW-14)*, pages 82–96, Cape Breton, Nova Scotia, Canada, June 2001. IEEE Computer Society.

[10] Yohan Boichut, Pierre-Cyrille Héam, Olga Kouchnarenko, 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, Barcelona, Spain, April 2004. The final version will be published in EN in Theoretical Computer Science, Elsevier.

[11] Jan Cederquist and Mohammad Torabi Dashti. An intruder model for verifying liveness in security protocols. In *Proceedings of the fourth ACM workshop on Formal methods in security*, FMSE '06, pages 23–32, New York, NY, USA, 2006. ACM.

[12] Iliano Cervesato. The dolev-yao intruder is the most powerful attacker. In *Proceedings of the Sixteenth Annual Symposium on Logic in Computer Science — LICS'01*, pages 16–19. IEEE Computer Society Press. Short, 2001.

[13] Y. Chevalier, L. Compagna, J. Cuellar, P. Hankes Drielsma, J. Mantovani, S. Mödersheim, and L. Vigneron. A high level protocol specification language for industrial security-sensitive protocols. In *Austrian Computer Society*, pages 193–205, 2004.

[14] Yannick Chevalier and Laurent Vigneron. Automated Unbounded Verification of Security Protocols. Research Report RR-4369, INRIA, 2002.

[15] Cisco. Introduction to mobile ip. 2001. Available at `http://www.cisco.com/en/US/docs/ios/solutions_docs/mobile_ip/mobil_ip.html#wp1030824`.

[16] John Clark and Jeremy Jacob. A survey of authentication protocol literature. Technical report, Carried out as part of a Strategic Research Plan managed by Peter Ryan of the Defence Evaluation Research Agency, 1997.

[17] G. Denker and J. Millen. Capsl integrated protocol environment. In *DARPA Information Survivability Conference and Exposition, 2000. DISCEX '00. Proceedings*, 2000.

[18] D. Dolev and A. Yao. On the security of public key protocols. *Information Theory, IEEE Transactions on*, 29(2):198 – 208, March 1983.

[19] Bill Fenner, Mark Handley, Hugh Holbrook, and Isidor Kouvelas. Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised). RFC 4601, IETF, August 2006.

[20] Leonard A. Giuliano. Next generation tv over the internet: This revolution will be televised. Technical report, 2011.

[21] Lawrence Harte. *Introduction to Data Multicasting, IP Multicast Streaming for Audio and Video Media Distribution*. Althos, 2008.

[22] M. Hussain and D. Seret. A comparative study of security protocols validation tools: Hermes vs. avispa. In *Advanced Communication Technology, 2006. ICACT 2006. The 8th International Conference*, volume 1, pages 303–308, 2006.

[23] Charlie Kaufman. Internet Key Exchange (IKEv2) Protocol. Internet-Draft draft-ietf-ipsec-ikev2-17, Internet Engineering Task Force, 2005. Work in progress.

[24] Thomas Kernen and Steve Simlo. Automatic ip multicast without explicit tunnels. *EBU TECHNICAL REVIEW*, 2010.

[25] A Lomuscio and W Penczek. Ldyis: a framework for model checking security protocols. *Fundamenta Informaticae*, 85(1-4):359–375, 2008.

[26] J.C. Mitchell, M. Mitchell, and U. Stern. Automated analysis of cryptographic protocols using mur phi;. In *Security and Privacy, 1997. Proceedings., 1997 IEEE Symposium on*, pages 141 –151, May 1997.

[27] Juniper Networks. Video distribution in a hybrid multicastunicast world. Mar 2005.

[28] Simon Blake-Wilson Paul Funk. EAP Tunneled TLS Authentication Protocol (EAP-TTLS). Internet-Draft draft-ietf-pppext-eap-ttls-05, Internet Engineering Task Force, 2004. Work in progress.

[29] Dawn Xiaodong Song. Athena: a new efficient automatic checker for security protocol analysis. In *Computer Security Foundations Workshop, 1999. Proceedings of the 12th IEEE*, 1999.

[30] The AVISPA Team. Avispa v1.1 user manual. page 0088, 2006. Available at `http://www.avispa-project.org/package/user-manual.pdf`.

[31] D. Thaler, M. Talwar, A. Aggarwal, L. Vicisano, and T. Pusateri. Automatic IP Multicast Without Explicit Tunnels (AMT). Internet-Draft draft-ietf-mboned-auto-multicast-10, Internet Engineering Task Force, 2010. Work in progress.

[32] Luca Viganò. Automated security protocol analysis with the avispa tool. *Electron. Notes Theor. Comput. Sci.*, 155:61–86, May 2006.