VALIDATION OF THE SECURITY OF PARTICIPANT CONTROL EXCHANGES IN SECURE MULTICAST CONTENT DELIVERY

Mohammad Parham

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

September 2011

Concordia University

School of Graduate Studies

This is to certify that the thesis prepared

By:	**Mohammad Parham**
Entitled:	**Validation of the Security of Participant Control Exchanges in Secure Multicast Content Delivery**

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 commitee:

______________________________ Chair
Dr. Yuhong Yan

______________________________ Examiner
Dr. Jaroslav Opatrny

______________________________ Examiner
Dr. Lingyu Wang

______________________________ Supervisor
Dr. J. William Atwood

Approved ______________________________
Chair of Department or Graduate Program Director

____________ 20 ______ ______________________________

Robin A.L. Drew, Ph.D.,ing., Dean
Faculty of Engineering and Computer Science

Abstract

Validation of the Security of Participant Control Exchanges in Secure Multicast Content Delivery

Mohammad Parham

In Content Delivery Networks (CDN), as the customer base increases, a point is reached where the capacity of the network and the content server become inadequate. In extreme cases (e.g., world class sporting events), it is impossible to adequately serve the clientele, resulting in extreme customer frustration. In these circumstances, *multicast content delivery* is an attractive alternative. However, the issue of maintaining control over the customers is difficult.

In addition to controlling the access to the network itself, in order to control the access of users to the multicast session, an Authentication, Authorization and Accounting Framework was added to the multicast architecture. A successful authentication of the end user is a prerequisite for authorization and accounting. The Extensible Authentication Protocol (EAP) provides an authentication framework to implement authentication properly, for which more than thirty different available EAP methods exist.

While distinguishing the multicast content delivery requirements in terms of functionality and security, we will be able to choose a smaller set of relevant EAP methods accordingly. Given the importance of the role of the ultimate chosen EAP method, we will precisely compare the most likely to be useful methods and eventually pick the Extensible Authentication Protocol - Flexible Authentication via Secure Tunneling (EAP-FAST) framework as the most suitable one.

Based on the work on receiver participant controls, we present a validation of the security of the exchanges that are required to ensure adequate control and revenue recovery.

Acknowledgments

Foremost, I wish to express my sincere gratitude to my supervisor, Prof. John William Atwood, whose encouragement, patience, guidance and immense knowledge from the initial to the final level enabled me to develop an understanding of the subject. His support and understanding helped me to enhance my everyday life in addition to the academic studies. I would also like to express my deepest gratitude to my beloved parents, Abbas Parham and Fereshteh Mohammadzadeh, for their endless love, support and patience. I also wish to thank all my university teachers, friends and the rest of my family members for their encouragement, kindness and support.
Mohammad Parham

Contents

List of Figures

List of Tables

List of Acronyms

AAA:	Authentication, Authorization and Accounting
AAAS:	Authentication, Authorization and Accounting Server
AR:	Access Router
AVP:	Attribute-Value Pair
CP:	Content Provider
CS:	Content Server
EAP:	Extensible Authentication Protocol
EAP-FAST:	Extensible Authentication Protocol-Flexible Authentication via Secure Tunneling
EAP-GTC:	Extensible Authentication Protocol-Generic Token Card
EAP-IKEv2:	Extensible Authentication Protocol-Internet Key Exchange version2
EAP-MD5:	Extensible Authentication Protocol-Message Digest 5
EAP-MSCHAP:	Extensible Authentication Protocol-Microsoft Challenge-Handshake Authentication Protocol
EAP-OTP:	Extensible Authentication Protocol-One Time Password
EAP-PSK:	Extensible Authentication Protocol-Pre-Shared Key
EAP-TLS:	Extensible Authentication Protocol-Transport Layer Security
EAP-TTLS:	Extensible Authentication Protocol-Tunneled Transport Layer Security
EAPS:	Extensible Authentication Protocol Server
EU:	End User
FI:	Financial Institution
GO:	Group Owner
IETF:	Internet Engineering Task Force
MR:	Merchant
NAS:	Network Access Server
NSP:	Network Service Provider

PAC:	Protected Access Credentials
PKI:	Public Key Infrastructure
PPP-CHAP:	Point-to-Point Protocol - Challenge-Handshake Authentication Protocol
RFC:	Request for Comments
TLS:	Transport Layer Security

Chapter 1

Introduction

1.1 Content Delivery

Content Delivery (for example, audio or video on demand) has become significantly easier in recent years. These applications have existed on broadcast media for a long time but now they are adapted to the new medium, the Internet, which is widely distributed and cheaply accessible. Upon a real time request of a client for such an application, content streams get delivered over the Internet through a direct connection from a source to the client. In order to justify the service, the client has to pay for the application he/she is using. This implies that some form of *electronic transaction* (e-transaction) is needed. In order to prevent fraud on payments, some form of *security* is needed too. So far, content delivery applications, e-transactions and required security over the Internet are available and work properly with a fairly large number of clients.

1.2 Scalability

As the number of clients grows, more data need to be produced and delivered to individual clients. Therefore, load in the network (the transfer medium) as well as on the server (the producer of the content) will grow. As this load grows, the number of patrons wanting identical or near-to-identical service is increased.

1.3 Former Solution

In order to serve them all, the easiest solution could be adding more equipment on both the server and transfer media. So that, the single server works as a group of farm computers and is not only responsible for generating content streams for every individual client, but also is responsible for keeping record of all of them for further financial charging issues. On the other hand, the transfer medium needs to manage to deliver the relatively larger content to the clients. Although in such a case we are able to increase the number of clients without a need to change the nature of the available tools and trust relationships, we have to pay a lot for equipment and we will face a low income-to-expenses ratio. More importantly, still we are restricted to capacity limitations on the transfer media.

1.4 Better Solution

Given the fact that we are considering that clients are asking for the same application, it becomes feasible to look at the case as simultaneous distribution of a similar content to many clients. In the extreme, we can provide a well-known soccer example. When millions of fans want to watch a soccer match in real time over the Internet, they particularly ask for the same thing simultaneously. However, still they face difficulties receiving a smooth video and very often they receive discrete images with pixel distortion.

The aforementioned perspective of view (simultaneous distribution of a similar content to many clients) is useful and results in a new architecture, which has two main benefits: reducing the network load and reducing the server load. However, the drawback is that previously-established trust relationships are going to be gone and consequently so is the cash.

In order to be revenue-productive again, we have to find ways to introduce some new form of trust relationships and propose them in new packages. In comparison with the first solution (adding more equipment), although a whole new set of trust relationships has to be established, we can pay less for equipment, have an unlimited number of clients and consequently a higher income. More importantly, we can save network resources as well as available bandwidth.

1.5 Thesis Structure

The actual new trust relationships and the actors have been previously defined [1] and a framework for moving the required information around [2] has also already been designed. From a top level perspective point of view, our thesis is written based on the knowledge about these previous tasks to demonstrate the functionality and associated security in the architecture. The contributions of our work will be: defining the communications needed to effect these trust relationships; ensuring that paths implementing the trust relationships are valid and secure; what the actors need to move around and how they have to do it in a secure way (since the security is essential for revenue generation to work properly); and eventually validation of the security.

In order to have a better understanding of the architecture, we will discuss previous work in the next two chapters and then in chapter 4 we will explicitly focus on the part of the architecture that we wish to contribute to. In chapter 5 we will propose our solution. The validation of the solution by means of a formal validation tool called AVISPA is in chapter 6. Chapter 7 contains a conclusion over the whole work followed by future work.

Chapter 2

Content Delivery

2.1 Unicast

Delivery of streaming data to individual users has been a challenging problem for many years. Scientists and engineers have tried to propose successful solutions with a proven reliability to show the feasibility of content delivery. One of the well-known solutions is called unicast technology, through which data is transmitted to all the destinations that are connected directly and separately to a single server. The unicast architecture has worked efficiently in various media. It has been used in the Internet to deliver data streams to individual users. This architecture establishes a one-to-one connection between a source and a user as the only two actors. In the architecture they are called the server and the client. The architecture provides protocols and rules for data transmission between them.

2.2 E-commerce and Trust Relationships

E-commerce models have been designed already to resolve the problem of taking money over the Internet. They have been designed to work well with the components of the unicast architecture too. Over the years, both the unicast architecture and e-commerce models have been developed gradually and worked well together to transfer the streaming data and make money.

Whenever two parties want to exchange data between themselves, they need to establish some sort of trust relationship. Whether the exchanged data is free or not, real time or not, requires credential or not; the trust relationship has to be established between the two parties to first, find each other and then to make sure that on the other side of the communication line there exists the intended party not somebody else.

If one party is a sender and the other party is a receiver, they still need to mutually authenticate each other. The receiver needs to believe that the sender is the actual sender (so that he can expect to receive valid content) and the sender needs to believe that the receiver is the real receiver. That is establishing the trust relationship.

In the specific case, when a client wants to ask for an application over the Internet, the unicast architecture, e-commerce models and established trust relationships among the actors are necessary. In fact, these trust relationships are naturally added (between the two actors) because if two persons want to negotiate with each other, first they must trust each other and then establish a connection. So far content delivery to individual users is done successfully. It is worth to note that also an acceptable level of security has been added to the procedures to prevent fraud and to prevent free access of non-paying users. Although the trust relationships are, as they called, trustful to transfer data between the actors, the path on which data flows needs to be kept secure too. This additional security, unicast architecture and e-commerce models developed over the time and work well together.

2.3 Multicast Distribution Technology

Inherently, a content producer wishes to increase the number of its clients without any restrictions. In a one-to-one environment, using unicast transmission, established trust relationships and e-commerce models for financial purposes, all of which have to be managed, are not sufficient to comply with the producer's wishes. In fact, in one-to-one transmissions, once trust relationships are established, the source machine is assigned some tasks and responsibilities regarding individual users. Typically, the source machine has two kinds of tasks. Not only does it have to produce data for each individual user and put the data on the transfer media, but also it has to manage each user's session. Session management is needed to deliver the right content at the right time to the user and eventually charge him properly. That is why, if the number of users increases beyond a certain limit, very likely the source machine (sender) will become overloaded and unable to respond properly to requests since the processing requirements of managing the unicast transmissions and e-commerce transactions increases proportionately.

As the number of customers increases, while it is possible to divide the load conceptually into the unicast data delivery load (to be done by a central content server) and the trust management and e-commerce relation load (to be assigned to a content provider), processing requirements still grow directly with the number of customers. Although delivery load and session management load are two separate concepts, they are still growing similarly and eventually, as the number of customers increases, one or both of the subsystems will fail.

In real time applications based on separate one-to-one relationships between a single

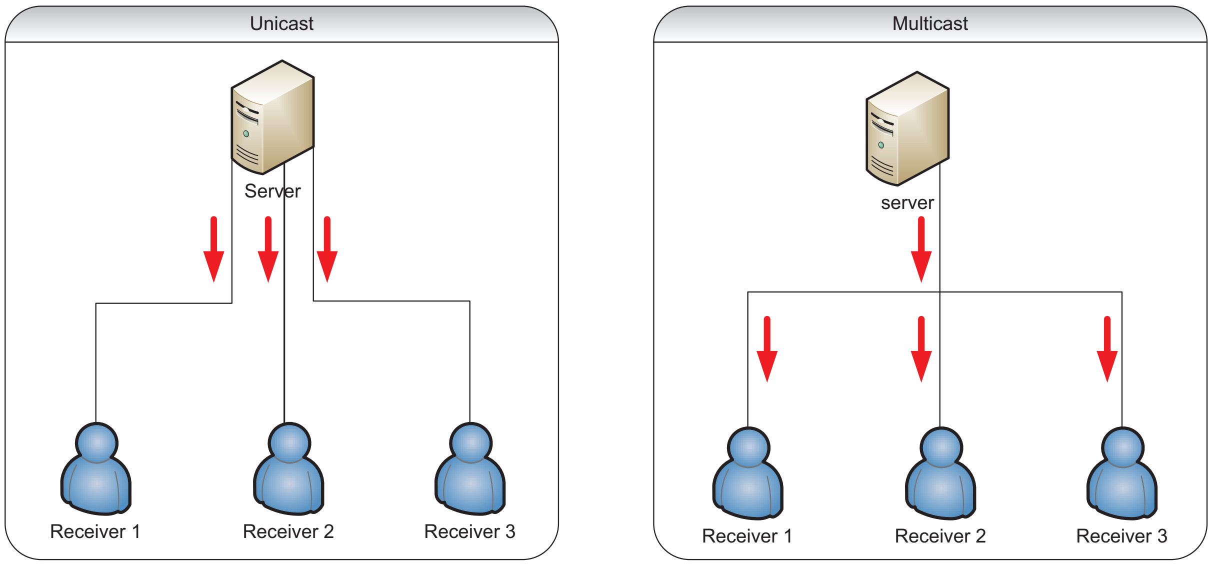


Figure 1: Unicast vs. Multicast

server and all of the paying clients, in fact, when the number of users increases enormously over one specific service being generated by a single server, it is not possible to serve all of them simultaneously due to the two following restrictions:

- Processing limits of the single source machine
- Bandwidth limitations

That is why decreasing the processing needs of the single server and also decreasing the bandwidth needs of the network operators are two important problems to be solved.

A specific form of group data delivery, known as *Multicasting* [3] (see figure 1), was introduced many years ago. In its original mode, it allows multiple senders and multiple receivers. This is a method of simultaneous data delivery to a group of destinations in one transmission attempt from one or more sender(s). It was initially set up in terms of "anybody can send, anybody can receive".

In a video on-demand environment, as the size of the group grows, the likelihood of several receivers wanting identical or near-to-identical service grows up. As specialists began to realize the overload characteristics of unicast data delivery, they found that receivers are likely to form a group since they all receive the same service. This approach was the beginning of the idea to think how to use multicast to solve the scalability problem in unicast (see sections 1.2 and 1.3). In fact, the multicast architecture is very useful in the delivery technology. It responds to the need of a scalable delivery technology for a fairly large number of paying clients (who want identical service) without the need for a prior knowledge of who or how many clients there are.

Unlike the unicast architecture where a sender is responsible for sending data streams to individual users and managing their sessions one-by-one, multicast uses network infrastructures efficiently by only requiring the source to send a packet once, even if it needs to be delivered to a large number of receivers. The nodes in the network take care of replicating the packet only when it is needed to reach multiple receivers. Also some of the network components participate in session management for individual users.

2.3.1 Multicast Scalability

The use of multicast in content delivery reduces the processing needs of the server (for both producing the content and managing individual sessions of each user) as well as the bandwidth needs of the network operators. As the result, the multicast architecture solves the scalability problem we faced in the unicast case. 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.

2.3.2 Multicast Generality

Despite the similarity between pay-per-view structures and multicast content delivery 'to send an identical data stream to a large group of clients in one transmission attempt', in pay-per-view structures, service provider is a private provider and delivery media belongs entirely to that service provider. Also, clients must chose their favorite service among a set of possible services, which is already specified by the company. In contrast, in multicast content delivery, *generality* plays a significant role to give the chance to clients to chose whatever they want in a wide variety of possible data streams and use Internet as the publicly-accessible medium.

2.3.3 Multicast Actors

Although the multicast distribution technology scales for unlimited number of end users, the previously existing trust relationships between the unicast architecture and e-commerce models are gone in the new technology. There is not a single server to produce all data and manage all individual users. Instead, new participants are defined in the multicast

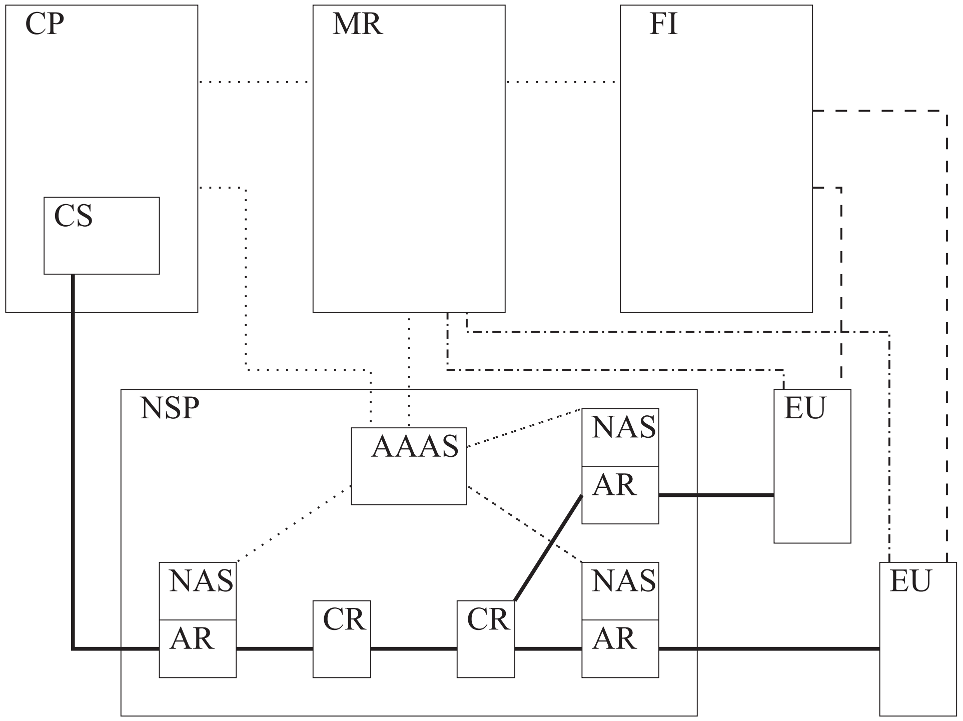


Figure 2: Multicast architecture

architecture, with new responsibilities and roles. In contrast to the unicast architecture in which trust relationships are mainly added across the two actors (the client and the server), a new whole set of more collaborative trust relationships have been defined in the multicast architecture [1] to be distributed among the new participants.

We divide the multicast trust relationships and the actors into *primary* (those on which we will focus in this thesis) and secondary.

Considering trust relationships in multicast area, video on schedule is an example here (as opposed to video on demand in unicast area). There are primary actors (End User, Network Service Provider and Merchant) participating the multicast procedures. These primary actors are cooperating at the same level. See figure 2.

The **End User** (EU) is the actual customer who needs to find a marketer and advertiser on the web to show him his interest in receiving a specific multicast content at the agreed-upon time. The EU is also being asked to prove his ability to pay. Once he provides valid credentials he will be issued a permission to order his favorite multicast service later on. Upon presenting his valid ticket to the closest network-side machine the EU should receive multicast data.

The **Merchant** is actually a web based participant who introduces and advertises the service to the public. It should be easily accessible for customers so that they can visit him anytime and anywhere in the Internet to choose a service. He is responsible for receiving the order from potential EUs, checking their ability to pay through trusted communications with financial institutions and issuing the end users a permission ticket to allow them to join a multicast session in the future.

The **Network Service Provider** 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.

In addition, there are secondary actors (Financial Institution and Content Server/Content Provider), which are out of scope in this thesis document.

- **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 end user's ability to pay. Once the FI approves the ability to pay for a specific end user, the MR becomes assured that the end user 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.

2.3.4 Multicast vs. Kerberos

Kerberos [4, 5] is an authentication protocol which similar to secure multicast authenticates nodes on the basis of tickets to allow them identifying each other in a secure manner for further communications. Although mutual authentication is another similarity and Kerberos messages are protected against spoofing and replay attack, Kerberos is built on symmetric key cryptography and requires a trusted third party while in secure multicast neither the MR plays as a trusted third party between the EU and the NSP, nor the AR plays that role between the EU and the EAPS. The need for a trusted third party in

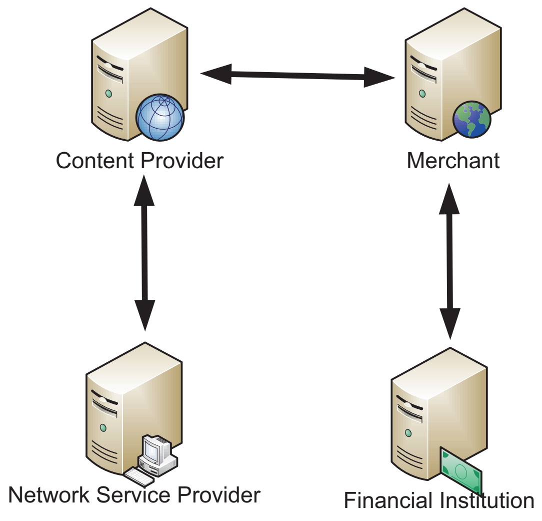


Figure 3: Server-side pre-negotiations

Kerberos protocol increases the overhead and decreases efficiency of the protocol in large scale systems.

Moreover, Kerberos user uses a long-term shared secret as the authentication ticket to receive another ticket (Ticket Granting Ticket) from the authentication server. The user can (re)use the TGT to get additional tickets without resorting to using the shared secret. The TGT is used to receive service from the server. In secure multicast, authentication of the EU is done in the same place as authorization and accounting and they are not separated jobs. Since everybody with acceptable financial proof is potentially able to ask to join a multicast session and receive a ticket, there is no need to separate the authentication and authorization to be done in different places and consequently adding more security steps in between. In Kerberos, for each level of authentication or authorization, there is a need to issue a ticket. In contrast, in secure multicast, once a fresh secret is generated and exchanged, both parties have the option of further secure exchange as much as needed without an extra overhead.

Kerberos has been recently updated [6] but still the main structure is kept the same.

2.3.5 Multicast Trust Relationships

There are some required offline pre-negotiations among network-side participants to be able to define later policies and connection rules based on these pre-negotiations (see figure 3).

As we explained roles and responsibilities of each of the actors (in section 2.3.3), there were indications of trust relationships among them too. Trust relationships are added to the multicast architecture to facilitate “on schedule” content delivery. Although they vary in nature and actors, they work cooperatively and closely. Considering on schedule content delivery, there are six different trust relationships (see figure 4) in this part of the architecture briefly as follows:

1. The CP distributes multicast policies to the MR and the NSP for pre-negotiation purposes.

2. The EU communicates with the MR to submit his request and the MR responds to his request.

3. The MR and the FI have established trust relationships between themselves, through which the FI responds to the MR's request to prove the end user's ability to pay.

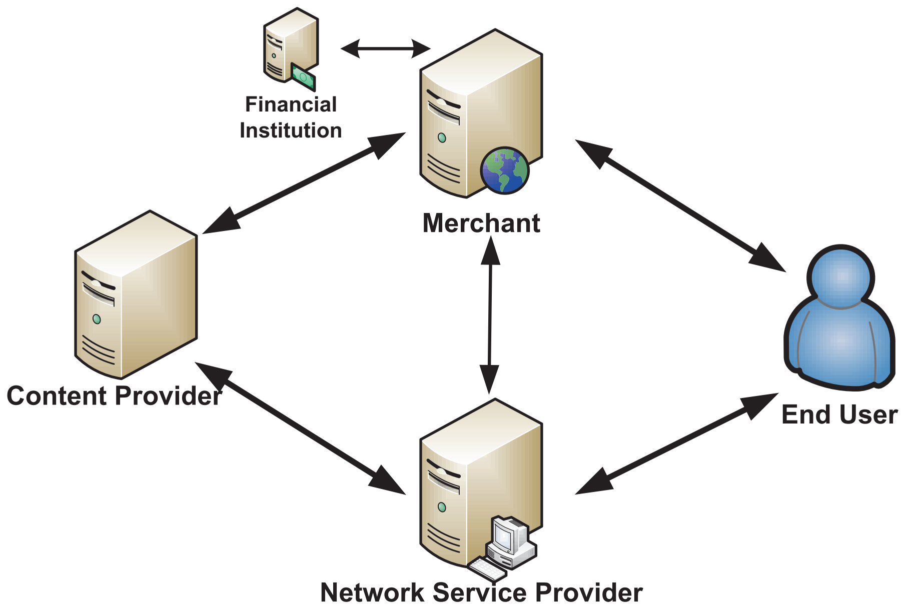


Figure 4: Multicast trust relationships

4. Once the EU receives a valid ticket showing his permission from the MR he presents the ticket (including proper information) to the NSP to receive multicast data.

5. The NSP, on the other side, has communication with the CP/CS to receive multicast data from the source. Clearly, trust relationships are established between them as well.

6. Finally, the NSP and the MR communicate for final charging issues of end users. It is needed since only the MR has direct negotiations with the FI and accounting is done mainly by the NSP components while generally the FI is responsible for actual charging of the EU. So, the NSP communicates with the MR to bridge the accounting steps to the final charging.

As seen so far, unlike the unicast case where only the two actors participate in the whole purchasing and delivery mechanisms, in the multicast case, at least six actors must perform their tasks to finalize purchasing, delivery and charging issues. These actors are located in different places and each one is assigned one or more task(s). These tasks, on the one hand, are distinct tasks since they are done by different actors in different places and times; and on the other hand, they are collaborative tasks since they are associated with individual users. That is why a piece of written information, so-called **token**, is needed to move around between the actors to make their communications feasible. The token is a piece of information basically as a ticket to enter to a multicast session. The token originally is issued by a MR and given to an End User. The end user uses his token to connect to a multicast session. It includes various information in order to mainly validate the MR's legitimacy and the EU's identity.

2.4 IP Multicast Architecture and Policy Framework

Although the classical model of multicast had not been widely deployed due to lack of access control over the sender and receiver, the IP multicast, as the developed technology, potentially can be in use since an accountable, distributed and secure architecture for multicasting has been designed by Atwood [1]. Also, a flexible, scalable and easily adaptable policy framework for multicast group control has been proposed by Islam and Atwood [7], which fully complies with the multicast secure framework.

2.5 Authentication, Authorization and Accounting Functionalities

The multicast distribution technology will be useful only if it makes revenue from purchasers and prevents non-purchasers from taking free advantage of the service. Reviewing the requirements for that reason, the multicast stream must be secured. There are two orthogonal criteria to be satisfied for securing multicast communications and as a result making revenue: managing the access to a multicast session (access control) and protecting multicast data (data control) by applying encryption and deploying secure transmission methods. For secure data transmission, the Multicast Group Security Architecture [8] has been developed by the Multicast Security (msec) Working Group of the IETF. The key to manage the access to a multicast session and generate revenue from purchasers lies on three main issues:

- Validate the end user's identity as a potentially-paying customer to join
- Define his/her rights in using the multicast data
- Check the potential end user's ability to pay and eventually charge him/her after the multicast data delivery

The three functionalities have to be executed in order. An end user who has already proved his ability to pay has to be *identified* first by the session managers once he submits his request to an AR to receive multicast data. Then, the NSP components *authorize* the EU, give him his rights and specify the agreed-upon type and time of the multicast service he has already ordered. Consequently, a successful authentication is a prerequisite for authorization. An authenticated and authorized end user is now eligible to receive multicast data. While the EU is receiving the multicast content, NSP components count the multicast delivered bytes of data and keep records of each EU at the NAS databases for *accounting* and charging purposes.

This implies that there has to be strong relationships to authentication, authorization and accounting of the end users. For this purpose, we can use Authentication, Authorization, and Accounting (AAA) [9] Protocols, which were originally designed to manage the access to the network as a whole.

2.5.1 AAA Framework in Unicast Model

In unicast models, the authentication, authorization and accounting (AAA) protocols [9] are being used successfully to make revenue by controlling the access to network resources. Similar to controlling the access to the network itself, AAA protocols potentially can be used for multicast based applications to authenticate end users, to establish end user's authority in the multicast group and to keep a record of their group activities for charging issues prior to deliver any service to clients. The key point to ensure revenue generation from multicast clients lies on these three consequent steps. This implies that there has to be a strong trust relationship between AAA protocols along with e-commerce interactions.

In unicast, AAA framework has two components: the AAA Server and the AAA Client. The AAA Server works as a central repository and contains AAA information. A Network

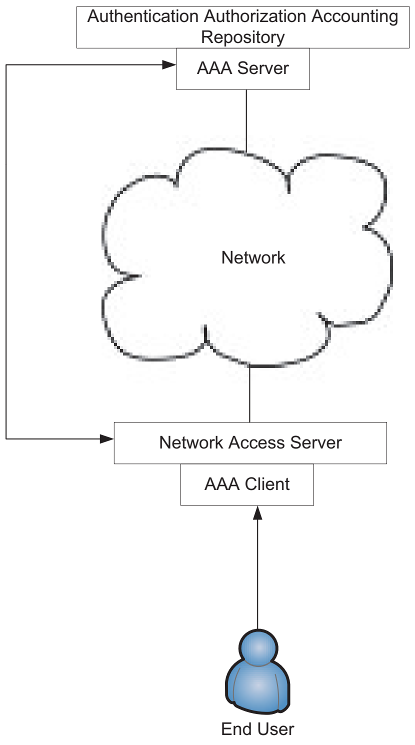


Figure 5: AAA framework in unicast

Access Server (NAS) works as the entry point into the network and contains a AAA client. The NAS collects authentication data of an end user and sends them to the AAA Server. Then, the AAA Server, based on its repository information returns an “accept” or “reject” response to the AAA Client.

The aforementioned AAA framework in unicast could be the simplest similar example for the use of AAA functionality in multicast where a successful authentication is a prerequisite for authorization. For example, in a secure multicast group, only a few members might have the rights to both send and receive multicast data and the rest of the members can join as a receiver only and are not permitted to send any multicast data.

2.5.2 AAA Framework in Multicast Distribution

Similar to managing the access to the network as a whole, which is up until now successfully done by AAA Protocols, Islam and Atwood [2] have proposed a AAA framework to control the access to a (multicast) session. Within this framework, AAA protocols are refined to be used for multicast based applications and individual sessions to authenticate the end users, to establish their authority to participate in the multicast session and to keep a record of their session activities.

Within this framework, once the end user informs the nearest access router of his interest in receiving multicast content, the access router performs the AAA client behaviors on behalf of the Content Provider. A flexible Policy Framework by Islam and Atwood [7] is proposed where the Group Owner (GO) is able to specify multicast policies that will be enforced at the access router.

2.6 Multicast Compatibility

Secure IP multicast is not tied to a particular IP version. Although IPv4 terminology is used in protocol descriptions and definitions, secure multicast content delivery is still applicable to IPv6. The architecture is independent of multicast routing protocols too. Therefore, there are no restrictions to expand the architecture for future improvements in later versions of Internet protocols.

2.7 Secure IP Multicast with Double-sided Access Control

Unlike regular shared media where "everyone can send; everyone can receive", in the secure multicast technology "only legitimate senders can send; only valid customers can receive". This is the result of the double-sided access control policy. It is basically composed of a sender access control [10] to implement the first rule (only legitimate senders can send) and a receiver access control [11] to implement the second one (only valid customers can receive). However, in insecure classical architecture of IP multicast "everyone can send; only valid customers can receive" due to lack of sender access control [10]. That is, the insecure classical architecture of IP multicast, allows any user to send data to a multicast group without a prior join to that group. Consequently, an attacker can take advantage of this shortcoming, weaken the usage of the network and waste the bandwidth by sending many bogus packets to a group. Obviously, data encryption techniques will not solve the problem either.

That is why access control is applied to both the sender and the receiver in the secure multicast architecture. But the sender access control is out of scope in this thesis and whenever we say access control, we mean the receiver (end user) access control. We believe that the receiver access control can guarantee making revenue. It protects the access to the multicast session. Through the receiver access control [11], paying customers are identified to use the session and are authorized to access their rights. Moreover, non-paying users are prohibited from taking advantage of free multicast service.

2.8 NSP in detail

IP multicasting as a content delivery architecture provides its service by means of its components, such as the Network Server Provider(s) over the Internet. The NSPs are working as the heart of the architecture since they provide multicast packet delivery as well as the structure for Authentication, Authorization and Accounting of end users. The NSP includes various participants, who are necessary actors playing significant roles in the architecture. Since the Extensible Authentication Protocol [12] is used for authentication purposes, the EAP Servers (EAPS) interpret the content of the token. Authentication, Authorization and Accounting Server (AAAS), Network Access Servers (NAS), Access Routers (AC) and Core Routers (CR) are the other participants in the NSP.

- **AR**: Access Routers are the closest network routers to end users (as well as to CP/CS). They serve the network link to which a sender (CP/CS) or a receiver (EU) is connected. Access Routers are also in a position to gather necessary information and create a data structure regarding all the individual users who are directly connected to them for further accounting purposes.

- **NAS**: Network Access Servers are associated with each Access Router. They keep records of the end users and maintain data structures of the access router while main task of the access routers is routing and finalizing data delivery.

- **EAPS**: An Extensible Authentication Protocol Server uses previously-provided policies (by the CP or the MR) to issue authentication of an end user who has sent his/her request to the AR to join the multicast session in advance. The EAPS first interprets the content of the token, which is in turn wrapped within the payload of an EAP method. Upon a successful authentication as the first completed A in AAA procedures, the EAP method is going to command the access router to execute the join request and deliver multicast data to the end user based on the end user's authorization. The EAP method also makes sure that the third A happens by commanding the AR to count the bytes (or whatever other accounting statistic it is commanded to accumulate).

- **AAAS**: An Authentication, Authorization and Accounting Server is associated with EAPS and is used to serve the carrier of the EAP packet through AAA protocols from the AR (NAS) to the EAPS.

- **CR**: Core Routers are the interior nodes of the transfer media and performs the task of routing within the network.

Chapter 3

The Extensible Authentication Protocol (EAP)

3.1 Introduction and Components

In the sense that the concept of AAA protocols has a sub-concept of authentication, identity information of the EU has to be transferred from the AAA peer (the EU) to an AAA Server (or its proxy in the NAS). The fact that the EU and the NAS talk to each other for authentication purposes, is done in some authentication protocols. The Extensible Authentication Protocol (EAP) [12] defines an authentication framework, which typically runs directly over data link layers and has three components: the EAP peer, the EAP authenticator and the EAP Server. In multicast content delivery, the three components of the EAP map onto the EU host, the AR (co-located with a NAS) and the "backend authentication server" (co-located with the AAAS) respectively, as shown in Figure 6.

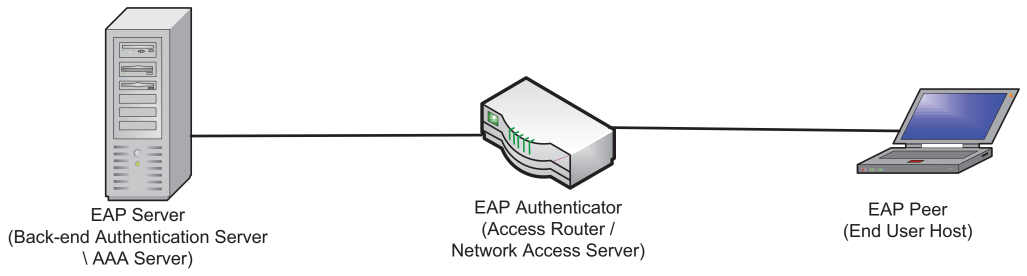


Figure 6: EAP components are mapped to the EU, AR and AAAS

3.2 Authentication in EAP

The EAP was originally designed for use in network access authentication but in the secure IP multicast technology it is used for the *multicast session access authentication*. The EAP protocol can support different multicast authentication mechanisms. In order to simplify credential management and authentication decision making regarding individual users, the AR acts as a pass-through agent for the back-end authentication server and does not collect any credential information about End Users.

3.3 EAP Methods

The Extensible Authentication Protocol (EAP), in addition to the authentication as a functional feature, has security considerations to protect the multicast content delivery against potential attacks and threats. Typically, the base EAP provides negotiation between the EAP peer and the EAP back-end authentication server to chose an *EAP method*, through which they will exchange data. Currently, there exist more than thirty different EAP methods as the IETF standards. In fact, one of the main differences among various available EAP methods is the way that they protect the exchange of information during and after the authentication procedures. Although the structure of the multicast design [1] and participants (see section: 2.3.3) dictates some restrictions on picking the right EAP method that matches the design, the more secure the EAP method that we choose, the more confidence we will have in the security of exchanges. Among more than thirty different possible EAP methods (see section 5.4.2), several of them in some manner satisfy our functionality requirements. However, we have a particular set of security requirements (see section 5.4.4) for which to satisfy we need to precisely analyze the most likely useful EAP methods and pick the right one. In this thesis, we have analyzed relevant EAP methods and chosen EAP-FAST [13] as the most appropriate one. We will show how EAP-FAST best suits our design and satisfies our security requirements in addition to functionality requirements.

There is a variety of different EAP methods. The EU and the NAS agree to chose one. The chosen EAP method then authenticates the EU and dictates authorization and accounting rules according to its payload information, which in fact is the token's content.

3.4 EAP Carriers

The host of the EU places the token in the EAP payload and the EAP method runs directly over the data link layer while the multicast session join request includes network layer information. In order to move the EAP packets from the EU to the NAS, Islam and Atwood [14] have proposed a mechanism, Internet Group Management Protocol with Access Control (IGMP-AC), to link the EAP level request to the network level request. The IGMP-AC performs access control of the end users (only if required for a specific session) and has the capability of carrying EAP packets from the EU to the NAS in addition to the IGMPv3 [15] functionalities. Moreover, EAP packets have to eventually reach the EAPS for interpretation. In fact, it is efficiently possible to move them through AAA protocols (e.g., Diameter [16]) from the NAS to the EAPS. As a result, an Authentication, Authorization and Accounting Server (AAAS) is required to be located with the EAPS. The AAA Server would be the destination of the Diameter packets from the NAS (see figure 8 in section 4.4).

Chapter 4

Problem Statement

4.1 Introduction

In this chapter, we attempt to emphasize on the multicast architecture, starting from an overall point of view and reaching to the specific point that we wish to talk about later on as this thesis contribution. Then, we will mention shortages of previously discussed tools in the multicast architecture (as basic aspects of our thesis problem) in terms of security and functionality. Our goals here are the problem definition and focusing to the point in multicast architecture in which our defined problem resides.

4.2 Multicast Privilege

The multicast distribution technology solves the scalability problem of unicast frameworks to serve unlimited number of end users for one specific service generated by a single server. The multicast architecture reduces the processing needs of the server machine as well as bandwidth requirements of the transfer media.

In the architecture for secure and accountable multicast proposed by Atwood [1], the CS/CP load is reduced significantly since the pair do not have to keep record of individual users any more and do not have to produce data for each end user either. In the multicast architecture, the CS is required to produce only one packet for all the customers and obviously, the packet has to be duplicated by other network components (when needed) and sent to lower stream components all the way down to eventually reach all EUs in real time.

4.3 Multicast Overview

In addition to defining new participants (see section 2.3.3), the multicast architecture takes advantage of network operators and components for routing. That is, the responsibility for routing is distributed over the network. The architecture prevents customers from connecting directly to the source machine and instead, requires them to establish primary trust relationships to:

1. Visit a Merchant (MR) to choose their favorite multicast service and order it.
2. Request from a Network Service Provider (NSP) to connect to the network and join the multicast session.

Upon a successful authentication, which is an exchange of a set of EAP messages, the authenticator decides to allow access to a multicast session to the peer; and the peer, who already has the permission to receive this access, joins the multicast session and receives his favorite multicast data. The EAP method is going to (as a side-effect of the agreeing request) command the AR to execute the join request and deliver multicast data. The EAP method also makes sure that the third A in AAA steps, accounting, happens by commanding the AR to count the amount of delivered data and record the required accounting information in the EU's data structure, which is already created by the NAS at the time of authentication.

Despite the unicast case, "purchasing" and "delivery" in multicast are separate procedures. They take place in different locations and CP is different from MR. However, one-to-one connections and unicast communications are still applicable between each two actors. Here, diffeences between the unicast case and multicast case are:

1. In multicast, purchasing (time) is distributed over time. Since we can add MRs to the architecture as much as needed and clients can order their favorite service anytime, MR interactions are small.
2. In multicast, delivery (time) is very focused in time while streaming data would be a heavy duty in case of an individual connection between each EU and the CP. That is why the architecture is designed to reduce the network load as well as the CP load.

When a multicast packet (destined to all the EUs) is generated by the CS, the packet is being routed within the network to reach the intended EUs. Subsequently, a dynamic Data Distribution Tree will be built to facilitate data delivery. The DDT will grow up while new end users join the multicast session. In the DDT, the root is the single server and the leaves are end users. Network components are responsible for building this tree. Since the

DDT is a minimal-cost tree, practically the multicast architecture reduces the bandwidth requirements of the transfer media too.

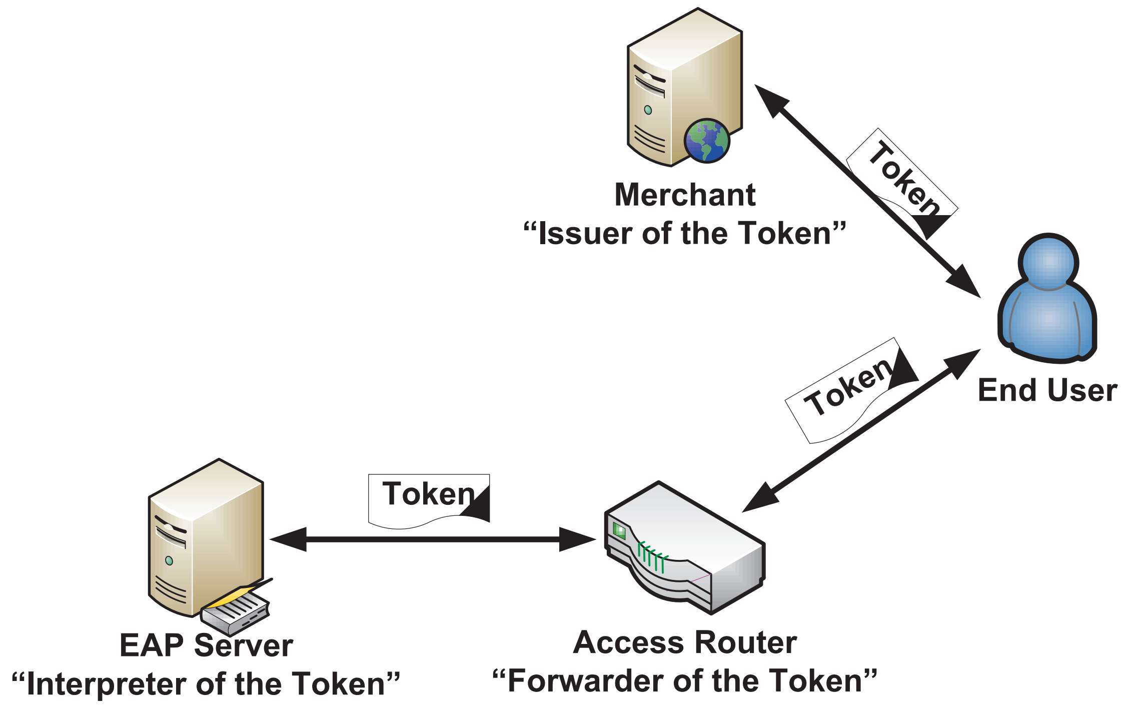


Figure 7: Token in use

4.4 AAA Framework for Multicast

Following the multicast architecture [1], Islam and Atwood have proposed a policy framework for multicast [7] and a framework to add AAA functionalities [2], which demonstrates a whole new set of trust relationships that are defined in the multicast architecture to be distributed over various actors. Two primary trust relationships per EU and many EUs per multicast session exist (see figure 7). Clearly, the result of the EU's first trust relationship is going to be used in his second trust relationship and although they are done in different places, they have the same origin and basically apply to one end user. Therefore, not only these two primary trust relationships are collaborative, but also other aforementioned trust relationships (see section 2.3.5) in multicast architecture are collaborative. That is why in order to facilitate communication of the distributed actors of these trust relationships a piece of information so-called "token" must be handed in by each EU. The token has to be issued by the right actor (a valid Merchant) and has to include proper information about the EU and his favorite multicast service. The token is then given to the EU and the EU presents it to the system once he wants to connect to the network and a multicast session. The token moves around in the system among distributed actors to track each EU's affairs while no direct and real-time communication between network-side parties regarding individual end users exists.

The token contains application layer information and is placed into an EAP packet (see figure 8) over the data link layer. For Authentication, Authorization and Accounting purposes, 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. The EAP packet is carried through IGMP-AC [14] framework between the EU and NAS (as shown in figure 8). The IGMP-AC is the developed version of the IGMPv3 [15] that enables communications of a network level request with a data link layer request. The EAP packet is then carried through AAA protocols (e.g., Diameter [16]) between the NAS and the EAP Server. 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.

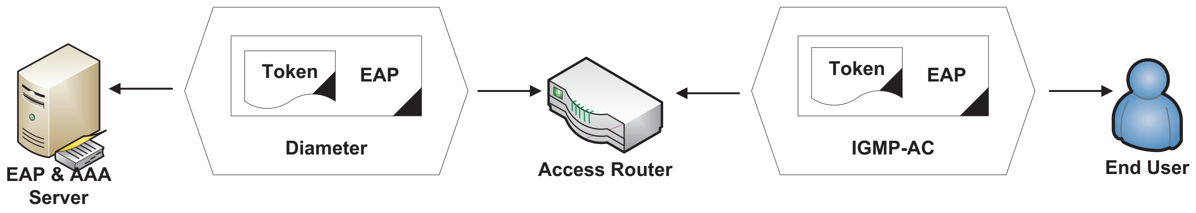


Figure 8: Token and its carriers

In order to control the receiver's access to secure multicast group communication Sultana and Atwood [17] have proposed an architecture to identify multicast end users. The AAA architecture [9] of the IETF is incorporated in the solution. The end user information in the system plays an important role for an Internet Service Provider (ISP) to control the distribution of the multicast traffic as well as to collect real time user accounting information. Islam and Atwood have extended these ideas [14] by using an EAP method to carry the user information and achieve authentication. Moreover, authorization and accounting information is included in the token to have Authentication, Authorization and Accounting (AAA) functionality for each end user.

But so far, no proposal for the content of the token has been made. Given the facts that the AAA functionalities are in tight relationships to the receiver access control protocols and moreover, the EAP method, the token and the carrier protocols play major roles in the authentication, authorization and accounting of end users, we will try to propose a framework to couple these relationships and move toward functionality of the secure IP multicast architecture.

4.5 Shortages and Problem Definition

This thesis is not directly about security. However, it is tied to and based on many security aspects [18, 19], according to which we wish to propose our solutions and ideas to be able to eventually collect money. Therefore, in addition to functionality features that we wish to add to the architecture, we have to add security features too. Moving the token around through unsecured paths will end up in serious frauds. It will prevent the multicast administrators from making revenue. Therefore, the path of the token has to be determined securely.

The architecture can potentially be a target for critical attacks. *Man-in-the-Middle* attack is likely to happen where EAP is tunneled within another protocol that omits mutual authentication of the receiver and the sender.Also, EAP packets may suffer from *spoofing*, *replay attacks* and *modification attacks*. *Dictionary attack* is another common attack among authentication mechanisms. *Connection to an entrusted authenticator* is a threat for the end users if not they are not protected securely. We will explain well-known and likely-to-happen threats and attacks and address security features of our proposed framework to prevent them.

Although in order to show the feasibility of the whole architecture, Islam and Atwood [11] have showed that the EAP-IKEv2 as a potential EAP method works fairly securely, we will discuss about the security and functionality advantages of the EAP-FAST over EAP-IKEv2 and other EAP methods. We believe that the chosen EAP method and a secure tunnel as a path for the token transmission play the most important roles in protecting the on-schedule multicast content delivery from various vulnerabilities. We will validate the claimed security of our framework by means of AVISPA as a network security demonstration tool.

Chapter 5

Participant Control Exchanges in Secure Multicast Content Delivery

5.1 General Scenarios

In secure IP multicast content delivery several different scenarios are likely to happen. They might differ on type of the multicast service, method and policy of charging the end user, given information to and taken information from the end user, interactions of the FI and MR and the steps of finalizing the accounting by the NSP. The EU might be interested in entertainment, news, distance learning or live audio/video content. He might choose to pay by credit card, personal checks, money order, certified checks or pay pal services. The EU might be charged in advance, every hour/day/session or at the end of the multicast delivery. According to these detailed differences, our solution covers a general scenario and is flexible towards future developments.

5.2 Token

5.2.1 Token in multicast content delivery

In secure IP multicast technology, different actors of the multicast content delivery are geographically scattered in a wide area. While no real-time negotiation exists among the actors, they need to communicate with each other to perform different tasks particularly such as pursuing the client's authentication, authorization and accounting. In order to facilitate the communication of different actors, a token [20] is in use and contains AAA information. The token as an application layer piece of information is placed into an EAP packet. The EAP packet in turn is carried using the IGMP-AC [14] framework from the

EU host to the AR and then using the Diameter [16] framework from the AR to the EAPS, which is co-located with the AAAS.

The token is tightly coupled to its immediate carrier, the EAP method, in two main ways.

Functionality

In one hand, the main task of the token is to facilitate the implementation of the authentication, authorization and accounting [9] functionalities. On the other hand, the EAP [12] as an authentication framework has the sub-concept of authentication on its own. Therefore, in addition to carrying the token in the payload, the EAP method basically performs the authentication step and uses the content of the token for that purpose.

Security

The security of the multicast content delivery [18] is composed of network security and data security. While the token itself has its own security features to provide data security, the carrier of the token and the communication path must provide network security in the architecture. For that reason, since the EAP method carries the token, it needs to have security considerations and preferably establish a secure communication path from the EU to the EAPS.

5.2.2 General Content of the Token

Some privacy surveys show consistently that eighty to ninety percent of all people are concerning about privacy and less than ten percent of online customers would be willing to exchange personal information. Therefore, in order to prove financial ability of a multicast client to pay, he is asked (by the MR) to provide only his ordinary credentials. Once the client is guaranteed by the FI, the MR collects identity information of the client and adds authentication clues to the token. Since the token would be placed in an EAP-FAST payload later on, the MR puts EAP-FAST identifiers into the token at this moment. The MR then asks the EU to clarify the features of the multicast service that he is interested in and adds them to the token too. This information must show the following features of the multicast service: Specific content server that produces the content, type of the service, duration of the delivery (start and end time) and number of bytes of the data stream. Authorization of the token will be done according to this part of the token's content. It specifies the multicast service in detail and the rights of the EU to receive it. The MR has to include accounting policy of the EU into the token's content too. The accounting policy is enforced based on the MR-FI interactions and specifies the instructions for the NSP to rule the AR and NAS for charging issues.

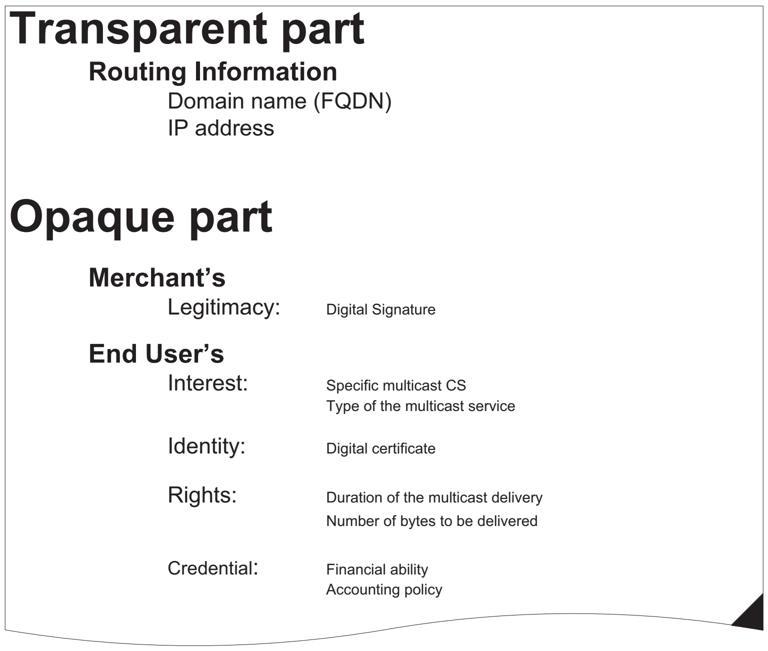


Figure 9: Token in general

Considering various authorization and accounting scenarios, the MR may either inform the NSP to include the new EU along with his authorization and accounting instructions in its list of potential end users or just add pre-negotiated authorization and accounting information to the token. In the second case which is more likely to happen, pre-negotiations (see section 2.3.5) had to be done among the MR and the NSP to facilitate the authorization and accounting procedures in future. These pre-negotiations include enforcing and agreeing on some policies. The MR and the NSP may create and exchange accounting and service tables for that reasons.

Since the token needs to give the network operators enough routing addresses to be routed from the AR towards a proper EAPS, the MR has to include routing information (e.g., an IP address or a domain name) in the token. Moreover, although the AR does not want to know the legitimacy of the issuer of the token (which the EU has presented), the MR adds a digital signature to the token to show the EAPS that the token has been issued by a valid MR.

5.3 Trust Relationships and Authentication

Whenever two distinguished parties want to communicate with each other and in a special case, transfer some important data, they have to first establish a trust relationship between themselves. Each one needs to ensure that s/he actually talks to the intended party, not somebody else. In multicast content delivery, authentication plays a significant role in mutual verification of the identity of the sender and the receiver to establish the required trust relationship. The sender wants to make sure that the receiver is a legitimate user with a proven financial ability to pay and the receiver wants to ensure that he is going to provide his credentials to the trustful sender and receive the ordered-upon multicast content.

Since EUs (as legitimate receivers) are prohibited from visiting directly the Content Provider (CP) (as the trustful sender) and subsequently there is no trust relationship between the EU and the CP, the EAP server works on behalf of the CP and there is a need to establish EU—NSP trust relationships. The EAP framework is added to the design to observe those trust relationships and to implement *mutual* authentication of the EU and the CP. As shown in Figure 6 (section 3.1), the two main components of EAP, the EAP peer and the EAP back-end authentication server, correspond to the EU and the EAP Server in the multicast architecture. The Access Router (AR), who works as the forwarder of EAP messages, represents the authenticator when it works in a pass-through mode.

5.4 EAP and Multicast Delivery Requirements

Generally, a successful authentication in multicast is an exchange of EAP messages, as a result of which the EAP Server on behalf of the CP decides to grant the access to the EU to join the multicast session. In point of fact, in the multicast architecture, the EAP Server's decision typically involves not only authentication, but also authorization and accounting aspects. In order to prevent fraud on the revenue generation, security aspects must be added to the design as well. Choosing the appropriate EAP method is not feasible unless otherwise we determine the *intended features* of the design.

Therefore, we need to clearly understand functional and security requirements of the multicast content delivery to be able to choose the right EAP method that corresponds to the requirements properly and also to be able to justify the advantages of that EAP method over other EAP methods.

5.4.1 Common Features of the Intended EAP Method

In general, EAP is an authentication framework that provides some common functions and features of EAP methods. In the literature, about forty different EAP methods are in wide use. There are many EAP methods defined by RFCs and a number of vendor specific methods exist too. Although they are all defined to perform authentication successfully, not necessarily all of them are useful in the multicast content delivery. According to the structure and requirements of the multicast content delivery technology briefly explained in previous sections, first we determine a set of common features for intended EAP methods and then we will compare the most relevant EAP methods (in section 5.4.2) based on those features. Thereby, we reduce the size of the set of potential EAP methods to be used in our multicast distribution to eventually reach the most suitable one.

Here, we are explaining a set of specifications according to which, we will build the first set of nine EAP methods (in section 5.4.2) out of almost forty different available ones.

- *Not sent in clear text*: Many EAP methods exchange authentication messages in clear text. In order to reduce the risk of replay and spoofing attacks, it is highly recommended to exchange EAP messages encrypted.

- *Dynamic key exchange*: This distributes a user specific encryption key to the peer. Without this feature, all users must share the same static encryption key which increases the chance of a *wide* attack to all the users.

- *Mutual authentication*: This enables the receiver to authenticate the sender in addition to the sender that typically authenticates the receiver.

- *Password-based authentication / Card-based token / Digital certificate*: This shows the nature of credentials to exchange and likely to use for authentication.

- *Secure tunnel*: It is desirable to establish an encrypted channel between the EAP peer and the EAP server to securely exchange authentication messages and encryption keys (if needed). Encrypted tunnel is useful also for securing further exchange of information.

- *Flexibility*: The right EAP method has to have the flexibility to enable support for most password authentication interfaces perfectly consistently. Moreover, the method must be expandable to use multiple existing authentication protocols (if required by the architecture) as well as to enable exchange of authorization and accounting information.

- *Efficiency*: The right EAP method must facilitate the use of a single strong shared secret by the EU in order to allow him to use that shared secret to secure the path of communication similar to use his user name and password to access to the network. This minimizes the device state of the server that the server has to cache and manage. The 'fast reconnect' mode reduces resource requirements for the servers too.

- *Security*: We will discuss a generic attack model (in section 5.4.6) against the implementation of AAA [9] functionalities. Therefore, the EAP candidate must provide protection against common attacks such as Man-in-the-middle, Dictionary attack and Replay attack.

5.4.2 Comparison of Candidate EAP Methods

As shown in table 1, we introduce the following EAP methods as candidates to use in multicast content delivery technology and compare them based on the aforementioned features (in section 5.4.1).

1. *EAP-MD5*: The EAP with MD5 challenge type [12] is similar to the PPP CHAP protocol [21] with MD5 as the specific algorithm. This algorithm is used by the peer to response to the "challenge" message request. All the messages are sent in clear text and the method is a password-based or pre-shared key authentication protocol. It does not support mutual authentication and can be seriously violated by a dictionary attack.

2. *EAP-GTC*: The EAP with Generic Token Card type [22, 12] is defined to work with various token cards supporting challenge/response authentication. The challenge/response messages contain the token card information that is necessary for authentication. The peer might read this information from the token card device and enter them in the communication path in ASCII text. Therefore, this method must not be used to provide support for clear text passwords in the absence of a secure path between the EAP peer and the EAP back-end authentication server.

3. *EAP-OTP*: The One-Time Password protocol [23, 24] contains an OTP challnge for which a response must be sent in reply to the request. The EAP-OTP method is designed to be used with the one-time password system only, and must not be used to provide support for clear text password. Although EAP-OTP method provides replay

Table 1: Comparison of Related EAP Methods vs. Basic Requirements

EAP method	Authentication messages sent in	Key exchange	Authentication	Secure tunnel	AAA security	Credential used (Peer)	Credential used (Server)
EAP-MD5	clear text	N/A	weak peer	x	low	MD5 hash on password	N/A
EAP-GTC	clear text	N/A	weak peer	x	low	token card information	N/A
EAP-OTP	clear text	N/A	strong peer	x	low	password (one time)	N/A
EAP-MSCHAP	hashed-password	N/A	weak mutual	x	low	MS hash on password	returned packet
EAP-PSK	encrypted format	pre-shared	weak mutual	✓	secure authentication	16-byte string	16-byte string
EAP-TLS	encrypted format	dynamic	strong mutual	✓	secure authentication	digital certificate	digital certificate
EAP-TTLS	encrypted format	dynamic	strong mutual	✓	secure authentication	tunneled method	digital certificate
EAP-IKEv2	encrypted format	diffie-hellman	strong mutual	x	secure authentication	see section 5.5.3	see section 5.5.3
EAP-FAST	encrypted format	dynamic	strong mutual	✓	secure AAA	PAC	optional

protection, it provides neither dictionary attack resistance nor mutual authentication.

4. *EAP-MSCHAP*: EAP-Microsoft Challenge Handshake Accept Protocol [12] provides mutual authentication. It uses a Microsoft algorithm to hash passwords for mutual authentication purpose and sends the hashed-password over the communication path. However, it is still not protected against dictionary attack.

5. *EAP-PSK*: The Pre-Shared Key Extensible Authentication Protocol [25] is an EAP method that uses a Pre-Shared Key for mutual authentication and session key derivation. It is documented as a experimental IETF standard for authentication without a need for any public-key cryptography.

6. *EAP-TLS*: The EAP-Transport Layer Security [26, 27] is an IETF standard with high level of security. It uses PKI (Public Key Infrastructure) to secure communications of the peer with the authentication server. EAP-TLS is originally designed as a wireless LAN (Local Area Network) authentication protocol. It is considered one of the secure EAP standards that support mutual authentication. Unlike the previously-mentioned EAP methods, EAP-TLS requires a client-side certificate for authentication. This gives EAP-TLS its authentication strength and illustrates the convenience vs. security trade-off.

7. *EAP-TTLS*: The EAP-Tunneled Transport Layer Security [28] is developed in response to the PKI barrier in EAP-TLS. It is a two-stage protocol that establishes a secure TLS tunnel in stage one and then exchanges authentication information in the form of Attribute Value Pairs (AVPs). It provides mutual authentication, uses digital certificates for server authentication and exchanges user credentials in the secure tunnel for peer authentication.

8. *EAP-IKEv2*: The EAP-IKEv2 [29] is an EAP method based on the Internet Key Exchange protocol version 2 [30]. It provides mutual authentication and session key establishment between the EAP peer and the EAP back-end authentication server. EAP-IKEv2 provides high security properties against potential attacks 5.4.6 and enables the EAP peer and the EAP server to authenticate each other with different

techniques. Further information will be provided in section 5.5.3.

9. *EAP-FAST*: The EAP-Flexible Authentication via Secure Tunneling [13] is proposed as an IETF standard that has two main and one optional phases. In phase 1, it uses a Protected Access Credential (PAC) to establish a TLS tunnel and in phase 2, the peer's credentials are exchanged and verified. Phase 0 is an optional phase in which the PAC can be provisioned manually or dynamically. EAP-FAST provides mutual authentication with an optional use of server certificate and its highly-secured TLS tunnel allows the peer and the server to exchange further credentials and important information.

5.4.3 Functionality Requirements

In the secure IP multicast technology, the agreed-upon EAP method between the EU and the AR requires a set of properties to be able to perform successfully. It has to mutually *authenticate* the EU and the CP successfully and have enough *flexibility* to extend the communications for authorization and accounting all in a *sufficient* way.

Authentication

In order to implement AAA [9, 31] functionalities, the MR issues the token including proper information. Authentication procedures start once the EU wants to connect to the multicast session and presents his token to the NSP components. The EU cannot connect to the multicast session unless his identity is verified in advance. In fact, authentication of the EU is not a separate process to be done after a set of EU-NSP negotiations. The authentication step presents within those negotiations.

Flexibility

Given the fact that within the architecture, a wide range of various multicast services [19] is provided for a wide range of different end users and many different ways exist to authenticate an end user, the architecture must have a flexible authentication method in accordance with the content of the token. Flexibility of the authentication method is mandatory to extend the communications of the EU and NSP components too. The growing complexity in network infrastructures for routing purposes and also the need to gain authorization and accounting features in the design best shows the need for a flexible authentication method.

Efficiency

With a large deployment of the architecture for on schedule multicast services, it is probable that there will be many end users connected to a small number of EAP Servers. Moreover, if the EU either lost the first authorization but not the connection to the multicast session or wishes to receive the multicast content on discrete time slots, the authentication method must facilitate a 'fast reconnect' mode to enable the EU to again reach the multicast session quickly to proceed the next two steps (authorization and accounting). So, it is highly desirable to reduce the EAP Server's per user requirements to gain efficiency in addition to flexibility.

5.4.4 Security Requirements

The secure IP multicast is useful only if it makes revenue from valid end users as well as prevents non-purchasers from taking free advantages of the multicast service. In general, security is essential for that reason. If no precise and reliable security exists, even a small leakage can give the attackers the opportunity to take big advantages and devastate the revenue generation.

In order to understand security requirements, based on the multicast content delivery architecture [1] and the EAP framework explained in chapter 3, a reasonable set of potential attacks (see section 5.4.6) are considered in this thesis, against which the intended EAP method must provide security. On this way, some security assumptions are made as follows:

5.4.5 Security Assumptions

According to the chapters 2 and 3, we are able to determine the type and number of the participants, their roles and the nature of their communications in the secure IP multicast technology. Considering these parameters ensures us that the model will not neglect any potential attack. In fact, there are three main participants in this part of the secure IP multicast technology that we observe to track AAA functionalities: the End User, the Access Router (co-located with a Network Access Server) and the EAP Server (co-located with a AAA Server). Communications of these three participants and the exchanged information between them are subject to various attacks. Since these three participants are not necessarily connected to each other via trusted paths and very likely an intruder is able to reside between them, any kind of attack that needs an attacker to sit between the actors must be considered in the threat model. Specifically, the attacker can listen to the communication from the beginning and gain access to all the public information (e.g., public keys and routing information). Also, the attacker may have the knowledge of *old* pre-shared secret keys.

However, we assume that:

1. All the initiations among the network-side participants have been already done securely and the attacker does not have a chance to access databases of the participants in advance.

2. No attacker has an advance knowledge about the content of the token.

3. Network routers have installed firewalls and proxies to secure themselves and their communications.

4. The AR has secure communications with the NAS and so does the EAP Server with the AAA Server.

5. The EAP method and even its immediate carriers (the IGMP-AC [14] and Diameter [16] frameworks) are run on top of the network layer and subsequently our model considers any attack that can happen on top of the network Layer. So, communications in the physical layer are assumed to be done securely.

5.4.6 A Generic Attack Model

We define the following generic threat model, in which the attacker may carry out a number of attacks to violate either the network security concepts or the data security concepts. We believe our model includes the most likely-to-happen attacks and threats during the implementation of AAA functionalities.

- An attacker may attempt to discover the user identity to violate secure authentication by spoofing the authentication traffic.

- An attacker may try to convince the parties that they are talking directly to each other while in a form of active eavesdropping he makes independent connections with the victims and carries out a man-in-the-middle attack.

- Mounting a man-in-the-middle attack, the attacker may try to convince the peer to connect to an untrusted authenticator.

- An attacker may try to modify or spoof EAP packets and subsequently the content of the token.

- An attacker may replay EAP packets or generate packets with overlapping identifiers to launch a denial of service attack. Obviously, he needs to spoof lower layer indications or Success/Failure packets for that reason.

- An attacker may try to defeat the authentication mechanism by mounting an online and/or offline dictionary attack and determining the decryption keys.

5.5 Most relevant EAP Methods

According to the significance of providing *mutual* authentication, *dynamic* key exchange and *secure tunnel* establishment during the authentication steps via the intended EAP method, we provide a detailed discussion on EAP-TLS, EAP-TTLS, EAP-IKEv2 and EAP-FAST methods since they satisfy the basic and important requirements.

If authentication of network users is not strong, unauthorized users may be able to access network resources as well as access multicast sessions. Strong authentication is a key component in the multicast content delivery technology to provide an acceptable level of authentication. Moreover, in *secure* multicast technology, only legitimate senders are allowed to distribute the multicast content on the transmission path. In comparison with *regular* multicast technology, in secure multicast technology intruders cannot deliver garbage to users and take their credentials. Mutual authentication, in addition to strong authentication is needed to provide enough security during the authentication procedures. Therefore, the intended EAP method must provide strong mutual authentication.

Strong authentication is achievable only if both parties dynamically exchange the keys for encryption. Pre-shared keys are vulnerable to replay and man-in-the-middle attacks and do not provide security for authentication. Consequently, the intended EAP method must provide dynamic key exchange too.

In multicast technology, authorization and accounting are the next steps after a successful authentication. Revenue generation is not guaranteed unless authorization information is transferred securely and so is accounting information. Once a secure tunnel is established between the EAP peer and the EAP back-end authentication server, they can exchange further information in addition to authentication information. As the result, a securely established tunnel is a privilege for the intended EAP method.

In response to the problems about weak authentication, a series of new authentication protocols have been developed to provide strong authentication. EAP-TLS, EAP-TTLS, EAP-IKEv2 and EAP-FAST provide strong authentication, each with its own strengths and weaknesses as shown in table 2.

5.5.1 EAP-TLS

EAP-TLS [27, 26] exchanges digital certificates to authenticate both the peer and the server as part of the TLS establishment and so requiring digital certificates is mandatory for both

Table 2: Detailed Comparison of the Most Related EAP Methods

EAP method	Authentication	Fast reconnect	User identity protection	Remarks	Certificate processing (Peer)	Certificate processing (Server)
EAP-TLS	strong mutual	No	No	Overhead (PKI), Mandates digital certificates	required	required
EAP-TTLS	strong mutual	Yes	Yes (AVP exchange)	Overhead (AVP), Replay and dictionary protection	optional	required
EAP-IKEv2	strong mutual	Yes	Yes (encryption)	Replay and dictionary protection	required	required
EAP-FAST	strong mutual	Yes	Yes(new key exchange)	MitM, replay and dictionary protection, Fresh secrets	required	optional

the peer and the server in EAP-TLS.

This method does not provide fast session reconnect mode. Authentication of peers by means of digital certificates, which mandates a PKI in place. Since usually PKI is not already there, the additional work in issuing and managing certificates is too large.

5.5.2 EAP-TTLS

EAP-TTLS [28] was developed originally in response to the PKI barrier in EAP-TLS. Instead, it exchanges "attribute-value pairs" (AVPs) for peer authentication between the peer and the TTLS server through the TLS tunnel. So, some extra work is required to be done by the peer to prepare AVPs and send them. In EAP-TTLS, server certificate is required (for server authentication) but peer certificate is optional as AVPs are in use. Therefore, the protocol has potential for man-in-the-middle attacks.

5.5.3 EAP-IKEv2

EAP-IKEv2 [29] provides mutual authentication via session key establishment. Typically it supports authentication techniques based on the following types of credentials:

- Public/Private key pairs (Asymmetric key pairs)

- High entropy bit strings (Symmetric keys)
- Low entropy bit strings (Passwords)

This enables the peer and the server to use different authentication information (and thus technique) in each direction. This method protects the authentication procedures and actors against replay and dictionary attacks. It also provides fast session reconnect. However, the session keys have potential for man-in-the-middle attack and therefore, it is not safe to exchange authorization and accounting information (in addition to authentication information) with them.

5.6 Extensible Authentication Protocol - Flexible Authentication via Secure Tunneling (EAP-FAST)

EAP-FAST [13] is an EAP method that enables secure communication between the EAP peer and the EAP server by means of the Transport Layer Security (TLS) to establish a secure and mutually authenticated tunnel. It has many functionality and security advantages over other EAP methods.

5.6.1 EAP-FAST features in general

According to the following features and properties (see also sections 5.6.2, 5.6.3 and 5.7), we believe that EAP-FAST best suits in secure multicast content delivery architecture. In this section, EAP-FAST is discussed from a top level of view and a formal validation is provided in chapter 6.

- The method, provides strong mutual authentication through its established secure TLS tunnel. It is important to note that it uses Protected Access Credentials (PACs) to establish the tunnel. This structure (unlike EAP-TLS, which uses PKI and unlike EAP-TTLS, which uses AVPs) does not add too much work on the components thanks to the simplicity of issuing and using PACs.

- EAP-FAST in phase 1, establishes its secure TLS tunnel and actually generates fresh secret keys during the authentication procedures. It can use the keys for further encryption of important information such as authorization and accounting credentials. Secure transmission of authorization and accounting credentials in multicast architecture is as essential as providing a successful strong authentication since revenue generation is related to all the three steps not only the first one.

- In EAP-FAST, neither peer nor server certificate is used in establishment of the TLS tunnel. Unlike EAP-TLS (which also established a TLS tunnel), the peer credentials are exchanged and protected inside the encrypted tunnel. Therefore, EAP-FAST is more secure and flexible to implement.

- Given the exchange of newly generated fresh secret keys between the peer and the server, the method offers high protection against (passive) dictionary attack, packet forgery (replay attack), authentication forging and man-in-the-middle attacks. It provides 'password expiration and change' to mitigate active dictionary attack too. No other EAP method mitigates all these vulnerabilities simultaneously. This also results in confidentiality and integrity of EAP-FAST.

- Similar to the actual implementation of the multicast content delivery that would be in real world, server authentication is optional in EAP-FAST while peer authentication is mandatory. Since it does not require any server-side certificate, it is very easy to deploy, scale and manage. Fast session reconnect mode is an addition to the flexibility and efficiency of EAP-FAST.

5.6.2 Functionality requirements of the architecture and EAP-FAST corresponding features

Although we wish to lay out the effective security aspects of the EAP-FAST method in the main, considering functionality requirements of the design (see section 5.4.3), here we discuss corresponding properties of EAP-FAST in brief to show that it does not fail to perform its authentication task having enough flexibility in an efficient way.

In the secure IP multicast technology, the agreed-upon EAP method between the EU and the AR requires a set of properties to be able to perform successfully. It must mutually authenticate the EU and the CP and have enough flexibility to extend the communications for authorization and accounting too, all in a sufficient way. We demonstrate below that the EAP-FAST method possesses the required functional features.

Authentication

The EAP-FAST provides authentication of the EU while the EU and the NSP negotiates with each other. We will see (in section 5.5) that the EAP-FAST provides an authentication of the NSP too.

Flexibility

In accordance with the flexibility requirements (see section 5.4.3), the EAP-FAST method has the flexibility to enable support for most password authentication interfaces (e.g., MSCHAP and OTP). Moreover, the method extends to use multiple existing authentication protocols (if required by the architecture) as well as to enable exchange of authorization and accounting information through its secure TLS channel.

Efficiency

As efficiency of the EAP method is highly demanded (see section 5.4.3), the EAP-FAST method facilitates the use of a single strong shared secret by the EU in order to allow him to use that shared secret to secure the TLS tunnel similar to use his username and password to access to the network. This minimizes the device state of the server that the server has to cache and manage. The 'fast reconnect' mode exists in the EAP-FAST method, which reduces resource requirements for the servers too.

5.6.3 Security requirements of the architecture and EAP-FAST corresponding features

Revenue generation in multicast is dependent on the security requirements (see section 5.4.4) being met. In the field of computer networks, the area of security includes two main concepts: the *network* security concepts (e.g., authenticity, availability) and the *information* security concepts (e.g., confidentiality, integrity). If an attacker violates any of these concepts, the whole security would be broken. We will discuss EAP-FAST corresponding security features in section 5.7.

5.7 EAP-FAST Security Claims

In this section, we articulate the security provided by the EAP-FAST method to protect the architecture against the previously mentioned vulnerabilities as a generic attack model in section 5.4.6. We will also justify our proposed security claims.

5.7.1 Confidentiality and Integrity

Although an identity exchange is optional within the EAP negotiations of the peer and the authenticator, it is not desirable to entirely omit it due to the complexity that would be added to the EAP packets for routing purposes.

Typically, the EAP-FAST provides message protection by establishing a secure tunnel via Transport Layer Security (TLS) for protecting the authentication method. Moreover, the TLS is employing a strong entropy shared master secret key that results in the confidentiality and integrity protection of the EAP packets.

5.7.2 Mutual Authentication and Mitigation of Man-in-the-Middle Attacks

Wrapping EAP within another protocol enables a rogue EAP packet authenticator to attack as a MitM by tunneling EAP to a legitimate server. Where the tunneling protocol does not require peer authentication, an attacker convincing a legitimate EU to connect to it will be able to tunnel EAP packets to a legitimate server. This allows the attacker to successfully establish himself as a MitM, gaining access to the multicast session as well as the ability to decrypt multicast content between the CP and the EU.

EAP-FAST mitigates this attack by using the PAC-Key to mutually authenticate both the EU and the EAPS during phase 1 of the EAP-FAST run. In phase 1, the EAP Server and the EU mutually authenticate each other.

5.7.3 Dictionary Attack Resistance

Since password authentication algorithms are known to be vulnerable to dictionary attacks, in general it is recommended to use authentication methods that are resistant to dictionary attacks. However, the EAP-FAST mitigates dictionary attacks by establishing the mutually authenticated and encrypted TLS tunnel to protect even weak credential based authentication methods. Therefore, regardless of the exchanged authentication credentials, EAP-FAST is dictionary attack resistant.

5.7.4 Protection against packet modification and forged clear text EAP packets

Since the identifier field in the EAP packet format is only a single octet, it is fairly easy to guess it and allow an attacker to successfully inject or replay EAP packets. If the header of the EAP packet is not protected, the attacker also will be able to modify EAP headers, which can cause packets to be inappropriately discarded or misinterpreted. Moreover, EAP Success and Failure packets are exchanged in clear text. An attacker may try to forge either a Failure packet to convince an EAP peer to disconnect or a Success packet to convince the EAP peer that authentication has succeeded even though the authenticator has not authenticated himself to the peer.

EAP-FAST provides protection against these attacks by providing message confidentiality and integrity. The EAP-FAST peer does not rely upon unprotected EAP Success and Failure messages. It accepts only Success/Failure decisions indicated by a protected mechanism within the EAP-FAST tunnel. As the result of the strong mutual authentication of the EU and the EAP Server, the TLS tunnel is established securely between the EU and the EAP Server and provides a protected path to exchange Success/Failure messages.

5.7.5 Mitigation of connecting to an Entrusted Authenticator

EAP-FAST provides the server certificate validation capability as part of the TLS negotiation during which the EAP Server presents a certificate to the EU. In order to determine whether the EAP Server can be trusted or not, the EU has to verify the validity of the certificate by examining the EAP Server's name presented in the certificate.

Chapter 6

Validation

6.1 Validation

Since network security has become a very important and complex field of research nowadays, in this section we would like to apply it to the discussion of exchanging sensitive data over the Internet through multicast communications, finding an appropriate and trustworthy mechanism to map the proposed frameworks and methods to the mechanism and eventually validating them.

6.2 AVISPA: The Validation Tool

A purely mathematical and analytical discussion on the problem of validating such protocols (more specifically, frameworks and methods) may quickly exhaust the reader. So, we tried to find a tool that can be used to automate the whole work of validating the exchanges within those frameworks and methods. In this chapter, we will provide a formal tool-based analysis of the claimed security properties for participant exchange controls.

The *AVISPA* (Automated Validation of Internet Security Protocols and Applications) Model Checker is a modern security tool to prove the correctness and attack-free working of the Internet security protocols. The tool provides a programming language called High Level Protocol Specification Language (HLPSL) for describing proposed security protocols, frameworks and methods as well as specifying their prospective and intended security features. It is worthwhile to note that mapping the input correctly to the AVISPA model is crucial to demonstrate the output results. Then, the AVISPA randomly considers all possible events to try and track as various simulation scenarios. The tool analyzes them systematically and explores the results whether it finds an attack and reports an unsafe message or finds no attack and reports a safe message. Obviously, the final results are reported according to the set of goals that has to be defined and determined precisely in the HLPSL code. Therefore, a safe result represents that no attacker can break the pre-defined security goals but it does not justify mitigation against other types of vulnerabilities.

The internal translator of the AVISPA tool translates the HLPSL code into the Intermediate Format (IF), which is read directly by a back-end. The AVISPA tool comprises four back-ends [32] as shown in Figure 10:

1. On-the-fly Model-Checker (OFMC) employs several symbolic techniques to explore the state space in a demand-driven way.

2. CL-AtSe (Constraint-Logic-based Attack Searcher) applies constraint solving with simplification heuristics and redundancy elimination techniques.

3. The SAT-based Model-Checker (SATMC) builds a propositional formula encoding all the possible traces (of bounded length) on the protocol and uses a SAT solver.

4. TA4SP (Tree Automata based on Automatic Approximations for the Analysis of Security Protocols) approximates the intruder knowledge by using regular tree languages and rewriting to produce under and over approximations.

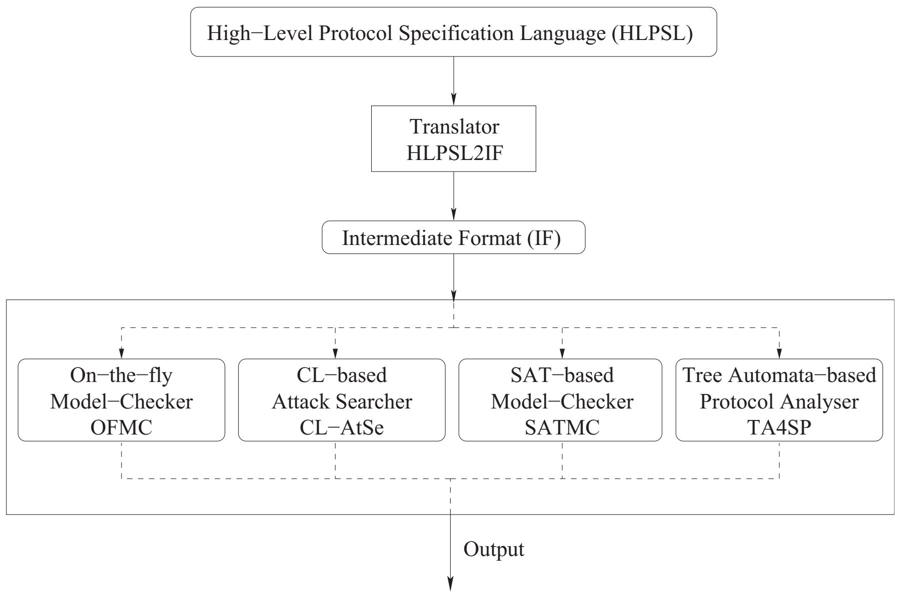


Figure 10: AVISPA back-ends

This list may later be extended with new back-ends for other purposes. However, since more than 85% of the attacks and threats can usually be found by the first two back-ends (according to the AVISPA guidelines) [32, 33] and the other two are not designed to track our particular security goals, we used the CL-AtSe back-end in addition to the default invoked back-end, OFMC.

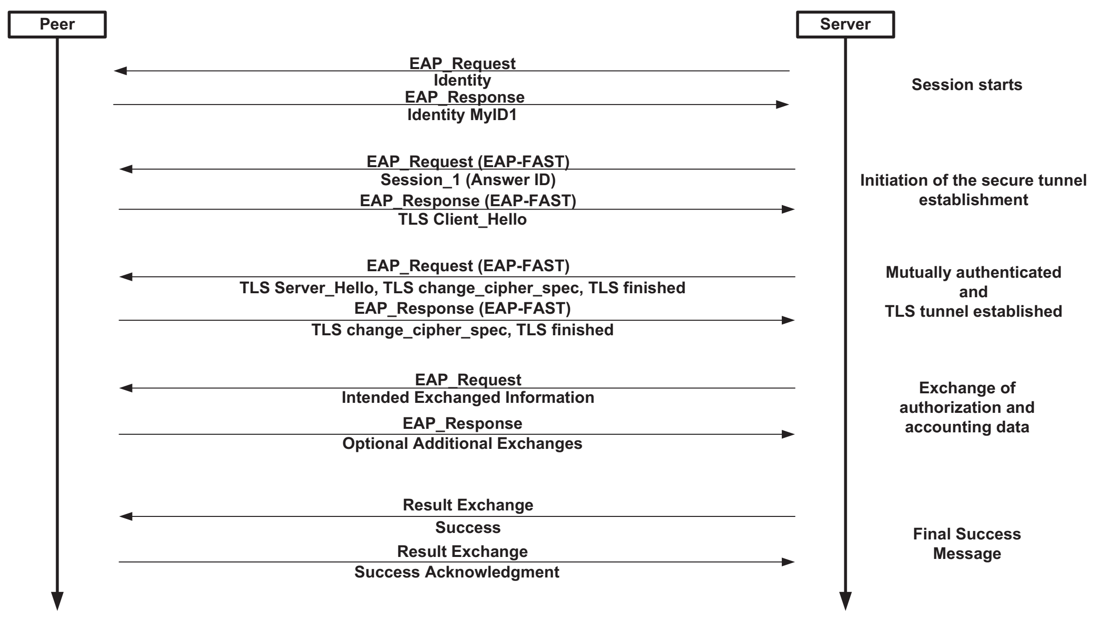


Figure 11: EAP-FAST message exchange

6.3 The Developed AVISPA Model

6.3.1 The Plan

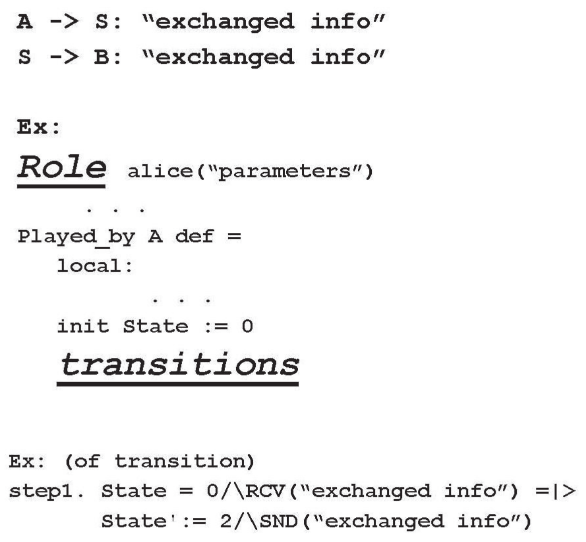
In order to model EAP-FAST with HLPSL language, see figure 11, there is a set of message exchanges between the peer (as the EU) and the server (as the EAPS). We have modeled our protocol through the following main steps.

It starts with an "EAP request" and "EAP reaponse" between the peer and the server. Then, EAP-FAST is established through a three-way handshake for establishing the TLS tunnel. Eventually, once the TLS tunnel is established and consequently the newly-generated secret keys are exchanged between the peer and the server, Authorization and Accounting information are transferred securely using the new keys. The final success messages terminate the protocol.

6.3.2 HLPSL Features

Here, first we are going to explain some of the HLPSL features, which helped us coding our protocol with HLPSL and defining the correctness of our model. Then, in section 6.3.4, we will explain various parts of our protocol that we have considered to develop our model.

1. In the real world of multicast communications, there are several actors playing their role. In this thesis, in order to being able to validate the security of participant control exchanges, we consider two of the participants work as the main actors: The EU and the EAPS (co-located with the AAAS). In HLPSL as a role-based language, we can define a "basic role" that specifies the actions of each kind of actors in a module. For each type of actor in the protocol, there is one basic role defining his sequence of actions. Here, we have defined "auth-server" to act as the EAPS and also defined "peer" to act as the EU.



```
A -> S: "exchanged info"
S -> B: "exchanged info"

Ex:
Role alice("parameters")
      . . .
Played_by A def =
   local:
           . . .
   init State := 0
   transitions


Ex: (of transition)
step1. State = 0/\RCV("exchanged info") =|>
       State':= 2/\SND("exchanged info")
```

Figure 12: Basic role in HLPSL

2. Since each actor in real world communicates with other actors, HLPSL enables us to define a set of parameters for each basic role. The section of Global parameters for each actor includes a prototype of other agents (specific instant of a basic role), with whom the actor communicates. Also it includes hash functions to encrypt data streams, send channels and receive channels. Each actor should have a separate "SND" channel to

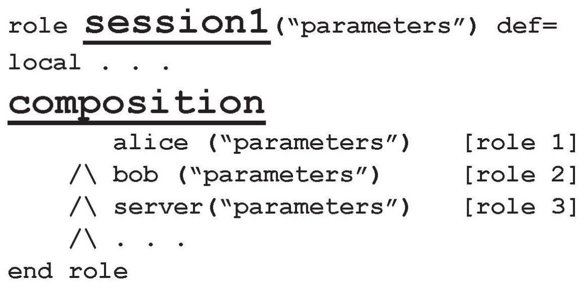
every other actor as well as a separate "RCV" channel from each of them. The section of local parameters, mostly includes a prototype of various nonces. A nonce may be used as a key, cipher suite, certificate and session id. In this section, local variable of "state" is also defined as a "natural" variable to be able to transit the state of the actor to model his actions in the real world.

3. Constants are also available to be defined to model "request-id", "response-id", "protocol-id" and "start-eap-fast" flag.

4. In real world of multicast communications, actions of actors means producing and possessing data streams, sending information, receiving information, hashing data streams and performing mathematical operations, all of which are possible to model in HLPSL. In fact, we can define different states for each actor and model the actions by different transitions from one state to another one.

5. A basic role with all its parameters, states and transitions best models an actor (See figure 12).

6. Once every individual basic role is defined in HLPSL, we need to model communications of the actors. For that reason, we can have a "composition" of different actors in a "session", which in fact models a set of actors and all their communications. Communications in a session are bounded within that session that is, no outside actor is able to join the communications at this step (see figure 13).



```
role session1("parameters") def=
local . . .
composition
       alice ("parameters")    [role 1]
    /\ bob ("parameters")      [role 2]
    /\ server("parameters")    [role 3]
    /\ . . .
end role
```

Figure 13: Composition of various roles to make a session

It is important to ensure that whatever is sent through a "SND" channel from actor A to actor B, is actually received by actor B at a moment. Otherwise, AVISPA will not be able to validate *secrecy* goal (see section 6.4 for further information about security goals).

7. In order to model the real multicast world more generically, HLPSL enables us to carry out more than one session simultaneously. We can model many multicast sessions over a specific multicast service. Consequently, more than one EU is acting and so the EAPS is. As shown in figure 14, a composition of many sessions is called "environment" in HLPSL. Therefore, many EUs and many EAPSs are present and communicate with each other in an environment. AVISPA engines generate sessions as much as needed and keep looking to find a violation of security goals between various actors from different sessions.

```
role environment() def=
    . . .
intruder knowledge = {. . .}
  composition
        session1("parameters")
     /\ session2("parameters")
     /\ . . .
end role
```

Figure 14: Composition of various sessions to make an environment

8. Generally speaking, an attacker has a set of information and knowledge about his target. Although we made some security assumptions in section 5.4.5, at attacker may have a prior knowledge about the number of actors, location of actors, path of communication between the actors, public keys of the actors and some pre-defined hash functions. Defining the intruder knowledge enables us to model as accurately as possible. There is an opportunity to add to the intruder's knowledge for future developments too.

6.3.3 Developing the HLPSL model

Developing the HLPSL code by means of the tools (see section 6.3.2) was a straightforward and sometimes tricky procedure. Here, we go into the steps that we passed to achieve the final results.

1. In the protocol, there are certain actors: The EU, the AR and the EAPS, playing their roles. Since the AR is a forwarder of messages between the other two and is not even able to interpret the token's content (except the routing information), we considered only the EU and the EAPS as main actors to be modeled with HLPSL. There is no need to model an actor that does not possess any message to exchange and only forwards messages from the EU to the EAPS and vice versa.

2. In the real world of multicast, there are thousands of EUs asking for a specific multicast application. Also each one may request different multicast data streams from an NSP. Therefore, we need to invent a way to distinguish an EU from others as well as any request of the EU from its other requests. At the beginning we did not model this step for each EU. An EU could start the negotiation with an EAPS and "start-eap-fast" right away. Gradually, we improved the code with a "request-id" from the EU and a "response-id" from the EAPS to be able to assign a random and unique number to each request of each of the EUs and consequently distinguish different requests of the same EU.

3. While an EU is able to make different requests in one session, similar to have a way to identfy the EU itself and its request, we improved our code with a session ID to be able to track the exchanged information within a multicast session. Without the "SessID" variable, in a composition of basic roles within one "session", it is not possible for AVISPA engines to track an exchanged message to monitor secrecy of the message. However, by means of "SessID" and concatenation of that to a set of exchanged information, as long as the whole string is not taken by another agent (obviously with a different "SessID" that might be an attacker), secrecy of the string is met.

   AVISPA engines have the ability to monitor a piece of exchanged information to see if it is possible for others (instead of the intended recipients) to capture it. In such a case, it reports an unsafe protocol. "SessID" provides this ability in our code.

   Before adding "SessID" to our code, we were not sure that once a piece of information is exchanged whether it is actually received by the intended receiver or not. Adding "SessID" to our code, fixed this shortage and increased the accuracy of one of our main goals, secrecy of exchanged information.

4. Strong mutual authentication is one of the most important features of the EAP-FAST that we had to compltely model it in HLPSL. As opposed to 'weak authentication', 'strong authentication' is used in the literature and it means one party authenticates

the other one not only based on a simple one-way exchange of identity but also, encryption and concatenation of nonces and identity are involved. Fortunately, AVISPA engines are able to validate strong authentication of each of the actors. During a few first steps, we failed modeling strong authentication with HLPSL. However, we eventually reached the correct code and here are the tips towards modeling strong authentication:

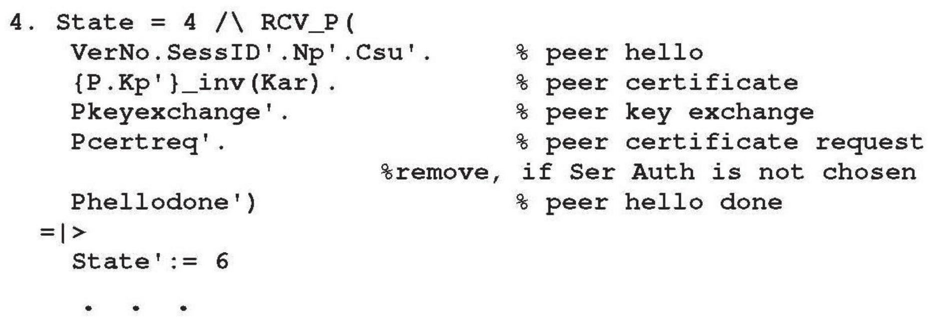
- In the first round of exchanging authentication information from A to B, once a nonce is generated, a "SessID" as an identifier must be attached to it. An encryption of the generator agents's name and its public key-encrypted with the opaque part of the PAC (called 'Kar' in the code) is attached to the string too.
- In the second round of exchanging information from B to A, the message from the previous round must be decrypted using the same 'Kar'. B is able to split the message into its components and use them to build the new string to send. B must generate its new nonce, concatenate his name and A's nonce to it, encrypt them together and add the result to the encrypted message he has received from A. The whole thing is now sent to A using a hash function (common between A and B).

```
% including option of the server authentication

 5. State  = 5 /\ RCV_S(VerNo'.sid0.Ns'.Csus')% server hello
    =|>
    State':= 7
     /\ Np'    := new()
     /\ SessID' := new()
     /\ Phellodone'  := new()

    . . .

     /\ SND_S(VerNo'.SessID'.Np'.Csu'.    % peer hello
                    {P.Kp}_inv(Kar).  % peer certificate
                    Pke'.            % peer key exchange
                    Pcertreq'.    % peer certificate request;

                         %Drop. If Server Auth is not chosen

                    Phellodone')        % server hello done
```

Figure 15: Server authentication is optional (peer part)

- On each step, we must correctly code the prototype of the exchanged message for the intended receiver. The prototype must be defined for the receiver, exactly similar to what has been modeled for the sender (generator of the nonce and the exchanged message).
- Server authentication in EAP-FAST is optional. In our HLPSL code we modeled this feature too. In fact, in the “role peer” in state 5 and 7, “Phellodone” must be removed from the code (see figure 15) to skip server authentication and in the “role auth-server” in state 4, the same part of the message (“Phellodone”) must be removed from the receiver message too (See figure 16).

  This means the peer does not wish to authenticate the EAP server. Therefore, the whole part in “SND-P” in the “role auth-server” in state 4 must be removed (since no more server certificate is going to be exchanged) and so must be the whole first “RCV-S” in the “role peer” in state 5 and 7.



```
4. State = 4 /\ RCV_P(
    VerNo.SessID'.Np'.Csu'.      % peer hello
    {P.Kp'}_inv(Kar).            % peer certificate
    Pkeyexchange'.               % peer key exchange
    Pcertreq'.                   % peer certificate request
                         %remove, if Ser Auth is not chosen
    Phellodone')                 % peer hello done
  =|>
    State':= 6
     .  .  .
```

Figure 16: Server authentication is optional (server part)

- In order to validate the modeled strong mutual authentication, based on the set of exchange information, we must ask AVISPA engines to monitor authenticity of A by B by means of a set of generated and exchanged nonces and vice versa.

5. Establishing the TLS tunnel means exchanging a pair of fresh secret keys after the

strong mutual authentication of both parties. In our model, we have successfully established the TLS tunnel and the next step was to exchange a stream of data as 'authorization' and 'accounting' information of the EU. We have defined "Finished" variable to encrypt it by new "ClientK" and "SeverK" secret keys and exchange it between the two agents. See figure 17.

```
role peer . . .

 =|>
State' := 9

 /\ ClientK'   := KeyGen(P.Ns.Np.PRF(PMS'.Ns.Np))
 /\ SND_S(Ccs'.{Finished'}_ClientK')
 /\ request(P,S,nps2,Ns.Np)
end role


role auth_server . . .

  6. State  = 6 /\ RCV_P(Ccs.{Finished}_ClientK) =|>
     State':= 8 /\ secret(ServerK,sec_serverK,{S,P})
                /\ secret(ClientK,sec_clientK,{S,P})
                /\ request(S,P,nps1,Ns.Np)
end role
```

Figure 17: Secure exchange of further information

For each of the agents, we had to add the agents's cipher suite too to the beginning of authorization and accounting information ("Finished") of that agent to be able to give a hint to AVISPA engines for tracking purposes of "Finished".

6.3.4 Our Model

In this section we describe how we modeled the proposed IETF document of the EAP-FAST method with HLPSL language to meet the AVISPA validation requirements.

- The EAP-FAST method uses a Protected Access Credential (PAC) to establish a secure TLS tunnel in which intended client credentials will be verified. EAP-FAST has two main phases and an optional phase 0. In Phase 1, the EU and the EAP Server uses the PAC to establish the TLS tunnel and in Phase 2, the EU credentials are exchanged, inside the secure tunnel. Typically, in Phase 0, the PAC can be provisioned manually or dynamically.

Phase 0 is outside the scope of the IETF document [13]. It does not have any effect on the nature of the two other phases either and has to be done before the first authentication step starts. In addition, considering the HLPSL privilege on variables, the PAC in EAP-FAST is modeled by means of a global variable 'Kar' that is defined as a *public key* to enable us to have a protected access information in our AVISPA model. Therefore, there is no need to consider provisioning the PAC in our model.

- The HLPSL language is a role-based language, which means we determine the sequence of actions and properties of each kind of participant in a module called a basic role. Here, we define two general types of basic roles: the *role peer* and the *role authentication server*, which gives us the ability to describe the two required agents and the way of exchanging security sensitive data between those agents.

  Agents are the Peer (which represents the EU) and the Server (which represents the EAP Server). A definition of the role is given as: 'played by', where we specify the agent and its local variable(s).

- We defined global variables for each role as well as local variables and constants. Usually, transmission channels (sending) to and (receiving) from other agents and public keys are defined as global variables while different nonces, session IDs, hash functions and secret keys are defined as local variables. Local variables may transfer between the agents through the transmission channels. This is the actual way that we naturally modeled exchange of nonces into the HLPSL language.

- In HLPSL, local variables have the ability to be changed and transferred but they cannot be shared. Yet, it is possible to share a constant value as well as to negotiate on the value of a global variable and share that one whenever we require roles to have pre-shared knowledge (e.g., a shared key).

- In the *transition* section, for each role we specified different *state(s)* to be able to model communications of the two agents, generating new strings of bytes mostly as nonces, sending and receiving credentials, performing algebraic operations as well as hashing on nonces; all regarding the secret key generation and exchange authentication of the agents and establishment of the secure TLS tunnel.

  As shown in Figure 11, exactly the same steps as in a real run of the EAF-FAST are done in our model to provide the strong mutual authentication and the secure TLS tunnel. As a matter of fact, the verification of the certificates is the strong mutual authentication and the exchange of the fresh secret keys is the establishment of the secure TLS tunnel, which enables both agents to transfer further intended information (for authorization and accounting purposes) in a secure encrypted way.

It is worthwhile to note that our model optionally sends the server certificate to the peer for verification in order to model the optional authentication of the server in the EAP-FAST method.

- We assigned state numbers to reflect the intended order of the send and receive events. That is, we modeled the order of exchanging information of the EAP-FAST in the HLPSL language and have them performed in the same order as they are done in the actual EAP-FAST method (see Figure 11). As an example, the EU first has to initiate an EAP session with his request to the EAP Server and receives the response back, then he can start establishment of the TLS tunnel. Therefore, it is required to assign a lower number to the first state and a higher number to the next one for both of the agents to follow the order.

- In order to have one whole protocol session, we defined a *composed role*, which instantiates one instance of each basic role actually by gluing them together so that they execute together in parallel with interleaving semantics. This enables us also to define an intruder session with certain knowledge of the public symmetric keys, channels and agents. This is a great capability of AVISPA to execute more than one session in parallel and place an intruder in a session to simulate various events and eventually justify the validation results. In fact, AVISPA looks for any possible scenario that can result in violating any of the security goals. Therefore, a *safe* summary result strongly claims that the protocol is attack free according to the model.

6.4 Specifying the AVISPA Security Goals

In order to have a meaningful validation of the security properties by AVISPA, in addition to model the protocol itself with HLPSL, it is required to specify the security goals in HLPSL too. It can be done by augmenting the transitions of the basic roles with so-called *goal facts*. Any kind of security 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, we need to then assign them a meaning by describing them in the HLPSL *goal* section. This will clarify what combination of such facts indicates an attack.

From our perspective of view as a user, we believe that we correctly *request*ed the AVISPA model checker to *witness* the *secrecy* of the intended data carefully and stated the goal section in the HLPSL properly according to the generic attack model (in section 5.6.3). In the goal section of our HLPSL code (see Figure 18), we explicitly ask the AVISPA model checker to validate the secrecy of both the peer and the server's fresh secret keys, which ensures the intended security of further communications (in the TLS channel); and also to validate the mutual authentication of both agents on their own nonces, which ensures the strong mutual authentication. Therefore, we claim that the final *safe* report is accurately obtained.

```
goal

        %secrecy_of ClientK, ServerK
        secrecy_of sec_clientK, sec_serverK

        %Peer authenticates Server on nps1
        authentication_on nps1
        %Server authenticates Peer on nps2
        authentication_on nps2

end goal
```

Figure 18: The goal section in our HLPSL code

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 the 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.

In fact, we used AVISPA to validate two main classifications of security goals:

- secrecy
- authenticity

The secrecy goal confirms the inability of a third party to discover even a part of the content exchanged between a sender and a receiver that is supposed to be kept secret. The authenticity goal verifies a distinguishing identifier (e.g., a newly-generated nonce encrypted by a pseudo random function) 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 (credentials) that corroborates the binding between the agent and the identifier.

Given the general concept of these two main security goals and the possibility of using them in accordance with any kind of exchanged data with choosing any agent in the model as the sender and receiver of the data, we believe they are capable of validating the protocol against a wide range of attacks and threats including what we proposed in section 5.6.3.

As a simple example on secrecy where the goal facts assert which values should be kept secret between whom and in the goal section they are described, any time the intruder learns a secret value which is not explicitly a secret between him and someone else, it should be considered an attack.

6.5 The AVISPA Results

We developed our code in HLPSL language to model the EAP-FAST method. Given the set of goal facts and goal section that we have defined for our AVISPA model, no attack is found by the two OFMC and CL-AtSe back-ends and the summary results are **safe**. This shows that considering our generic threat model, the EAP-FAST method in reality mitigates against all those potential attacks and threats.

Chapter 7

Conclusion and Future Work

In this thesis we have proposed a complete survey (in terms of required features for multicast content delivery technology) of various EAP methods and chosen the most suitable one. We discussed that the EAP-FAST best suits the secure IP multicast technology to satisfy the set of functional and security requirements. We also explained in detail the superiority of EAP-FAST over other relevant EAP methods. Then, we modeled the method by means of HLPSL description language and proposed a generic threat model accordingly, to validate the intended security properties of the EAP-FAST method successfully by AVISPA.

Plans for prospective researchers might be discovering the functionality requirements of the EAP-FAST in detail and precisely specifying the content of the token to add an acceptable level of functionality to the main project, multicast content delivery. Also, a correct implementation of the project is a must in the industry.

The results of this project are published by Parham and Atwood [34] in the 9th annual conference of Privacy, Security and Trust 2011.

Appendix A

The HLPSL Source Code

```
%EAP-FAST
%HLPSL: Mutual Auth.  & Key (Tunnel) Establishment

% Server Auth (certificate) is optional

%Once the clients connect to the network, he can have the 'Kar'. So, option of the PAC i
%OR, the client receives a 'Kar' in his token from the MR.

%Request/Response
%Start EAP-FAST, Establish TLS
%Hello (1)
%Hello (2)
%Certificate (1) =|> Peer Authentication
%Certificate (2) : Optional =|> Server Authentication
%Key exchange (1)
%Key exchange (2)
%Change Cipher (1) , (2)
%Finished



role auth_server (S, P           : agent,
           H, KeyGen, PRF      : hash_func,
%                       : hash_func,
           Kar, Ks        : public_key,
           SND_P, RCV_P   : channel (dy))
played_by S def=



local  Ns, Csus, PMS                    : text,
         SessID                              : text,
% Ph,
         Np, VerNo, Csu, Pcertreq         : text,
%        Sc, Cke, Cv,
Pkeyexchange, Phellodone, Cciphspec     : text,
         State                          : nat,
         Finished : hash(hash(text.text.text).agent.agent.text.text.text),
         ClientK, ServerK : hash(agent.text.text.hash(text.text.text)),
         Kp                             : public_key,
   %      Nps                            : text.text

  const sec_clientK,
        sec_serverK,
        nps1, nps2 : protocol_id,
        sid0       : text,  % session id = 0
        request_id : text,
        respond_id : text,
        start_eap_fast  : text

  init State := 0

  transition

  0. State  = 0 /\ RCV_P(request_id) =|>
     State':= 2 /\ SND_P(respond_id.S)

  2. State  = 2 /\ RCV_P(start_eap_fast) =|>
     State':= 4 /\ Ns'   := new()
                /\ Csiphspec' := new()
                /\ VerNo'  := new()
```

```
              /\ SND_P( VerNo'.sid0.Ns'.Cciphspec' )   % server hello (SeID=0)



  % including option of the server auth

  4. State = 4 /\ RCV_P(
              VerNo.SessID'.Np'.Csu'.         % peer hello
              {P.Kp'}_inv(Kar).           % peer certificate
              Pkeyexchange'.                       % peer key exchange
              Pcertreq'.                     % peer certificate request
    %Drop, if Ser Auth is not chosen
              Phellodone')                       % peer hello done
     =|>
      State':= 6
       /\ PMS'      := new()
       /\ Cciphspec'      := new()
       /\ Finished' := H(PRF(PMS'.Ns.Np').S.P.Ns.Csu'.SessID')
       /\ ServerK'  := KeyGen(S.Ns.Np'.PRF(PMS'.Ns.Np'))
       /\ ClientK'  := KeyGen(P_.Ns.Np'.PRF(PMS'.Ns.Np'))
       /\ SND_P({S.Ks}_inv(Kar).           % server certificate
    %Drop, if Ser Auth is not chosen
              {PMS'}_Kp'.                 % server key exchange
              {H(Ns.Np'.P.PMS')}_inv(Ks). % server certificate verify
              Cciphspec'.                      % change cipher spec
              {Finished'}_ServerK')       % finished
       /\ witness(S,P,nps2,Ns.Np')


  6. State  = 6 /\ RCV_P(Ccs.{Finished}_ClientK) =|>
     State':= 8 /\ secret(ServerK,sec_serverK,{S,P})
              /\ secret(ClientK,sec_clientK,{S,P})
              /\ request(S,P,nps1,Ns.Np)

end role
```

```
----------------------------------------


role peer (S, P           : agent,
             H, KeyGen, PRF      : hash_func,
%                       : hash_func,
             Kp, Kar        : public_key,
             SND_S, RCV_S   : channel (dy))
played_by P def=

  local  Np, SessID                        : text,
         PMS                             : text,
         Ns, Csus, VerNo, Csu, Pke : text,
% Sc, Cke, Cv, Sh,:text,
         Ccs, Pcertreq, Phellodone         : text,
         State                           : nat,
         Finished : hash(hash(text.text.text).agent.agent.text.text.text),
         ClientK, ServerK : hash(agent.text.text.hash(text.text.text)),
         Ks                              : public_key

  const nps1, nps2 : protocol_id,
        sid0       : text,  % session id = 0
        request_id : text,
        respond_id : text,
        start_eap_fast  : text

  init State := 1

  transition

  1. State  = 1 /\ RCV_S(start) =|>
     State':= 3 /\ SND_S(request_id)


  3. State  = 3 /\ RCV_S(respond_id.S) =|>
     State':= 5 /\ SND_S(start_eap_fast)
  % including option of the server authentication

  5. State  = 5 /\ RCV_S(VerNo'.sid0.Ns'.Csus')     % server hello
     =|>
      State':= 7
       /\ Np'   := new()
       /\ SessID' := new()
       /\ Phellodone'  := new()
       /\ Pcertreq'  := new()
       /\ Pke'  := new()
       /\ Csu'  := new()
       /\ SND_S(VerNo'.SessID'.Np'.Csu'.         % peer hello
                {P.Kp}_inv(Kar).             % peer certificate
                Pke'.                        % peer key exchange
                Pcertreq'.                      % peer certificate request;
%Drop. If Server Auth is not chosen
                Phellodone')                       % server hello done
       /\ witness(P,S,nps1,Ns'.Np')


  % with the option of server auth

  7. State  = 7
       /\ RCV_S({S.Ks'}_inv(Kar).            % server certificate;
%Drop, if Server Auth is not chosen
                {PMS'}_Kp.                   % server key exchange
                {H(Ns.Np.P.PMS')}_inv(Ks').  % server certificate verify
                Ccs'.                        % change cipher spec
                {Finished'}_ServerK'         % finished
               )
       /\ Finished' = H(PRF(PMS'.Ns.Np).S.P.Ns.Csu.SessID)
       /\ ServerK'  = KeyGen(S.Ns.Np.PRF(PMS'.Ns.Np))
     =|>
     State' := 9
       /\ ClientK'  := KeyGen(P.Ns.Np.PRF(PMS'.Ns.Np))
       /\ SND_S(Ccs'.{Finished'}_ClientK')
       /\ request(P,S,nps2,Ns.Np)




end role

----------------------------------------


role session(P, S           : agent,
             Kp, Ks, Kar    : public_key,
             H, KeyGen      : hash_func,
             PRF            : hash_func)
def=

  local SP, SS, RP, RS : channel (dy)

  composition
           peer(P,S,H,KeyGen,PRF,Kp,Kar,SP,RP)
        /\ auth_server(P,S,H,KeyGen,PRF,Ks,Kar,SS,RS)

end role



----------------------------------------

role environment()
def=

   const p,s             : agent,
         kp, ks, ki, kar : public_key,
         h, keygen       : hash_func,
         prf             : hash_func
   intruder_knowledge = {p,s, h,keygen,prf, kp,ks,kar,ki,inv(ki),

                         {i.ki}_inv(kar)
                        }


   composition
        session(p,s,kp,ks,kar,h,keygen,prf)
%   /\  session(p,i,kp,ki,kar,h,keygen,prf)
    /\  session(i,s,ki,ks,kar,h,keygen,prf)

end role

----------------------------------------
goal


        %secrecy_of ClientK, ServerK
        secrecy_of sec_clientK, sec_serverK

        %Peer authenticates Server on nps1
        authentication_on nps1
        %Server authenticates Peer on nps2
        authentication_on nps2

end goal

----------------------------------------


environment()
```

Bibliography

[1] J. W. Atwood, "An Architecture for Secure and Accountable Multicasting," in *Proceedings of the 32nd Conference on Local Computer Networks*, no. 123, (Dublin, Ireland), pp. 73–78, 2007.

[2] S. Islam and J. W. Atwood, "A Framework to Add AAA Functionalities in IP Multicast," in *Proceedings of the Advanced International Conference on Telecommunications*, (Guadeloupe, French Caribbean), February 2006.

[3] S. Deering, *Host extensions for IP multicasting*. IETF, August 1989. RFC 1112.

[4] J. Kohl and C. Neuman, *The Kerberos Network Authentication Service (V5)*. IETF, September 1993. RFC 1510.

[5] C. Neuman, T. Yu, S. Hartman, and K. Raeburn, *The Kerberos Network Authentication Service (V5)*. IETF, July 2005. RFC 4120.

[6] L. Zhu, P. Leach, and S. Hartman, *Anonymity Support for Kerberos*. IETF, April 2011. RFC 6112.

[7] S. Islam and J. W. Atwood, "A policy framework for multicast group control," in *Consumer Communications and Networking Conference, 2007. CCNC 2007. 4th IEEE*, pp. 1103 –1107, January 2007.

[8] T. Hardjono and B. Weis, *The Multicast Group Security Architecture*. IETF, March 2004. RFC 3740.

[9] C. Metz, "AAA Protocols: Authentication, Authorization and Accounting for the Internet," *IEEE Internet Computing*, December 1999.

[10] S. Islam and J. Atwood, "Sender access control in ip multicast," in *Local Computer Networks, 2007. LCN 2007. 32nd IEEE Conference on*, pp. 79 –86, oct. 2007.

[11] S. Islam and J. W. Atwood, "Multicast Receiver Access Control by IGMP-AC," *Computer Networks*, vol. 53, no. 7, pp. 989–1013, 2009.

[12] B. Aboba, L. Blunk, J. Vollbrecht, J. Carlson, and H. Levkowetz, *Extensible Authentication Protocol (EAP)*. IETF, June 2004. RFC 3748.

[13] N. Cam-Winget, D. McGrew, J. Salowey, and H. Zhou, *The Extensible Authentication Protocol-Flexible Authentication via Secure Tunneling Method*. IETF, January 2007. RFC 4851.

[14] S. Islam and J. W. Atwood, "The Internet Group Management Protocol with Access Control," in *Proceedings of the 31st IEEE Conference on Local Computer Networks*, (Tampa, FL), 2006.

[15] B. Cain, S. Deering, I. Kouvelas, B. Fenner, and A. Thyagarajan, *Internet Group Management Protocol, Version 3*. IETF, October 2002. RFC 3376.

[16] P. Calhoun, J. Loughney, E. Guttman, G. Zorn, and J. Arkko, *Diameter Base Protocol*. IETF, September 2003. RFC 3588.

[17] N. Sultana and J. Atwood, "Secure multicast communication: end user identification and accounting," in *Electrical and Computer Engineering, 2005. Canadian Conference on*, pp. 1691 –1694, may 2005.

[18] P. Judge and M. Ammar, "Security Issues and Solutions in Multicast Content Distribution: A Survey," *IEEE Network*, January/February 2003.

[19] B. Quinn and K. Almeroth, *IP Multicast Applications: Challenges and Solutions*. IETF, September 2001. RFC 3170.

[20] S. Deering, *Group Security Policy Token v1*. IETF, June 2006. RFC 4534.

[21] W. Simpson, *PPP Challenge Handshake Authentication Protocol (CHAP)*. IETF, August 1996. RFC 1994.

[22] L. Blunk and J. Vollbrecht, *PPP Extensible Authentication Protocol (EAP)*. IETF, March 1998. RFC 2284.

[23] N. Haller, C. Metz, P. Nesser, and M. Straw, *A One-Time Password System*. IETF, February 1998. RFC 2289.

[24] C. Metz, *One-Time Password Extended Responses*. IETF, November 1997. RFC 2243.

[25] F. Bersani and H. Tschofenig, *A Pre-Shared Key Extensible Authentication Protocol (EAP-PSK) Method*. IETF, January 2007. RFC 4764.

[26] D. Simon, B. Aboba, and R. Hurst, *The EAP-Transport Layer Security Authentication Protocol*. IETF, March 2008. RFC 5216.

[27] B. Aboba and D. Simon, *PPP EAP TLS Authentication Protocol*. IETF, October 1999. RFC 2716.

[28] P. Funk and S. Blake-Wilson, *EAP Tunneled TLS Authentication Protocol Version 0(EAP-TTLSv0)*. IETF, April 2008. RFC 5281.

[29] H. Tschofenig, D. Kroeselberg, A. Pashalidis, Y. Ohba, and F. Bersani, *The Extensible Authentication Protocol-Internet Key Exchange Protocol version 2 (EAP-IKEv2) Method*. IETF, February 2008. RFC 5106.

[30] E. C. Kaufman, *Internet Key Exchange (IKEv2) Protocol*. IETF, December 2005. RFC 4306.

[31] C. Rigney, S. Willens, A. Rubens, and W. Simpson, *Remote Authentication Dial In User Service (RADIUS)*. IETF, June 2000. RFC 2865.

[32] www.avispa project.org, "Hlpsl tutorial: A beginner's guide to modelling and analysing internet security protocols." IST-2001-39252, The AVISPA team, June 2006.

[33] www.avispa project.org, "Avispa v1.1 user manual." IST-2001-39252, The AVISPA team, June 2006.

[34] M. Parham and J. W. Atwood, "Validation of security for participant control exchanges in multicast content distribution," in *9th Annual Conference on Privacy, Security and Trust*, (Montreal, Quebec, Canada), IEEE and Concordia University, July 2011.