UNIVERSITAT POLITÈCNICA
DE CATALUNYA
BARCELONATECH

Secure Identity Management in Structured Peer-to-Peer (P2P) Networks

Author:
Juan Caubet Fernández

Advisors:
Dr. Óscar Esparza Martín and Dr. José L. Muñoz Tapia

Dissertation Submitted to the Department of Telematics Engineering in Partial Fulfillment of the Requirements for the Degree of *Doctor of Philosophy* of the Technical University of Catalonia (UPC)

Ph.D. Dissertation

August 2015

Acknowledgements

First of all, I would like to express my gratitude to my supervisors, José L. Muñoz and Óscar Esparza, whose expertise, understanding, and patience, added considerably to my graduate experience. I appreciate their vast knowledge and skills in many areas, and their assistance in writing reports. Above all and the most needed, they provided me unflinching encouragement and support in various ways. Their truly scientist intuition has made them as a constant oasis of ideas and passions in science, which exceptionally inspire and enhance my growth as a student, a researcher and a scientist want to be.

I am also deeply indebted to Jorge Mata and Juanjo Alins. Without their guidance, support and good nature, I would never have been able to develop this thesis successfully. I benefited greatly from their ideas and insights. Their involvement with their originality has triggered and nourished my intellectual maturity that I will benefit from, for a long time to come.

I am also specially grateful to Miquel Soriano, and the whole Information Security Group (ISG) in general, for giving me the opportunity to develop this thesis in their group as well as for helping me in everything I have needed during these years; great personal experience and totally enriching.

Some debts are hard to put into words. My research colleagues Carlos Gañán and Sergi Reñé, they know why their names are here.

My last, but not least gratitude is for my parents, who gave me everything and keep on giving; it is difficult to find words to express my gratitude and thanks to both of you.

I realize that not all people who contributed either directly or indirectly to my study are mentioned in this page. From the deepest of my heart, I would like to thank all of you...

Preface

Structured Peer-to-Peer (P2P) networks were proposed to solve routing problems of big distributed infrastructures. But the research community has been questioning their security for years. Most prior work in security services was focused on secure routing, reputation systems, anonymity, etc. However, the proper management of identities is an important prerequisite to provide most of these security services.

The existence of anonymous nodes and the lack of a centralized authority capable of monitoring (and/or punishing) nodes make these systems more vulnerable against selfish or malicious behaviors. Moreover, these improper usages cannot be faced only with data confidentiality, nodes authentication, non-repudiation, etc. In particular, structured P2P networks should follow the following secure routing primitives: (1) secure maintenance of routing tables, (2) secure routing of messages, and (3) secure identity assignment to nodes. But the first two problems depend in some way on the third one. If nodes' identifiers can be chosen by users without any control, these networks can have security and operational problems. Therefore, like any other network or service, structured P2P networks require a robust access control to prevent potential attackers joining the network and a robust identity assignment system to guarantee their proper operation.

The main objective of this thesis is to provide methods for managing identities in DHT-based structured P2P networks. This research intends to develop new protocols to control the user access, to assign node identifiers in a secure way and to manage revocation data, improving the accessibility and avoiding bottleneck problems. These new protocols should be used to provide security in structured P2P networks, and thus allowing the deployment of commercial applications and new services.

Contents

List of Figures

List of Tables

Chapter 1

General Introduction

Contents

1.1 About this Thesis

This thesis has been carried out at the Information Security Group (ISG)[1] of the Department of Telematics Engineering (ENTEL)[2] at the Technical University of Catalonia (UPC)[3]. We would like to thank the Spanish Research Council that has partially funded the development of this thesis under the following projects:

- ARES: Advanced REsearch on information Security and privacy (CONSOLIDER CSD2007-00004).
- SERVET: A SEcure and Robust transport architecture for infotainment services in Vehicular nETworks (CICYT TEC2011-26452).

[1]ISG home: `http://isg.upc.edu`
[2]ENTEL home: `http://www.entel.upc.edu`
[3]UPC home: `http://www.upc.edu`

1.2 Context

Most of our daily activities are carried out over the Internet, from home-banking and on-line teaching to watching on-line content, social networking and file sharing. Considering file sharing as one of the Internet top activities, different generations of P2P networks have been proposed, designed and implemented; resulting in a wide set of distributed networks with different performances and goals. Over the last years, these networks have been evolving from centralized approaches to fully decentralized and structured systems (structured P2P networks).

Today, structured P2P networks, also known as P2P overlays, are considered the best way to share/distribute large amounts of data, regardless of the involved devices and underlying networks used below them. This is mainly due to they are mostly based on Distributed Hash Tables (DHT). However, despite their benefits, P2P overlays are hardly being used to provide commercial services. Although these networks have been analyzed in depth to guarantee scalability and efficiency, there are some important security problems that deter their use in open networks. Most P2P overlays neither control the user access nor the behavior of the nodes within their virtual space. In addition, the existence of anonymous nodes and the lack of a centralized authority capable of monitoring (or punishing) nodes make these systems more vulnerable against selfish or malicious behaviors.

Ideally, DHT-based structured P2P networks should follow the secure routing primitives described by Wallach in [121], which are:

1. Secure maintenance of routing tables.

2. Secure routing of messages.

3. Secure identity assignment to nodes.

But the first two problems depend in some way on the third one. One of the main vulnerabilities of most P2P overlays is the fact that the new users can choose their own identifiers within the network (hereinafter referred to as nodeIDs). This allows them to deliberately locate themselves at a certain place within the P2P virtual space where they know they will be responsible for some specific data. In this way, attackers can lie and say these data do not exist whenever someone asks making that content unavailable. Attackers could also lie and send fake data.

On the other hand, the uncontrolled number of nodeIDs per user is also a significant issue. If an attacker is able to manage a considerable set of nodeIDs, she could separate a part of the overlay from the rest, or even improve her own reputation by using good feedback that comes from these fake identities. Therefore, P2P overlays require robust processes of identity assignment to prevent potential attackers can damage the network.

In addition, without security, all nodes are potentially vulnerable to the misbehavior of the attackers. Hence, it is also necessary to evict compromised, defective and illegitimate nodes. One basic solution to achieve these requirements, widespread in other networks, is to use a Public Key Infrastructure (PKI). PKIs were developed to meet with the main security requirements in the Internet environment, adopting the idea of the public key cryptography and employing the digital certificate concept to bind an entity with a cryptographic public key. Many existing solutions manage these certificates by means of a central Certification Authority (CA), although there are also distributed solutions such as [28], which issues a valid certificate for each node of the network. Therefore, an efficient certificate management is crucial for the robust and reliable operation of any PKI. Using digital certificates implies the need to validate them, since they have a limited lifetime and may even be revoked. To revoke a digital certificate, in the traditional approach, the CA adds its serial number to a Certificate Revocation List (CRL) [45]. Then, this CRL is issued and broadcasted to the network.

On the other hand, the network scale of the P2P overlays is expected to be very large. Hence, the distribution of the CRLs is prone to long delays and the CAs are prone to be overloaded. In practice, revocation of misbehaving nodes should take place as fast as possible to prevent these users from jeopardizing the performance of the network.

Consequently, robust identity management systems[1] are necessary to turn P2P overlays into a better platform for commercial applications and services that handle sensitive information.

The work of this thesis follows two axes: (1) how to generate secure nodeIDs; and (2) how the revocation data distribution can be improved, both in current P2P overlays.

[1]Systems that control the information that authenticates the identity of a user, the information that describes the information and actions that users are authorized to access and/or perform, and the descriptive information about users.

1.3 Research Objectives

This thesis aims to mitigate the security issues related to the identity management in structured P2P networks. Therefore, the objectives of this thesis are as follows:

1. Analyze the operation of the current structured P2P networks when managing identities to identify what security problems are related to the nodes' identifiers within the overlay. Specifically, this involves analyzing the nodeIDs generation process, the bootstrapping phase and the user access control.

2. Propose a series of requirements to be accomplished by any generated nodeID to provide more security to a DHT-based structured P2P network. These requirements should consider all possible security issues related to the nodes' identifiers.

3. Propose the use of implicit certificates to provide more security and to exploit the improvement in bandwidth, storage and performance that these certificates present compared to explicit certificates. In our opinion, this type of certificates may be a good solution to increase the security of P2P overlays and to facilitate the use of these networks by resource-constrained devices.

4. Design protocols to assign nodes' identifiers avoiding the identified problems, while maintaining user anonymity and allowing users' traceability. These protocols should involve the issuance of digital certificates to make P2P overlays secure.

5. Analyze the operation of the most used mechanisms to distribute revocation data in the Internet, with special focus on the proposed systems to work in P2P networks.

6. Design a new mechanism to distribute revocation data more efficiently in a structured P2P network. This mechanism should take into account that a P2P overlay may have a large number of users and use nodeIDs with a limited lifetime.

To fulfill these objectives, we have followed four different work lines. The first one analyzes the operation of some of the current P2P overlays and proposes a series of requirements which should be fulfilled by any mechanism to assign nodeIDs. The second work line introduces the use of implicit certificates in P2P overlays. In the third line, we design a protocol which accomplishes all the proposed requirements together.

And finally, the last work line deals with the problem of distributing large amounts of revocation data within a P2P overlay.

1.4 Thesis Organization

This thesis is organized in eight Chapters. Chapter 2 introduces the evolution of the P2P networks over the years and explains in detail the operation of the most representative DHT-based structured P2P networks. Specifically, it focuses on how these networks implement the bootstrapping process, manage the identifiers of the nodes and control the access of the users. Then, the main security issues related with the users' identities in P2P overlays are analyzed to understand their origin. In addition, there is a discussion about how several research works try to avoid, prevent or limit the most common problems related to the nodes' identifiers in P2P overlays, and how some proposals try to distribute revocation data in these networks more efficiently. Finally, this Chapter also provides to non-expert readers enough background in cryptography to read this thesis.

Chapter 3 carries out a discussion about the main problems related to the way of generating nodeIDs, which are used in some current P2P overlays. Then, a series of design requirements to be fulfilled by the nodeIDs of a P2P overlay are also proposed. Finally, we discuss some possible services over P2P overlays that require a tight identity management system and thus, that would benefit from our proposals.

Chapter 4 describes the proposal of an identity assignment protocol that allows the creation of nodeIDs as a cooperative process between the user and the Trusted Third Party (TTP) in such a way that none of the parts has complete control over which is the final identifier assigned. The algorithm is based on implicit certificates.

Chapter 5 describes another robust identity assignment protocol based on the use of implicit certificates. In this case two TTPs are used to also guarantee full user anonymity at the same time that their activities can be traceable.

Chapter 6 presents a generic description of a new identity assignment protocol which uses explicit certificates and two TTPs to provide security and guarantee full user anonymity, traceability and nodeID stability at the same time.

Chapter 7 describes the operation of a new system to distribute revocation data in the Kad network, which stores the revoked certificates in different CRL segments and distributes them independently using that P2P overlay itself.

Finally, Chapter 8 concludes this thesis summarizing the main findings and making suggestions for the future research.

1.5 Related Publications

Most of the research results presented in this dissertation have been published in journals and conferences. In this section we provide a list of such publications, together with their complete bibliographic information. Further, we include other complementary articles that are not directly related with the research topic of this thesis, but which are especially significant from the state-of-the-art perspective.

International Journal publications:

1. Juan Caubet, Oscar Esparza, José L. Muñoz, Juanjo Alins, and Jorge Mata-Díaz. "RIAPPA: A Robust Identity Assignment Protocol for P2P overlAys", *Security and Communication Networks*, vol. 7, issue 12 (December 2014), pp. 2743-2760. DOI: 10.1002/sec.956. **(Impact Factor: 0.72)**

    - In this article we presented a first version of the RIAPPA protocol, a protocol to control the access to a P2P overlay and assign identities in a secure way; all this preserving the anonymity of users. This protocol involves two Trusted Third Parties (TTPs), thanks to which it is possible to preserve the users' anonymity within the network without losing traceability. This protocol also provides revocability and protection against Sybil attacks, Eclipse attacks, whitewashers, etc.

2. Juan Caubet, Carlos Gañan, Oscar Esparza, José L. Munoz, Jorge Mata-Díaz, and Juanjo Alins. "Certificate Revocation List Distribution System for the Kademlia Network", *The Computer Journal*, vol. 57, issue 2 (May 2013), pp. 273-280. DOI: 10.1093/comjnl/bxt037. **(Impact Factor: 0.787)**

    - In this article we proposed a new distributed revocation system for the Kademlia network in order to improve the accessibility, to increase the availability and to guarantee the freshness of the revocation data. To do so, this system distributes the revocation data using the overlay itself and generating CRL (Certificate Revocation List) segments independently.

International Conference publications:

1. Juan Caubet, Oscar Esparza, Juanjo Alins, Jorge Mata-Díaz, and Miguel Soriano. "Securing Identity Assignment Using Implicit Certificates in P2P Overlays", in proceedings of the *7th IFIP WG 11.11 International Conference on Trust Management, IFIPTM 2013*, pages 151–165, Malaga, Spain, June 2013.
   DOI: 10.1007/978-3-642-38323-6_11.

   - In this article we proposed a new nodeID assignment protocol based on the issuance of implicit certificates, which present certain advantages over the use of traditional certificates (explicit certificates). This approach is based on the use of certificates and the joint generation of nodeIDs between a Certification Authority (CA) and the user during the certificate generation process.

2. Juan Caubet, Carlos Gañán, Oscar Esparza, José L. Muñoz, Jorge Mata-Díaz, and Juanjo Alins. "CRL Distribution System for KAD Network", in proceedings of *The 2012 FTRA International Workshop on Human centric computing, P2P, Grid and Cloud computing (HPGC-12)*, Jeju, South Korea, November 2014.

   - In this article we proposed the first version of a new distributed revocation system for the Kademlia network. This mechanism aims to improve the accessibility, increase the availability and guarantee the freshness of the revocation data through the use of CRL (Certificate Revocation List) segments.

3. Carlos Gañán, Juan Caubet, Sergi Reñé, Jorge Mata-Díaz, Juanjo Alins, and Óscar Esparza. "NeuroCast: Adaptive Multi-source P2P Video Streaming Application for Wireless Networks", in proceedings of the *9th IFIP TC 6 International Conference on Wired/Wireless Internet Communications, WWIC 2011*, pages 272-284, Vilanova i la Geltrú, Spain, June 2011.
   DOI: 10.1007/978-3-642-21560-5.

- In this article we presented the design and implementation of NeuroCast: an unstructured P2P application for video streaming. NeuroCast implements a robust scheduling algorithm which minimizes the scheduling delay. Moreover, given heterogeneous contents, delays and bandwidths. Thus, NeuroCast becomes suitable for wireless scenarios due to its capability to adapt to changing network conditions.

Finally, we list the complementary articles mentioned at the beginning of this section.

1. Juan Caubet, José L. Muñoz, and Oscar Esparza. "Utilizando Certificados Implícitos para Asignar Identidades en Overlays P2P", in proceedings of the *XIII Reunión Española sobre Criptología y Seguridad de la Información (RECSI 2014)*, Alicante, Spain, September 2014.

2. Carlos Gañán, Juan Caubet, Oscar Esparza, José A. Montenegro, and Jorge Mata-Díaz. "Estructuras de Datos Autenticadas para gestionar Datos de Revocación en VANETs", in proceedings of the *XII Reunión Española sobre Criptología y Seguridad de la Información (RECSI 2012)*, Donosti-San Sebastián, Spain, September 2012.

3. Juan Caubet, Carlos Gañán, Oscar Esparza, and José L. Muñoz. "Nuevo Sistema de Emisión de CRLs para la Red KAD", in proceedings of *Jornadas de Ingeniería Telemática (JITEL 2013)*, Granada, Spain, October 2013.

4. Juan Caubet, Carlos Gañán, Sergi Reñé, Juanjo Alins, and Jorge Mata-Díaz. "NeuroCast: Aplicación adaptativa multi-fuente para streaming de vídeo en redes P2P inalámbricas", in proceedings of *Jornadas de Ingeniería Telemática (JITEL 2011)*, Santander, Spain, September 2011.

5. Sergi Reñé, Carlos Gañán, Juan Caubet, Juanjo Alins, Jorge Mata-Díaz, and José L. Muñoz. "Analysis of Video Streaming Performance in Vehicular Networks", in proceedings of *The First International Conference on Advanced Communications and Computation (INFOCOMP 2011)*, Barcelona, Spain, October 2011.

6. Juan Caubet, José L. Muñoz, Juanjo Alins, Jorge Mata-Díaz, and Oscar Esparza. "Deploying Internet Protocol Security in satellite networks using Transmission Con- trol Protocol Performance Enhancing Proxies", *International Journal of Satellite Communications and Networking*, vol. 31, issue 2 (2013), pp. 51-76. DOI: 10.1002/sat.1017. **(Impact Factor: 0.744)**

7. Juan Caubet, José L. Muñoz, Juanjo Alins, Jorge Mata-Díaz, and Oscar Esparza. "Implementación de IPsec en una arquitectura TCP splitting", in proceedings of the *XI Reunión Española sobre Criptología y Seguridad de la Información (RECSI 2010)*, Tarragona, Spain, September 2010.

Chapter 2

Background

Contents

2.1 Peer-to-Peer (P2P) Networks

Peer-to-peer (P2P)[1] networks emerged as an incipient paradigm of communications to share resources and services in a highly decentralized way. They have had to evolve over time giving rise to three different generations.

[1]The term *peer* refers to a software process which interacts as an equal with other peers, and it can act both as client and server simultaneously. In this work, we will use the term *node* as a synonym of *peer*.

Among others, the availability of increasingly cheap bandwidth and the growing number of computers sharing services and resources over the years are some factors which have contributed to the success of these networks. P2P networks are being massively used and will remain so in the coming years since their performance is being improved over time and their application is increasingly widespread. According to the annual Cisco Visual Networking Index (VNI) Forecast [42], the file sharing P2P traffic represented the 23.65% of global IP traffic in 2012 and it will grow at a compound annual growth rate (CAGR) of 7 percent from 2012 to 2017. This means that these systems will continue to play an important role in the future Internet for sure; many types of services may be built based on P2P protocols: file sharing applications (BitTorrent [3]), Voice-over-IP (VoIP) services and instant messaging clients (Skype [13]), video streaming applications (CoolStreaming [5]), music sharing portals (Jamendo [9]), file synchronization tools (BitTorrent Sync [4]), payment methods and digital currencies (Bitcoin [2]), Massively Multiplayer Online Games (MMOG) platforms (Badumna [75]), etc. But above all, video streaming applications are the ones that are experiencing a special growth.

IP video traffic, excluding the amount of video exchanged through P2P file sharing, will be 73 percent of all IP traffic (both business and consumer) by 2017, up from 60 percent in 2012. And the sum of all forms of video (TV, Video on Demand (VoD), Internet, and P2P) will continue falling in the range of 80 and 90 percent of global consumer traffic by 2017 [43]. This is because it is expected that more content providers and distributors will adopt P2P as a distribution mechanism. P2P is a low-cost content delivery system but unfortunately until now most content providers and distributors have opted for direct distribution.

P2P video streaming applications emerged as a cheap and efficient solution to provide video streaming over the Internet. PPStream [12] and PPLive [11] are two examples of such applications that have had great success in China. SopCast [14], TVUnetworks [17] and Zattoo [20] are also examples of video streaming applications that have been developed so far. However, most of them are proprietary video streaming platforms, which do not use the last generation of the P2P networks or distribute contents without any type of access control or security. So the last generation of P2P networks is hardly being used for commercial applications, such as pay-per-view video streaming applications, because they have important security problems. Therefore, if we want to use these networks to implement commercial applications (like paid VoD services), it is necessary to solve a series of security problems. This is the main objective of this thesis.

2.1.1 Three Generations of P2P Networks

The operation of the P2P networks has changed over the years, trying to adapt (and survive) to several different problems (even legal), generating up to three different generations.

First generation of P2P networks are known as *centralized P2P networks* because they depend on centralized servers to perform some functions, typically a centralized directory to find resources. This makes them vulnerable from the security point of view. This type of P2P networks suffer from a single point of failure, since the network may stop working if the central server goes down (or court orders to shut it down). The most widely known example of this kind of networks is the initial music sharing service Napster [10], which had legal problems.

The second generation of P2P networks appeared to avoid the above vulnerability, but they needed two phases to achieve the expected success. The first attempt was to design *pure decentralized P2P networks*; which use a P2P scheme in all their processes, and there is no central server at all. They are characterized by the arbitrariness of the links between nodes, and by the use of flooding of messages to search resources. Unfortunately, the use of broadcasting techniques limits their scalability, mainly because they generate congestion or infinite loops. Moreover, searching processes are not deterministic, and they do not guarantee that an unpopular file will be found. The typical example of such networks is Gnutella [8]. The second attempt to improve the P2P networks was to introduce some degree of centralization leading to the *hybrid P2P networks*, where some nodes (*supernodes*) manage certain extra functions. These networks do not suffer from a single point of failure, and searches perform better because they are partially centralized. However, they are still partially vulnerable if some of these supernodes are attacked (or closed), like happened some years ago with the file sharing network eDonkey2000 [6].

Finally emerged the third generation of P2P networks, totally decentralized networks (so there is not a point of failure) but with a certain structuring of resources. For this reason they are called *structured P2P networks*; or also *P2P overlays*, as they create an *overlay*; that is, a logic high-level layer built over an existent network, which is used to structure nodes and connections between them. The objective of this structure is to speed up searches to obtain positive results in a limited number of hops independently of the popularity of the searched resource. Structured P2P networks, or P2P overlays from here on, can provide properties as scalability, fault-tolerance, self-organization, and low-latency. Unlike pure P2P networks, P2P overlays do not allow random connections between nodes. Instead, overlay nodes are connected via virtual links, which are known as *paths*. These paths can be constructed by using different physical links in the lower networks and the way that packets are routed on those underlying networks is not controlled by the overlay. A P2P overlay routing protocol uses the logical identifiers of the nodes (*nodeIDs*) to decide the routing, instead of using directly their IP addresses.

The most typical P2P overlay protocols (CAN [99], Chord [116], Pastry [106], Tapestry [129], Kademlia [85] or BitTorrent [44, 80], among others) are implemented using a DHT [98], which stores $\{key, value\}$ pairs together with the nodeIDs creating a virtual space. A *value* can be a certain resource (for instance, a file), or the way to reach this resource within the overlay (a pointer), and *keys* are used to locate these resources into the network (for instance, the hash of the file name). Moreover, DHTs are divided in subtables, which correspond to a zone of the virtual space, and these subtables are assigned to different overlay nodes. So each node is responsible for a zone, and hence it is responsible for the $\{key, value\}$ pairs contained in that zone (storing contents, or pointers to them, and routing messages). Usually, a zone is assigned to the node whose nodeID is numerically close to the keys stored in the corresponding subtable of the DHT. So if a node wants to download certain content from the overlay, it will send a query message towards the key in the hope that some node with information about that content will receive the message by proximity between the key and its nodeID. If so, the receiver will answer to this query by sending the corresponding file or the pointer to this file if resources are stored in the owner nodes. Therefore, it is obvious that the location of the nodes in the virtual space is directly related to their nodeIDs, and this deserves special attention.

The most widespread P2P overlay is the Kad network [115], a non-commercial file sharing service based on the Kademlia DHT routing protocol and implemented by the popular eMule [7] and aMule [1] clients (among others), and BitTorrent, another non-commercial file sharing service implemented by tens of open-source, freeware, adware or shareware applications [67]. However, nowadays the success of such networks is far from being completely achieved. These networks still have important security problems, which makes them unsuitable for providing certain services, such as pay-per-view video streaming applications.

Many P2P overlay protocols have been analyzed in depth to guarantee scalability and efficiency. Less attention has been paid to the security threats, most P2P overlays assume that nodes behave honestly and neither control the user access nor the nodes behavior. However, this assumption may not be acceptable in open environments like the Internet. The existence of anonymous nodes and the lack of a centralized authority capable of monitoring (or punishing) make these systems more vulnerable against selfish or malicious behaviors. Moreover, these improper usages cannot be faced in these networks with classical security solutions (data confidentiality, nodes authentication, non-reputation, etc.). P2P overlays should also follow the secure routing primitives described by Wallach in [121], which are: (1) secure maintenance of routing tables, (2) secure routing of messages, and (3) secure identity assignment to nodes. But the first two primitives depend in some way on the third one, which is not usually meet in most P2P overlays. Important problems related to the identity of the nodes, *Identity Problem* from here on, arise with the uncontrolled assignment of nodeIDs in these networks [119, 125]: Sybil attacks, Eclipse attacks, Man-In-The-Middle (MITM) attacks, the presence of whitewashers, the non-uniform distribution of nodeIDs, etc.

2.1.2 Identity Management in Existing P2P Overlays

In this Section we summarize the working of some typical P2P overlays: CAN, Chord, Pastry, Tapestry, Kademlia and BitTorrent. We analyze how these P2P overlays work, emphasizing in the aspects related to the bootstrapping[1], identity management and access control.

2.1.2.1 CAN (Content-Addressable Network)

CAN [99] uses a virtual d-dimensional Cartesian coordinate space on a *d-torus* to store $\{key, value\}$ pairs. To do so, a *key* is deterministically mapped onto a point p in the

[1]In a P2P overlay, bootstrapping refers to the process by which one or more internal nodes (bootstrapping nodes) provide initial configuration information to newly joining nodes so that they may successfully join the network.

coordinate space using a uniform hash function. The corresponding $\{key, value\}$ pair is stored in the node that owns the zone within which the point p lies. Each node inside the overlay is responsible for a zone and keeps information on its immediate neighbors. Nodes in the CAN network do not have a nodeID. Instead, they are directly identified by their assigned zone within the virtual space.

Figure 2.1 shows the bootstrapping phase in CAN[1]. Let us consider that a newcomer node N wants to join the CAN network. To do so, it must contact with an internal node (bootstrapping node) B which will guide it until the bootstrapping phase finishes. Then, N randomly chooses a point p in the CAN virtual space, and sends a "join" request towards that point p using the node B as relay. Since B is an internal node, it can use the CAN routing mechanism to forward this "join" message until it reaches the node that manages the zone in which p lays (manager node M). Then, M splits its zone and assigns a half to N, transferring all the $\{key, value\}$ pairs located in that half to N. N also learns from M the IP addresses of its close neighbors, and with that information N can now generate its own routing and neighbor tables.

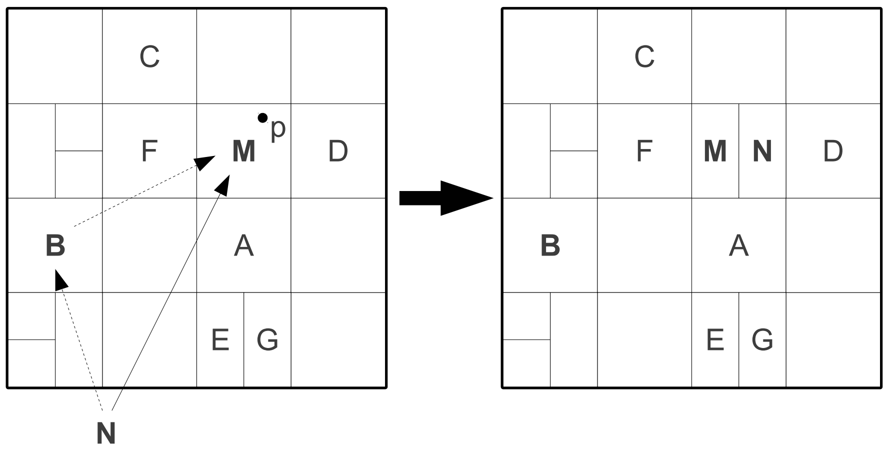


Figure 2.1: CAN bootstrapping phase.

Finally, we must mention that the authors of CAN do not define any mechanism to control who joins the network, so bootstrapping nodes allow any newcomer to join the network.

[1]For simplicity, in this figure we draw the virtual space as a flat plane, but remember that the real space is a *d-torus*.

2.1.2.2 Chord

The virtual space of Chord [116] is circular. NodeIDs and resources (*keys*) are ordered according to a circular identifier that uses a modulo 2^m operation. Each $\{key, value\}$ pair is stored in the *successor* node of the *key* k (denoted as *successor(k)*), i.e., in the first node whose nodeID is equal to or follows the corresponding *key*. Values can be directly resources, like a file, and the *keys* of the resources are generated using a hash function over their names. NodeIDs are also constructed using a hash function, specifically over the users' IP addresses. The goal is to balance the load of the overlay between all the nodes. To ensure this, hash functions used must have the property that their outputs are equidistant between them with high probability.

In Figure 2.2 we can see an example of an identifier circle modulo 2^3 in which there are three nodes (nodeIDs 0, 1 and 3) and three resources (*keys* 1, 2 and 6). In this particular example, node 1 stores the $\{key, value\}$ pair corresponding to the *key* 1, node 3 stores the pair corresponding to the *key* 2, and node 0 stores the pair corresponding to the *key* 6.

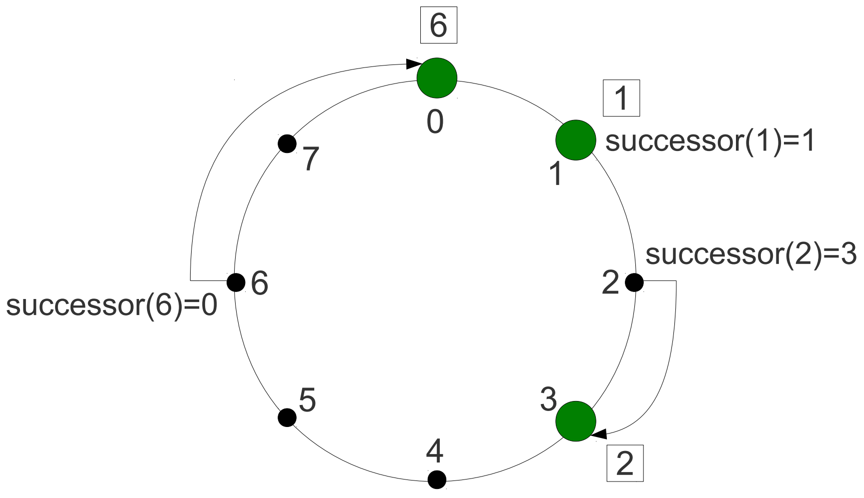


Figure 2.2: Example of a Chord identifier circle modulo 2^3.

A bootstrapping node B helps the newcomer N during the joining process, mainly to initialize its state and add itself to the existing Chord identifier circle. N contacts with the node that currently manages the zone that should be transferred to it (manager node M). This is possible since B is an internal node and it can use the routing mechanisms of the overlay to lookup M. Then, M transfers the corresponding $\{key, value\}$ pairs to N and also shares its routing information to help N to construct its routing table. Neighbors are also informed about the presence of N to properly re-establish the overall routing of the overlay.

Regarding the access control during the bootstrapping phase, the authors mention that this task could be delegated to an external server or even to an internal node, but they do not explicitly define any mechanism.

2.1.2.3 Pastry

In CAN or Chord, the overlay is responsible for storing resources, i.e., a creator node (a node that wants to introduce a resource) delivers resources to the overlay (in particular to a manager node) which is responsible for its storage and management in a structured way. Instead, Pastry [106] is only responsible for publishing the location of resources (in a structured way), but this resources continue being stored in the creator node. The virtual space of Pastry is also circular, i.e., an *identifier circle* modulo 2^m with $m = 128$ bits, and *keys* are computed as the digest of the name and the owner of the resources concatenated. The location of a resource is stored in a node if its nodeID is the numerically closest to the *key*. To lookup resources and route messages, each node not only manages a *routing table*, but also a *neighborhood set* and a *leaf set*. All these state tables are used to route messages in a small number of hops taking into account the geographic location of the nodes, unlike CAN or Chord.

The bootstrapping phase of Pastry is quite similar to that of Chord, and nodeIDs can be computed in two ways, using a hash function over the node's public key or the node's IP address. In most cases, this last option is the one used. Like in Chord, the use of a hash function assures that the computed nodeIDs will be uniformly distributed in the identifier circle, and with high probability, diverse in geography, ownership, jurisdiction, etc.

Regarding the user access control at bootstrapping phase, the authors do not also describe any mechanism to do that.

2.1.2.4 Tapestry

Tapestry [129] uses routing and location schemes similar to the ones presented by Plaxton et al. in [98]. When a creator node wants to publish a certain resource (for instance, a file) in the overlay, it sends a message towards the *key* of that resource, in the hope of reaching the responsible node of that *key*. Specifically, this node is responsible for storing the $\{key, value\}$ pair of that resource, and it is chosen because its nodeID is the numerically closest to the *key*. As Pastry does, Tapestry publishes the location of resources instead of storing the resources themselves, and the routing scheme also takes into account the location within the network to route messages. In Tapestry, multiple replicas of each $\{key, value\}$ pair are also created and stored in *replica*, or *surrogate* nodes. So, if a replica node is found prior to reach the original responsible node, this node will provide the location of the desired resource. This mechanism not only increases flexibility and saves bandwidth, but also alleviates the problem of a single point of failure.

Regarding bootstrapping phase, a newcomer N also contacts with a bootstrapping node B, which will start routing the "join" message towards the nodeID of N. Then, N informs the relevant nodes of its presence to update their neighbor maps. Once all potential neighbors are located, the relevant $\{key, value\}$ pairs are copied to N.

NodeIDs and *keys* may be distributed in the virtual space randomly using a hash function, but authors do not state which information should be used. The same happens with the user access control system; authors do not define any mechanism to carry on this task.

2.1.2.5 Kademlia

Kademlia [85] is similar to the other P2P overlays in the sense that $\{key, value\}$ pairs are stored in nodes with nodeIDs "close" to the *keys*. However, this closeness relies on a different notion of distance. Authors define the distance between two points in the key space as their bitwise exclusive-or (XOR) interpreted as an integer. NodeIDs and *keys* (generated using a hash function) have a length of $B = 160$ bits and *values* are pointers to the files. The $\{key, value\}$ pairs are replicated in several nodes.

Kademlia treats nodes as leaves in a binary tree, in which the position of each node is determined by the shortest unique prefix of its nodeID. For any given node, the binary tree is divided into a series of successively lower subtrees that do not contain the node itself. To organize this knowledge, nodes use *K-buckets*, which represent subtrees with a group of a maximum of K contacts, and divide them into more subtrees in the case that the *buckets* already contain K contacts. The *K-buckets* are organized by the distance between the node and its contacts. Specifically, for *bucket* j, where $0 <= j < B$, it is guaranteed that

$$2^j <= distance(node, contact) < 2^{j+1}.$$

This means that the *bucket* zero has only one possible member, the *key* which differs from the nodeID only in the low order bit. And on the other hand, if nodeIDs are uniformly distributed, it is very likely that half of all nodes will lie in the range of *bucket* $B - 1 = 159$.

An example of this distribution can be seen in Figure 2.3. This example shows the three smallest subtrees (*buckets*) constructed by the node with nodeID 111...110 (red point). These *buckets* have the prefixes 111...111, 111...10 and 111...0 respectively. The Kademlia protocol establishes that every node should know at least one node in each of these subtrees to locate any other node in a prefix basis. Otherwise, its location would not be guaranteed.

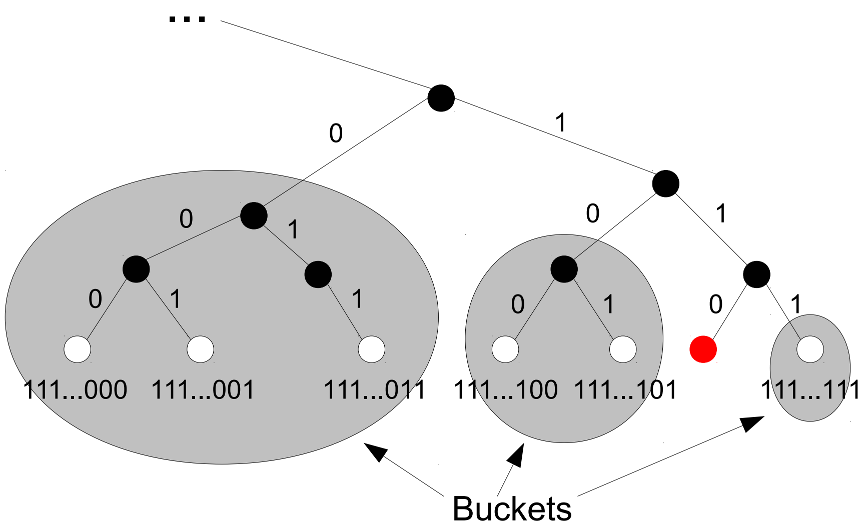


Figure 2.3: Generation of *buckets* in the node 111...110.

In Kademlia, every message transmitted by a node includes its nodeID, permitting the receiver to record the sender's existence if necessary. In this way, this topology has the property that every message exchanged conveys useful contact information. Using this information, a node can send parallel asynchronous query messages which tolerate node failures without imposing timeout delays on users. A nodeID-based routing algorithm lets anyone locate servers near a destination *key*.

Regarding the bootstrapping phase, first of all, a newcomer N must calculate its own nodeID in the 160-bit key space, generated by hashing its IP address as in Chord, and insert the nodeID of the bootstrapping node B into the appropriate *K-bucket*. Then, N must perform a node lookup for its own nodeID and refresh all *K-buckets* further away than its closest neighbor. During this last process, N populates its own *K-buckets* and inserts itself into other nodes' *K-buckets* as necessary. Unfortunately, authors neither include any mechanism for controlling the user access.

Next, we describe a particular case of Kademlia implementation, the Kad network, since we propose a new distribution system of revocation data for this P2P overlay in the Chapter 7.

The Kad Network. Kad is one of the widest deployed P2P overlays in the Internet with an estimated number of concurrent online users of around 4 million. It is based on the Kademlia DHT routing protocol and is implemented by eMule [7] and aMule [1] file sharing applications, both open source. Each node in Kad has a 128 bits `KADID` which defines its position in the virtual space, and the distance between any two points (nodes or contents) is defined as their bitwise exclusive-or (XOR) interpreted as an integer. Kad treats nodes as leaves in a binary tree, in which their position is determined by the shortest unique prefix of the `KADIDs`. Nodes arrange their tree contacts in buckets and store these lists as routing tables. Each node registers a maximum of K contacts per level i, contacts that are at a distance of between 2^{128-i} and 2^{127-i} from the `KADID` of the node regarding the XOR metric. Routing to a `KADID` is done in an iterative way, messages are forwarded to the n closest contacts to the target `KADID` and each node on the paths return the next hop. The closer the contact is to the node, the better it knows this part of the DHT. This mechanism provides routing in $O(log\ n)$.

As in many other P2P overlays, the purpose of the Kad DHT is to bind values and keys, more specifically files and keywords. To share a file, raw data and keywords must be hashed separately using the MD5 function and then published in a distributed way several times (at least 10 times). Kad only publishes metadata and sources. These references are stored in nodes which are close enough in the virtual space to `keywordIDs` and `sourceIDs`, respectively. This distance is called the *tolerance zone* of a `KADID`, and it is calculated using the 8 most significant bits of the identifier. Moreover, to improve the availability, resources are periodically republished: `sourceIDs` every 5 hours and `keywordIDs` every 24 hours. Analogously, a node, on which a `sourceID` or `keywordID` was published, will delete this information after 5 and 24 hours respectively, if it is active. Republishing is done exactly in the same way as publishing.

2.1.2.6 BitTorrent

BitTorrent [44] is a second generation P2P protocol for distributing files, where resource lookup is performed on a *web server* and resource dissemination is managed by a *tracker*. Trackers neither store resources nor participate in their exchange, they only coordinate the set of *peers* that participate in the resource exchange (*swarm*) and keep track of the active peers of the set. Web servers index metadata files (*.torrent*) which describe the resource exchanged by a swarm. This includes information such as the address of its tracker and the names, sizes and checksums of all *chunks* in which the resource is split.

In the traditional BitTorrent network, whenever a user wants to download a resource, she contacts to a tracker to obtain the list of peers that make up the swarm. And then, she contacts to some peers of the list to download all chunks that make up the resource using the peer wire protocol, implemented over TCP. A peer communicates with the tracker regularly while it is part of the swarm to inform about the volume of bytes she has downloaded or uploaded, and the tracker responds with the list of active peers at each time.

Additionally, few years ago, Loewenstern and Norberg proposed to introduce a decentralized tracking system [80] to avoid single points of failure and improve the performance of the tracking process, where any peer can act as a tracker (trackerless torrent). This new tracking system was implemented using a DHT based on Kademlia (Mainline DHT, MDHT) to store and locate information about which peers hold what resources. Other DHT, also based on Kademlia, was also proposed but it is only implemented by one client software (Vuze [19], previously Azureus) and is not compatible with the MDHT, as they have non-trivial differences. For these reasons we only take into account the solution that uses the MDHT.

In the past, a BitTorrent client only included a peer, instance of the program to which other clients connect and transfer data. But now, most clients also include a *MDHT node*, another instance used to find peers in a decentralized way.

As Kademlia does, MDHT stores nodeIDs and *keys* (*infohashes* from here on) as leaves in a binary tree. Whenever a user wants to download a resource in DHT BitTorrent, firstly she must decide which nodes to contact to get the peers list to download from using her peer instance. For that, the node instance uses the XOR distance metric between the *infohash* of the resource and the nodeIDs of the nodes in her own routing table. Note that a node knows many nodes with nodeIDs "close" to its own, but it has few contacts with nodeIDs that are "far". Then, the original node contacts some nodes with nodeIDs closest to the *infohash*[1] and obtains information about peers currently uploading that resource. Obviously, if some contacted node does not know about peers for that resource it must respond with information of the nodes in its routing table that are closest to that *infohash*. This process finishes when the original node cannot find any closer nodes. Finally, the node stores in appropriate buckets the peers' information for a small number of the responding nodes with nodeIDs closest to the *infohash* of the resource. The protocol which allows peers within the same swarm to share their peer lists is the peer exchange (PEX) protocol, implemented over UDP.

Regarding the nodeID assignment and the user access control, unfortunately, at first, little attention was paid to security. In the BitTorrent context, nodeIDs were generated at random from the same 160-bit virtual space as *infohashes* and no access control mechanism was proposed. Today some security extensions related with the nodeID generation have been proposed, such as [92], which is discussed in Section 3.1.

The Bootstrapping phase is similar to that performed in Kademlia, with the difference that the nodeIDs are randomly selected. In addition, unlike Kademlia, the MDHT only stores each *infohash* on one node, resulting in no easy way to unambiguously determine which peers are responsible for a certain resource, complicating any replication or migration strategy.

2.1.2.7 JXTA

Juxtapose [66] is a set of open protocols that enable the creation and deployment of P2P networks. JXTA protocols enable users to discover and observe other nodes, to commu-

[1]An *infohash* is calculated as the hash of the "info" section of the original *.torrent* file.

nicate among them, or to offer and localize resources within the network. In order to access those resources, JXTA completely relies in the usage of advertisements published by the resource owner. JXTA-Overlay [123] extends such protocols in a framework which increases the reliability of JXTA-based applications and supports group management and file sharing. This framework differs from above P2P protocols because it introduces the concept of peer group, one of its main features. The overlay network is divided into hierarchical groups of nodes, which offer a context for accessing services. Users are organized into different overlapping groups, so only members of the same group may interact between them. Peers must join the group that offers the services in which they are interested.

Brokers are special nodes which control access to the network, taking care of user authentication as well as helping client nodes interact between them by propagating their related information. Brokers are very important since they exchange information about all client nodes, maintaining a global index of available resources, which allows all nodes to find network services. Brokers also act as beacons used by client nodes which have recently gone on-line to join the network.

Unfortunately, the design focus on JXTA-Overlay was completely concerned with system performance, with the only exception of the user authentication. Users are authenticated using pairs username and password before they join the network, which are issued without any control. As a result, JXTA is vulnerable to Sybil attack and other security threats which may jeopardize the network.

2.1.3 Identity Problems in P2P Overlays

Uncontrolled assignment of nodes' identities within a P2P overlay (nodeIDs) can give rise to a series of problems (the presence of whitewashers, for instance) and vulnerabilities (the Sybil attack, the Eclipse attack, the Man-In-The-Middle (MITM) attack, etc.). But the identity problem is not only important due to security considerations, but also due to performance ones, as the efficiency of P2P overlay routing protocols is based on the uniform distribution of nodeIDs within the virtual space. The load balancing is very important, and if the nodeID density is higher in some particular zones of the virtual space than in others, the network performance could be globally degraded [72]. In this Section we define some of the most important identity-related problems that arise in P2P overlays.

2.1.3.1 The Sybil Attack

The management of multiple nodeIDs (Sybils) by the same (malicious) node simultaneously is known as Sybil[1] attack [50]. Carrying out this attack, a malicious user increases her presence within a P2P overlay by artificially simulating the existence of several different nodes. Thus, the attacker can manage a group of colluding virtual nodes, which could damage the proper operation of the overlay. For instance, an attacker performing the Sybil attack can improve its own reputation by using good feedback that comes from fake nodes (Sybils). According to Douceur [50], it is impossible to discern whether the same entity controls several nodeIDs or not, even asking other nodes; and the only way to avoid this attack is to use a TTP that certifies nodeIDs.

2.1.3.2 The Eclipse Attack

The Eclipse attack [111] is a way of routing poisoning which aims to separate a part of the P2P overlay from the rest. The attacker tries to intercept all the messages directed to, or sent by, a group of nodes (or a specific node) by means of a set of nodes with nodeIDs numerically close to the nodeID of the target group. To achieve this purpose, this colluding group of nodes manipulates the entries of the routing table of the nodes of the group to monitor, or filter, all the messages exchanged by them. Therefore, all messages will be routed across the attacker nodes and the correct nodes will be "eclipsed" by them. Note that the routing tables are constructed using routing information exchanged between nodes, but if we are not able to assign stable identifiers to nodes, it is difficult to tell the difference between honest from malicious routing information. Some studies [105] conclude that controlling a 10% of nodeIDs it is possible to control 65% of all routing paths of a P2P overlay.

2.1.3.3 The Man-In-The-Middle (MITM) Attack

As its name implies, in this attack, the attacker locates herself undetected between two target nodes with the purpose of spying on their communications or even manipulating them. Usually, in P2P overlays, the goal of these attackers is to spoof nodeIDs and/or dispatch false information. A possible attack could be spreading polluted content on

[1]This name comes from the subject of the book *Sybil*, a case study of a woman with multiple personality disorder.

behalf of a trusted entity. Therefore, if we take into account the routing properties of these networks and allow that nodeIDs can be selected/manipulated by the users without any control, there is no doubt that these networks will be extremely vulnerable to the MITM attacks.

2.1.3.4 Other Threats

The efficiency of P2P overlay routing protocols is based on the uniform distribution of nodeIDs within the virtual space. Therefore, the overlay performance can be globally degraded if most nodeIDs belong to a particular area of that virtual space. Unfortunately, if nodeIDs can be selected by the users, nobody is able to ensure that these identifiers are uniformly distributed, which should be considered as a serious threat.

Other security threat related to the identity of the nodes is the presence of white-washers[1] within a P2P overlay. Reputation systems can be used to prevent malicious behaviors and to promote honest collaboration among nodes. However, the effectiveness of these systems just depends on the stability of the nodeIDs. If a node can leave the network and rejoin it with a new nodeID, its accumulated reputation (good or bad) will be removed, which makes reputation systems useless.

The uncontrolled assignment of nodeIDs can provide another problem if resources are not stored by their owners. If a malicious user can choose a certain nodeID to become directly responsible for a target $\{key, value\}$ pair, she will be able to censor or corrupt that resource.

2.1.4 Distribution of Revocation Data

One of the most common digital forms of identification is the use of digital certificates, which essentially bind an entity with a public key. This binding is certified (signed) by a CA. Thanks to the use of digital certificates we can provide support for public key cryptography in P2P overlays, as well as a single authentication method, among other services. The safe use of digital certificates relies on the possibility to revoke them in certain situations, for example, if a private key is compromised.

A PKI takes care of issuing and making accessible digital certificates as well as of revoking them. It also verifies the authenticity and validity of certificates, and if they

[1]Nodes that purposefully leave and rejoin the network with a new nodeID in an attempt to shed any bad reputation they have accumulated under their previous nodeIDs [84].

are trustworthy. All these services are essential for the proper functioning of digital certificates, which are based on a distributed model by using CAs.

In P2P overlays, deploying a PKI would allow nodes to communicate and share information securely. Thus, users would be able to perform transactions, like electronic payments, or even to sign digital contracts without risking their confidential data. But validation process of certificates implies a considerable cost to both users and networks. If a user wants to access a confidential resource, the provider needs not only to find a certificate chain from the provider to the user, but also to check that all certificates in the trust chain are valid. And for that, she needs to check if certificates are revoked.

So far, several revocation systems have been proposed in the literature. Next, we briefly describe some approaches and standards related to the certificate revocation.

2.1.4.1 Revocation Approaches and Standards

The Internet X.509 Public Key Infrastructure (PKI) Certificate Management Protocol (CMP) [22] defines protocol messages for certificate creation and management. In particular, CMP defines a revocation request format and a protocol for publishing the status of the certificates.

The Certificate Revocation List (CRL) [45] is the most common mechanism to manage revocation data. A CA periodically issues a digitally signed list with all certificates that have been revoked. Then users download this list to check if a certain certificate is revoked or not. Moreover, these lists can also be stored in several servers to improve its availability.

Other improvements are, Overissued CRL (O-CRL) [46], which tries to reduce the peak request rate of CRLs in servers by the issuance of CRLs with overlapping validity periods. Indirect CRL (I–CRL), which stores revocation data issued by several CAs in a single CRL. Delta CRL (Δ-CRL), which tries to reduce the consumption of bandwidth by the issuance of small CRLs that only contain the certificates whose status has changed since the last entire CRL was issued. CRL Distribution Point (CRL–DP), which is an extension included in the standard X.509 version 3, where CRLs stored in these DPs only contain the status information of a certain subgroup of certificates. In this case, each certificate stores the address of its CRL-DP, and the criteria for creating these subgroups can be geographic, scope of use, etc. Finally, another way to improve the performance of the traditional list-based method of distributing revocation data is to segment the CRLs. This strategy reduces the size of each CRL, and therefore it allows repositories to serve CRL requests at a faster rate. Moreover, revoked certificates may be stored in the CRL segments in a way that attempts to minimize the number of segments that a verifier will need to download. Unfortunately, segmenting CRLs does not reduce the peak request rate for them.

A standard alternative to CRLs, and proposed by the PKIX workgroup of the IETF, is the Online Certificate Status Protocol (OCSP) [108]. OCSP enables users to determine the status of a specific certificate sending a certificate status request to an online server able to verify the revocation status of certificates. Then, this server responds with a signed message indicating whether the specified certificate is currently revoked. Once the OCSP client has sent the request, it suspends the acceptance of the certificate until the CA provides a response. An OCSP response can contain four times:

- *thisUpdate:* The most recent time at which the status being indicated is known by the CA to have been correct.
- *nextUpdate:* The time at or before which newer information will be available about the status of the certificate.
- *producedAt:* The time at which the CA signed this response.
- *revocationTime:* The time at which the certificate was revoked or placed on hold.

In [74], an OCSP improvement was suggested by Kocher, the Certificate Revocation Tree (CRT). This proposal is based on the use of a hash tree to replicate signed revocation data in many validation servers to handle the load of all CAs. The leaves of the hash tree are the currently revoked certificates sorted by serial number from left to right, and the root of the hash tree is signed by the CA to ensure authenticity and integrity of the whole tree. In this way, a user can validate a certificate sending a request to the closest validation server, since any of these servers can send a convincing proof that the certificate in question is (or is not) on the CRT.

However, the CRT has a significant problem, it must be reconstructed and redistributed every time a new certificate is revoked. For this reason, many authors have proposed the use of data structures which allow dynamic updates, such as skip-lists [87] or 2-3 trees [91].

2.1.4.2 Distribution of Revocation Data in P2P Overlays

Each of the above systems, or mechanisms, have different features in terms of network traffic overhead, load on the servers which provide revocation information, freshness of this information and suitability for offline usage. In addition, most of these systems are typically client/server structures, where a CA plays the role of a central server, and suffer from the common problems of a single-point of failure. Therefore, not all are equally suitable to run on P2P overlays.

In the P2P overlay context, on the one hand, client/server structures are not suitable due to their centralized nature, and on the other hand, if only one or few CAs are capable of distributing the revocation data to all nodes, they can become overloaded due to the large number of users who usually use these networks. Moreover, if we take into account that each user can obtain several identifiers depending on the method used to bind those nodeIDs to users, the problem is accentuated since the CRL size[1] grows with the number of network users and certificates per user. Thus, large networks where users have a set of certificates bear high certificate revocation rates.

Therefore, in these networks, revocation data must be efficiently distributed and frequently updated, otherwise, the checked data by the users will not be fresh. Unfortunately, the updating policies of the standard mechanisms are not enough to ensure the freshness of the revocation data in these environments, and to replicate the CRLs in several servers it is not enough to improve its availability.

For these reasons, we find in the literature many different proposals to distribute revocation data more efficiently in these networks, and some of them are described in the Section 2.1.5.4. In Chapter 7 we describe our own proposal, a new distributed revocation system which stores revocation data in several CRL segments and distributes them using the Kad network itself.

2.1.5 Related Work

In this Section we discuss some research works that attempt to tackle some of the previously mentioned attacks/threats, which are classified in three categories: centralized proposals, where the identity management is performed by one or several fixed entities;

[1]Note that in this thesis we talk about size of an entire CRL, or segment, to refer to the number of certificates that are stored in that list.

distributed proposals, where a group of nodes are responsible for controlling and issuing the nodes' identities; and social network-based proposals, where the social relationships are used to detect and limit network access by malicious users. Finally, we describe some systems that try to improve the distribution of revocation data in P2P networks.

2.1.5.1 Centralized Proposals

Douceur, in [50], was the first to deal with the Sybil attack in P2P overlays, and he comments the impossibility to know if two nodes are managed by two real users, or if there is only one user managing them, even asking other nodes within the network. In this way, the author concludes that a trusted entity that certifies nodeIDs is the only solution to completely avoid the Sybil attack in these networks. However, he also suggests the use of methods for adding extra computational cost to the process of obtaining nodeIDs and certain system conditions to mitigate this attack. Following this line many approaches have been proposed until now.

In [36], Castro et al. propose two centralized ways to generate node identifiers. The first one is to delegate the identity assignment problem to a set of trusted CAs, which signs certificates that bind a random nodeID to a public key and an IP address. They assume that every node in the overlay has a static IP address. Therefore, when a user changes her IP address, the old certificate and the corresponding nodeID are no longer valid. Taking into account that in the current Internet scenario devices often change their IP addresses, distributing revocation data in large P2P overlays can be a problem. Moreover, authors allow multiple certificates per IP address, which is an important drawback from the point of view of the Sybil attack. The second proposal is to charge money for certificates or bind nodeIDs to real-world identities to mitigate the Sybil attack. However, both solutions may not be liked by users. On the one hand, charging money for certificates makes the system not suitable for free services, and on the other hand, using real-world identities, users lose their anonymity in front of the CA.

Srivatsa and Liu propose the use of certificates with a short life-time issued by a bootstrap server which also generates random nodeIDs [114]. This technique limits the number of nodeIDs that an attacker can obtain during a time period, depending on the life-time of the certificates, and maintains complete anonymity of the users. However, the life-time can affect the security of the system. If it is too short, the server can become a bottleneck since the update process of certificates may introduce a significant computational overhead. On the other hand, longer life-times can cause greater exposure to compromise. Therefore, it is very important to set the system parameters taking into account this trade-off between security and performance.

In [34], Butler et al. consider the use of the identity-based encryption (IBE) to improve the security in P2P overlays, where users' public keys are directly derived from nodeIDs. In the centralized proposal, nodeIDs are randomly generated by a TTP, as well as the private keys. Then, a bootstrapping node verifies nodeIDs and provides users with a token of authenticity. Unfortunately, the node authentication is performed via callback using the user's IP address. This means that any node capable of receiving a TCP connection at an IP address is deemed to be the legitimate owner of that IP address; which is the main drawback of this scheme.

Baumgart and Mies propose to use a hash function over a public key to generate nodeIDs in the Kad network [29]. Those public keys must be additionally signed by a trustworthy CA. Thus, this signature prevents the Sybil attack in the bootstrapping phase. In the absence of this authority, they propose to use a cryptographic puzzle to limit Sybil and Eclipse attacks.

In [54], Friedman and Resnick propose a centralized system of anonymous certificates using standard encryption techniques. Users receive a signed nodeID which is unrelated to their real-world identity. In addition, the certificate provider guarantees that each user will only obtain a single certificate (once-in-a-lifetime identifier). The trusted entity signs nodeIDs which have previously been blinded, so it knows the users' real-world identities but not their nodeIDs within the overlay. This system allows using a reliable reputation system but it does not allow that certificates can be revoked.

In [23, 24, 25, 53], Aiello et al. present Likir (Layered Identity-based Kademlia-like InfRastructure), an architectural model of a new DHT system that offers both a very high protection level to the most common attacks in P2P overlays and a simple framework supporting identity-based services. Authors divide the architecture in three main modules, one of them a "User Registration Service". In this module, Aiello et al. propose, on the one hand, to involve the human interaction with a centralized server in the authentication phase using the OpenID protocol, and on the other hand, to use a trusted entity to bind the user real-world identity to the user's public key and to a 160-bit random string (nodeID) by a LikirID. As this system requires human interaction, the automatic nodeID generation is unfeasible for an attacker. In addition, Likir guarantees the users' anonymity since users do not need to reveal their real-world identities. However, the use of the OpenID protocol has several problems [48]. As this work is similar to ours, in Section 6.5 we compare the user registration module of Likir with the features provided by our protocol RIAPPA.

2.1.5.2 Distributed Proposals

In [36], Castro et al. propose the use of a cryptographic puzzle to limit the number of nodeIDs managed by a user in a P2P overlay. Users must choose a key pair so that the hash of the public key has the first p bits to zero, which will be their nodeID. In this way, the value of p defines the security level of the system since the expected number of operations needed to compute a key pair is 2^p. Then, authors also propose to bind those nodeIDs to users' IP addresses and define another computational challenge. Now, users have to find a string x such that $H(H(IPaddress, x), nodeID)$ has p' bits equal to zero, and present the string x to be accepted by the other nodes. Finally, Castro et al. also comment that updating the p and p' values periodically the security of the network would be improved. In all cases, authors limit the number of computable nodeIDs by users. However, the computational cost normally is not a problem for attackers since they usually have enough computing power; not so the normal users, which may be using mobile devices such as smartphones, tablets, or netbooks. Furthermore, IP addresses are vulnerable to IP spoofing attacks and on many times dynamically assigned.

In the same line than Castro et al., a cryptographic puzzle mechanism has also been proposed by Rowaihy et al. to limit Sybil attacks [105]. Authors present an admission control system using a self-organized hierarchy of cooperative nodes and a chain of cryptographic puzzles. They exploit a hierarchical structure to distribute load and increase resilience to targeted attacks, and to refresh the challenges constantly to avoid pre-computation. When a node wishes to join the network, it must contact a leaf node. Then, this node sends a cryptographic puzzle based on a hash function. Once the joining node has solved the puzzle, it is redirected to the leaf's parent. This challenge is recursively repeated until it reaches the root node. Finally, the root node issues a special token and a nodeID to the node. This nodeID is just the hash function over the node public key, previously selected by the user, and a random number generated by the root node. As with the above solution, this mechanism also negatively affects to the nodes that have limited resources, and it does not solve the problem because malicious hosts with enough resources can manage a large number of nodeIDs. The effectiveness of this solution depends on the cost and the degree of hardness of solving the puzzles. Moreover, if an attacker is a member of the hierarchy, she can take advantage of her position, since she will need a smaller number of puzzles to obtain a nodeID.

In [47], Da Costa et al. try to minimize computing problems that affect honest nodes when they have to solve cryptographic puzzles to obtain their nodeIDs. Authors propose the use of adaptive computational puzzles to limit the spread of Sybils but without affecting the honest nodes. This proposal parameterizes the complexity of puzzles according to the nodes behavior. Users of the nodes whose behavior is more similar to the average behavior of the rest of the network are benefited with less complex puzzles. Otherwise, users are forced to solve more complex puzzles to obtain nodeIDs. However, detecting Sybil nodes in function of their behavior is usually very inefficient, since at first glance they behave correctly. To do so, they need an efficient reputation system to avoid malicious users have computational privileges, which implies to use stable nodeIDs within the network.

Lu proposes, in [81], a conundrum verification scheme which allows access to the P2P overlay through a more expensive process of identity acquisition. It works over a structured network with hierarchy; super nodes manage regions which include a lot of guard nodes and normal nodes. The solution is composed of two phases, the first where nodes join the network paying a certain price (e.g., solving a cryptographic puzzle) and the second where the super nodes use the guard nodes to obtain the nodeIDs of the normal nodes and to verify the validity of those nodeIDs. This verification is performed based on the statistics result. The main weakness of this solution is in the binding of nodeIDs, since they are selected by the users using their IP addresses to allow the verification.

In [34], Butler et al. also develop two decentralized identity assignment protocols. In the first one, the bootstrapping node plays the role of the TTP and generates the random nodeIDs, the users' private keys and the tokens of authenticity. In the second one, the authors propose a method that uses both ID-based and symmetric cryptography. The bootstrapping node only generates the private keys and delegates the authority of assigning nodeIDs to one of many trusted nodes, which share a symmetric key. These nodes also generate the tokens of authenticity. But in the same way as in the centralized proposal, in these two protocols, nodes are weakly authenticated via callback using their IP addresses, which is insufficient to avoid the Sybil attack.

2.1.5.3 Social Network-Based Proposals

Wang et al. propose, in [122], a practical system for detecting Sybil identities using server-side clickstream models. Their approach groups "similar" user clickstreams[1] into behavioral clusters. Authors base their work on two main features. On the one hand, on the fact that Sybils and real users have very different goals in their usage of online services. Real users likely take part of numerous features in the system and Sybils focus on specific actions (i.e. acquiring friends and disseminating spam) while trying to maximize utility per time spent. And on the other hand, on the hypothesis that these differences will manifest as significantly different (and distinctive) patterns in clickstreams. Finally, they test their prototype on Renren and LinkedIn server-side data achieving positive results.

Leveraging the real-world trust relationships between users, many authors have developed social-graph-based algorithms to detect Sybil nodes on social graphs [35, 118, 127, 128]. These solutions are mostly built on the assumption that the social network graph can be partitioned into two loosely linked regions, a non-Sybil region and a Sybil region. Although this assumption may hold in certain settings, real-world social connections possibly tend to divide users into multiple inter-connected small regions instead of a single uniformly connected large region. Given this fact, the applicability of existing schemes would be greatly undermined for inability to distinguish Sybil users from valid ones in the small non-Sybil regions.

In [128], Yu et al. present *SybilGuard*, a social networks based protocol which limits Sybil attacks. NodeIDs are represented as nodes in a graph and an edge between two nodeIDs indicates a human-established trust relationship. SybilGuard exploits the fact that all the malicious nodes created by a given physical attacker are only sparsely connected to the real social network. The edges connecting the honest and the Sybil regions are called attack edges, and its number is independent of the number of Sybil nodes. Each node constructs its own partition of the network using a procedure based on random routes for each of its edges, a special kind of random walk. When a node wants

[1] Clickstreams are traces of click-through events generated by online users during each web browsing "session".

to verify that another node is honest, it checks for intersections in their routes. A node accepts a suspect node only if at least half of its routes intersect with any of suspect's routes, meaning that most likely that node will belong to the same (honest) region as it. SybilGuard allows accepting $O(\sqrt{n} \log n)$ Sybil nodes assuming up to $O(\sqrt{n}/\log n)$ attack edges, where n is the number of honest nodes.

In [127], Yu et al. propose an update to SybilGuard, called *SybilLimit*. This proposal allows accepting $O(\log n)$ Sybil nodes assuming up to $O(n/\log n)$ attack edges. Unfortunately, in terms of functionality, these solutions are not suitable for P2P overlays since there are several difficulties that complicate their real-world deployment. Firstly, they require previous trustworthy relationships among nodes. Secondly, they require symmetric key sharing before the creation of links between nodes. And finally, they force all nodes to store a different symmetric key per friend since each social connection has a unique symmetric key.

In [118], Tran et al. propose Gatekeeper, an optimal distributed Sybil-resilient admission control protocol that significantly improves SybilLimit. For the case of $O(1)$ attack edges, it admits only $O(1)$ Sybil nodeIDs in a random expander social networks. In the face of $O(n/\log n)$ attack edges, the protocol admits $O(\log n/\log n)$ Sybils per attack edge.

In [124], Xue et al. extend the social graph by including user interactions of initiating and accepting links. They propose to use the friend request data as a directed graph, with an edge between the sender and the receiver and a weight (1/0) that indicates whether the invitation is accepted. This new graph model utilizes two parameters to improve the detection phase: on the one hand, the number of requests to become a friend, which are rarely sent to Sybils and non-popular users, and, on the other hand, the information of the accepting/rejecting friend request. Sybils and non-popular users send friend requests to gain friends, however, Sybils' requests are more likely to be rejected. Authors present VoteTrust, a global voting-based system that nicely combine link structure and users feedback (accept or reject friend requests) to detect Sybils.

In [110], Shi et al. present SybilShield, a protocol that utilizes a multi-community social network structure in the real-world to defend against Sybil attacks. This scheme leverages the sociological property that the number of cutting edges between a non-Sybil community and a Sybil community, which represent human-established trust relationships, is much smaller than that among non-Sybil communities. Moreover, authors use agent nodes to greatly reduce the false positive rate of non-Sybils among multiple communities, while effectively identifying Sybil nodes.

The previous systems do not avoid Sybil attacks, since they do not assign nodeIDs in a controlled manner, but they can detect possible Sybil nodes with a certain probability. A drawback of these schemes is that the use of social networks to detect Sybils compromises the users' anonymity, since all nodeIDs must be related with a social network account.

Lesueur et al. present, in [77], a sybilproof distributed identity management system based on invitations. Thus, they rely on social relationships to prevent Sybil attacks. Their scheme is based on a balanced tree that represents the social relationships between users. However, they limit the number of invitations for each member and each of the members he has invited comparing the members between themselves. In this case, Sybil attacks are limited but having to restrict the number of potential newcomers by the use of invitations.

2.1.5.4 Distribution Systems of Revocation Data

Ying and Jiang, in [126], propose a new certificate revocation system based on the use of the Chord network [116] and bloom filters to distribute the revocation data. They use bloom filters to avoid bottleneck problems in the network. Their method consists in flooding all nodes with a bloom filter vector for each CRL segment. In this way, they save storage and bandwidth capacity. However, the false positives that occur in bloom filters complicate the certificate validation process. Furthermore, using flooding mechanisms may be inefficient in P2P networks with hundreds of thousands of users.

In [63], authors propose to introduce a hierarchy based on *Super Peers* in the P2P architecture to reduce the number of requests to the CA. Super peers act as authorities to clients, and the CRL distribution is carried out through a pull mechanism. However, this hierarchy does not prevent storage problems in peers nor the great bandwidth consumption if the CRLs are very large.

Morogan and Mutfic describe, in [90], a general distributed system for revocation data based on the use of a P2P network to distribute CRLs. Authors aim to achieve a good off-line functionality and to improve the CRLs' freshness using Delta CRLs. However, their system is not completely distributed, which implies that in networks with many revocations, updates have considerable size. In addition, the distribution problems that arise when a CA issues a new CRL are not avoided by this system.

Authors of [28] propose a new distributed trust infrastructure based on the use of the Chord network, which includes a new distributed revocation mechanism. They use nodes themselves to store the revocation information rather than using CRLs. In this system, each node is responsible for storing a set of certificates, and if a certificate is revoked, then the node stores a tag that indicates the revocation status.

On the other hand, *trust* is another approach for managing access control in P2P networks. In these networks, trust captures the liability, trustworthiness, and satisfaction of a node relative to another one. The main goals of trust models are to increment the percentage of successful exchanges and to ensure models' scalability and simplicity. For instance, the BBK model [31] describes a quantified trust, which divides trust into two types: direct trust and reference trust. Unfortunately, it gives the same weight to both types of information, therefore, malicious references can significantly alter the trust between users. Authors in [40] present a distributed trust model for the JXTA platform [66]. In this model, trust is computed based on users' interests and keywords. Nevertheless, a configuration table is needed for the whole group, which may be inefficient in a dense P2P network. Authors in [60] describe different types of trust and define the relation between trust values and roles in an access control model. However, this model lacks to determine how the role based on access control model works well in a trusty network.

2.2 Cryptography

2.2.1 Elliptic Curve Cryptography (ECC)

Victor S. Miller and Neal Koblitz introduced the Elliptic Curve Cryptography (ECC) in the 80s but it was not until the late 90s when it began its use [73, 88]. ECC is based on the use of elliptic curves over finite fields. The equation that defines the elliptic curve over $E(\mathbb{F}_p)$ is the following:

$$E: \ y^2 \equiv x^3 + a \cdot x + b \ (mod \ p)$$

Where p is the order of a field $\mathbb{F}_p$ and $a, b \in \mathbb{F}_p$ are two elements of this finite field. If the previous parameters follow certain relationships the previous equation describes a set of points in $\mathbb{F}_p$ with coordinates (x, y). In addition, a sum operation over elliptic curve is defined and also a special point $\mathcal{O}$ is defined as the identity element.

Then, a point G of the elliptic curve is defined as a base point and the sum operation of the elliptic curve is used to generate a set of elements (points). The order of a generator n is the number such that $nG = \mathcal{O}$ and $G \neq \mathcal{O}$. Also, $G + \mathcal{O} = G$, which is said as that G generates a cyclic group over the elliptic curve.

As any public key system, elliptic curve cryptosystems are based on the intractability of certain mathematical problem. More specifically, ECC is based on the Elliptic Curve Discrete Logarithm Problem (ECDLP), which assumes that calculating the discrete logarithm of a random elliptic curve point with respect to a base point in an elliptic curve is infeasible. More accurately, this means that there is no probabilistic polynomial-time algorithm (polynomial in the security parameter $l = \lfloor log_2\, p \rfloor$) which with non-negligible probability:

On input: $Q \in_R \langle G \rangle$, $Q \neq \mathcal{O}$

Can output: $d \in [1,\ p-1]$ satisfying $Q = dG$.

Until now, no efficient algorithms are known to solve the ECDLP, and likely it is harder than both the integer factorization problem and the Discrete Logarithm Problem (DLP) modulo p [61].

Since the structure of the classical Discrete Logarithm Problem (DLP) and the Elliptic Curve Discrete Logarithm Problem (ECDLP) are similar, many algorithms based on DLP can be easily adapted to ECC. Among others, some remarkable examples are the Elliptic Curve Diffie–Hellman (ECDH) key agreement scheme (based on the DH scheme), the Elliptic Curve Digital Signature Algorithm (ECDSA) (based on DSA) and the ECMQV key agreement scheme (based on the MQV key agreement scheme).

In 2005, the National Security Agency (NSA) of the United States publicly announced “Suite B Cryptography”, which exclusively uses ECC for digital signatures and key exchange. These protocols were proposed to protect National Security Systems and National Security Information (CNSSP-15) [93].

ECC-based protocols are increasingly being used because their performance is better than classical cryptography. ECDLP based systems provide roughly 10 times greater efficiency than either integer factorization systems or discrete logarithm systems, in terms of computational overheads, key sizes and bandwidth consumption. Although elliptic curve arithmetic is slightly more complex per bit than RSA arithmetic [104], the added strength per bit makes up for any extra compute time. Table 2.1 shows a comparison between the ECC and RSA key size (recommended by the National Institute of Standards and Technology (NIST)) and the ECDSA [71] and RSA certificate[1] size.

Table 2.1: Key and certificate size comparison between ECC and RSA.

Security level (bits)	**Public key size (bits) ECC**	**RSA**	**Ratio ECC/RSA public keys**	**Certificate size (bits) ECDSA**	**RSA**	**Ratio ECDSA/RSA certificates**
80	192	1024	1:5	577	2048	1:3
112	224	2048	1:9	673	4096	1:6
128	256	3072	1:12	769	6144	1:8
192	384	7680	1:20	1153	15360	1:13
256	521	15360	1:29	1546	30720	1:19

Nowadays ECC is also widely used because it provides new tools for building cryptography such as bilinear pairings [82].

2.2.2 Implicit Certificates

A standard certificate like a X.509 certificate [45] can be viewed as a tuple $< Q_x, I_x, \sigma_x >$; where Q_x is the public key of the user "x" being certified, σ_x is the CA's signature of the certificate and I_x is identity information and other metadata such as serial number, validity period, issuer identity, user identity, etc.

On the other hand, implicit certificates [32, 86, 97] super impose the public key and the signature into a single parameter Z_x called "reconstruction public parameter". Thus, an implicit certificate can be viewed as the tuple $< Z_x, I_x >$.

If the metadata (I_x) is not sent explicitly with the certificate, the size of the certificate is just the size of the reconstruction parameter Z_x. Since the cryptographic portion of an implicit certificate is the size of an elliptic curve point, an implicit certificate is considerably smaller than a comparable explicit certificate. Smaller certificates are useful in highly constrained environments, such as Radio-frequency Identification RFID tags, where memory or bandwidth are scarce. In our case, we are not so interested in the size of the implicit certificate but rather in the form of construction of the implicit certificate. As we will show, this type of cryptographic construction will allow us to

[1]Data are only based on the size of signatures and public keys.

create efficient and secure ways of building identity assignment mechanisms for P2P networks.

In particular, in this thesis we use the Elliptic Curve Qu-Vanstone (ECQV) implicit certificate scheme [103] as the base to create a suitable type of implicit certificates for assigning identities to P2P networks (we will use a modified version of ECQV to create two new identity management systems for P2P networks in Chapters 4 and 6). ECQV is a general purpose implicit certification scheme proposed by the Standards for Efficient Cryptography Group (SECG). Figure 2.4 describes how a user x obtains an implicit certificate Z_x and derives her public and private keys.

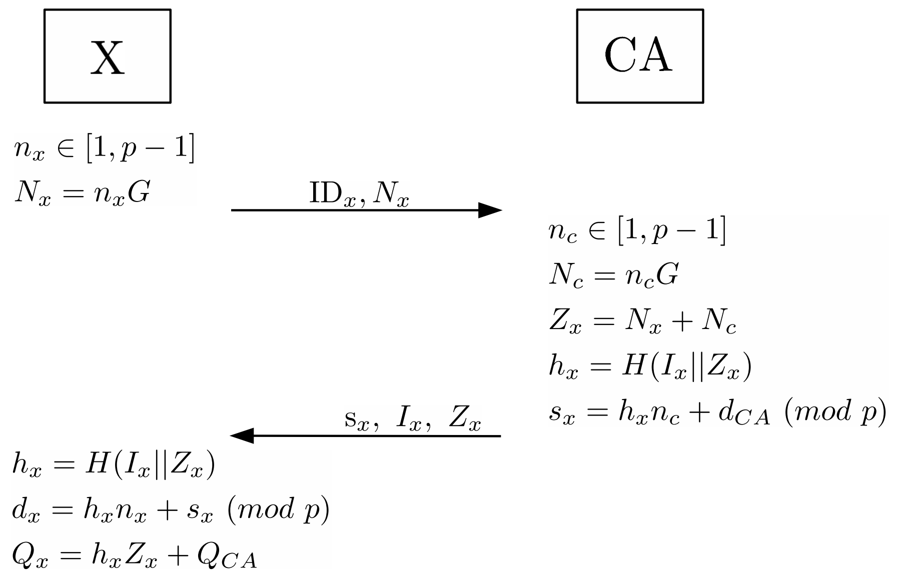


Figure 2.4: ECQV certificate issuance protocol.

User to CA

The user creates a random number n_x and with this number and the generator creates and a random elliptic curve point $N_x = n_x G$. Then, N_x together with her identity data (ID_x) is sent to the CA.

CA to User

The CA also creates a random number n_c and with this number and the generator creates a random point $N_c = n_c G$. Then, the CA generates the reconstruction public parameter for user x as $Z_x = N_x + N_c$. Next, the CA takes the identity information sent by the user (ID_x) and adds its metadata to obtain I_x. Then, the CA creates a signature so that the user can derive its secret key. The signature contains in a suitable way the metadata of the certificate, the reconstruction parameter, the random number selected by the CA and the CA's secret key: $s_x = h(I_x|Z_x)n_c + d_{CA}$.

Finally, the CA sends the tuple $< s_x, I_x, Z_x >$ to the user.

Key Derivations

In the last part of the ECQV scheme, the user derives her public and private keys. In addition, other users can also easily derive the public key of the user with the public parameters.

The private key is derived (only by the user) as: $d_x = h_x n_x + s_x$

The public key is derived (by anybody) as: $d_x G = h_x Z_x + Q_{CA}$

Note that with the public information provided by the CA only the user x can derive her secret key (not any other user or the CA).

On the other hand, any receiver of a certificate must validate two things: (i) the authenticity of the binding identity/public-key and, (ii) the authentication of the user (validate that the user is who claims to be).

With explicit certificates, (i) is validated using the certificate's signature and (ii) is validated by enforcing the user to demonstrate knowledge of her private key. For this purpose, some cryptographic protocol is followed, for example, verifying some type of digital signature performed by the user.

With implicit certificates, the validation of (i) and (ii) cannot be separated. These two things can be validated with a cryptographic protocol, for example, by verifying a digital signature performed by the user.

We end the discussion about implicit certificates showing some performance figures. Regarding the computational cost of each system, with implicit certificates based on ECC, the receiver needs to calculate the sender's public key to verify the validity of the certificate using one elliptic curve scalar multiplication operation and one point addition operation, whereas with ECDSA certificates she needs three modular operations, two elliptic curve scalar multiplication operations and one point addition operation to verify the CA's signature on the certificate. Therefore, three less modular operations and one less scalar multiplication are necessary if we use implicit certificates. Table 2.2 shows a comparison between the required operations to construct an ECC-based implicit public key and verify an ECDSA signature.

Table 2.2: Operations required for constructing an ECC-based implicit public key and verifying an ECDSA signature.

	Modular operation	**Scalar multiplication**	**One point addition**
Implicit PK generation	0	1	1
ECDSA signature verification	3	2	1

Table 2.3 shows a comparison between the ECDSA and ECQV certificate[1] size, where it is observed that implicit certificates are three times smaller than the explicit certificates independently of the security level.

Table 2.3: Certificate size comparison between ECDSA and ECQV.

Security level (bits)	**Certificate size (bits)**		**Ratio ECDSA/ECQV certificates**
	ECDSA	**ECQV**	
80	577	193	1:3
112	673	225	1:3
128	769	257	1:3
192	1153	385	1:3
256	1546	522	1:3

Finally, the most relevant aspect of ECQV for this thesis is that in the key pair generation, half of the material is generated by the user and half by the CA.

2.2.3 Blind Signatures

In 1982, Chaum introduced a new form of digital signature, the blind signature [39]; where a signer signs a message previously blinded by a requester. The signer does not know the content of the signed message, she only signs the message and does not decrypt it, and then the requester can derive a valid signature for that message from the signer. Once the message and its signature are published, the signer can verify the

[1]Data is only based on the size of signatures and public keys.

genuineness of the signature and anyone is able to verify the blind signature using the public key of the signer. However, no one can know about the correspondence of the message-signature pair, even the signer.

Any blind signature scheme consists of four phases:

1. **Blinding phase:** A requester selects a blinding factor to mess her message such that the signer will not know its content.

2. **Signing phase:** The signer signs the blinded message using her private key.

3. **Unblinding phase:** The requester uses the blinding factor to derive the digital signature on the message.

4. **Signature Verification phase:** Anyone can use the public key of the signer to verify the blind signature.

And they should satisfy the following properties:

1. **Correctness:** The correctness of a digital signature on a message should be verified by anyone using the public key of the signer.

2. **Unforgeability:** Only the signer should be able to generate a valid signature for a certain message.

3. **Blindness:** The signer of a blind signature scheme should not know the content of a blinded message. In fact, there should be only one valid blinding factor per message-signature pair.

4. **Untraceability:** The signer of a blind signature should not be able to link the message-signature pair under any circumstances.

These algorithms prevent fraudulent actions by the signer and provide authentication and non-repudiation to the original sign request sent from a requester. For these reasons, they are widely used in many important cryptographic schemes such as electronic voting and untraceable payment protocols.

The first blind signature protocol proposed by Chaum [39] is based on RSA cryptosystem (factorization problem) but today many other protocols have already been proposed to satisfy the four properties mentioned above. They can be based on the factorization problem, DLP, ECDLP [69] or the quadratic residues, among others.

In Chapter 6 of this thesis we have adapted a blind signature protocol based on ECDLP [112] to preserve the users' anonymity in a new identity assignment protocol for a P2P overlay.

2.2.4 Commitment Schemes

A commitment scheme allows one to commit to a chosen value while keeping it hidden to others, with the ability to reveal the committed value later. Commitment schemes are designed so that a party cannot change an already committed value. These schemes normally consist of two phases (additionally can also be considered another first phase, Setup):

1. Phase 1 (Commit): the Sender commits to a certain value p generating a commitment value c and sending it to the Receiver.
2. Phase 2 (Reveal): the Sender proves to the Receiver that the value p has not been changed sending to her the value p and the decommitment value d, which is used to calculate p from c.

Regarding security, all commitment schemes must accomplish two properties:

1. *Binding:* the Reveal phase can successfully decommit to one value only (uniqueness of the Commit phase).
2. *Hiding:* the Commit phase cannot reveal any information about the value (perfect secrecy of the value).

Both Binding and Hiding can be accomplished in a perfect way (statistical or computational) depending on the power needed to break them.

Commitment schemes are very important and useful primitives in cryptography; they are implemented in diverse cryptographic protocols, such as zero-knowledge proofs, multiparty computations, digital auctions and e-commerce [49, 58, 59].

In Chapter 4 of this thesis we have defined a new commitment scheme based on ECC to improve the security of the bootstrapping process of P2P overlays.

2.2.5 The AVISPA Tool

The AVISPA (Automated Validation of Internet Security Protocols and Applications) tool is a push-button tool for the automated validation of large-scale Internet security-sensitive protocols and applications [16, 27]. This tool has been developed by a group of institutions formed by the Artificial Intelligence Laboratory (AI-Lab) at DIST, Università di Genova, Italy; the Information Security Group at ETHZ, Zürich, Switzerland; the CASSIS group at INRIA, Nancy, France; and the Siemens AG, Munich, Germany. The objective of this technology is to speed up the development of the next generation of network protocols, improve their security, and therefore increase the public acceptance of advanced, distributed Information Technology (IT) applications based on them.

The AVISPA tool can be used to demonstrate proof-of-concept on a large collection of practically relevant, industrial protocols. It uses a modular and expressive specification language for formalizing protocols, security goals, and threat models of industrial complexity; the High-Level Protocol Specification Language (HLPSL).

AVISPA integrates different back-ends which implement a variety of automatic analysis techniques for protocol falsification and abstraction-based verification methods; both for finite and infinite number of sessions. Specifically, it implements the On-the-Fly Model Checker (OFMC), the Constraint Logic (CL-AtSe), the SAT-based Model-Checking (SATMC) and the Tree Automata-based Protocol Analyser (TA4SP) back-ends.

Many protocols have been verified using this tool obtaining relevant information about their validity. For this reason, we have used this tool to verify the correctness and the security properties of the protocols proposed in this thesis.

Chapter 3

Secure Identity Management

Contents

Identity management plays a crucial role in the security of P2P overlays, as we have seen in Section 2.1.3. For this reason, it is very important to pay attention on how nodes obtain their nodeIDs when they join a P2P overlay, and also how users check the validity of nodeIDs and authenticate their owners. In this context, nodeIDs should be generated taking into account certain requirements to ensure some control over the process and to avoid attacks.

In this Chapter, we analyze the advantages and drawbacks of the most common ways of generating nodeIDs that we can find in the literature, and we define a set of requirements which should be accomplished to construct a robust nodeID. Finally, we also describe some scenarios where P2P overlays can be used to improve services or applications and where providing security is indispensable.

3.1 NodeID generation

3.1.1 Using Random Numbers

The easiest way to generate a nodeID is by using a Random Number Generator (RNG). In this case, the resulting identifiers are uniformly distributed in the virtual space and guarantee the anonymity of their owners. However, if the generation is carried out in the client application, it can be easily manipulated by the user allowing an attacker to select her nodeID. An example of this kind of generation can be found in the eMule and aMule clients, where the obtained 128-bit identifiers are used in the Kad network allowing the attackers to tamper the proper operation of the network, to censor contents, or to damage other entities in the Internet out of the P2P overlay, such as web servers [79].

The problem is that this way of auto-generating nodeIDs allows users to generate more than one valid nodeID. Also, it avoids that other nodes can verify whether a nodeID has been properly generated or it belongs to whoever is using it, since nodeIDs are not bound to their owners in any way. In this context, if a P2P overlay does not implement an access control system, the network will be vulnerable to Sybil attacks and whitewashers, since any user can manage a set of nodeIDs and change them in an uncontrolled way. Thereby, BitTorrent and the Kad network are vulnerable to these attacks.

A special case of random nodeIDs are the CAN identifiers [99]. As we have explained in Section 2.1.2.1, a CAN newcomer chooses a random point p to select her assigned zone and this point makes the functions of nodeID within the network. Therefore, a malicious node can select a certain point within the virtual space to be placed in a target zone, for instance, to censor a certain file. Remember that if a malicious node can choose the zone that contains the *key* of a certain file, she will control this file being able to deny the access to it or to share a tampered version, among other actions. In addition, any user can select a different point p whenever they rejoin the CAN network, since the identifier generation is not related to any scarce resource that users can prove that they currently possess. In this context, a nodeID can never be verified and nothing prevents that a malicious user can use a set of points (p, p', p'', etc.) at a time (Sybil attack) to manage several zones within the overlay in order to corrupt the routing system, to encircle a victim node to control its communications or to isolate it (Eclipse attack). At the same time, this way of proceeding with the nodeIDs allows users to remain anonymous within the overlay.

Obviously, the nodeID self-generation problem could be avoided simply delegating the generation to a TTP (Trusted Third Party), which would generate these random numbers. This TTP could be a bootstrapping node, a Certificate Authority (CA), etc., but in most cases it does not apply any kind of access control. Therefore it is noteworthy that the provided security level will depend on the role of that TTP. If this entity is part of the overlay, it may place the new nodes (select the nodeIDs) seeking the benefit of the network, or maybe its own benefit. In Table 3.1 we analyze the use of a TTP only for generating random nodeIDs.

3.1.2 Using IP Addresses

One of the most common procedures to bind a nodeID to a scarce resource, which theoretically only has one owner, is to use an IP address. A priori, this is a good solution since all Internet users have an IP address and the nodeID generation can be easily verified. Moreover, this kind of nodeIDs could be revoked if necessary.

Some networks use the IP address directly, others use the digest of the IP address, and even it is also possible to generate a nodeID as any operation that takes the user's IP address as an argument. Chord [116], Pastry [106], Kademlia [85] and a security extension of BitTorrent [92] are some examples of P2P overlays which use the hash value of users' IP addresses to generate nodeIDs. However, these identifiers can cause some problems.

First of all, it is difficult to guarantee the nodeID stability based on an IP address. There are several reasons for this, for example, in today's Internet not all the assigned IP addresses are permanent but most Internet Service Providers (ISPs) assign dynamic IP addresses to their customers. On the other hand, a criminal organization running a botnet can use many physical nodes with different IP addresses. In addition, in the future IPv6 based Internet, a malicious node can use virtually thousands of IP addresses. For these reasons, the use of IP addresses to generate nodeIDs makes difficult to guarantee the nodeID stability. Without a proper access control system, P2P overlays are also vulnerable to Sybil attacks since users can easily manage more than one nodeID. The number of nodeIDs available to an attacker would be linearly bounded by the number of IP addresses that she can force the network to route to them.

Secondly, this way of generating identifiers does not prevent that a user can self-generate, to a greater or lesser extent, her nodeID. If a user has access to a pool of IP addresses, she can choose one of them interchangeably to generate her nodeID and locate herself in a certain zone of the virtual space. In addition, IP addresses are vulnerable to the IP-spoofing attack. Therefore, any user can supplant the identity of a node (identity theft attack), by spoofing her IP address. Obviously, they can also simply change her nodeID or camouflage herself spoofing an IP address.

Thirdly, if a nodeID generation simply uses an operation over the users' external IP addresses, a hash function for instance; all users behind a NAT are forced to use the same nodeID within the overlay. For this reason, the use of a random number, or another parameter, in the generation process of a nodeID, is absolutely necessary to guarantee that users behind a NAT also have a unique nodeID.

And finally, the anonymity of users can be compromised by the use of static IP addresses, since a malicious user can infer information about a node from her IP address. Even in the case of using a hash function to generate nodeIDs, the level of anonymity provided is poor or inexistent if the IP address is necessary to verify the nodeID.

In the security extension of BitTorrent DHT [92], we can find a special case of the use of IP addresses to construct nodeIDs using a hash function. On the one hand, the author proposes to use a random number in the range [0, 7] to allow users behind a NAT to have unique nodeIDs. Thus, eight users behind the same external IP address can obtain a different nodeID while attackers are limited to obtain eight nodeIDs per IP address class. And on the other hand, IP addresses are restricted at each class level to further restrict the number of nodeIDs that users can construct from large IP address blocks. To do so, a mask is applied to each IP address before hashing it together with the selected random number. This mask allows masking out more bits of the most significant octets to decrease the number of available nodeIDs for smaller IP blocks. This way, the total number of available nodeIDs grows slower than the number of IP addresses that an attacker can manage.

The proposed expression to construct a valid nodeID from IPv4 addresses is:

$$crc32c((IPv4\&0x030F3FFF)||(r << 29))$$

And for IPv6 addresses is (using the high 64 bits of IPv6 addresses):

$$crc32c((IPv6\&0x0103070F1F3F7FFF)||(r << 61))$$

Where CRC32C (Castagnoli) is the selected hash function, r is the random number in the range [0, 7] and the || operator means bit-wise OR. Finally, the most significant 21 bits of the result are the valid nodeID, and the last byte must correspond to the random number r. Unfortunately, this scheme potentially impacts the DHT routing and potentially break the $O(log\ n)$ lookup complexity [78].

Note that unlike other cases, using a TTP to generate nodeIDs based on IP addresses does not solve the problems discussed above, since the TTP cannot know if the IP address belongs to the user or has been spoofed, or even if the user is an attacker that controls a set of IPs.

3.1.3 Using Public Keys

Another way to bind a nodeID to a scarce resource is to use the user's cryptographic Public Key (PK) used within the overlay. More specifically, to use the value of a hash function over the PK, as Pastry [106] does. In the same way that using IP addresses, the use of PKs allows users to verify the identifiers of other nodes, and in general, this way of proceeding may also solve most of the previous problems of using IP addresses. However, nodeIDs can have different problems depending on the way that PKs are distributed and who has selected them.

Usually, PKs are distributed by means of Public Key Certificates (PKC)[1] to protect their integrity. However, the way in which these PKCs are generated is critical to our purpose. In particular, there are three important aspects: (1) which information related to nodes is contained in PKCs[2], (2) who selects the cryptographic key pairs, and (3) who issues (and manages) these PKCs.

On the one hand, PKCs may contain certain information related to the node, for instance, its IP address or even the real name of the user who is behind this node (or

[1]A PKC, also known as identity certificate, is an electronic document which uses a digital signature to bind a PK with an identity.

[2]PKCs usually contain many fields that are used for management purposes (serial number, validity period, issuer, etc.). We will omit these management fields in our discussion because they do not contain information related to the node.

some kind of pseudonym). Obviously, the inclusion of this information in PKCs can cause the loss of the anonymity of users.

Nevertheless, from our point of view, the most critical aspect is who generates the PKs. If users select their cryptographic key pairs they can obtain a nodeID located in a target zone of the virtual space. Otherwise, if PKs are selected by a TTP, nodeIDs cannot be chosen by the users, but the TTP can generate the PKs to locate the nodes in certain zones of the overlay. In this context, PKCs can be issued by a CA (Certificate Authority), by the node itself (self-signed certificate) or by a group of nodes. If PKCs are issued (and signed) by a CA, which is a trusted party for all entities of the network, it assures us that:

- Users manage only one node (PKC), or a small set of them.
- All information contained in PKCs is real (IP addresses, names of users, etc.).
- All this information is cryptographically bound to the PKs of nodes thanks to the signature of the CA.
- We can establish a secure connection with a node with the assurance that it is the intended principal.
- PKCs can be revoked if required.

Despite the use of CAs can present many advantages regarding security related to nodeIDs (*stability, revocability, uniqueness and not self-generation*), it also has some drawbacks. Besides the loss of anonymity, this type of generation forces all users to obtain a PKC issued by a CA. And this last fact can cause problems in terms of performance, since the CA is a centralized entity and a single point of failure.

As an alternative, nodes can use self-signed certificates, i.e., certificates issued and signed by their owners. This mechanism is totally decentralized, users do not have to contact with a CA, but it has more security problems:

- Users can select specific PKs to construct certain nodeIDs.
- Anyone can create new identities by simply generating new cryptographic key pairs and issuing the corresponding certificates.

- A single user can manage more than one PKC at a time, which is equivalent to manage several nodes.
- Nobody can guarantee that the information inside a PKC is real.

In this context, Marti and Garcia-Molina [83] also proposed to use self-signed certificates to guarantee the anonymity of users and avoid the problem of identity theft through the IP-spoofing attacks. But in their case, nodeIDs are generated randomly.

Finally, PKCs can be issued by a group of nodes (usually with high availability and good reputation) thanks to the use of the threshold cryptography [109]. This type of cryptography provides a mechanism to distribute cryptographic primitives guaranteeing that these are secure against a group of up to $t-1$ members, a threshold. Usually, secrets are distributed among a group of n members so that at least t members are needed to sign, encrypt, validate, etc. a message. This alternative is decentralized, scalable and prevents users select their nodeIDs because PKs are generated cooperatively by the group of members. In addition, the information contained in the PKCs is cryptographically bound to the nodes' PKs thanks to the signatures of the group members. However, some drawbacks of using self-signed certificates are also related to this kind of PKs generation:

- A single user can manage more than one PKC at a time, which is equivalent to manage several nodes.
- Nobody can guarantee that the information inside a PKC is real.

Note that if we want to provide a higher level of security, those nodes that make up the group should also carry out an access control process based on the identity of the nodes, their reputation, etc.

3.1.4 Comparison

Tables 3.1 and 3.2 summarize the characteristics of the six ways to generate nodeIDs that we have discussed in this Section. We describe the behavior of each of these ways to generate nodeIDs in relation to the protection against self-placement of nodes within the overlay (Eclipse protection), the provided stability of nodeIDs, the anonymity provided, the protection against Sybil attacks and the ability to verify the validity of nodeIDs. To do so, we have defined three different levels of fulfillment: Null, Medium and High.

Note that the involved entities in the nodeIDs generation are not the same in these six cases. Therefore, trust in such methods cannot be the same. In the methods LRN and SPK (see Tables 3.1 and 3.2), the user is responsible for self-generating her nodeIDs, which provides a poor level of trust. Users could select specific nodeIDs to locate themselves in a certain zone of the overlays' virtual space, which would allow them to carry out attacks as the Eclipse attack.

Something similar happens when IP addresses are used to generate the nodeIDs (method IPA), since users can manipulate them. A malicious user could select a certain IP address to locate herself in a certain zone of the overlays' virtual space, and then spoof that IP to generate valid IP packets from it. In this way, the receiver would verify the nodeID using the used IP address successfully. Unfortunately, today there are many applications to carry out this illegitimate action. However, we want to remark that this technique also has legitimate uses, as the performance testing of websites.

Other option is to delegate the nodeIDs generation to a TTP (methods TRN and TPK), which avoid that users can select their nodeIDs. In both cases, we cannot know if this TTP will have interest in choosing certain nodeIDs to users to obtain particular profit. A Certification Authority (CA) is a special case of TTP, but in the method TPK we will assume that CAs have a high degree of trust regarding access control.

Finally, in the method GPK, we consider the use of a group of nodes to generate the users' nodeIDs. Those nodes usually have good reputation within the overlay and the implemented system can hinder a collusive behavior, but this is not enough to consider this way of generating nodeIDs as a robust solution. On the one hand, the reputation of the nodes should be provided by a robust reputation system, which is very difficult if the problem with the nodeIDs is not solved yet. And on the other hand, there is a tradeoff between practicality and security. Requiring the signature of a large group of nodes could bring performance and availability problems. Otherwise the security would be poor.

As a result we conclude that an ideal construction, for security concerns, would be very close to the use of PKs issued by a CA. But in that case, the problem to solve is the user anonymity.

Table 3.1: Evaluation of the most used six ways to construct nodeIDs in current P2P overlays (I).

Construction	***Eclipse protection***	***Stability***	***Anonymity***	***Sybil protection***	***Verifiability***
Using local random numbers (LRN)	Null Users can manipulate the random number generator or simply use the number they want.	Null NodeIDs are not linked to users and nothing prevents their change.	High It is impossible to link a user with a nodeID.	Null Anyone can manage a large number of nodeIDs.	Null It is impossible to verify whether nodeIDs were properly generated.
Using random numbers generated by a TTP (TRN)	High The TTP sends the number that users must use.	Null NodeIDs are not linked to users and nothing prevents they use new nodeIDs.	High It is impossible to link a user with a nodeID.	Null Anyone can manage a large number of nodeIDs.	High The TTP certifies the nodeID validity and no one can fake it. Anyone can verify a nodeID.
Using IP addresses (IPA)	Medium Vulnerable to IP-spoofing attacks. Choosing a specific IP-based nodeID is difficult but not impossible.	Medium Many nodes use dynamic IP addresses. IPv6 can provide a lot of addresses to a single node.	Medium In many cases a user could be linked to a static IP address.	Medium Users can manage several nodeIDs managing a set of IP addresses. Malicious users running a botnet have many IP addresses.	Medium It is possible to check a nodeID with the IP address of messages, but NATs can complicate matters.

Table 3.2: Evaluation of the most used six ways to construct nodeIDs in current P2P overlays (II).

Construction	***Eclipse protection***	***Stability***	***Anonymity***	***Sybil protection***	***Verifiability***
Using self-generated PKs (SPK)	Null Users can generate the PKs to construct a certain nodeID.	Null PKs are not linked to users and nothing prevents their change.	High It is impossible to link a user with a self-managed certificate.	Null Anyone can manage a large number of self-managed certificates.	Null It is impossible to check whether a PK has been selected maliciously.
Using PKs issued by a CA (TPK)	High The CA sends the crypto-graphic key pairs that users must use.	High Users can only change their certificates in case of revocation, and controlled by the CA.	Null Real-world certificates are linked to a real person.	High Users can only obtain one PKC.	High Anyone can calculate the nodeID of a certain user using the PK contained in her PKC.
Using PKs issued by a group of nodes (GPK)	High A group of nodes takes part in the generation of the crypto-graphic key pairs.	Null These certificates are not linked to real users and nothing prevents they use a new PKC.	High It is impossible to link a real user with one of these certificates.	Null Anyone can manage a large number of PKCs.	High The group of nodes certifies the PK validity and no one can fake it. Anyone can verify a PK.

3.2 Security Requirements for nodeIDs

Many authors have already addressed the problem of assigning robust identifiers to nodes. Some of these proposals have focused their efforts on the context of anonymity and reputation [51, 70, 107, 120]. Some others have tried to detect or limit the scope of certain attacks taking advantage of the computational cost of generating nodeIDs [24, 53, 81, 105]. And more recently, several systems have also begun to fight the effect of Sybil attacks by understanding the social context in which they occur [118, 127, 128]. However, little attention has been paid to the way that nodeIDs should be constructed in P2P overlays and the security requirements that they should meet.

Obviously, not all applications/services require the same security level, but there is no doubt that any commercial proposal should always take into account a minimum of security requirements to avoid the above stated problems. For this reason, P2P overlays should implement proper identity management systems to generate and assign robust identifiers to nodes, to verify whether nodeIDs were properly generated and assigned, and to revoke them if necessary. Next, we define a set of requirements which should be accomplished by any nodeID to be considered robust.

- *Uniqueness:* each user, ideally, should only manage one node (nodeID) within the system, or a small set of nodeIDs at worst. This requirement is necessary to prevent (or limit) the Sybil attack. Using a TTP is probably the best way to limit this attack, and for that reason, in this thesis we consider this method as the only way to properly control the user access to the p2p network. We will consider that each user has a unique real-world identity, for instance, the one contained in an ID card. Obviously, it is not impossible to have more than one of these real-world identities, but their use can provide a sufficient level of control for most commercial services.

- *Stability:* nodes should not have the possibility to change their nodeIDs each time they join the network. This requirement is necessary among other purposes to avoid that malicious users can rejoin the network if they have been expelled or to implement efficient reputation systems within the overlay (without whitewashers). In this sense, TTPs should store a link between users and nodeIDs to control users only use one nodeID.

- *Joint Management:* neither nodes themselves nor a TTP should be able to choose the users' nodeIDs unilaterally, that is, the location within the virtual space. Ideally, users and TTPs should generate nodeIDs jointly, for example, selecting half of the bits of the nodeID each. This requirement is necessary to avoid the Eclipse attack (the creation of plots), among other vulnerabilities.
- *Verifiability:* all nodes should be able to check if a nodeID has been properly generated and whether it belongs to whoever is using it. To make this possible, it is very important to consider how nodeIDs are generated. This requirement is necessary to minimize attacks within the network.
- *Revocability:* any user should be able to revoke her certificate in case of losing it or seeing compromised the associated private key. If possible, this user whose certificate has been revoked should be able to get a new certificate maintaining the same nodeID. In addition, if a dishonest or malicious behavior is detected (and demonstrated) within the overlay, the TTP may be able to revoke the certificate of that node (and her nodeID), avoiding malicious users to access the overlay again.
- *Traceability:* if a user commits an illegal action within the overlay and she must be judged for it, authorities should have the possibility to trace the user and match the nodeID with the user's real-world identity. This requirement needs a link between nodeIDs and users' real-world identities, which can be provided by a TTP. Unfortunately, this requirement would jeopardize the anonymity of the users, which is one of the main strengths of P2P networks. Therefore, a way of making traceability and anonymity compatible in these networks should be found.
- *Anonymity:* no internal or external entity should have the possibility to relate the real-world identity of a user with her nodeID. In an environment where a TTP is being used, nodes should be anonymous even for the TTP.
- *Uniformity:* nodeIDs should be uniformly distributed in the virtual space. This requirement is necessary to achieve the proper load-balancing among all nodes of the overlay. Otherwise, P2P overlays do not achieve the proper operation. To this respect, in our protocols we always use a cryptographic hash function to obtain the final value of nodeIDs, which provides a high grade of uniformity when distributing the values.

Obviously, most of the above requirements need of a centralized entity to ensure their compliance and to limit the number of nodeIDs per user. For this reason, throughout this thesis, we have decided to use TTPs to generate nodeIDs in the form of digital certificates, both standard and implicit certificates.

3.3 Scenarios

So far we have discussed several problems related to the management of identities in P2P overlays and we have defined some requirements to be fulfilled by those identifiers in order to improve the security and the performance in these networks. But now let us consider some examples of scenarios where it is relevant that some or all of these requirements are fulfilled.

The first of such scenarios could be a multi-player payment game over a P2P overlay. In this scenario, we might have to avoid situations such as that a user who has lost a game tries to rejoin to the same game with another nodeID, or that a certain user tries to locate herself numerically close to the user that has more score to for example try to eclipse that user. Obviously, in this scenario it is also necessary to control the access to the service to avoid Sybil attacks and to charge each player. At the same time, users might be interested in preserving their anonymity because they might not want to make public the type of games they play.

Another scenario where it can be necessary to have a controlled assignment of identities is a video streaming or Video on Demand (VoD) service that uses a P2P overlay to distribute the content. VoD services are currently being widely used in the Internet and the traffic that they generate is a big percentage of the total traffic of the current Internet. In such a scenario, the service provider might control the content distribution and the access to the service, but users might not want to reveal their real identity to the service provider. Thus, on one hand users might want to be anonymous but on the other hand, the service provider might want to have a robust management of nodeIDs. A solution for this issue that we develop later in the thesis is to use an independent TTP to aid in the identity generation process. This TTP could be used by many different service providers as a proxy to avoid that users can obtain more than one nodeID and also to avoid the forgery or theft of these identifiers. This solution would benefit both parties; service providers and users. Service providers can protect their networks from identity related attacks and guarantee a better quality of service, while users can maintain their anonymity and benefit from a good service. Moreover, this type of solution can also guarantee the nodeIDs stability and enable service providers the implementation of a robust reputation system.

Finally, other scenarios that can use P2P overlays and in which a secure and robust identity management is necessary can be ad hoc meetings, where confidential information can be revealed, or instant personal matching services, where a teenager can reveal her tastes and opinions among other relevant information. These do not need to preserve the anonymity of users, users can utilize their real identities, but it is very important to avoid that an attacker can steal an identity to join the overlay impersonating another user. Obviously, it is also very important to control the access to the service to guarantee that only authorized users join the network.

Chapter 4

An Implicit Certificate-based Identity Assignment Protocol for P2P overlAys

Contents

In this chapter, we propose an Implicit Certificate-based Identity Assignment Protocol for the P2P overlAys (ICIAPPA). Leveraging the issuance of implicit certificates and authenticating all newcomers, ICIAPPA provides traceability of users' actions, partial anonymity[1] and a secure and efficient way of assigning nodeIDs.

[1]We consider partial users anonymity when network users cannot bind the real-world identity of another user to his nodeID but a TTP can do it.

Implicit certificates provide digital identification in the same way that explicit certificates do but presenting certain advantages. In particular, implicit certificates are smaller because they do not include neither the issuer signature nor the user public key, but a single value combining both. The main property of implicit certificates for our purposes is that they enable us to generate robust nodeIDs. In ICIAPPA, each nodeID is computed as a digest of the user's public key. However, unlike other proposals of the literature, each public key is generated under the supervision and participation of a TTP using as foundation the ECQV implicit certificate scheme [103], explained in the Section 2.2.2. In more detail, ECQV has been modified to ensure that neither of the two parties involved in the scheme has the ability to choose the value of the resulting public key. Consequently, a user cannot decide unilaterally what will be her nodeID and the TTP cannot impersonate a user[1]. On the other hand, nodeIDs are easy to verify and they meet several of the security requirements discussed in Section 3.2. In particular, joint management, verifiability, revocability and uniformity.

4.1 Assumptions and Clarifications

The ICIAPPA protocol essentially uses ECQV but it starts with a commitment scheme. Remember that in the basic version of ECQV, the user that is going to be certified is who first creates and sends a random point N_X to the TTP. After that, the TTP selects another random point N_C to finally build a public key reconstruction parameter Z_X. However, for a robust assignation of the nodeIDs, we must prevent that the TTP can deliberately choose the reconstruction public parameter of any user (Z_X), since the nodeID is derived from this value. To fix this, in our protocol, the TTP must select N_C before receiving N_X and this is accomplished using a commitment scheme. On one hand, the commitment scheme allows the TTP to demonstrate that it selected its part of the public key a priori and, on the other hand, the commitment scheme allows the user to verify that the certificate received has been generated using a proper committed value.

The commitment scheme that we propose for ICIAPPA uses elliptic curve cryptography. Our scheme is inspired from the Exclusive-OR (XOR) encryption algorithm

[1]Note that we consider that a TTP is trusted for the action of issuing certificates following a certain protocol, but we do not require trusting it for the rest of the actions performed in the overlay, such as content distribution.

[41]. Figure 4.1 illustrates our commitment scheme. As shown in 4.1, we assume that a sender S possesses a private key d_S and a public key $Q_S = d_S G$ and wants to make a commitment for a random value v. Then, the sender creates a nonce u and its associated elliptic curve point $U = uG$. Then, $c = v \otimes H(d_S U)$ is the commitment value for v, where $H()$ is a hash function. In the first phase (Commit), the sender commits v by sending the tuple (c, U) to the receiver. In the second phase (Reveal), the sender reveals u. The receiver checks that $U = uG$ and calculates v using the expression $v = c \otimes H(uQ_S)$.

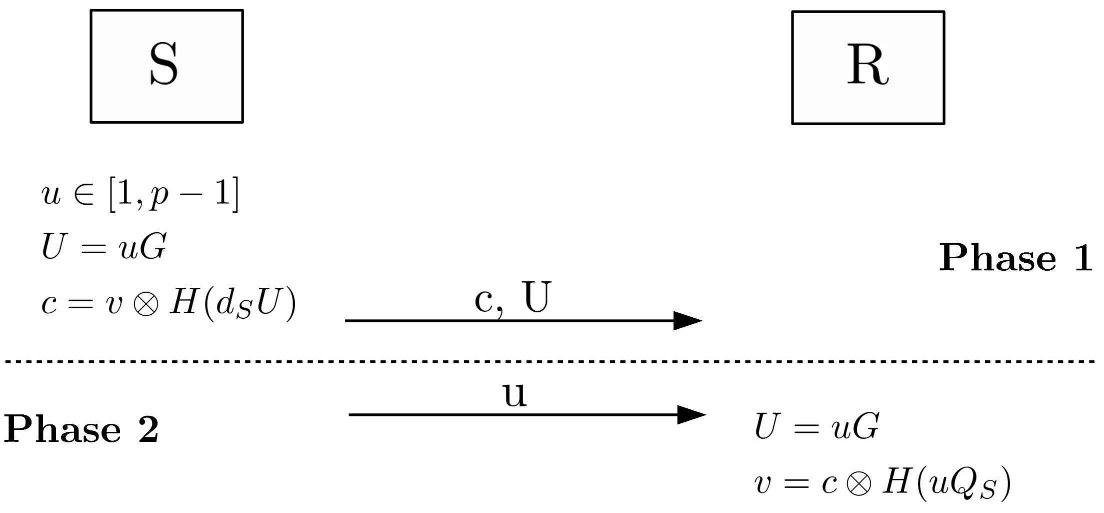


Figure 4.1: ICIAPPA Commitment Scheme.

The strength of the previous commitment scheme resides in the fact that an attacker cannot create a commitment value c for v without knowing the source's private key. Also, the source cannot modify v once it has sent the corresponding commitment value.

During the protocol specification, we will use encryption mainly to provide confidentiality. However, sometimes it will also be used to bind different parts of a message. Digital signatures are mainly used to ensure both message integrity and authentication. Although we do not assume that the identity of the signing principal can be deduced from this signature, since this deduction is not possible in all digital signature schemes. For this reason, we always state the identity of the signing principal within the same message to avoid classical security protocol vulnerabilities [21].

Regarding other protocol details, specially the control parameters, we use a timestamp to fulfill two purposes. Firstly, the timestamp is used as a proof of timeliness to guarantee the freshness of the user's request and to avoid replay attacks. To do so we will assume a certain synchronism between the clocks of all involved parties, precise enough so the recipient of a message can consider it valid if the timestamp is within a reasonable interval of this recipient's local time. In this sense, a protocol like the Network Time Protocol (NTP) [89] may be enough for this purpose. If any of the involved entities consider that the timestamp of a message is not fresh enough, the identity assignment process is canceled automatically. For this, it sends an error message to the rest of the involved parties to delete all the information related to that request. Secondly, we also use this timestamp as a request identifier, that is, this value will be used by all the involved entities to unequivocally identify a request. This also means that this timestamp should be unique in all the system, so nodes should generate them to have enough resolution to assure this uniqueness. In this sense, timestamps of 64 bits, as the ones used by the NTP protocol, may be a good option to use.

4.2 Protocol Specification

In this Section we describe ICIAPPA: what information is exchanged between the two parties (Newcomer and TTP), how data is exchanged, what security mechanisms are used and so on. Figure 4.2 shows the essential data (without several details) exchanged in this protocol.

Regarding the notation used along the protocol description, we use $\{m\}_K$ to represent the ciphertext of a message m encrypted under a key K and $\{m\}_{K^{-1}}$ to represent a signature on a message m using the private key K^{-1} just as Abadi and Needham adopt in [21]. Table 4.1 shows a summary of the global notation.

Parameter	Legend
p	The order of the underlying finite field $\mathbb{F}_p$.
G	The generator of the elliptic curve defined over $\mathbb{F}_p$ ($E(\mathbb{F}_p)$).
ID_X	The identity of X.
ID_{TTP}	The identity of the TTP.
P_X	The pseudonym of X within the overlay (nodeID).
C_X	The digital certificate of X in the real-world.
d_X	The private key of X in the real-world.
Q_X	The public key of X in the real-world.
d_{XO}	The private key of X within the overlay.
Q_{XO}	The public key of X within the overlay.
d_{TTP}	The private key of the TTP.

Q_{TTP}	The public key of the TTP.
t_X	The timestamp (request identifier) generated by X.
I_X	The information that is included in the X's certificate.
Z_X	The X's reconstruction public parameter.
h_X	The digest of the new certificate ($h_X = I_X \|\| Z_X$).
s_X	TTP signature of X's certificate.
(I_X, Z_X)	The X's overlay certificate.
$n_C,\ u_C,\ n_X$	Private parameters of the X's request ($\in [1,\ p-1]$).
$N_C,\ U_C,\ N_X$	Public parameters of the X's request ($\in E(\mathbb{F}_p)$).
C_C	The commitment point sent by the TTP.
$H_1(m)$	A secure hash function on a message m.
$H_2(P)$	A secure hash function on a point P which also outputs a point.
$i \rightarrow j :$	The sending of a message from the entity i to the entity j.
$\{m\}_Q$	The ciphertext of a message m encrypted using the public key Q.
$\{m\}_d$	The signature on a message m using the private key d.

Table 4.1: Notation for ICIAPPA.

4.2.1 Protocol Steps

Step 1:

When a user wants to join the P2P overlay, she must send a "HELLO MESSAGE" to the corresponding TTP. This message contains the identity of the user and the TTP (ID_X and ID_{TTP}), a timestamp (t_X), and the user's digital certificate in the real world (C_X). The HELLO MESSAGE is signed by the user using her real-world private key and encrypted using the TTP's public key.

$$HELLO\ MESSAGE,\ X \rightarrow TTP:\ \{C_X, \{ID_X, ID_{TTP}, t_X\}_{d_X}\}_{Q_{TTP}}$$

The request identifier (t_X) must always be encrypted to avoid that other users can use the same t_X for malicious purposes. It is also noteworthy that the inclusion of the identities in the signed message is needed to avoid that the receiver can forward this message to another recipient posing as X [21]. In fact, to avoid this attack the identities of the sender and the receiver are included in all the messages of the protocol.

Step 2:

The TTP generates two nonces: $\{n_C,\ u_C\} \in [1,\ p-1]$ and their corresponding points: $N_C = n_C G$ and $U_C = u_C G$.

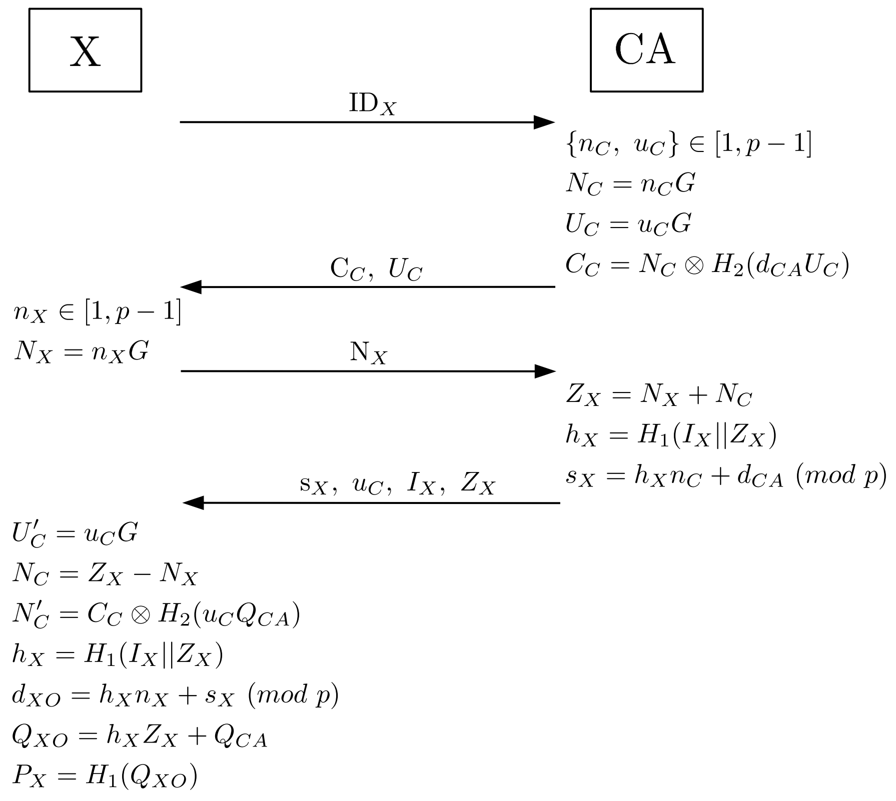

X
CA
ID_X
$\{n_C,\ u_C\} \in [1, p-1]$
$N_C = n_C G$
$U_C = u_C G$
$C_C = N_C \otimes H_2(d_{CA} U_C)$
$\mathrm{C}_C,\ U_C$
$n_X \in [1, p-1]$
$N_X = n_X G$
N_X
$Z_X = N_X + N_C$
$h_X = H_1(I_X \| Z_X)$
$s_X = h_X n_C + d_{CA}\ (mod\ p)$
$\mathrm{s}_X,\ u_C,\ I_X,\ Z_X$
$U'_C = u_C G$
$N_C = Z_X - N_X$
$N'_C = C_C \otimes H_2(u_C Q_{CA})$
$h_X = H_1(I_X \| Z_X)$
$d_{XO} = h_X n_X + s_X\ (mod\ p)$
$Q_{XO} = h_X Z_X + Q_{CA}$
$P_X = H_1(Q_{XO})$

Figure 4.2: Basics of the Certificate and nodeID Generation Scheme.

Then, the TTP calculates $C_C = N_C \otimes H_2(d_{TTP} U_C)$ and it sends C_C and U_C (the commitment values) to X, all signed and encrypted together with the involved identities and the timestamp.

$$ACCEPT\ MESSAGE,\ TTP \rightarrow X:\ \{\{ID_{TTP}, ID_X, t_X, C_C, U_C\}_{d_{TTP}}\}_{Q_X}$$

Step 3:

Using the "ACCEPT MESSAGE", X verifies the signature and decrypts the message. After that, X generates a random private number: $n_X \in [1,\ p-1]$ and calculates a public point $N_X = n_X G$. Then, X sends N_X to the TTP, signed and encrypted together with the identities and the timestamp.

$$REQUEST\ MESSAGE,\ X \rightarrow TTP:\ \{\{ID_X, ID_{TTP}, t_X, N_X\}_{d_X}\}_{Q_{TTP}}$$

Note that the freshness of the timestamp t_X must be checked in all the steps. If the timestamp is outdated at any step, the protocol must be canceled.

Step 4:

Using the "REQUEST MESSAGE", the TTP verifies the signature and decrypts the message. After that, the TTP calculates the reconstruction public parameter as $Z_X = N_X + N_C$. Then, it calculates the hash value of that parameter concatenated with the certificate information (I_X): $h_X = H(I_X||Z_X)$. Finally, the TTP signs the certificate calculating $s_X = h_X n_C + d_{TTP}\ mod\, p$, and provides X with $\{s_X, u_C, I_X, Z_X\}$, all signed and encrypted together with the identities and the timestamp.

$$RESPONSE\ MESSAGE,\ TTP \rightarrow X:\ \{\{ID_{TTP}, ID_X, t_X, s_X, u_C, I_X, Z_X\}_{d_{TTP}}\}_{Q_X}$$

Note that once Z_X and h_X have been calculated, if $Q_{XO} = h_X Z_X + Q_{TTP} = \mathcal{O}$, the TTP must request the user a different private parameter (N_X) sending a "REPLAY MESSAGE", and repeat the process until $Q_{XO} \neq \mathcal{O}$.

$$REPLAY\ MESSAGE,\ TTP \rightarrow X:\ \{\{ID_{TTP}, ID_X, t_X, N_X\}_{d_{TTP}}\}_{Q_X}$$

Step 5:

X receives her new certificate and the signature of the TTP in the "RESPONSE MESSAGE". Then, she computes $U'_C = u_C G$ and compares it with U_C, calculates $N_C = Z_X - N_X$ and compares it with $N'_C = H(u_C Q_{TTP}) \otimes C_C$ to verify that the TTP has used the initial value n_C (if this check fails, X cancels the joining process). Then, X generates her private key $d_{XO} = h_X n_X + s_X\ mod\ p$, her public key $Q_{XO} = d_{XO} G$ and calculates her pseudonym (nodeID) as: $P_X = H(Q_{XO})$. Finally, X creates a "CONFIRMATION MESSAGE" which contains her nodeID, identities and the timestamp, all signed (using the overlay private key) and encrypted (using the TTP's public key).

$$CONFIRMATION\ MESSAGE,\ X \rightarrow TTP:\ \{\{ID_X, ID_{TTP}, t_X, P_X\}_{d_{XO}}\}_{Q_{TTP}}$$

Step 6:

In the last step, the TTP receives the "CONFIRMATION MESSAGE" from the newcomer, decrypts it, and verifies the signature and the nodeID of X using the overlay public key (Q_{XO}).

4.2.2 Public Key Generation

Whenever a user receives a message, she needs to generate the sender's public key to authenticate her and verify the included signatures. To do so, she uses the sender's implicit certificate, which includes the certificate's information (I_X) together with the reconstruction public parameter (Z_X), and follows the next steps (X is considered the sender):

1. Calculates the parameter $h_X = H(I_X || Z_X)$.
2. Generates the sender's public key as $Q_{XO} = h_X Z_X + Q_{TTP}$.
3. Verifies the message signature using Q_{XO}.

Note that this is accomplished because:
$Q_{XO} = d_{XO}G = h_X n_X G + s_X G = h_X n_X G + h_X n_C G + d_{TTP} G = h_X N_X + h_X N_C + Q_{TTP} = h_X(N_X + N_C) + Q_{TTP} = h_X Z_X + Q_{TTP}$.

4.2.3 NodeID Validation

Whenever a user receives information from another user (contents or routing info) she should validate her nodeID. To do so, the receiver only needs to compute the digest of the sender's public key ($P_X = H(Q_{XO})$) and compare the result with the used identifier. Note that she must generate the X's public key previously.

4.3 Security Analysis

ICTIAPPA provides a mechanism for issuing robust and traceable identities in a P2P overlay while the anonymity of the users is partially preserved; only the TTP knows the real-world identity of users and their nodeIDs. Next, we analyze the behavior of this protocol against some of the most common attacks in a P2P overlay.

Sybil attack. As we discussed in Section 2.1.3, to fully avoid the Sybil attack in a P2P overlay it is necessary to guarantee that a user can only obtain a unique nodeID. If that is not possible, to guarantee a limited number of nodeIDs per user at least minimizes the potential damage of the attack. In ICIAPPA, users must authenticate themselves to the TTP using a unique real-world identity (like the one provided by identity cards or passports). Thus, our way of dealing Sybil attacks is based on the hypothesis that these real-world identities are much harder to spoof than any other type of identity. In this way, if it is difficult for a (normal) person to assume multiple IDs in the real-world, it will then be equally difficult for that person to assume multiple identities within the network [50]. Obviously, we consider that it would be adequate that the credentials used by users to authenticate themselves should be issued by trusted and recognized entities, for instance CAs which depend on a government. In a less strict and commercial scenario, users could use their credit cards as a credentials while they make a payment. In this case, if a user has more than one credit card she can obtain more than one nodeID, but the number of credit cards per user is normally quite limited.

Eclipse attack. To avoid an eclipse attack it is imperative to avoid two different actions which can be carried out by attackers. On the one hand, attackers should not be able to place themselves close to a target node. In this way, attackers could not select a certain victim to launch the attack. On the other hand, it is also necessary to avoid (or limit) the Sybil attack to prevent a single attacker can encircle a victim. Only in the case that both actions are avoided we can say that the Eclipse attack is avoided. As we have seen above, ICIAPPA avoids both actions and therefore the Eclipse attack.

MITM attack. During the bootstrapping phase, it is theoretically impossible for an attacker to extract information or to modify the proper working of ICIAPPA, since all messages of the protocol are signed and encrypted using public key cryptography. Hence, attackers only have a way to issue a valid overlay certificate, compromising the cryptographic keys of the TTP. In the same sense, during the normal working of the P2P overlay, an attacker cannot impersonate another user if she does not compromise her private/public keys pair, since all nodes should sign all the messages they send.

Whitewashers. ICIAPPA tries to limit the number of nodeIDs that a user can manage binding their real-world identities with their nodeIDs. However, it does not guarantee the stability of those nodeIDs, since every time an overlay certificate is revoked the nodeID changes. Specifically, each implicit certificate has bound a different cryptographic public key, which is used to genera the associated nodeID. Therefore, we must to clarify that whitewashers can take advantage of this feature but the TTP can monitor their changes of nodeIDs.

4.3.1 Cryptographic Analysis

Each of the cryptographic tools used in this protocol has a range of security levels, which are measured in bits, and each extra bit of security doubles the amount of computation needed by a brute-force attack to compromise the security of a system. We must take into account that the security level of each selected tool must meet or exceed the desired security level of the whole system.

The used RNGs must fulfill the ANSI X9.82 [26] or the NIST Special Publication 800-90A [94], since the secrecy of the cryptographic keys depends on the security level of these generators. In [101], a suitable RNG is described.

In ICIAPPA, the hash functions are used to compute the user's nodeID, certificate digests and message digests. The security level associated with a hash function depends on its application and whether collision resistance is necessary or not. In the first case, the security level is at most half the output length (in bits) of the hash function. In the second case, it is at most the output length of the function. In ICIAPPA, collision resistance is needed in all situations. For this reason, a 256-bit hash function as SHA256 [95] could be a good option to implement the protocol, since it provides a 128-bit security level.

Regarding the used elliptic curve domain parameters, their generation and validation should follow the guidelines in [101]. These domain parameters consist of six values:

1. p, the size of the finite field $\mathbb{F}_p$.
2. $a, b \in \mathbb{F}_p$, the two parameters which specify the elliptic curve $E(\mathbb{F}_p)$ defined by the equation: $y^2 \equiv x^3 + ax + b \pmod{p}$.
3. G, the base point generator.
4. n, the order of the base point generator.
5. And h, the cofactor, which is the number such that $h \cdot n$ is the number of points on the elliptic curve.

The figure $\lceil log_2\ p \rceil$ is another parameter that indicates the security level which is associated with an elliptic curve. In particular, an elliptic curve with $\lceil log_2\ p \rceil = 2t$ supplies approximately t bits of security, which means that solving the logarithm problem on this elliptic curve is believed to take approximately $2t$ operations. In [102] there is a list of recommended elliptic curves for cryptography. From them, a 256-bit Koblitz curve could be a good choice to combine with the SHA256 function, since both provide a 128-bit security level.

Regarding the proposed commitment scheme, it accomplishes the two main security properties that are required in this kind of protocols: *Binding* and *Hiding*. Our scheme is perfectly binding because two different values of u_C cannot output the same correct value of v_c. It is also computationally hiding because given the values of C_C, U_C and G, an attacker cannot solve the ECDLP to obtain the random value u_C, and then to calculate v_c (this is an intractable problem as commented in Section 2).

Once an implicit certificate is issued, its owner must demonstrate the knowledge of the associated private key (d_{XO}) by using a signature algorithm, such as ECDSA. Note that the fact of generating her associated public key (Q_{XO}) is not a sufficient proof that Q_{XO} is authentic. In this scheme, d_{XO} is generated using the TTP's signature (s_X), and this signature is calculated using the TTP's private key (d_{TTP}) together with the private parameter selected by the TTP (n_C). Q_{XO} is generated by the other users of the network using the TTP's public key (Q_{TTP}) and the reconstruction public parameter (Z_X) which is included in the user's certificate. Thus, if a user validates a signature using Q_{XO}, she can be sure that the correct TTP's public key has been used in generating that key. Otherwise, the signature validation will fail. The same occurs if a wrong Z_X is used.

In the case that a malicious user wants to find the private parameter selected by the TTP (n_C) to select the value of her new public key (Q_{XO}), she should also solve the ECDLP, which is computationally unfeasible.

4.3.2 Discussion of NodeID Requirements

In this Section, we discuss how some of the design requirements exposed in Section 3.2 are satisfied by nodeIDs obtained with ICIAPPA. Thanks to the binding between the users' real-world identities and their nodeIDs, each user can only obtain a limited number of nodeIDs. In particular, we can create a ***unique*** nodeID if, for instance, the user's national identity card is used.

NodeIDs are ***jointly generated*** between users and the TTP, avoiding that a user can select her location within the overlay. Regarding the nodeID generation process, it also guarantees that nodeIDs are randomly generated; therefore their distribution within the virtual space of the network is ***uniform*** with a high probability.

ICIAPPA can also provide ***partial anonymity*** within the overlay, since nodes are not able to match the users' real-world identities with the nodeIDs. This is achieved thanks to the use of encryption during the message exchange process, since all exchanged messages are encrypted, so no information is leaked to external entities. However, users are not anonymous to the TTP. Regarding the requirement of ***traceability***, thanks to the fact that the TTP knows the real-world identity of the users, it can easily relate their identities with their nodeIDs.

On the other hand, all implicit certificates are ***verifiable*** by anybody. Any user can verify any nodeID using the public key of its owner derived from the certificate's information and the public key of the TTP. For this reason, nobody can generate a valid nodeID without the supervision and participation of the TTP.

Finally, implicit certificates are also ***revocable***. Any certificate can be revoked if a user is misbehaving or the involved private key has been compromised. In this last case, the user needs to prove the real-world identity related to the overlay certificate. Although this protocol does not provide ***stability*** of nodeIDs. Every time a user obtains a new overlay certificate, it will be generated a different public/private key pair for the overlay. Therefore her nodeID will also change.

4.3.3 Formal Validation of the Protocol

Moreover, we have performed an automated validation of the message exchange using the AVISPA tool to validate the security of ICIAPPA. We ran the tool using three different back-ends (OFMC, CL-AtSe and SATMC), and in all three cases the output was "SAFE".

Table 4.2 shows the five messages exchanged between a newcomer X and the TTP. ID_X and ID_{TTP} are the identifiers of the user X and the TTP, and K_X and K_{TTP} are their cryptographic keys respectively. Functions $\{m\}_K$ and $\{m\}\_inv(K)$ represent the encryption and signing of the message m using the key pair K and $inv(K)$ respectively. Finally, "." indicates concatenation.

1	X	$\rightarrow$	TTP	:	$\{C_X\{ID_X.ID_{TTP}.t_X\}\_inv(K_X)\}\_K_{TTP}$
2	X	$\leftarrow$	TTP	:	$\{\{ID_{TTP}.ID_X.t_X.C_C.U_C\}\_inv(K_{TTP})\}\_K_X$
3	X	$\rightarrow$	TTP	:	$\{\{ID_X.ID_{TTP}.t_X.N_X\}\_inv(K_X)\}\_K_{TTP}$
4	X	$\leftarrow$	TTP	:	$\{\{ID_{TTP}.ID_X.t_X.s_X.u_C.I_X.Z_X\}\_inv(K_{TTP})\}\_K_X$
5	X	$\rightarrow$	TTP	:	$\{\{ID_X.ID_{TTP}.t_X.P_X\}\_inv(K_X)\}\_K_{TTP}$

Table 4.2: ICIAPPA Message Exchange for the AVISPA Tool.

In Appendix B.1 we show the HLPSL code of the validation performed and outputs obtained.

4.4 Performance Analysis

We have developed a secure identity assignment protocol to avoid most of the threats related to identities in a P2P overlay. However, security measures consume bandwidth, memory and processing time. In this context, we can state that implicit certificates decrease the resource consumption of the security mechanisms implemented. Implicit certificates use a reconstruction public parameter (Z_X). This reduces the number of bytes that a sender needs to send to a receiver, since the TTP's signature no longer needs to be sent (64 bytes if a 256-bit elliptic curve is used, 33 bytes in the compressed format), and the processing overhead, since authenticating a message with an implicit certificate requires one signature validation less than when using explicit certificates. A standard certificate requires two signature validations: one for the signature on the certificate and another for the signature of the received message. With implicit certificates there is only one signature validation for each message. However, the receiver needs to calculate the sender's public key before performing these validations. Considering the use of the ECDSA algorithm to verify a signature, three modular operations, two elliptic curve scalar multiplication operations and one point addition operation are needed. Using the ECQV scheme, only one elliptic curve scalar multiplication operation and one point addition operation are needed. Therefore, the three modular operations needed to verify an ECDSA signature are not necessary to check an implicit certificate.

Regarding the complexity (bandwidth overhead, delay, computational cost, and so on) of the issuance process, we must mention that it only affects the first time that a user joins the overlay (or each time that a user must update or revoke her certificate) and that it depends on the used encryption and digital signature schemes. For this reason, we have defined a generic version of the protocol, which can be implemented using the most appropriate schemes for each situation.

Finally, we must mention that ICIAPPA ensures that all nodeIDs are uniformly distributed in the virtual space, which provides a proper load-balancing among all nodes of the overlay and thus a good performance. We achieve this requirement by using a hash function to generate the nodeIDs.

4.5 Conclusions

Many proposals in the literature use cryptographic public keys with the aim of generating robust nodeIDs. But none of those proposals is concerned about the characteristics of the cryptographic keys. Unfortunately, as we have seen in Chapter 3, the robustness of this kind of nodeIDs lies in the way that public keys are generated and issued. For this reason we have proposed ICIAPPA (An Implicit Certificate-based Identity Assignment Protocol for the P2P overlAys), a new protocol for issuing implicit certificates which provides an appropriate type of public keys to generate robust nodeIDs for a P2P overlay. Moreover, implicit certificates have a shorter length than explicit certificates; therefore they provide a more efficient alternative.

Specifically, on the one hand, public key certificates provide the basic security services to P2P overlays: authentication, data confidentiality, data integrity, non-repudiation and access control. On the other hand, the particular use of implicit certificates guarantees the random generation of the public keys without trusting in third parties. And finally, these keys are used to obtain very robust nodeIDs (unique, random, uniformly distributed, verifiable and revocable).

Regardless of the type of the used certificates, ICIAPPA can also provide traceability of malicious users. This feature depends on the access control carried out by the TTP, which can be done using users' real-world certificates. Although this measure only allows partial anonymity of users. On the other hand, this protocol cannot provide stability of nodeIDs, since every time a certificate is revoked or renewed its associated public key changes; therefore the nodeID of the owner will also change. This feature is provided in another of our proposals, more specifically in RIAPPA (Chapter 6).

Finally, regarding the performance implications, we can conclude that the cost of ICIAPPA is reasonable according to our analysis. It is very important to take into account that a user only executes this protocol the first time she joins the overlay, or when her overlay certificate has been revoked or must be renewed. On the other hand, the use of digital certificates always implies the distribution of revocation data, which can be a challenging task in this kind of networks. For this reason, this issue is specifically addressed in Chapter 7.

Chapter 5

A Two-level Identity Assignment Protocol for P2P overlAys

Contents

In this chapter, we propose a Two-level Implicit Certificate-based Identity Assignment Protocol for the P2P overlAys (TIAPPA). Also leveraging the issuance of implicit certificates, authenticating newcomers, using two TTPs and a blind signature scheme, TIAPPA provides traceability, full anonymity[1] and a secure and efficient way of assigning nodeIDs. In this case, each nodeID is also computed as a digest of the user's public key. However, unlike ICIAPPA (Section 4), each public key is generated under the

[1]We consider full users anonymity when neither the TTPs nor the other users can bind the real-world identity of a user to her nodeID by themselves.

supervision and participation of two TTPs using a new modified version of the ECQV implicit certificate scheme.

In the same way that in ICIAPPA, ECQV has also been modified to ensure that neither the TTPs nor the newcomers has the ability to choose the value of the resulting public key; it includes the commitment scheme explained in Section 4.1. In addition, TIAPPA uses two collaborating TTPs: an *internal* TTP that assigns and manages the nodeIDs, and an *external TTP* that authenticates and manages the real-world identities of newcomers. Thanks to the use of these two collaborating TTPs and a blind signature scheme, users remain anonymous to other users and to the internal TTP, and the external TTP is not able to bind the real-world identity of a user to her nodeID. To do so, the external TTP takes part in the certificates issuance process without knowing the information they contain.

In addition, the TTPs share a Link Number (LN) for each newcomer who obtains an overlay certificate. This LN binds the real-world identity of the users to their nodeIDs. Thus, the anonymity is compatible with a robust protection against identity-based attacks. In this sense, TIAPPA can work in a similar way to an *anonymous blacklisting system* (also called anonymous revocation system) [62], because the TTPs have the ability to revoke access from dishonest/abusive anonymous users without revealing their real-world identities. But in the case that a malicious user commits a serious offense or illegal action within the network, both TTPs can de-anonymize the identity of this user. We are aware that providing anonymity and traceability at the same time may seem conflicting, but notice that the cases in which we pretend to use this traceability feature are those which require starting a legal investigation. In these cases, TIAPPA can also work like a *revocable anonymity system* [76] if necessary, and fully de-anonymize malicious users committing illegal actions. Even so these nodeIDs are easy to verify by other users and meet almost all security requirements discussed in Section 3.2.

5.1 Assumptions and Clarifications

We assume that users trust in both TTPs, but this trustworthiness relationship is only for specific aspects related to the identity management and anonymity. TTPs are only responsible for cooperatively issuing the implicit certificates. In addition, we assume that these TTPs will never collude to de-anonymize a user.

Regarding the cryptographic aspects of the protocol, in TIAPPA we also use encryption (both symmetric and asymmetric) to provide confidentiality and sometimes to bind different parts of a message, and digital signatures to ensure both message integrity and authentication. In addition, we also include the identity of the involved parts in a communication within all the exchanged messages to avoid the classical security protocol vulnerabilities. However, this only occurs when we use asymmetric cryptography, since when we use symmetric encryption we assume that the involved parties are already authenticated because the symmetric key is only known by them. Obviously, we assume that the TTPs share a symmetric cryptographic key.

To preserve the users' anonymity within the P2P overlay we have adapted a blind signature protocol based on ECDLP [112] to our implicit certificate issuance scheme. Figure 5.1 shows the original scheme where the signer S must sign the message m provided by the requester R. First of all, S selects a random number $u \in [1, p-1]$ and provides R with its associated elliptic curve point $U = uG$. Next, R selects two random numbers $v,\ w \in [1, p-1]$; calculates $T = vU + wG$, $e = H(T||m)$ and $e' = e/v$, where $H()$ is a hash function and $||$ means concatenation; and sends e' to S. Then, S generates the blind signature $s' = e'd_S + u$ using her private key and sends it to R. Finally, R unblinds the S's signature ($s = s'v + w$) and verifies the result ($sG == eQ_S + T$) using the public key of S. Now the S's signature on the message m is the pair (s, T).

But the use of a double signature on an implicit certificate modifies its usual format. In this case, the implicit certificate that identifies a node and provides the information needed to generate its public key is composed of three parameters (not two as the ECQV implicit certificate version explained in Section 2.2.2 and used in ICIAPPA). These three parameters are the information contained in the certificate (I) and two public reconstruction parameters (Z and B).

Regarding other protocol details, we also use timestamps to fulfill two purposes; timeliness proofs and request identifiers. In the first case, we guarantee the freshness of the requests to minimize the impact of a replay attack. Although, as we explained in ICIAPPA, this measure requires a certain synchronism between the clocks of all involved parties. The second purpose is to provide a value used by all the involved entities to unequivocally identify a certain request. More specifically, TIAPPA uses two different timestamps for each of the certificate requests. One of them (t_X) is used in all exchanged messages during the entire process and the other one (t_{XO}) is only

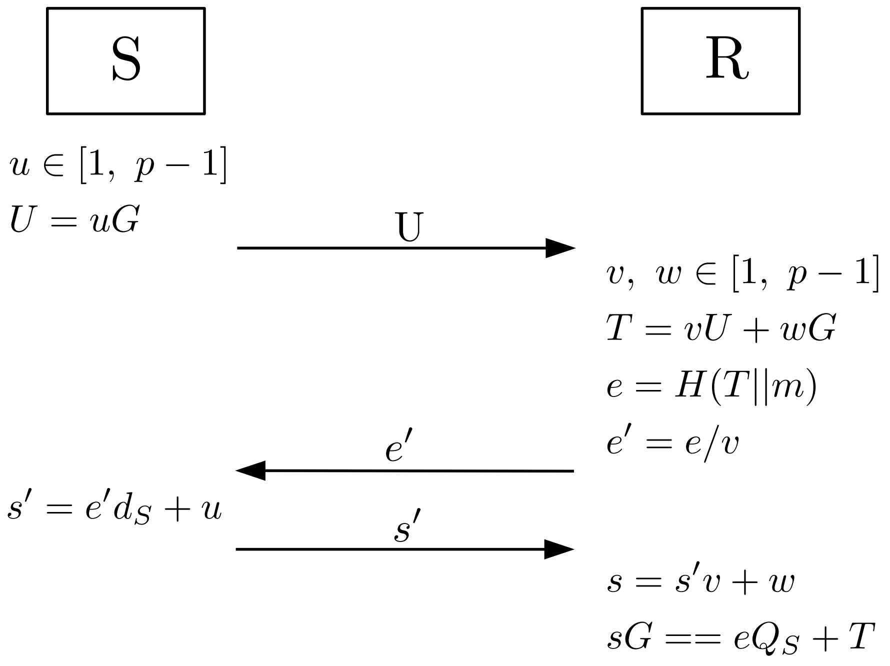


Figure 5.1: Original blind signature scheme

used within the messages exchanged between the newcomer and the internal TTP via the external TTP. Fortunately, the Network Time Protocol (NTP) [89] may be a good solution for these purposes. It is also noteworthy that these two entities use a session key pair to secure their communications via the external TTP, since the newcomer does not have her cryptographic key pair to be used within the overlay until the implicit certificate is issued. Finally, in the case that there is any kind of error during the transactions (due to lack of freshness, invalid signatures, corruption of messages, etc.), an error message is immediately sent to cancel the request process.

5.2 Protocol Specification

In this Section we describe TIAPPA: what information is exchanged between the three parties (Newcomer and TTPs), how data is exchanged, what security mechanisms are used and so on. Figure 5.2 shows the essential data (without several details) exchanged in this protocol.

By way of summary, a newcomer (X) must always have a digital certificate for the real-world (C_X) which contains a public key (Q_X). This certificate can be issued by any

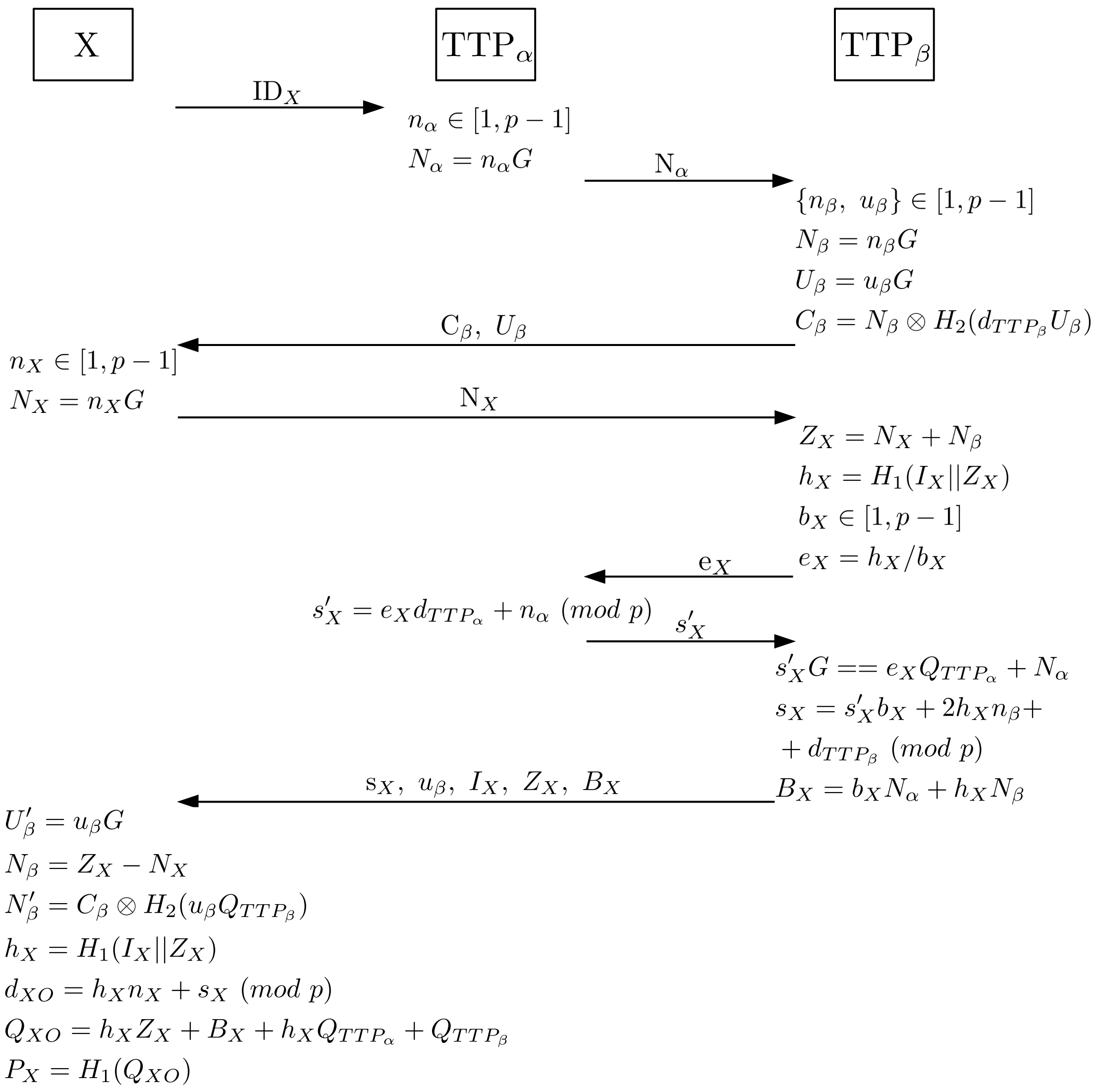

X
TTP$_\alpha$
TTP$_\beta$
ID$_X$
$n_\alpha \in [1, p-1]$
$N_\alpha = n_\alpha G$
N$_\alpha$
$\{n_\beta,\ u_\beta\} \in [1, p-1]$
$N_\beta = n_\beta G$
$U_\beta = u_\beta G$
$C_\beta = N_\beta \otimes H_2(d_{TTP_\beta} U_\beta)$
C$_\beta$, U_β
$n_X \in [1, p-1]$
$N_X = n_X G$
N$_X$
$Z_X = N_X + N_\beta$
$h_X = H_1(I_X || Z_X)$
$b_X \in [1, p-1]$
$e_X = h_X / b_X$
e$_X$
$s'_X = e_X d_{TTP_\alpha} + n_\alpha\ (mod\ p)$
s'_X
$s'_X G == e_X Q_{TTP_\alpha} + N_\alpha$
$s_X = s'_X b_X + 2h_X n_\beta +$
$+ d_{TTP_\beta}\ (mod\ p)$
s$_X$, u_β, I_X, Z_X, B_X
$B_X = b_X N_\alpha + h_X N_\beta$
$U'_\beta = u_\beta G$
$N_\beta = Z_X - N_X$
$N'_\beta = C_\beta \otimes H_2(u_\beta Q_{TTP_\beta})$
$h_X = H_1(I_X || Z_X)$
$d_{XO} = h_X n_X + s_X\ (mod\ p)$
$Q_{XO} = h_X Z_X + B_X + h_X Q_{TTP_\alpha} + Q_{TTP_\beta}$
$P_X = H_1(Q_{XO})$

Figure 5.2: Certificate/nodeID generation scheme

CA and must allow the user to be authenticated by the external TTP (TTP_α). Then, the internal TTP (TTP_β) generates an implicit certificate for the user using a public parameter sent by her and the blind signature of TTP_β. And finally, TTP_β issues the new certificate of X (C_{XO}). At the end of the description, Figure 5.3 shows the message exchange between the three parties.

Regarding the notation used along the protocol description, we use a similar notation to that used in ICIAPPA but more extensive due to the complexity of this protocol. Table 5.1 shows a global summary of this notation.

Parameter	Legend
p	The order of the underlying finite field $\mathbb{F}_p$.
G	The generator of the elliptic curve defined over $\mathbb{F}_p$ ($E(\mathbb{F}_p)$).
$TTP_\alpha,\ TTP_\beta$	The external and internal TTPs respectively.
$ID_{TTP_\alpha},\ ID_{TTP_\beta}$	The identity of TTP_α and TTP_β respectively.
ID_X	The real-world identity of the newcomer.
P_X	The nodeID of the newcomer.
I_X	The information that is included in X's certificate.
$C_{TTP_\alpha},\ C_{TTP_\beta}$	The digital certificates of the TTPs.
C_X	The real-world certificate of the newcomer.
C_{XO}	The new overlay certificate of the newcomer (I_X, Z_X, B_X).
$d_{TTP_\alpha},\ d_{TTP_\beta}$	The ECC private key of the TTPs.
$Q_{TTP_\alpha},\ Q_{TTP_\beta}$	The ECC public key of the TTPs.
$K_{\alpha\beta}$	The symmetric key shared between the two TTPs.
Q_X	The real-world public key of the newcomer.
d_X	The real-world private key of the newcomer.
Q_{XS}	The session public key of the newcomer.
d_{XS}	The session private key of the newcomer.
Q_{XO}	The new overlay public key of the newcomer.
d_{XO}	The new overlay private key of the newcomer.
s'_X	The blind signature of TTP_α on the X's certificate.
s_X	The double signature of the TTPs on the X's certificate.
$Z_X,\ B_X$	The X's reconstruction public parameters.
h_X	The digest of part of the new certificate $(I_X \| Z_X)$.
e_X	The value of the digest h_X blinded.
$n_X,\ n_\alpha,\ n_\beta, u_\beta$	Private parameters of the X's request ($\in [1,\ p-1]$).
$N_X,\ N_\alpha,\ N_\beta,\ U_\beta$	Public parameters of the X's request ($\in E(\mathbb{F}_p)$).
b_X	The blinding factor.
C_β	The commitment value.
$H_1(m)$	A secure hash function on a message m.
$H_2(P)$	A secure hash function on a point P which also outputs a point.
$i \to j:$	The sending of a message from the entity i to the entity j.
$\{m\}_K$	The ciphertext of a message m encrypted using the symmetric key K.
$\{m\}_Q$	The ciphertext of a message m encrypted using the public key Q.
$\{m\}_d$	The signature on a message m using the private key d.
t_X	The timestamp also used as request identifier.
t_{XO}	The timestamp also used as newcomer identifier with TTP_β.
S_{ID}	The identifier of a service.

Table 5.1: Notation for TIAPPA.

5.2.1 Protocol Steps

Step 1:

In the first step, the newcomer X contacts the external TTP (TTP_α) to start the message exchange. First of all, X generates a timestamp (t_X), which will serve as a request identifier along the protocol. Then she sends that timestamp together with the identifier of the service she wants to access (S_{ID}) and her real-world certificate (C_X) to TTP_α. Always including the sender's and receiver's identities, and signing and encrypting the message. She signs the $HELLO\ MESSAGE$ using her private key in the real-world (d_X) and encrypts it using the public key of TTP_α (Q_{TTP_α}).

$$HELLO\ MESSAGE,\ X \rightarrow TTP_\alpha :\ \{\{ID_X, ID_{TTP_\alpha}, t_X, S_{ID}, C_X\}_{d_X}\}_{Q_{TTP_\alpha}}$$

The X's certificate allows TTP_α to authenticate the sender and the message, and X encrypts the entire message to provide confidentiality. Note that the request identifier (t_X) must always be encrypted to avoid other users can use the same t_X for malicious purposes. The service identifier (S_{ID}) and the X's certificate are encrypted to preserve the user's privacy. It is also noteworthy that the inclusion of the identities in the signed message is necessary to avoid that a malicious user can forward this message to another recipient posing as X [21]. Therefore, from here on we will include the identities of the sender and the receiver in all messages.

Step 2:

In the second step, after the decryption of the $HELLO\ MESSAGE$ and the verification of its signature, TTP_α authorizes X to acquire an identity for the requested P2P overlay. Therefore, it will answer to her with the certificate of the internal TTP responsible for that service (TTP_β). Otherwise, X would be expelled from the identity acquisition process. But before that, TTP_α stores X's identity (ID_X), her real-world certificate (C_X), the service identifier (S_{ID}), the request identifier (t_X) and the identity of TTP_β, to manage X's request. Regarding the answer to X, it sends a signed and encrypted message which includes the certificate of TTP_β (C_{TTP_β}), t_X and the involved identities. Obviously, the $ACCEPT\ MESSAGE$ is signed using its private key and encrypted using the public key of X in the real-world.

$$ACCEPT\ MESSAGE,\ TTP_\alpha \rightarrow X:\ \{\{ID_{TTP_\alpha}, ID_X, t_X, C_{TTP_\beta}\}_{d_{TTP_\alpha}}\}_{Q_X}$$

Note that the freshness of the timestamp t_X must be checked in all steps, and if it is out of time the identity request must be automatically canceled.

Step 3:

In this step, X contacts TTP_β via TTP_α to obtain a valid implicit certificate for the overlay. But before that, she generates a session key pair (d_{XS},Q_{XS}) and another timestamp (t_{XO}), which will also be used as a newcomer's identifier, and stores the certificate of TTP_β (C_{TTP_β}) and its identity (ID_{TTP_β}). Then, X sends the generated public key (Q_{XS}) and t_{XO} to TTP_β, all protected to prevent TTP_α can access to that information. To do so, she generates a signed and encrypted message ($A1$) using her new session key pair and the public key of TTP_β, which contains Q_{XS}, t_{XO} and the identity of the internal TTP. Finally, X sends the $A1$ message within the $REQUEST\ MESSAGE$ 1 to TTP_α. This message also contains t_X and the involved identities, and it is also signed and encrypted using the public key of TTP_α.

$$REQUEST\ MESSAGE\ 1,\ X \rightarrow TTP_\alpha:\ \{\{ID_X, ID_{TTP_\alpha}, t_X, A1\}_{d_X}\}_{Q_{TTP_\alpha}}$$

where $A1 = \{\{ID_{TTP_\beta}, Q_{XS}, t_{XO}\}_{d_{XS}}\}_{Q_{TTP_\beta}}$.

Note that the message $A1$ can only be decrypted by TTP_β although X cannot communicate directly with it since she still does not have an overlay certificate. Thus, the anonymity of the user within the overlay network is preserved.

Step 4:

Once TTP_α receives the $REQUEST\ MESSAGE$ 1, it forwards the message $A1$ to TTP_β. But before that, TTP_α generates a private random number ($n_\alpha \in [1,\ p-1]$) and its associated public point (N_α), which will be used to blind sign the new certificate. Then, it sends the message $A1$ together with N_α, t_X and the involved identities to TTP_β, all encrypted using the symmetric key preshared with TTP_β ($K_{\alpha\beta}$).

$$REQUEST\ MESSAGE\ 2,\ TTP_\alpha \rightarrow TTP_\beta : \{ID_{TTP_\alpha}, ID_{TTP_\beta}, t_X, N_\alpha, A1\}_{K_{\alpha\beta}}$$

Note that now may not be necessary to include the sender and the receiver identities in the message, since the symmetric key is only shared by these two entities, but however we include them to avoid a possible replay attack, as we have been able to check using the AVISPA tool. The problem is that in this protocol we send two pairs of message with the same number of parameters in opposite directions ($REQUEST\ MESSAGE$ 2-$SIGN\ REQUEST\ MESSAGE$ for instance) and the TTPs are not able to distinguish between the two messages of a pair. In this way an attacker can replay the first messages ($REQUEST\ MESSAGE$ 2 and/or $SIGN\ RESPONSE\ MESSAGE$) in order to alter the identity acquisition process.

Step 5:

In this step, TTP_β receives the encrypted message $A1$ and starts the certificate generation process. Specifically, first of all, it generates two random private parameters ($\{n_\beta,\ u_\beta\} \in [1,\ p-1]$), the two associated elliptic curve points ($N_\beta = n_\beta G$ and $U_\beta = u_\beta G$) and calculates the commitment value as $C_\beta = N_\beta \otimes H_2(d_{TTP_\beta} U_\beta)$. Then, TTP_β generates the signed and encrypted message $A2$ with C_β, U_β, the timestamp t_{XO} and the involved identities. It signs the message using her private key (d_{TTP_β}) and encrypts it using the received newcomer's public key Q_{XS}. Finally, TTP_β sends the message $A2$ together with t_X and the involved identities to TTP_α using the symmetric key $K_{\alpha\beta}$.

$$COMMIT\ MESSAGE\ 1,\ TTP_\beta \rightarrow TTP_\alpha : \{ID_{TTP_\beta}, ID_{TTP_\alpha}, t_X, H(N_\alpha), A2\}_{K_{\alpha\beta}}$$

where $A2 = \{\{ID_{TTP_\beta}, t_{XO}, C_\beta, U_\beta\}_{d_{TTP_\beta}}\}_{Q_{XS}}$.

Note that this is the first time that TTP_α contacts TTP_β for this request, therefore it also stores the two request identifiers/timestamps (t_X and t_{XO}), the session public key of the newcomer (Q_{XO}), and the point generated by TTP_α (N_α). It is also noteworthy that $H(N_\alpha)$ is included in this message to avoid replay attacks in the future using this message as others explained below ($ISSUE\ MESSAGE$ 1 or $LINK\ MESSAGE$).

Step 6:

Once TTP_α receives the $COMMIT\ MESSAGE$ 1, it forwards the message $A2$ to the newcomer X. More specifically, it signs the message $A2$ together with t_X and the involved identities, encrypts the signed message using the public key of X in the real-world and sends the result to X.

$$COMMIT\ MESSAGE\ 2,\ TTP_\alpha \rightarrow X : \{\{ID_{TTP_\alpha}, ID_X, t_X, A2\}_{d_{TTP_\alpha}}\}_{Q_X}$$

Step 7:

In this step, X receives from TTP_α the TTP_β's commit to value n_β, which is not yet revealed, and generates a random private parameters ($n_X \in [1,\ p-1]$) and its related public point ($N_X = n_X G$) to take part in the certificate generation process. Then, X generates the message $A3$ in order to provide TTP_β with the public point of that commit value (N_X). This message contains the timestamp t_{XO}, the identity of TTP_β and N_X, and is signed by X using her session public key and encrypted using the public key of TTP_β. Finally, the signed and encrypted message $A3$ is sent by X to TTP_α together with the timestamp t_X and the involved identities, all signed and encrypted using the public key of TTP_α.

$$INVOLVEMENT\ MESSAGE\ 1,\ X \rightarrow TTP_\alpha :\ \{\{ID_X, ID_{TTP_\alpha}, t_X, A3\}_{d_X}\}_{Q_{TTP_\alpha}}$$

where $A3 = \{\{t_{XO}, ID_{TTP_\beta}, N_X\}_{d_{XS}}\}_{Q_{TTP_\beta}}$.

Step 8:

Once TTP_α receives the $INVOLVEMENT\ MESSAGE$ 1, it forwards the message $A3$ to TTP_β. More specifically, it sends the message $A3$ together with t_X and the involved identities, all encrypted using the symmetric key preshared with TTP_β.

$$INVOLVEMENT\ MESSAGE\ 2,\ TTP_\alpha \rightarrow TTP_\beta : \{ID_{TTP_\alpha}, ID_{TTP_\beta}, t_X, A3\}_{K_{\alpha\beta}}$$

Step 9:

At this point, TTP_β has already received the two points generated by the newcomer X and TTP_α, N_X and N_α respectively, and it is ready to generate the new implicit certificate. It calculates the reconstruction public parameter (1), the hash value of this parameter concatenated with the certificate information (2), and blinds the certificate using a blind number ($b_X \in [1,\ p-1]$) so that TTP_α can sign the certificate without seeing the content (3).

1. $Z_X = N_X + N_\beta$
2. $h_X = H_1(I_X||Z_X)$
3. $e_X = h_X/b_X$

Finally, TTP_β sends e_X to TTP_α together with t_X, the digest of the point generated by TTP_α ($H_1(N_\alpha)$) and the involved identities, all encrypted using the symmetric key preshared between the two TTPs.

$$SIGN\ REQUEST\ MESSAGE,\ TTP_\beta \rightarrow TTP_\alpha :$$
$$\{ID_{TTP_\beta}, ID_{TTP_\alpha}, t_X, e_X, H_1(N_\alpha)\}_{K_{\alpha\beta}}$$

Note that $H_1(N_\alpha)$ is included in this message to avoid future replay attacks using this message as others explained below ($ISSUE\ MESSAGE$ 1 or $LINK\ MESSAGE$).

Step 10:

In this step, once TTP_α has received the $SIGN\ REQUEST\ MESSAGE$, first of all it decrypts that message and verifies the digest of N_α. If all is correct, then it signs the blinded certificate (e_X) using her private key (d_{TTP_α}) and the previously generated private number n_α. Finally, TTP_α sends the signature (s'_X) to TTP_β together with the timestamp t_X, the digest of the blinded certificate ($H_1(e_X)$) and the involved identities, all signed and encrypted using the symmetric key $K_{\alpha\beta}$. Note that the signature is calculated as $s'_X = e_X d_{TTP_\alpha} + n_\alpha\ (mod\ p)$.

$$SIGN\ RESPONSE\ MESSAGE,\ TTP_\alpha \rightarrow TTP_\beta : \{ID_{TTP_\alpha}, ID_{TTP_\beta}, t_X, H_1(e_X), s'_X\}_{K_{\alpha\beta}}$$

Note that TTP_α uses its common public key to blind sign the new certificate thanks to the use of ECC in all processes. On the other hand, $H_1(e_X)$ is included in this message to avoid future replay attacks using this message as others explained below (e.g. *LINK ACK MESSAGE*).

Step 11:

Once TTP_β has received the blind signature of TTP_α, it must verify the signature, remove the blindness on the blinded certificate and generate the double signature on it. To do so, TTP_β takes two operations:

1. $s'_X G == e_X Q_{TTP_\alpha} + N_\alpha$ (verification).
2. $s_X = s'_X b_X + 2h_X n_\beta + d_{TTP_\beta}\ (mod\ p)$ (unblindness and signing).

In addition, it also generates another point ($B_X = b_X N_\alpha + h_X N_\beta$) that will be used by users to generate the related public key of that certificate. So these new implicit certificates consist of three parameters (I_X, Z_X and B_X), instead of two as the used in ICIAPPA. Once the new certificate is generated, TTP_β calculates the new public key and the nodeID of X:

1. $Q_{XO} = h_X Z_X + B_X + h_X Q_{TTP_\alpha} + Q_{TTP_\beta}$
2. $P_X = H_1(Q_{XO})$

Then, TTP_β sends the double signature (s_X), the random number used in the commitment phase (u_β) and the new implicit certificate C_{XO} (I_X, Z_X and B_X), together with the timestamp t_{XO} and the involved identities to X via TTP_α, all signed and encrypted using the session public key of X (message $A4$). Finally, TTP_β encrypts the message $A4$ together with t_X and the involved identities using the symmetric key $K_{\alpha\beta}$, and sends it to TTP_α.

$$ISSUE\ MESSAGE\ 1,\ TTP_\beta \rightarrow TTP_\alpha:\ \{ID_{TTP_\beta}, ID_{TTP_\alpha}, t_X, A4\}_{K_{\alpha\beta}}$$

where $A4 = \{\{ID_{TTP_\beta}, P_X, t_{XO}, s_X, u_\beta, I_X, Z_X, B_X\}_{d_{TTP_\beta}}\}_{Q_{XS}}$.
Note that P_X is already included in the new message $A4$.

Step 12:

Once TTP_α receives the $ISSUE\ MESSAGE\ 1$, it forwards the message $A4$ to the newcomer X. More specifically, it signs the message $A4$ together with t_X and the involved identities, encrypts the signed message using the public key of X in the real-world and sends the result to X.

$$ISSUE\ MESSAGE\ 2,\ TTP_\alpha \rightarrow X:\ \{\{ID_{TTP_\alpha}, ID_X, t_X, A4\}d_{TTP_\alpha}\}_{Q_X}$$

Step 13:

In this step, X contacts TTP_β directly, for the first time, to confirm that she has received her new implicit certificate and everything is correct. But before that, she must prove that TTP_β used the committed value to generate the certificate and generate her new private key (d_{XO}) and nodeID (P_X). To do so, X performs the following operations:

1. $U'_\beta = u_\beta G$
2. Compares U_β with U'_β.
3. $N_\beta = Z_X - N_X$
4. $N'_\beta = C_\beta \otimes H_2(u_\beta Q_{TTP_\beta})$
5. Compares N_β with N'_β
6. $h_X = H_1(I_X || Z_X)$
7. $d_{XO} = h_X n_X + s_X\ (mod\ p)$
8. $Q_{XO} = h_X Z_X + B_X + h_X Q_{TTP_\alpha} + Q_{TTP_\beta}$
9. Compares Q_{XO} with $d_{XO}G$.
10. $P_X = H_1(Q_{XO})$

Finally, if the three previous comparisons have been successful, she sends h_X together with t_X and the involved identities to TTP_β, all signed and encrypted using its public key.

$$ISSUE\ ACK\ MESSAGE,\ X \rightarrow TTP_\beta:\ \{\{P_X, ID_{TTP_\beta}, t_X, h_X\}_{d_{XO}}\}_{Q_{TTP_\beta}}$$

Note that the point 9 is accomplished because $d_{XO}G = h_Xn_XG + s_XG = h_Xn_XG + s'_Xb_XG + 2h_Xn_\beta G + d_{TTP_\beta}G = h_Xn_XG + e_Xb_Xd_{TTP_\alpha}G + b_Xn_\alpha G + 2h_Xn_\beta G + d_{TTP_\beta}G = h_XN_X + h_XQ_{TTP_\alpha} + b_XN_\alpha + 2h_XN_\beta + Q_{TTP_\beta} = h_X(N_X + N_\beta) + b_XN_\alpha + h_XN_\beta + h_XQ_{TTP_\alpha} + Q_{TTP_\beta} = h_XZ_X + B_X + h_XQ_{TTP_\alpha} + Q_{TTP_\beta} = Q_{XO}$.

It is also noteworthy that now X already uses her new private key (d_{XO}) to sign the message.

Step 14:

Once TTP_β knows that X has received her new certificate correctly, it generates a Link Number (LN) to her, and adds this LN to the user list together with her nodeID (P_X). Then, TTP_β sends LN together with t_X and the involved identities to TTP_α, all encrypted using the symmetric key $K_{\alpha\beta}$

$$LINK\ MESSAGE,\ TTP_\beta \rightarrow TTP_\alpha:\ \{ID_{TTP_\beta}, ID_{TTP_\alpha}, t_X, LN\}_{K_{\alpha\beta}}$$

In the case that TTP_β does not receive the $ISSUE\ ACK\ MESSAGE$ after a certain time, it will revoke that certificate to avoid that the user X joins the network without finishing the request process correctly.

Step 15:

In this step, TTP_α adds ID_X to the user list with the received LN and confirms to TTP_β that it has received the LN correctly. More specifically, TTP_α calculates the digest of LN, encrypts this value together with t_X and the involved identities using the symmetric key $K_{\alpha\beta}$, and sends the encrypted message to TTP_β.

$$LINK\ ACK\ MESSAGE,\ TTP_\alpha \rightarrow TTP_\beta:\ \{ID_{TTP_\alpha}, ID_{TTP_\beta}, t_X, H_1(LN)\}_{K_{\alpha\beta}}$$

Note that LN is shared between the TTPs allowing that the real-world identity of X is related to her nodeID.

Step 16:

In the last step, TTP_β receives the $LINK\ ACK\ MESSAGE$, verifies the digest of LN and ends the request process.

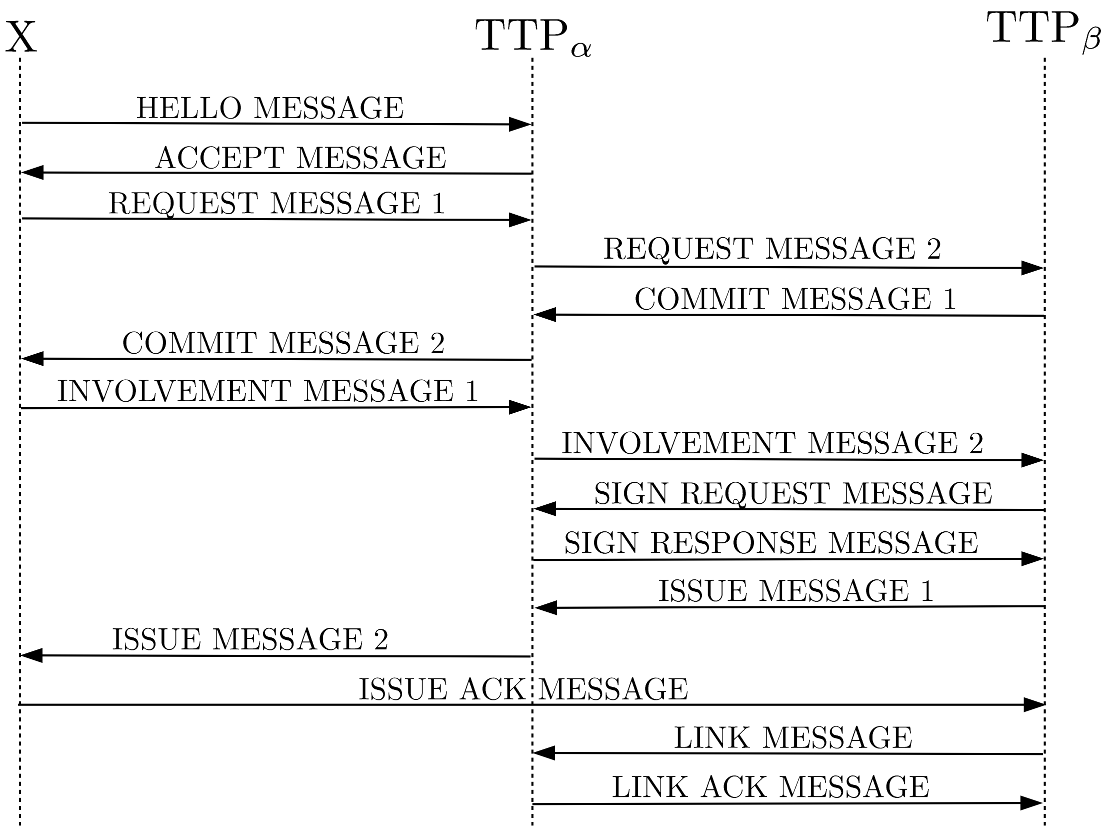


Figure 5.3: Message exchange

5.2.2 Public Key Generation

Every time a user wants to authenticate the sender of a certain message and/or verify its signature, she needs to generate the sender's public key. To do so, she uses the sender's implicit certificate, which includes the certificate's information (I_X) together with the two reconstruction public parameters (Z_X and B_X), and follows the next steps (X is considered the sender):

1. Calculates the parameter $h_X = H_1(I_X||Z_X)$.

2. Generates the sender's public key as $Q_{XO} = h_X Z_X + B_X + h_X Q_{TTP_\alpha} + Q_{TTP_\beta}$.

3. Verifies the message signature using Q_{XO}.

Note that this is accomplished because:
$Q_{XO} = d_{XO}G = h_X n_X G + s_X G = h_X n_X G + s'_X b_X G + 2h_X n_\beta G + d_{TTP_\beta} G = h_X n_X G + e_X b_X d_{TTP_\alpha} G + b_X n_\alpha G + 2h_X n_\beta G + d_{TTP_\beta} G = h_X N_X + h_X Q_{TTP_\alpha} + b_X N_\alpha + 2h_X N_\beta + Q_{TTP_\beta} = h_X(N_X + N_\beta) + b_X N_\alpha + h_X N_\beta + h_X Q_{TTP_\alpha} + Q_{TTP_\beta} = h_X Z_X + B_X + h_X Q_{TTP_\alpha} + Q_{TTP_\beta}$.

5.2.3 NodeID Validation

Every time a user receives data from another user X (overlay contents or routing info) she should validate the sender's identity (nodeID). To do so, the receiver only needs to compute the digest of the sender's public key ($P_X = H(Q_{XO})$) and compare the result with the used nodeID. Note that she must previously generate the X's public key.

5.3 Security Analysis

TIAPPA provides a mechanism for issuing robust and traceable identities in a P2P overlay while the full anonymity of the users is preserved. For this, two TTPs share a Link Number (LN) which relates the real-world identity of users and their identities within the overlay. Obviously, this is a security weakness that we must assume to ensure both features at the same time (traceability and full anonymity). Next, we analyze the behavior of this protocol against some of the most common attacks in a P2P overlay.

Sybil attack. Taking into account that the access control to request a new overlay certificate is identical to that used in ICIAPPA, we can also affirm that the Sybil attack is not a threat when RIAPPA is deployed in a P2P overlay.

Eclipse attack. To avoid an eclipse attack it is imperative to avoid (or limit) the Sybil attack and not to allow attackers to place themselves close to a target node. Therefore, we can affirm that TIAPPA avoids the Eclipse attack because it fulfills this two requirements; nodeIDs are randomly generated and users can only obtain one (or a few) overlay certificate.

MITM attack. In the same way that in ICIAPPA, during the bootstrapping phase it is theoretically impossible for an attacker to disrupt the normal behavior of the protocol, since all messages are signed and encrypted using public key cryptography or protected using the preshared symmetric cryptographic key between the TTPs. In addition, during the normal working of the P2P overlay all nodes will sign all the messages they will send. Therefore, an attacker should compromise the symmetric key preshared by the TPPs or the private/public keys pair of the TTPs or some user in order to impersonate someone. Obviously, the importance of the attack would be not the same if the compromised keys belong to a user or a TTP.

Whitewashers. In the same way that ICIAPPA, TIAPPA tries to limit the number of nodeIDs which can be managed by a user binding their real-world identities with their nodeIDs, but the stability of those nodeIDs is not guaranteed. Every time an overlay certificate is renewed the user's nodeID changes. Therefore, whitewashers can take advantage of this feature but the external TTP can monitor their changes of nodeIDs.

5.3.1 Cryptographic Analysis

As we have already explained in depth in ICIAPPA (Section 4.3.1), each of the used cryptographic tools has a range of security levels (measured in bits), therefore we must select them taking into account that their security level must meet or exceed the desired security level of the whole system. In addition, the selected RNGs must fulfill the ANSI X9.82 [26] or the NIST Special Publication 800-90A [94] to ensure proper implementation; the hash functions that computes the users' nodeIDs and all the necessary digests must be resistant to collisions; regarding the used elliptic curves, its domain parameters (p, a, b, G, n and h) must be generated and validated following the guidelines published in [101]; and the proposed commitment scheme must accomplish the two main security properties that are required in this kind of protocols: *Binding* and *Hiding*.

On the other hand, using implicit certificates, the fact of generating a public key (Q_{XO}) once you get a certificate is not a sufficient proof that the key is authentic. For this reason the owner (X) needs to demonstrate the knowledge of the private key (d_{XO}) associated to that certificate using a signature algorithm, such as ECDSA. In this scheme, d_{XO} is generated using the TTPs' signature (s_X), and this signature is calculated using the TTPs' private keys (d_{TTP_α} and d_{TTP_β}) together with the private parameter selected by the TTP_α (n_α) and the two private parameters selected by TTP_β (n_β and b_X). Q_{XO} is generated by the other users of the network using the TTPs' public keys (Q_{TTP_α} and Q_{TTP_β}) and the reconstruction public parameters (Z_X and B_X) which are included in the user's certificate. Thus, if a user validates a signature using Q_{XO}, she can be sure that the correct TTPs' public keys have been used in generating that key. Otherwise, the signature validation will fail. The same occurs if a wrong Z_X or B_X are used.

Note that in the case that a malicious user wants to find the private parameters selected by the TTPs (n_α, n_β and b_X) to select the value of her new public key (Q_{XO}), she should also solve the ECDLP, which is computationally unfeasible.

5.3.2 Discussion of NodeID Requirements

In this Section, we discuss how most of the described requirements in Section 3.2 are satisfied by the nodeIDs obtained using TIAPPA. Thanks to the binding between the users' real-world identities and their nodeIDs, each user can only obtain a limited number of nodeIDs. In particular, we can create a ***unique*** nodeID if, for instance, the user's national identity card is used. Regarding the nodeID generation process, thanks to the use of the public keys generated during the issuance of explicit certificates, we can consider that they are ***jointly generated*** between the users and the TTPs. In this way, neither the newcomer can choose her location within the virtual space unilaterally nor the internal TTP can place a node at a certain position seeking its own benefit. Moreover, this procedure also guarantees that nodeIDs are randomly generated; therefore their distribution within the virtual space of the network is ***uniform*** with a high probability, which improves the load-balancing within the virtual space.

Unlike ICIAPPA, TIAPPA can provide full ***anonymity*** of users. A single entity is not able to match the real-world identity of users with their nodeIDs. This is achieved thanks to the use of two TTPs, one of them using a blind signature over the certificate, and encryption, all exchanged messages are encrypted to provide confidentiality (no information is leaked to external entities). As we previously mentioned, this is valid as long as the two TTPs do not collude. Obviously, if this happens, our system is not able to provide a total degree of anonymity. In particular, this system would be equivalent to a single-TTP system. We do not expect to be cheated by the TTPs (none of them), because they are supposed to be trusted. However, in the case of having just one dishonest TTP, this entity cannot alter the protocol to obtain self-benefit, since it depends on the other TTP to issue the overlay certificate. For this reason, we have recommended that at least one of these TTPs should be a reputed and known CA or institution managed by a government or local administration.

Regarding the requirement of ***traceability***, it may seem that is in contradiction with the anonymity requirement, but that is not true. In most P2P overlays, there are nodes that misbehave, especially when anonymity is assured. Reputation systems are widely used mechanisms to punish improper usages like distribution of fake contents or routing tables poisoning. Moreover, if those bad behaviors are repeated nodes could be even revoked and hence ejected from the overlay keeping their anonymity. However, there can be some malicious behaviors (for instance, the distribution of child pornography) that should be addressed in a different manner. For this reason TIAPPA is also prepared to de-anonymize users that commit serious offenses within the overlay (thanks to the Link Number shared between the TTPs), and so they can be prosecuted by the law.

On the other hand, all implicit certificates are ***verifiable*** by anybody. Any user can verify any nodeID using the public key of its owner derived from the certificate's information (I_X, Z_X, B_X) and the public keys of the TTPs. For this reason, nobody can generate a valid nodeID without the supervision and participation of the TTPs. In addition, implicit certificates are also ***revocable***. Any certificate can be revoked if a user is misbehaving within the overlay or the involved private key has been compromised. In this last case, the user needs to prove that she knows the private key related to the certificate to revoke it. On the other hand, there is the possibility that the TIAPPA protocol does not finish properly but the certificate has been delivered to the newcomer, in that case the issued certificate must also be revoked. Note that this situation happens when a newcomer does not send the confirmation message ($ISSUE\ ACK\ MESSAGE$) to the internal TTP reporting that she has received her new certificate.

Finally, taking into account that every time a user's certificate is revoked, a new public/private key pair must be issued, and then the related nodeID changes. Therefore, in TIAPPA nodeIDs are ***not stable***.

5.3.3 Formal Validation of the Protocol

In order to validate the proper security level of TIAPPA, we have also performed an automated validation of the message exchange using the AVISPA tool [16]. We ran the tool using OFMC, CL-AtSe and SATMC backends, and in all three cases the output was the same: SAFE.

Table 5.2 shows the messages exchanged between a newcomer X and the two TTPs represented in the Alice and Bob notation. ID_X, ID_{TTP_α} and ID_{TTP_β} are the identifiers of the user X and the TTPs respectively, and $K_{\alpha X}$, K_{TTP_α} and K_{TTP_β} are their cryptographic public keys. $K_{\alpha\beta}$ is the symmetric key shared between the two TTPs. The functions $\{m\}_K$ and $\{m\}\_inv(K)$ represent respectively the encryption and signing of the message m using the key pair K and $inv(K)$, and "." indicates concatenation.

1	X	$\rightarrow$	TTP_α	:	$\{\{ID_X.ID_{TTP_\alpha}.t_X.S_{ID}.C_X\}\_inv(K_X)\}\_K_{TTP_\alpha}$
2	X	$\leftarrow$	TTP_α	:	$\{\{ID_{TTP_\alpha}.ID_X.t_X.C_{TTP_\beta}\}\_inv(K_{TTP_\alpha})\}\_K_X$
3	X	$\rightarrow$	TTP_α	:	$\{\{ID_X.ID_{TTP_\alpha}.t_X.A1\}\_inv(K_X)\}\_K_{TTP_\alpha}$
4	TTP_α	$\rightarrow$	TTP_β	:	$\{ID_{TTP_\alpha}.ID_{TTP_\beta}.t_X.N_\alpha.A1\}\_K_{\alpha\beta}$
5	TTP_α	$\leftarrow$	TTP_β	:	$\{ID_{TTP_\beta}.ID_{TTP_\alpha}.t_X.H(N_\alpha).A2\}\_K_{\alpha\beta}$
6	X	$\leftarrow$	TTP_α	:	$\{ID_{TTP_\alpha}.ID_X.t_X.A2\}\_inv(K_{TTP_\alpha})\}\_K_X$
7	X	$\rightarrow$	TTP_α	:	$\{ID_X.ID_{TTP_\alpha}.t_X.A3\}\_inv(K_X)\}\_K_{TTP_\alpha}$
8	TTP_α	$\rightarrow$	TTP_β	:	$\{\{ID_{TTP_\alpha}.ID_{TTP_\beta}.t_X.A3\}\_K_{\alpha\beta}$
9	TTP_α	$\leftarrow$	TTP_β	:	$\{ID_{TTP_\beta}.ID_{TTP_\alpha}.t_X.e_X.H(N_\alpha)\}\_K_{\alpha\beta}$
10	TTP_α	$\rightarrow$	TTP_β	:	$\{\{ID_{TTP_\alpha}.ID_{TTP_\beta}.t_X.H(e_X).s'_X\}\_K_{\alpha\beta}$
11	TTP_α	$\leftarrow$	TTP_β	:	$\{ID_{TTP_\beta}.ID_{TTP_\alpha}.t_X.A4\}\_K_{\alpha\beta}$
12	X	$\leftarrow$	TTP_α	:	$\{ID_{TTP_\alpha}.ID_X.t_X.A4\}\_inv(K_{TTP_\alpha})\}\_K_X$
13	X	$\rightarrow$	TTP_β	:	$\{ID_X.ID_{TTP_\beta}.t_X.h_X\}\_inv(K_{XO})\}\_K_{TTP_\beta}$
14	TTP_α	$\leftarrow$	TTP_β	:	$\{ID_{TTP_\beta}.ID_{TTP_\alpha}.t_X.LN\}\_K_{\alpha\beta}$
15	TTP_α	$\rightarrow$	TTP_β	:	$\{\{ID_{TTP_\alpha}.ID_{TTP_\beta}.t_X.H(LN)\}\_K_{\alpha\beta}$

Table 5.2: Message exchange represented using the Alice and Bob notation.

In Appendix B.2 can be seen the HLPSL code of the performed simulation and the obtained outputs.

5.4 Performance Analysis

Providing security always implies a cost in terms of bandwidth, memory and processing time. Although, TIAPPA also takes advantage of the ECC and the use of implicit certificates to achieve a better level of security and a lower impact on the overlays' performance. In fact, the operation of this protocol has almost no direct impact on the overlay performance. Once a user has obtained her overlay certificate, she does not need to contact directly with the TTPs again unless her overlay certificate has expired or must be revoked. In addition, if one of the two TTPs fails, the certificate issuing service becomes unavailable but users with valid certificates can join and operate normally within the overlay without contacting the TTPs. Obviously, the use of digital certificates involves performing cryptographic operations and managing revocation data. In this context, as we have already commented in Section 4.4, we can emphasize that the implicit certificates decrease the resource consumption of the implemented security mechanisms. However, in TIAPPA, due to the overlay certificates are signed by two TTPs they include two reconstruction public parameters (Z_X and B_X), which increase the number of bytes that a sender needs to send to a receiver compared to ICIAPPA. The same goes for the processing overhead, but less so than in ICIAPPA.

Regarding to secure the communications between TTPs, this protocol has been defined to use symmetric cryptography, since this type of cryptography has lower computational cost than the public key cryptography, and both parties are automatically authenticated by sharing a symmetric key. In the case of communications between the newcomers and the TTPs we decided to use asymmetric encryption, even being more computationally expensive, because these communications are only composed of a few small messages and we think that adding an extra key exchange process is not worth.

The computational cost of TIAPPA for the TTPs and newcomers depends largely on the implemented encryption and signature schemes. For this reason, we have defined a generic version of the protocol, which can be implemented using the most appropriate schemes for each situation. On the other hand, for the TTPs, this cost also depends on the number of requests that they receive. Although in the case that this number becomes very high, both TTPs could be replicated creating two distributed access levels. Each group of TTPs should only share a synchronized database where they would store the user or node information.

Finally, we must also highlight that the nodeIDs generated using TIAPPA are uniformly distributed in the virtual space, which provides a proper load-balancing among all nodes of the overlay and thus a good performance. TIAPPA achieves this requirement by using a hash function over the randomly generated public keys to obtain the nodeIDs.

5.5 Conclusions

Following the line of ICIAPPA, but trying to guarantee the full anonymity of the users, we have proposed TIAPPA (A Two-level Identity Assignment Protocol for P2P overlAys). A protocol with the aim of turning these networks into a better platform for commercial applications. Unlike most proposals in the literature, this protocol uses two TTPs to provide revocability, anonymity and traceability of the users at the same time; requirements that seem to conflict. However, if you have to deploy a commercial service over these networks you should not treat equally all misbehaviors. In addition, in the same way that in ICIAPPA, the use of implicit certificates guarantees the random generation of the public keys without trusting in third parties. Therefore, these keys can be used to obtain very robust nodeIDs (unique, random, uniformly distributed, verifiable and revocable).

Regardless of the type of the used certificates, TIAPPA can also provide traceability of malicious users guaranteeing the full anonymity of other users. This feature depends on the access control carried out by the TTPs, which store a Link Number to bind real-world identities to overlay users. A drawback is that this protocol cannot provide stability of nodeIDs, since every time a certificate is revoked o renewed its associated public key changes, the nodeID of the owner has to change too. This feature is provided in another of our proposals, more specifically in RIAPPA (Chapter 6).

Finally, regarding the cost of using TIAPPA to issue a new implicit certificate in terms of performance, we can affirm that it is reasonable according to our analysis. TIAPPA is only used the first time a user wants to join the network, or when her certificate must be revoked or renewed. Moreover, the use of digital certificates always implies the use of revocation data, which is a challenge in these networks. For this reason, Chapter 7 addresses this issue proposing a new mechanism to distribute revocation data in a P2P overlay.

Chapter 6

A Robust Identity Assignment Protocol for P2P overlAys

Contents

In this chapter, we propose a Robust Identity Assignment Protocol for the P2P overlAys (RIAPPA) [37]. Also leveraging the use of two TTPs and authenticating newcomers, RIAPPA provides a secure way of assigning nodeIDs, full anonymity, traceability and stability. This last feature is provided in this protocol since unlike in ICIAPPA (Section 4) and TIAPPA (Section 5), in this protocol the nodeIDs are not bound to public keys.

In the same way that TIAPPA, RIAPPA uses an *internal* TTP to assign and manage the nodeIDs, an *external TTP* to authenticate and manage the real-world identities of the newcomers and a LN to bind these two types of identities. However, this protocol provides each newcomer with an explicit certificate which contains her nodeID, the associated public key and the signature of the two TTPs, among other information. To preserve the full anonymity of users within the overlay, RIAPPA also uses a blind signature scheme with the aim that the external TTP can sign certificates without knowing the information they contain (nodeID, public key, etc.).

Unlike most current systems, in which nodeIDs are selected solely by users or by the TTP, in RIAPPA nodeIDs are generated jointly by the newcomer and the internal TTP. Therefore, neither the newcomer can choose her location within the virtual space unilaterally nor the internal TTP can place a node at a certain position seeking its own benefit. In addition, RIAPPA also guarantees the stability of nodeIDs and their uniform distribution. On the one hand, the nodeIDs' stability allows using trust and reputation systems efficiently to enforce fair cooperation and punish improper usages of the network. On the other hand, if nodeIDs are uniformly distributed in the virtual space it is possible to achieve a proper load-balancing within the space.

6.1 Assumptions and Clarifications

In the same way that in ICIAPPA, we assume that users only trust in both TTPs for specific aspects related to the identity management and anonymity. In RIAPPA, TTPs are responsible for cooperatively issuing explicit certificates, but users are who choose their cryptographic key pairs. Moreover, we also assume that the TTPs will never collude to de-anonymize users.

At the end of a successful transaction, a newcomer obtains its certificate for the overlay, whose format can be seen in Figure 6.1.

The certificate contains a unique *Serial Number*; the identifier of the overlay where the certificate is valid (*Service Identifier*); the identity of the two issuers, the internal TTP (TTP_β) and the external TTP (TTP_α); the *Validity Period*[1]; the nodeID of the

[1]Two dates, the date on which the certificate validity period begins ($notBefore$) and the date on which the certificate validity period ends ($notAfter$).

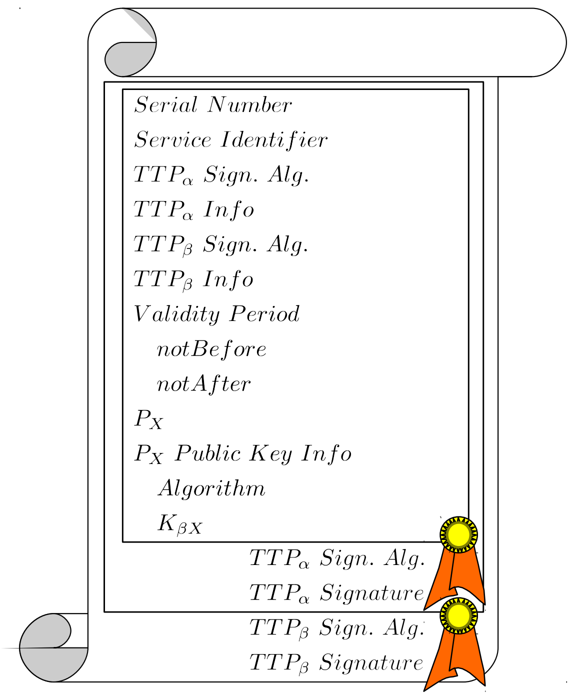


Figure 6.1: Overlay certificate format

user (P_X); the algorithm identifier for the algorithm with which the key is used (*Algorithm*); the public key of the user X ($K_{\beta X}$); and the signatures of both TTPs (TTP_α *Signature* and TTP_β *Signature*) with each algorithm identifier for the algorithms and parameters used to sign (TTP_α *Sign. Alg.* and TTP_β *Sign. Alg.*). Note that the format is similar to a X.509 standard certificate [45] but with two issuers and two signatures. Figure 6.2 shows the ASN.1 (Abstract Syntax Notation One) notation for the overlay certificate (Appendix A includes the complete definition). Note that in this notation *subject* refers to the node identity (nodeID).

The signature of the external TTP is essential to ensure the validity of a certificate although this may not seem so at first sight. The external TTP's signature certifies that the certificate belongs to a user who was previously authenticated by it and she followed the identity assignment process correctly. In this way, we avoid the internal TTP can issue valid certificates unilaterally.

Regarding the protocol description, in RIAPPA we also use encryption to provide confidentiality and bind different parts of messages, and digital signatures to ensure both message integrity and authentication. In addition, we also include the identity

```
Certificate ::= SEQUENCE {
        tbsSignedCertificate            TBSSignedCertificate,
        externalSignatureAlgorithm      AlgorithmIdentifier,
        externalSignatureValue          BIT STRING }

TBSSignedCertificate ::= SEQUENCE {
        tbsCertificate                  TBSCertificate,
        internalSignatureAlgorithm      AlgorithmIdentifier,
        internalSignatureValue          BIT STRING }

TBSCertificate ::= SEQUENCE {
        serialNumber                    INTEGER,
        serviceIdentifier               INTEGER,
        internalSignature               AlgorithmIdentifier,
        internalIssuer                  Name,
        externalSignature               AlgorithmIdentifier,
        externalIssuer                  Name,
        validityPeriod                  Validity,
        subject                         INTEGER,
        subjectPublicKeyInfo            SubjectPublicKeyInfo }
```

Figure 6.2: ASN.1 syntax of an overlay certificate

of the involved parts in a communication within all the exchanged messages encrypted and signed using asymmetric cryptography. On the other hand, in the same way as we explained in depth in ICIAPPA, we also use a timestamp in this protocol both as a prof of timeliness and as a request identifier.

Finally, in the case that there is any kind of error during the transactions (due to lack of freshness, invalid signatures, corruption of messages, etc.), an error message is immediately sent to cancel the request process.

6.2 Protocol Specification

In this Section we describe RIAPPA: what information is exchanged between the three parties (Newcomer and TTPs), how data is exchanged, what security mechanisms are used and so on. In a nutshell, a newcomer (X) must have a digital certificate for the real-world ($C_{\alpha X}$) which contains a public key ($K_{\alpha X}$). This certificate can be issued by any CA and must allow the user to be authenticated by the external TTP (TTP_α). Then, X and the internal TTP (TTP_β) jointly select the new nodeID of X. Finally, TTP_α and TTP_β sign and issue the overlay certificate of X ($C_{\beta X}$). At the end of the description, Figure 6.4 shows the message exchange between the three involved parties.

Regarding the notation used along the protocol description, we have tried to use a similar notation to that used in TIAPPA although the protocols have some important differences. Table 6.1 presents a global summary of the used notation.

Parameter	Legend
TTP_α, TTP_β	The external and internal TTPs respectively.
ID_{TTP_α}, ID_{TTP_β}	The identity of TTP_α and TTP_β respectively.
ID_X	The real-world identity of the newcomer.
P_X	The nodeID of the newcomer.
$\frac{1}{2}P_X$	The half of bits selected by the newcomer.
C_{TTP_α}, C_{TTP_β}	The digital certificates of the TTPs.
K_{TTP_α}, K_{TTP_β}	The public keys of the TTPs.
$K^{-1}_{TTP_\alpha}$, $K^{-1}_{TTP_\beta}$	The private keys of the TTPs.
$K_{\alpha\beta}$	The symmetric key shared between the TTPs.
$C_{\alpha X}$	The real-world certificate of the newcomer.
$C_{\beta X}$	The overlay certificate of the newcomer.
$K_{\alpha X}$	The real-world public key of the newcomer.
$K^{-1}_{\alpha X}$	The real-world private key of the newcomer.
$K_{\beta X}$	The overlay public key of the newcomer.
$K^{-1}_{\beta X}$	The overlay private key of the newcomer.
$i \rightarrow j :$	The sending of a message from the entity i to the entity j.
$\{m\}_K$	The ciphertext of a message m encrypted under the key K.
$\{m\}_{K^{-1}}$	The signature on a message m using the private key K^{-1}.
t_X	The timestamp generated by the newcomer.
S_{ID}	The identifier of a service.
$blind\_param$	The blinding parameters to initialize the blind signature process.
$blinded\_cert$	The blinded certificate before being signed.
$blinded\_cert\_signed$	The blinded certificate only signed by the first signer TTP.

Table 6.1: Notation for RIAPPA.

6.2.1 Protocol Steps

Step 1:

Briefly, in this step, the newcomer X contacts the external TTP (TTP_α) to start the process of issuing an overlay certificate. She sends her real-world based certificate ($C_{\alpha X}$)

to TTP_α, together with the timestamp (t_X) that will serve as a request identifier along the protocol, and the identifier of the service she wants to access (S_{ID}).

$$HELLO\ MESSAGE,\ X \rightarrow TTP_\alpha :\ \{\{ID_X, ID_{TTP_\alpha}, t_X, S_{ID}, C_{\alpha X}\}_{K^{-1}_{\alpha X}}\}_{K_{TTP_\alpha}}$$

More specifically, X performs the following operations: generates the timestamp (t_X); signs t_X and S_{ID} together with her real-world certificate ($C_{\alpha X}$), her identity and the receiver's identity using her private key in the real-world ($K^{-1}_{\alpha X}$); encrypts the signed message using the public of TTP_α (K_{TTP_α}); and finally she sends the encrypted message.

In the same way that in TIAPPA, $C_{\alpha X}$ allows TTP_α to authenticate the sender and the message, and X encrypts the message to improve the security. Note that the request identifier (t_X), the service identifier (S_{ID}) and the certificate must be encrypted to provide security and privacy. In addition, the identities of the involved parts in the communication are included in the message to avoid that a malicious user can forward this message to another recipient posing as X [21]. Therefore, from here on we will always include the identities of the sender and the receiver in all messages.

Step 2:

At this point, TTP_α answers to X with the certificate of the internal TTP responsible for the required service (TTP_β) if she is authorized to acquire an identity for that network. Otherwise, X is expelled from the identity acquisition process. TTP_α also stores X's identity (ID_X), her real-world certificate ($C_{\alpha X}$), the service identifier (S_{ID}), the request identifier (t_X) and the identity of TTP_β to manage the X's request.

$$ACCEPT\ MESSAGE,\ TTP_\alpha \rightarrow X :\ \{\{ID_{TTP_\alpha}, ID_X, t_X, C_{TTP_\beta}\}_{K^{-1}_{TTP_\alpha}}\}_{K_{\alpha X}}$$

Specifically, TTP_α decrypts the $HELLO\ MESSAGE$; checks X's certificate; verifies the signature of X; and checks the freshness of the timestamp t_X. Note that if the timestamp is out of time the identity request must be automatically canceled, this in all steps. Then, if the above is correct, TTP_α selects the internal TTP with which X must contact (from here on TTP_β) using the service identifier (S_{ID}); signs the certificate of TTP_β (C_{TTP_β}) together with t_X and the involved identities; encrypts the signed message using the public key of X in the real-world; and finally it sends the encrypted message to X.

Step 3:

In this step, X contacts TTP_β via TTP_α to obtain a valid nodeID for the P2P overlay. But before that, she generates a cryptographic key pair ($K_{\beta X}$,$K_{\beta X}^{-1}$) that is going to be used in the overlay network and selects the half of bits of her future nodeID ($\frac{1}{2}P_X$). Then, X sends $\frac{1}{2}P_X$ together with her public key ($K_{\beta X}$) to TTP_β, all protected to prevent TTP_α can access to this information. Finally, X also stores the certificate of TTP_β (C_{TTP_β}) and its identity (ID_{TTP_β}).

$$REQUEST\ MESSAGE\ 1,\ X \rightarrow TTP_\alpha :\ \{\{ID_X, ID_{TTP_\alpha}, t_X, A\}_{K_{\alpha X}^{-1}}\}_{K_{TTP_\alpha}}$$

where $A = \{K_{\beta X}, \{\frac{1}{2}P_X, ID_{TTP_\beta}\}_{K_{\beta X}^{-1}}\}_{K_{TTP_\beta}}$.

In more detail, X decrypts the $ACCEPT\ MESSAGE$; verifies the signature of TTP_α; checks the timestamp freshness and the TTP_β's certificate (C_{TTP_β}). Then, if all is correct, X selects the half of bits of her future nodeID ($\frac{1}{2}P_X$) and a cryptographic key pair ($K_{\beta X}$,$K_{\beta X}^{-1}$); signs $\frac{1}{2}P_X$ together with the identity of TTP_β using her new private key within the overlay ($K_{\beta X}^{-1}$); and encrypts that signed message together with her new public key within the overlay ($K_{\beta X}$) using the public key of TTP_β. For simplicity, from here on we denote this encrypted message as A. Finally, X signs A together with t_X and the involved identities using her private key $K_{\alpha X}^{-1}$; encrypts the previous signed message using the public key of TTP_α; and sends the encrypted message to TTP_α.

Note that the message A can only be decrypted by TTP_β although X still cannot communicate directly with it since she has not a valid nodeID. Thus, the anonymity of the user within the overlay network is preserved. It is also noteworthy that the half of bits of the new nodeID of X can be selected in many different ways (see Section 6.2.2).

Step 4:

In this step, TTP_α generates the blinding parameters needed to initialize the blind signature process with TTP_β ($blind\_param$) and sends these parameters together with the encrypted message A to TTP_β.

$$\begin{gathered} \mathit{REQUEST\ MESSAGE}\ 2,\ TTP_\alpha \rightarrow TTP_\beta : \\ \{ID_{TTP_\alpha}, ID_{TTP_\beta}, t_X, blind\_param, A\}_{K_{\alpha\beta}} \end{gathered}$$

Specifically, TTP_α decrypts the *REQUEST MESSAGE* 1; verifies the signature of X; and checks the timestamp freshness. Then, if everything is correct, TTP_α generates the blinding parameters ($blind\_param$) and encrypts A together with t_X, $blind\_param$ and the involved entities. It uses the symmetric key shared with TTP_β ($K_{\alpha\beta}$) to prevent malicious users can forge certificates. Finally, TTP_α sends the encrypted message to TTP_β.

In the same way that in TIAPPA, we include the identity of the sender and the receiver in the message to avoid a possible replay attack, as we have been able to check using the AVISPA tool. The problem is that in this protocol we also send several message with the same number of parameters in both directions of communication (*REQUEST MESSAGE* 2, *SIGN REQUEST MESSAGE*, *SIGN RESPONSE MESSAGE*, and so on) and in some cases the TTPs are not able to distinguish between two different messages. In this way an attacker can replay one of these messages in order to alter the identity acquisition process. It is also noteworthy that the blinding parameters are only required if the used blind signature scheme includes an initialization phase.

Step 5:

Briefly, in this step, TTP_β adds the other half of bits to the new nodeID of X (P_X) and generates the certificate using P_X and the public key sent by X ($K_{\beta X}$), among other relevant information (see Figure 6.1). Then, TTP_β blinds the certificate using the blind parameters ($blind\_param$) so that TTP_α can sign the certificate without seeing the content. Finally, it sends the blinded certificate ($blinded\_cert$) to TTP_α to be signed.

$$\begin{gathered} \mathit{SIGN\ REQUEST\ MESSAGE},\ TTP_\beta \rightarrow TTP_\alpha : \\ \{ID_{TTP_\beta}, ID_{TTP_\alpha}, t_X, blinded\_cert, H(blind\_param)\}_{K_{\alpha\beta}} \end{gathered}$$

More specifically, TTP_β decrypts the *REQUEST MESSAGE* 2; checks the timestamp freshness; decrypts the message A; and verifies the signature of X. Then, if all is correct, TTP_β adds the other half of bits to the nodeID of X (P_X); generates the X's certificate, unsigned yet; and blinds it using *blind_param*. Note that the blinded certificate (*blinded_cert*) is only the structure *TBSCertificate* blinded (see Figure 6.2). Finally, TTP_β encrypts *blinded_cert* together with t_X, the digest of the used blinding parameters ($H(blind\_param)$) and the involved identities using the symmetric key $K_{\alpha\beta}$; and sends the encrypted message to TTP_α.

In addition, TTP_β generates a Link Number (LN) to the user X and adds this LN and P_X to the user list. Since this is the first time that TTP_α contacts TTP_β for this request, it also stores the request identifier (t_X), the nodeID of X (P_X), her public key within the overlay ($K_{\beta X}$), and the blind parameters (*blind_param*).

Note that LN will be shared with TTP_α and will allow that the real-identity of X can be related to her nodeID. It is also noteworthy that $H(blind\_param)$ is included in this message to avoid replay attacks in the future using this message as others explained below ($ISSUE\ MESSAGE$ 1 or $LINK\ MESSAGE$).

Step 6:

In this step, TTP_α signs the blinded certificate (*blinded_cert*) using the appropriate cryptographic algorithm and sends it (*blinded_cert_signed*) to TTP_β.

$$SIGN\ RESPONSE\ MESSAGE,\ TTP_\alpha \rightarrow TTP_\beta :$$
$$\{ID_{TTP_\alpha}, ID_{TTP_\beta}, t_X, blinded\_cert\_signed\}_{K_{\alpha\beta}}$$

In more detail, TTP_α decrypts the $SIGN\ REQUEST\ MESSAGE$; and checks the timestamp freshness. Then, if the above is correct, TTP_α signs the blinded certificate; encrypts the blinded certificate previously signed (*blinded_cert_signed*) together with t_X and the involved identities using the symmetric key shared with TTP_β; and sends the encrypted message to it.

It is noteworthy that TTP_α signs the blinded certificate using a special cryptographic key (used only for blind signatures) and not using its normal public key, since the used cryptographic algorithms are different.

Step 7:

Briefly, in this step, TTP_β removes the blindness on the blinded certificate, signs the certificate signed by TTP_α and sends the final certificate ($C_{\beta X}$) to X via TTP_α.

$$ISSUE\ MESSAGE\ 1,\ TTP_\beta \rightarrow TTP_\alpha :\ \{ID_{TTP_\beta}, ID_{TTP_\alpha}, t_X, B\}_{K_{\alpha\beta}}$$

where $B = \{\{ID_{TTP_\beta}, P_X, C_{\beta X}\}_{K^{-1}_{TTP_\beta}}\}_{K_{\beta X}}$.

More specifically, TTP_β decrypts the *SIGN RESPONSE MESSAGE*; checks the timestamp freshness; removes the blindness on the blinded certificate; verifies the signature of TTP_α; and signs the X's certificate using its private key. Then, if everything is correct, TTP_β constructs a signed message with the new certificate of X ($C_{\beta X}$), P_X and its identity; and encrypts the message using the public key of X within the overlay ($K_{\beta X}$). For simplicity, from here on we denote this encrypted message as B. Finally, TTP_β encrypts the message B together with t_X and the involved identities using the symmetric key $K_{\alpha\beta}$; and sends it to TTP_α.

Step 8:

Very briefly, in this step, TTP_α forwards to X her new overlay certificate ($C_{\beta X}$).

$$ISSUE\ MESSAGE\ 2,\ TTP_\alpha \rightarrow X :\ \{\{ID_{TTP_\alpha}, ID_X, t_X, B\}^{-1}_{K_{TTP_\alpha}}\}_{K_{\alpha X}}$$

In more detail, TTP_α decrypts the *ISSUE MESSAGE* 1; and checks the timestamp freshness. Then, if all is correct, TTP_α signs the message B together with t_X and the involved identities; encrypts the signed message using the public key of X in the real-world; and sends the encrypted message to X.

Figure 6.3 shows the blind signature process performed by the two TTPs. We have detailed the exchanged information between the involved parties in the previous four steps (5, 6, 7 and 8).

Step 9:

Briefly, in this step, X contacts TTP_β, for the first time, to confirm that she has received her certificate and everything is correct.

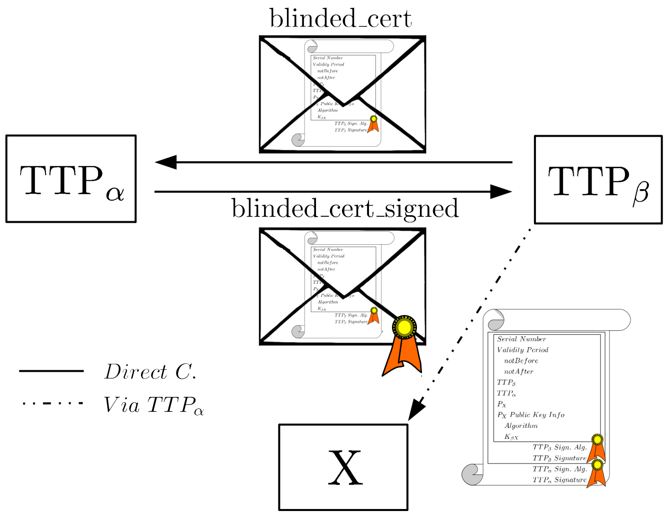


Figure 6.3: Blind signature process carried out by TTP_α

$$ISSUE\ ACK\ MESSAGE,\ X \rightarrow TTP_\beta:\ \{\{P_X, ID_{TTP_\beta}, t_X, H(C_{\beta X})\}_{K_{\beta X}^{-1}}\}_{K_{TTP_\beta}}$$

Specifically, X decrypts the *ISSUE MESSAGE* 2; verifies the signature of TTP_α; checks the timestamp freshness; decrypts the message B; verifies the signature of TTP_β; checks her new nodeID (P_X); and verifies her new certificate ($C_{\beta X}$). Then, if everything is correct, X calculates the hash value of her certificate ($H(C_{\beta X})$); signs $H(C_{\beta X})$ together with t_X and the involved identities; encrypts the signed message using the public key of TTP_β; and sends the encrypted message to TTP_β.

Note that X also verifies whether half of bits of her new nodeID are the same bits that she sent to TTP_α in the *Step* 3. In the case of those bits are not the same, X could cancel the request process sending an error message to the TTPs.

Step 10:

In this step, TTP_β reports to TTP_α that X has received her certificate correctly and sends to it LN bound to X.

$$LINK\ MESSAGE,\ TTP_\beta \rightarrow TTP_\alpha:\ \{ID_{TTP_\beta}, ID_{TTP_\alpha}, t_X, LN\}_{K_{\alpha\beta}}$$

In more detail, TTP_β decrypts the $ISSUE\ ACK\ MESSAGE$, verifies the signature of X, checks the timestamp freshness; and verifies the hash value $H(C_{\beta X})$. Then, if all is correct, TTP_β encrypts LN bound to X together with t_X and the involved identities using the symmetric cryptographic key $K_{\alpha\beta}$; and sends the encrypted message to TTP_α.

In the case that TTP_β does not receive the $ISSUE\ ACK\ MESSAGE$ after a certain time, it will revoke that certificate to avoid that the user X joins the network without finishing the process.

Step 11:

Briefly, in this step, TTP_α adds ID_X to the user list with the received LN and confirms to TTP_β that it has received LN correctly.

$$LINK\ ACK\ MESSAGE,\ TTP_\alpha \rightarrow TTP_\beta :\ \{ID_{TTP_\alpha}, ID_{TTP_\beta}, t_X, H(LN)\}_{K_{\alpha\beta}}$$

More specifically, TTP_α decrypts the $LINK\ MESSAGE$; and checks the timestamp freshness. Then, if all is correct, TTP_α calculates the hash value of LN; encrypts this value together with t_X and the involved identities using the symmetric key $K_{\alpha\beta}$; and sends the encrypted message to TTP_β.

Step 12:

In this step, successfully ends the identity request process by X.

Specifically, TTP_β decrypts the $LINK\ ACK\ MESSAGE$; checks the timestamp freshness and the received hash value; and gives for finished the request process by X.

The real-world identity of X is stored by TTP_α, the new certificate has been received by the newcomer and stored by TTP_β, and TTP_α and TTP_β share LN to relate the identity of X with her new nodeID (P_X).

6.2.2 NodeID Selection

Using RIAPPA, nodeIDs are jointly selected by users and the internal TTP (TTP_β) to prevent malicious users can chose their location within the overlay or the TTP_β can place users in certain positions by self-interest.

There are lots of ways to jointly select a random number between two parts, but we have decided to divide the selection of each bit between that TTP and the newcomer

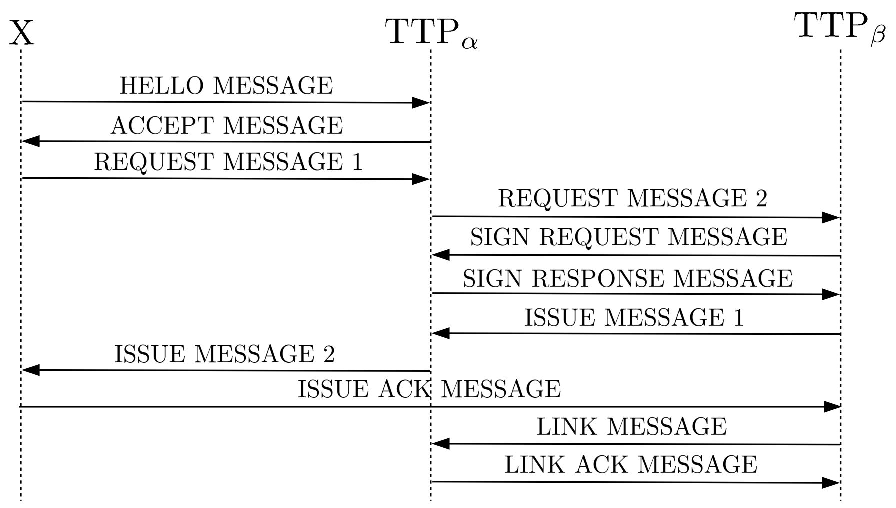


Figure 6.4: Message exchange

taking into account that not all bits are equally significant. For instance, in the Kad network, an attacker is within the tolerance zone of a target node if they have in common the most 8 significant bits of their nodeIDs [33]. Therefore, we cannot allow an entity (user or TTP) to select all those bits.

So, in our particular case, we use a scheme in which the user and the internal TTP decide randomly the half of bits of the nodeID, alternating them. The user selects the even bits (second, fourth, sixth bit, etc.), and the TTP selects the odd bits (first, third, fifth bit, etc.). Thus, no entity totally controls the most significant bits, and the nodeIDs generated are valid for any P2P overlay, independently of the number of bits used to mark off the tolerance zone. Randomness is needed to guarantee the uniform distribution of nodeIDs within the overlay, and this is also achieved with this way to proceed.

6.2.3 Node Operation

Deploying any of our protocols in an existing P2P overlay would affect the way in which nodes interact with each other. All nodes would need a valid overlay certificate (not revoked and properly signed by one or two TTPs, depending upon the protocol used) to operate within the network and securely communicate with other nodes. Before establishing any secure communication with a node, a sender has to obtain the receiver's certificate and verify its correctness. This includes verifying if the certificate has been properly signed by both TTPs and if it is still valid (not expired and not revoked). Then, if all is correct, this node will be able to use the normal cryptographic primitives to provide authentication, integrity and confidentiality, among other security services. On the other side of the communication, the receiver must also verify the sender's certificate.

6.3 Security Analysis

In the same way that TIAPPA, RIAPPA uses two TTPs to provide a mechanism for issuing robust and traceable identities in a P2P overlay while the full anonymity of the users is preserved. On the other hand, RIAPPA also provides stability of the nodeIDs since they do not change every time a user must change her overlay certificate. Next, we analyze the behavior of this protocol against some of the most common attacks in a P2P overlay.

Sybil attack. Taking into account that the access control to request a new overlay certificate is identical to that used in ICIAPPA and TIAPPA, we can also affirm that the Sybil attack is not a threat when RIAPPA is deployed in a P2P overlay.

Eclipse attack. To avoid an eclipse attack it is imperative to avoid (or limit) the Sybil attack and not to allow attackers to place themselves close to a target node. As we have seen above, RIAPPA avoids these two actions and therefore the Eclipse attack. Our protocol does not allow the self-generation of valid nodeIDs, they are jointly selected by the newcomer and the internal TTP, and users can only obtain one (or a few) nodeID per real-world certificate.

MITM attack. In the same way that in ICIAPPA and TIAPPA, both during the bootstrapping phase and the normal working of the P2P overlay, it is theoretically impossible for an attacker to impersonate some of the involved parties. Then, it is also very hard for an attacker to disrupt the normal behavior of the protocol.

Whitewashers. RIAPPA binds the real-world identity of users with their nodeIDs. So, assuming that users only have a real-world identity (or a small set of them), this protocol guarantees the nodeIDs' stability. This is possible because a user can renew her overlay certificate without changing her nodeID. Thus, a misbehaving node cannot purposefully leave the overlay and rejoin it again with a different nodeID in an attempt to shed any bad reputation she has accumulated under her nodeID. Thanks to this, robust trust and reputation systems can be used to prevent malicious behaviors and to promote honest collaboration among nodes.

6.3.1 Discussion of NodeID Requirements

In this Section, we discuss how all the design requirements exposed in Section 3.2 are satisfied by the nodeIDs generated using RIAPPA. As we have explained previously, and thanks to the use of real-world identities, each user only obtains a ***unique*** and ***stable*** nodeID; which is ***jointly selected*** between the user and the internal TTP, avoiding thus the user can select her location within the overlay. Moreover, the selection of alternate bits in nodeIDs guarantees a pseudo-random location of nodes. As we explain in Section 6.2.2, this allows us to obtain nodeIDs ***uniformly distributed*** within the virtual space with high probability.

In the same way that TIAPPA, RIAPPA can also provide full ***anonymity*** of the users within the overlay, since neither users nor TTPs by themselves are able to match the users' real-world identities with the nodeIDs. This is achieved thanks to the use of two TTPs, a blind signature over the users' certificates, and the encryption of messages. Obviously, we trust that the two TTPs will not collude, because in that case the system would be equivalent to a single-TTP system.

On the other hand, RIAPPA provides ***revocability*** of certificates. An overlay certificate can be revoked by several reasons. On the one hand, certificates can be revoked if an involved private key has been compromised. In this case, the internal TTP revokes the certificate and issues a new one with the same nodeID and a new cryptographic key pair. But before that, the user must contact the external TTP to be authenticated, and thus the external TTP requests the internal TTP to renew the user certificate. This is possible without losing the anonymity thanks to the Link Number (LN) that both TTPs share. Note that the user cannot request the certificate renewal to the internal TTP directly because she could be an attacker. The user needs to prove that she is the owner of the certificate. On the other hand, there is the possibility that the RIAPPA protocol does not finish properly but the certificate has been delivered to the user, or that a user misbehaves within the network. In both cases, the overlay certificate is revoked to prevent its use. Note that the first situation happens when a newcomer does not send the confirmation message ($ISSUE\ ACK\ MESSAGE$) to the internal TTP reporting that she has received her new certificate.

In addition, all certificates are ***verifiable*** by any user. They are signed by the two TTPs and contain the user's nodeID, her public key, and other information related to the owner.

Finally, regarding the requirement of ***traceability***, thanks to the fact that one of the TTPs (the external TTP) knows the real-world identity of the users, it can easily relate their identities with their nodeIDs. Therefore, RIAPPA is also prepared to de-anonymize users that commit serious offenses within the overlay (thanks again to the Link Number shared between TTPs), and so they can be prosecuted by the law.

6.3.2 Formal Validation of the Protocol

We have also performed an automated validation of the message exchange using the AVISPA tool [16] to validate the security of TIAPPA. We ran the tool using three different back-ends (OFMC, CL-AtSe and SATMC), and in all three cases the output was "SAFE".

Table 6.2 shows the messages exchanged between a newcomer and the two TTPs represented in the Alice and Bob notation. ID_X, ID_{TTP_α} and ID_{TTP_β} are the identifiers of the user X and the TTPs; $K_{\alpha X}$, K_{TTP_α} and K_{TTP_β} are their cryptographic public keys respectively; and $K_{\alpha\beta}$ is the symmetric key shared between the two TTPs. The functions $\{m\}_K$ and $\{m\}\_inv(K)$ represent respectively the encryption and signing of the message m using the key pair K and $inv(K)$, and "." indicates concatenation.

1	X	$\rightarrow$	TTP_α	:	$\{\{ID_X.ID_{TTP_\alpha}.t_X.S_{ID}.C_{\alpha X}\}\_inv(K_{\alpha X})\}\_K_{TTP_\alpha}$
2	X	$\leftarrow$	TTP_α	:	$\{\{ID_{TTP_\alpha}.ID_X.t_X.C_{TTP_\beta}\}\_inv(K_{TTP_\alpha})\}\_K_{\alpha X}$
3	X	$\rightarrow$	TTP_α	:	$\{\{ID_X.ID_{TTP_\alpha}.t_X.A\}\_inv(K_{\alpha X})\}\_K_{TTP_\alpha}$
4	TTP_α	$\rightarrow$	TTP_β	:	$\{ID_{TTP_\alpha}.ID_{TTP_\beta}.t_X.blind\_param.A\}\_K_{\alpha\beta}$
5	TTP_α	$\leftarrow$	TTP_β	:	$\{ID_{TTP_\beta}.ID_{TTP_\alpha}.t_X.blinded\_cert.H(blind\_param)\}\_K_{\alpha\beta}$
6	TTP_α	$\rightarrow$	TTP_β	:	$\{ID_{TTP_\alpha}.ID_{TTP_\beta}.t_X.blinded\_cert\_signed\}\_K_{\alpha\beta}$
7	TTP_α	$\leftarrow$	TTP_β	:	$\{ID_{TTP_\beta}.ID_{TTP_\alpha}.t_X.B\}\_K_{\alpha\beta}$
8	X	$\leftarrow$	TTP_α	:	$\{\{ID_{TTP_\alpha}.ID_X.t_X.B\}\_inv(K_{TTP_\alpha})\}\_K_{\alpha X}$
9	X	$\rightarrow$	TTP_β	:	$\{\{P_X.ID_{TTP_\beta}.t_X.H(C_{\beta X})\}\_inv(K_{\beta X})\}\_K_{TTP_\beta}$
10	TTP_α	$\leftarrow$	TTP_β	:	$\{ID_{TTP_\beta}.ID_{TTP_\alpha}.t_X.LN\}\_K_{\alpha\beta}$
11	TTP_α	$\rightarrow$	TTP_β	:	$\{ID_{TTP_\alpha}.ID_{TTP_\beta}.t_X.H(LN)\}\_K_{\alpha\beta}$

Table 6.2: Message exchange represented using the Alice and Bob notation.

In Appendix B.3 we show the HLPSL code of the performed validation and the obtained outputs.

6.4 Performance Analysis

Providing security always implies a cost in terms of delay in communications, bandwidth overhead, computational overhead, etc., which will be quantified in this section. Anyway, we should take into account that RIAPPA is only needed the first time that a user joins the overlay (or when she wants to renew or revoke her certificate). In other words, the operation of our protocol has almost no direct impact on the overlay performance. Once a user has obtained her overlay certificate (with the corresponding nodeID), she does not need to contact directly with the TTPs again unless her overlay certificate has expired or must be revoked. In addition, if one of the two TTPs fails, the certificate issuing service becomes unavailable but users with valid certificates can join and operate normally within the overlay without contacting the TTPs. Obviously, because of the use of digital certificates, all parts will need to manage revocation data.

To secure the communications between TTPs, we have decided to use symmetric cryptography, since this type of cryptography has lower computational cost than the public key cryptography, and both parties are automatically authenticated by sharing the symmetric key. In the case of communications between users and the TTPs we use asymmetric encryption, even being more computationally expensive, because their communications are only composed of a few small messages and adding an extra key exchange process is not worth. Table 6.3 shows an estimate of the size of each exchanged message in the protocol.

To make this estimate, we have considered the following algorithms and parameters. On the one hand, we have considered the use of the cryptographic public key protocol RSA using 2048-bit key length, the digital signature scheme DSA using also 2048-bit key length, the symmetric encryption protocol AES using 256-bit key length, the blind signature scheme based on ECC proposed by Fan et al. in [52] and the 256-bit hash algorithm MD6. On the other hand, we have taken into account the following parameters size: user/entity identifiers (16 bytes), timestamps (8 bytes), the service identifier (1 byte), RSA digital certificates (2500 bytes), DSA digital signatures (40 bytes), RSA public keys (256 bytes), DSA public keys (256 bytes), RSA encryption

Table 6.3: Size of the messages.

	Size (bytes)
HELLO MESSAGE	2656
ACCEPT MESSAGE	2655
REQUEST MESSAGE 1	814
REQUEST MESSAGE 2	731
SIGN REQUEST MESSAGE	474
SIGN RESPONSE MESSAGE	474
ISSUE MESSAGE 1	1255
ISSUE MESSAGE 2	1402
ISSUE ACK MESSAGE	411
LINK MESSAGE	12
LINK ACK MESSAGE	264

overhead (75 bytes), the blinding parameter (64 bytes), the blinded certificate (466 bytes), the blind signature (466 bytes), the overlay certificate (1100 bytes), hash values (256 bytes), and the link number (4 bytes). Note that we have considered the use of a blind signature scheme based on ECC because it has an inherent advantage in terms of smaller key size and lower computational overhead and security over other public key cryptosystems such as RSA and ElGamal.

As we previously stated, we have defined RIAPPA to be flexible in terms of cryptographic algorithms, including the type of encryption, signature and blind signature schemes. For this reason, we have evaluated the performance of the protocol in terms of the number of required cryptographic operations. Table 6.4 shows those cryptographic operations. As it can be seen in that Table, only 18 out of the 51 operations are performed by the user while both TTPs have a similar computational load. Therefore, if we take into account that a user only runs the protocol the first time she joins the network, we can consider that the computational cost is manageable for all kinds of today's devices (smartphones, tablets, netbooks, laptops, etc.).

The computational cost of RIAPPA will depend on the encryption and signature schemes which are implemented, and also on the number of requests that the system receives. In the case that the number of users becomes very high, both TTPs could be replicated creating two distributed access levels. Each group of TTPs should only share

Table 6.4: Summary of the cryptographic operations.

	$User$	TTP_α	TTP_β	**Total**
Asymmetric encryptions	4	2	1	**7**
Symmetric encryptions	0	3	3	**6**
Signatures	4	2	2	**8**
Asymmetric decryptions	3	2	2	**7**
Symmetric decryptions	0	3	3	**6**
Signature verifications	5	3	2	**10**
Blind signatures	0	1	0	**1**
Blind signature verifications	1	0	1	**2**
Hash function calculations	1	1	2	**4**
Total	**18**	**17**	**16**	**51**

the database where they store the user or node information.

Next, we have also calculated the computational cost of the cryptographic operations in each of the steps considering the same algorithms as previously: RSA-2048, DSA-2048, AES-256, the blind signature scheme proposed by Fan et al. [52] and MD6-256. Table 6.5 shows the time in milliseconds needed to realize all involved cryptographic operations in each step with the exception of the operations related with the blind signature process, since we only have the time needed to complete the entire process (signature and verification), 213.8227 ms. Note that this time should be divided between the steps 5, 6 and 7, and in the step 9 is also needed to verify the blind signature. All times in Table 6.5 have been obtained considering the use of a processor Intel Core i5-4570S 2.9 GHz [30, 117].

6.5 Comparison with Similar Proposals

Despite it is difficult to compare the three protocols proposed in this thesis with other similar proposals, since the requirements and the design goals are different, we can say that in general TIAPPA and RIAPPA are more complex than ICIAPPA and the proposals summarized in Section 2.1.5, both in terms of computational cost and bandwidth consumption. However, RIAPPA is the most complete protocol in terms of security because it is the only one that is able to fulfill all those considered requirements. Table 6.6 shows a summary of all requirements fulfilled by the three proposed protocols.

Table 6.5: Computational cost of the cryptographic operations.

	Computational Cost (ms)
Step 1	0.4588
Step 2	2.8087
Step 3	108.1758
Step 4	2.3534
Step 5	2.3556
Step 6	0.0045
Step 7	0.8925
Step 8	0.4647
Step 9	5.6564
Step 10	2.4233
Step 11	0.1184
Step 12	0.1171

	ICIAPPA	**TIAPPA**	**RIAPPA**
Uniqueness	✓	✓	✓
Stability	✗	✗	✓
Joint Management	✓	✓	✓
Verifiability	✓	✓	✓
Revocability	✓	✓	✓
Traceability	✓	✓	✓
Anonymity	Partial	Full	Full
Uniformity	✓	✓	✓

Table 6.6: Requirements comparison between ICIAPPA, TIAPPA and RIAPPA.

In [105], the complex structure proposed by authors requires a potentially large number of exchanges with varying servers to obtain a single nodeID, in addition to solving cryptographic puzzles. This is just to limit the Sybil attack. The same occurs in [36, 47, 81, 114] or [127].

In [34], Butler et al. propose three new identity assignment protocols where users are weakly authenticated via callback using their IP addresses, which limits the Sybil attack poorly. We use the real-world identity of the user to authenticate them, which is the best way to prevent this attack [50]. Moreover, in all three cases, authors propose that trusted third parties generate nodeIDs randomly, which can be a major problem. TTPs can place new nodes within the overlay according to their qualities seeking the benefit of the network, of another node, or even its own benefit.

Finally, the Likir system [23, 24, 25, 53] by Aiello et al. is a complete security proposal for P2P overlays and the one that has conceptually more similarities with TIAPPA and RIAPPA. In particular, in its user registration module. To prevent/limit the Sybil attack, Aiello et al. assume that users have an OpenID account in the same way that we assume that users have a digital certificate bound to their real-world identity. Obviously, our design requirement of having a real-world identity is more demanding than the requirement of Likir, and it may cause problems to honest users that do not have a digital certificate. But our three protocols are designed to prevent the Sybil attack more aggressively. Using one of them, an attacker needs to steal, or forge, a real-world identity (digital certificate) to launch a Sybil attack. In Likir, the attacker only needs to generate a set of OpenID identities, which is in general less hard to do. Moreover, the OpenID protocol is not exempt from security problems [48]. On the other hand, in Likir, new users must also contact two central entities to obtain their overlay certificates. Users are authenticated by an OpenID provider and then redirected to a trusted entity (Certification Service) which is responsible for generating nodeIDs and issuing certificates. One of those entities does not depend on their system but it is used to add human interaction in the process of obtaining the overlay certificate, making this process more expensive, and thus, limiting the Sybil attack. Moreover, in the same way than our TTPs (in TIAPPA and RIAPPA) keep a link number to bind nodes with users, Likir's certification service also keeps a track of the association between userIDs and LikirIDs, but not with real-world identities. As TIAPPA and RIAPPA, Likir also guarantees the full users' anonymity; however, it does not consider providing traceability since users do not need to reveal their real-world identity at any time. In fact, Likir only uses that track to avoid the unnecessary issuance of new LikirIDs. Finally, another difference between Likir and our protocols is that the Eclipse attack problem is addressed differently. Likir uses a random nodeID generation, while one of our design goals is to implement a joint generation between a user and the internal TTP. In this way, we prevent that the internal TTP, or the main TTP in ICIAPPA, can place a node at a certain location in the overlay. Finally, the protocol exchange, messages and operation proposed by Aiello et al. are rather different from ICIAPPA, TIAPPA and RIAPPA.

6.6 Scenarios

ICIAPPA, TIAPPA and RIAPPA could be deployed in a variety of scenarios in which security is a must, including commercial applications. ICIAPPA could be deployed in countless scenarios in the current Internet; however TIAPPA and RIAPPA are limited to a small number of situations. Specially, we consider that the kind of scenario which better fits with those protocols consists of one external TTP, and one or more internal TTPs. As we previously mentioned, the external TTP is responsible for authenticating users (by the use of real-world identities) and each of the internal TTPs manage a different overlay. Those overlays may be different in many ways, for instance, the overlays can provide different resources (cpu sharing, storage capacity, virtualization, etc.), offer different services (music sharing, video on demand, audio or video conference, etc.), or distribute different contents (sports, news, cartoons, adult content, etc.).

On the other hand, a scenario in which different external TTPs work with the same overlay (managed by only one internal TTP) is also possible. But if we want to assure that users do not obtain multiple nodeIDs, it should be mandatory that users only ask one external TTP for an overlay certificate, which is uncontrollable. Therefore, different external TTPs should be perfectly synchronized to avoid (or limit) that.

To better understand the usability of TIAPPA and RIAPPA, we will introduce two examples of possible scenarios. Let us first consider an anonymous and distributed video sharing service, in which users share their own private videos with the rest of the users. This service could be similar to the one provided by YouTube, but using the capabilities of a totally distributed overlay and assuring the anonymity of the users. In the case of being a commercial service, it may be provided by a private company (service provider) at flat rate. This company would act as an external TTP, and hence it would be responsible for authenticating users when they hire the service, using real-word identities. As users need a real-word certificate, they can ask the company to issue a new one, or they can use any existing and valid certificate (issued by a known CA, Government or institution in which the company trusts). On the other hand, the internal TTP should be any other entity with no relationship with the previous one (to avoid collusion attacks from the TTPs), for instance a commercial CA. In our opinion, CAs may be good candidates for being internal TTPs since they have experience in managing certificates. Also, they will probably have no interest in performing collusion attacks since their business depends on trust and reputation. Obviously, CAs are not for free, and the cost of issuing certificates must be included in the fare that users would pay.

In practice, a newcomer that wants to access to the service starts one of the two protocols to contact the service provider (external TTP) to issue a valid overlay certificate (with the cooperation of the internal TTP). Once the protocol has ended successfully, the user has all the necessary to start sharing her private videos within the overlay in a secure and anonymous way. These contents may be of any type: sports, politics, adult content, etc. For this reason, we have included in the protocol a service identifier (S_{ID}), which allows users to indicate which kind of contents they want to share. In addition, in RIAPPA, the stability of nodeIDs guarantees that a robust trust and reputation system can be used to encourage good users, and to isolate dishonest ones. Even, in the case that a user performs an illegal action, it is possible to trace which user is responsible for that action and start a legal investigation.

As a second example, we can consider an e-democracy service, in which citizens can share opinions/contents about the proper working of the political processes in their region. For instance, the different local governments can act as external TTPs, authenticating their citizens[1]. Notice that in the most cases new identity cards and passports are equipped with digital certificates, so the cost of this service would not be so high. On the other hand, another independent institution will act as internal TTP, for instance one dependent on the judicial system (assuming independence of legislative/executive/judicial powers, to avoid collusion attacks).

These are only two examples of possible scenarios, but there may be more in which users do not totally trust the service provider and do not want to reveal their real-world identities.

[1]We will assume that citizens only depend on a local government to guarantee uniqueness of the nodeIDs.

6.7 Conclusions

Public key certificates provide the basic security services to P2P overlays: authentication, data confidentiality, data integrity, non-repudiation and access control. But as we have seen in Chapter 3, using these services in these networks is not enough to avoid many of its threats. Even the use of a trust and reputation system by itself is not enough to guarantee a certain level of security. For this reason we have proposed RIAPPA (A Robust Identity Assignment Protocol for P2P overlAys), a new protocol for issuing stable identities (digital certificates) in a secure and anonymous way minimally altering the current operation of the overlays. Specially, this protocol provides stability of nodeIDs because its generation does not depend on any information contained in the certificate; they are jointly selected by the internal TTP and the newcomers. In this way, users are placed in the virtual space pseudo-randomly.

Other important point on today's Internet, it is the capability to identify misbehaviors and punish the perpetrators. For this reason, RIAPPA has been designed to provide traceability of malicious users. On the one hand, this feature depends on the LNs shared by the two TTPs, which provide a link between users and nodeIDs; and on the other hand, it depends on the access control carried out by the internal TTP, which can be done using users' real-world certificates. Therefore, this protocol is the only one that achieve at the same time full anonymity, revocability, stability and traceability. These requirements may seem to be in conflict, but when you have to deploy a commercial service over these networks you cannot treat equally all misbehaviors.

Finally, we can conclude that the cost of RIAPPA in terms of performance is reasonable according to our analysis, and considering that a user only executes the protocol the first time she wants to join the service (or when her overlay certificate must be renewed or revoked). On the other hand, the use of digital certificates always implies the distribution of revocation data, which can be a challenging task in this kind of networks. For this reason, this issue is specifically addressed in Chapter 7.

Chapter 7

CRL Distribution System for the Kad Network

Contents

The use of digital certificates in a P2P overlay network involves having an efficient revocation system. However, traditional revocation systems are not suitable for these type of networks since, as we have stated in Chapter 2.1.4.2, two major issues arise in these networks. On the one hand, the amount of revocation data can be quite large due to the number of users and the use of pseudonyms, which would hinder their distribution. On the other hand, there is a trade-off between the freshness of the revocation data and the overhead caused by its distribution. Therefore, we consider that it is necessary to define a new way to distribute revocation data taking into account the specific operation of the P2P overlays.

In this Chapter, we present a new distributed revocation system [38] that tries to optimize both the availability of the revocation data and its freshness. The system is based on the use of CRL segments and it is designed to work in a particular P2P overlay, the Kad network; although it could be easily deployed in other networks. CRL segments are distributed using the overlay and several replicas of `keywordIDs` and `sourceIDs` are stored in different nodes.

With our mechanism, the CA is just another node of the network responsible for publishing and storing all new CRL segments in the P2P overlay. Unlike in traditional methods, the normal nodes do not only download CRL segments from the CA, but they provide such segments to other nodes. In this way, the number of potential servers of revocation data increases exponentially, which certainly increases the availability of the revocation data. Moreover, this novel way of distributing CRL segments allows us to improve the CRL updating policy maintaining all the security properties of a traditional revocation system; CRL segments are updated independently and provided when a CA considers necessary. To this respect, a CA can generate a segment before a previous one has been expired. This is called over-issuance and it can be used to improve the freshness of the revocation data with a low bandwidth cost.

7.1 System Requirements

Analyzing the features of P2P overlays, we conclude that a revocation system for these networks should take into account the following requirements:

1. **Scalability:** The total cost of validating the status of a certificate is critical in P2P networks because such networks are designed to be very large networks, involving from tens of CAs and millions of users conducting billions of transactions.

2. **Load balancing:** The cost of storing and distributing the revocation data must be shared out among all nodes.

3. **Tolerance to frequent changes:** Since P2P networks are characterized by the lack of node stability, the system must be tolerant with frequent joins and leaves.

4. **Performance:** The entire system must be efficient, without requiring excessive computational complexity or network overhead by any participant.

5. **Unforgeability:** No entity must be capable of generating fake revocation data.

7.2 Proposal Overview

Our proposal uses CRL segments to avoid managing entire CRLs of large size, although it does not simply divide a CRL into several segments, but it generates the CRL segments separately. In this way, each CRL segment can be updated when the issuer CA considers necessary, allowing the existence of over-issued segments. Then, these CRL segments are stored and shared in a distributed way, improving their availability and preventing CAs from becoming bottlenecks.

Within the P2P overlay, CAs act like normal nodes, and all of them are considered as potential servers of CRL segments. However, they are solely responsible for issuing these segments and inserting them into the network. The introduction of the CRL segments into the overlay is performed by using a pull mechanism, that is, initially the only server to download a certain CRL segment from is the CA issuer of that segment. Regarding the validity and the integrity of the revocation data, CAs are also responsible for avoiding that a malicious node can generate fake CRL segments, or modify a valid one. To achieve this degree of security, CAs sign all segments using its cryptographic public key in the same way they do with standard CRLs.

CRL segments are shared by nodes from the first time these lists are stored in them. Thus, each time a node downloads a CRL segment, this node becomes a new server of that segment within the overlay. Whenever a CA updates a CRL segment the process starts again.

Taking into account the Kad network features, once a node has a chunk[1] of a segment it begins to share this chunk with the rest of nodes, although it has not downloaded the entire segment yet. In this way, the CA does not become a bottleneck when a new CRL segment is issued, or the network is initialized, even if the number of requests in a short period of time is very large. The overlay itself is in charge of distributing the requests to the new servers (nodes).

Next, we briefly describe the system operation by means of a simple example (3-bit key space) from the point of view of a node A (001), which is shown in Figure 7.1. Let us suppose that A wants to verify whether the certificate c is valid or not. First of all, the node A calculates in which CRL segment the certificate c should be stored if

[1]Chunks are the smallest pieces of information which are downloaded or managed by the P2P programs.

it has been revoked. In this particular case, let's consider that the certificate c should be stored in the CRL segment i. Once A knows which CRL segment needs to verify, it checks if the segment i is stored in its local memory or not. If the segment is not stored in the local memory, A sends a CRL segment i request to the known nodes of each bucket (nodes B (110), C (011) and D (000)). Then, this request reaches the node C, among others, which in turn responds to A indicating that the node E (010) is closer to the CRL segment i. Next, A sends another CRL segment request to the node C and downloads the segment. Note that the Kad network operation is quite more complex than the illustrated example. In this case, the number of nodes which would respond to a CRL segment request and the number of iterations would be very large. Remember that the size of the Kad key space is 2^{128} (see Section 2.1.2.5).

Finally, A has to check whether the serial number of c is stored within that segment or not. If the serial number of the certificate c is stored within the segment means that the certificate is not valid, otherwise c is not revoked.

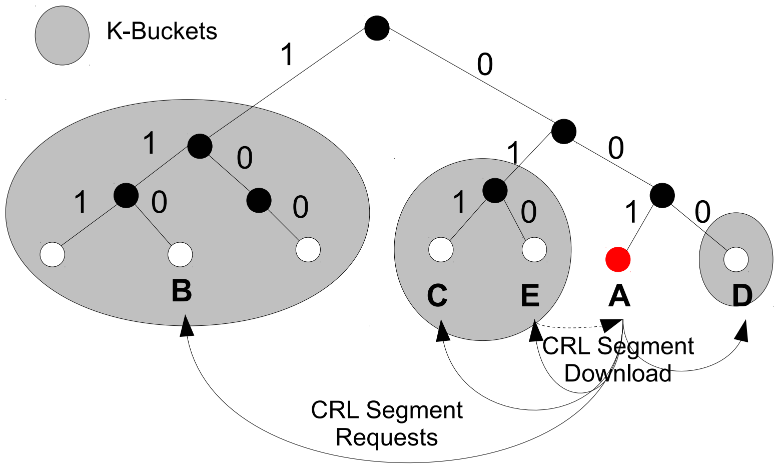


Figure 7.1: Example of a CRL segment search in the Kad network.

7.3 CRL Segment Generation

Digital certificates are distributed into 2^k CRL segments according to their serial numbers. To that end, CAs use a hash function ($h(\cdot)$) which given a certificate serial number returns the number of the segment where that certificate should be stored in the case of being revoked. In this way, each time a CA revokes a certificate, it calculates the hash of the certificate serial number and stores it in the CRL segment indicated by the output of that hash function. In the same way, nodes also calculate the hash of a certificate serial number if they want to verify the certificate validity, and so, they obtain the number of the segment that they must consult.

Regarding the implementation of the system, the number k is given by the length of the hash function used to obtain the number of a certain CRL segment. This length is independent of the length of the certificate serial numbers used by each CA[1]. Therefore, each segment will always contain the same potential range of certificate serial numbers.

The simplest hash function that meets the above requirements is the modular hash function; $h(c) = c \ mod \ 2^k$, where c is a certificate serial number. In this way, the maximum size of the segments will be limited and equal for all of them. Taking into account that the number of issued certificates can be at most 2^n, each segment will store a maximum of 2^{n-k} revoked certificates. And thanks to the simplicity of this hash function, the computational overhead introduced in the CAs and the nodes is minimum. In Section 7.6 we analyze the proper behavior of the modulo operation to uniformly distribute the certificate serial numbers among the CRL segments.

To create/update different CRL segments, or to calculate a desired CRL segment number, it is not necessary to know the total number of revoked certificates at that time. However, CAs need to take into account the potential maximum number of revoked certificates in the worst case to properly dimensioning the system, that is, properly select k. If the number of revoked certificates is expected to be really large, k must be large too and vice versa. In this way, the system performance will be improved.

Figure 7.2 illustrates the distribution of a group of revoked certificates among 128 CRL segments. For that, a CA calculates the *modulo 128* of the serial number of each revoked certificate, and obtains the number of the segment where it must store each serial number. Using the same operation, all users obtain the number of the CRL segment where they need to search a certain serial number. Note that, for simplicity, the CRL of the example only contains the certificate serial numbers, since our system does not need to use other information.

It is also noteworthy that in the network initialization phase there may be empty CRL segments, but even in the unlikely case that a segment does not store any revoked

[1]The X.509 v3 certificate format only specifies that the length of the serial numbers of the certificates must be no longer than 160 bits [45].

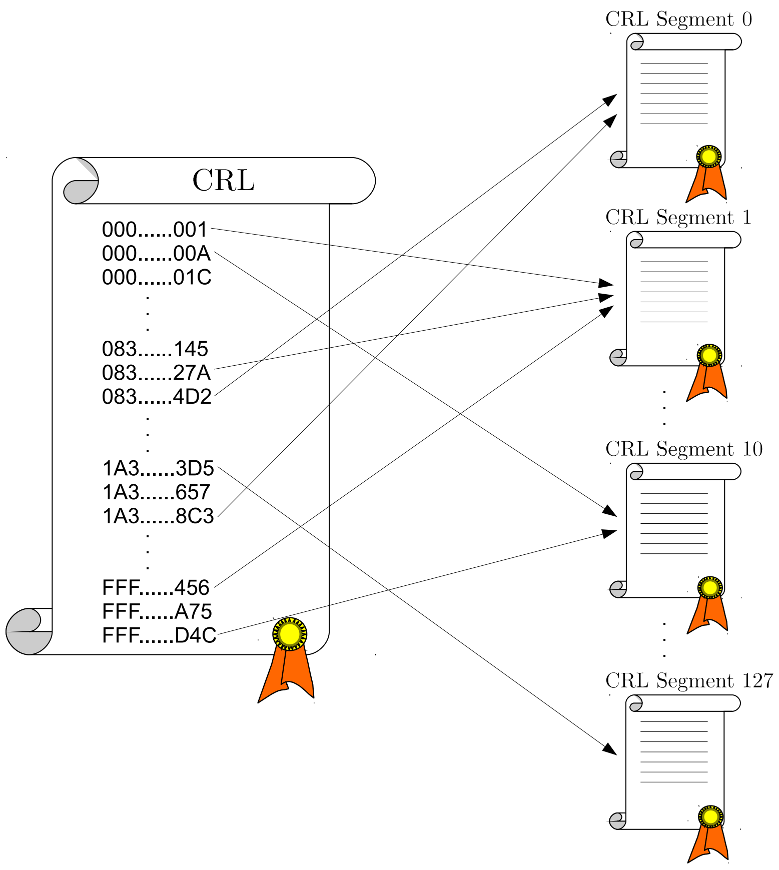


Figure 7.2: CRL segments generation.

certificate its publication is also necessary. Nodes must be able to check whether a certificate is valid or not by means of a document signed by a CA. For security reasons, the fact that a node does not find a certain segment is not enough to assure that a certificate is valid. Therefore, empty signed CRL segments should also be published and distributed among the network nodes.

7.4 CRL Segment Sharing

The CRL segments are shared by the Kad network like any other resource. As we have mentioned above, each CRL segment has an identifier number which is used by the nodes to find the segment that they need to check. On the one hand, the name of the CRL segments within the overlay is of the form "CRL_CA_i $Segment\_xxx$ $Date\_yyyyyy$", where CA_i identifies the CA that issued the segment, xxx represents the CRL segment number and *yyyyyy* is the date that the segment was issued (following the format mmddyy), e.g., CRL_CA_1 *Segment*_010 *Date*_062513. And on the other hand, their metadata and sources are published in the network and stored in several nodes with three associated keywords: CRL_CA_i, *Segment*_*xxx* and *Date*_*yyyyyy*. The first keyword identifies the segments issued by the same CA, the second keyword is used to differentiate each particular CRL segment and the last keyword shows the data freshness.

As we have explained in Section 2.1.2.5, resources are located within the Kad network through their `keywordIDs` and `sourceIDs` ($hash(keyword)$ and $hash(file)$ respectively). Therefore, when a node wants to download a new CRL segment, it performs a search for one or more related keywords, which give pointers to multiple sources. Then, these sources return which nodes have the desired segment. Finally, the node starts to download the desired CRL segment.

Once the node already has at least a chunk of the CRL segment stored in its memory, it performs the publication of that segment. To that end, the node publishes two types of references, *metadata* and *sources*, which are sent to nodes in its *tolerance zone*. Keywords are distributed metadata that reference to sources. Sources are the location information pointing directly to the node, which keeps a copy of the CRL segment. In Kad, metadata and sources are replicated in tens of nodes within the overlay, so the searches are performed faster and return more results.

Continuing with the example of Figure 7.2, let us consider that a node needs to verify the validity of a certificate with serial number "FFF......D4C". First of all, it calculates the hash value of this serial number and obtains the value 10. Then, the node searches for the CRL_CA_1 *Segment*_010 and identifies a considerable amount of nodes which have stored this segment. Note that it will find this segment in several versions (different dates). Finally, the node downloads the most current CRL segment from her preferred peers and publishes it to make it more available.

7.5 CRL Segment Issuance

As time goes by, CAs must update CRLs they have issued. The new revoked certificates must be added to the lists and expired revoked certificates must be removed from them.

In the traditional way of issuing these lists, new CRLs are issued periodically (e.g., hourly, daily, or weekly). Every time a new CRL is issued, all nodes that want to verify the validity of a given certificate must download the new CRL. However, updating CRLs periodically without taking into account the certificate revocation rate and the network activity may cause problems about the freshness of the revocation data. If the overlay revocation rate is high, a node can be consulting a stale CRL, which contains outdated information or lacks of relevant information.

For this reason, in this proposal, revocation data are not issued periodically. New revocation data are issued when the CA considers that it has enough information to update a segment or when the segment has not been updated for a considerable period of time. Moreover, our revocation mechanism takes into account that when a standard CRL is going to be updated, there are long pieces of the list which do not have new information to update. Therefore, it is more efficient to issue each CRL segment separately and only when it is necessary. So, CRL segments are managed as entire CRLs and issued independently by CAs, that is, a CA adds or removes a certificate from the corresponding CRL segment whenever the certificate is revoked or expired, and when necessary, the CA issues the updated CRL segment.

In this way, we define a threshold value of revoked certificates per segment daily (U_s) above which a CA must issue the CRL segment again. This value is compared by the CAs for each CRL segment every time they have a new revoked certificate. If the number of revoked certificates added to a segment during the last 24 hours (Δ) is greater than U_s ($\Delta > U_s$), or if the segment was last updated 24 hours ago, then the CA must issue that CRL segment updated. Otherwise, it is not necessary to issue that CRL segment. Note that most CAs issue new CRLs every 24 hours or less [45], for this reason we update the CRL segments after 24 hours in the worst case.

The defined threshold value depends on the number of CRL segments ($f = 2^k$), the certificate revocation rate per day (R_c) and a setting parameter (η):

$$U_s = \eta \frac{R_c}{f} \tag{7.1}$$

We define the parameter η to provide CAs some degree of control over the threshold value. This parameter allows CAs to increase or decrease the threshold value depending on whether they want to prioritize the freshness of the revocation data, or on the contrary they want to reduce their consumption of resources. η is simply the weight assigned by the CA to U_s. It is also noteworthy that we only take into account the number of new revoked certificates during the last 24 hours (Δ) to compare with U_s, and not the number of revoked certificates which have expired. Both types of data must be updated in the CRL segments: new revoked certificates have to be added to the segment and expired revoked certificates have to be removed from the segment. However, note that from the point of view of security, removing expired certificates is not critical or urgent. It is the process of adding new revoked certificates who must decide the segment updates.

From the point of view of nodes, whenever they download a certain CRL segment, first of all they must verify the validity period of that segment. Note that the name of the files indicate the issuance date of the segment, but nothing prevents they have downloaded a fake CRL segment. To do so, and taking into account that all segments are updated at different times, each segment is time-stamped using two fields of it, *This Update* and *Next Update*. The first field indicates the time when the segment was issued and the second one when that segment will be updated in the worst case, since any segment can be updated when the CA considers necessary. However, nodes do not have any mechanism to know if the CRL segment they are validating has been over-issued or not. They can only compare the issuance date between different CRL segments.

7.6 Performance Analysis

In our revocation data distribution system, we store data in different CRL segments which are distributed by the CAs to some nodes of the network. And from that time, these nodes can also distribute those segments to other nodes. In this way, we reduce the number of CRL segment requests received by the CAs. The storage requirements for each node are also reduced thanks to the segmentation of the revocation data.

To analyze the request rate of CRL segments in CAs we have used the model proposed by David A. Cooper [46], which defines the probability density function with which a node sends a CRL segment request to a CA. Taking into account that CAs issue a new CRL segment at time $t = 0$, the probability of a node requesting a CRL segment is defined, for the first time, at time t.

If a CRL segment is issued at time $t = 0$, the probability that a node will send the CRL segment request within the interval $[t, t + dt]$ depends on the probability that the node will need to perform a validation within this interval. A node will request a certain CRL segment within the interval $[t, t + dt]$, if and only if, it needs to validate a certificate that requires the use of that segment and has not validated any certificate which had required the use of this segment within the interval $[0, t)$.

Assuming a large number of nodes for the Kad network, we can assume that certificate validations are mutually independent and that their time distribution follows a Poisson Law. With these considerations, the probability that a node attempts n certificate validations in t can be expressed as:

$$P[n\ certificate\ validations\ in\ t] = \left[\frac{(vt)^n}{n!}\right] e^{-vt},\ n = 0, 1, 2, 3... \tag{7.2}$$

Where e^{-vt} is the probability that the node will not perform a validation within the interval $[0, t)$ and v is the certificates validation rate. In addition, we assume that all certificate validations are equally likely to require access to any of the CRL segments. If we have f CRL segments, there is a probability of $\frac{1}{f}$ that a certain segment will be needed to perform a certificate validation. Thus, the probability that a certain segment will not be needed for any of n certificate validations is:

$$P[no\ request\ a\ certain\ segment\ for\ validating\ n\ certificates] = \left(1 - \frac{1}{f}\right)^n \tag{7.3}$$

Combining equations (7.2) and (7.3), and again according [46], the probability that a node will not request a certain segment within the interval $[0, t)$ is:

$$P[no\,request\,a\,certain\,segment\,in\,[0,t)] = \sum_{n=0}^{\infty}\left(1 - \frac{1}{f}\right)^n \left[\frac{(vt)^n}{n!}\right] e^{-vt} = e^{-vt/f} \tag{7.4}$$

The probability that a node needs to request a segment within the interval $[t, t+dt]$ is $ve^{-vdt}\ dt$. However, due to the interval $[t, t+dt]$ is infinitesimally small, we can assume that the probability that a node needs to validate more than one certificate within the interval $[t, t + dt]$ (in the limit $dt \to 0$) is 0 ($e^{-vdt} \approx 1$). Therefore, probability that a node needs to request a segment within the interval $[t, t + dt]$ is:

$$P[request\ a\ segment\ in\ [t, t + dt]] = ve^{-vdt}\ dt \approx v\ dt \tag{7.5}$$

And as the probability that a validation will require the use of a certain segment between all possible segments is $\frac{1}{f}$ (equally likely), the probability that this segment will be needed in the interval $[t, t + dt]$ is:

$$P[request\ a\ certain\ segment\ in\ [t, t + dt]] = \frac{v\ dt}{f} \tag{7.6}$$

Using equations (7.4) and (7.6), and multiplying by the number of nodes within the network (N), the total number of requests that we expected of a certain segment within the interval $[t, t + dt]$ can be expressed as:

$$N_s(t) = \frac{Nve^{-vt/f}\ dt}{f} \tag{7.7}$$

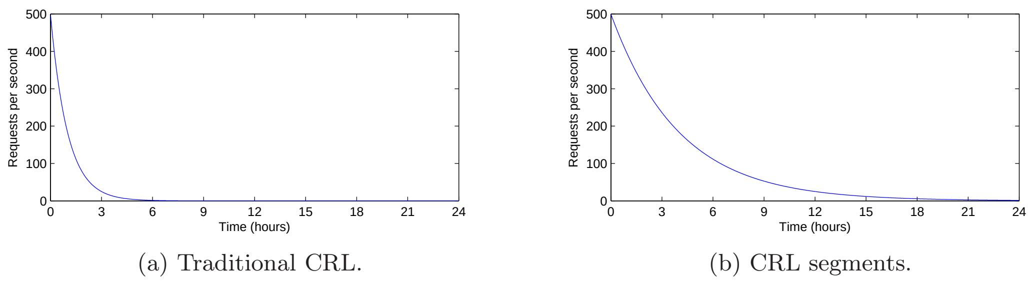
And the total request rate of a CRL segment is:

$$R_s(t) = \frac{fN_s(t)}{dt} = Nve^{-vt/f} \tag{7.8}$$

Taking into account that the request rate for an entire CRL to a CA at time t is $R(t) = Nve^{-vt}$, the CRL segment request rate drops off with the number of segments in which the revocation data are stored, but not so with the peak request, since $R_s(0) = R(0) = Nv$.

This difference can be seen by comparing Figure 7.3a with Figure 7.3b. Figure 7.3a shows the request rate for an entire CRL and Figure 7.3b the request rate for the CRL segments, both graphs over the course of 24 hours. We have assumed that the entire CRL and the CRL segments were issued at time 0, and that no others were issued during the course of that 24 hours. We have also assumed that the number of nodes within the Kad network (N) is 1 million, the validation rate (v) is 50 certificates per day, and the number of CRL segments is $2^7 = 128$.

In our distribution system, a CRL segment is also retrieved by several nodes, therefore we also determine the request rate of a certain node for a given segment. In addition, the CRL segments are updated independently. To do so, we assume that all server nodes are selected with the same probability, since nodes usually download the segments from the closest server node.



(a) Traditional CRL. (b) CRL segments.

Figure 7.3: CRL and CRL segments request rates.

The probability that a given node will not perform the request of a certain segment to a certain node within the interval $[0, t)$ is:

$$P[no\ request\ a\ certain\ segment\ in\ [0,t)] = (N/P)e^{\frac{-vt(N/P)}{f}} \tag{7.9}$$

Where N is the number of nodes within the network, P is the number of potential server nodes of a certain CRL segment and $\frac{N}{P}$ is the average number of nodes which download that segment from the same server node. The probability that one of these nodes will request that segment to a certain node within the interval $[t, t + dt]$ is:

$$P[request\ a\ certain\ segment\ in\ t] = \frac{ve^{-vdt}\ dt}{f} = \frac{v\ dt}{f} \tag{7.10}$$

Combining the equations (7.9) and (7.10), we can determine the total expected number of requests for a certain segment to a certain node in the interval $[t, t + dt]$:

$$N'_s(t) = \frac{(N/P)ve^{\frac{-vt(N/P)}{f}}\ dt}{f} \tag{7.11}$$

And the total request rate:

$$R'_s(t) = \frac{fN'_s(t)}{dt} = (N/P)ve^{\frac{-vt(N/P)}{f}}s \tag{7.12}$$

As can be seen in Figure 7.4, the peak request has decreased, since $R'_s(0) = (N/P)v$ and the CRL request rate continues decreasing with the number of segments in which revocation data are divided. Thus, our system improves the distribution of revocation data compared with the traditional CRL and fragmented CRL methods.

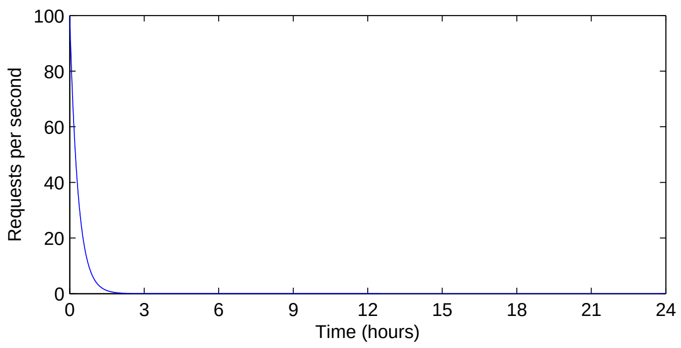


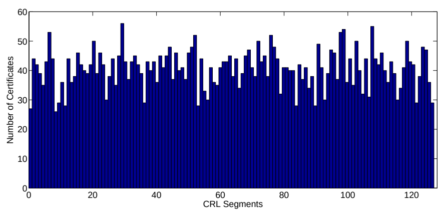
Figure 7.4: CRL segments distributed.

Regarding performance in the client side, this system introduces a small computational cost for every certificate validation. However, the time required to calculate a modular hash function with current devices may be considered negligible in most cases. In addition, segmentation reduces the storage space that clients must dedicate to store revocation data, since in most cases they only need to store certain CRL segments and not the entire CRL.

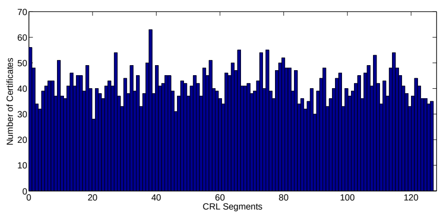
Regarding CAs performance, the problem is the number of hash functions that they must calculate, one for each revoked certificate. Anyway, all hash operations can be done previously and stored in a database. Not so in the case of signatures since CAs must sign all issued CRL segments, but the fact that segments are updated independently minimizes this latter problem. We have decided to use a modular hash function because it does not consume too many resources, but also because this kind of hash functions distribute the serial numbers with a good uniformity. This can be observed in Figure 7.5, where four different CRLs of important CAs are divided in 128 segments. The first two graphs represent the distribution of two CRLs issued by Verisign [18], and the next two show two CRLs issued by Thawte [15]. Note that we have been able to obtain these CRLs since they are completely public data.

7.7 Conclusions

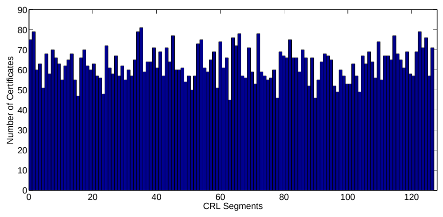
Today, P2P overlays are not yet mature, in terms of security, to implement commercial applications, such as pay-per-view video streaming applications. To address this problem, the adoption of a PKI seems to fulfill the main security requirements, al-



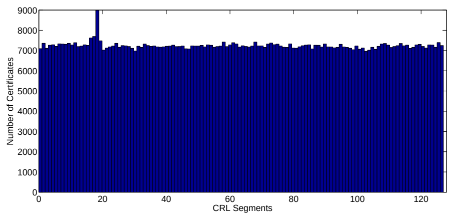
(a) CRL with 5.190 revoked certificates.



(b) CRL with 5.346 revoked certificates.



(c) CRL with 8.054 revoked certificates.



(d) CRL with 926.757 revoked certificates.

Figure 7.5: Distribution of different revocation data sets in CRL segments.

though in these networks it has some associated problems, such as the distribution of the revocation data.

CRLs seems to be the most widely used mechanism to distribute revocation data, but CAs can become a bottleneck since CRLs must be updated frequently to maintain data freshness. Also, their size grows exponentially with the number of network users and the use of several pseudonyms. For these reasons, we propose a new revocation data distribution system for the Kad network, where CAs store revocation data in several CRL segments, which are stored in a large number of nodes to improve accessibility and availability. In addition, these segments can be issued independently, which improves data freshness without inducing a great cost for the network.

Distributing and replicating CRL segments instead of entire CRLs decreases the peak request rates towards server nodes and improves the availability of revocation data. Nodes only need to download the CRL segment which contains the serial number of the certificate that they need to validate. Moreover, server nodes can deal with requests at a faster rate.

Regarding the possible implementation of this revocation data distribution system in other P2P overlay, such as BitTorrent or P2PStream, it is necessary to take into account a series of network features. First of all, all users' data must be stored by owners, while pointers or references to them must be stored by the network. Chord network is an example of network which does not fulfill this requirement, since nodes store content of other owners. Finally, it is important that the P2P network replicates pointers or references in a considerable amount of nodes. Then searches are faster and requests are more distributed.

Chapter 8

Conclusions and Further Work

We started with the research work described in this thesis motivated by the need of providing security to a video streaming platform that operates over a P2P network [57]. Today, in the Internet, P2P networks are widely used to provide services such as video streaming, file sharing, video conferencing, gaming, etc. Many services are implemented using structured P2P networks (P2P overlays). However, in general, security measures must be implemented separately because P2P overlays do not provide security by design. In particular, open P2P overlays have important security problems related to the identity of the nodes, which can affect the maintenance of routing tables, the routing of messages and the operation of reputation systems, among others. These problems make these networks unsuitable for providing commercial services.

We have shown in this thesis that the generation and assignment of nodeIDs are important security problems that P2P overlays should address to provide secure services. We have presented three different identity assignment protocols which leverage the issuance of digital certificates by Trusted Third Parties (TTPs). First, we propose the issuance of an implicit certificate for each user and the use of their public keys to generate secure and verifiable nodeIDs. Second, also using the users' public keys to generate their nodeIDs, we propose the issuance of implicit certificates by two TTPs to provide full user anonymity. Third, we propose the issuance of explicit certificates by two TTPs and a jointly generation of the nodeIDs by a TTP and users to provide identity stability. We have also presented in this thesis a new mechanism to distribute revocation data in a P2P overlay which improves the freshness of certification status data and its accessibility and availability.

The remainder of this chapter summarizes the main results from our research and identifies some future research lines.

8.1 Conclusions

In this thesis, we have designed three different protocols to assign robust identities to nodes in P2P overlays: ICIAPPA, TIAPPA and RIAPPA. The three protocols use public key cryptography and digital certificates to provide basic security services such as authentication, data confidentiality, data integrity or non-repudiation. In terms of performance, it is noteworthy that the deployment of any of these three proposed protocols will only affect during the joining process to the network (bootstrapping), that is, when digital certificates are issued or renewed (because of expiry or key compromise).

ICIAPPA uses implicit certificates to generate and assign pseudorandom nodeIDs. These implicit certificates are issued by a TTP, and contain the nodeID that users will use in the overlay. The TTP controls the access to the network by authenticating users with their real identity (e.g. a public key certificate issued by a public CA). This measure prevents a user can obtain more than one implicit certificate (and hence more than one nodeID). This means that Sybil attacks are avoided, and that robust and effective reputation systems can be used. In this way, attackers cannot wash or improve their reputation dishonestly. When using ICIAPPA, users cannot select their nodeIDs. This mainly avoids that attackers can perform the well-known "Eclipse" attack to isolate a node, and Man-In-The-Middle (MITM) attack to intercept communications to and from the targeted nodes. ICIAPPA assures that nodeIDs are distributed roughly uniformly in the virtual space, ensuring approximately balanced load among nodes, which improves the network performance. Other features of the nodeIDs generated by ICIAPPA are verifiability, revocability, traceability and partial-anonymity. This latest one means that ICIAPPA can assure that users remain anonymous in the overlay, but the TTP can match the real-world identity with the nodeID, so full-anonymity is not achieved.

We have demonstrated that ICIAPPA is a good mechanism to assign secure nodeIDs to users, and its features should be enough for the deployment of commercial services using p2p overlays. Examples of these commercial services may be collaborative P2P video streaming services, online multiplayer games, projects of citizen science, etc. The use of P2P overlays would minimize the investment on infrastructure and would improve the overall performance of the system. However, ICIAPPA does not provide full-anonymity. If this feature is required by users, the protocols TIAPPA and RIAPPA could be used.

The structure of TIAPPA is similar to the one of ICIAPPA, it also issues implicit certificates and the main difference is that TIAPPA uses two TTPs. The reason to use two TTPs is to provide full-anonymity, that is, a single entity (even the TTPs) can match the real identity of the user and the corresponding identity within the overlay. Newly issued implicit certificates are signed by the two TTPs. An external TTP controls the access of users to the network using public digital certificates with real identities of users. The external TTP avoids users can neither change their nodeIDs uncontrollably nor obtain more than one nodeID. This avoids users being able to "wash" their reputation or isolating a node of the rest of the network. The external TTP may charge for the service, if necessary, while maintaining the anonymity of the users within the network as long as their behavior is lawful. The internal TTP is responsible for generating the implicit certificates which contain the identifiers used within the overlay. The internal TTP avoids users can select their nodeIDs for malicious purposes. Both TTPs share a Link Number (LN) to provide traceability and deanonymize users, if strictly necessary. As a result, TIAPPA can avoid Sybil, Eclipse and MITM attacks, ensuring approximately balanced load among nodes.

TIAPPA could be used to provide services that require full anonymity for users. Examples of these may be online auctions, eGovernment services which maintain the anonymity of citizens, e-Health, etc. For instance, one possible service of this kind could be the secure distribution of real-time video content among different operating rooms located in different hospitals around the world, providing full-anonymity to patients.

Implicit certificates, which are based on Elliptic Curve Cryptography (ECC), are used in both ICIAPPA and TIAPPA. As we have shown in the Thesis, this type of certificates provides advantages in terms of performance compared to traditional explicit public certificates. As we discussed in section 2.2.2, three less modular operations and one less scalar multiplication are necessary to construct an ECC-based implicit public key and to verify its ECDSA signature. This computational cost saving will occur every time a node needs to validate a nodeID or encrypt a message using public key cryptography, which is not negligible. Another slight advantage of implicit certificates against explicit certificates is their size. Implicit certificates are three times smaller than explicit certificates independently of the provided security level, despite this might be negligible in P2P networks since we are only saving some hundreds of bits. However, our main reason to use implicit certificates was that they allowed us to generate pseudorandom public keys in an easy way. Unfortunately, these nodeIDs cannot be considered totally stable as every time the implicit certificate is renewed the nodeID changes. If nodeIDs depend on the cryptographic public keys (which is the case of ICIAPPA and TIAPPA), every time a certificate is renewed the associated nodeID will directly change. This can be an important issue for some trust and reputation systems. To provide a solution for those services in which the total stability of nodeIDs is a must, we introduced our third contribution on identity assignment called RIAPPA.

RIAPPA uses explicit certificates and two TTPs. NodeIDs are jointly generated by the owner user and an internal TTP. These nodeIDs are assigned to nodes using explicit certificates, which are signed by the two TTPs (internal and external). An ECC-based blind signature is used to perform this double signature over certificates maintaining the full-anonymity of users.

Omitting all these changes in the type of certificates used and how nodeIDs are generated, at the end RIAPPA provides the same security features than TIAPPA: protection against Sybil, Eclipse, MITM and whitewashing attacks. Generated nodeIDs are pseudorandom, fully-anonymous, verifiable, traceable and revocable. So, the enhancement of RIAPPA regarding TIAPPA is total stability. When using RIAPPA, if a digital certificate is renewed, the associated nodeID does not have to be changed. This can help trust and reputation systems to follow the behavior of a user independently if she changes or not her credentials. RIAPPA is equally suitable for the same scenarios than TIAPPA, but keeping in mind that RIAPPA provides more stability to nodeID, so reputation systems can benefit from that. Unfortunately, the use of explicit certificates would slightly increase the computational cost of the issuance service, which may not matter much in most P2P environments, but this should be considered.

On the other hand, the use of digital certificates, both implicit and explicit, always implies the use of a system to distribute revocation data. The distribution of these data has been a challenge in all types of networks, and many mechanisms have been proposed for that purpose. However, most of them do not provide an efficient distribution of revocation data in P2P overlays. In these networks the amount of revocation data is very high due to the large number of users, and its freshness is usually sacrificed in order to maintain a good performance of the network. For these reasons, we have also proposed a new revocation data distribution mechanism for the Kad network, which can also be implemented in other P2P overlays, such as BitTorrent or P2PStream. We have tried to optimize both the availability of the revocation data and its freshness. In our system, revocation data are distributed in certificate Revocation List (CRL) segments, which are independently issued by a CA and then shared by other nodes of the network. The CA is also able to over-issue those segments in order to improve the freshness of the revocation data. Moreover, unlike the standard technique of CRL segmentation which divides an entire CRL equally to obtain the segments, our proposal also generates the CRL segments independently using the serial number of the revoked certificates and a modular operation.

We have demonstrated that the use of our CRL segments decreases the request rate at the CA and improves the availability of the revocation data. The segmentation of CRLs reduces the size of each distributed piece of revocation data improving the performance of the network and reducing the storage requirements for nodes, and the independent issuance of CRL segments and the use of the over-issuance technique decrease the peak request rate at the CA and server nodes. The CA decides to issue a new CRL segment taking into account the time from the last update and the amount of new revocation data in the segment during this period of time. In Section 7.6, we have analyzed the performance of this mechanism in depth, and we have concluded that the peak request rate decreases exponentially with the number of potential nodes that would distribute the CRL segments, and the size of the segments decreases exponentially with the number of segments in which the revocation data is stored. Moreover, we have also demonstrated that the use of a modular operation to distribute the revoked certificates into the CRL segments provides a uniform distribution.

8.2 Further Work

In this section we explore possible improvements and open research directions based on ideas and results provided in this dissertation.

8.2.1 Implicit Certificates Application

In this thesis we have used implicit certificates in two of the three proposed protocols with intent to leverage its generation scheme to assign robust nodeIDs in P2P overlays. But we have also seen that these certificates have several advantages against explicit certificates, which could be exploited in other kind of networks.

Content-Centric Networks

Content-Centric Networking (CCN) [68] is a new network architecture that focuses on contents access rather than end-point communications; it proposes an environment based on named data instead of named hosts. CCNs only rely on the use of two types of packets, *Interest* and *Data* packets, and a very simple way to operate. Users send out an *Interest* packet containing the name of the content they want to access, intermediate routers forward it to the nodes that may have that content, and the closest node answers with a *Data* packet. Later, answers from other devices are discarded on their way.

This architecture enables scalable, collaborative and pervasive networking. However, the nodes publishing and forwarding contents become quite uncertain. For this reason, security problems were taken into account from the outset [113]. The CCN security scheme relies on ensuring the authenticity, consistency and integrity of the contents. Each data packet contains the publisher's digital signature over the content and a way to retrieve the associated public key (or certificate). However, once the users obtain that public key (or certificate) from the network, the problem is to check its validity.

PKI is the most widely deployed architecture to manage certificates in the Internet, but in distributed environments this architecture may be hard to implement. In addition, intermediate devices (e.g. CCN routers) also tend to check data packets, which can produce significant performance degradation whether the validity process is hard. For these reasons we think that the use of implicit certificates as they were defined by the Standards for Efficient Cryptography Group [103], or leveraging its generation scheme to propose alternative solutions, would be a good way to secure this kind of networks.

Vehicular Ad-Hoc Networks

The open-medium nature of Vehicular Ad Hoc Networks (VANETs) makes it necessary to integrate in these networks security mechanisms such as authentication, message integrity, non-repudiation, confidentiality and privacy [100]. The solution envisioned to achieve these functionalities was to use digital certificates provided by a centralized Certification Authority (CA) [64, 96], but most of the existing proposals use explicit certificates. Taking into account the advantages of implicit certificates in terms of performance and storage against explicit certificates, we think that the use of implicit certificates in these networks could also provide very interesting alternative mechanisms to identify vehicles and provide security.

Internet of Things

The Internet of Things (IoT) concept refers to the use of standard Internet protocols to provide human-to-thing (H2T) or thing-to-thing (T2T) communications. Nowadays, many working groups have focused their research on the (re)design, application, and use of standard Internet technology in the IoT. However, standard security solutions for IP networks are not suitable for most scenarios drawn in this context. Authentication and secure data transmission are also vital in IoT, but the constraints of smart objects demand for more lightweight security mechanisms.

Considering the work carried out by Hummen et. al [65], the use of digital certificates is also a valid solution in these networks. Nevertheless, the use of digital certificates on resource-constrained devices (limited available CPU, ROM, RAM, and energy resources) motivates new challenges such as the processing of long certificate chains, the transmission of large certificates, how to download large revocation lists and so on. For these reasons, we consider that the use of implicit certificates, and of course the use of Elliptic Curve Cryptography (ECC), could be a way to reduce some performance issues.

8.2.2 Revocation Data Distribution

In Chapter 7 we have proposed a new mechanism to distribute revocation data in a P2P overlay to improve its availability and freshness. This mechanism is based on P2P overlay distribution but its way to issue the revocation data could be also used in other networks to achieve similar objectives.

Vehicular Ad-Hoc Networks

In a Vehicular Ad Hoc Network (VANET) where digital certificates are used to identify vehicles and provide certain security mechanisms, if a vehicle is no longer trusted (e.g., due to evidence of malfunction or malicious behavior), its certificates must be revoked. Therefore, the status information of each vehicle has to be made available to other vehicles as soon as possible by means of an efficient mechanism. In this kind of networks, many mechanisms have been proposed in recent years to both distribute and issue revocation data [55, 56]. The main issue is that none of these studies considers issuing CRL segments independently, according to the number of revoked certificates in a period of time or using a modular operation to distribute those certificates into the CRL segments. For this reason, we think that a deeper study of these ideas can be an interesting future research area.

References

[1] aMule Home Page. http://www.amule.org (accessed February 21, 2014). 14, 21

[2] Bitcoin Home Page. https://bitcoin.org (accessed February 24, 2014). 12

[3] BitTorrent Home Page. http://www.bittorrent.com (accessed February 24, 2014). 12

[4] BitTorrent Sync Home Page. http://www.bittorrent.com/sync (accessed February 24, 2014). 12

[5] CoolStreaming Home Page. http://www.coolstreaming.us (accessed February 24, 2014). 12

[6] Edonkey Home Page. http://www.edonkey.co.nr (accessed February 21, 2014). 13

[7] eMule Project Home Page. http://www.emule-project.net (accessed February 21, 2014). 14, 21

[8] Gnutella Home Page. http://rfc-gnutella.sourceforge.net (accessed February 21, 2014). 13

[9] Jamendo Home Page. http://www.jamendo.com (accessed February 24, 2014). 12

[10] Napster Home Page. http://www.napster.com (accessed February 21, 2014). 13

[11] PPLive Home Page. http://join.pplive.com (accessed February 24, 2014). 12

[12] PPStream Home Page. http://pps.tv (accessed February 24, 2014). 12

[13] Skype Home Page. http://www.skype.com (accessed February 24, 2014). 12

[14] SopCast Home Page. http://www.sopcast.org (accessed February 24, 2014). 12

[15] Thawte Home Page. https://www.thawte.com (accessed February 5, 2015). 135

[16] The AVISPA Project. http://www.avispa-project.org (accessed June 20, 2014). 45, 95, 114

[17] TVUnetworks Home Page. http://www.tvunetworks.com (accessed February 24, 2014). 12

[18] Verisign Home Page. https://www.verisign.com (accessed February 24, 2015). 135

[19] Vuze Home Page. http://www.vuze.com (accessed February 26, 2014). 22

[20] Zattoo Home Page. http://zattoo.com (accessed February 24, 2014). 12

[21] Martín Abadi and Roger Needham. Prudent Engineering Practice for Cryptographic Protocols. *IEEE Transactions on Software Engineering*, **22**(1):6–15, 1996. 63, 64, 65, 83, 104

[22] C. Adams, S. Farrell, T. Kause, and T. Mononen. Internet X.509 Public Key Infrastructure Certificate Management Protocol (CMP). RFC 4210 (Proposed Standard), September 2005. 27

[23] Luca Maria Aiello, Marco Milanesio, Giancarlo Ruffo, and Rossano Schifanella. Tempering Kademlia with a Robust Identity Based System. In *Proceedings of the 8th International Conference on Peer-to-Peer Computing*, P2P'08, pages 30–39. IEEE Computer Society, 2008. 31, 119

[24] Luca Maria Aiello, Marco Milanesio, Giancarlo Ruffo, and Rossano Schifanella. An identity-based approach to secure P2P applications with Likir. *Peer-to-Peer Networking and Applications*, **4**:420–438, 2011. 31, 57, 119

[25] Luca Maria Aiello and Giancarlo Ruffo. LotusNet: Tunable privacy for distributed online social network services. *Computer Communications*, **35**(1):75–88, january 2012. 31, 119

[26] American National Standards Institute (ANSI). Random Number Generation Part 1: Overview and Basic Principles, 2006. ANSI X9.82-1-2006. 70, 93

[27] Alessandro Armando, David Basin, Yohan Boichut, Yannick Chevalier, Jorge Cuellar, and Others. The AVISPA Tool for the Automated Validation of Internet Security Protocols and Applications. In *Computer Aided Verification*, **3576** of *Lecture Notes in Computer Science*, pages 281–285. Springer Berlin Heidelberg, 2005. 45

[28] Agapios Avramidis, Panayiotis Kotzanikolaou, Christos Douligeris, and Mike Burmester. Chord-PKI: A distributed trust infrastructure based on P2P networks. *Computer Networks*, **56**(1):378–398, 2012. 3, 37

[29] Ingmar Baumgart and Sebastian Mies. S/Kademlia: A practicable approach towards secure key-based routing. In *Proceedings of the 13th International Conference on Parallel and Distributed Systems*, **2**, pages 1–8. IEEE Computer Society, december 2007. 31

[30] Daniel J. Bernstein and Tanja Lange. eBACS: ECRYPT Benchmarking of Cryptographic Systems. Website. 117

[31] Thomas Beth, Malte Borcherding, and Birgit Klein. Valuation of Trust in Open Networks. In *Proceedings of the European Symposium on Research in Computer Security*, ESORICS'94, pages 3–18, November 1994. 37

[32] Daniel R.L. Brown, Robert Gallant, and Scott A. Vanstone. Provably Secure Implicit Certificate Schemes. In *Financial Cryptography*, **2339** of *Lecture Notes in Computer Science*, pages 156–165. Springer Berlin Heidelberg, 2002. 39

[33] René Brunner. *A performance evaluation of the Kad-protocol*. Master's thesis, Fakultät für Mathematik und Informatik, Universität Mannheim, Germany., november 2006. 111

[34] Kevin R.B. Butler, Sunam Ryu, Patrick Traynor, and Patrick D. McDaniel. Leveraging Identity-Based Cryptography for Node ID Assignment in Structured P2P Systems. *IEEE Transactions on Parallel and Distributed Systems*, **20**(12):1803–1815, december 2009. 31, 33, 118

[35] Qiang Cao, Michael Sirivianos, Xiaowei Yang, and Tiago Pregueiro. Aiding the Detection of Fake Accounts in Large Scale Social Online Services.

In *Proceedings of the 21nd USENIX Security Symposium*, USENIX'12. USENIX Association, june 2012. 34

[36] Miguel Castro, Peter Druschel, Ayalvadi Ganesh, Antony Rowstron, and Dan S. Wallach. Secure Routing for Structured Peer-to-peer Overlay Networks. *SIGOPS Oper. Syst. Rev.*, **36**(SI):299–314, december 2002. 30, 32, 118

[37] Juan Caubet, Oscar Esparza, José L. Muñoz, Juanjo Alins, and Jorge Mata-Díaz. RIAPPA: A Robust Identity Assignment Protocol for P2P overlAys. *Security and Communication Networks*, **7**(12):2743–2760, 2014. 99

[38] Juan Caubet, Carlos Gañan, Oscar Esparza, José L. Munoz, Jorge Mata-Díaz, and Juanjo Alins. Certificate Revocation List Distribution System for the Kademlia Network. *The Computer Journal*, **57**(2):273–280, 2014. 123

[39] David Chaum. Blind Signatures for Untraceable Payments. In *Proceedings of the Advances in Cryptology*, pages 152–156, 1983. 42, 43

[40] Rita Chen and William Yeager. Poblano: A Distributed Trust Model for Peer-to-Peer Networks. JXTA Security Project White Paper, Sun Microsystem, 2002. 37

[41] Robert F. Churchhouse. *Codes and ciphers: Julius Caesar, the Enigma, and the internet*. Cambridge University Press, 2002. 63

[42] Cisco Systems, Inc. Cisco Visual Networking Index: Forecast and Methodology, 2012-2017. http://www.cisco.com/c/en/us/solutions/collateral/service-provider/ip-ngn-ip-next-generation-network/white_paper_c11-481360.html (accessed February 24, 2014). 12

[43] Cisco Systems, Inc. The Zettabyte Era - Trends and Analysis, 2013. http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/VNI_Hyperconnectivity_WP.html (accessed February 24, 2014). 12

[44] Bram Cohen. The BitTorrent Protocol Specification, january 2008. http://www.bittorrent.org/beps/bep_0003.html (accessed February 21, 2014). 14, 22

[45] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 5280 (Updates by RFC 6818), May 2008. 3, 27, 39, 101, 127, 130

[46] David A. Cooper. A Model of Certificate Revocation. In *Proceedings of the 15th Annual Computer Security Applications Conference*, ACSAC'99, pages 256–264, December 1999. 27, 131, 132

[47] Weverton Luis Da Costa Cordeiro, FlÁVio Roberto Santos, Gustavo Huff Mauch, Marinho Pilla Barcelos, and Luciano Paschoal Gaspary. Identity management based on adaptive puzzles to protect P2P systems from Sybil attacks. *Comput. Netw.*, **56**(11):2569–2589, july 2012. 33, 118

[48] The Identity Corner. The problem(s) with OpenID. http://www.untrusted.ca/cache/openid.html (accessed July 2, 2014). 32, 119

[49] Ivan Damgård. Commitment Schemes and Zero-Knowledge Protocols. In *Lectures on Data Security*, **1561** of *Lecture Notes in Computer Science*, pages 63–86. Springer Berlin Heidelberg, 1999. 44

[50] John R. Douceur. The Sybil Attack. In *Proceedings of the First International Workshop on Peer-to-Peer Systems*, IPTPS'02, pages 251–260, London, UK, 2002. Springer-Verlag. 25, 30, 69, 119

[51] Bhavani Elangovan and Bharath. Trust model: Providing witness anonymity in P2P network. In *Proceedings of the third International Conference on Advanced Computing*, ICoAC'11, pages 100–105, Chromepet, Chennai, India, december 2011. 57

[52] Chun-I Fan, Wei-Zhe Sun, and Vincent Shi-Ming Huang. Provably secure randomized blind signature scheme based on bilinear pairing. *Computers & Mathematics with Applications*, **60**(2):285–293, 2010. 115, 117

[53] Romano Fantacci, Leonardo Maccari, Matteo Rosi, Luigi Chisci, Luca Maria Aiello, and Marco Milanesio. Avoiding Eclipse Attacks on Kad/Kademlia: An Identity Based Approach. In *Proceedings of the IEEE International Conference on Communications*, ICC'09, pages 983–987. IEEE Press, june 2009. 31, 57, 119

[54] Eric J. Friedman and Paul Resnick. The Social Cost of Cheap Pseudonyms. *Journal of Economics & Management Strategy*, **10**(2):173–199, 2001. 31

[55] Carlos Gañán, Jose L. Muñoz, Oscar Esparza, Jonathan Loo, Jorge Mata-Díaz, and Juanjo Alins. BECSI: bandwidth efficient certificate status information distribution mechanism for vanets. *Mobile Information Systems*, **9**(4):347–370, 2013. 146

[56] Carlos Gañán, Jose L. Muñoz, Oscar Esparza, Jorge Mata-Díaz, Juan Hernández-Serrano, and Juanjo Alins. COACH: collaborative certificate status checking mechanism for vanets. *J. Network and Computer Applications*, **36**(5):1337–1351, 2013. 146

[57] Carlos Gañán, Juan Caubet, Sergi Reñé, Jorge Mata-Díaz, Juanjo Alins, and Óscar Esparza. NeuroCast: Adaptive Multi-source P2P Video Streaming Application for Wireless Networks. In *Wired/Wireless Internet Communications*, **6649** of *Lecture Notes in Computer Science*, pages 272–284. Springer Berlin Heidelberg, 2011. 139

[58] Oded Goldreich. Secure Multi-Party Computation, 2002. http://www.wisdom.weizmann.ac.il/ oded/pp.html. 44

[59] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The Knowledge Complexity of Interactive Proof Systems. *SIAM J. Comput.*, **18**(1):186–208, February 1989. 44

[60] Abhilash Gummadi and Jong P. Yoon. Modeling Group Trust For Peer-to-Peer Access Control. In *Proceedings of International Workshop on Database and Expert Systems Applications*, pages 971–978. IEEE Computer Society, August 2004. 37

[61] Darrel Hankerson, Alfred J. Menezes, and Scott Vanstone. *Guide to Elliptic Curve Cryptography*. Springer Verlag New York, 2003. 38

[62] Ryan Henry and Ian Goldberg. Formalizing Anonymous Blacklisting Systems. In *Proceedings of the IEEE Symposium on Security and Privacy*, SP'11, pages 81–95, Berkeley, CA, USA, May 2011. IEEE Computer Society. 78

[63] Jun Huang, Zhao Wang, Zhao Qiu, and Mingrui Chen. Theoretical Analysis of Issuing Mechanism in Distributive Digital Certificate Revocation List. In *Proceedings of the International Conference on Computer and Electrical Engineering*, ICCEE'08, pages 199–203. IEEE Computer Society, December 2008. 36

[64] Jean-Pierre Hubaux, S. Capkun, and Luo Jun. The security and privacy of smart vehicles. *IEEE Security and Privacy*, **2**(3):49–55, May 2004. 145

[65] René Hummen, Jan H. Ziegeldorf, Hossein Shafagh, Shahid Raza, and Klaus Wehrle. Towards viable certificate-based authentication for the internet of things. In *Proceedings of the 2nd ACM workshop on Hot topics on wireless network security and privacy*, pages 37–42. ACM, 2013. 145

[66] Sun Microsystems Inc. JXTA v2.0 Protocols Specification, 2007. http://java.net/projects/jxta-spec (accessed January 9, 2015). 23, 37

[67] ipoque. Internet Study 2008/2009. http://www.ipoque.com/sites/default/files/mediafiles/documents/internet-study-2008-2009.pdf (accessed February 21, 2014). 15

[68] Van Jacobson, Diana K. Smetters, James D. Thornton, Michael F. Plass, Nicholas H. Briggs, and Rebecca L. Braynard. Networking Named Content. In *Proceedings of the 5th ACM International Conference on Emerging Networking Experiments and Technologies*, CoNEXT'09, pages 1–12, December 2009. 144

[69] Fuh-Gwo Jeng, Tzer-Long Chen, and Tzer-Shyong Chen. An ECC-Based Blind Signature Scheme. *Journal Of Networks*, **5**(8):921–928, 2010. 44

[70] Wen Ji, Shoubao Yang, Dong Wei, and Weina Lu. GARM: A Group - Anonymity Reputation Model in Peer-to-Peer System. In *Proceedings of the 6th International Conference on Grid and Cooperative Computing*, GCC'07, pages 481–488, Washington, DC, USA, 2007. IEEE Computer Society. 57

[71] Don Johnson and Alfred Menezes. The Elliptic Curve Digital Signature Algorithm (ECDSA). Technical Report CORR 99-34, Dept. of C&O, University of Waterloo, Canada, 1999. 39

[72] Fabius Klemm, Sarunas Girdzijauskas, Jean-Yves Le Boudec, and Karl Aberer. On Routing in Distributed Hash Tables. In *Proceedings of the 7th IEEE International Conference on Peer-to-Peer Computing*, P2P'07, pages 113–122, Los Alamitos, CA, USA, 2007. IEEE Computer Society. 24

[73] Neal Koblitz. Elliptic Curve Cryptosystems. *Mathematics of computation*, **48**(177):203–209, 1987. 37

[74] Paul C. Kocher. On Certificate Revocation and Validation. In *Financial Cryptography*, **1465** of *Lecture Notes in Computer Science*, pages 172–177. Springer Berlin Heidelberg, 1998. 28

[75] Santosh Kulkarni. Badumna Network Suite: A decentralized network engine for Massively Multiplayer Online applications. In *Proceedings of the 9th IEEE International Conference on Peer-to-Peer Computing*, P2P'09, pages 178–183, Seattle, WA, USA, september 2009. 12

[76] Stefan Köpsell, Rolf Wendolsky, and Hannes Federrath. Revocable Anonymity. In *Emerging Trends in Information and Communication Security*, **3995** of *Lecture Notes in Computer Science*, pages 206–220. Springer Berlin Heidelberg, 2006. 78

[77] François Lesueur, Ludovic Mé, and Valérie Viet Triem Tong. A Sybil-proof Distributed Identity Management for P2P Networks. In *Proceedings of the IEEE Symposium on Computers and Communications*, ISCC'08, pages 246–253. IEEE Computer Society, july 2008. 36

[78] libtorrent blog. DHT security. http://blog.libtorrent.org/2012/12/dht-security (accessed April 3, 2014). 51

[79] Thomas Locher, David Mysicka, Stefan Schmid, and Roger Wattenhofer. Poisoning the KAD Network. In *Distributed Computing and Networking*, **5935** of *Lecture Notes in Computer Science*, pages 195–206. Springer Berlin, Heidelberg, 2010. 48

[80] Andrew Loewenstern and Arvid Norberg. BitTorrent DHT Protocol Specification, january 2008. http://bittorrent.org/beps/bep_0005.html (accessed February 21, 2014). 14, 22

[81] Chuiwei Lu. Detection and Defense of Identity Attacks in P2P Network. In *Advances in Computation and Intelligence*, **5821** of *Lecture Notes in Computer Science*, pages 500–507. Springer-Verlag Berlin Heidelberg, 2009. 33, 57, 118

[82] Ben Lynn. *On the Implementation of Pairing-based Cryptosystems*. PhD thesis, Stanford University, June 2007. 39

[83] Sergio Marti and Hector Garcia-Molina. Identity Crisis: Anonymity vs. Reputation in P2P Systems. In *Proceedings of the 3rd International Conference on Peer-to-Peer Computing*, P2P'03, pages 134–141. IEEE Computer Society, 2003. 53

[84] Sergio Marti and Hector Garcia-Molina. Taxonomy of Trust: Categorizing P2P Reputation Systems. *Computer Networks*, **50**(4):472–84, 2006. 26

[85] Petar Maymounkov and David Mazières. Kademlia: A Peer-to-Peer Information System Based on the XOR Metric. In *Proceedings of the 1st International Workshop on Peer-to Peer Systems*, IPTPS'02, pages 53–65, march 2002. 14, 19, 49

[86] Alfred J. Menezes, Scott A. Vanstone, and Paul C. Van Oorschot. *Handbook of Applied Cryptography*. CRC Press, Inc., Boca Raton, FL, USA, 1st edition, 1996. 39

[87] Roberto Tamassia Michael T. Goodrich and Andrew Schwerin. Implementation of an authenticated dictionary with skip lists and commutative hashing. In *Proceedings of the II DARPA Information Survivability Conference & Exposition*, **2** of *DISCEX'01*, pages 68–82. IEEE Computer Society, June 2001. 28

[88] Victor S. Miller. Use of Elliptic Curves in Cryptography. In *Advances in Cryptology: Proceedings of CRYPTO '85*, **218** of *Lecture Notes in Computer Science*, pages 417–426. Springer Berlin Heidelberg, 1986. 37

[89] D. Mills, J. Martin, J. Burbank, and W. Kasch. Network Time Protocol Version 4: Protocol and Algorithms Specification. RFC 5905 (Standards Track), June 2010. 64, 80

[90] Matei Ciobanu Morogan and Sead Muftic. Certificate Revocation System Based on Peer-to-Peer CRL Distribution. In *Proceedings of the 9th International Conference on Distributed Multimedia Systems*, DMS'03, September 2003. 36

[91] Moni Naor and Kobbi Nissim. Certificate Revocation and Certificate Update. *IEEE Journal on Selected Areas in Communications*, **18**(4):561–570, April 2000. 28

[92] Arvid Norberg. BitTorrent DHT Security Extension. http://bittorrent.org/beps/bep_0042.html (accessed March 31, 2014). 23, 49, 50

[93] The National Security Agency (NSA). Suite B Cryptography/Cryptographic Interoperability, 2010. http://www.nsa.gov/ia/programs/suiteb_cryptography (accessed March 14, 2014). 38

[94] National Institute of Standards and Technology (NIST). Recommendation for Random Number Generation Using Deterministic Random Bit Generators, January 2012. NIST Special Publication 800-90A. 70, 93

[95] National Institute of Standards and Technology (NIST). Secure Hash Standard (SHS), March 2012. FIPS PUB 180-4. 70

[96] P. Papadimitratos, L. Buttyan, Jean-Pierre Hubaux, F. Kargl, A. Kung, and Maxim Raya. Architecture for secure and private vehicular communications. In *Proceedings of the 7th International Conference on ITS Telecommunications*, ITST'07, pages 1–6, June 2007. 145

[97] Leon A. Pintsov and Scott A. Vanstone. Postal Revenue Collection in the Digital Age. In *Financial Cryptography*, **1962** of *Lecture Notes in Computer Science*, pages 105–120. Springer Berlin Heidelberg, 2001. 39

[98] C. Greg Plaxton, Rajmohan Rajaraman, and Andréa W. Richa. Accessing Nearby Copies of Replicated Objects in a Distributed Environment. *Theory of Computing Systems*, **32**(3):241–280, 1999. 14, 18

[99] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker. A Scalable Content-Addressable Network. In *Proceedings of the ACM Conference on Applications, Technologies, Architectures and Protocols for Computer Communication (SIGCOMM)*, pages 161–172, San Diego, CA, USA, 2001. 14, 15, 48

[100] Maxim Raya and Jean-Pierre Hubaux. The security of vehicular ad hoc networks. In *Proceedings of the 3rd ACM workshop on Security of ad hoc and sensor networks*, SASN'05, pages 11–21, 2005. 145

[101] Certicom Research. Standards for Efficient Cryptography 1 (SEC 1): Elliptic Curve Cryptography, May 2009. Version 2.0. 70, 93

[102] Certicom Research. Standards for Efficient Cryptography 2 (SEC 2): Recommended Elliptic Curve Domain Parameters, January 2010. Version 2.0. 71

[103] Certicom Research. Standards for Efficient Cryptography 4 (SEC 4): Elliptic Curve Qu-Vanstone Implicit Certificate Scheme (ECQV), April 2014. Version 1.2. 40, 62, 144

[104] Ronald L. Rivest, Adi Shamir, and Leonard Adleman. A Method for Obtaining Digital Signatures and Public-key Cryptosystems. *Commun. ACM*, **21**(2):120–126, February 1978. 38

[105] Hosam Rowaihy, William Enck, Patrick McDaniel, and Thomas La Porta. Limiting Sybil Attacks in Structured P2P Networks. In *Proceedings of the 26th IEEE International Conference on Computer Communications*, pages 2596–2600, Anchorage, Alaska, USA, may 2007. 25, 32, 57, 118

[106] Antony Rowstron and Peter Druschel. Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems. In *Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms*, pages 329–350, 2001. 14, 18, 49, 51

[107] Ehsan Saboori and Maghsoud Abbaspour. Dual-Path Peer-to-Peer Anonymous Approach. In *Proceedings of the 16th IEEE International Conference on Parallel and Distributed Systems*, ICPADS'10, pages 835–838, Washington, DC, USA, 2010. IEEE Computer Society. 57

[108] S. Santesson, M. Myers, R. Ankney, A. Malpani, S. Galperin, and C. Adams. X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP. RFC 6960 (Standards Track), June 2013. 28

[109] N. Saxena, G. Tsudik, and J.H. Yi. Threshold cryptography in P2P and MANETs: The case of access control. *Computer Networks*, **51**(12):3632–3649, 2007. 53

[110] Lu Shi, Shucheng Yu, Wenjing Lou, and Thomas Hou. SybilShield: An Agent-Aided Social Network-Based Sybil Defense among Multiple Communities. In *Proceedings of the 34th IEEE International Conference on Computer Communications*, pages 1034–1042, april 2013. 35

[111] Atul Singh, Tsuen-Wan "Johnny" Ngan, Peter Druschel, and Dan S. Wallach. Eclipse Attacks on Overlay Networks: Threats and Defenses. In *Proceedings of the 25th IEEE International conference on Computer communications*, pages 1–12, Barcelona, Spain, april 2006. 25

[112] Nitu Singh and Sumanjit Das. Cryptanalysis of Blind Signature Schemes. *International Journal of Computer Applications*, **71**(19):39–43, June 2013. 44, 79

[113] Diana K. Smetters and Van Jacobson. Securing Network Content. Technical Report TR-2009-1, PARC, October 2009. 144

[114] Mudhakar Srivatsa and Ling Liu. Vulnerabilities and Security Threats in Structured Overlay Networks: A Quantitative Analysis. In *Proceedings of the 20th Annual Computer Security Applications Conference*, pages 252–261, december 2004. 30, 118

[115] Moritz Steiner, Taoufik En-Najjary, and Ernst W. Biersack. A Global View of KAD. In *Proceedings of the 7th ACM SIGCOMM Internet Measurement Conference*, IMC'07, pages 117–122, New York, NY, USA, 2007. ACM. 14

[116] Ion Stoica, Robert Morris, David R. Karger, M. Frans Kaashoek, and Hari Balakrishman. Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications. In *Proceedings of the ACM Conference on Applications, Technologies, Architectures and Protocols for Computer Communication (SIGCOMM)*, pages 149–160, San Diego, CA, USA, 2001. 14, 17, 36, 49

[117] Jonathan Taverne, Armando Faz-Hernández, Diego F. Aranha, Francisco Rodríguez-Henríquez, Darrel Hankerson, and Julio López. Software Implementation of Binary Elliptic Curves: Impact of the Carry-Less Multiplier on Scalar Multiplication. In *Cryptographic Hardware and Embedded Systems – CHES 2011*, **6917** of *Lecture Notes in Computer Science*, pages 108–123. Springer Berlin Heidelberg, 2011. 117

[118] Nguyen Tran, Jinyang Li, Lakshminarayanan Subramanian, and Sherman S.M. Chow. Optimal Sybil-resilient node admission control. In *Proceedings of the 30th IEEE International Conference on Computer Communications*, INFOCOM'11, pages 3218–3226, Shanghai, China, april 2011. IEEE Press. 34, 35, 57

[119] Zied Trifa and Maher Khemakhem. Taxonomy of Structured P2P Overlay Networks Security Attacks. *World Academy of Science, Engineering and Technology*, **6**(4):468 – 475, 2012. 15

[120] Andreea Visan, Florin Pop, and Valentin Cristea. Decentralized Trust Management in Peer-to-Peer Systems. In *Proceedings of the 10th International*

Symposium on Parallel and Distributed Computing, ISPDC'11, pages 232–239, Washington, DC, USA, 2011. IEEE Computer Society. 57

[121] Dan S. Wallach. A Survey of Peer-to-Peer Security Issues. In *Proceedings of the Mext-NSF-JSPS international conference on Software security: theories and systems*, ISSS'02, pages 42–57, Tokyo, Japan, 2002. Springer-Verlag Berlin, Heidelberg. 2, 15

[122] Gang Wang, Tristan Konolige, Christo Wilson†, Xiao Wang, Haitao Zheng, and Ben Y. Zhao. You are How You Click: Clickstream Analysis for Sybil Detection. In *Proceedings of the 22nd USENIX Security Symposium*, USENIX'13, pages 241–256. USENIX Association, august 2013. 34

[123] Fatos Xhafa, Raul Fernandez, Thanasis Daradoumis, Leonard Barolli, and Santi Caballé. Improvement of jxta protocols for supporting reliable distributed applications in p2p systems. In *Network-Based Information Systems*, **4658** of *LNCS*, pages 345–354. Springer-Verlag Berlin, Heidelberg, 2007. 24

[124] Jilong Xue, Zhi Yang, Xiaoyong Yang, Xiao Wang, Lijiang Chen, and Yafei Dai. VoteTrust: Leveraging Friend Invitation Graph to Defend against Social Network Sybils. In *Proceedings of the 34th IEEE International Conference on Computer Communications*, pages 2400–2408, april 2013. 35

[125] Yu Yang and Lan Yang. A Survey of Peer-to-Peer Attacks and Counter Attacks. In *Proceedings of the International Conference on Security and Management*, SAM'12, pages 176–182, Las Vegas, NV, USA, july 2012. 15

[126] Gao Ying and Zhan Jiang. Research on CRL distribution in P2P systems. In *Proceedings of the 2nd IEEE International Conference on Computer Science and Information Technology*, ICCSIT'09, pages 574–577. IEEE Computer Society, August 2009. 36

[127] Haifeng Yu, Phillip B. Gibbons, Michael Kaminsky, and Feng Xiao. SybilLimit: A Near-Optimal Social Network Defense against Sybil Attacks. *IEEE/ACM Transactions on Networking*, **18**(3):885–898, 2010. 34, 35, 57, 118

[128] Haifeng Yu, Michael Kaminsky, Phillip B. Gibbons, and Abraham Flaxman. SybilGuard: Defending Against Sybil Attacks via Social Networks. *IEEE/ACM Transactions on Networking*, **16**(3):576–589, 2008. 34, 57

[129] Ben Y. Zhao, Ling Huang, Jeremy Stribling, Sean C.Rhea, Anthony D. Joseph, and John D. Kubiatowicz. Tapestry: A Resilient Global-Scale Overlay for Service Deployment. *IEEE Journal on Selected Areas in Communications*, **22**(1):41–53, 2004. 14, 18

Appendix A

ASN.1 RIAPPA Certificate Syntax

```
Certificate ::= SEQUENCE {
        tbsSignedCertificate            TBSSignedCertificate,
        externalSignatureAlgorithm      AlgorithmIdentifier,
        externalSignatureValue          BIT STRING }

TBSSignedCertificate ::= SEQUENCE {
        tbsCertificate                  TBSCertificate,
        internalSignatureAlgorithm      AlgorithmIdentifier,
        internalSignatureValue          BIT STRING }

TBSCertificate ::= SEQUENCE {
        serialNumber                    CertificateSerialNumber,
        serviceIdentifier               ServiceIdentifier,
        internalSignature               AlgorithmIdentifier,
        internalIssuer                  Name,
        externalSignature               AlgorithmIdentifier,
        externalIssuer                  Name,
        validityPeriod                  Validity,
        subject                         NodeID,
        subjectPublicKeyInfo            SubjectPublicKeyInfo }

CertificateSerialNumber ::= INTEGER

ServiceIdentifier ::= INTEGER

AlgorithmIdentifier ::= SEQUENCE {
        algorithm               OBJECT IDENTIFIER,
        parameters              ANY DEFINED BY algorithm OPTIONAL }

Name ::= CHOICE {
        rdnSequence             RDNSequence }

RDNSequence ::= SEQUENCE OF RelativeDistinguishedName

RelativeDistinguishedName ::= SET SIZE (1..MAX) OF
    AttributeTypeAndValue

AttributeTypeAndValue ::= SEQUENCE {
        type                    AttributeType,
        value                   AttributeValue }

AttributeType ::= OBJECT IDENTIFIER

AttributeValue ::= ANY -- DEFINED BY AttributeType

DirectoryString ::= CHOICE {
        teletexString           TeletexString (SIZE (1..MAX)),
        printableString         PrintableString (SIZE (1..MAX)),
        universalString         UniversalString (SIZE (1..MAX)),
        utf8String              UTF8String (SIZE (1..MAX)),
        bmpString               BMPString (SIZE (1..MAX)) }

Validity ::= SEQUENCE {
        notBefore               UTCTime,
        notAfter                UTCTime }

NodeID ::= INTEGER

SubjectPublicKeyInfo ::= SEQUENCE {
        algorithm               AlgorithmIdentifier,
        subjectPublicKey        BIT STRING }
```

Appendix B

The AVISPA Tool Simulations

B.1 ICIAPPA

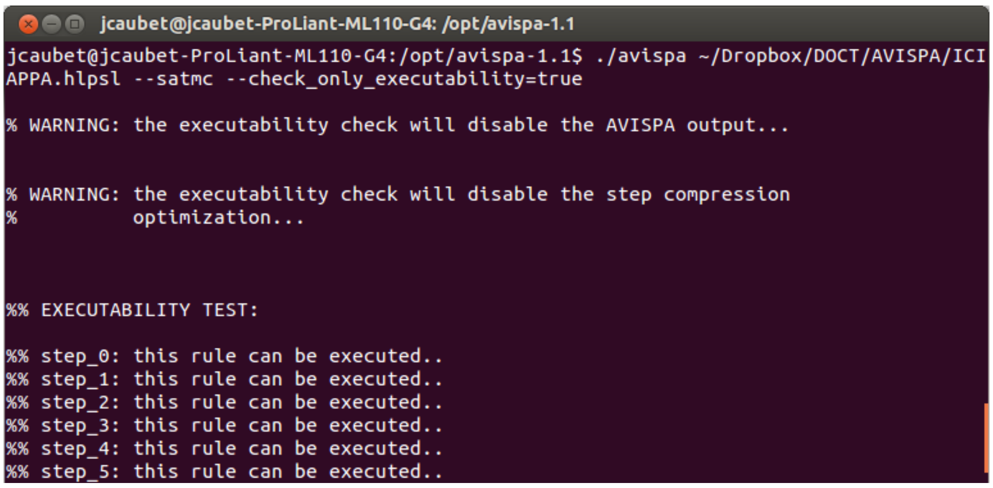
```
%% HLPSL:
% An Implicit Certificate-based Identity Assignment Protocol
% for the P2P overlAys
%%%%%%%%%%%%%%% Created by Juan Caubet %%%%%%%%%%%%%%%%%%

role user (
  U, CA                    : agent,
  Ku, Kca                  : public_key,
  Hash                     : hash_func,
  SND_CAU, RCV_CAU         : channel(dy))

played_by U
def=

  local
    State                          : nat,
    Tx, Ux, Nx, Info_u, N          : text,
    C, Z, H, S, Px                 : message

  const
    cert_u                         : text

init State := 0

transition
```

```
1. State = 0 /\ RCV_CAU(start) =|>
 State' := 2 /\ Tx' := new()
             /\ SND_CAU({cert_u.{U.CA.Tx'}_inv(Ku)}_Kca)
             /\ witness(U,CA,ca_user_timestamp,Tx')

2. State = 2 /\ RCV_CAU({{CA.U.Tx.C'.Ux'}_inv(Kca)}_Ku) =|>
 State' := 4 /\ Nx' := new()
             /\ SND_CAU({{U.CA.Tx.Nx'}_inv(Ku)}_Kca)

3. State = 4 /\ RCV_CAU({{CA.U.Tx.S'.N'.Info_u'.Z'}_inv(Kca)}_Ku)
             /\ C = Hash(N'.Ux) /\ Z' = Hash(Nx.N')
             /\ H' = Hash(Info_u'.Z') /\ S' = Hash(H'.Kca) =|>
 State' := 6 /\ Px' := Hash(S'.Kca)
             /\ SND_CAU({{U.CA.Tx.Px'}_inv(Ku)}_Kca)
             /\ request(U,CA,user_ca_timestamp,Tx)

end role

%---------------------------------------------------------------

role ca (
  CA, U                  : agent,
  Kca, Ku                : public_key,
  Hash                   : hash_func,
  SND_UCA, RCV_UCA       : channel(dy))

played_by CA
def=

  local
    State                                : nat,
    Cert_u, Tx, N, Ux, Nx, Info_u        : text,
    C, Z, H, S, Px                       : message

init State := 1

transition

1. State = 1 /\ RCV_UCA({Cert_u'.{U.CA.Tx'}_inv(Ku)}_Kca) =|>
 State' := 3 /\ N' := new() /\ Ux' := new() /\ C' := Hash(N'.Ux')
```

```
                /\ SND_UCA({{CA.U.Tx'.C'.Ux'}_inv(Kca)}_Ku)
                /\ witness(CA,U,user_ca_timestamp,Tx')

2. State = 3  /\ RCV_UCA({{U.CA.Tx.Nx'}_inv(Ku)}_Kca) =|>
 State' := 5  /\ Info_u' := new() /\ Z' := Hash(Nx'.N)
                /\ H' := Hash(Info_u'.Z') /\ S' := Hash(H'.Kca)
                /\ SND_UCA({{CA.U.Tx.S'.N.Info_u'.Z'}_inv(Kca)}_Ku)
                /\ secret(S',s,{CA,U})

3. State = 5  /\ RCV_UCA({{U.CA.Tx.Px'}_inv(Ku)}_Kca)
                /\ Px' = Hash(S.Kca) =|>
 State' := 7  /\ request(CA,U,ca_user_timestamp,Tx)

end role

%-----------------------------------------------------------------------

role session(
  U, CA          : agent,
  Ku, Kca        : public_key,
  Hash           : hash_func)

def=

  local
    SCAU, RCAU, SUCA, RUCA      : channel (dy)

composition

        user (U,CA,Ku,Kca,Hash,SCAU,RCAU)
     /\ ca (CA,U,Kca,Ku,Hash,SUCA,RUCA)

end role

%-----------------------------------------------------------------------

role environment()

def=

  const
```

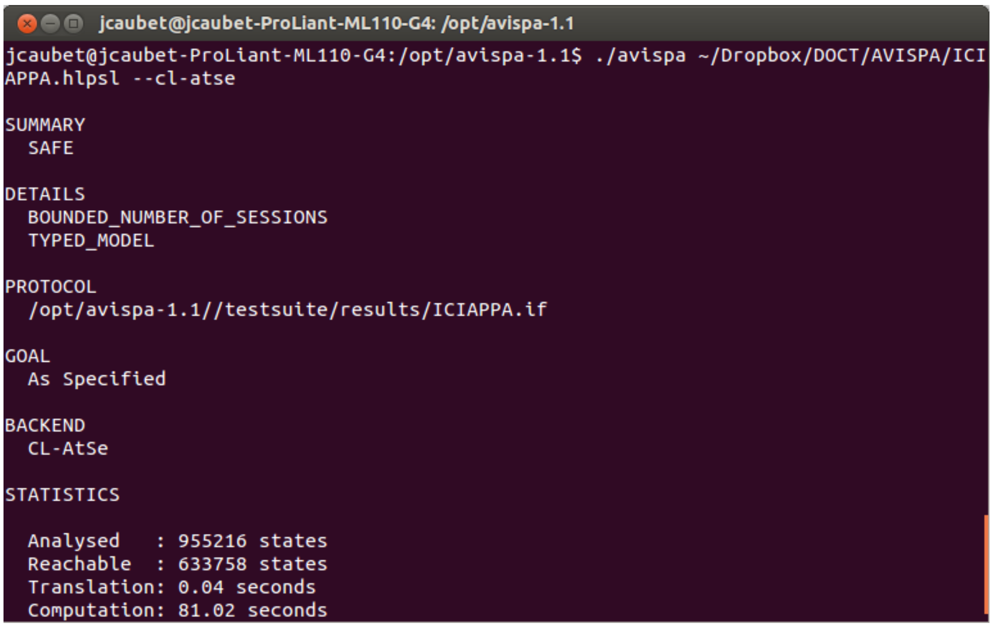
```
    s, user_ca_timestamp, ca_user_timestamp       : protocol_id,
    u, ca, i                                      : agent,
    kca, ku, ki                                   : public_key,
    fhash                                         : hash_func

intruder_knowledge = {u,ca,i,ku,kca,ki,inv(ki),fhash}

composition

           session(u,ca,ku,kca,fhash)
        /\ session(u,ca,ku,kca,fhash)
        /\ session(i,ca,ki,kca,fhash)

end role

%————————————————————————————————————————————————————————————————————————

goal

%U authenticates CA on user_ca_timestamp
  authentication_on user_ca_timestamp
%CA authenticates U on ca_user_timestamp
  authentication_on ca_user_timestamp

  secrecy_of s

end goal

environment()
```

B.1.1 Results



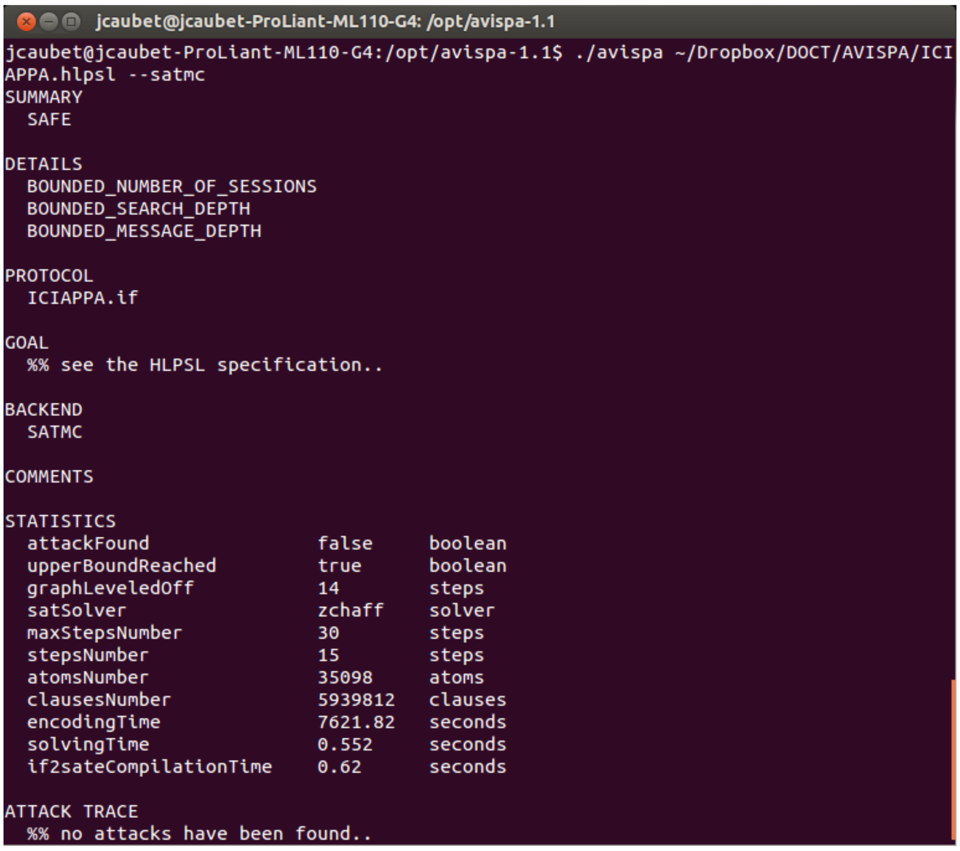
```
jcaubet@jcaubet-ProLiant-ML110-G4: /opt/avispa-1.1
jcaubet@jcaubet-ProLiant-ML110-G4:/opt/avispa-1.1$ ./avispa ~/Dropbox/DOCT/AVISPA/ICI
APPA.hlpsl --satmc --check_only_executability=true

% WARNING: the executability check will disable the AVISPA output...


% WARNING: the executability check will disable the step compression
%          optimization...



%% EXECUTABILITY TEST:

%% step_0: this rule can be executed..
%% step_1: this rule can be executed..
%% step_2: this rule can be executed..
%% step_3: this rule can be executed..
%% step_4: this rule can be executed..
%% step_5: this rule can be executed..
```

Figure B.1: ICIAPPA - AVISPA output of the executability test.



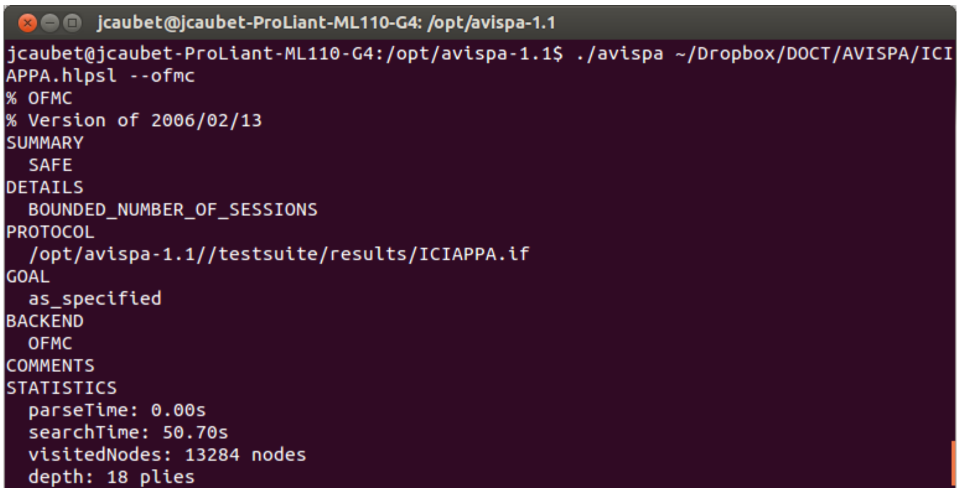
```
jcaubet@jcaubet-ProLiant-ML110-G4: /opt/avispa-1.1
jcaubet@jcaubet-ProLiant-ML110-G4:/opt/avispa-1.1$ ./avispa ~/Dropbox/DOCT/AVISPA/ICI
APPA.hlpsl --cl-atse

SUMMARY
  SAFE

DETAILS
  BOUNDED_NUMBER_OF_SESSIONS
  TYPED_MODEL

PROTOCOL
  /opt/avispa-1.1//testsuite/results/ICIAPPA.if

GOAL
  As Specified

BACKEND
  CL-AtSe

STATISTICS

  Analysed   : 955216 states
  Reachable  : 633758 states
  Translation: 0.04 seconds
  Computation: 81.02 seconds
```

Figure B.2: ICIAPPA - AVISPA output using CL-AtSe back-end.



```
jcaubet@jcaubet-ProLiant-ML110-G4: /opt/avispa-1.1
jcaubet@jcaubet-ProLiant-ML110-G4:/opt/avispa-1.1$ ./avispa ~/Dropbox/DOCT/AVISPA/ICI
APPA.hlpsl --satmc
SUMMARY
  SAFE

DETAILS
  BOUNDED_NUMBER_OF_SESSIONS
  BOUNDED_SEARCH_DEPTH
  BOUNDED_MESSAGE_DEPTH

PROTOCOL
  ICIAPPA.if

GOAL
  %% see the HLPSL specification..

BACKEND
  SATMC

COMMENTS

STATISTICS
  attackFound                false      boolean
  upperBoundReached          true       boolean
  graphLeveledOff            14         steps
  satSolver                  zchaff     solver
  maxStepsNumber             30         steps
  stepsNumber                15         steps
  atomsNumber                35098      atoms
  clausesNumber              5939812    clauses
  encodingTime               7621.82    seconds
  solvingTime                0.552      seconds
  if2sateCompilationTime     0.62       seconds

ATTACK TRACE
  %% no attacks have been found..
```

Figure B.3: ICIAPPA - AVISPA output using SATMC back-end.



```
jcaubet@jcaubet-ProLiant-ML110-G4: /opt/avispa-1.1
jcaubet@jcaubet-ProLiant-ML110-G4:/opt/avispa-1.1$ ./avispa ~/Dropbox/DOCT/AVISPA/ICI
APPA.hlpsl --ofmc
% OFMC
% Version of 2006/02/13
SUMMARY
  SAFE
DETAILS
  BOUNDED_NUMBER_OF_SESSIONS
PROTOCOL
  /opt/avispa-1.1//testsuite/results/ICIAPPA.if
GOAL
  as_specified
BACKEND
  OFMC
COMMENTS
STATISTICS
  parseTime: 0.00s
  searchTime: 50.70s
  visitedNodes: 13284 nodes
  depth: 18 plies
```

Figure B.4: ICIAPPA - AVISPA output using OFMC back-end.

B.2 TIAPPA

```
% HLPSL:
% A TWO-LEVEL IDENTITY ASSIGNMENT PROTOCOL FOR P2P OVERLAYS %
%%%%%%%%%%%%%% Created by Juan Caubet %%%%%%%%%%%%%%%
role user (
  U, A, B                                 : agent,
  Ku1, Ka                                 : public_key,
  Hash                                    : hash_func,
  SND_AU, RCV_AU, SND_BU, RCV_BU          : channel(dy))

played_by U
def=

  local
    State                                 : nat,
    Rid, Sid, Tu, Nx, Ub1, Ix, Bx         : text,
    Cu2, Cb, Ub, Px, Sx, Zx, Hx           : message,
    Kb, Ku2                               : public_key

  const
    cu1                                   : text

init State := 0

transition

1. State = 0 /\ RCV_AU(start) =|>
 State' := 2 /\ Rid' := new() /\ Sid' := new()
             /\ SND_AU({U.A.Rid'.Sid'.cu1.
             {Hash(U.A.Rid'.Sid'.cu1)}_inv(Ku1)}_Ka)
             /\ witness(U,A,alpha_user_rid,Rid')
             /\ secret(Rid',requestid,{U,A,B})
             /\ secret(Sid',serviceid,{U,A})

2. State = 2 /\ RCV_AU({A.U.Rid.Kb'.
             {Hash(A.U.Rid.Kb')}_inv(Ka)}_Ku1) =|>
 State' := 4 /\ Ku2' := new() /\ Tu' := new()
             /\ SND_AU({U.A.Rid.{B.Ku2'.Tu'.
{Hash(B.Ku2'.Tu')}_inv(Ku2')}_Kb'.{Hash(U.A.Rid.{B.Ku2'.Tu'.
{Hash(B.Ku2'.Tu')}_inv(Ku2')}_Kb')}_inv(Ku1)}_Ka)
```

```
                /\ witness(U,B,beta_user_ku2,Ku2')
                /\ secret(Ku2',userkey2,{U,B})
                /\ secret(Tu',timestamp,{U,B})

3. State = 4 /\ RCV_AU({A.U.Rid.{B.Tu.Cb'.Ub'.
{Hash(B.Tu.Cb'.Ub')}_inv(Kb)}_Ku2.{Hash(A.U.Rid.
{B.Tu.Cb'.Ub'.{Hash(B.Tu.Cb'.Ub')}_inv(Kb)}_Ku2)}_inv(Ka)}_Ku1) =|>
 State' := 6 /\ Nx' := new()
                /\ SND_AU({U.A.Rid.{Tu.B.Nx'.
{Hash(Tu.B.Nx')}_inv(Ku2)}_Kb.{Hash(U.A.Rid.{Tu.B.Nx'.
{Hash(Tu.B.Nx')}_inv(Ku2)}_Kb)}_inv(Ku1)}_Ka)

4. State = 6 /\ RCV_AU({A.U.Rid.{B.Px'.Tu.Sx'.Ub1'.Ix'.Zx'.Bx'.
{Hash(B.Px'.Tu.Sx'.Ub1'.Ix'.Zx'.Bx')}_inv(Kb)}_Ku2.
{Hash(A.U.Rid.{B.Px'.Tu.Sx'.Ub1'.Ix'.Zx'.Bx'.
{Hash(B.Px'.Tu.Sx'.Ub1'.Ix'.Zx'.Bx')}_inv(Kb) }_Ku2)}_inv(Ka)}_Ku1)
                /\ Px' = Hash(Zx'.Ix'.Bx') /\ Ub = Hash(Ub1') =|>
 State' := 8 /\ Hx' := Hash(Ix'.Zx')
                /\ SND_BU({Px'.B.Rid.Hx'.
                {Hash(Px'.B.Rid.Hx')}_inv(Ku2)}_Kb)

end role

%------------------------------------------------------------

role alpha (
  A, B, U                                   : agent,
  Ka, Kb, Ku1                               : public_key,
  Kab                                       : symmetric_key,
  Hash                                      : hash_func,
  SND_UA, RCV_UA, SND_BA, RCV_BA            : channel(dy))

played_by A
def=

  local
    State                                   : nat,
    Rid, Sid, Cu1, Na, Ln                   : text,
    Ha, Ex, Sx                              : message,
    Z: {agent.public_key.text.{message}_inv(public_key)}_public_key,
    T: {text.agent.text.{message}_inv(public_key)}_public_key,
    R: {agent.text.message.message.
    {message}_inv(public_key)}_public_key,
    S: {agent.message.text.message.text.text.message.text.
    {message}_inv(public_key)}_public_key,
    Ra: (agent.text) set

init State := 1

transition

1. State = 1 /\ RCV_UA({U.A.Rid'.Sid'.Cu1'.
              {Hash(U.A.Rid'.Sid'.Cu1')}_inv(Ku1)}_Ka)
              /\ not(in(U.Rid',Ra)) =|>
 State' := 3 /\ Ra' := cons(U.Rid',Ra)
              /\SND_UA({A.U.Rid'.Kb.{Hash(A.U.Rid'.Kb)}_inv(Ka)}_Ku1)

2. State = 3 /\ RCV_UA({U.A.Rid.Z'.
              {Hash(U.A.Rid.Z')}_inv(Ku1)}_Ka) =|>
 State' := 5 /\ Na' := new()
              /\ SND_BA({A.B.Rid.Na'.Z'}_Kab)
              /\ witness(A,B,beta_alpha_rid,Rid)
              /\ secret(Na',nonce,{A,B})

3. State = 5 /\ RCV_BA({B.A.Rid.Ha'.R'}_Kab) /\ Ha' = Hash(Na) =|>
 State' := 9 /\ SND_UA({A.U.Rid.R'.
              {Hash(A.U.Rid.R')}_inv(Ka)}_Ku1)

4. State = 9 /\ RCV_UA({U.A.Rid.T'.
              {Hash(U.A.Rid.T')}_inv(Ku1)}_Ka) =|>
State' := 13 /\ SND_BA({A.B.Rid.T'}_Kab)

5. State = 13 /\ RCV_BA({B.A.Rid.Ex'.Ha}_Kab) /\ Ha = Hash(Na) =|>
 State' := 17 /\ Sx' := Hash(Ex'.Na)
               /\ SND_BA({A.B.Rid.Hash(Ex').Sx'}_Kab)

6. State = 17 /\ RCV_BA({B.A.Rid.S'}_Kab) =|>
 State' := 21 /\ SND_UA({A.U.Rid.S'.{Hash(A.U.Rid.S')}_inv(Ka)}_Ku1)

7. State = 21 /\ RCV_BA({B.A.Rid.Ln'}_Kab) =|>
 State' := 25 /\ SND_BA({A.B.Rid.Hash(Ln')}_Kab)
               /\ request(A,U,alpha_user_rid,Rid)
```

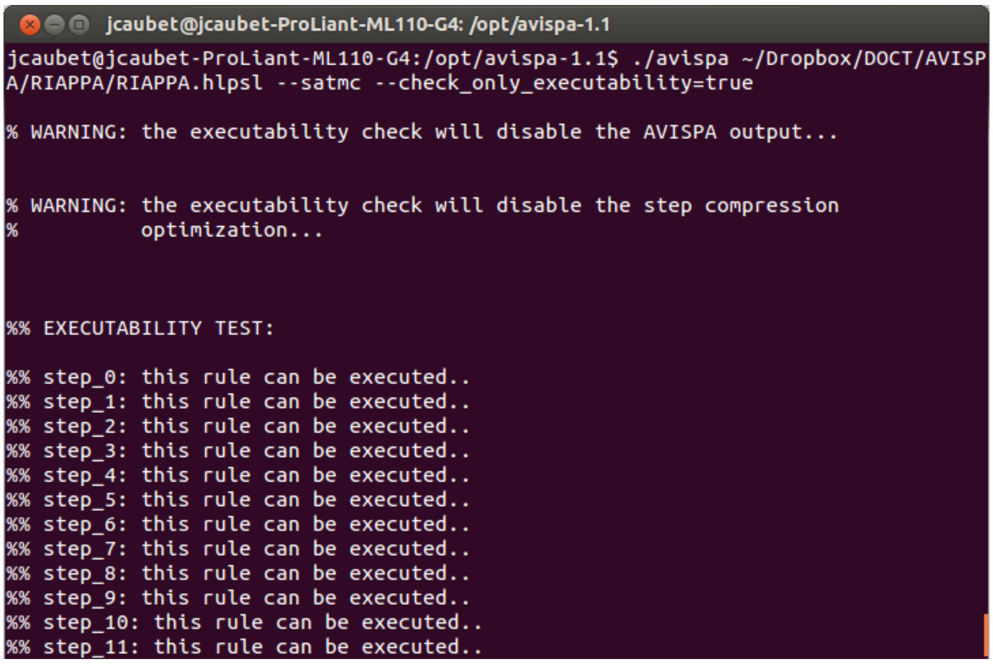
```
end role

%————————————————————————————————————————

role beta (
  B, A, U                                   : agent,
  Ka, Kb                                    : public_key,
  Kab                                       : symmetric_key,
  Hash                                      : hash_func,
  SND_AB, RCV_AB, SND_UB, RCV_UB            : channel(dy))

played_by B
def=

  local
    State                                         : nat,
    Rid, Na, Tu, Ub1, Nb, Nx, Ix, Bx, Ln          : text,
    Cb, Ha, Ub, Zx, Ex, Hx, Sx, Px                : message,
    Ku2                                           : public_key,
    Rb                                            : (message.text) set

init State := 7

transition

1. State = 7 /\ RCV_AB({A.B.Rid'.Na'.{B.Ku2'.Tu'.
             {Hash(B.Ku2'.Tu')}_inv(Ku2')}_Kb}_Kab)
             /\ not(in(Tu'.Rid',Rb)) =|>
State' := 11 /\ Ha' := Hash(Na')
             /\ Nb' := new()
             /\ Ub1' := new()
             /\ Ub' := Hash(Ub1)
             /\ Cb' := Hash(Nb'.Ub')
             /\ Rb' := cons(Tu'.Rid',Rb)
             /\ SND_AB({B.A.Rid'.Ha'.{B.Tu'.Cb'.Ub'.
             {Hash(B.Tu'.Cb'.Ub')}_inv(Kb)}_Ku2'}_Kab)
             /\ secret(Ha',ha,{A,B})
             /\ secret(Cb',cb,{U,B})
             /\ secret(Ub',ub,{U,B})
```

```
2.State = 11 /\ RCV_AB({A.B.Rid.{Tu.B.Nx'.
                {Hash(Tu.B.Nx')}_inv(Ku2)}_Kb}_Kab) =|>
State' := 15 /\ Zx' := Hash(Nx'.Nb) /\ Ix' := new()
             /\ Hx' := Hash(Ix'.Zx') /\ Bx' := new()
             /\ Ex' := Hash(Hx'.Bx') /\ SND_AB({B.A.Rid.Ex'.Ha}_Kab)

3. State = 15 /\ RCV_AB({A.B.Rid.Hash(Ex).Sx'}_Kab)
              /\ Sx' = Hash(Ex.Na) =|>
 State' := 19 /\ Px' := Hash(Zx.Ix.Bx)
              /\ SND_AB({B.A.Rid.{B.Px'.Tu.Sx'.Ub1.Ix.Zx.Bx.
              {Hash(B.Px'.Tu.Sx'.Ub1.Ix.Zx.Bx)}_inv(Kb)}_Ku2}_Kab)

4. State = 19 /\ SND_UB({Px.B.Rid.Hx.
              {Hash(Px.B.Rid.Hx)}_inv(Ku2)}_Kb) =|>
 State' := 23 /\ Ln' := new()
              /\ SND_AB({B.A.Rid.Ln'}_Kab)
              /\ secret(Ln',linknumber,{A,B})

4. State = 23 /\ RCV_AB({A.B.Rid.Hash(Ln)}_Kab) =|>
 State' := 27 /\ request(B,A,beta_alpha_rid,Rid)
              /\ request(B,U,beta_user_ku2,Ku2)

end role

%------------------------------------------------------------

role session(
  U, A, B                : agent,
  Kab                    : symmetric_key,
  Ku1, Ka, Kb            : public_key,
  Hash                   : hash_func)
def=

  local
    SAU, RAU, SBU, RBU,
    SUA, RUA, SBA, RBA,
    SAB, RAB, SUB, RUB  : channel (dy)

composition

         user (U,A,B,Ku1,Ka,Hash,SAU,RAU,SBU,RBU)
```

```
     /\ alpha (A,B,U,Ka,Kb,Ku1,Kab,Hash,SUA,RUA,SBA,RBA)
     /\ beta (B,A,U,Ka,Kb,Kab,Hash,SAB,RAB,SUB,RUB)

end role

%——————————————————————————————————————————

role environment()
def=

  const
    requestid, serviceid, linknumber, nonce,
    timestamp, cu2, cb, ub, ha, userkey2,
    blindsignature, alpha_user_rid,
    beta_alpha_rid, beta_user_ku2                : protocol_id,
    u, a, b, i                                   : agent,
    kab, kai, kib                                : symmetric_key,
    ku1, ka, kb, ki                              : public_key,
    fhash                                        : hash_func

intruder_knowledge = {u,a,b,i,ka,kb,ku1,ki,kib,kai,fhash,inv(ki)}

composition

          session(u,a,b,kab,ku1,ka,kb,fhash)
       /\ session(u,a,b,kab,ku1,ka,kb,fhash)
       /\ session(i,a,b,kab,ku1,ka,kb,fhash)

end role

%——————————————————————————————————————————

goal

  authentication_on alpha_user_rid
  authentication_on beta_alpha_rid
  authentication_on beta_user_ku2

  secrecy_of serviceid, userkey2, timestamp, linknumber, requestid,
    nonce, blindsignature, ha, cu2, cb, ub
```

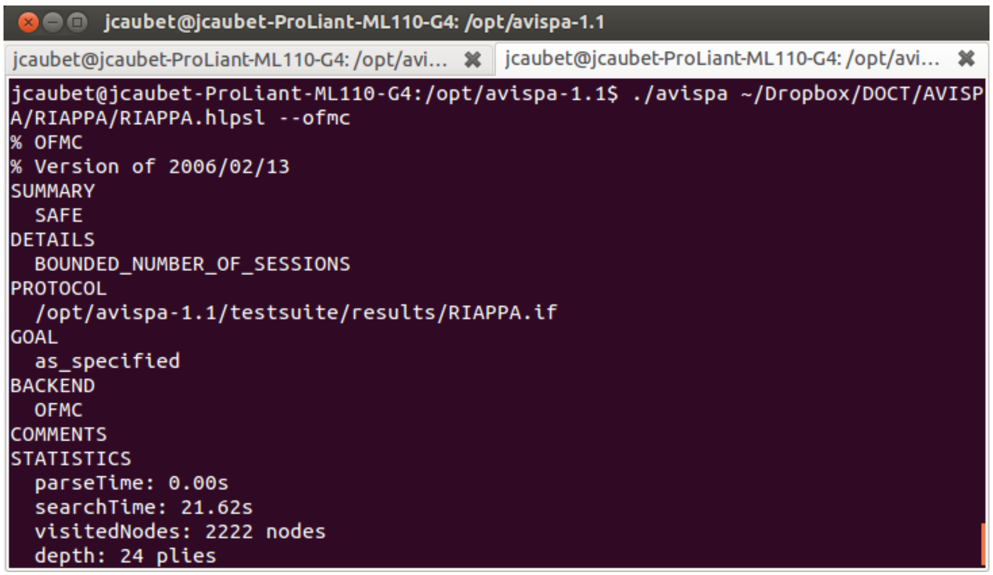
```
end goal

environment()
```

B.2.1 Results



```
jcaubet@jcaubet-ProLiant-ML110-G4:/opt/avispa-1.1$ ./avispa ~/Dropbox/DOCT/AVISP
A/RIAPPA/RIAPPA.hlpsl --satmc --check_only_executability=true

% WARNING: the executability check will disable the AVISPA output...


% WARNING: the executability check will disable the step compression
%          optimization...



%% EXECUTABILITY TEST:

%% step_0: this rule can be executed..
%% step_1: this rule can be executed..
%% step_2: this rule can be executed..
%% step_3: this rule can be executed..
%% step_4: this rule can be executed..
%% step_5: this rule can be executed..
%% step_6: this rule can be executed..
%% step_7: this rule can be executed..
%% step_8: this rule can be executed..
%% step_9: this rule can be executed..
%% step_10: this rule can be executed..
%% step_11: this rule can be executed..
```

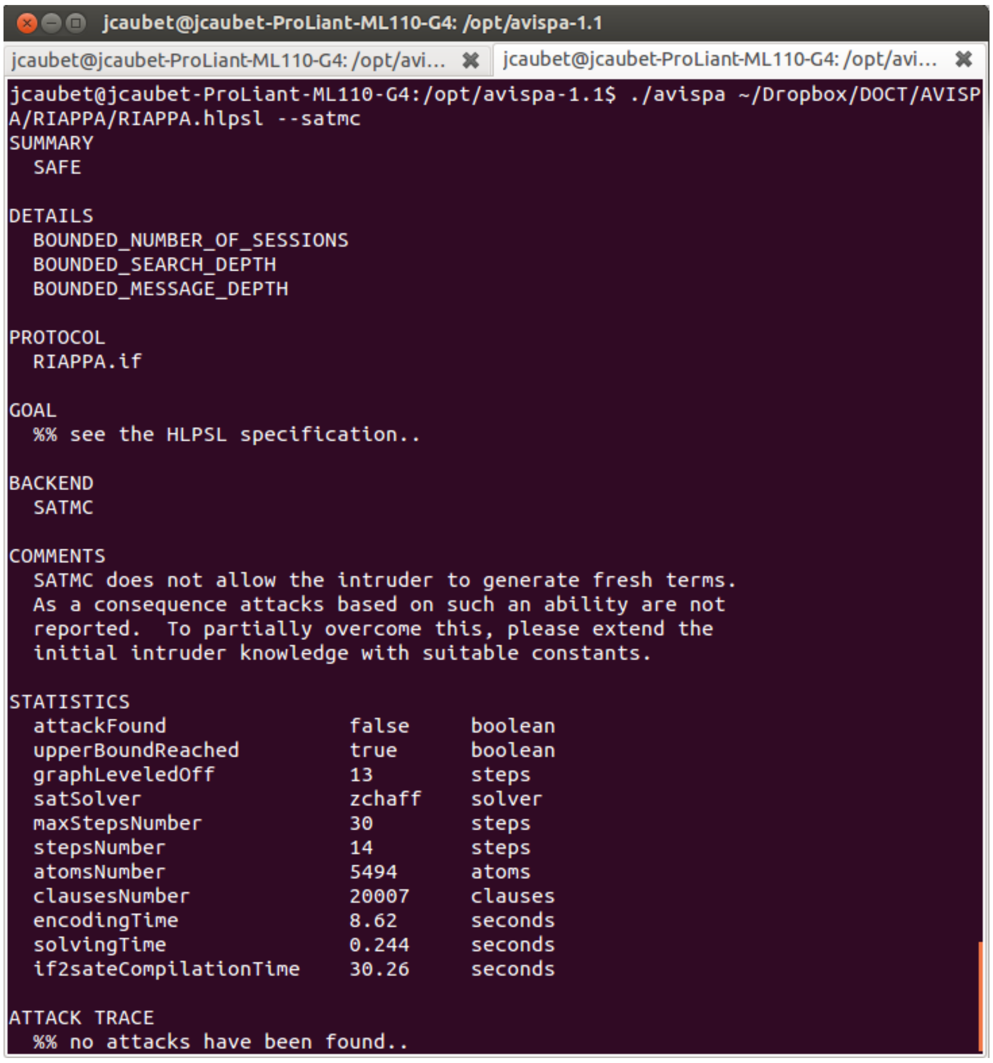
Figure B.5: TIAPPA - AVISPA output of the executability test.



jcaubet@jcaubet-ProLiant-ML110-G4: /opt/avispa-1.1
jcaubet@jcaubet-ProLiant-ML110-G4: /opt/avi...
jcaubet@jcaubet-ProLiant-ML110-G4: /opt/avi...

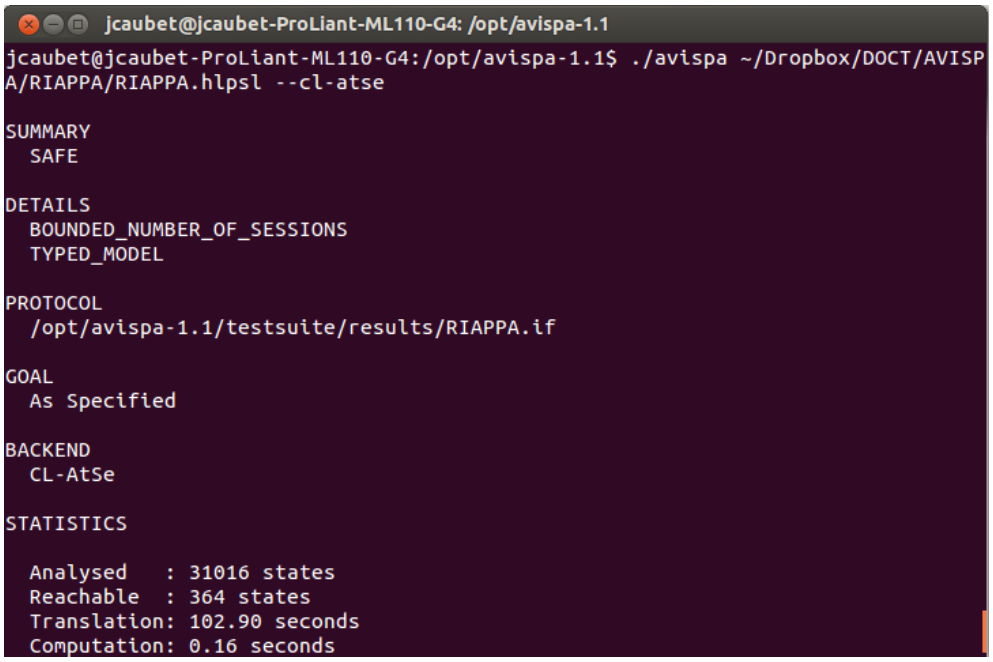
```
jcaubet@jcaubet-ProLiant-ML110-G4:/opt/avispa-1.1$ ./avispa ~/Dropbox/DOCT/AVISP
A/RIAPPA/RIAPPA.hlpsl --ofmc
% OFMC
% Version of 2006/02/13
SUMMARY
  SAFE
DETAILS
  BOUNDED_NUMBER_OF_SESSIONS
PROTOCOL
  /opt/avispa-1.1/testsuite/results/RIAPPA.if
GOAL
  as_specified
BACKEND
  OFMC
COMMENTS
STATISTICS
  parseTime: 0.00s
  searchTime: 21.62s
  visitedNodes: 2222 nodes
  depth: 24 plies
```

Figure B.6: TIAPPA - AVISPA output using OFMC back-end.


jcaubet@jcaubet-ProLiant-ML110-G4: /opt/avispa-1.1
jcaubet@jcaubet-ProLiant-ML110-G4: /opt/avi...
jcaubet@jcaubet-ProLiant-ML110-G4: /opt/avi...

```
jcaubet@jcaubet-ProLiant-ML110-G4:/opt/avispa-1.1$ ./avispa ~/Dropbox/DOCT/AVISP
A/RIAPPA/RIAPPA.hlpsl --satmc
SUMMARY
  SAFE

DETAILS
  BOUNDED_NUMBER_OF_SESSIONS
  BOUNDED_SEARCH_DEPTH
  BOUNDED_MESSAGE_DEPTH

PROTOCOL
  RIAPPA.if

GOAL
  %% see the HLPSL specification..

BACKEND
  SATMC

COMMENTS
  SATMC does not allow the intruder to generate fresh terms.
  As a consequence attacks based on such an ability are not
  reported.  To partially overcome this, please extend the
  initial intruder knowledge with suitable constants.

STATISTICS
  attackFound              false     boolean
  upperBoundReached        true      boolean
  graphLeveledOff          13        steps
  satSolver                zchaff    solver
  maxStepsNumber           30        steps
  stepsNumber              14        steps
  atomsNumber              5494      atoms
  clausesNumber            20007     clauses
  encodingTime             8.62      seconds
  solvingTime              0.244     seconds
  if2sateCompilationTime   30.26     seconds

ATTACK TRACE
  %% no attacks have been found..
```

Figure B.7: TIAPPA - AVISPA output using SATMC back-end.



```
jcaubet@jcaubet-ProLiant-ML110-G4:/opt/avispa-1.1$ ./avispa ~/Dropbox/DOCT/AVISP
A/RIAPPA/RIAPPA.hlpsl --cl-atse

SUMMARY
  SAFE

DETAILS
  BOUNDED_NUMBER_OF_SESSIONS
  TYPED_MODEL

PROTOCOL
  /opt/avispa-1.1/testsuite/results/RIAPPA.if

GOAL
  As Specified

BACKEND
  CL-AtSe

STATISTICS

  Analysed   : 31016 states
  Reachable  : 364 states
  Translation: 102.90 seconds
  Computation: 0.16 seconds
```

Figure B.8: TIAPPA - AVISPA output using CL-AtSe back-end.

B.3 RIAPPA

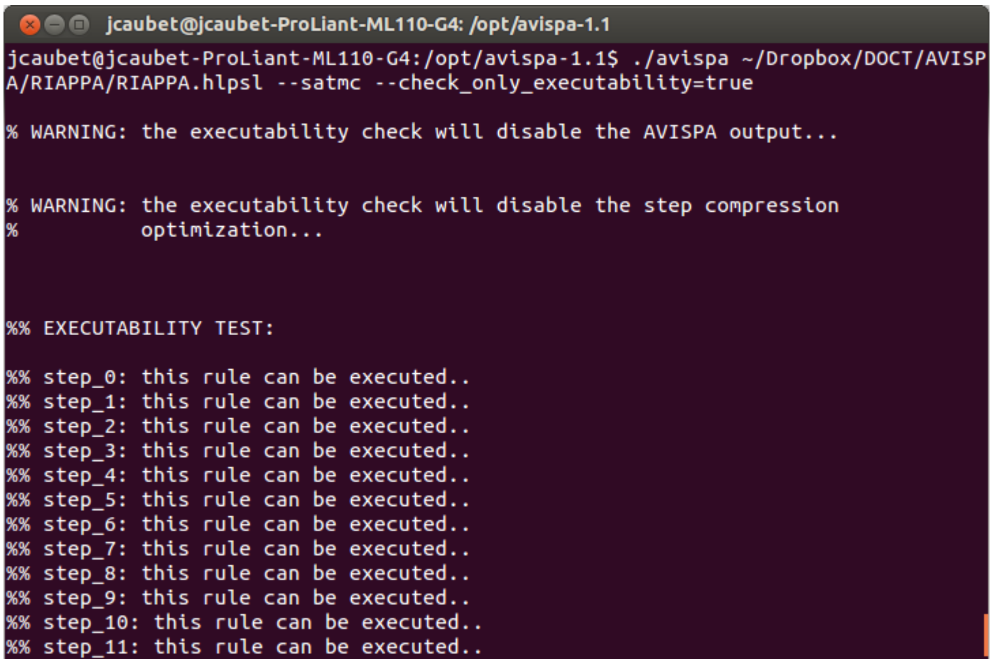
```
%% HLPSL:
% A Robust Identity Assignment Protocol for P2P overlAys
%%%%%%%%%%%%%% Created by Juan Caubet %%%%%%%%%%%%%%%%%%

role user (
  U, A, B                                    : agent,
  Ku1, Ka                                    : public_key,
  Hash                                       : hash_func,
  SND_AU, RCV_AU, SND_BU, RCV_BU             : channel(dy))

played_by U
def=

  local
    State                    : nat,
    Rid, Sid, Pu             : text,
    Cu2, S                   : message,
    Ku2, Kb                  : public_key

  const
    cu1                      : text

init State := 0

transition

1. State = 0 /\ RCV_AU(start) =|>
 State' := 2 /\ Rid' := new() /\ Sid' := new()
             /\ SND_AU({U.A.Rid'.Sid'.cu1.
                 {Hash(U.A.Rid'.Sid'.cu1)}_inv(Ku1)}_Ka)
             /\ witness(U,A,alpha_user_rid,Rid')
             /\ secret(Rid',requestid,{U,A,B})
             /\ secret(Sid',serviceid,{U,A})

2. State = 2 /\ RCV_AU({A.U.Rid.Kb'.
                 {Hash(A.U.Rid.Kb')}_inv(Ka)}_Ku1) =|>
 State' := 4 /\ Ku2' := new() /\ Pu' := new()
             /\ SND_AU({U.A.Rid.{Pu'.B.Ku2'
                 .{Hash(Pu'.B.Ku2')}_inv(Ku2')}_Kb'.
```

```
                    {Hash(U.A.Rid.{Pu'.B.Ku2'
                    .{Hash(Pu'.B.Ku2')}_inv(Ku2')}_Kb')}_inv(Ku1)}_Ka)
                 /\ witness(U,B,beta_user_ku2,Ku2')
                 /\ secret(Ku2',userkey2,{U,B})
                 /\ secret(Pu',nodeid,{U,B})

3. State = 4 /\ RCV_AU({A.U.Rid.{B.Pu.Cu2'.S'
                    .{Hash(B.Pu.Cu2'.S')}_inv(Kb)}_Ku2.
                    {Hash(A.U.Rid.{B.Pu.Cu2'.S'
                    .{Hash(B.Pu.Cu2'.S')}_inv(Kb)}_Ku2)}_inv(Ka)}_Ku1)
                 /\ Cu2' = B.A.Pu.Ku2.S'
                    .{Hash(B.A.Pu.Ku2.S')}_inv(Kb) =|>
 State' := 6 /\ SND_BU({Pu.B.Rid.Hash(Cu2').
                    {Hash(Pu.B.Rid.Hash(Cu2'))}_inv(Ku2)}_Kb)

end role

%------------------------------------------------------------------------

role alpha (
  A, B, U                                   : agent,
  Ka, Kb, Ku1                               : public_key,
  Kab                                       : symmetric_key,
  Hash                                      : hash_func,
  SND_UA, RCV_UA, SND_BA, RCV_BA            : channel(dy))

played_by A
def=

  local
    State                         : nat,
    Rid, Sid, Bp, Ln, Cu1         : text,
    Cu2b, S                       : message,
    Ra                            : (agent.text) set,
    T             : {{agent.text.text}_inv(public_key)}_public_key,
    Z    : {public_key.{text.agent}_inv(public_key)}_public_key

init State := 1

transition
```

```
1. State = 1 /\ RCV_UA({U.A.Rid'.Sid'.Cu1'.
                  {Hash(U.A.Rid'.Sid'.Cu1')}_inv(Ku1)}_Ka)
             /\ not(in(U.Rid',Ra)) =|>
 State' := 3 /\ Ra' := cons(U.Rid',Ra)
             /\ SND_UA({A.U.Rid'.Kb.
                  {Hash(A.U.Rid'.Kb)}_inv(Ka)}_Ku1)

2. State = 3 /\ RCV_UA({U.A.Rid.Z'.
                  {Hash(U.A.Rid.Z')}_inv(Ku1)}_Ka) =|>
 State' := 7 /\ Bp' := new()
             /\ SND_BA({A.B.Rid.Bp'.Z'}_Kab)
             /\ witness(A,B,beta_alpha_rid,Rid)
             /\ secret(Bp',blindparameters,{A,B})

3. State = 7 /\ RCV_BA({B.A.Rid.Cu2b'}_Kab) =|>
State' := 11 /\ S' := {Hash(Cu2b')}_inv(Ka)
             /\ SND_BA({A.B.Rid.S'}_Kab)
             /\ secret(S',blindsignature,{A,B,U})

4. State = 11 /\ RCV_BA({B.A.Rid.Hash(Bp).T'}_Kab) =|>
 State' := 15 /\ SND_UA({A.U.Rid.T'.
                   {Hash(A.U.Rid.T')}_inv(Ka)}_Ku1)

5. State = 15 /\ RCV_BA({B.A.Rid.Ln'}_Kab) =|>
 State' := 19 /\ SND_BA({A.B.Rid.Hash(Ln')}_Kab)
              /\ request(A,U,alpha_user_rid,Rid)

end role

%-----------------------------------------------------------------

role beta (
  B, A, U                               : agent,
  Ka, Kb                                : public_key,
  Kab                                   : symmetric_key,
  Hash                                  : hash_func,
  SND_AB, RCV_AB, SND_UB, RCV_UB        : channel(dy))

played_by B
def=
  local
    State                 : nat,
    Rid, Pu, Bp, Ln       : text,
    Cu2, Cu2b, S          : message,
    Ku2                   : public_key,
    Rb                    : (message.text) set

init State := 5

transition

1. State = 5 /\ RCV_AB({A.B.Rid'.Bp'.{Pu'.B.Ku2'.
                   {Hash(Pu'.B.Ku2')}_inv(Ku2')}_Kb}_Kab)
             /\ not(in(Pu'.Rid',Rb)) =|>
 State' := 9 /\ Cu2b' := B.A.Pu'.Ku2'.Hash(Bp')
             /\ Rb' := cons(Pu'.Rid',Rb)
             /\ SND_AB({B.A.Rid'.Cu2b'}_Kab)
             /\ secret(Cu2b',cu2b,{A,B})

2. State = 9 /\ RCV_AB({A.B.Rid.S'}_Kab)
             /\ S' = {Hash(Cu2b)}_inv(Ka) =|>
State' := 13 /\ Cu2' := B.A.Pu.Ku2.S'.{Hash(B.A.Pu.Ku2.S')}_inv(Kb)
             /\ SND_AB({B.A.Rid.Hash(Bp).{B.Pu.Cu2'.S'.
                   {Hash(B.Pu.Cu2'.S')}_inv(Kb)}_Ku2}_Kab)
             /\ secret(Cu2',cu2,{U,B})

3. State = 13 /\ RCV_UB({Pu.B.Rid.Hash(Cu2).
                   {Hash(Pu.B.Rid.Hash(Cu2))}_inv(Ku2)}_Kb) =|>
 State' := 17 /\ Ln' := new() /\ SND_AB({B.A.Rid.Ln'}_Kab)
              /\ secret(Ln',linknumber,{A,B})

4. State = 17 /\ RCV_AB({A.B.Rid.Hash(Ln)}_Kab) =|>
 State' := 21 /\ request(B,A,beta_alpha_rid,Rid)
              /\ request(B,U,beta_user_ku2,Ku2)

end role

%------------------------------------------------------------------------

role session(
  U, A, B          : agent,
  Kab              : symmetric_key,
  Ku1, Ka, Kb      : public_key,
  Hash             : hash_func)
def=

  local
    SAU, RAU, SBU, RBU,
    SUA, RUA, SBA, RBA,
    SAB, RAB, SUB, RUB   : channel (dy)

composition

         user  (U,A,B,Ku1,Ka,Hash,SAU,RAU,SBU,RBU)
      /\ alpha (A,B,U,Ka,Kb,Ku1,Kab,Hash,SUA,RUA,SBA,RBA)
      /\ beta  (B,A,U,Ka,Kb,Kab,Hash,SAB,RAB,SUB,RUB)

end role

%----------------------------------------------------------------------------

role environment()
def=

  const
    requestid, serviceid, linknumber,
    blindparameters, nodeid, cu2, cu2b,
    userkey2, blindsignature, alpha_user_rid,
    beta_alpha_rid, beta_user_ku2                 : protocol_id,
    u, a, b, i                                    : agent,
    kab, kai, kib                                 : symmetric_key,
    ku1, ka, kb, ki                               : public_key,
    fhash                                         : hash_func

intruder_knowledge = {u,a,b,i,ka,kb,ku1,ki,kib,kai,fhash,inv(ki)}

composition

              session(u,a,b,kab,ku1,ka,kb,fhash)
           /\ session(u,a,b,kab,ku1,ka,kb,fhash)
           /\ session(i,a,b,kab,ku1,ka,kb,fhash)
```

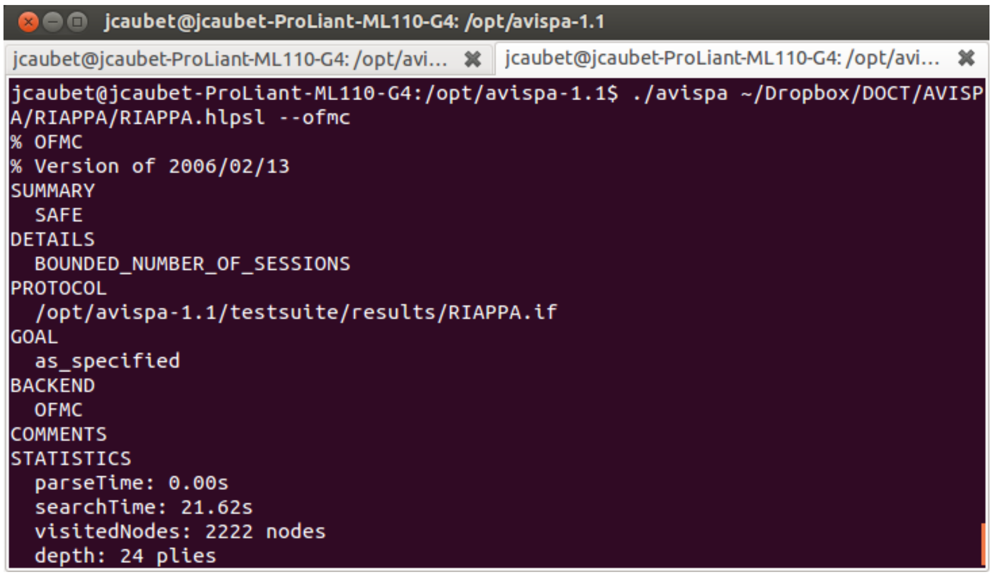
```
end role

%------------------------------------------------------------------------

goal

  authentication_on alpha_user_rid
  authentication_on beta_alpha_rid
  authentication_on beta_user_ku2

  secrecy_of serviceid, userkey2, nodeid, linknumber,
  requestid, blindparameters, blindsignature, cu2b, cu2

end goal

environment()
```

B.3.1 Results



```
jcaubet@jcaubet-ProLiant-ML110-G4:/opt/avispa-1.1$ ./avispa ~/Dropbox/DOCT/AVISP
A/RIAPPA/RIAPPA.hlpsl --satmc --check_only_executability=true

% WARNING: the executability check will disable the AVISPA output...


% WARNING: the executability check will disable the step compression
%          optimization...



%% EXECUTABILITY TEST:

%% step_0: this rule can be executed..
%% step_1: this rule can be executed..
%% step_2: this rule can be executed..
%% step_3: this rule can be executed..
%% step_4: this rule can be executed..
%% step_5: this rule can be executed..
%% step_6: this rule can be executed..
%% step_7: this rule can be executed..
%% step_8: this rule can be executed..
%% step_9: this rule can be executed..
%% step_10: this rule can be executed..
%% step_11: this rule can be executed..
```

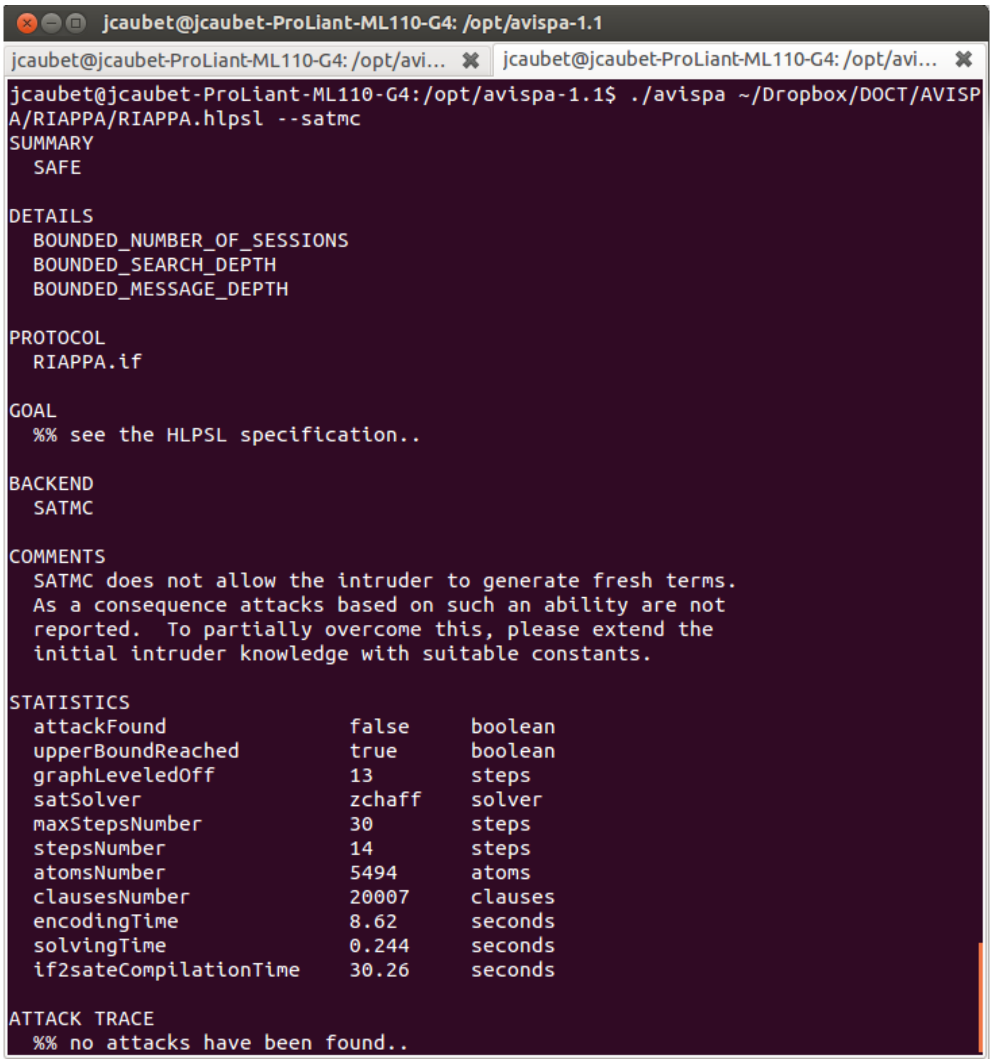
Figure B.9: RIAPPA - AVISPA output of the executability test.



jcaubet@jcaubet-ProLiant-ML110-G4: /opt/avispa-1.1
jcaubet@jcaubet-ProLiant-ML110-G4: /opt/avi...
jcaubet@jcaubet-ProLiant-ML110-G4: /opt/avi...

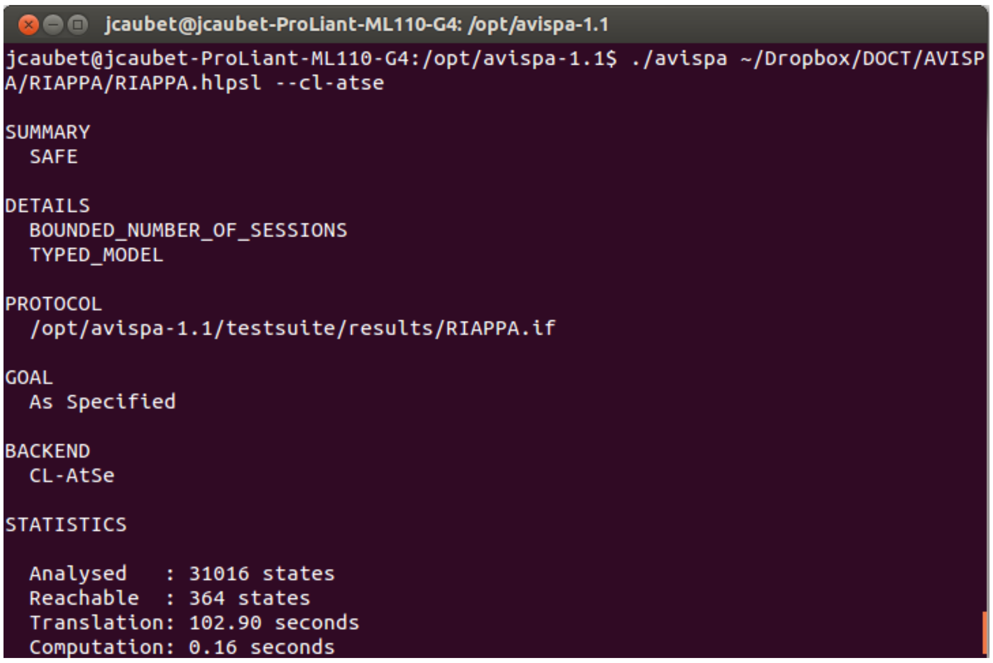
```
jcaubet@jcaubet-ProLiant-ML110-G4:/opt/avispa-1.1$ ./avispa ~/Dropbox/DOCT/AVISP
A/RIAPPA/RIAPPA.hlpsl --ofmc
% OFMC
% Version of 2006/02/13
SUMMARY
  SAFE
DETAILS
  BOUNDED_NUMBER_OF_SESSIONS
PROTOCOL
  /opt/avispa-1.1/testsuite/results/RIAPPA.if
GOAL
  as_specified
BACKEND
  OFMC
COMMENTS
STATISTICS
  parseTime: 0.00s
  searchTime: 21.62s
  visitedNodes: 2222 nodes
  depth: 24 plies
```

Figure B.10: RIAPPA - AVISPA output using OFMC back-end.


jcaubet@jcaubet-ProLiant-ML110-G4: /opt/avispa-1.1
jcaubet@jcaubet-ProLiant-ML110-G4: /opt/avi...
jcaubet@jcaubet-ProLiant-ML110-G4: /opt/avi...

```
jcaubet@jcaubet-ProLiant-ML110-G4:/opt/avispa-1.1$ ./avispa ~/Dropbox/DOCT/AVISP
A/RIAPPA/RIAPPA.hlpsl --satmc
SUMMARY
  SAFE

DETAILS
  BOUNDED_NUMBER_OF_SESSIONS
  BOUNDED_SEARCH_DEPTH
  BOUNDED_MESSAGE_DEPTH

PROTOCOL
  RIAPPA.if

GOAL
  %% see the HLPSL specification..

BACKEND
  SATMC

COMMENTS
  SATMC does not allow the intruder to generate fresh terms.
  As a consequence attacks based on such an ability are not
  reported.  To partially overcome this, please extend the
  initial intruder knowledge with suitable constants.

STATISTICS
  attackFound                false     boolean
  upperBoundReached          true      boolean
  graphLeveledOff            13        steps
  satSolver                  zchaff    solver
  maxStepsNumber             30        steps
  stepsNumber                14        steps
  atomsNumber                5494      atoms
  clausesNumber              20007     clauses
  encodingTime               8.62      seconds
  solvingTime                0.244     seconds
  if2sateCompilationTime     30.26     seconds

ATTACK TRACE
  %% no attacks have been found..
```

Figure B.11: RIAPPA - AVISPA output using SATMC back-end.



```
jcaubet@jcaubet-ProLiant-ML110-G4:/opt/avispa-1.1$ ./avispa ~/Dropbox/DOCT/AVISP
A/RIAPPA/RIAPPA.hlpsl --cl-atse

SUMMARY
  SAFE

DETAILS
  BOUNDED_NUMBER_OF_SESSIONS
  TYPED_MODEL

PROTOCOL
  /opt/avispa-1.1/testsuite/results/RIAPPA.if

GOAL
  As Specified

BACKEND
  CL-AtSe

STATISTICS

  Analysed   : 31016 states
  Reachable  : 364 states
  Translation: 102.90 seconds
  Computation: 0.16 seconds
```

Figure B.12: RIAPPA - AVISPA output using CL-AtSe back-end.