SECURITY AND COMMUNICATION NETWORKS
Security Comm. Networks 2014; **7**:2743–2760
Published online 20 February 2014 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/sec.956

SPECIAL ISSUE PAPER

RIAPPA: a Robust Identity Assignment Protocol for P2P overlays

Juan Caubet*, Oscar Esparza, José L. Muñoz, Juanjo Alins and Jorge Mata-Díaz

Department of Telematics Engineering (ENTEL), Universitat Politècnica de Catalunya (UPC), Spain

ABSTRACT

Peer-to-peer (P2P) overlay networks have been proposed to solve routing problems of big distributed infrastructures, even for Internet scale. But the research community has been questioning the security of these networks for years. Most prior work in security services was focused on trust and reputation systems, anonymity, and secure routing. However, the proper management of identities in overlays is an important prerequisite to provide most of these security services. In this paper, we propose a protocol to control the access to a P2P overlay and to 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. Users are authenticated by a TTP using real-world digital certificates, they select their network identifier jointly with the other TTP, and finally, the two TTPs issue the internal certificate to them. The protocol also provides revocability and protection against Sybil attacks, Eclipse attacks, whitewashers, and so on. A detailed protocol description is presented, and a performance and security analysis of the protocol is also provided. Copyright © 2014 John Wiley & Sons, Ltd.

KEYWORDS
P2P overlay; DHT; Sybil attack; eclipse attack; identity assignment

***Correspondence**
Juan Caubet, Universitat Politècnica de Catalunya, Campus Nord, C/ Jordi Girona 1-3, Building C3, 08034, Barcelona, Spain.
E-mail: juan.caubet@entel.upc.edu

1. INTRODUCTION

Peer-to-peer (P2P) overlays are being massively used, and their performance every day is better. According to the annual Cisco Visual Networking Index Forecast [1], the P2P traffic represented around 30% of global Internet protocol traffic in 2011, and it will grow at a compound annual growth rate of 23% from 2010 to 2015. Therefore, these systems will play an important role in the future Internet for sure. However, overlays are hardly being used for commercial applications, such as paid video streaming applications, because they have important security problems.

P2P video streaming applications have recently emerged as a cheap and efficient solution to provide real-time streaming services over the Internet. Also, Video-on-Demand (VoD) applications will produce three times more traffic by 2015 [1]. It is estimated that the amount of VoD traffic in 2015 will be equivalent to 3 billion DVDs per month. SopCast, PPTV, CoolStreaming, TVUnetworks, and Zattoo are some of the many streaming applications and services that have been developed so far. However, most of them are proprietary video streaming and distribution platforms, which use previous generations of P2P networks or distribute contents without any type of access control or security. Therefore, if we want to use overlays to implement commercial applications (such as paid VoD services), it is necessary to solve a series of security problems.

P2P overlays have been analyzed in depth to guarantee scalability and efficiency. However, few security mechanisms are being applied today. Most P2P overlays neither control the user access nor the behavior of the nodes within the network. 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. However, these improper usages cannot be faced only with data confidentiality, nodes authentication, non-repudiation, and so on. In particular, P2P overlays should follow the secure routing primitives described by Wallach in [2], which are (i) secure maintenance of routing tables, (ii) secure routing of messages, and (iii) secure identity assignment to nodes. But the first two problems depend in some way on the third one.

If the identity of the nodes within the overlay (nodeIDs) can be chosen by users without any control, we can have security and operational problems. Therefore, like any other network or service, these networks require a robust access control to prevent potential attackers joining the network. Moreover, a robust identity assignment system is necessary to improve the users' confidence in P2P overlays, so they can even use them for commercial applications. But unfortunately, little attention has been paid so far to the way that nodeIDs should be constructed or how to make access control mechanisms more robust.

For these reasons, we propose a robust identity assignment protocol for P2P overlays (RIAPPA), which should be executed the first time a user accesses the network (bootstrapping). RIAPPA provides to the newcomer a certificate that contains his or her identifier within the overlay (nodeID). This certificate is issued by two trusted third parties (TTPs): one external TTP that manages the real-world identity of the users and one internal TTP that manages the nodeIDs. The certificate can be revoked if necessary (for instance, in case of key compromise). RIAPPA has been designed to assure that nodeIDs are unique and uniformly distributed in the virtual space to achieve a proper load balancing in the overlay. The use of nodeIDs permits the user not to use his or her real-world identity and remain anonymous. Also, as our mechanism promotes the stability of nodeIDs, it is possible to efficiently use reputation systems to enforce fair cooperation and to punish improper usage of the network. However, in case a malicious user commits a serious offense or illegal action within the overlay, both TTPs can de-anonymize the identity of this user (traceability), for example, to start a legal investigation. Unlike most current systems in which nodeIDs are selected solely by the users, in RIAPPA, nodeIDs are decided jointly by the user and the internal TTP. This permits to prevent some attacks related to the identity of the nodes in a variety of scenarios that may include most commercial applications.

The rest of the paper is organized as follows: Section 2 summarizes some identity problems that arise when certain kinds of nodeIDs are used. Section 3 presents some proposals, which attempt to prevent, detect, and/or limit the identity problems experienced by these networks. Section 4 introduces a new protocol to assign identities in a secure and anonymous way in P2P overlays. Finally, the conclusions can be found in Section 5.

2. IDENTITY PROBLEMS IN P2P OVERLAYS

Most typical P2P overlay networks [3–5] are implemented using a distributed hash table (DHT), which stores $\{key, value\}$ pairs together with the node identifiers (nodeIDs) creating a virtual space. A *value* can be a certain resource (for instance, a file), or the way to reach this resource in the overlay (a pointer), and the associated *key* is used to locate this resource into the network. The DHT is divided in subtables, which correspond to a certain zone of the virtual space and which are assigned to different nodes. So each node is responsible for one zone, and hence, it is responsible for the {key,value} pairs contained in that zone (storing content and routing messages). Usually, a zone is assigned to a node whose nodeID is numerically close to the key values stored in the corresponding subtable of the DHT. Therefore, the location of the nodes in the virtual space is directly related to their nodeIDs. Unfortunately, in most current P2P overlay networks, these identifiers are generated by the nodes locally. This means that users can choose their nodeIDs. For example, users in CAN [3] are identified by their assigned zone within the virtual space, zones selected by them freely; in Chord [4] and Kademlia [5], nodeIDs are generated by the users using a hash function over their IP addresses; Pastry nodeIDs [6] are assigned randomly by the client software.

Several identity-related problems arise with the uncontrolled assignment of nodeIDs [7,8]: Sybil attacks, Eclipse attacks, the presence of whitewashers,[†] Man-in-the-Middle (MITM) attacks, the non-uniform distribution of nodeIDs, and so on.

2.1. The Sybil attack

The management of multiple nodeIDs (Sybils) by the same (malicious) node simultaneously is known as Sybil attack [10]. Carrying out this attack, a malicious user can increase his or her presence within the overlay by artificially simulating the existence of several nodes. Thus, the attacker can manage a group of colluding virtual nodes, which could damage the proper operation of the P2P network. For instance, an attacker performing the Sybil attack can improve its own reputation by using good feedback that comes from fake identities.

2.2. The eclipse attack

The Eclipse attack [11] is a way of routing poisoning, which aims to separate a part of the P2P overlay network from the rest. The attacker tries to intercept all the messages directed to a specific node (or resource) by means of a set of nodes with nodeIDs numerically close to the nodeID of the target node (or the resource's value). In this way, an honest node can be "eclipsed" by an attacker.

3. RELATED WORK

In this section, we discuss some research works that attempt to tackle some of the previously mentioned attacks/threats.

[†] 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 [9].

3.1. Centralized solutions

Douceur, in [10], comments the impossibility to know if two overlay nodes are managed by two real identities, or if there is only one user managing them, even asking other nodes within the network. Finally, he concluded that a trusted entity that certifies nodeIDs is the only solution to completely avoid the Sybil attack. However, he was the first to suggest methods for imposing computational cost on creating identities and system conditions to mitigate the Sybil attack.

Srivatsa and Liu proposed the use of certificates with a short lifetime issued by a bootstrap server, which also generates random nodeIDs [12]. This technique limits the number of nodeIDs that an adversary can obtain during a time period, depending on the lifetime of the certificates, and maintains complete anonymity of the nodes. However, the lifetime can affect the security of the system. If it is too short, the server can become a bottleneck as the update process of the certificates may introduce a significant computational overhead. On the other hand, longer lifetimes 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 cost.

In [13], Butler *et al.* considered the use of identity-based encryption to improve the critical assignment of user identities in P2P overlay networks, where users' public keys are directly derived from their nodeIDs, which are calculated randomly by a trusted authority. In this proposal, a single host plays the role of both trusted authority and bootstrap node, and the node authentication is performed via callback using their IP addresses, main drawback of the scheme.

Baumgart and Mies propose to use a hash function over a public key to generate the nodeIDs [14]. But the signature's public key must be additionally signed by a trustworthy certificate authority. Thus, this signature impedes the Sybil attack in the bootstrapping phase. In the absence of this authority, they propose to use a crypto puzzle to impede Sybil and Eclipse attacks.

In [15–18], Aiello *et al.* presented Likir (Layered Identity-based Kademlia-like InfRastructure), which is the architectural model of a new DHT system that offers both a very high protection level to most common attacks against structured P2P networks, and a simple framework supporting identity-based services. In their architecture, there is a "User Registration Service" that uses the OpenID protocol to authenticate users. In this context, a trusted entity binds the nodeID to the user public key. The nodeID or LikirID is a random string of 160 bits. This procedure requires the human interaction, which makes unfeasible the automatic nodeID generation. Likir guarantees the users' anonymity because users do not need to reveal their real-world identity. As this work is similar to ours, in Section 4.6, we compare the user registration module of Likir with the features provided by our RIAPPA protocol.

3.2. Distributed solutions

In the same line with Castro *et al.* [19], a cryptographic puzzle mechanism has also been proposed by Rowaihy *et al.* to limit Sybil attacks [20]. 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. They also propose to refresh the challenges constantly to avoid pre-computation. When a node wishes to join the network, it contacts a leaf node. Then, the leaf 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 a hash function over the node public key, previously selected by the user, and a random number generated by the root node. As with the aforementioned 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, he or she can take advantage of his or her position, as he or she will need a smaller number of puzzles to obtain a nodeID.

In [21], Da Costa *et al.* tried to minimize computing problems, which 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.

Lu proposed, in [22], a conundrum verification scheme, which allows access to the P2P network 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 these nodeIDs. This verification is performed on the basis of the statistics result. The weakness of this solution is in the binding of the nodeIDs, selected by the users, with their IP addresses to verify the identities.

In [13], Butler *et al.* also developed two decentralized identity assignment protocols. A fully decentralized ID-based assignment scheme and an approach that retains the separation of duties in a decentralized model at a low cost by using a hybrid of ID-based and symmetric key cryptography. But in the same way as in the centralized protocol, nodes are weakly authenticated via callback using their IP addresses, which is insufficient to prevent the Sybil attack.

3.3. Social network-based solutions

Wang *et al*. propose, in [23], a practical system for detecting Sybil identities using server-side clickstream models. Their approach groups "similar" user clickstreams[‡] 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 partake 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 [24–27]. 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 [28], 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 provides two information to improve the detection phase. On the one hand, the number of requests to befriend, 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 like 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 [29], Shi *et al.* present SybilShield, a protocol that utilizes multi-community social network structure in real world to defend against Sybil attack. This scheme leverages the sociological property that the number of cutting edges between a non-Sybil community and a Sybil community, which represents human-established trust relationships, is much smaller than that among non-Sybil communities. Moreover, authors use agent nodes to greatly reduce false positive rate of non-Sybils among multiple communities, while effectively identifying Sybil nodes.

4. ROBUST IDENTITY ASSIGNMENT PROTOCOL FOR P2P OVERLAYS

There are many P2P-based services that are widely used by Internet users that work without the need to clearly identify the nodes that are within the network. This is probably because most of these services are for free and anonymous. For these reasons, there are no quality-of-service agreements, and the users assume that there is a certain intrinsic risk when using such applications. However, if we want to use such networks to provide commercial applications, it is mandatory to solve security vulnerabilities, starting with the identity assignment problem, as stated by Wallach in [2].

In the literature, there are some contributions that have addressed the identity assignment problem (discussed in Section 3). However, in our opinion, none of these contributions have presented a usable mechanism (even for commercial purposes) to construct nodeIDs in a robust way, including, at the same time, among others, requirements such as revocability, anonymity, and traceability. For this reason, we have designed a new protocol to control the access of new users to an overlay and to assign identifiers in a secure way.

The RIAPPA protocol uses two collaborating TTPs to issue certificates to newcomers at bootstrapping. One of the TTPs (the external TTP) is responsible for authenticating users by using their real-world identities; meanwhile, the other TTP (the internal TTP) is responsible for deciding, jointly with the newcomer, the new nodeID. Finally, overlay certificates are issued and signed by both TTPs. RIAPPA can provide full anonymity, that is, the issued certificate contains the nodeID of the newcomer, but no entity will be able to match his or her real-world identity with that nodeID, even the same TTPs.

To preserve the user anonymity between TTPs, RIAPPA uses a blind signature scheme [30] with the aim that the external TTP can sign certificates without knowing the information they contain (nodeID, public key, etc.). However, TTPs share a link number (LN) for each user who obtains an overlay certificate. This LN binds the real-world identity of the users with their nodeIDs. Thus, anonymity is compatible with a robust protection against identity-based attacks. Certificates can be renewed when expired or revoked when necessary. This form of controlling users and nodes ensures complete stability of the nodeIDs, allowing the overlay to implement effective trust and reputation systems, which are known mechanisms to promote honest cooperation and to punish improper usages. In this sense, RIAPPA can work in a similar way to an *anonymous blacklisting system* (also called anonymous

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

revocation system) [31], because the TTPs have the ability to revoke access from dishonest/abusive anonymous users without revealing their real-world identity. However, in case a malicious user commits a serious offense or illegal action within the overlay, both TTPs can de-anonymize the identity of this user (traceability). 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 that require starting a legal investigation. In this case, RIAPPA can also work like a *revocable anonymity system* [32] if necessary and fully de-anonymize malicious users committing illegal actions.

Regarding the nodeID generation, it is performed collaboratively by the node itself and the internal TTP, so neither the user nor the TTP can choose the location of the nodeID in the virtual space of the overlay unilaterally. In this way, our protocol also prevents the internal TTP to place a node at a certain position seeking its own benefit.[§]

4.1. Scenario

Robust identity assignment protocol for P2P overlays could be deployed in a variety of scenarios in which security is a must, including commercial applications. We consider that the scenario that better fits our protocol consists of one external TTP and one or many internal TTPs. As we previously mentioned, the external TTP is responsible for authenticating users (by using real-world identities). Just notice that it is mandatory that users only ask one external TTP for an overlay certificate if we want to assure that these users only have one nodeID.

On the other hand, there can be one or several internal TTPs, each of them managing one different overlay. These overlays may be different in many ways, for instance, overlays can provide different resources: CPU sharing, storage capacity, virtualization, and so on. They can offer different services: music sharing, video on demand, audio or video conference, and so on; or they can offer different contents: sports, news, cartoons, adult content, and so on.

The user can only access an overlay if it has the corresponding certificate, signed by both TTPs.

A scenario in which different external TTPs access to the same overlay (managed by only one internal TTP) is possible, but only if users have the possibility to be authenticated by just one of these external TTPs. This is necessary to guarantee nodeID uniqueness.

To better understand the usability of RIAPPA, we will introduce two possible scenarios. Let us first consider an anonymous and distributed video sharing service, in which users publish their own private videos with the rest of 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 case of being a commercial service, it may be provided by a private company (service provider) at flat rate. This company will act as the external TTP, and hence, it will be responsible for authenticating users when they hire the service, using real-word identities. As the 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 certification authority, 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 between TTPs), for instance, a commercial Certification Authority (CA). In our opinion, CA may be good candidates for being the internal TTP as they have experience in managing certificates. Also, they will probably have no interest in performing collusion attacks as 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 the user pays.

Newcomers that want to access to the service contact to the service provider (external TTP), which starts the RIAPPA protocol to issue a valid certificate to the overlay (with the cooperation of the internal TTP). After the protocol, the user has all the necessary to start sharing its private videos within the overlay in a secure and anonymous way. These contents may be of any type: sports, politics, adult content, and so on. 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. 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 case 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.[¶] Notice that in 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.

4.2. Design requirements

The RIAPPA protocol has been designed to satisfy all the following requirements at the same time:

[§] Notice that we consider that the TTPs are trusted for the action of issuing certificates following the RIAPPA protocol, but we do not require trusting them for the rest of actions performed in the overlay such as content distribution.

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

- *Uniqueness:* each user identified by a real-world identity should only manage one node (nodeID) in the system. This requirement is necessary to limit Sybil attacks.
- *Joint decision:* neither the nodes themselves nor the TTPs should be able to choose the users' nodeIDs unilaterally. This requirement is necessary to avoid Eclipse or MITM attacks.
- *Stability:* nodes should not have the possibility to change their nodeIDs uncontrollably. This requirement is necessary to implement efficient trust and reputation systems within the overlay.
- *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.
- *Anonymity:* no external entity (neither the nodes nor the TTPs) should have the possibility to relate the real-world identity of a user with his or her nodeID.
- *Verifiability:* all nodes should be able to check if certificates and nodeIDs have been properly generated.
- *Revocability:* any user should be able to revoke his or her certificate (for instance, in case of compromise of his or her private key) and obtain a new one. Moreover, if a dishonest or malicious node's behavior within the overlay is detected, the TTP should be able to revoke his or her certificate and thus prevent the user to access the overlay again .
- *Traceability:* if a user commits an illegal action within the overlay and he or she must be judged for it, the TTPs have the possibility to trace this user and match the nodeID with the user's real-world identity.

4.3. Assumptions and clarifications

We assume that all users trust both TTPs, as their name suggests. But this trustworthiness relationship is only for specific aspects related to the identity management and anonymity. In this sense, these TTPs are responsible to cooperatively issue certificates for the overlay, and obviously, the users are the ones who will choose their cryptographic key pairs. Also, we assume that the two TTPs will never collude.

We have defined the RIAPPA protocol in a generic way in order not to be algorithm dependent. For this reason, in the protocol specification, we do not directly define what cryptographic algorithms should be used, so future deployments of this protocol may use one of the many algorithms available. This also applies to the blind signature scheme, which is used to provide anonymity in the issuance of certificates.

In 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 signature is mainly used to ensure both message integrity and authentication. We will not assume that the identity of the signing principal can be deduced from this signature, as this deduction is not possible in all the digital signature schemes. For this reason, we always state the identity of the signing principal within the same message to avoid classical protocol vulnerabilities [33]. However, these assumptions are taken into account only for asymmetric cryptography, as using symmetric encryption, we can assume that the involved parties are already authenticated because the symmetric key is only known by them. We also assume that the two TTPs share a symmetric cryptographic key $K_{\alpha\beta}$.

Regarding control parameters, we have included a timestamp, which is used for two main purposes. First, 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 may be enough for this purpose. If any of the principals involved consider that the timestamp included in the message is not fresh enough, the entire process of identity assignment is canceled automatically by sending an error message to the rest of involved parties to delete all the information related to that request. Second, we also use this timestamp as request identifier, that is, this value will be used by all the entities involved to unequivocally identify this 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 like the ones used by the network time protocol may be a good option.

At the end of a successful transaction, the user obtains its certificate for the overlay, whose format can be seen in Figure 1. The certificate contains a unique *serial number*;

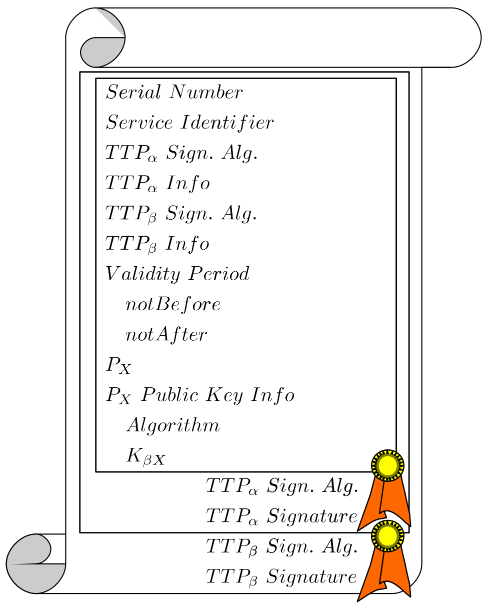

Figure 1. Overlay certificate format.

```
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 2. The Abstract Syntax Notation One of an overlay certificate.

the identifier of the overlay where the certificate is valid (*service identifier*); the identity of the two issuers, the internal TTP and the external TTP; the *validity period*;[‖] the nodeID of the 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 used to sign and parameters if needed (TTP_α *Sign. Alg.* and TTP_β *Sign. Alg.*). Note that it is similar to a X.509 standard certificate [34] but with two issuers and two signatures. Figure 2 shows an Abstract Syntax Notation One for the certificate, which follows the standard as far as possible. Appendix A includes the whole definition.

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 he or she followed the identity assignment process correctly. In this way, we avoid the internal TTP to issue valid certificates unilaterally.

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

4.4. Protocol specification

In a nutshell, a newcomer (X) must have a digital certificate for the real world ($C_{\alpha X}$), which contains his or her 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 within the overlay. And finally, TTP_α and TTP_β sign and issue the new certificate of X.

Next, we describe the protocol in more detail but without defining which encryption, signature, and blind signature schemes should be used, as many options can be chosen. 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 [33]. Table I presents a global summary of the used notation.

Figure 3 shows the message exchange between the three involved parties.

4.4.1. 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. He or she sends to it his or her real-world identity-based certificate ($C_{\alpha X}$), a timestamp (t_X) that will serve as a request identifier along the protocol, and the identifier of the service he or she wants to access (S_{ID}).

$$HELLO\ MESSAGE\ (Message\ 1),\ X \rightarrow TTP_\alpha :$$
$$\left\{C_{\alpha X}, \{ID_X, ID_{TTP_\alpha}, t_X, S_{ID}\}_{K_{\alpha X}^{-1}}\right\}_{K_{TTP_\alpha}}$$

More specifically, X performs the following operations: generates the timestamp (t_X); signs t_X and S_{ID} together with her identity and the receiver's identity using his or her private key in the real world $\left(K_{\alpha X}^{-1}\right)$;

‖ Two dates, the date on which the certificate validity period begins (*notBefore*) and the date on which the certificate validity period ends (*notAfter*).

Table I. Notation.

TTP_α, TTP_β	The external and internal TTPs, respectively
ID_{TTP_α}, ID_{TTP_β}	Identity of TTP_α and TTP_β, respectively
ID_X	Real-world identity of the newcomer
P_X	NodeID of the newcomer
$\frac{1}{2}P_X$	Half of the 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 two 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
$i \to j$:	The sending of a message from the entity i to the entity j
$\{m\}_K$	The ciphertext of a message m encrypted under a key K
$\{m\}_{K^{-1}}$	The signature of a node on a message m using its private signing key K^{-1}
t_X	The timestamp generated by the newcomer (also used as a request identifier)
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

TTP, trusted third party.

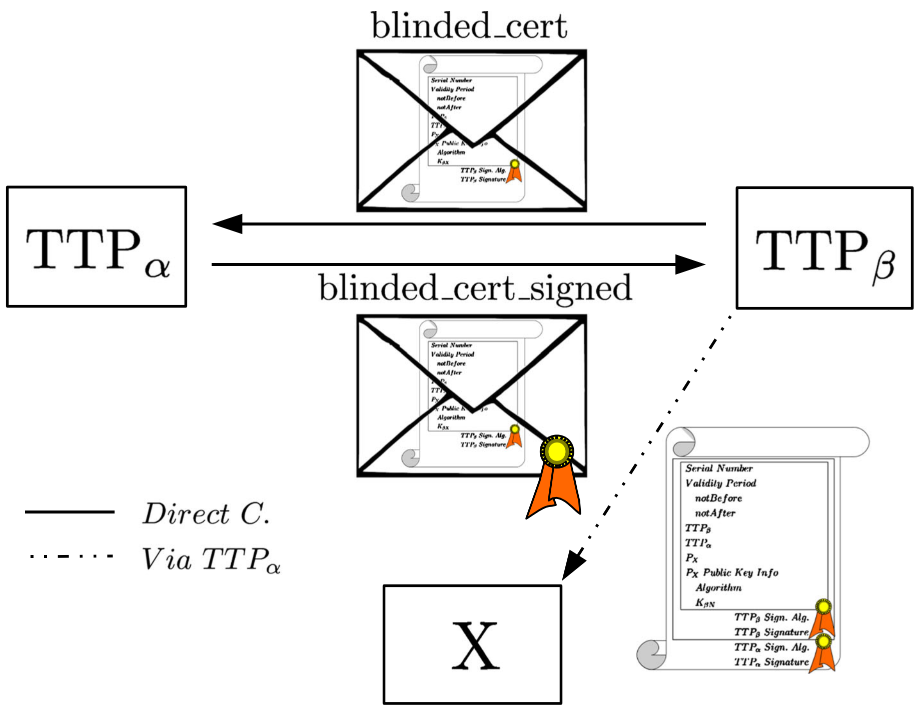

Figure 3. Message exchange.

encrypts the signed message together with $C_{\alpha X}$ using the public of TTP_α (K_{TTP_α}); and finally he or she sends the encrypted message.

The $C_{\alpha X}$ certificate allows the receiver 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, and the service identifier (S_{ID}) and certificates 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 the receiver can forward this message to another recipient posing as X [33]. Therefore, from here on, we will 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 he or 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), his or 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 X's request.

$$\textit{ACCEPT MESSAGE}\ (\textit{Message}\ 2),\ TTP_\alpha \to X:$$
$$\left\{\left\{ID_{TTP_\alpha}, ID_X, t_X, C_{TTP_\beta}\right\}_{K^{-1}_{TTP_\alpha}}\right\}_{K_{\alpha X}}$$

Specifically, TTP_α decrypts the *HELLO MESSAGE* (*message* 1), checks the X's certificate, verifies the signature of X, and checks the freshness of the timestamp. Then, if the preceding text is correct, TTP_α selects the TTP with which X must contact (from here on TTP_β), using the service identifier (S_{ID}), signs the certificate of 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 sends the encrypted message to 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 in order to obtain a valid nodeID for the overlay network. But before that, he or she generates a cryptographic key pair $\left(K_{\beta X}, K^{-1}_{\beta X}\right)$ to use within the overlay and selects half of the bits of his or her future nodeID $\left(\frac{1}{2}P_X\right)$. Then, X sends $\frac{1}{2}P_X$ together with his or her public key ($K_{\beta X}$) to TTP_β, all protected to prevent TTP_α can access the information. Finally, X also stores the certificate of TTP_β $\left(C_{TTP_\beta}\right)$ and its identity $\left(ID_{TTP_\beta}\right)$.

$$\textit{REQUEST MESSAGE}\ 1\ (\textit{Message}\ 3),$$
$$X \to TTP_\alpha : \left\{\left\{ID_X, ID_{TTP_\alpha}, t_X, A\right\}_{K^{-1}_{\alpha X}}\right\}_{K_{TTP_\alpha}}$$

$$\text{where } A = \left\{K_{\beta X}, \left\{\tfrac{1}{2}P_X, ID_{TTP_\beta}\right\}_{K^{-1}_{\beta X}}\right\}_{K_{TTP_\beta}}.$$

In more detail, X decrypts the *ACCEPT MESSAGE* (*message* 2), verifies the signature of TTP_α, and checks the timestamp freshness and the TTP_β's certificate $\left(C_{TTP_\beta}\right)$. Then, if all is correct, X selects half of the bits of his or her future nodeID $\left(\frac{1}{2}P_X\right)$ and a cryptographic key pair $\left(K_{\beta X}, K^{-1}_{\beta X}\right)$, signs $\frac{1}{2}P_X$ together with the identity of TTP_β using his or her new private key within the overlay $\left(K^{-1}_{\beta X}\right)$, and encrypts that signed message together with his or 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 his or her private key $K^{-1}_{\alpha X}$, 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 because he or she has no valid overlay nodeID. Thus, the anonymity of the user within the network is preserved. It is also noteworthy that these half of the bits of the new nodeID of X can be selected in many different ways (Section 4.4.2).

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.

$$\textit{REQUEST MESSAGE}\ 2\ (\textit{Message}\ 4),$$
$$TTP_\alpha \to TTP_\beta : \{t_X, \textit{blind\_param}, A\}_{K_{\alpha\beta}}$$

Specifically, TTP_α decrypts the *REQUEST MESSAGE* 1 (*message* 3), 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 and *blind_param*, using the symmetric key shared with TTP_β $\left(K_{\alpha\beta}\right)$, to prevent malicious users that can forge an overlay certificate. Finally, TTP_α sends the encrypted message to TTP_β.

Note that now, it is not necessary to include the sender and the receiver identities in the message, as the symmetric key is only shared by these two entities. 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 the bits to the nodeID of X (P_X) and generates the certificate using P_X and the public key sent by X ($K_{\beta X}$), among other information (Figure 1). Then, TTP_β blinds the certificate using the blind parameters (*blind_param*) in order 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.

$$\textit{SIGN REQUEST MESSAGE (Message 5)}, \quad TTP_\beta \to TTP_\alpha : \{t_X, blinded\_cert\}_{K_{\alpha\beta}}$$

More specifically, TTP_β decrypts the *REQUEST MESSAGE* 2 (*message* 4), 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 the 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 (Figure 2). Finally, TTP_β encrypts *blinded_cert* together with t_X using the symmetric key $K_{\alpha\beta}$ and sends the encrypted message to TTP_α.

In addition, TTP_β generates a LN to the user X and adds this *LN* and P_X to the user list. Note that this number will be shared with TTP_α and will allow that the real identity of X will be related to his or her nodeID. And because 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), his or her public key within the overlay $\left(K_{\beta X}\right)$, and the blind parameters (*blind_param*).

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.

$$\textit{SIGN RESPONSE MESSAGE (Message 6)}, \quad TTP_\alpha \to TTP_\beta : \{t_X, blinded\_cert\_signed\}_{K_{\alpha\beta}}$$

In more detail, TTP_α decrypts the *SIGN REQUEST MESSAGE* (*message* 5) and checks the timestamp freshness. Then, if the preceding text is correct, TTP_α signs the blinded certificate, encrypts the blinded certificate previously signed (*blinded_cert_signed*) together with t_X 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 its public key, because the 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.

$$\textit{ISSUE MESSAGE 1 (Message 7)}, \quad TTP_\beta \to TTP_\alpha : \{t_X, B\}_{K_{\alpha\beta}}$$

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

More specifically, TTP_β decrypts the *SIGN RESPONSE MESSAGE* (*message* 6), 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 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 his or her new overlay certificate ($C_{\beta X}$).

$$\textit{ISSUE MESSAGE 2 (Message 8)}, \quad TTP_\alpha \to X : \left\{ \left\{ ID_{TTP_\alpha}, ID_X, t_X, B \right\}^{-1}_{TTP_\alpha} \right\}_{K_{\alpha X}}$$

In more detail, TTP_α decrypts the *ISSUE MESSAGE* 1 (*message* 7) 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 4 shows the blind signature process by TTP_α. We can see the exchanged information between the involved parties in the previous four steps.

Step 9:

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

$$\textit{ISSUE ACK MESSAGE (Message 9)}, \quad X \to TTP_\beta : \left\{ \left\{ P_X, ID_{TTP_\beta}, t_X, H(C_{\beta X}) \right\}_{K^{-1}_{\beta X}} \right\}_{K_{TTP_\beta}}$$

Specifically, X decrypts the *ISSUE MESSAGE* 2 (*message* 8), verifies the signature of TTP_α, checks the timestamp freshness, decrypts the message B, verifies the signature of TTP_β, checks his or her new nodeID (P_X), and verifies his or 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

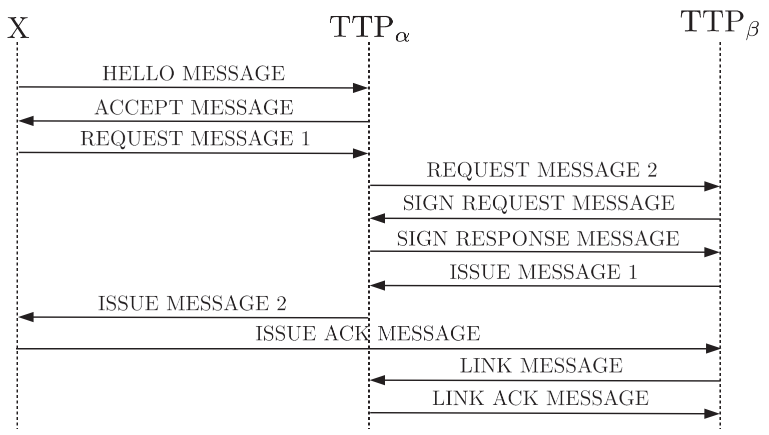

Figure 4. Blind signature of TTP_α.

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

Step 10:

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

$$\textit{LINK MESSAGE } (\textit{Message } 10),$$
$$TTP_\beta \rightarrow TTP_\alpha : \{t_X, LN\}_{K_{\alpha\beta}}$$

In more detail, TTP_β decrypts the *ISSUE ACK MESSAGE* (*message* 9), 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 the LN bound to X together with t_X 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, after a certain time, it will revoke the 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 the LN correctly.

$$\textit{LINK ACK MESSAGE } (\textit{Message } 11),$$
$$TTP_\alpha \rightarrow TTP_\beta : \{t_X, H(LN)\}_{K_{\alpha\beta}}$$

More specifically, TTP_α decrypts the *LINK MESSAGE* (*message* 10) 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 using the symmetric key $K_{\alpha\beta}$, and sends the encrypted message to TTP_β.

Step 12:

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

Specifically, TTP_β decrypts the *LINK ACK MESSAGE* (*Message* 11), 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 user and stored by TTP_β, and TTP_α and TTP_β share the LN to relate the identity of the user with his or her new nodeID P_X.

4.4.2. NodeID selection.

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

There are lots of ways to jointly select bits between two entities, but we must take into account that not all bits are equally significant. For instance, in the KAD network, an attacker will be within the tolerance zone of a target node if they have in common the eight most significant bits of their nodeIDs [35]. Therefore, we cannot allow an entity (user or TTP) to select those bits.

So, in our particular case, we use a scheme in which the user and the internal TTP decide randomly half of the bits of the nodeID, alternating them. The user selects even bits (second bit, fourth bit, sixth bit, etc.), and the TTP selects odd bits (first bit, third bit, fifth bit, etc.). This way, no entity totally controls the most significant bits, and the nodeIDs generated are valid for any overlay network, 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.

4.4.3. Node operation.

Deploying RIAPPA in an existing overlay network will affect the way in which nodes interact with each other. First, all nodes need a valid overlay certificate (not revoked and properly signed by both TTPs, the internal and the external) to operate in the network and securely communicate with other overlay nodes. Before establishing any communication with a receiver, the sender node 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). If all is correct, nodes will be able to use the normal cryptographic primitives to provide authentication, integrity, and confidentiality, among other security services. Unfortunately, most current overlays do not provide these needed cryptographic primitives to perform these actions, but this is out of the scope of this paper.

RIAPPA not only can help in having secure communications between nodes but also providing a better overlay performance. For instance, KAD new contacts can be obtained in three different ways: (i) through a bootstrap request; (ii) within the standard iteration process; or (iii) if a client is passively contacted by an unknown contact. But in all cases, the client inserts the contacts into its routing table without first verifying the correctness of nodeIDs, making quite easy to attack the routing process [35]. In case of adapting RIAPPA for its use in the KAD network, each node must ask and verify the associated overlay certificate before inserting the contacts into its routing table. In this case, routing tables would be much more stable and coherent as users cannot change their nodeIDs.

4.5. Performance and security analysis

In this section, we include an analysis of our protocol in terms of performance and security.

4.5.1. Performance analysis.

Providing security always implies a cost in the form of overhead, delay, computational cost, and so on. In this section, we will try to quantify this cost. However, we must remark that RIAPPA only is needed the very first time that a user joins the overlay. In other words, our protocol does not impact the performance of the normal operation in the overlay. Once a user has obtained his or her overlay certificate (with the corresponding nodeID), he or she does not need to contact again the TTPs unless the overlay certificate expires or it is revoked. In addition, if one of the two TTPs fails, the certificate issuing service becomes unavailable, but users with valid nodeIDs can join and operate in the overlay without contacting TTPs.

Regarding the communications between TTPs, we have decided that they use symmetric cryptography to communicate with each other, as this type of cryptography has lower computational cost, and both parties are automatically authenticated by sharing the symmetric key. Note that the internal TTP has a similar computational load than the external. Regarding the communication between users and TTPs, we use asymmetric encryption, even being more computationally expensive. This is because this exchange only is composed of a few messages of small size. More specifically, an estimate of the size of each message can be seen in Table II. To make this estimate, we have considered the following parameters: user/entity identifiers (16 bytes), timestamps (8 bytes), a service identifier (1 byte), RSA digital certificates (2500 bytes), digital signature algorithm (DSA) (40 bytes), RSA public keys (256 bytes), DSA public keys (256 bytes), RSA encryption overhead (75 bytes), a blinding parameter (64 bytes), a blinded certificate (466 bytes), a blind signature (466 bytes), an overlay certificate (1100 bytes), hash values (256 bytes), and a LN (4 bytes). Note that we have considered the use of a blind signature scheme based on elliptic-curve cryptography, for example [36], because it has an inherent advantage in terms of smaller key size and lower computational overhead over 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 evaluate the performance of the protocol in terms of the number of cryptographic operations needed. Table III shows the required cryptographic operations. As it can be seen in the Table, only 18 out of the 51 operations are performed by the user. If we take into account that a user only runs the protocol the first time he or she joins the network, the computational cost seems quite reasonable.

Obviously, the computational cost of RIAPPA will depend on the encryption and signature schemes that are used and also on the number of requests that assume the system. In 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 the database where they store the user or node information.

Next, we have also calculated the computational cost of cryptographic operations in each of the steps considering the use of the following algorithms: RSA-2048, DSA-2048, AES-256, the blind signature scheme of Fan *et al.* [36], and MD6-256.

Table II. 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

Table III. Summary of cryptographic operations.

	User	*TTP*$_\alpha$	*TTP*$_\beta$	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

Table IV. Computational cost of 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

Table IV 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, as we only have the time needed to complete the entire process, (signature and verification), 213.8227 *ms*. Note that this time should be divided in steps 5, 6, and 7; and step 9 is also needed to verify the blind signature. All these times have been obtained considering the use of a processor Intel Core i5-4570S 2.9 GHz (Intel Corporation, Malaysia) [37,38].

4.5.2. Security analysis.

Robust identity assignment protocol for P2P overlays provides a mechanism for issuing robust identities in a P2P overlay network while the anonymity of users is preserved. For this, two TTPs share a LN to relate the real-world identity of the users and their identities within the overlay, which is a security weakness that we must assume to ensure both features (robustness and anonymity). Next, we analyze the behavior of our protocol against some of the most common attacks in P2P overlay networks.

Sybil attack. To avoid this attack, it is necessary to guarantee that users can only obtain a unique nodeID for each overlay network. As discussed in Section 4.1, users must authenticate themselves in only one external TTP, and they should use a unique real-world identity (like the one provided by identity cards or passports). So, our solution for the problem of Sybils is based on the hypothesis that these real-world identities are much harder to spoof than any other type of identity. For this reason, we consider that it would be adequate that the credentials that users use to be authenticated in the external TTP should be issued by trusted and recognized entities, for instance, government-run or known certification authorities. In this way, if it is difficult for a (normal) person to assume multiple identities in the real-world, it will then be equally difficult for that person to assume multiple identities within the network [10].

Eclipse attack. To avoid an eclipse attack, it is imperative to firstly avoid the Sybil attack, and RIAPPA ensures this. Also, it is necessary that attackers cannot place themselves close to a target node. RIAPPA does not allow the self-generation of valid nodeIDs, because the newcomer and the internal TTP select jointly the users' nodeID. In brief, attackers cannot neither self-generate their nodeIDs, nor inject bad routing information in a target node because they cannot use nodeIDs of other contacts to fool victims.

MITM attack. During bootstrapping, it is impossible for an attacker to extract information or to modify the proper working of RIAPPA. This is due to all messages of the protocol that are signed/encrypted using public key cryptography or protected using the pre-shared symmetric cryptographic key between the TTPs. Hence, the only way that an attacker has to manipulate the proper issuance of an overlay certificate is due to compromise of the secret key of the user (or of the TTPs).

In the same sense, during the normal working of the overlay, an attacker cannot impersonate another user as this will mean that the attacker can use the overlay certificate of another node on its behalf, and this is impossible without knowing the private key of that user, because all nodes sign the messages they send.

Whitewashers. RIAPPA binds the users' real-world identities with their nodeIDs. So, assuming that users only have a real-world identity, our protocol guarantees the nodeIDs' stability. 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 he or she has accumulated under his or her previous nodeID. Furthermore, if the old nodeID has an associated reputation, it is inherited. Thanks to this, robust trust and reputation systems can be used to prevent malicious behaviors and to promote honest collaboration among nodes.

4.5.3. Requirements discussion.

Next, we discuss how all the design requirements exposed in Section 4.2 are satisfied. As we have explained previously, and thanks to the use of the real-world identities, users can only obtain a *unique* and *stable* nodeID. Moreover, this nodeID is *jointly selected* between the user and the internal TTP. Avoiding that the user can select his or her location within the overlay guarantees a pseudorandom location. As explained in Section 4.4.2, this allows us to obtain nodeIDs *uniformly distributed* within the virtual space of the overlay with high probability.

Robust identity assignment protocol for P2P overlays can provide full *anonymity*, because a single entity is not able to match the user's real-world identity with the nodeID. This is achieved thanks to the use of two TTPs, one of them using blind signature over the certificate. Also, all the exchanged messages in RIAPPA are encrypted to provide confidentiality, so no information is leaked to external entities. Obviously, this is valid as long as the two TTPs do not collude. As we previously mentioned, RIAPPA is vulnerable to collusions between TTPs. 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. In general, we do not expect to be cheated by the TTPs (none of them), because they are supposed to be trusted. However, in case of having just one dishonest TTP, this entity cannot alter the protocol to obtain self-benefit, as it depends on the other TTP to issue the overlay certificate. For this reason, we recommended in Section 4.1 that at least one of these TTPs should be a reputed and known certification authority or institution managed by a government or local administration.

On the other hand, all certificates are *verifiable* by anyone because they are signed by the two TTPs and they contain the user's nodeID, his or her public key, and other information related to the owner.

The RIAPPA protocol also provides *revocability* of certificates. An overlay certificate can be revoked by some reasons but mainly because of compromise of any of the cryptographic keys involved. In this case, the internal TTP revokes the certificate and issues a new certificate with a new public key but with the same nodeID. The user should contact the external TTP, and after authenticating him or her (to prove that she is the owner of the certificate), the certificate can be revoked. This is possible thanks to the LN that both TTPs share. Also, there is the possibility that the RIAPPA protocol does not finish properly, but the certificate has been delivered to the user. This can happen when the newcomer does not send the confirmation message (*ISSUE ACK MESSAGE*) to the internal TTP reporting that he or she has his or her new certificate. If the newcomer persist not sending the *ISSUE ACK MESSAGE*, the overlay certificate is revoked (prior its use).

To distribute this revocation information and to know what the revoked certificates are, we can use certificate revocation lists (CRLs) [34]. However, in the P2P overlay context, if only one or few CAs are capable of distributing the CRLs to all peers, they will become overloaded. For this reason, we can find in the literature many proposals to distribute this information more efficiently. The following are some of them. In [39], the authors propose a system based on Chord [4], where they use a bloom filter to resolve the bottleneck problem of some peers. Huang *et al.*, in [40], propose to introduce hierarchy in the distributed structure to reduce the number of requests to the CA. In [41], we propose a new distributed revocation system, which divides the CRLs into segments and distributes them using the overlay itself. Authors of [42] propose a new revocation system based on the Chord network where nodes store the revocation information but without using CRLs. If a certificate is revoked, the same node that stores the certificate stores a tag that indicates the revocation.

Finally, regarding the requirement of *traceability*, it may seem that it is in contradiction with the anonymity requirement, but it is not. In fully decentralized P2P networks, there are always nodes that misbehave, especially when anonymity is assured. In most of the cases, reputation systems are known mechanisms to punish improper usages like distribution of fake contents or routing tables poisoning. If these nodes remain misbehaving, they could be even revoked and hence ejected from the overlay, but they will continue 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 those users that commit a serious offense in the overlay, eviction from the overlay is necessary, but our system is prepared also to de-anonymize these users (thanks again to the LN between TTPs), so they can be prosecuted by law.

4.6. Comparison with similar proposals

Despite that it is difficult to compare our protocol with others because the requirements and the design goals are different, we can say that RIAPPA is in general more complex than the proposals summarized in Section 3, 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 the requirements considered.

In [20], the complex structure proposed by the authors requires a potentially large number of exchanges with varying servers to obtain a single nodeID, in addition to solving cryptographic puzzles, and all this just to limit the Sybil attack. The same occurs in [12,19,21,22] or [25].

In [13], 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 [10]. Moreover, in all three cases, the authors propose that TTPs generate the nodeIDs randomly, which can be a problem. The 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 [15–17] by Aiello *et al.* is a complete security proposal for overlay networks and the one that has conceptually more similarities with 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 identity is more demanding and may cause problems to honest users that do not have a real-world certificate than the requirement of Likir. However, our system is designed to prevent the Sybil attack more aggressively. By using our protocol, an attacker needs to steal, or forge, the real identity (digital certificate) to launch a Sybil attack. In Likir, the attacker needs to generate a set of OpenID identities, which is in general, less hard to do. 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 keep a LN to bind nodes with users, Likir's certification service also keeps a track of the association between userIDs and LikirIDs. As RIAPPA, Likir also guarantees the users' anonymity; however, it does not consider to provide traceability because 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 RIAPPA is that the Eclipse attack problem is addressed differently. Likir uses a random nodeIDs 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 can place a node at any position in the overlay network. In addition, the protocol exchange, messages, and operation are rather different from RIAPPA.

5. CONCLUSIONS

Vulnerability to certain attacks is a strong obstacle to develop critical services (e.g., commercial applications) in P2P overlay networks. In this paper, we have proposed the RIAPPA protocol with the aim of solving some vulnerabilities and turning these networks into a better platform for commercial applications. RIAPPA issues identities in a secure and anonymous way minimally altering the current operation of the overlays. Our protocol neither allows the users to select their nodeIDs nor the internal TTP to select the complete nodeIDs of the users, ensuring that users are placed in virtual space pseudo-randomly. In addition, RIAPPA also guarantees the stability and uniqueness of the nodeIDs, as the TTPs bind the real-world identities of the users with their nodeIDs, avoiding some of the most dangerous attacks (Sybil, Eclipse, and MITM) of P2P systems. Our protocol is the only one that achieves at the same time anonymity, revocability, and traceability. These requirements may seem to be in conflict but when deploying a commercial service over P2P overlays, you cannot treat equally all misbehaviors. Finally, we can conclude that the cost of RIAPPA is reasonable according to our analysis and considering that a user only executes the protocol the first time he or she wants to join the network.

ACKNOWLEDGEMENTS

We would like to thank the editors and reviewers for the opportunity to improve our paper and express our sincere gratitude to the anonymous reviewers for their valuable comments and suggestions.

This work has been supported partially by the Spanish Research Council with Project SERVET TEC2011-26452, Spanish Ministry of Science and Education with Project CONSOLIDER CSD2007-00004 (ARES), and Generalitat de Catalunya with Grant 2009 SGR-1362 to consolidated research groups.

REFERENCES

1. *Cisco visual networking index: forecast and methodology*, 2011-2016. Available from: http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-481360_ns827_Networking_Solutions_White_Paper.html.
2. Wallach DS. 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. Springer-Verlag Berlin: Heidelberg, 2002; 42–57.
3. Ratnasamy S, Francis P, Handley M, Karp R, Shenker S. A scalable content-addressable network, *Proceedings of the ACM Conference on Applications, Technologies, Architectures and Protocols for Computer Communication (SIGCOMM)*, San Diego, CA, USA, 2001; 161–172.
4. Stoica I, Morris R, Karger D, Kaaskoek M, Balakrishman H. Chord: a scalable peer-to-peer lookup service for Internet applications, *Proceedings of the ACM Conference on Applications, Technologies, Architectures and Protocols for Computer Communication (SIGCOMM)*, San Diego, CA, USA, 2001; 149–160.
5. Maymounkov P, Mazières D. Kademlia: a peer-to-peer Information System Based on the XOR Metric, *Proceedings of the 1st International Workshop on Peer-to Peer Systems (IPTPS)*, Cambridge, MA, USA, 2002; 53–65.
6. Rowstron A, Druschel P. Pastry: scalable, decentralized object location, and routing for large-scale peer-to-peer systems, *Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms*, Heidelberg, Germany, 2001; 329–350.
7. Yang Y, Yang L. A survey of peer-to-peer attacks and counter attacks. In *Proceedings of the International Conference on Security and Management*, SAM'12: Las Vegas, NV, US, 2012; 176–182.
8. Trifa Z, Khemakhem M. Taxonomy of structured P2P overlay networks security attacks. *World Academy of*

Science, Engineering and Technology 4 2012; **6**(4): 468–475.

9. Marti S, Garcia-Molina H. Taxonomy of trust: categorizing P2P reputation systems. *Computer Networks* 2006; **50**(4): 472–84.
10. Douceur JR. The Sybil attack. In *Proceedings of the First International Workshop on Peer-to-Peer Systems*, IPTPS'02. Springer-Verlag: London, UK, UK, 2002; 251–260.
11. Singh A, Ngan T, Druschel P, Wallach D S. Eclipse attacks on overlay networks: threats and defenses, *Proceedings of the 25th IEEE International Conference on Computer Communications*, Barcelona, Spain, 2006; 1–12.
12. Srivatsa M, Liu L. Vulnerabilities and security threats in structured overlay networks: a quantitative analysis, *Proceedings of the 20th Annual Computer Security Applications Conference*, Tucson, AZ, USA, 2004; 252–261.
13. Butler K, Ryu S, Traynor P, McDaniel P. Leveraging identity-based cryptography for node ID assignment in structured P2P systems. *IEEE Transactions on Parallel and Distributed Systems* 2009; **20**(12): 1803–1815.
14. Baumgart I, Mies S. S/Kademlia: a practicable approach towards secure key-based routing. In *Proceedings of the 13th International Conference on Parallel and Distributed Systems*, Vol. 2. IEEE Computer Society: Washington, DC, USA, 2007; 1–8.
15. Aiello L M, Milanesio M, Ruffo G, Schifanella R. Tempering Kademlia with a robust identity based system. In *Proceedings of the 8th International Conference on Peer-to-Peer Computing (P2P)*. IEEE Computer Society: Washington, DC, USA, 2008; 30–39.
16. Fantacci R, Maccari L, Rosi M, Chisci L, Aiello LM, Milanesio M. Avoiding eclipse attacks on kad/kademlia: an identity based approach. In *Proceedings of the IEEE International Conference on Communications (ICC)*. IEEE Press: Piscataway, NJ, USA, 2009; 983–987.
17. Aiello LM, Milanesio M, Ruffo G, Schifanella R. An identity-based approach to secure P2P applications with Likir. *Peer-to-Peer Networking and Applications* 2011; **4**: 420–438.
18. Aiello LM, Ruffo G. LotusNet: tunable privacy for distributed online social network services. *Computer Communications* 2012; **35**(1): 75–88.
19. Castro M, Druschel P, Ganesh A, Rowstron A, Wallach DS. Secure routing for structured peer-to-peer overlay networks. *ACM Operating Systems Review (OSR)* 2002; **36**: 299–314.
20. Rowaihy H, Enck W, McDaniel P, La Porta T. Limiting Sybil attacks in structured P2P networks, *Proceedings of the 26th IEEE International Conference on Computer Communications*, Anchorage, Alaska, USA, 2007; 2596–2600.
21. Da Costa Cordeiro WL, Santos FR, Mauch GH, Barcelos MP, Gaspary LP. Identity management based on adaptive puzzles to protect P2P systems from Sybil attacks. *Computer Networks* July 2012; **56**(11): 2569–2589.
22. Lu C. Detection and defense of identity attacks in P2P network. In *Advances in Computation and Intelligence*, vol. 5821, Lecture Notes in Computer Science. Springer Berlin: Heidelberg, 2009; 500–507.
23. Wang G, Konolige T, Wilson† C, Wang X, Zheng H, Zhao BY. You are how you click: clickstream analysis for Sybil detection. In *Proceedings of the 22nd Usenix Security Symposium*, USENIX'13. USENIX Association: Washington, DC, USA, 2013; 241–256.
24. Yu H, Kaminsky M, Gibbons P, Flaxman A. SybilGuard: defending against Sybil attacks via social networks. *IEEE/ACM Transactions on Networking* 2008; **16**(3): 576–589.
25. Yu H, Gibbons P, Kaminsky M, Xiao F. SybilLimit: a near-optimal social network defense against Sybil attacks. *IEEE/ACM Transactions on Networking* 2010; **18**(3): 885–898.
26. Tran N, Li J, Subramanian L, Chow SS. Optimal Sybil-resilient node admission control. In *Proceedings of the 30th IEEE International Conference on Computer Communications*, INFOCOM'11. IEEE Press: Shanghai, China, 2011; 3218–3226.
27. Cao Q, Sirivianos M, Yang X, Pregueiro T. Aiding the detection of fake accounts in large scale social online services. In *Proceedings of the 21nd Usenix Security Symposium*, USENIX'12. USENIX Association: Boston, MA, USA, 2012; 1–14.
28. Xue J, Yang Z, Yang X, Wang X, Chen L, Dai Y. VoteTrust: leveraging friend invitation graph to defend against social network Sybils, *Proceedings of the 34th IEEE International Conference on Computer Communications*, Turin, Italy, 2013; 2400–2408.
29. Shi L, Yu S, Lou W, Hou T. SybilShield: an agent-aided social network-based Sybil defense among multiple communities, *Proceedings of the 34th IEEE International Conference on Computer Communications*, Turin, Italy, 2013; 1034–1042.
30. Chaum D. Blind signatures for untraceable payments, *Proceedings of the Advances in Cryptology*, Santa Bárbara, CA, USA, 1983; 152–156.
31. Henry R, Goldberg I. Formalizing anonymous blacklisting systems. In *Proceedings of the IEEE Symposium on Security and Privacy*, SP'11. IEEE Computer Society: Washington, DC, USA, 2011; 81–95.
32. Köpsell S, Wendolsky R, Federrath H. Revocable anonymity, *Proceedings of the International*

Conference on Emerging Trends in Information and Communication Security, Freiburg, Germany, 2006; 206–220.

33. Abadi M, Needham R. Prudent engineering practice for cryptographic protocols. *IEEE Transactions on Software Engineering* 1996; **22**(1): 6–15.
34. Cooper D, Santesson S, Farrell S, Boeyen S, Housley R, Polk W. *Internet X.509 public key infrastructure certificate and certificate revocation list (CRL) profile*, May 2008. RFC 5280 (Updates by RFC 6818), Available from: http://www.ietf.org/rfc/rfc5280.txt.
35. Brunner R. A performance evaluation of the Kad-protocol. *PhD Thesis*, Fakultät für Mathematik und Informatik, Universität Mannheim, Germany, November 2006.
36. Fan CI, Sun WZ, Huang VSM. Provably secure randomized blind signature scheme based on bilinear pairing. *Computers & Mathematics with Applications* 2010; **60**(2): 285–293.
37. Bernstein DJ, Lange T. *eBACS: ECRYPT benchmarking of cryptographic systems*. Available from: http://bench.cr.yp.to [Accessed 4 September 2013].
38. Taverne J, Faz-Hernández A, Aranha DF, Rodríguez-Henríquez F, Hankerson D, López J. Software implementation of binary elliptic curves: impact of the carry-less multiplier on scalar multiplication. In *Cryptographic Hardware and Embedded Systems - CHES 2011*, vol. 6917, Lecture Notes in Computer Science. Springer Berlin: Heidelberg, 2011; 108–123.
39. Ying G, Jiang Z. Research on CRL distribution in P2P systems. In *Proceedings of the 2nd IEEE International Conference on Computer Science and Information Technology. ICCSIT' 09*. IEEE Computer Society: Beijing, China, 2009; 574–577.
40. Huang J, Wang Z, Qiu Z, Chen M. Theoretical analysis of issuing mechanism in distributive digital certificate revocation list. In *Proceedings of the International Conference on Computer and Electrical Engineering. ICCEE' 08*. IEEE Computer Society: Phuket, Thailand, 2008; 199–203.
41. Caubet J, Gañan C, Esparza O, Munoz JL, Mata-Díaz J, Alins J. Certificate revocation list distribution system for the KAD network. *The Computer Journal* 2013. DOI: 10.1093/comjnl/bxt037.
42. Avramidis A, Kotzanikolaou P, Douligeris C, Burmester M. Chord-PKI: a distributed trust infrastructure based on P2P networks. *Computer Networks* 2012; **56**(1): 378–398.

APPENDIX A. ABSTRACT SYNTAX NOTATION ONE 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 }
```