Security for Decentralised Service Location

Exemplified with Real-Time Communication Session Establishment

Jan Seedorf

August 2012

Dissertation

eingereicht an der Fakultät für Informatik und Mathematik der Universität Passau zur Erlangung des akademischen Grades eines Doktors der Naturwissenschaften

Erster Gutachter: Prof. Dr. Joachim Posegga
Zweiter Gutachter: Prof. Dr. Dieter Gollmann

Eidesstattliche Erklärung

Ich versichere an Eides statt, dass ich die vorliegende Dissertation selbständig, ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel verfasst habe und dass alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, als solche gekennzeichnet sind.

Heidelberg, den 30. August 2012

Jan Seedorf

Acknowledgements

Many people have supported me over the years and helped me in various ways to finish this thesis. I am thankful to Joachim Posegga and Dieter Gollmann for their scientific advise and help regarding my work and this thesis, but also for all the other things I could learn from them. I am also very thankful to my family for their support and understanding.

In no particular order, I also would like to thank the following people for their support: Saverio Niccolini, Martin Stiemerling, Nico d'Heureuse, Heinrich Stüttgen, Jürgen Quittek, Stefan Schmid, Marco Liebsch, Dirk Westhoff, Jens-Matthias Bohli, Sebastian Kiesel, Thomas Schmidt, Matthias Wählisch, Henrich Pöhls, Daniel Schreckling, Bastian Braun, Martin Johns, Hannah Lee, Sebastian Schinzel, Sebastian Gajek, Robert Olotu, Vijay Gurbani, Enrico Marocco, Christian Muus, Kristian Beckers, Stephan Sutardi, Frank Ruwolt, Ilona Rappu, Marco Cornolti, Da Zheng, Frank Fransen, Hendrik Scholz, Eric Chen.

Most of all, I am deeply grateful to my wife, Tatjana, for her continuous understanding, support without doubt, thoughtful advise, and frequent encouragement.

Heidelberg, August 2012 - Jan Seedorf

Contents

Chapter 1

Introduction

Contents

1.1 Motivation

For many years, distributed computer systems have been dominated by the *client-server paradigm*. Certain dedicated entities (*servers*) in the network provide services which other entities (*clients*) can use by connecting to those special entities. A simple example for this paradigm is the World-Wide-Web, where content can be accessed by clients which connect to dedicated web-servers. However, around 1999 a new approach for distributed systems has appeared: *Peer-to-Peer (P2P) computing*. In networks based on this novel paradigm, all entities are considered equal and provide equivalent services to other entities. At the same time, all entities can use services from all other participants of the network. Examples for popular and widely used P2P-systems are *Skype* (a P2P-based multimedia communications network) [6], *BitTorrent* (P2P-based file-sharing) [3], or *PPlive* (a P2P-based video streaming service) [16]. P2P applications contribute a significant part to the overall traffic on the Internet, to the extent that properly handling the huge amounts of traffic generated by P2P applications is a major concern for Network Service Providers [236].

Compared to client-server systems, P2P-computing offers the advantages of load-balancing, resilience to failure of participating nodes, and scalability [158] [43]. However, securing such systems imposes new research challenges. To better understand why P2P-computing cannot simply re-use existing and well-understood security approaches from the client-server world, consider the following important differences between client-server and P2P-based systems:

1. P2P-systems lack a central, persistent authority in the network which can provide security services such as authentication of entities.

2. P2P-systems are characterised by constant fluctuation in membership (so-called *churn*), which implies that a single entity may use the same service from different entities over time.

3. The entities a service is used from cannot be considered trustworthy, whereas in client-server systems security solutions often rely on the assumption that a server is non-compromised and follows a protocol in a non-malicious way.

These differences have important implications when considering security. For instance, the security of many client-server applications is based on credentials which have been exchanged between clients and servers a priori. Consider pre-installed TLS [107] certificates in web-browsers as an example. These certificates enable authentication of *Hyper Text Transport Protocol (http)* [115] servers as well as confidentiality and integrity of http traffic (i.e. *https* [214]). In contrast, P2P-systems are characterised by a high-degree of membership fluctuation. As a consequence, entities may connect each time to a different network participant for a single service, rendering solutions infeasible which are based on *persistent* credentials with the entity that is providing the service.

In addition, peers offering services may in principle act maliciously. This opens the door for attacks on availability of the P2P-application where participants refuse to follow the P2P-protocol, disrupting overall functionality. In summary, the decentralised nature of P2P-computing calls for new approaches in order to ensure confidentiality, integrity, authentication, and availability in such systems.

More recently, researchers have proposed *Distributed Hash Tables (DHTs)*, e.g. Chord [266], CAN [210], Pastry [223], or Tapestry [293]. DHTs essentially constitute a distributed database with formally proven properties regarding scalability, load balancing, and robustness against failure of nodes. P2P networks based on DHTs are often referred to as *Structured Overlay Networks*[1] [158]. DHTs enable new kinds of P2P applications because they can offer reliable decentralised storage and retrieval of data items: Given that a data item is stored in the network, DHTs can guarantee retrieval of the data item as well as an upper bound on the necessary number of messages to locate the desired data item in the DHT.

Several of these new kinds of P2P applications use a DHT to locate services offered by entities in the network. As an example, a fully decentralised replacement of the *Domain Name System (DNS)* [179] [180] has been proposed [99] [207] [196], where the resolution of hostnames is facilitated by a DHT-based P2P-network. As another example, locating multimedia communication—e.g. *Voice-over-IP (VoIP)*—users with a DHT has been investigated by researchers [252] [78] and is currently being standardised by the *Internet Engineering Task Force (IETF)* [140]. This approach is commonly referred to as *P2PSIP*, because technically a P2P-based version of the session establishment part of

[1] In the literature, the terms *P2P network* and *Overlay Network* are often used interchangeably, especially when referring to Structured P2P networks [266] [158] [159].

the *Session Initiation Protocol (SIP)* [222] is being envisioned. We refer to these kinds of P2P-applications as *Decentralised Service Location* because these applications use the P2P-network solely for locating services: Once the node offering the desired service has been located (e.g. a web-server in the case of decentralised DNS, or a VoIP terminal in the case of decentralised VoIP), communication takes place in a client-server fashion directly between the initiator of the DHT lookup request and the desired host offering the service.

Decentralised Service Location, i.e. finding an application communication endpoint based on a DHT, is a fairly new concept. The precise security implications of this approach have not been studied in detail. More importantly, a detailed analysis regarding the applicability of existing security solutions to this concept has not been conducted. As highlighted above, in many cases existing client-server approaches to security may not be feasible. In addition, to understand the necessity for such an analysis, it is key to acknowledge that Decentralised Service Location has some unique security requirements compared to other P2P applications such as filesharing or live streaming.

First, in contrast to other DHT-based applications, Decentralised Service Location means that a *location-binding*, i.e. the binding between a service and a location in the network, is stored in the DHT. This implies that location-bindings are stored at arbitrary, non-trustworthy nodes in the network. Hence, particular attention must be paid to the protection of the integrity of these bindings. Without proper integrity protection, man-in-the-middle attacks or impersonation attacks (e.g. on VoIP identities) are feasible. Second, Decentralised Service Location has *soft* real-time requirements [44]. This means that the answer for a location query is expected to be retrieved from the DHT within a certain amount of time (e.g. in the order of several seconds for a VoIP call establishment or a DNS request). Maintaining availability of the DHT lookup service in the presence of attackers is hence of particular importance for this type of application. Third, considering specifically the case for multimedia communication session establishment, unique application requirements need to be considered such as Spam prevention, reachability of emergency services, or the possibility to conduct Lawful Interception by law enforcement agencies.

Studying and understanding in detail the aforementioned security requirements for Decentralised Service Location is a core motivation for this thesis. In addition, a main objective of our work is to develop innovative security solutions for addressing those requirements identified which are not sufficiently addressed by existing solutions.

1.2 Research Questions and Goals

This thesis concerns the security challenges for Decentralised Service Location. As outlined above, the goals of our work are on the one hand to precisely understand the security requirements and research challenges for Decentralised Service Location, and on the other hand to develop and evaluate corresponding security mechanisms.

One simple and straightforward way to secure P2P-networks would be to add cen-

tralised components (e.g. a Certificate Authority to authenticate nodes or identities). Indeed, many existing P2P-networks use exactly this approach to fulfil application security requirements, e.g. *Skype* [6]. However, from a research perspective, adding a central component is a rather trivial solution. It can therefore not be considered a research challenge. More importantly, adding a central authority does not lie in the true spirit of P2P-computing because with this approach, systems cannot be considered *pure P2P* anymore. True P2P networks are per definition decentralised, and each node offers the same functionality. In particular Distributed Hash Tables—the focus of our work—are fully decentralised structures in the sense that functionality is evenly distributed among nodes in the network, with no specific entity playing a central or special role. The primary objective of this thesis is therefore to develop—as much as possible—*decentralised* solutions in response to the security challenges derived.

It is important to realise that completely decentralised security solutions for P2P-systems may not be possible in all cases. Douceur has shown that completely preventing forged entities in distributed systems is not possible without a central authority [109]. It is an open research question whether a central authority is necessary to completely prevent other important attacks on P2P-networks. Intuitively, it seems that security must be bootstrapped at some persistent, trustworthy entity. Consequently, the urging research question is rather how to decentralise security solutions for pure, fully-decentralised P2P-systems best, with as few centralised security service dependency as possible. This thesis investigates this research question for Decentralised Service Location.

The goals of this thesis can be summarised as follows:

1. ***Security Analysis for Decentralised Service Location*** A first goal is to analyse the security challenges for Decentralised Service Location. From this analysis, security requirements for this class of applications will be derived. This analysis will be driven by the example application of Voice-over-IP (VoIP).

2. ***Examination and Analysis of Existing Solutions*** A further goal of this thesis is to provide a detailed analysis of existing solutions to secure Decentralised Service Location based on the security requirements identified. The objective is to carve out the remaining research challenges by assessing to what extent existing mechanisms are insecure, inefficient, or based on centralised components.

3. ***Investigation and Assessment of Innovative, Decentralised Security Solutions*** The primary objective of our work is to develop innovative *decentralised* security mechanisms and assess the effectiveness of these mechanisms. The goal is to design effective security solutions which rely on as few central components as possible. The proposed solutions shall be evaluated qualitatively as well as quantitatively regarding the security, the performance, and the degree of decentralisation they offer.

The underlying research questions to be answered are the following:

Can the security requirements for Decentralised Service Location be fulfilled with fully decentralised security mechanisms? If not, to what degree is the decentralisation of security solutions possible? For each security requirement, our goal is to develop a decentralised solution, i.e. relying on as few central entities as possible. It is an open research question to what extent decentralised solutions can secure the individual security challenges. For instance, is it technically possible to design fully decentralised solutions for all security challenges? If such solutions cannot be found in response to all challenges, it is natural to ask to what extent decentralisation is possible. In other words, for each case (i.e. challenge) we are interested to investigate with how few centralised components we can design an adequate solution.

What is the effect on application-level performance of such solutions? Is there a trade-off between decentralisation, performance, and security? In other words, an interesting research question is what the implications of decentralisation are regarding application performance. For instance, is it possible not only to secure Decentralised Service Location with decentralised solutions, but can these solutions still provide location of a service within a reasonable amount of time? Further, we are interested in investigating if there are cases where decentralisation and performance are potentially conflicting goals. If yes, an interesting question would be to analyse the trade-off between different application objectives and requirements.

1.3 Thesis Overview

This thesis is organised as follows. First, fundamentals are explained and the scope of the thesis is defined. Chapter 2 provides fundamentals for the scope of our work. It explains the necessary background information—regarding P2P networks and IP-based multimedia communication systems—to understand the technical challenges and corresponding solutions presented in this thesis. Further, Decentralised Service Location is defined and *P2PSIP* is explained technically as a prototypical example. Chapter 3 defines the scope of this thesis. A security analysis for P2PSIP is presented. Based on this security analysis, security requirements for Decentralised Service Location and the corresponding research challenges—i.e. security concerns not suitably mitigated by existing solutions—are derived.

Second, several *decentralised* solutions are presented and evaluated to tackle the security challenges for Decentralised Service Location. Chapter 4 presents decentralised algorithms to enable availability of the DHTs lookup service in the presence of adversary nodes. These algorithms are evaluated via simulation and compared to analytical bounds. In Chapter 5, a cryptographic approach based on self-certifying identities is illustrated and discussed. This approach enables decentralised integrity protection of location-bindings. A decentralised approach to assess unknown identities is introduced in Chapter 6. The approach is based on a Web-of-Trust model. It is evaluated via prototypical implementation.

For the requirement of Lawful Interception, a satisfactory technical solution seems very

difficult to design. Instead, Chapter 7 provides a technical analysis of Lawful Interception in P2PSIP systems. Potential technical solutions are outlined and discussed. Chapter 8 presents a prototype implementation of a security-enhanced P2PSIP system. Several of the security techniques proposed in Chapters 4 and 5 have been evaluated with this prototype. The results of these experiments are presented and discussed.

Finally, the thesis closes with a summary of the main contributions and a discussion of open issues. Chapter 9 provides this reflection of the main contributions and contrasts them with the original goals of our work. Further, it discusses open issues and possible future work based on the results obtained through this thesis.

This thesis also contains several appendices. A considerable part of this thesis including figures has been originally published in scientific conferences and journals or by other means [189] [203] [235] [238] [239] [240] [241] [242] [243] [244] [246] [247] [248]. Appendix A lists these associated publications which have been published prior to the publication of this thesis and explains how they relate to this thesis. In Appendix B, details about simulations, prototypical implementations, and the setup of experiments conducted during the course of our work are provided.

Chapter 2

Background

Contents

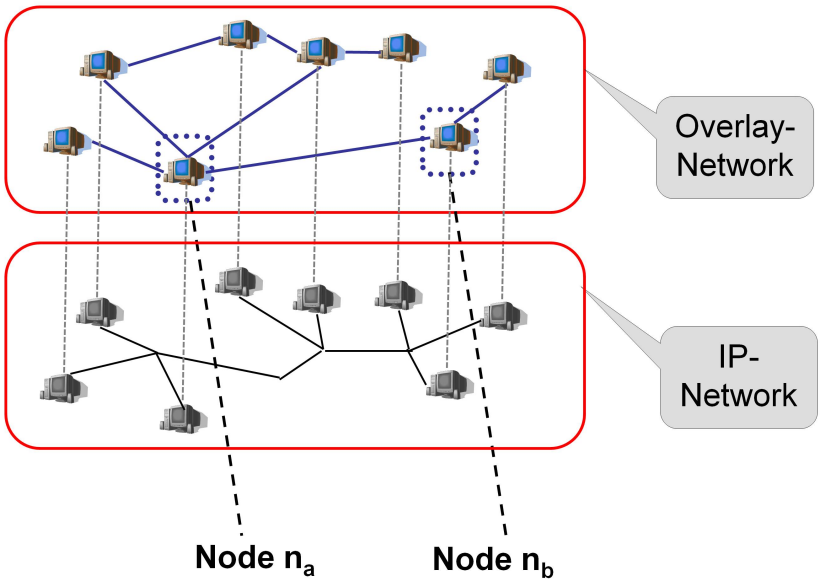

Figure 2.1: Different Layers in a P2P System

2.1 Peer-to-Peer Networks

Peer-to-peer networks follow a different paradigm than client-server based systems. A key underlying attribute is that each node participates in the network by offering and using services at the same time. There is no central control and the network organizes itself in a dynamic way. Another key characteristic is that P2P networks are constructed at the application layer: The P2P network uses the underlying network (e.g. a TCP/IP network) for transporting messages between nodes which have a direct link at the P2P-layer. Thus, a direct link on the overlay layer may imply several hops on the underlying network layer. Figure 2.1 [241] shows the different layers in a typical P2P network. Note that nodes n_a and n_b in the figure are directly connected on the P2P overlay layer, but on the underlying IP-network several routing hops exist among these nodes.

There exists a vast amount of literature, proposed algorithms, and deployed systems in the field of P2P computing. Here we try to give an overview on different kinds of P2P systems. Instead of providing an exhaustive listing of existing P2P systems, we will outline key approaches and survey important types of applications which have been deployed.

While the term *P2P network* refers rather to a certain network topology and the term *P2P system* is usually used to describe a complete system (which provides an application or an overall functionality besides routing of messages), the distinction between these two terms is not always clear in the literature and they are often used non-discriminable. We will use the notation *P2P network* preferably to refer to a network topology where routing of messages is important; we will use the notation *P2P system* preferably when

we refer to an overall communication system which provides a specified functionality (in the sense of an upper-layer application) other than topology-based routing of messages to participating nodes. It should be clear from the context what a certain usage of one of these terms refers to. As a more general term which comprises both P2P networks and P2P systems, we will sometimes use the term *P2P computing*. Because P2P networks are built at the application layer and use the underlying network for the exchange of messages, P2P systems are also called overlay networks by many researchers. We follow this usage and use the terms *P2P network* and *overlay network* interchangeably in this thesis.

2.1.1 Defining Peer-to-Peer Computing

Peer-to-Peer (P2P) computing is not easy to define and various definitions exist in the literature. We first cite some existing definitions for P2P computing and then define P2P computing for the course of this thesis.

A very imprecise definition is given in [281]:

> *In a short word, P2P is a special distributed system on the application layer, where each pair of peers can communicate [with] each other through the routing protocol in P2P layers. [281]*

Still, this definition highlights two important characteristics of P2P-systems: P2P systems are formed at the application layer and P2P networks define a specific routing protocol to communicate which each other on the application layer. Hence, P2P-routing refers to application-layer routing. Lua et al. [158] give a long and also somewhat imprecise definition of P2P networks, which they refer to as *P2P overlay networks*:

> *Peer-to-peer (P2P) overlay networks are distributed systems in nature, without any hierarchical organization or centralized control. Peers form self-organizing overlay networks that are overlayed on the Internet Protocol (IP) networks, offering a mix of various features such as robust wide-area routing architecture, efficient search of data items, selection of nearby peers, redundant storage, permanence, hierarchical naming, trust and authentication, anonymity, massive scalability, and fault tolerance. Peer-to-peer overlay systems go beyond services offered by client-server systems by having symmetry in roles where a client may also be a server. It allows access to its resources by other systems and supports resource sharing, which requires fault-tolerance, self-organization, and massive scalability properties. Unlike Grid systems, P2P overlay networks do not arise from the collaboration between established and connected groups of systems and without a more reliable set of resources to share. [158]*

An important aspect of this definition is the distinction between P2P networks and GRID-systems. In [155] a peer-to-peer (P2P) system is defined as follows:

> *Peer-to-peer (P2P) systems are distributed systems without any centralized control or hierarchical organization, where the software running at each node is equivalent in functionality. [155]*

This definition (and also the previous definition by Lua et al. [158]) defines pure P2P architectures only. Yet many networks are also considered P2P even though they employ a central authority (so-called *hybrid* P2P systems) or use nodes that offer more functionality than others (so-called *super nodes*). There are more general definitions of P2P networks which try to include such subclasses. For instance, according to Shirky [250], what makes P2P systems distinctive is the fact that P2P applications take advantage of the resources at the edge of the Internet:

> *Peer-to-peer is a class of applications that take advantage of resources (storage, cycles, content, human presence) available at the edges of the Internet. [250]*

This definition is very general, but includes hybrid systems as well as P2P-systems with supernodes. It rather focuses on the key property of P2P computing. Androutsellis-Theotokis et al. give a similar definition of P2P systems [43]:

> *Peer-to-peer systems are distributed systems consisting of interconnected nodes able to selforganize into network topologies with the purpose of sharing resources such as content, CPU cycles, storage and bandwidth, capable of adapting to failures and accommodating transient populations of nodes while maintaining acceptable connectivity and performance, without requiring the intermediation or support of a global centralized server or authority. [43]*

However (compared to the definition by Shirky [250]), their definition also includes the properties of *failure-adaptation* and *handling transient node populations* by *self-organisation*. In addition, they demand *acceptable connectivity and performance*, a rather imprecise property.

This thesis focuses on fully decentralised, structured[1] P2P networks. Self-organisation and the ability to handle frequent joining and leaving of nodes (so-called *churn*[2]) are key properties of such systems. Further, resiliency to node failures is an important part of self-organisation in fully decentralised P2P systems. However, the extent to which such systems provide *connectivity* and *performance* should not be part of a general P2P definition. In fact, many unstructured[3] P2P networks cannot guarantee either of these two properties. Therefore, connectivity and performance are left out of our general definition of a P2P network below.

[1] Structured peer-to-peer networks are defined in Subsection 2.1.2.

[2] *Churn* describes the degree of node fluctuation caused by nodes leaving and entering the network.

[3] Unstructured peer-to-peer networks are defined in Subsection 2.1.2.

Thus—based on the previously listed definitions found in the literature ([158] [155] [250] [43])—for the course of this thesis we define a P2P network (and a fully decentralised P2P network as a special case of a P2P network) as follows:

> **Definition 1** (**P2P network**): *A P2P network is a scalable distributed system consisting of interconnected nodes with the purpose of sharing resources such as content, CPU cycles, storage, or bandwidth which is able to self-organise into a network topology such that the network is capable of adapting to node failures and able to handle joining and leaving of participating nodes.*

> **Definition 2** (**Fully decentralised P2P network**): *A fully decentralised P2P network is a P2P network according to Definition 1 without any centralised control or hierarchical organisation, where the software running at each node is equivalent in functionality, so that each node can be considered resource provider and resource consumer at the same time.*

The extent to which a P2P network is able to handle node failures and churn is determined by its structure, e.g. whether it is an unstructured or structured network and by the properties of this structure (see further Subsection 2.1.2). But even unstructured P2P networks are characterised by the property that nodes can frequently join or leave the network (or fail) while overall network functionality can be provided. Therefore, these properties are included in Definition 1.

2.1.2 Unstructured vs. Structured P2P Systems

In general, P2P networks can be classified into *Unstructured* and *Structured* [158]. Below we describe the main characteristics of each type of P2P system.

Unstructured P2P networks Early systems (e.g. the original *Gnutella* [226]) used flooding for message routing in the network. Any node receiving a search request will broadcast this message to all its neighbours. The message has a time-to-live (TTL) value which is reduced at every hop to prevent messages from being routed in the network forever. Such *unstructured* systems cannot give any formal guarantees that a message in the network will reach its destination. Furthermore, broadcast messages impose an unnecessary traffic burden on the network.

Example 2.1: *Figure 2.2 [158] shows the flooding approach used for search of data items by Gnutella. In the example, node* `Peer_1` *sends out a query to all its neighbours. The neighbours in return forward the query to all of their neighbours. Eventually,* `Peer_5` *responds with a message saying that it has the desired content.* `Peer_2` *forwards this response message to* `Peer_1` *so that it can start downloading the content from* `Peer_5`*. It can be observed that this approach is not very effective and causes heavy traffic loads.*

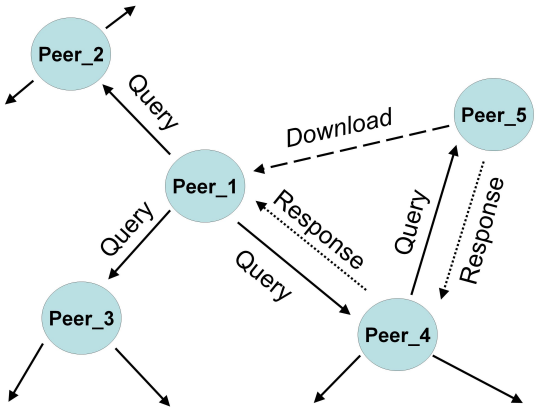

Figure 2.2: Flooding in a Gnutella Network

More importantly, it cannot be guaranteed that a data item which is stored in the network will always be found with this approach: If the query does not reach a target node which stores the requested resource within the TTL, the query ends without success.

Lv et al. characterise unstructured P2P networks as not having "any precise control over the network topology or file placement" and in that "the placement of files is not based on any knowledge of the topology (as it is in structured designs)" [159]. They observe that in unstructured P2P networks "The most typical query method is flooding, where the query is propagated to all neighbors within a certain radius" and conclude that such "search mechanisms are extremely unscalable, generating large loads on the network participants" [159]. Similarly, Lua et al. describe unstructured P2P systems as follows: "An unstructured P2P system is composed of peers joining the network with some loose rules, without any prior knowledge of the topology. The network uses flooding as the mechanism to send queries across the overlay with a limited scope." [158].

Structured P2P networks To improve lookup time for a search request, so-called *Structured Overlay Networks*[4] have been proposed by researchers. These networks provide load balancing and efficient routing of messages. Structured overlay networks offer a substrate on which different applications can be built rather than a separate P2P network for every application (like most unstructured P2P file-sharing systems). The overlay provides content distribution and search for content to an application using it. Specifically, given a key (as a search request inserted by a participating node into the network), the network returns the node responsible for storing data belonging to that key. This *key-based routing* of structured P2P networks is based on so-called *Distributed Hash Tables (DHTs)*: Routing as well as node responsibility for keys is based on a hash function. In many cases, node-IDs and key-IDs are computed using the same, system-wide hash function. The node which has the ID *closest* to a certain key-ID is responsible for storing data for this key-ID.

[4]For structured P2P networks researchers often use the term *Structured Overlay Networks*. We follow this notation and use the terms Structured Overlay Network and structured P2P network interchangeably.

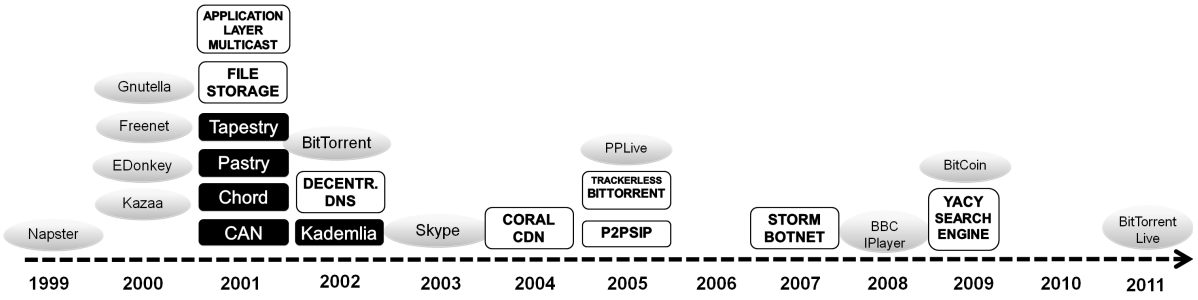

Figure 2.3: Evolution of P2P Networks

Lv et al. define structured P2P networks by the fact "that the P2P network topology (that is, the set of connections between P2P members) is tightly controlled and that files are placed not at random nodes but at specified locations that will make subsequent queries easier to satisfy. In systems with *tight structure*, the structure of the P2P network and the placement of files is extremely precise and so subsequent queries can be satisfied very efficiently." [159]. Lua et al. define *structured* in the context of P2P networks as follows: "The technical meaning of structured is that the P2P overlay network topology is tightly controlled and content is placed not at random peers but at specified locations that will make subsequent queries more efficient." [158].

Today, unstructured P2P networks are very common on the Internet due to their simple routing algorithms and low overhead [158]. In addition, many types of applications (e.g. file-sharing) do not need the formal guarantees offered by structured P2P networks. DHT-based (i.e. structured) P2P systems are heavily studied by researchers and have started to appear as real applications on the Internet. One example is the distributed tracker used in *BitTorrent* file-sharing networks, which is technically based on the Kademlia DHT [263]. In this thesis we focus on structured P2P systems, and in particular on Chord [267]: The types of applications we consider, i.e. service location, require guaranteed lookup success; a property which unstructured P2P networks cannot offer.

Figure 2.3 shows a time-lime which displays the year of appearance for selected P2P systems, focussing on DHTs and DHT applications. Unstructured P2P Networks are displayed in grey ellipses. DHTs are displayed in filled black boxes; DHT applications are displayed in black boxes with white background. It can be observed that unstructured P2P networks appeared prior to structured ones. Famous and widely used unstructured P2P-systems include *BitTorrent* (filesharing) [3], *PPLive* (live streaming) [16], *BBC IPlayer*[5] (live streaming) [2], *BitCoin* (a digital currency using a P2P network) [185], and *BitTorrent Live* (live streaming) [4]. *Skype* [6] is a very popular P2P-based multimedia communication software. If or to what extent Skype uses DHT technology as part of its session establishment procedure remains a secret [62] [69]. The most famous DHTs (Chord [267], CAN [210], Pastry [223], and Tapestry [293]) all were proposed at roughly

[5]The original IPlayer used P2P technology for content distribution. Later versions have been client-server based.

the same time, namely in 2001. Some of the first DHT applications that appeared were application-layer multicast [211] [85] [296], distributed file storage [110] [215] [103], and decentralised DNS [99]. Later, DHT-based *Content Delivery Networks (CDNs)* [117] and multimedia communications (*P2PSIP*) [252] have been proposed. A DHT-based BitTorrent tracker is widely used which spreads the tracker functionality among peers (so-called *trackerless* BitTorrent). More recent DHT applications include the use of Kademlia by the *Storm Botnet* [132] and *YaCy*, a DHT-based search engine [27].

2.1.3 Distributed Hash Tables

Distributed Hash Tables (DHTs) provide the substrate used by Structured Overlay Networks to provide efficient storage and location of data items with formal guarantees. A Distributed Hash Table essentially is a hash table that is distributed among the nodes participating in the network. Each node stores only a small part of the whole hash table for which it is responsible. Most DHTs use *consistent hashing* [147], which has the advantageous property that only a small fraction of data items has to be re-assigned each time a node enters or leaves the network. Thus, the network is capable of handling nodes entering and leaving the network at a high frequency without having to redistribute large amounts of data among nodes.

The two basic primitives provided by a DHT are *store* (*key*, *data*), and *lookup* (*key*) = *data*, similar to a regular hash table. Using the *lookup* primitive, any node can query the DHT with a key and the DHT returns the data item that corresponds to that key. Each node stores only a small part of all the data items. The DHT provides rules that specify which node is responsible for which part of the keyspace. The DHT protocol is designed in such a way that a message querying a particular key can be routed efficiently to the node responsible for that key and will always succeed in reaching the responsible node (in the absence of attackers).

Further, DHTs are capable of handling nodes entering and leaving the network at a high frequency. The design is such that it can guarantee consistent data storage and load balancing even when nodes enter and leave the network at a high frequency. Specifically, each DHT defines a *join* and a *leave* operation. These operations take care of the reassignment of key responsibility among nodes, and of updating routing tables at nodes, in the case that nodes *join* or *leave* the network (where node failure can be seen as a special kind of leave).

For the purpose of our work we define a DHT as follows:

> **Definition 3 (Distributed Hash Table):** *A Distributed Hash Table (DHT) is a distributed system that provides consistent distributed storage and retrieval of data items using one or more hash functions. Data is stored as (key, value) pairs. The DHT provides two basic functionalities which are distributed among all participating nodes and available to all participating nodes: store (key, data) to store a data item for a given key, and lookup*

(key) = data to retrieve a data item for a given key. In the absence of malicious nodes or large amounts of failures, a DHT can guarantee efficient data retrieval that scales with the number of participating nodes.

2.1.3.1 DHT Structure, Components, and Routing

For the course of this thesis we specify certain components of a DHT more formally[6]. A DHT consists of several structural elements and can be characterised by the elements and functions presented below. We use this formalisation to analyse the DHT we use in our work (Chord), categorise threats on DHTs, and to precisely specify our proposed extensions to Chord in an unambiguous representation.

A DHT consists of the following components:

- **Participating Nodes:** A DHT consists of participating nodes, which are all equal with respect to the functionality they offer to other nodes in the DHT. Nodes can send and receive messages based on the underlying network structure, e.g. a TCP/IP network. A node is thus a unique entity participating in the DHT which is able to send and receive messages and to provide routing of messages to other nodes. Each node has a node identifier $n \in I$, where I is an m-bit node identifier space.
- **Node Identifier Space:** In order to distinguish nodes, a unique external identifier $eID \in E$ is used, where E is an external identifier space. For instance, if the IP-address is used as an external identifier eID, the external identifier space E would be all possible IP-addresses, i.e. $E = [0 \ldots 255].[0 \ldots 255].[0 \ldots 255].[0 \ldots 255]$. In order to uniquely identify participating nodes in the system, nodes get assigned a node-ID $n \in I$, based on their external identifier $eID \in E$.
- **Node-ID Mapping:** A DHT provides a node mapping function $f_{node-mapping}{:}E \rightarrow I$ for mapping an external identifier $eID \in E$ onto a node identifier $n \in I$. Thus, $f_{node-mapping}(eID) = n \in I$. We denote with n_i the node-ID with the value i in decimal representation, assuming that $I \subset \mathbb{N}$ or that I can be mapped onto a subset of $\mathbb{N}$. We will use n (without any index) to denote a node with a node-ID which is of no particular importance in the context. Usually, $f_{node-mapping}$ is a one-way hash function $h_{node}()$ such as *SHA-1* [28]. Hence, an external identifier gets hashed to obtain the corresponding m-bit node-ID $n \in I$. The possible set of input values for this hash function $h_{node}()$—i.e. its domain, $domain(h_{node}())$—defines the external identifier space E. In the literature, the mapping from external identifier to node-ID is also referred to as *node-ID assignment*. Some DHTs let nodes chose their node-ID, e.g. CAN [210]. Other DHTs provide strict limitations on the external identifier to use and on node-ID assignment. For instance, Chord [267] computes the node-ID n of a participating node x by hashing the node's IP-address with a predefined hash function $h_{node}()$ such that $n = h_{node}(IP-address(x))$. In this case, the node identifier space I is determined by the range of the hash function $h_{node}()$.

[6] Our model is in part an extension of the model presented in [261].

- **Key Identifier Space:** In order to provide distributed storage of data items, a *key* is used to index data items. A $key \in K_S$ is a string used to uniquely identify a data item stored in a Distributed Hash Table, where K_S is the keyspace which defines the set of possible keys. A key for a specific data item gets mapped onto a key-ID by hashing its value. More generally, a mapping function is used to map keys onto key-IDs. A DHT provides a function $f_{key-mapping}$:$K_S \rightarrow K$ for mapping a key $\in K_S$ onto a key-ID k so that $f_{key-mapping}(key) = k \in K$, where K is an l-bit key identifier space. We denote with k_i the key-ID with the value i in decimal representation, assuming that $range(f_{key-mapping}) \subset \mathbb{N}$ or that $range(f_{key-mapping})$ can be mapped onto a subset of $\mathbb{N}$. We will use k (without any index) to denote a key where the key-ID is of no particular importance in the context. Usually, $f_{key-mapping}$ is a one-way hash function $h_{key}()$ such as *SHA-1* [28]. Hence, a key gets hashed to obtain the corresponding l-bit key-ID $k \in K$. The possible set of input values for this hash function defines the keyspace K_S.

- **Data Responsibility:** In order to enable reliable storage and retrieval of the data item for a given key, a data placement function is needed to map keys onto the node-ID space. Further, a function is needed to define which node is responsible for which key. A DHT provides a function $f_{data-placement} : K \rightarrow I$ that maps a key-ID $k \in K$ onto the node-ID space I. A responsibility function $f_{resp} : I \rightarrow I$ states which node $n \in I$ is responsible for storing $f_{data-placement}(k)$ for a given key $k \in K$. Thus, the data item for key k is stored at node $n = f_{resp}(f_{data-placement}(k))$. For any given key-ID k, the node $n \in I$ which is responsible for storing data items for k is called the *root* node for key-ID k, denoted $root_k$:

$$root_k = f_{resp}(f_{data-placement}(k)) \tag{2.1}$$

  For reliability, data items are redundantly replicated for each key k. A replication function $f_{replicate} : I \rightarrow I^{r-1}$ maps each key k onto $r-1$ other nodes which store the data for k as well (besides $root_k$). Hence, for each key overall r redundant data items are stored in the DHT (one at $root_k$ and $r-1$ ones at the other replica roots).

- **Routing Structure and Routing Scheme:** Any node participating in the DHT can invoke `store(key-id, value)` and `lookup(key-id)=value`. To facilitate this functionality which is provided to upper layers, a DHT uses a routing structure (e.g. a Cartesian space or a virtual ring) and a routing scheme (i.e. a set of rules that specify how routing takes place). The routing structure specifies the distance between nodes in the DHT. The routing structure of a DHT provides a distance function $dist : I \times I \rightarrow \mathbb{Z}^M$ that enables to calculate the distance between any two nodes n_i and n_j in the DHT. For unidimensional DHTs, $dist : I \times I \rightarrow \mathbb{Z}$ (i.e. $M = 1$). For multidimensional DHTs M is the dimension of the DHT. Since the routing structure determines the topology between nodes in the DHT, we will use the terms *routing structure* and *DHT structure* interchangeably. Each node stores locally routing tables with links to other nodes. A routing table T_r at each node n contains t links to nodes at some distance in the ID-space. Further, a second routing table T_s at each node n contains s direct neighbours in the DHT structure. Thus, each node stores a routing table T_r with links to t participating nodes in the

DHT and a second routing table T_s with links to s direct neighbours in the routing structure. A routing table function $f_{routing-table} : I \rightarrow I^t$ determines which t nodes are in the routing table $T_r(n)$ of any node n in the system. A neighbourhood function $f_{neighbor-table} : I \rightarrow I^s$ determines for each node n in the system which s nodes are in its routing table $T_s(n)$. For some DHTs, the size of the routing tables T_r and T_s are constants t and s, respectively, that depend on the size of the node identifier space I. In order to decide which nodes are placed in the routing tables, the functions $f_{routing-table}$ and $f_{neighbor-table}$ depend on the distance function provided by the routing structure. A routing function determines how to route a message which a node receives for a certain key. This routing function $f_{route} : I \rightarrow I$ specifies which entry a routing table returns upon receiving a message (lookup or storage) for key-ID k. Usually, DHTs route *greedy*:

$$f_{route}(f_{data-placement}(k)) = \\ (n_i \in T_r | dist(n_i, f_{data-placement}(k)) \leq dist(n_j, f_{data-placement}(k)) \forall n_j \in T_r \wedge j \neq i) \tag{2.2}$$

In other words, it is routed to exactly the node in the routing table that is closest to the node storing the data in the node identifier space.

- **Functions for Join, Leave, and DHT Maintenance:** Each DHT defines a set of rules for joining and leaving of nodes, and to maintain the routing tables of other nodes during such joining and leaving operations. These rules enable the DHT to handle the dynamic nature of P2P systems, so-called *churn*. Since we do not examine joining and leaving of nodes in our work, we do not define rules for handling churn formally.

- **State:** Since the system is dynamic, its *state* changes constantly. Σ denotes the set of possible states. At any state $\sigma_i \in \Sigma$ we have a set of nodes in the system, $\mathcal{N}(\sigma_i)$ (where $\mathcal{N}(\sigma_i) \subseteq I$). The set of all $|\mathcal{N}(\sigma_i)| = N$ nodes, their routing tables T_r and T_s, and all the data items stored in the system define the current state σ_i. We will analyse the system at a certain state $\sigma_i \in \Sigma$. We do not consider node joins and leaves. For simplicity, we will denote $\mathcal{N}(\sigma_i)$ with $\mathcal{N}$, given a specified state σ_i. Similarly, we will denote the number of nodes in the system at state σ_i, $|\mathcal{N}(\sigma_i)|$, simply with N.

Example 2.2: *Consider the following example for key-ID mapping: If a P2P-system would be used to replace the centralised Domain Name System (DNS) with a distributed DHT-based solution, the URL of a website would be the key for a lookup. Assuming that a predefined hash function $h_{key}()$ is used as $f_{key-mapping}$, any participating node in such a system could request the current IP-address for a given URL, u, by invoking $lookup(h_{key}(u)) = lookup(key\text{-}ID(u))$ in the DHT.*

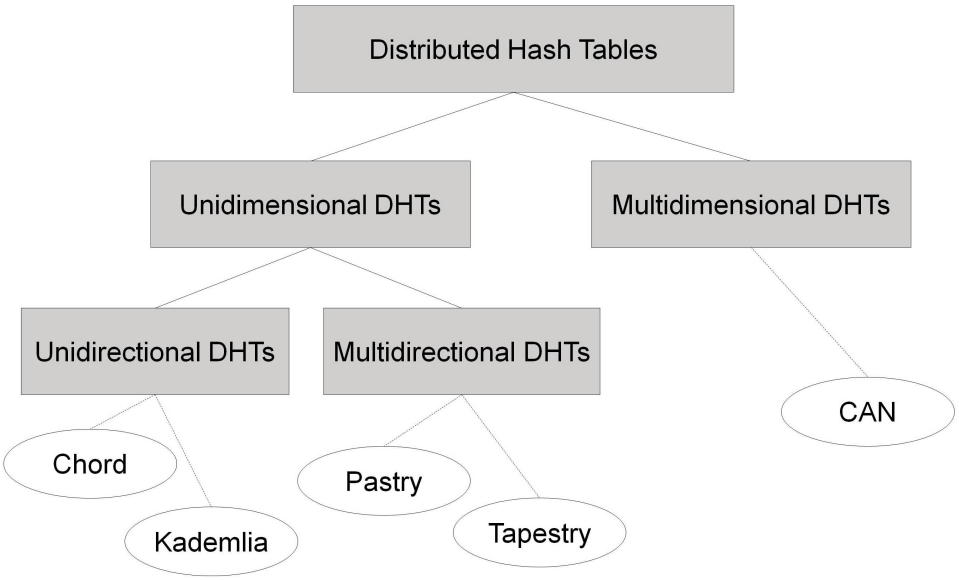

Figure 2.4: Classification of Distributed Hash Tables based on their Routing Structure

2.1.3.2 DHTs Proposed in the Literature

Several different DHT algorithms have been proposed, e.g. Chord [267], CAN [210], Pastry [223], Tapestry [293], or Kademlia [171]. Each of these DHTs provides its own routing scheme and can give formal guarantees (upper or lower bounds) on the number of hops needed for a search request to succeed.

Figure 2.4 shows a classification of some well-known DHTs based on their routing structure. As described previously, a multidimensional DHTs is characterised by the fact that the range of the distance function has more than one dimension, i.e. $dist : I \times I \rightarrow \mathbb{Z}^M$ where M is the dimension of the DHT. CAN [210] is an example of a multidimensional DHT. Unidimensional DHTs can be further classified into being either multidirectional (examples are Pastry [223], Tapestry [293]) or unidirectional (examples are Chord [267], Kademlia [171]). Unidirectional routing means that in the node-ID space "For any given point x and distance $d > 0$, there is exactly one point y such that $d(x, y) = d$" [158], where $d()$ is the DHT distance function ($dist()$ in our model, compare 2.1.3.1).

Below we provide an overview of the architecture and routing scheme for some popular and well-studied DHTs: CAN [210], Pastry [223], and Chord [267]. Chord is the DHT we will examine closely in this thesis, and for which we will provide routing extensions. Thus, our description of Chord is more detailed than for CAN and Pastry. Other DHTs that have proposed in the literature are Kademlia [171], Viceroy [165], Bamboo (based on Pastry) [216], or Accordion (based on Chord) [154]. For a more exhaustive tutorial on Distributed Hash tables and details of other DHT algorithms, the reader is referred to [158] [43] [164] [264].

CAN *CAN (Content-Addressable Network)* is a multidimensional DHT that uses a d-dimensional Cartesian coordinate space as the DHT node-ID space (I) and key identifier space (K) [210]. Each node is responsible for a *zone* in the coordinate space. Nodes randomly select a point p in the multidimensional node identifier space (i.e. $p \in I$) as their node-ID. When a new node n_{new} joins with randomly selected node-ID p_{new}, it contacts the node which is currently responsible for p_{new} in the CAN network. Then, the zone of the currently responsible node gets split into two halves, and one half is assigned as a new zone to the joining node. Keys are mapped to the coordinate space using a hash-function, i.e. $f_{key-mapping}() = h_{CAN}()$, where $h_{CAN}()$ is the chosen hash function for a given CAN DHT. Nodes are responsible for storing data items for the key-IDs within their zone.

Example 2.3: *Figure 2.5 shows a 2-dimensional CAN network with a coordinate space of* $[0,1] \times [0,1]$ *and six zones (i.e. 6 nodes). Three data items are stored in the network. If a new node* n_{new} *with randomly selected node-ID* $(0.44, 0.33)$ *would join, the node currently responsible for zone* $[(0.0, 0.5); (0.0, 0.5)]$*,* n_{z1}*, would split its zone and would assign the new zone* $[(0.25, 0.5); (0.0, 0.5)]$ *to* n_{new}*. Further,* n_{z1} *would transfer the data item for key-ID* $(0.43, 0.09)$ *to* n_{new}*.*

Each node n has a routing table, T_r, with links to the nodes responsible for the zones adjacent to the node-ID of n in the ID-space. Routing is greedy: nodes route via the link in T_r which is closest to the desired key. The average lookup path length in CAN is $O(n^{\frac{1}{d}})$; the size of each node's routing table T_r is $2d$ on average [210]. Hence, increasing the dimension of the coordinate space reduces the average routing path length at the price of increasing the size of the routing table maintained at each node. Application-level multicast is one example of a distributed application that has been built on top of CAN [211]. For a more detailed introduction to CAN the reader is referred to [210] [164] [158].

Pastry Pastry [223] is a unidimensional DHT that uses prefix based routing. Pastry assumes that each node is assigned a random, 128-bit node-ID. Node-ID space and key-ID space are the same and viewed as a circle ranging from 0 to $2^{128} - 1$. Node-IDs and key-IDs are expressed as a sequence of digits with base 2^b (b is a design parameter normally set to 4). The node with the ID closest to a key in the ID-space is responsible for that key. Pastry uses three routing tables. In a routing table T_r, each node n maintains $log_{2^b} N \times 2^b - 1$ entries to other nodes (where N is the maximum number of nodes in the network): Each row a ($a = 1, \ldots, log_{2^b} N$) in $T_r(n)$ contains $2^b - 1$ links to other nodes that share exactly the first $a - 1$ digits with n in the node-ID space. In a so-called *leaf set* T_s, each node maintains links to l (typically set to 2^b or 2×2^b) closest nodes in the ID-space. In each routing hop, the query is forwarded to a node in $T_r(n_c)$ which shares at least one more digit in its prefix with the key than the current hop node n_c. If the key k is within the leaf set of the current hop node (i.e. $k \in T_s(n_c)$), the root node for the key is found and the query is forwarded to the root node. Additionally, each node n has a *neighbour set* which contains links to m (typically set to 2^b or 2×2^b) nodes which are close to n not in the ID-space but according to a *network proximity metric* (e.g. IP-hop

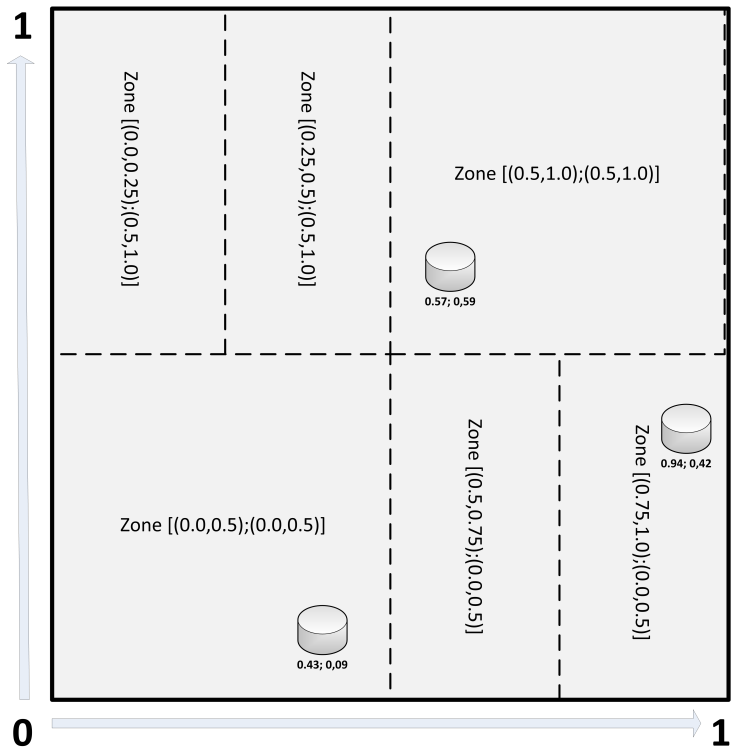

Figure 2.5: Example of a 2-dimensional CAN Network

distance or IP-layer round-trip-time). The neighbour set enables optimizations such as *proximity neighbour selection* [86], where during DHT join or maintenance operations the links in the normal routing table T_r are filled preferably with nodes that are *close* on the underlying routing structure.

Example 2.4: *Figure 2.6 shows an example for a routing table T_r of a Pastry node with a* 16*-bit node-ID* 2201*; $b = 2$, $N = 256$. In each row, entries share at least $a - 1$ digits with node* 2201*. Further, each column contains nodes with a specific digit at position a which is different than the digit at position a of node* 2201*. The nodes in parentheses* (2202)*,* (2203) *in the fourth row are supposed to indicate that there might not be such nodes with the required node-ID in the network (e.g. in case there are currently less than* 256 *nodes in the network).*

Routing in Pastry succeeds after a maximum of $log_{2^b} N$ hops [223]. Applications based on Pastry which have been suggested include large-scale distributed file storage [110], a publish/subscribe system [224], a cooperative messaging system [178], and application-level multicast [85]. Tapestry [293] is a DHT which has certain similarities to Pastry, e.g. routing is prefix based. Applications built on top of Tapestry include a large-scale distributed file storage system (called *Oceanstore*) [215], application-level multicast [296], and a decentralised spam-prevention system [295].

a	Node 2201			
1	0220	1333	---	3002
2	2013	2101	---	2331
3	---	2211	2220	2232
4	2200	---	(2202)	(2203)

Figure 2.6: Routing Table T_r of a Pastry Node with Node-ID 2201 in a Network with 256 Nodes

Chord Chord [267] uses the IP-address of a node as its external identifier (eID). A predefined, cryptographic hash function $h_{Chord}()$ (i.e. *SHA-1* [28] as suggested in [267], but in principle other collision-resistant hash functions could be used) maps any eID onto an m-bit node-ID n and also any key onto an m-bit key-ID k, $n = h_{Chord}(eID)$ and $k = h_{Chord}(key)$. Thus, for Chord we have: $I = K$ $(m = l)$ and $f_{key-mapping}() = f_{node-mapping}() = h_{Chord}() =$ SHA-1(). SHA-1 has the property of *collision-resistance*. This property of the hash function is needed to guarantee load balancing for *consistent hashing* [147] with high probability [266]. A virtual ring is the structure for the node identifier space I. In this ring the nodes are ordered clockwise from 0 to 2^{m-1} according to their node-ID n. Each node in the ring is responsible for storing the content of all key-IDs that are equal to or less than its own identifier but larger than the identifier of the node's direct predecessor in the Chord ring. For reliability against node failures, the data for k is also stored at $r-1$ nodes directly succeeding $root_k$ in the ring. When nodes join or leave the network, data items get re-assigned by neighbouring nodes in the node-ID space (through a so-called *stabilize* function which is frequently executed by nodes).

Formally, we have for Chord:

$$f_{data-placement}(k) = k, \ \forall k \in K \tag{2.3}$$

since $I = K$ and $f_{key-mapping}() = f_{node-mapping}() = h_{Chord}()$. Further, the root node for a given key with key-ID k is defined as follows:

$$root_k = succ(k) \tag{2.4}$$

where

$$succ(x) = \begin{cases} n_o \in \mathcal{N} | \, (n_o \geq x \wedge (\neg \exists n_p \in \mathcal{N} | n_o > n_p \geq x)) & \exists n_o \in \mathcal{N} | n_o \geq x \\ n_o \in \mathcal{N} | \, (n_o \leq x \wedge (\neg \exists n_p \in \mathcal{N} | n_o > n_p)) & otherwise \end{cases} \tag{2.5}$$

$succ()$ returns the *successor* node for any node or key in the Chord ring. The successor

can informally be described as "If identifiers are represented as a circle of numbers from 0 to $2^{m...1}$, then successor(k) is the first node clockwise from k." [267]. Note that the *clockwise* successor of a node or key is defined across $2^m - 1$ in the ring, i.e. the successor of a node or key clockwise closest to $2^m - 1$ in the ID-space is the node with the smallest node-ID in the ID-space (compare definition 2.5).

In its routing table T_r each node n stores links to m succeeding nodes in the ring. The jth entry in T_r contains the IP-address of the first node that follows n by at least 2^{j-1} in the virtual ring:

$$f_{routing-table}(n) = \left[succ((n + 2^0)\ mod\ 2^m), succ((n + 2^1)\ mod\ 2^m), ..., succ((n + 2^{m-1})\ mod\ 2^m)\right] \quad (2.6)$$

The first entry in T_r is the node directly succeeding n. The last entry in T_r contains a link to a node at least $\frac{2^m}{2}$ away from n in the ring. Routing is *greedy*: nodes forward messages to the node with the highest ID in their routing table that is smaller than the key-ID. Routing succeeds when the direct successor of a node has a larger ID than the key-ID k. This successor node is responsible for the key with key-ID k, i.e. this node is $root_k$. In Chord, $\vartheta = \frac{1}{2}\log N$ is the mean hop length of a routing path [164].

Additionally, each node keeps a link to its direct predecessor in the ring. Further, each node keeps a list of its s *direct successors*, T_s, to handle node failures. It is precisely specified how routing tables are filled, making routing tables in Chord *constrained*. This protects against certain attacks on routing tables [84] [251]. We will examine these attacks and how Chord protects against them closer in Chapter 4.

In Chord, any lookup has to pass the predecessor of the node storing the content for the key looked up. This is also referred to as the *shield problem* [184] [246] and a consequence of unidirectional greedy routing. We denote the predecessor of $root_k$ with $shield_k$ for any key k. We will look into the *shield problem* in more detail in Chapter 4.

Chord can route iteratively or recursively. With *iterative routing* a contacted node n_i returns the closest node to the key k from its routing table, node n_j, to the querying node. The querying node then contacts n_j to iteratively get closer to the key. With *recursive routing* a query is forwarded directly from n_i to n_j. n_j continues to route the query recursively, until it reaches $root_k$.

Example 2.5: *Figure 2.7 shows iterative routing in a Chord network of size $m = 6$ [246]. In the routing tables displayed the rightmost column shows to which other nodes in the DHT links exist. The two leftmost columns point out how to compute precisely which node is in the particular routing table entry, i.e. determining the value where the first node 'succeeding' this value in the ring must be in that routing table entry. In this example, a query node with node-ID 8, n_8, starts a lookup for key-ID $k = 57$. It starts an iterative lookup by sending a message to the node in its routing table that has a node-ID closest to but not larger than k which is n_{46} in this example $\langle 1 \rangle$. n_{46} replies by returning to n_8*

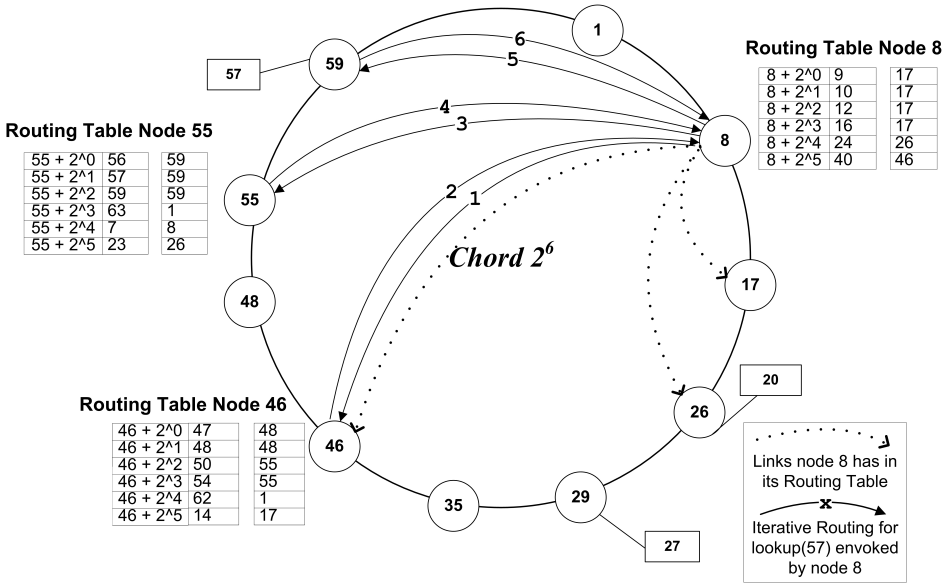

Figure 2.7: Iterative Routing in Chord

the node with the highest node-ID from its routing table not larger than the key-ID which is n_{55} *in this example* $\langle 2 \rangle$*.* n_8 *sends out a query message to* n_{55} $\langle 3 \rangle$*.* n_{55} *determines that the first node in its routing table, i.e. its direct successor in the ring, has a node-ID (*59*) which is higher than the key-ID (*57*).* n_{55} *concludes that this node must be responsible for key-ID* 57*, i.e. it must be* $root_{57}$*, and returns* n_{59} *to the query node* n_8 $\langle 4 \rangle$*. To retrieve the data item for key-ID* 57*,* n_8 *contacts* n_{59} $\langle 5 \rangle$ *which answers by sending the corresponding data item to* n_8 $\langle 6 \rangle$*.*

Many different applications have been proposed on top of Chord. For instance, an approach for using Chord to build a decentralised version of the *Domain Name System (DNS)* has been suggested in [99]. A distributed file-system built on top of Chord has been proposed in [103]. Other examples of applications researchers have designed on top of Chord are decentralised VoIP [252], a DHT-based version of the online citation index *CiteSeer* [268], or a DHT-based version of the *Usenet* [257].

2.1.4 Security Challenges for Distributed Hash Tables

Our main focus is the security of *structured* P2P Systems, and more specifically Distributed Hash Tables. However, some general attacks on P2P systems are relevant to DHTs as well. In principle, nodes in a P2P network must be regarded as not trustworthy and attacker nodes may drop, modify, or misroute messages. This opens the door to a broad range of attacks. Thus, in general it cannot be guaranteed that nodes in a DHT-network behave according to the DHT-protocol. Douceur has shown that in any distributed system without some form of a central entity for identity assertion, attackers are able to create virtual, fake identities [109]. Thus, decentralised authentication of participating nodes is a challenge for any P2P application. Further, due to the dynamic

nature of P2P systems (e.g. nodes can join and leave the network frequently) and the lack of central entities on routing paths, many existing security solutions are not (or at least not directly) applicable to P2P networks.

Sit and Morris were one of the first to analyse the security challenges for Distributed Hash Tables [256]. They classify attacks on DHTs into routing attacks, storage and retrieval attacks, and miscellaneous attacks. For routing, they consider "incorrect lookup routing", where an adversary node forwards DHT routing messages to a false or non-existing node. In addition, with "incorrect routing updates", adversary nodes *poison* the routing tables of other nodes by returning incorrect, malicious nodes when new nodes join or during DHT maintenance messages. Further, Sit and Morris recognise that partitioning attacks are possible, where a newly joining nodes gets fooled into joining a forged network. Regarding the correct storage and retrieval of data items in the DHT, Sit and Morris point out that in principle any node in the DHT can deny access to data items for keys it is responsible for. In addition, nodes can return falsified data items for keys they are responsible for. Other attacks they consider are targeted Denial-of-Service attacks on single DHT nodes (e.g. through message flooding) or parts of the DHT (e.g. through fast joining and leaving of adversary nodes, overloading DHT maintenance operations).

Wallach provides a study on P2P security challenges with a focus on DHTs [280]. Similarly to Sit and Morris [256], he mainly classifies the security challenges into routing and storage and discusses similar attacks as in [256].

Castro et al. [84] show that in order to achieve a secure DHT routing service, one must guarantee three properties: 1) *Secure node-ID assignment*, 2) *Protection against routing table poisoning*, and 3) *Secure message forwarding*.

Below we explain the previously mentioned and other attacks on DHTs known in the literature and discuss the corresponding research challenges. Further, we summarise the most important existing approaches to mitigate these well-known attacks. For an exhaustive survey of DHT security techniques presented in the literature, the reader is referred to [278].

Sybil attacks and secure node-ID assignment In the absence of a central authority in a P2P network, there is no authentication and nodes cannot trust each other. This enables identity spoofing, man-in-the-middle attacks, or denial-of-service attacks. The lack of a trusted authority enables a single adversary to control a large fraction of an overlay network with only a few external identities, a so-called *Sybil attack* [109]. Thus, one major DHT security challenge is to impede an attacker from gaining a large number of virtual node-IDs in the network. A related problem is secure bootstrapping: how can a node which joins the network be sure that the initial node it uses to join the overlay (the *bootstrap node*) is not controlled by an adversary?

To counter Sybil attacks several mechanisms for *secure node-ID assignment* have been proposed. Secure node-ID assignment has the goal to ensure "uniform random distribution of node-IDs that cannot be controlled by an attacker" [84]. In addition, secure node-ID assignment should prevent Sybil attacks by stopping an attacker from

creating an extremely large number of node-IDs in relation to the number of external IDs controlled by the attacker. Cuevas et al. present a centralised node-id assignment service that uses a low discrepancy sequence (LDS) in order to provide random node identifier assignment in a Chord DHT even under high churn [101]. However, their approach does not provide any protection against Sybil attacks. Castro et al. suggest the use of a public key infrastructure to ensure randomness of node-IDs and to prevent Sybil attacks [84]. Certificate authorities issue random node-IDs along with node-ID certificates. To prevent Sybil attacks, they propose to either charge for certificates or to bind certificates to physical identities.

A more decentralised approach for secure node-ID assignment is to use computational puzzles, as proposed e.g. by Baumgart [65] or Borisov [72]. Such an approach makes the generation of valid node-IDs computationally very expensive, hence potentially mitigating Sybil attacks. Danezis et al. present an approach that generates a *bootstrap graph* among nodes that joined a DHT to protect against Sybil attacks [106]. They propose a technique called *diversity routing* that tries to balance the number of times each node on the bootstrap graph of a particular query node is used during DHT routing. Thus, this approach tries to mitigate the effect of Sybil attacks by not using any node on the *bootstrap graph* disproportionally high compared to other nodes.

Awerbuch and Scheideler consider so-called *Join-Leave* attacks [55]. Assuming that the node-ID function $f_{node-mapping}()$ of a DHT assigns node-IDs randomly on the node identifier space I, attackers can still intensively populate a certain interval in I by repeatedly leaving and re-joining the DHT for every assigned node-ID which is not in the desired interval. They show analytically that in principle it is possible to design distributed algorithms that can make $f_{node-mapping}()$ of a DHT secure against Join-Leave attacks [55] and further present some concrete algorithms [54] [227]. Fiat et al. consider similar attacks they refer to as *Byzantine Join attacks* [114]. They present a concrete decentralised extension to Chord to prevent such attacks.

Eclipse attacks and routing table poisoning Even if an attacker controls only a certain fraction of nodes in the network, he can still aim at influencing DHT routing disproportionate to the number of nodes he controls. For instance, an attacker can try to populate other nodes' routing tables disproportionately high with links to nodes under his control. Singh et al. refer to such attacks as *Eclipse* attacks: "In an Eclipse attack, a modest number of malicious nodes conspire to fool correct nodes into adopting the malicious nodes as their peers, with the goal of dominating the neighbour sets of all correct nodes" [251]. Sometimes, such behaviour is also referred to as *routing table poisoning* [98] because essentially such attacks have the objective of maliciously influencing (hence *poison*) the selection of routing table entries by non-adversary nodes.

Castro et al. suggest the use of *Constrained Routing Tables* against routing table poisoning. They introduce an additional routing table $\tilde{T}_r$ to Pastry. In this *constrained* routing table, $\tilde{T}_r$, nodes are only accepted as entries if they are closest to a certain point in the node-ID space I. Thus, routing table poisoning for the constrained routing table $\tilde{T}_r$ is not possible with arbitrary node-IDs. Castro et al. provide modified algorithms for

join and routing table maintenance operations with constrained routing tables.

Singh et al. propose a technique where nodes try to choose neighbours that do not have significantly more DHT-links (inbound or outbound) to other nodes than the average node in order to mitigate Eclipse attacks [251]. They show analytically and through simulations that their solution called *anonymous auditing* can effectively protect against Eclipse attacks.

Condie et al. propose a technique they refer to as *induced churn* against routing table poisoning attacks [98]. Essentially, their technique combines a) periodic resetting of routing-tables, b) unpredictable node-ID assignment, and c) a rate limit on routing-table updates. They introduce a central entity called *timed randomness service*, which frequently provides the system with a verifiable, random identifer. This frequently changing nonce is included in the process of node-ID generation, i.e. as input to $f_{node-mapping}()$. Nodes are frequently required to obtain a new node-ID, hence the term *induced churn*. By making node-IDs random and at the same time frequently changing them, attackers cannot maintain a steady location in the node-ID space, I. This prevents attackers from increasing their presence in other nodes' routing tables over time (e.g. by using forged DHT maintenance messages). The proposed mechanisms have been evaluated on top of the Bamboo DHT [216], a modified version of Pastry. The results show that *induced churn* effectively prevents attackers from conducting coordinated routing table poisoning.

It is noteworthy that Eclipse attacks are not a concern in DHTs which impose tight constraints on node-IDs for routing tables entries, such as Chord or CAN [256] [84] [251]. These DHTs have constrained routing tables built-in within their design, and hence routing table poisoning attacks are not a concern for these DHTs.

Secure DHT routing and message forwarding A core problem for any DHT under attack is routing. In principle, an adversary node on the path from the query node to some key can drop the message, alter the message, or route the message to another adversary node. Secure routing techniques try to ensure that a message will reach the root for a key, even in the presence of adversary nodes.

Srivatsa and Liu [261] provide a theoretical investigation of DHT routing attacks. They present an analytical study as well as simulation results regarding the lookup success of DHT routing for CAN and Chord. Their study provides insight into the effect of independent routing paths, detection of invalid lookups, and result caching on protecting DHTs against routing attacks.

Castro et al. enhance Pastry with multipath recursive routing [84]. In addition, they introduce routing failure tests that can detect malicious nodes based on the density (in the node-ID space) of node-IDs within routing tables. Their simulation results show that their techniques can reach all replica roots for a given DHT key with a success rate of 0.999 in case of less than 30% attackers.

Baumgart and Mies suggest routing extensions to Kademlia that provide disjoint, and thus independent, lookup paths [65]. Their simulation results show that their technique

can achieve a lookup success rate of 99% in networks with up to 20% attackers.

Techniques for secure DHT routing in Chord have been proposed by Danezis et al. [106], Artigas et al. [49], and Marti et al. [167]. We will inspect these works and the methodologies used as evaluation closely in Chapter 4; we postpone a detailed description of the mechanisms proposed in [167] [106] [49] to Section 4.5.

Attacks on storage/retrieval and security of data items Another problem DHTs face is data integrity. Let us assume routing succeeds and the querying node receives some data item as a result to a lookup for a key. How can the query node assure that the data item is correct, i.e. that the data indeed belongs to the key and that it has not been altered since storage?

Data integrity can be achieved by cryptographically signing data, either in conjunction with a public key infrastructure or through self-certifying data. When data is signed, nodes can cryptographically verify the integrity of data they received from the DHT. With self-certifying data, integrity can be verified without relying on a trusted third party. This property can be achieved by statically binding data items to a private/public key pair, e.g. by representing data as the hash of a public key [50].

The general use of self-certifying data in DHTs has been suggested by Castro et al. [84] and Sit and Morris [256]. Dabek et al. present *CFS*, a distributed file system based on Chord which uses self-certifying path names [103]. A self-certifying URI-scheme for DHT-based multimedia communication session establishment has been proposed in [239] (we will present this scheme in detail in Chapter 5). Baumgart presents a scheme for secure DHT node-ID generation based on self-certifying node-IDs [63].

Other threats and challenges Other attacks on DHTs include *free-riding*, i.e. nodes using services without providing services to other nodes. Cuevas et al. investigate routing fairness in Chord and present enhancements to increase the fairness of routing load among Chord nodes [102].

A different security challenge for DHTs is lookup privacy, i.e. providing anonymous communications regarding the initiator and target key of lookups which nodes perform. McLachlan et al. present *Torsk* [173], a relay selection and directory service for the *Tor* anonymity network [108]. Their design includes a mechanism for an anonymous DHT lookup which is based on a randomly selected *buddy* node which performs the lookup on behalf of the original query node. Panchenko et al. propose algorithms for communication anonymity on top of iterative Chord routing [195]. By retrieving the entire routing table at each iterative routing hop, the query node does not disclose the lookup key to intermediate routing nodes, thereby hiding the target of lookups. Wang and Borisov present Octopus, an anonymous DHT lookup scheme [283]. Their approach is based on *Onion routing* [213] and additionally uses dummy queries to disguise the target key of lookups. Fessi et al. present an approach for privacy-preserving DHT-based real-time communication session establishment [112]. Their approach also uses *Onion*

Routing [213] to establish tunnels to protect nodes' and users' privacy. The entry point to such a *privacy tunnel* is stored as a data item in the DHT.

2.2 Session Establishment for Real-Time Communications

This section explains signalling for client-server based *Multimedia Communications* (also referred to as *Real-Time Communications*). In particular, we will examine the session establishment process for multimedia communication applications running over IP networks. First, a general introduction to IP-based multimedia communications and in particular VoIP is given (2.2.1), highlighting the differences between packet-switched signalling and circuit-switched signalling (the latter being used in the *PSTN*, the *Public Switched Telephone Network*). This introduction is followed by a more detailed description of how real-time communication and session establishment (the focus of our work) can be facilitated with the Session Initiation Protocol (SIP) [222] (2.2.2). Finally, we list threats for client/server-based multimedia communication systems and provide a survey on the state of the art regarding research and approaches for securing such systems (2.2.3).

Real-time requirements Multimedia communication applications—such as instant messaging (text), Voice-over-IP (audio), or Video-over-IP (video)—often have real-time requirements. Therefore, in the literature these types of applications are commonly subsumed under the term *Real-Time Communications* [93]. We refer to multimedia communication applications with certain timing constraints as *real-time communication* applications, as described in [44] as follows:

> *"Many ... services, such as voice, video and other applications, will have stringent real-time constraints and will demand not only high-bandwidths, but a predictable quality-of-service (QOS) ... Traditional communication network applications such as file transfer, electronic mail and remote login are examples of non-real-time applications for which the performance metrics of interest are typically average message packet delay and throughput. The characteristics of real-time communication applications differ significantly from those of non-real-time applications. As in real-time computing, the distinguishing feature of real-time communication is the fact that the value or utility of the communication depends upon the times at which messages are successfully delivered to the destination application. Typically, the desired delivery time for each message across the network is bounded by a specific maximum delay or latency, resulting in a deadline being associated with each message. ... this delay bound is an application-layer, end-to-end timing constraint. If a message arrives at the destination after its deadline has expired, its value to the*

end application may be greatly reduced. ... Without loss of generality, generally only the queueing delay is considered, as the packetization, switching and propagation delays are assumed known and fixed." [44]

Real-time communication applications are different from real-time computing even though the term *real-time* implies some similarities. Real-time communication refers to applications where instances need to communicate in real-time over a network. In short, real-time communication applications can be characterised by two properties: First, a maximum delay applies to communication messages sent across the network. Second, this maximum delay is an application-layer, end-to-end delay. Real-time communication applications can be further classified into *soft* or *hard* real-time communications: "Real-time communication applications are commonly classified as either soft or hard real-time. *Soft real-time applications* can tolerate some amount of lost messages, while *hard real-time applications* have zero loss tolerance."[44]. In this thesis we are only concerned with soft real-time communication applications, i.e. lost messages are to some extent tolerable and can be compensated for by resending messages. We therefore refer to soft real-time communication applications simply as real-time communication applications.

Voice-over-IP as typical application A prototypical example of a popular and widely deployed real-time communication application is *Voice-over-IP (VoIP)*. In this thesis, we use the term Voice-over-IP (VoIP) to comprise the transmission of digitised voice over an IP-based network in real-time. The audio signal may be digitised and encoded either before or concurrently with packetisation [122]. As other real-time communication applications, VoIP has only *soft* real-time requirements [44]. We will use VoIP as a prototypical real-time communication application in this thesis to illustrate key concepts, signalling and session establishment for multimedia communications, and the corresponding security threats. Further, we will frequently use VoIP and related terms such as *caller* and *callee* to exemplify our proposed solutions. However, it is important to notice that the signalling and in particular the session establishment process to set up multimedia communications is mostly independent of the actual communication medium (such as audio/voice). Therefore, the concepts we explain below in this section as well as the solutions we propose in the remainder of this thesis apply to any kind of session-based multimedia communication application, not only VoIP.

We refer to systems where user-clients need dedicated servers to determine the location of a multimedia communication partner as *client/server-based* real-time communication systems. In contrary, there exist research proposals to locate the callee of a multimedia communication with a P2P-network without any server involved in the process of determining the address of the callee. We refer to this approach as *P2P-SIP or P2PSIP* because such an approach solely affects the signalling part of real-time communications (or other session-based multimedia applications) and SIP is the signalling protocol used almost exclusively in research and standardisation efforts for this approach.

2.2.1 Introduction to IP-based Multimedia Communications

In recent years, the digitised transmission of audio signals over IP-based networks—commonly called *Voice-over-IP (VoIP)*—has emerged to a widely used application. VoIP is one example application for session-based communication with real-time constraints. In addition, instant messaging (*chatting*) and user-to-user video calls (or video conferences among multiple users) comprised of an audio and a parallel video stream are popular real-time communication applications. Below we focus our introduction somewhat on VoIP: The comparison of VoIP to traditional circuit-switched audio transmission makes a good case for illustrating some key characteristics of IP-based real-time communications (which also affect security). The general concepts described below regarding separation of signalling and payload and the corresponding protocols apply to any kind of multimedia communication.

For VoIP, *in real-time* usually is defined as the acceptable threshold for end-to-end delay of audio packets sent between participants of a call being $400ms$[7] [29]. However, this thesis is mostly not concerned with the exchange of audio packets but only with signalling and more specifically the session establishment part of signalling. For VoIP session establishment, *in real-time* refers to the timing constraint on the delay of a session establishment request reaching the callee of a call after the caller has sent it off. This delay is less stringent than the maximum end-to-end delay for audio packets. A call setup delay in the range of several seconds is usually considered acceptable (e.g. $8s$ [30]).

In contrary to traditional telephone networks in the PSTN, VoIP traffic—because it uses IP networks as the underlying technology—is transmitted not via circuit-switched but packet-switched networks. Thus, audio signals get transported in single packets which do not traverse a predefined route in the network between the sender and the receiver. Additionally, one of the fundamental characteristics of VoIP is the separation of signalling and media transfer. This means that the transmission of signalling messages (in order to set-up a session, negotiate parameters, or terminate a session) and the transmission of the payload (i.e. the actual transmission of digitised voice via IP-packets) is handled by different protocols and can take different routes in the network.

In general, multimedia communication implies that audio or video signals are digitised at some point in the network and then forwarded to the destination via IP-networks. However, neither the origin nor the destination must necessarily support digital audio/video signals or have an IP-interface. Instead, gateways on the path between sender and receiver can transform audio signals from digital to analog and vice-versa.

A signalling protocol is used to establish, negotiate, manage, and terminate a real-time communication session. The most widely used signalling protocols are the Session Initiation Protocol (SIP) standardised by the IETF [222] or H.323 [39] which is defined by the ITU. Both protocols offer similar functionality. While H.323 is still widely deployed

[7] ITU-T Recommendation G.114 recommends one-way transmission time limits for connections with adequately controlled echo as follows: 0 to 150 ms: acceptable for most user applications; 150 to 400 ms: acceptable for international connections; 400 ms: unacceptable for general network planning purposes. [29] [122]

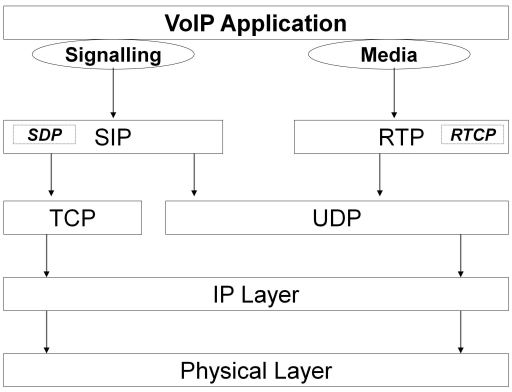

Figure 2.8: VoIP Layers and Protocols [111]

it seems that SIP is more and more preferred for new deployments and will dominate the market [203]. SIP has been selected as the core signalling protocol in the ETSI/TISPAN NGN [36] and in the *IP Multimedia Core Network Subsystem (IMS)* standardised by 3GPP [119] [31] [82]. For SIP, either TCP or UDP can be used as the transport protocol [222]. A detailed comparison of H.323 and SIP can be found in [122]. A detailed description of how VoIP signalling can be facilitated with SIP is given in 2.2.2.

Once a session has been set-up, a different protocol is used for transmitting the actual multimedia payload packets. Example protocols for transporting real-time multimedia data and providing QoS feedback are the *Real-Time Transport Protocol (RTP)* [232] or the *Real-Time Streaming Protocol (RTSP)* [234] which offers controlling the delivery of streaming media. Due to the real-time requirements of multimedia communications, UDP is the preferred transport protocol for the transmission of the payload.

Figure 2.8 shows the layering of several IETF protocols which are commonly used for VoIP [111] (the same protocol stack can also be used for other real-time communication applications than VoIP). While RTP solely uses UDP as a transport protocol, SIP messages can be transported via either UDP or TCP. SIP uses the *Session Description Protocol (SDP)* [125] to select the type of media and to negotiate a codec for media transmission. The *Real-Time Control Protocol (RTCP)* [232] is a control protocol for RTP sessions.

Differences between Voice-over-IP and the PSTN VoIP networks and the transmission of audio signals over these networks are characterised by some important differences if compared to the PSTN. In short, these differences are the following [203] [238] [240]:

- Due to the nature of packet-switched networks, with VoIP audio signals that belong to a single call can take different routes in the network and may traverse different (or even a disjoint set) of intermediate hops. Circuit-switched networks, as used in the PSTN, set up a route between participants of a conversation which is used for all audio signals that belong to a call.

- In the PSTN, signalling is done in a separate and closed network. With VoIP, signalling is done in an open, highly insecure network (e.g. the Internet).
- Traditional telephones are simple devices with limited functionality. VoIP terminals, on the other hand, are complex devices with their own TCP/IP stack.
- VoIP offers mobility: users can change their location and still use the same identity in the network. A VoIP-user only needs access to the Internet. By contrast, in the PSTN there is no mobility.
- Because there is no mobility in the PSTN, authentication is not necessary. Anybody who has physical access to a socket in the wall can use that line. As VoIP can be used from anywhere in the Internet, additional authentication must be utilised.

A more in-depth introduction to VoIP and the resulting research challenges can be found in [122].

2.2.2 Signalling with SIP

Much of the evolution of VoIP to a now widely used application is owed to the *Session Initiation Protocol (SIP)*. SIP is an IETF standard for signalling in multimedia communications over IP [222], i.e. a signalling protocol for real-time communications. Here we describe the basic elements of SIP and give examples of SIP operation to establish communication sessions. For a more detailed description of SIP and an introduction to advanced services with SIP the reader is referred to [255].

SIP Request	**Description**
INVITE	Initiates a call signalling sequence
ACK	Acknowledge
CANCEL	Used to cancel a request in progress
BYE	Terminates a session
OPTIONS	Queries an entity about its capabilities
REGISTER	Used to register location information at a registrar

Table 2.1: SIP Requests

SIP messages SIP is a client-server protocol similar to HTTP. Signalling in SIP is based on (ASCII compatible) text messages. A message is composed of a message header and an optional message body. Messages are either requests or responses. The SIP specification in RFC 3261 [222] specifies six types of requests: `INVITE` (initiates a call signalling sequence), `BYE` (terminates a session), `ACK` (acknowledge), `OPTIONS` (querying of capabilities), `CANCEL` (cancelling a request in progress), and `REGISTER` (used to register location information at a registrar). When a SIP entity receives a request, it performs the corresponding action and sends back a response to the originator of the request.

Responses are three-digit status codes (as in HTTP/1.1), categorised into six classes (e.g. `3xx` for redirect messages). Examples for response codes are `180-ringing`, `200-ok`, or `302-moved temporarily`. Table 2.1 lists SIP requests; Table 2.2 lists classes for SIP response codes.

Response Code	Class
1xx	informational
2xx	ok
3xx	redirection
4xx	client error
5xx	server error
6xx	global failure

Table 2.2: SIP Response Codes

Addressing Addressing in SIP is done with Uniform Resource Identifiers (URIs). A SIP-URI is similar to an e-mail address and generally of the type `sip:user@domain`. Users can register their location at a special SIP-entity (a so-called *SIP registrar*) by sending a message that binds their SIP-URI to their current location (e.g. IP-address and port). This binding gets stored by the registrar at a location server which can be used for location lookup by other SIP entities. For locating the location server of a domain, SIP relies on DNS [221]. The static SIP-URI of a user is called *address-of-record* (for example `bob@example.de`). The current location of the user is also stored as a URI, the so-called *contact address* (for example `bob@192.168.2.1:5060`). Generalising this concept, we will denote the SIP *address-of-record* as the SIP-URI and the *contact address* as the user location.

Listing 2.1: Example of a SIP INVITE Message [135]

```
INVITE sip:7170@iptel.org SIP/2.0
Via: SIP/2.0/UDP 195.37.77.100:5040;rport
Max-Forwards: 10
From: "jiri" <sip:jiri@iptel.org>;tagi=76ff7a07-c091-4192-84a0-d56e91fe104f
To: <sip:jiri@bat.iptel.org>
Call-ID: d10815e0-bf17-4afa-8412-d9130a793d96@213.20.128.35
CSeqi: 2 INVITE
Contact: <sip:213.20.128.35:9315>
User-Agent: Windows RTC/1.0
Proxy-Authorization: Digest username="jiri", realm="iptel.org",
 algorithm="MD5", uri="sip:jiri@bat.iptel.org",
 nonce="3cef753900000001771328f5ae1b8b7f0d742da1feb5753c",
 response="53fe98db10e1074
 b03b3e06438bda70f"
Content-Type: application/sdp
Content-Length: 451
v=0
o=jku2 0 0 IN IP4 213.20.128.35
s=session
c=IN IP4 213.20.128.35
b=CT:1000
t=0 0
m=audio 54742 RTPi/AVP 97 111 112 6 0 8 4 5 3 101
a=rtpmap:97 red/8000
a=rtpmap:111 SIREN/16000
a=fmtp:111 bitrate=16000
a=rtpmap:112 G7221/16000
a=fmtp:112 bitrate=24000
a=rtpmap:6 DVI4/16000
a=rtpmap:0 PCMU/8000
a=rtpmap:4 G723/8000
a=rtpmap: 3 GSMi/8000
a=rtpmap:101 telephone-event/8000
a=fmtp:101 0-16
```

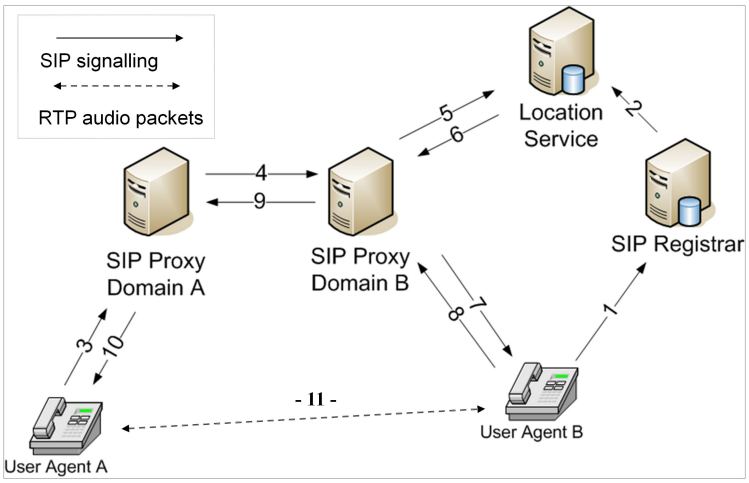

Figure 2.9: Schematic View of SIP Session Establishment and Locating a Callee [203]

Example 2.6: *Listing 2.1 shows an example of a SIP* `Invite` *message [135]. The message contains several headers and a body with several SDP parameters. The* `From` *and* `To` *headers state the SIP-URI of the caller and callee, respectively. The tag in the* `From` *header is chosen by the caller's user agent to uniquely identify the request. Similarly, the* `call-id` *uniquely identifies the session.*

SIP entities SIP designates different (logical) entities: *user agent*, *proxy*, *registrar*, *redirect server*, and *location server*. A User agent is a terminal (hardware or software) participating in SIP-communications. A user agent which sends out a SIP-request is referred to as a *user agent client (UAC)* while a user agent which responds to a request is referred to as a *user agent server (UAS)*. A proxy receives messages and forwards them to another SIP entity. A redirect server redirects the sender of the message to another SIP entity instead of forwarding the message. Users can register their current location (i.e. IP-address and port) with the registrar of their domain. This enables mobility: A location server is used by a registrar to store the location of users (the binding of a SIP-URI with a current IP-address). The location server provides a directory for other SIP entities to look up the current location for a given SIP-URI.

Example 2.7: *Figure 2.9 [203] shows how a callee of a VoIP call can be located with SIP in a simplified schematic view. In this example, user agent A and B are in different domains and have different proxies. First, the callee (user agent B) needs to register with its local registrar (1) to be able to receive calls. The registrar stores the location information at a location server (2). When user agent A wants to call user agent B, it sends an INVITE-request to its local SIP-proxy (3) which passes on the request (possibly after a DNS lookup) to the proxy of user B's domain (4). The proxy in domain B needs to look up the IP-address of user agent B at the location server (5, 6) before it can send*

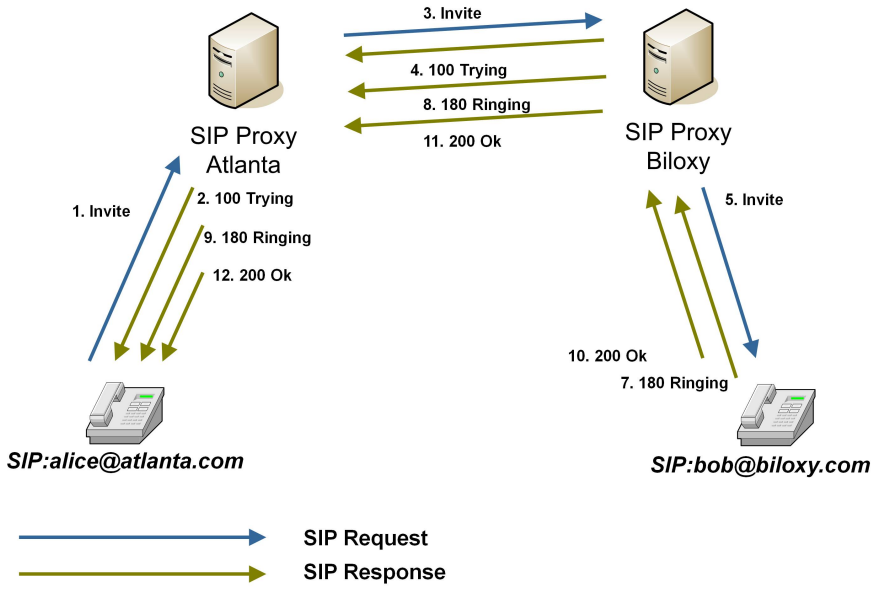

Figure 2.10: SIP Messages Exchanged During Session Establishment

the request to user agent B (7). In this example, the response message for user agent A takes the same route back (8, 9, 10), possibly for billing purposes. Note that all the servers are necessary to facilitate mobility of the participating users. Once a session has been established, the transmission of audio packets via RTP is started (11).

Example 2.8: *The set up and cancellation of a voice connection between two users is illustrated in Figure 2.10. It shows the SIP messages (requests and responses) that are being exchanged if a user with SIP-URI* `alice@atlanta.com` *wants to initiate a session with a user with SIP-URI* `bob@biloxi.com`*. In this example, two proxies are involved. The user agent of* `alice@atlanta.com` *sends an* `INVITE` *request to the proxy of the* `atlanta.com` *domain (1). The proxy sends back a* `100-trying` *response message (2) and forwards the* `INVITE` *request to the proxy of the* `biloxy.com` *domain (3). This proxy responds with a* `100-trying` *response message to the* `atlanta.com` *proxy (4) and forwards the* `INVITE` *request to the user agent for* `bob@biloxi.com` *(5). The user agent* `bob@biloxi.com` *responds to the* `INVITE` *request first with a* `180-ringing` *message (7), and eventually with a* `200-ok` *response after a user has picked up the phone (10). Both of these messages are forwarded to user agent for* `alice@atlanta.com` *by the intermediate proxies (8,9,11,12). Having received the* `200-ok`*, the user agent for* `alice@atlanta.com` *can now request the start of the media transfer by sending an* `ACK` *request directly to* `bob@biloxi.com` *(not shown in the picture). Note that a proxy is not needed for this: After a session has been established, user agents can communicate directly. At the end of the conversation, either one of the user agents terminates the session by sending a* `BYE` *request to its counterpart (not shown in the picture).*

SIP Security Mechanisms The SIP standard as specified in RFC 3261 [222] includes several security mechanisms:

- *S/MIME*: Because SIP is using MIME for message bodies, S/MIME [208] can be used to send authenticated and encrypted messages between user agents.[8]
- *Digest Authentication*: SIP entities sharing a secret (e.g. a password) can authenticate each other with a challenge-response mechanism. To prevent replay attacks, this challenge-response authentication includes nonces.
- *TLS or IPSec*: Hop-by-hop security for SIP signalling can be achieved either on the transport layer (TLS) [107] or on the network layer (IPsec) [149].

2.2.3 Security Challenges for Multimedia Communication Signalling

We will focus on VoIP as an application to illustrate attacks and countermeasures for IP-based multimedia communications. However, as noted previously, the signalling of multimedia communications is largely independent of the communication medium. Therefore, most attacks and countermeasures we discuss in this section also apply to other real-time communication applications.

Most security problems that VoIP faces arise from the significant differences to the PSTN (see 2.2.1). This is especially true in the consumer market where phone calls are carried out over the Internet. From the perspective of mobility and authentication, VoIP is similar to mobile phone networks such as GSM. However, GSM differs from VoIP because it uses smartcards in terminals and consists of a limited number of providers that trust each other. Hence, VoIP has its specific security challenges.

Below we provide a non-exhaustive list of threats for multimedia comunication systems (2.2.3.1). Our focus is on signalling and more specifically on client/server SIP. For a more exhaustive list of threats to VoIP and SIP the reader is referred to [42] [152] [237] [290]. Further, we provide an overview of the corresponding research challenges and ongoing work in the area of SIP security (2.2.3.2).

2.2.3.1 Threats for Client/Server-based Real-Time Communication Systems

SIP is an application layer protocol that uses underlying IP networks for setting up multimedia communications. Thus, all threats that are well-known in IP-networks (e.g. denial-of-service, spoofing, sniffing, and many more) are inherited. Furthermore, implementation vulnerabilities (e.g. buffer overflows) are likely because SIP servers and terminals are complex IP-devices [203].

[8] Even though envisioned by the authors of the SIP specification, S/MIME in conjunction with SIP is hardly deployed.

It is well known that the SIP protocol, without any dedicated security mechanisms, can be attacked in several ways. The SIP specification acknowledges five possible attacks (cf. [222], pp. 232-236 and [203]):

- *Registration hijacking*: Registration authorization in SIP is implemented by the registrar, which checks the `FROM` header field in the registration message to determine if the sender is allowed to modify registrations for the identity in the `TO` header. As the `FROM` header field in a SIP-message can easily be altered by any user agent, malicious registrations are possible if no additional mechanisms are used by the registrar to authenticate the request.
- *Impersonating a SIP server*: Attackers can impersonate any SIP server and forge answers. For instance, any redirect server can impersonate another SIP-server by forging answers to requests from a user agent to this other server, thereby redirecting future calls to itself, or to any other SIP server.
- *Tampering with message bodies*: Any intermediate SIP entity (e.g. proxy server) on the route from one user agent to another user agent can read messages and also modify headers as well as message bodies.
- *Tearing down sessions*: An attacker capable of sniffing and sending packets on an IP-network can also tear down SIP-sessions by sending forged `BYE` or `CANCEL` messages to either of the participating parties. This can be done by forging the `FROM` header field, thus claiming the message is coming from one of the participants of the session.
- *Denial of service*: The availability of any SIP-server or user agent can be threatened by an exorbitant number of requests coming from an attacker, eventually exhausting the capacity of the targeted server/client to answer requests.

In addition, among others, the following threats arise on the SIP layer [203] [237]:

- *Call hijacking*: Having seen the contents of an `INVITE` request, an eavesdropper can inject a spoofed `200-ok` or `302-moved temporarily` response, thereby redirecting calls to another user agent or server. Potentially, this can result in a hijacked call [237].
- *Client impersonation*: A malicious attack on a client could install a Trojan horse which impersonates the client. Such an impersonated client can be used by the attacker to send out (authenticated) calls or registrations from the client [203].
- *Eavesdropping*: If an attacker can sniff packets on the network, he has also access to the content included in the header and body of a SIP-packet, unless encryption is used [203].
- *Spam*: Spam-messages (also called *SPIT* for *Spam-over-IP-Telephony*) are possible, where an attacker sends out automatically generated packets to user agents [203] [220].

- *Billing fraud*: Several attacks exist where an attacker tries to charge phone calls to some victim [292]. For instance, an attacker can modify and replay an intercepted `INVITE` request, in order to bill a call to a victim user agent, thus avoiding to pay for the call [237].
- *Traffic logging and analysis*: Any SIP-server (and any network node) on the route between user agents can log the messages it receives during a session. Some users might not want their connections to be traceable on the entire route from sender to receiver. However, such logging might in part be necessary for billing purposes [203].

2.2.3.2 Research Challenges and Approaches to SIP Security

Authentication and identity assertion One of the fundamental problems for SIP security is end-to-end authentication of communication partners. SIP messages sent via TLS may pass many application layer hops between sender and receiver, and some intermediary entities may not be trustworthy. In addition, certificates bind identities to key pairs—enabling authentication—but certificate verification does not necessarily assure trustworthiness of the authenticated principals: For instance, a certificate will usually not provide any assurance about the security policy or the protection against malicious intrusion the authenticated principal has in place. In other words, mutual authentication on the entire route between user agents does not guarantee a standard of security functionality at proxies and redirect servers. Another example is spam (see below): even authenticated parties might be the source for spam messages [203].

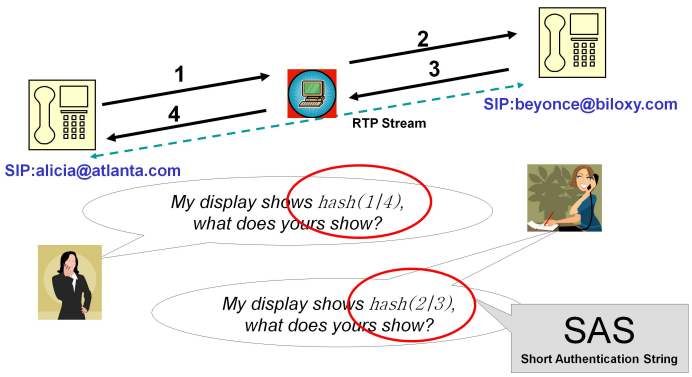

Figure 2.11: End-to-End VoIP Authentication with ZRTP [241]

The SIP community has realised that hop-to-hop security as offered by e.g. TLS is insufficient for authentication in many cases. RFC 3325 [141] specifies a SIP header in which a proxy of a domain can assert the identity used in a SIP message. However, this assertion is not signed. Thus, "The use of these extensions is only applicable inside an

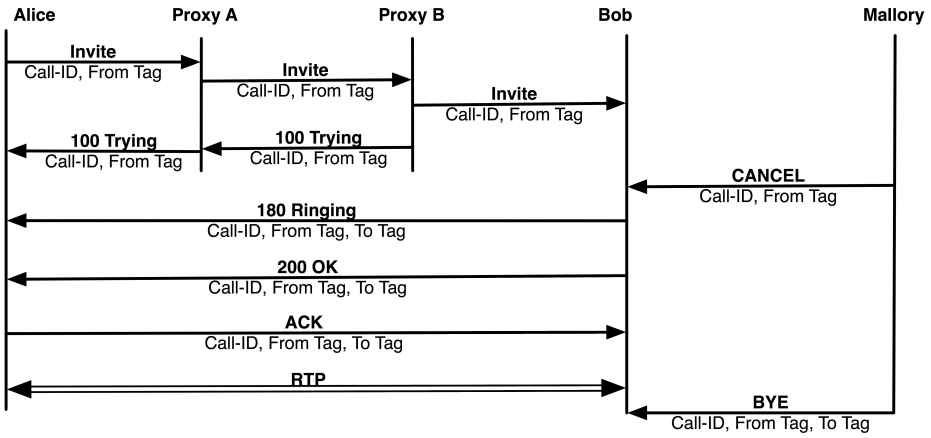

Figure 2.12: Single-message Denial-of-Service attacks on SIP Devices [242]

administrative domain with previously agreed-upon policies for generation, transport and usage of such information" [141]. RFC 4474 [201] provides two additional SIP headers (`Identity`, `Identity-Info`). These headers allow a SIP proxy to assert proper authentication of an identity by its domain, by signing such an assertion cryptographically (so-called *identity assertion*).

A different approach to end-to-end VoIP authentication is *ZRTP* [297]. ZRTP is not a solution on the signalling layer; it enables to securely authenticate communication endpoints *after* a session has been established, e.g. with SIP. ZRTP enables a Diffie-Hellman key exchange over an RTP media stream. However, the key exchange is protected against man-in-the-middle attacks through a short authentication string (called *SAS*). Technically, the SAS is part of the hash of the parameters exchanged in the Diffie-Hellman key exchange. This short authentication string is displayed to the users which can compare it (thereby detecting attacks) by reading it over the RTP stream. Thus, an attacker would need to forge the voice of users in real-time in order to still launch an undetected man-in-the-middle attack on the key exchange. Figure 2.11 shows an example of a key exchange with ZRTP where two users compare the SAS (which is displayed in their VoIP client) over a voice media stream. If the SAS is the same on both sides, the key exchange was successful and secure. In this way, ZRTP offers authentication of a communication partner (and key exchange) without using any certificate infrastructure [241].

Platform security of terminals and servers Because SIP devices are complex, implementation weaknesses seem unavoidable. Vulnerabilities for SIP implementations are found frequently [33]. RFC 4475 [206] describes various test messages that can be used to *torture* a SIP implementation. Many simple tools exist to carry out tests on SIP implementations [19]. More advanced tools offer the construction of sophisticated test-cases for SIP [24] [270] [286] [40].

As an example of the many works on testing SIP implementations, in [242] SIP phones have been tested regarding their proper authentication of `CANCEL` and `BYE` messages. Without such message authentication, an attacker may terminate a session while it is

being establised or may end an existing session. To distinguish between different sessions, SIP uses so-called *dialog identifiers*. A SIP *dialog ID* is composed of the `Call-ID`, the `From-tag` (contained in the From-header) and the `To-tag` (contained in the To-header) [242]. Figure 2.12 [242] shows the establishment of a SIP session including the dialog-ID components sent between the entities. The `Call-ID` and the `From-tag` are set by the caller and the `To-tag` by the callee. These tags remain in every SIP message throughout the dialog, and if a message does not match an existing dialog it should be discarded [243]. Figure 2.12 shows how an attacker *Mallory* could carry out *Cancel* and *Bye* attacks. However, she would have to know several components of the corresponding dialog in order to succeed. In [242] and [243] it has been shown that many SIP implementations do not properly authenticate `CANCEL` and `BYE` messages; these implementations can thus be attacked without having previously sniffed the dialog-ID of an ongoing session.

Spam over Internet Telephony (SPIT) There are several reasons why *Spam over Internet Telephony (SPIT)* can be considered a possible problem for VoIP. Automatic generation of SIP-based phone calls is feasible and cheap. In addition, VoIP Spam will be much more intrusive than e-mail Spam is today: A phone will actually ring with each SPIT occurrence (possibly in the middle of the night). VoIP deals with real-time audio signals. Thus, many countermeasures which have been successful in fighting e-mail spam may not work for SPIT.

A comparison to e-mail spam and an overview on possible solutions against SPIT in SIP networks can be found in [220]. Researchers have proposed innovative algorithms against SPIT [58] [205]. Also, SIP extensions for feedback on SPIT detection and prevention have been suggested [187] [188] [277]. A prototype for an anti-SPIT solution has been described in [228] [245]. In [204], the authors describe a holistic solution against SPIT.

Lawful Interception Most countries legally allow for authorized wiretapping of telephone calls by law enforcement agencies, so-called Lawful Interception. Depending on the use case and national law, Lawful Interception legislation may apply to VoIP. However, Lawful Interception for VoIP is much harder than in the PSTN for several reasons. First, the SIP provider and the Internet Service Provider (ISP) may be different. Second, signalling and payload usually take a different route; traffic is only linked in terminals. And third, the signalling and payload of the conversation may be encrypted.

Several scientists have realised the potential problems of Lawful Interception for VoIP. They have made a proposal arguing that the benefit of Lawful Interception for VoIP may be outweighed by the negative consequences for society [66]. An analysis of Lawful Interception for VoIP and potential technical solutions can be found in [209].

Other challenges Users would like to have the *privacy* of their communications preserved. However, some headers of SIP messages must contain information about the sender's identity (e.g. contact header). Thus, any SIP-server can log information about

connection requests and connection termination. Although users might accept this on some proxies (e.g. for billing purposes), it can be done by any SIP-server on the route of a SIP-request [203]. A potential solution for providing privacy of SIP messages is a *pseudonymity service* [200]. Such a service replaces sensitive headers before passing messages on; it acts as a transparent user agent in both directions [203].

Emergency calls are another technical challenge for VoIP [233]. Most importantly, a VoIP identity (e.g. a SIP-URI) cannot be statically allocated to a physical location. The IETF *ECRIT (Emergency Context Resolution with Internet Technologies)* working group is working on technical solutions for using Internet technologies to determine the geographical location of a user and to conduct reliable call routing in case of emergencies [127] [271].

2.3 Decentralised Service Location

2.3.1 Service Location as Part of Communication Session Establishment

In computer science, a *service* is characterised by a certain functionality it offers. The syntax of a service is specified by its interfaces. Further, a service usually has a rather informal associated semantic, e.g. a natural language description of what functionality the service is supposed to provide. Fischer and Hofer define a service as a "collection of functionality which is offered via well-defined interfaces" [116]. Similarly, the W3C *Web Services Glossary* defines a service as follows: "A service is an abstract resource that represents a capability of performing tasks that form a coherent functionality from the point of view of providers entities and requesters entities." [23].

Tanenbaum and van Steen note that "in distributed systems, services are generally specified through interfaces" which "specify precisely the names of the functions that are available together with types of the parameters, return values, ..." [273]. Further, they note that the semantics are often specified informally "by means of natural language" [273]. According to this distributed systems perspective, OASIS defines a service as "A mechanism to enable access to one or more capabilities, where the access is provided using a prescribed interface and is exercised consistent with constraints and policies as specified by the service description." [160]. Our work concerns distributed systems (in particular the Internet). In this thesis, the term *service* is therefore seen in the context of distributed systems. For the course of our work we define a service as follows:

> **Definition 4 (Service):** *A Service in a distributed system is a collection of functionality with an associated semantic which is offered via well-defined interfaces.*

In data communications, a *session* describes several messages that are exchanged in an end-to-end connection between communicating entities [153]. For instance, in SIP a

session is defined as the exchange of data between an association of participants [222]. We refer to a *session* as an active, temporary connection among two or more entities in a distributed system in which communication data is exchanged among these entities. To accesses a service in a distributed system, a session has to be established, so that a service consumer can access a given service via its interface. *Session establishment* thus refers to the process which enables participating entities (i.e. service consumers and service providers) to establish a session. For real-time communications (see 2.2), sessions are being established for exchanging content of one communication medium (such as audio, video, or text) or several such media in parallel. In the case a session is being set up for real-time communications, we refer to *multimedia session establishment* or *communication session establishment*.

Service Location To establish a (communication) session in a distributed system, the *location* of the destination on the network needs to be determined by the initiator of the session. In computer networks, *addresses* are used to specify a certain location on the network [262]. Computers use addressing so that a source of data communications can "indicate the identity of the intended destination" [262]. In practise, on the Internet, the combination of an IP address and a port number is used for addressing. Humans cannot easily remember long numerical addresses. Name resolution can resolve human-readable names to addresses: Given a string-like identifier, it returns the corresponding address. When the address being resolved is the location of a service, we refer to the name resolution process as service location. We can define service location as part of communication session establishment:

> **Definition 5 (Service Location):** *Service Location is the sub-process of the session establishment process which is responsible for determining the location of a given service. It takes as input an identifier for a known service and works on an Internet scale. It returns the location for the desired service.*

This sub-process of session establishment is sometimes also called *user location* [222] because for some applications (e.g. Voice-over-IP) the user's mobility requires the frequent necessity to determine the current end system to be used for communication. However, from a technical perspective also for these applications it is the *service* of accepting and answering to a session establishment request which is offered by the callee of a Voice-over-IP phone call (this fact is also reflected by the SIP notion of a user agent *server* as the SIP entity that receives communication requests from a user agent *client*).

Service location assumes that the initator of the session knows about the service and has a corresponding identifier (e.g. a URL/URI). This distinguishes service *location* from service *discovery* (see related concepts below). *Internet scale* means that service location is not restricted to a single network or only works within an administrative domain.

For the process of service location as part of establishing a real-time communication session, the real-time constraints might not be as stringent as for the actual data transfer within an established session. For instance, for VoIP the maximum delay to *establish* a session is considered to be $8s$ [30], while the maximum delay for an RTP packet to arrive

within a session is $400ms$ [29]. Nevertheless, for most applications there is a maximum delay (e.g. in the order of seconds) on session establishment and hence also on service location. Thus service location has —in many cases—real-time requirements (i.e. "If a message arrives at the destination after its deadline has expired, its value to the end application may be greatly reduced." [44]).

The Domain Name System (DNS) A typical example of a system for service location is the *Domain Name System (DNS)* [179] [180]. Essentially, DNS constitutes a "directory lookup service that provides a mapping between the name of a host on the Internet and its numerical address" [262]. Many Internet applications use DNS to determine the location of service endpoints. For instance, SIP uses DNS to locate the SIP server in the domain of the callee [221].

DNS is an application-layer client-server protocol [153]. A hierarchical name-space is used, based on the concept of *domains*. A domain describes a group of hosts that are under the same administrative control [262]. Domains are organized in a hierarchical tree. For instance, `amazon.com` is a subdomain of the `.com` domain. *Name servers* are used to resolve hostnames to locations. So-called DNS *root servers* are responsible for the top-level domains. Other name servers are responsible for a certain sub-domain. The name server which stores the actual hostname-to-location binding for a given hostname is called the *authoritative* name server for that hostname.

Mappings are stored in so-called *Resource Records (RRs)*, e.g. the mapping between a hostname (e.g. a URL) and its current location (e.g. its IP-address). Depending on the type of a Resource Record, it stores the actual location for a hostname (`Type=A`), or a mapping to an authoritative name server for a domain (`Type=NS`)[9]. The latter type is used to traverse the domain hierarchy towards the authoritative name server for a given domain.

To resolve a hostname, typically a query to a local name server is sent. If this name server is not authoritative, the query is forwarded according to the domain hierarchy until it reaches the authoritative name server. Name servers use caching extensively to be able to respond to queries directly without the need to forward them. For a more detailed description of how the Domain Name System works the reader is referred to [262] [153].

DNS is a distributed system: the hostname-to-location bindings are distributed among name servers all around the world. However, the domain hierarchy implies that certain nameservers are more important to its overall function than others. In fact, there have been attacks on the DNS root servers [37], exemplifying that these root servers form a single point of failure within the DNS. Thus, the DNS —as used today on the Internet—is not a fully decentralised system. From the perspective of decentralisation, it is merely a distributed database; request routing follows the client-server paradigm with central entities playing a major role.

[9]Other RR types exist; we omit the description of these for brevity.

Related concept: service discovery A related—but different—concept to service location is *service discovery*. Service discovery targets the automatic detection and discovery of not precisely defined services. Moreover, service discovery is—not necessarily per definition, but de-facto—limited to local area networks because underlying techniques and assumptions are not supported or valid on an Internet scale. *Service location*, on the other hand, assumes that a given identifier (e.g. a URI/URL) is used to determine the location (e.g. network address) of a particular service. In contrary to existing approaches for service discovery (which in practise only work in local area networks, due e.g. to the fact that they are based on IP-multicast), service location has the objective to function on an Internet-scale.

Several protocols for service discovery in local area networks exist. The *Service Location Protocol (SLP)* [124] enables the *discovery* of network services in local area networks (LANs): "Traditionally, users have had to find services by knowing the name of a network host (a human readable text string) which is an alias for a network address. SLP eliminates the need for a user to know the name of a network host supporting a service. Rather, the user supplies the desired type of service and a set of attributes which describe the service. Based on that description, the Service Location Protocol resolves the network address of the service for the user." [124]. Different to our definition of *service location*, SLP hence aims at *discovering* loosely-defined services that are being announced in the network, instead of locating a particular service based on an accurate identifier. Further, SLP targets local networks and does not work on an Internet scale ("SLP is intended to function within networks under cooperative administrative control. Such networks permit a policy to be implemented regarding security, multicast routing and organization of services and clients into groups which are not be feasible on the scale of the Internet as a whole."[124]).

Bonjour is a so-called *zeroconf* protocol that enables to discover services on a local area network without explicit identification of such services by the user (so-called *Service Instance Enumeration* or *network browsing* [91]). It is technically based on so-called *multicast DNS* [92] and *DNS based service discovery* (*DNS-SD*) [91]. Multicast DNS enables name resolution without a DNS server by having requests being sent to a multicast address. The corresponding host answers with its IP-address. A similar approach is *Link-Local Multicast Name Resolution (LLMNR)* [41]. DNS based service discovery (DNS-SD) uses DNS records (e.g. multicast DNS records in the case of Bonjour) to announce services. A client (looking for instances of a given type of service) provides the *type of service* it is looking for, as well as a domain; in return, the client receives "a list of named instances of that desired service" [91]. Hence, a Bonjour client (using DNS-SD) queries for a type of service using multicast DNS. Any host on the local network that provides a service of the given type answers to the multicast DNS request, so that eventually the client will have received a "list of named instances" [91]. Bonjour offers thus functionality comparable to SLP [124]: Querying for a certain service type, the client receives a list of all services with this service type that are running in the local network. The *Simple Service Discovery Protocol (SSDP)* [120] is a solution very similar to *Bonjour* which uses IP-multicast but http (instead of DNS) to announce services in a local network.

2.3.2 Locating a Service on the Internet without Servers

Researchers have made proposals for fully decentralised service location. These proposal are almost exclusively based on Distributed Hash Tables[10]; the DHT is used to locate services offered by entities in the network. Compared to other DHT applications, service location has some unique aspects and requirements (see also [93]): Most importantly, the guaranteed and timely retrieval of the data items stored in the DHT, i.e. the location for a service, is important. In addition, the DHT stores more sensitive information, i.e. location-bindings, than e.g. in filesharing applications. Hence, service location has unique security requirements.

Several reasons make DHTs the suitable choice for decentralised service location. First, DHTs naturally offer a distributed, fully decentralised database which is necessary for storing identifier-location bindings. Second, unlike unstructured P2P networks, DHTs can guarantee that a data item stored in the network will be found (at least in the absence of attackers). Third, also unlike unstructured P2P networks, DHT can offer upper bounds on the number of hops a lookup will take. The fact that DHTs cannot only guarantee reliable storage and retrieval of data items, but that in addition also retrieval within a certain amount of routing hops is ensured, is important for service location. It enables to guarantee to find the location of a service within a certain amount of time, fulfilling the real-time requirements of service location. In summary, only structured, DHT-based P2P networks are fulfilling the requirements for service location.

As an example, several proposals exist for a fully decentralised replacement of the Domain Name System (DNS), where the resolution of hostnames is facilitated by a DHT-based P2P-network [99] [207] [196]. As another example, locating real-time communication users with a DHT is investigated by researchers [252] and currently being standardised in the IETF [79]. We refer to these kinds of P2P-applications as *Decentralised Service Location* because these applications use the P2P-network solely for locating services: Once the node offering the desired service has been located (e.g. a web-server in the case of decentralised DNS or a VoIP terminal in the case of decentralised VoIP), communication takes place directly between the initiator of the DHT lookup request and the desired host offering the service. In other words, the DHT is only used for service location. Being based on a DHT makes service location *decentralised*, as there are no servers or centralised entities involved in locating a service.

We will examine DHT-based real-time communication session establishment as a typical example in order to investigate the security challenges of decentralised service location. Most approaches for DHT-based real-time communication session establishment combine a DHT-underlay for service location with SIP for overall session establishment. These approaches are commonly subsumed as *P2PSIP*. We consider these approaches a prototypical example of decentralised service location: As highlighted previously, the DHT is only used for the service location part of session establishment. In 2.3.3, we describe in detail how such an approach works technically. In Section 3.1 we will examine the

[10]One notable exception is the unstructured P2P approach for SIP proposed in [90] which focusses on mobile scenarios.

corresponding security challenges which are the main motivation for our work.

2.3.3 P2PSIP as a Prototypical Example of Decentralised Service Location

Several researchers have proposed the use of a Distributed Hash Table for SIP registration and user location as an alternative to SIP servers. The first publication which proposed such an approach to SIP-signalling was by Singh and Schulzrinne [252], followed by Bryan et al. [78]. These publications contain proposals to use a Distributed Hash Table—namely Chord—to support mobility in SIP communications. The general approach of using a P2P network for locating users in SIP is commonly referred to as *P2PSIP* (or *P2P-SIP*). With P2PSIP, all central components used for locating the callee in a session establishment attempt in SIP (e.g. proxy server, registrar, location server) are replaced with a structured P2P network and SIP user registrations are stored distributedly in a DHT. Technically, P2PSIP enables the establishment of any kind of real-time user-to-user multimedia session (e.g. voice, video, instant messaging).

2.3.3.1 General Architecture

To use a DHT for locating SIP identities, users register their location (i.e. the IP-address through which they are reachable and intend to receive calls) not at a central server. Instead, the locations for SIP-URIs are stored distributedly in a Distributed Hash Table. The SIP-URI is the DHT key and any node can compute the key-ID for a particular SIP-URI by computing the hash-value for the SIP-URI. Any node can request the current location (e.g. IP-address and port) for a SIP-URI by inserting a key lookup request into the network. The network routes this lookup request to the node responsible for the key. This node delivers the content to the requesting node.

Example 2.9: *Figure 2.13 [241] shows an example of a P2P-SIP network using a DHT for locating the callee of a multimedia communication. In the example a Chord [266] network is displayed. In the simplified example network eight nodes are in the network, including two users Alice (with node-ID* 33*) and Bob (with node-ID* 220*). Further, some nodes store data items for keys they are responsible for (e.g. node* 170 *stores data for key-ID* 137*). In the example, Bob has stored his SIP user registration (i.e. the binding between his current IP-address and his SIP-URI) in the network. Assuming the hash of Bob's SIP-URI is* 210*, node* 215 *is responsible for the key-ID of Bob's SIP-URI and stores the corresponding location. When Alice tries to call Bob, she sends a lookup request for key-ID* 210 *to the closest node to the key-ID she has in her local routing table, node* 170 *(1). This node forwards the request in the same manner*[11] *(2). Finally, the request reaches the node responsible for key-ID* 210*, node* 215*, and it gets routed back to the query node*

[11] This example uses *recursive* routing where a lookup request gets forwarded until it reaches the node responsible for the key-ID.

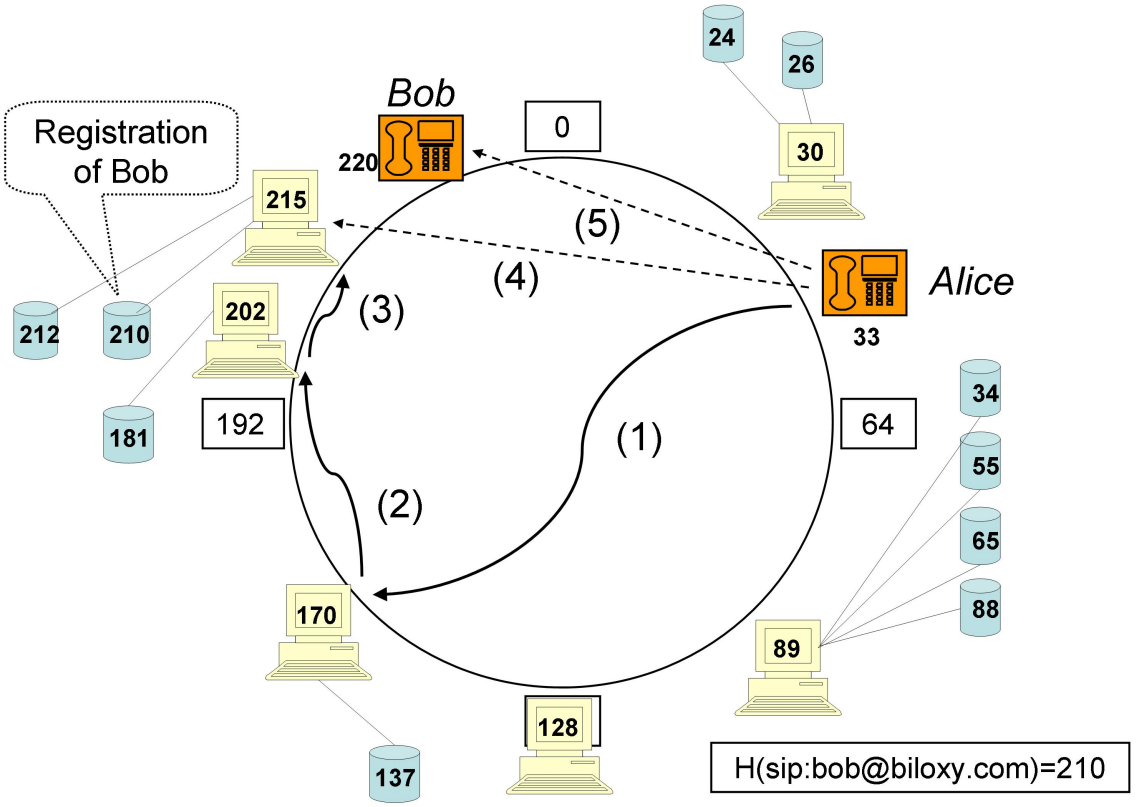

Figure 2.13: Locating the Callee of a VoIP Session with P2PSIP [241]

(not displayed in the figure). At this point, the query node can contact node 215 *directly—without using the overlay network (depicted with a dotted line in the figure)—in order to receive the corresponding content, i.e. Bob's current location (4). Subsequently, Alice contacts Bob directly peer-to-peer to establish a SIP session with regular SIP signalling (5).*

Note that after the callee has been located, normal SIP signalling as in client/server SIP takes place. Only the service location part is different in P2PSIP compared to client/server SIP.

2.3.3.2 State of the Art

At the time of writing quite some work on P2PSIP as well as several prototype implementations for P2PSIP exist. Additionally, the IETF has formed a P2PSIP working group which has the goal of defining an IETF protocol for P2P-based SIP signalling [79]. Below we provide a short summary of the history and current developments regarding research and standardisation of P2PSIP systems.

Research approaches The first works on P2PSIP [252] [78] proposed SIP [222] as the protocol for DHT operations. With such an approach—sometimes called *P2P-over-SIP* [254]—DHT routing primitives are expressed as SIP messages (e.g. INVITE, REGISTER). This approach, however, was later rejected because SIP messages were regarded to incur too much overhead for DHT routing [145]. Later works follow a design where any DHT protocol can be used to locate SIP identities—also called *SIP-using-P2P* [254] [76]. The research community (as well as the standardisation community) has accepted this approach where the process of finding a location for a SIP-URI (i.e. the service location process) does not use the SIP protocol itself.

More recent research focusses on the application of P2PSIP in specific scenarios, e.g. mobile networks [170] [151]. Several proposals argue for a hierarchy of multiple DHTs for P2PSIP to enable interconnection among different P2P networks [249] [169]. In [113] an approach for combining client-server SIP with P2PSIP is presented.

Another area of research has been on security and privacy concerns for P2PSIP [235]. Fessi et al. present a solution to preserve location privacy and user-interaction privacy for P2PSIP [112]. Other studies consider availability [248] or secure node-ID generation [63] for P2PSIP. Security is the main focus of our work, and we will examine related, existing security approaches for P2PSIP during the course of our work.

Several prototype implementations of P2PSIP exist. One of the first proof-of-concept implementations of P2PSIP was *SIPDHT* [18], an open-source software using Chord or CAN as the underlying DHT. In [64], a P2PSIP prototype implementation has been connected to an emulation framework. In [265], P2PSIP has been implemented on top of a flexible transport overlay system. *OpenVoIP* [61] is an open-source implementation of P2PSIP based on the p2pp protocol [60]. An experimental system of OpenVoIP is running with around 1000 peers on approximately 300 planetlab [14] nodes [59]. The OpenVoIP prototype has implemented Kademlia, Chord, and Bamboo [216] as underlying DHTs.

Standardisation efforts The IETF has formed a P2PSIP working group for standardising a P2PSIP protocol. The charter of the working group defines its goals as follows:

> "The Peer-to-Peer (P2P) Session Initiation Protocol working group (P2PSIP WG) is chartered to develop protocols and mechanisms for the use of the Session Initiation Protocol (SIP) in settings where the service of establishing and managing sessions is principally handled by a collection of intelligent endpoints, rather than centralized servers as in SIP as currently deployed. A number of cases where such an architecture is desirable have been documented ... P2PSIP peers manifest a distributed name space in which overlay users are identified and provides mechanisms for locating users or resources within the P2PSIP overlay ..." [79]

Several originally separate and different proposals (e.g. [60] [143] [166]) for a P2PSIP protocol have been merged into a single approach called *REsource LOcation And Discovery (RELOAD)* [140]. The P2PSIP WG has agreed on RELOAD as its core *peer* protocol

[79]. RELOAD is a generic protocol which can be used by various applications. Therefore, the specification of RELOAD has been split into a generic baseline document [140] and a document specifying how SIP can be used in conjunction with RELOAD [139].

RELOAD has chosen Chord [266] as its underlying DHT and is essentially specifying a protocol for the exchange of Chord messages among peers. User identities are authenticated at an *enrolment server* which serves as a certificate authority. The role of the enrolment server is to secure node-ID assignment. Further, the integrity of location-bindings stored in the DHT is protected by certificates issued by the enrolment server. RELOAD requires an implementation of *Interactive Connectivity Establishment (ICE)* [218] for NAT traversal.

Other ongoing work in the P2PSIP WG concerns alternative DHT routing modes [299] [300], discovery of specific services which certain nodes may offer [162], and dynamic adaptation of nodes to changes in operating conditions [163]. In addition, the working group has agreed to define a specification for the collection of diagnostic information (e.g. for network monitoring or fault detection) [258].

Our work is orthogonal to the ongoing standardisation of P2PSIP because we do not focus on protocol details. Further, we study solely the service location part of session establishment. Thus, our research targets a generic P2PSIP architecture (as depicted in figure 2.13) where the process of locating a communication partner (steps 1-3 in figure 2.13) is not using SIP as the message protocol and can be investigated separatedly from the following negotiation of session parameters (steps 4-5 in figure 2.13, likely using SIP).

2.3.3.3 P2PSIP vs. Skype

An existing P2P-based multimedia application with a large user community is *Skype* [6]. Skype offers service location: Given a unique identifier (a user name), Skype establishes a communication session with the current location of that identifier. Multimedia sessions (audio, video) among Skype nodes take place directly peer-to-peer (with the exception of relaying in case of NAT/Firewall traversal [150]). However, Skype uses a proprietary protocol and closed source code [69]. The details of how service location is executed exactly are therefore hard to determine. Skype uses a central login server; in addition, several nodes play a more important role in the network than others (so-called *super nodes*) [62]. Because of these conditions it is unknown and hard to infer how the service location process works in detail and to what extent it is decentralised.

Skype's security model is based on two principles:

- *Central login server*: Nodes authenticate themselves at the Skype login server; it stores user names and passwords [62].
- *Security-by-obscurity paradigm*: Skype follows a security-by-obscurity scheme regarding the communication protocol and peer software [62]: Communication between peers is encrypted and the peer software is only available in binary code.

Even further, obfuscated code makes reverse engineering of the Skype algorithms hard [69].

Under these constraints, researchers have analysed Skpye through (mostly passive) measurements in order to derive a view into Skype's architecture and interior functioning [62] [133] [89] [94]. Also, researchers have tried to understand the role of super nodes in Skype and their relaying of traffic [150] [123] [288]. Given that Skype traffic is encrypted, several works present algorithms for merely detecting that traffic is related to Skype [269] [70] [71].

Reverse engineering of Skype's software has been performed in detail by Biondi and Desclaux [69]. This study revealed that Skype's code uses obfuscation techniques to make reverse engineering of the software difficult. Further, the reverse engineering analysis showed that Skype uses RC4 [17] for encryption of messages. Unfortunately, however, the results of such studies are quickly outdated as Skype updates its code frequently. It is therefore unknown, to what extent the results obtained in [69] still hold for the latest version of Skype in use today.

From a security research perspective, Skype is not suitable for investigating secure distributed algorithms for communication and data storage in P2P-based multimedia communication systems: Its closed source code and security-by-obscurity paradigm forbid a thorough analysis of its interior algorithms. Moreover, Skype's obfuscated binary code and encrypted messaging make it infeasible for researchers to conduct experiments with novel algorithms. In particular, the service location process applied within Skype is a secret, and it is therefore almost impossible for researchers to investigate novel alternatives or to conduct practical, reproducible experiments with new algorithms.

In contrary to Skype, P2PSIP resembles an *open-specification* paradigm. All research proposals for P2PSIP openly describe the chosen architecture and routing algorithms. The corresponding IETF standardisation work targets an open specification of P2PSIP, and all intermediate designs are publicly available [13]. Hence, P2PSIP enables the research and experimental application of new algorithms. Most importantly for our work, the service location process is decentralised and well-defined. We therefore choose P2PSIP as a suitable prototypical example of decentralised service location for our study of the security issues and the investigation of corresponding decentralised solutions.

Chapter 3

Thesis Scope

Contents

3.1 Security Analysis of P2PSIP

The initiators of P2PSIP claim higher robustness (against failure) as well as easier configuration and maintenance as the main motivation for P2PSIP. To analyse the advantages and business model of P2PSIP is outside the scope of our work. We consider P2PSIP as a decentralised alternative to client-server SIP and solely look at the security issues.

Obviously, a peer-to-peer setting imposes new security threats to SIP communications. For instance, the lack of a central authority makes authentication of peers a hard problem. Our goal is to look at known results on security of P2P networks and identify if and to what extent these results hold for P2PSIP. We will distinguish between threats on the DHT Routing layer (i.e. security issues regarding the functionality offered by the DHT) and threats on the DHT application layer (i.e. security issues regarding multimedia session establishment offered by SIP if run over a P2P network)[1].

[1]Parts of this chapter (including figures) have been previously published in [235]. See also Appendix A.

3.1.1 Security on the DHT Routing Layer

Castro et al. identify three requirements for secure DHT routing [84]: secure node-ID assignment, secure routing table maintenance, and secure message forwarding. We examine attacks on these requirements in the context of P2PSIP, looking first at node-ID assignment and then on overlay routing (i.e. routing table maintenance and message forwarding).

3.1.1.1 Attacks on Node-ID Assignment

Douceur showed in [109] that without a trusted agency which certifies identities, adversary nodes can control a large fraction of an overlay network. Much of the security of structured overlay networks is based on the assumption that joining nodes are assigned IDs at random. To analyse node-ID generation in P2P-SIP[2], we take the model from [227]:

> "At any time t we have n honest nodes and $e \times n$ adversary nodes in the Chord ring ($e < 1$). A malicious entity owning multiple nodes can control the Chord ring by launching join-leave attacks: The attacker joins and leaves the network with its nodes until a sequence of $O(log\ n)$ adversary nodes is reached. The goal is to find a joining-strategy that prevents an adversary from gaining such a sequence. In that case, it can be guaranteed that any message reaches its destination (though at a high performance cost) by simply routing each message to $O(log\ n)$ nodes directly following in the ring."[227]

This model focuses on availability of the network; the goal is to guarantee that a message will reach the responsible node. It also gives a lower bound for a routing strategy that will provide availability (if the joining strategy of the overlay prevents an attacker from gaining sequences) by using multiple lookup paths.

With IPv6 vast numbers of IP-addresses can be available to an attacker to form such join-leave attacks on a P2PSIP Chord network. But even with IPv4, join-leave attacks on node-ID generation are possible when an attacker gets assigned IP-addresses dynamically (e.g. by its ISP). Some early proposals for P2PSIP (e.g. [75]) suggested to generate node-IDs by hashing `[IP-address:Port]` of the participating node. Clearly, such an approach is not desirable from the perspective of secure node-ID assignment and join-leave attacks: An attacker can try ports at will until the hash of `[IP-address:Port]` matches the desired value. In general, note that an attacker does not have to actually join and leave the network: Because node-ID assignment in Chord is deterministic by hashing the IP-address, attackers can compute node-IDs in advance.

Scheideler suggests using a rotation joining strategy to perturb nodes frequently [227], resulting in random node-ID assignment and making join-leave attacks hard for an attacker. Another way to prevent join-leave attacks would be to make the join-operation

[2]We exemplify our analysis with Chord. Nonetheless, the revealed attacks are also possible on other Distributed Hash Tables and our general results apply for other Distributed Hash Tables as well.

in some way costly for the joining node. For example, computational puzzles or micro-payment systems can make it costly for an attacker to repeatedly join the network (or to pre-compute node-ID values). However, these strategies cannot be used for P2P-SIP unless nodes can be forced to change their IP-address or node-ID generation will be based on something different than the IP-address (for example by adding a timed randomness service [98]).

Any node wanting to join the overlay needs to know at least one node that already participates in the overlay. Possible options for such *bootstrapping* are: static (persistent) nodes, cached nodes, or broadcast mechanisms (e.g. SIP-multicast). In any case, if the initial bootstrap node is malicious, the joining node can easily be attacked. Without a pre-established trust relationship with the bootstrap node, secure bootstrapping is an open question.

Without a central authority, another open problem for P2PSIP is identity enforcement. The overlay network has to prevent duplicate IDs. This applies to node-IDs (i.e. what happens if two IP-addresses hash to the same value) as well as to key-IDs (two SIP-URIs hashing to the same value).

3.1.1.2 Attacks on Overlay Routing

Any malicious node within the overlay can drop, alter, or wrongly forward a message it receives instead of routing it according to the overlay protocol. This can result in severe degradation of the overlay's availability. But further: Confidentiality and integrity cannot be provided for P2PSIP registration and location lookup messages by the overlay. In P2PSIP this means that an attacker cannot only prevent access to location information, but also forge responses to SIP-messages.

Attackers can also use DHT maintenance operations in order to infiltrate routing table entries with links to attacker nodes. In [251] such attacks and possible countermeasures are studied in the context of Pastry [223] as the DHT. Chord uses *constrained* routing tables (i.e. it is precisely defined which nodes can occupy certain routing table entries). Therefore, routing table infiltration attacks are not a considerable threat in Chord [251].

Srivatsa and Liu observe in [261] that Chord nodes can check the results of routing queries as follows: The node looking up a key in the network exploits the fact that with any routing hop it should get closer to the Node-ID responsible for the key. Thus, detecting certain invalid lookups is possible in P2PSIP networks if iterative routing is used: A requesting node simply hashes the IP-address it receives as a response for the next routing hop from the DHT. It then checks if that hash value is closer to the key-ID than the node-ID it received on the previous hop. However, an attacker can still perform an undetected man-in-the-middle attack if he can place itself close enough to the desired target key (using attacks on node-ID generation described previously). In this case it is likely that the attacker is the last node before the node responsible for the key in the Chord ring.

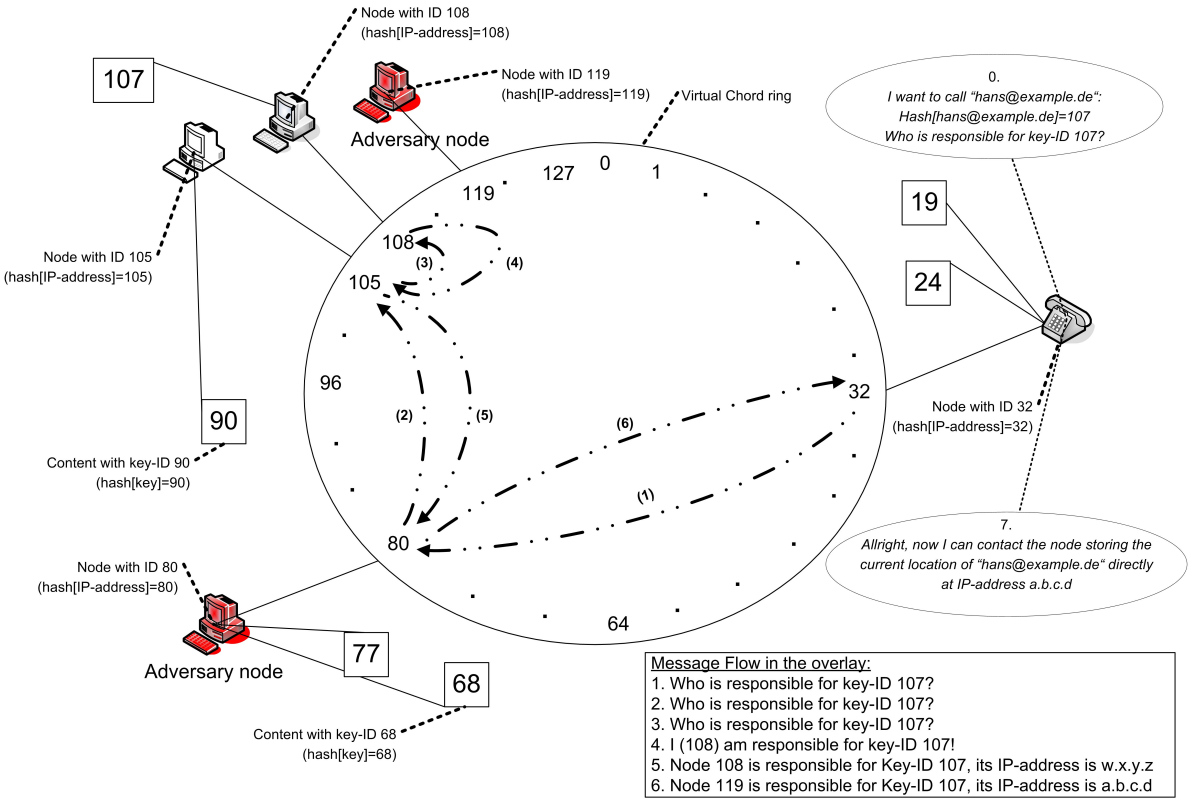

Figure 3.1: Man-in-the-middle Attack on P2PSIP

Example 3.1: *Figure 3.1 shows an example of a man-in-the-middle attack on P2PSIP. The nodes with ID* 80 *and ID* 119 *conspire to form an attack on the content of key-ID 107. The (honest) node responsible for the key is node* 108[3]*. Because node* 119 *could also be responsible for the key, the initiator of the key lookup (node* 32*) cannot detect the attack. Note that messages (2) to (5) are not necessary to form the attack. However, obtaining the real location for key-ID* 107 *can be useful information for the adversary nodes* 80 *and* 119 *(perhaps to perform further man-in-the-middle attacks on the media stream). In the example recursive routing is used. It can be observed that with iterative routing node* 80 *would not be able to perform the attack (node* 32 *would get suspicious if node* 80 *redirected it directly to node* 119 *because it assumes there exists some node with an ID lower than* 107[4]*). To perform a man-in-the-middle attack with iterative routing the adversary has to place itself close to the targeted key-ID (e.g. between node* 105 *and* 108 *in our example).*

In any P2P system there is a risk of so-called *free-riding*: Nodes use services but fail to provide services to the network. Certainly, in a P2PSIP setting there is a risk of free-riding: nodes use the overlay itself for registration and location service but drop messages from other nodes. A huge amount of nodes that *ride free* would eventually result in reduction of the overlay's availability.

[3]For simplicity, in this example we denote node x as the node with Chord-ID x.

[4]Remember that in Chord a node can only route to a node with higher ID than the key-ID if this node is the first in its routing table; otherwise it must route to the node with highest ID below the key-ID.

3.1.2 Security on the DHT Application Layer

Some security issues for client-server SIP are presumably even harder to solve for P2PSIP. Below we examine P2PSIP security considerations on the SIP-layer.

Integrity of data items stored in the DHT One of the main security problems of using SIP in a peer-to-peer setting is the authentication of content stored in the DHT: How can a node which retrieves SIP registrations, i.e. the binding of SIP-URI and user location, verify the integrity in the absence of a central authority? Without a central authority, authentication of location-bindings stored in the P2P network is a non-trivial task. However, without authentication of the content stored in the network the service offered by such a network is of little use: Nodes cannot verify that messages they receive from the overlay have not been altered on some (overlay-) hop by an adversary node.

Anonymity With P2PSIP, any node in the overlay can keep track of location lookups that it routes[5]. Moreover, any P2PSIP node can keep a profile of registrations and lookups for SIP-URIs the node is responsible for. Furthermore, using join-leave attacks, an adversary can place itself intentionally at any location in the Chord ring to log access to a particular SIP-URI. In SIP, anonymity for SIP-messages can be achieved by using anonymous values in SIP-headers. However, certain headers must contain information about the sender/path of a message to successfully route response-messages back to the sender; the same requirement holds for P2PSIP.

Reliable and secure emergency services It is desirable to specify a dedicated identifier for emergency services (comparable to e.g. 911 in the US or 110 in Germany used for such purpose in the PSTN). For client-server SIP, "successful delivery of an emergency service call ... requires both an association of the physical location of the originator with an appropriate emergency service center and call routing to deliver the call to the center" [157]. Thus, in the context of P2PSIP, two challenges arise: a) reliable signalling for emergency calls in an overlay network, b) ascertaining the physical location of users in real-time. Given the distributed nature of P2PSIP, these problems seem harder to solve than for client-server SIP and possibly cannot be solved with existing mechanisms.

Lawful Interception *Lawful Interception (LI)* is the process of legally authorised wiretapping of communications carried out by law enforcement organisations. Regular SIP signalling can use a different path in the network than the RTP payload. Further, there is no predefined route for RTP traffic, making Lawful Interception of such traffic in real-time

[5]Note that such attacks are also possible in Skype: In a Skype network any node with a public IP-address that is not behind a middlebox (e.g. Nat/Firewall) eventually becomes a super node. Super nodes in Skype are used for routing of content for Skype users who are located behind a middlebox. Though Skype traffic is encrypted, logging of traffic data (who called whom for how long) is possible [150].

technically challenging. With P2PSIP, there is not even a predefined route for signalling traffic (SIP), due to the highly dynamic nature of P2P-networks. Thus, any Lawful Interception solution needs to dynamically determine an appropriate location in the network for intercepting SIP messages and then to correlate SIP signalling with interception of RTP traffic. Further, P2PSIP allows for a fully decentralised system. Thus, there may not be trustworthy legal entity as reference point for Lawful Interception requests. In summary, P2PSIP implies several new challenges for Lawful Interception.

Spam prevention Spam-over-Internet-Telephony (SPIT) is not a serious problem in VoIP networks yet. However, several characteristics possibly make VoIP highly attractive for marketers to use this technology for unsolicited calls [204]. With P2PSIP, Spam-prevention can only be implemented at the receiving user agent: Due to the highly dynamic nature of an overlay, the node responsible for a key and the routing path vary over time and may therefore not be relied on for spam-filtering. To be decentralised, such Spam-prevention mechanisms residing in terminals must not rely on centralised components.

3.2 Existing Work and Remaining Challenges

We have identified several security challenges for P2PSIP. We will now assess to what extent existing mechanisms can be used to mitigate the aforementioned security problems or to what extent research work has specifically addressed these issues. The remaining *research gap*, i.e. the challenges which have not been sufficiently addressed by researchers, are the motivation for our work and confine our goals.

Our overall goal is the design and evaluation of mechanisms which not only mitigate P2PSIP security issues, but which constitute *decentralised* security solutions. We will therefore pay particular attention to what extent existing solutions are decentralised. If solutions for certain problems exist, yet these solutions rely largely on centralised components, we regard such solutions insufficient and still consider a research gap.

Secure DHT routing There exists quite some work on secure DHT routing (see 2.1.4). In particular, several solutions for secure node-ID assignment have been proposed, some of which are decentralised [98]. In fact, for Chord (the DHT we mostly consider in our work), several decentralised solutions for secure node-ID assignment have been proposed [227] [114]. Therefore, we regard the problem of secure, decentralised node-ID assignment sufficiently addressed by researchers and in principle solved. We will not investigate alternative solutions, but assume this property for our work.

Chord has very constrained routing tables. Therefore, routing table poisoning—so-called *Eclipse attacks* [251]—are not a serious threat in Chord [251]. We consider Chord sufficiently protected against this threat. For other DHTs, partially decentralised countermeasures against routing table poisoning have been proposed [251] [98]. We regard

existing countermeasures sufficient and assume secure routing table maintenance, or at least that routing table poisoning attacks have a negligible effect on DHT routing.

The properties of secure node-ID assignment and secure routing table maintenance assure that attackers cannot control a disproportionately high amount of nodes in the network or fractions of routing tables, respectively. However, attackers nodes can still severely attack the availability of the DHT routing layer, e.g. by dropping messages or routing messages wrongly. Such attacks on message forwarding and on the availability of the DHT routing layer have insufficiently been addressed for Chord. As we will see (in detail in Chapter 4), existing work is one the one hand flawed, and one the other hand cannot effectively guarantee DHT lookup success in the presence of adversary nodes that mis-route messages. Hence, there is a need for algorithms which can achieve high lookup availability despite attacks on message forwarding. We consider the design and analysis of such algorithms—in particular the design of decentralised solutions—a core challenge for our work.

Considering DHT routing challenges in summary, we regard the challenges of secure node-ID assignment and secure routing table maintenance sufficiently addressed by previous work, in particular for Chord. The major remaining DHT routing challenge is to guarantee message forwarding, i.e. lookup availability, in the presence of attackers.

Integrity of data items stored in the DHT An obvious solution to protect the integrity of location-bindings stored in the DHT is to rely on a *Public Key Infrastructure (PKI)*: A centralised certificate authority (CA) issues certificates which nodes can use to sign location-bindings. Indeed, researchers have proposed such a solution for P2PSIP [76]. However, such an approach can hardly be regarded as decentralised. All nodes in the system would need to agree on a single root-CA as being trustworthy.

The semantics of integrity in the context of SIP registrations stored in a P2PSIP network need to be defined. Further, there exists no design of a completely decentralised mechanism specifically suited to protect the integrity of P2PSIP location-bindings. We hence consider addressing these challenges part of our work.

Anonymity Several mechanisms can in principle provide pseudonymity with respect to using the location service provided by P2PSIP. If recursive DHT routing is used, an option to ensure anonymity within the overlay would be to replace the source of messages at every hop in the overlay: This disguises the originator of the request. A similar approach would be to use a *friend-to-friend* model where nodes only exchange messages with other nodes they consider trustworthy and reliable. A message can be delivered anonymously between two nodes that do not trust each other: If a transitive trust path exists, the message can be sent indirectly using hops between *friends* that trust each other. These *friends* hide each other's origin before passing on messages in the network.

As with client-server SIP, privacy could also be achieved by using a pseudonymity-service which acts as a *back-to-back user agent (B2BUA)* [203]. Such a B2BUA would

be placed as a node in the overlay and would provide services to user agents within or outside of the overlay. It replaces SIP-headers in both directions in order to disguise message origin.

Fessi et al. present an approach for a privacy-preserving P2PSIP signalling protocol called *Pr²-P2PSIP* [112]. Their approach relies on a central login server to provide secure node-ID assignment and authentication of SIP-URIs. Essentially, *Pr²-P2PSIP* uses *Onion Routing* [108] to establish tunnels to protect nodes' and users' privacy. The entry point to such a *privacy tunnel* is stored as data item in the DHT.

In summary, several approaches exist for protecting the privacy of SIP location-lookups in P2PSIP. *Pr²-P2PSIP* [112] is the most advanced of these approaches and specifically targeted at P2PSIP. Although *Pr²-P2PSIP* is not completely decentralised[6], we regard the issue of privacy for P2PSIP as sufficiently addressed by previous work. We will thus not investigate the design of novel solutions to provide privacy for P2PSIP in our work.

Emergency services The technical challenge of mapping the physical location of a device to a responsible emergency service is considered in the IETF *Emergency Context Resolution with Internet Technologies (ecrit)* working group [157]. The *Location-to-Service Translation protocol (LoST)* [127] defines mechanisms to translate the combination of a service identifier (e.g. Uniform Resource Names (URNs) for emergency services [231]) and location information (e.g. civic location information [274]) to a corresponding SIP-URI which is to be used for emergency services. These mechanisms can be used to determine the target SIP-URI of the appropriate emergency service for a given location.

The problem of mapping a location to an emergency service SIP-URI is similar in P2PSIP to client-server SIP. LoST [127] is in principle applicable in a P2PSIP setting. However, the LoST protocol uses servers to resolve queries. Though interesting research, we regard the design of a decentralised, P2P-based version of LoST (or similar approaches) of doubtful usefulness: the critical importance of emergency services demands a highly dependable and highly secure service. It is questionable if a fully decentralised, P2P-based service can fulfil these stringent requirements.

The second challenge for P2PSIP emergency services is reliable signalling for emergency calls. This challenge technically boils down to message forwarding and availability of the lookup service provided by the DHT. We specifically address this challenge in the context of secure DHT routing (see above). Our solutions for DHT lookup availability in the presence of attackers are applicable in the context of emergency services and can enable reliable signalling in a P2PSIP network.

[6]The authentication server in *Pr²-P2PSIP* serves two purposes: a) secure node-ID assignment, and b) authentication of user identities. As highlighted previously, decentralised approaches for secure node-ID assignment exist. We will investigate decentralised authentication for SIP-URIs in our work. Thus, in principle *Pr²-P2PSIP* could be modified towards a decentralised solution based on existing approaches and our work regarding decentralised integrity protection of location-bindings.

Lawful Interception There are significant technical challenges for conducting Lawful Interception in a P2PSIP network. Further, the problem is more complex than in client-server SIP systems. Thus, existing approaches for Lawful Interception of multimedia communications traffic (targeting client-server SIP) are most likely not applicable. No publicly available work has addressed the technical challenges of Lawful Interception in a P2PSIP system. We therefore regard a technical analysis of Lawful Interception in P2PSIP as well as studying potential solutions an important challenge to be addressed in our work.

Spam prevention P2PSIP Spam-prevention must reside in terminals and not rely on central entities. These constraints imply that most existing mechanisms for SPIT prevention are not applicable to P2PSIP because they are positioned at SIP servers. In addition, those mechanisms which can potentially run on terminals [245] often rely on SIP servers to function properly. It is thus a challenge to develop decentralised mechanisms for SPIT detection which solely run at terminals. We consider the development of such solutions an important goal of our work.

Chapter 4

Algorithms for Increased Lookup Availability

Contents

In this chapter we analyse how malicious nodes can attack the availability of a DHT lookup and investigate how to prevent such attacks with decentralised mechanisms[1]. Focusing on Chord [267] as the DHT, we show analytically that a small fraction of adversary nodes can severely attack the availability of the DHT's lookup service (Section 4.2). Consequently, our goal is to derive novel decentralised solutions, i.e. solutions which rely on no or only a few centralised components, and to demonstrate their effectiveness.

We present mechanisms to mitigate the negative effects of malicious nodes on lookup availability (Section 4.3) and assess the proposed algorithms analytically and through simulations (Section 4.4). Further, we compare our approach with existing solutions and discuss related work (Section 4.5). A thorough analysis of existing approaches for lookup availability in Chord reveals that these approaches are insufficient to address the problem. Further, we reveal several methodological flaws in related publications of existing approaches. This chapter concludes with a summary of results and the main contributions (Section 4.6). Throughout this chapter we use the notation and definitions introduced in Chapter 2.

4.1 Rationale

4.1.1 Defining Lookup Availability

A DHT lookup can consist of many routing attempts from the query node to the key. Thus, a lookup can use several different paths and is finished if either it succeeds, a threshold t_h (limiting the number of hops used in the lookup) is reached, or all possible paths between the query node and the node responsible for storing the corresponding data item (i.e. $root_k$) have been tried without success. We define a path in a DHT as follows:

> **Definition 6 (Path):** *A* path $p(n_q, k) \subseteq \mathcal{N}$ *from a query node* $n_q \in \mathcal{N}$ *for key* $k \in K$ *is any set of nodes such that routing from* n_q *for key* k *will pass through these nodes including* $root_k$*. Two different paths are called* alternate *if at least one node (other than* n_q *and* $root_k$*) is on both these paths and* independent *if they share no common node other than* n_q *and* $root_k$*.*

If at least one adversary node is on a path, the root node for the desired key might not be reached via this path. Thus, adversary nodes may degrade the availability of the key lookup service which a DHT essentially provides. We define the availability of a DHT

[1]Parts of this chapter (including figures) have originally been published in [246] and [247]. See also Appendix A.

lookup service as follows:

> **Definition 7** (**Lookup Availability**): *The* Availability of the Lookup Service *(or* Lookup Availability *for short) is the probability that the corresponding, unmodified data item is returned by the DHT after a node has invoked a lookup for an arbitrary, existing key.*

Note that our definition of lookup availability explicitly contains a notion regarding the integrity of data items: only *unmodified* data items are considered. From the perspective of the query node, only the retrieval of a *correct* data item constitutes a successful lookup. Looking solely at the DHT lookup service, a data item can be considered correct if it has not been altered since (or during) storage. Any kind of *data origin authentication* [121] to determine if the data is also coming from the *correct* source must be performed by higher application layers. For lookup availability, we hence consider a data item returned by the DHT as correct if it has not been modified (not even by the originator itself) since the originator has stored it (or during the storage process itself).

Since a DHT does not offer a `modify(key)` primitive, the only way for an originator to alter previously stored data is to store a new version of the data item for the key. In practise, data items can contain timestamps (or expiration dates) to express which is the latest, correct data item (e.g. to overwrite an existing data item). Thus, in a strict (low-level) sense, even the *owner* of a data item cannot modify it directly once it has been stored in the DHT. We abstract from such details (e.g. timestamps); without loss of generality, we assume that a query node always retrieves the latest data item for a key and that there is no need for an originator to modify already stored data items. Therefore, any unmodified data item can be considered correct, and it suffices to consider only completely unmodified data items as correct.

One important implication of considering only unmodified data items in our definition of lookup availability is that a query node must have a way to verify that a data item it retrieved has not been modified, in order to determine if a lookup was successful or not. In this chapter, we assume that query nodes have this ability. We provide a concrete solution for nodes to verify the integrity of data items in Chapter 5. Also, we will define data integrity in the context of a DHT lookup more precisely in Chapter 5.

Note further that Definition 7 already implies a metric for lookup availability: the degree to which arbitrary lookups are successful. The *success-rate* of a random lookup is hence the metric we use for analysing (and measuring) lookup availability in a DHT. Consequently, we define the success-rate ρ as the probability that an arbitrary lookup will succeed in a given DHT:

> **Definition 8** (**Lookup Success-Rate**): *The* success-rate ρ *of a DHT lookup service is the probability that in a given DHT a lookup for an arbitrary, existing key will return the corresponding, unmodified data item.*

Formally, we can define ρ as follows:

$$\rho = P\left(\exists p(n_q, k) | \forall n_i \in p(n_q, k) : n_i \; is \; good\right) \tag{4.1}$$

where n_q is a random query node and k is a random key.

Further, we denote the mean hop count for a series of lookups with χ, defined as follows:

> **Definition 9** (**Mean Hop Count**): *The* mean hop count*, χ, is the arithmetic mean of the number of hops each lookup needed to succeed in a series of lookups.*

We can derive χ as follows:

$$\chi = \frac{\sum_{i=1}^{l} c(i)}{l} \tag{4.2}$$

where l is the number of lookups and $c(i)$ is the hop-count for each individual lookup $i = 1 \ldots l$. If a lookup did not succeed, the hop-count of this lookup is either determined by trying all possible paths paths between query node and root node or by a hop threshold, t_h, which limits the number of total hops for each lookup.

4.1.2 Choosing Chord as the Prototypical DHT

For our investigation, we choose Chord [267] as the prototypical DHT. There are several reasons why Chord is an adequate choice for our investigation of security for decentralised service location:

- We use VoIP as the prototypical application for studying the security of service location[2]. Chord has been chosen by the IETF P2PSIP working group [79] for standardising a P2PSIP protocol [140]. Thus, it is likely that most P2P-based VoIP implementations will be using Chord as the underlying DHT protocol. Indeed, several current P2PSIP prototype implementations are based on Chord [161] [294]. Therefore, choosing Chord brings more relevance to our analysis and security improvements because our results will be directly applicable to the main industry standard for P2P-VoIP.

- Among all DHTs, Chord is the one which has been studied the most by researchers. For instance, in the *CiteSeer* statistics on *Most Cited Articles in Computer Sciences* [20] the original research paper describing Chord [266] is listed as the fourth most cited Computer Science paper within the period from 1990 to 2012. There are three important consequences:

[2] see Chapter 3

1. Chord is formally well-understood and has been analysed in-depth [102] [291] [191]. We can use such results for our work and to verify that the security solutions we propose do not violate key properties of Chord.
2. A lot of previous work regarding security exists for Chord. Some of these solutions fulfil a subset of the security requirements for decentralised service location we derived in 3. We can thus build (at least to a certain extent) on existing work to fulfil our goal of securing decentralised service location.
3. A multitude of applications has been suggested on top of Chord (e.g. [104] [103] [80]. This implies that the security solutions we derive are applicable to many existing applications.

- Compared to other DHTs, Chord offers some advantageous security properties:

  1. Chord is by default protected against *routing table poisoning attacks* [251]. This is due to the fact that in Chord it is precisely specified for each routing table entry to which DHT-node it has to contain a link. This property is often referred to as *constraint routing tables*. Thus, unless attacker nodes can forge their node-ID, routing table poisoning attacks are no significant threat in Chord [251].
  2. There exist several decentralised solutions which can guarantee secure node-ID assignment for DHTs [53] [98] and specifically for Chord [114]. Since this security challenge has been addressed sufficiently by researchers, we can assume the property of secure node-ID assignment for Chord. This property guarantees that attacker nodes are distributed uniformly over the node-ID space. Further, it implies that attacker nodes cannot forge their node-ID. Hence routing table poisoning attacks are a negligible threat for Chord (see above).

- Chord is a unidirectional DHT [158]. This enables caching of content along routing paths because all lookup paths for a key converge on the unidirectional routing structure. For service location such caching can be beneficial because data items for popular keys can be cached and thus retrieved with less routing hops.

4.1.3 Goals

While there has been work on improving lookup availability for non-unidirectional DHTs, we will show that no sufficient solution for increasing DHT lookup availability in Chord has been proposed. Given the advantages of Chord for security as well as its particular suitability for service location and our prototypical application (real-time communications session establishment), our goal is to extend Chord in such a way that lookup availability can be achieved even in the presence of adversary nodes.

The main security weakness of Chord compared to other DHTs is that—due to its unidirectional routing structure—only a single routing path between query node and root node exist. As we will show, this makes Chord susceptible to attacks on lookup

availability because a single adversary node on this routing path can prevent a lookup from succeeding.

The main goal of our work in this chapter is therefore to extend Chord in a way that lookup availability is increased while maintaining its advantageous security properties. Our intention is to extend regular Chord as little as possible so that formal properties and existing analyses still apply to our extensions. Further, extensions being minimal ensures that our solutions will be applicable to a large number of Chord applications.

Following our overall thesis goal, we envision decentralised enhancements to Chord which do not require a central authority. Moreover, we intend to design DHT algorithms where query nodes can retrieve a data item for a given key without relying on other nodes's assessment. Thus, our objective is to enable a node which starts a DHT lookup to make routing decisions entirely *autonomously*, i.e. neither relying on a central authority nor on the advise or assessment of other nodes.

4.1.4 Attacker Model and Assumptions

General assumptions We assume that the integrity of data items (which are stored in the DHT) can be verified through the application using the DHT (e.g. by using self-certifying data[3] [103] [239]). This implies that nodes can check if a lookup resulted in receiving correct data from the DHT (through calls to the upper-layer application). Consequently, we leave the problem of data integrity to higher layers on top of the DHT (and refer to Chapter 5 for a corresponding solution). Further, we assume that secure node-ID assignment against *Sybil attacks* [84] [98] and prevention techniques against *Eclipse attacks*[4] [84] [251] are used. We expect the use of such techniques as a basis for our extensions. More precisely, we assume that $f_{node-mapping}()$ and $f_{routing-table}()$ return correct nodes (as defined by Chord) even in presence of attacker nodes.

Under these assumptions, attackers cannot control a disproportionately high fraction of nodes relative to the number of attacker nodes in the network. However, attackers can still degrade the availability severely through attacks on *message forwarding* (i.e. $f_{route}()$) of the DHT. We concentrate on these attacks and secure algorithms for $f_{route}()$.

Attacker model We consider the following attacker model: a network of originally only good nodes is infiltrated over a certain period of time by attacker nodes which either join the system or compromise good nodes. After this period, the network is *infiltrated* and has reached a certain state $\sigma_i \in \Sigma$ where the system contains $N_a = f \times N$ adversary nodes and $N_g = (1 - f) \times N$ good nodes, where $f < 1$. The set of adversary nodes, $\mathcal{N}_a$, and the set of non-adversary nodes, $\mathcal{N}_g$, are disjoint, i.e. $\mathcal{N}_a \cap \mathcal{N}_g = \oslash$. We look at this state σ_i where $f \times N$ attacker nodes are distributed uniformly over the node-ID space I. That is, with high probability, any part of the node-ID space I consists to a fraction f

[3]Note that we propose such a solution for P2PSIP in Chapter 5.

[4]Chord has tightly constrained routing tables. Therefore, Eclipse attacks are not a serious threat in Chord.

of adversary nodes and to a fraction $1 - f$ of good nodes. The assumption that attacker nodes are distributed uniformly over the node-ID space holds because we presume secure node-ID assignment (see above).

All adversary nodes in the network may collude (e.g. because they are controlled by a single external identity): the $f \times N$ adversary nodes in the system are aware of each other and share information. Note that this is a strong attacker model because all nodes colluding is the worst case from the perspective of the good nodes in the network. Thus, any mechanism that will work under this assumption will also work (probably even better) if only $g \times N < f \times N$ of the $f \times N$ adversary node collude.

Adversary nodes route exclusively to adversary nodes and do not drop messages: $\forall n_i \in \mathcal{N}_a : f_{route}(k) = n_j \in \mathcal{N}_a$ (i.e. adversary node *suppress* existing good nodes in their routing tables); good nodes route to good and adversary nodes: $\forall n_i \in \mathcal{N}_g : f_{route}(k) = n_j \in \mathcal{N}$. In principle, adversary nodes could also drop messages. However, this would result in a less severe attack on lookup availability because this behaviour can easily be detected through time-outs. In contrary, by continuing to route amongst them (never reaching the target data item) colluding adversary nodes can absorb more DHT routing resources in vain. Thus, by expecting adversary nodes not to drop messages we consider a stronger attacker model. Since nodes are protected against routing table poisoning, for reasonable network sizes, at state σ_i the routing table of any good node in the system contains with high probability $f \times d$ bad nodes and $(1 - f) \times d$ good nodes, where $d < S$ is the number of distinct nodes in the routing table.

Additionally, we assume that any message sent on a single DHT-hop will arrive unchanged (i.e. attacks on the IP-layer are out of scope). Exploring the interaction of attacks on DHT routing with attacks on the underlying IP-layer is out of the scope of our work.

This attacker model is reasonable for analysing attacks on *message forwarding*: The state where all adversary nodes have joined the network and collude constitutes the worst case scenario with respect to attacks on message forwarding. Further, by using variation on f, our model is able to embrace different states of network infiltration by attackers.

4.2 Lookup Availability: Analytical Observations

To mathematically analyse lookup availability in Chord, we start by performing analytical calculations. In particular, we are interested in the maximum success rate that can be achieved in Chord if f is the fraction of uniformly distributed adversary nodes.

4.2.1 The Shield Problem

Consider a lookup for a random key k in a Chord DHT, invoked by a random query node n_q. In Chord any path that leads from n_q to $root_k$ has to pass the predecessor of $root_k$

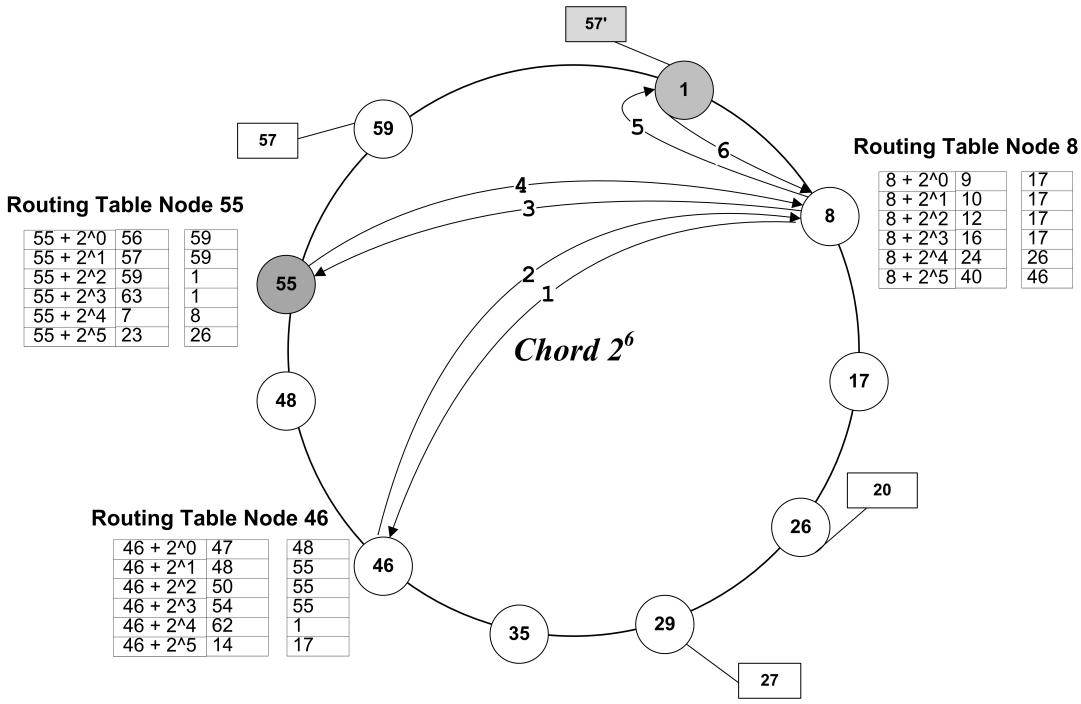

Figure 4.1: The Shield Problem in Chord

(unless $n_q = root_k$). This is due to the fact that in Chord a node can only route to a node with ID higher than the key-ID if it routes to its direct successor in the ring. This direct successor must then be responsible for the key. We call the predecessor of $root_k$ the *shield* for key k and denote this node with $shield_k$:

> **Definition 10 (Shield Node):** *For a given key k, the* shield node*, denoted $shield_k$, is the direct predecessor of $root_k$ in the Chord ring.*

We can define the shield for a key k formally as follows:

$$shield_k = pred(root_k) \tag{4.3}$$

where

$$pred(n_j) = \begin{cases} n_k \in \mathcal{N} | \, (n_j > n_k \wedge (\neg \exists n_l \in \mathcal{N} | n_j > n_l > n_k)) & \exists n_k \in \mathcal{N} | n_k < n_j \\ n_k \in \mathcal{N} | \, (n_k > n_j \wedge (\neg \exists n_p \in \mathcal{N} | n_k < n_p)) & otherwise \end{cases} \tag{4.4}$$

Since in Chord any path to $root_k$ has to pass $shield_k$, the shield can effectively deny access to the data item belonging to key k. This is why we call this node the *shield*. If the shield for a key is a passive attacker it simply drops any message with destination k. If it is an active attacker (and $root_k$ is not an adversary), it routes to the first adversary

node that succeeds $root_k$ in the ring[5]. The querying node n_q does not know which node is $root_k$. Thus, it can be fooled into believing that the first adversary node in the ring that succeeds $root_k$ were actually $root_k$.

Example 4.1: *Figure 4.1 shows an example Chord network. Adversary nodes are depicted grey.* n_{55} *is the shield for key* 57*. Thus, it can fool the query node* n_8 *and return another adversary node,* n_1*, as* $root_{57}$ $\langle 4 \rangle$*.* n_8 *has no means to determine per se if* n_1 *is indeed* $root_{57}$*. Note that* n_1 *might return a false data item for key* 57*,* $57'$ $\langle 6 \rangle$*. This can be detected in our attacker model. However, the important thing to notice is that the lookup will fail if* $shield_{57}$ *is an adversary node, regardless if* n_1 *returns data or not.*

4.2.2 An Upper Bound on Lookup Success in Chord

When analysing the shield problem it becomes evident that any lookup in Chord can only succeed if *both* the shield *and* the root node for a specific key k are non-adversary nodes. We have seen that a lookup will never succeed if the shield is an adversary node. Clearly, if the root node is an adversary, no lookup can succeed because the root node can deny answering to any lookup request or reply with a forged value for the key.

Analytically, we consider a random lookup for a key. Our sample space Ω consists of the root node and the shield node of such a random lookup each being adversary or not. Our sample space implies that we only consider paths of length ≥ 2. Thus, our results hold only for networks with $N \geq 16$ ($N \geq 16 \Rightarrow \vartheta \geq 2$) and lookups where $n_q \neq root_k$. We denote the following events for such a random lookup:

$$E = \{ \text{“root node is good”} \} \tag{4.5}$$

$$\Phi = \{ \text{“shield node is good”} \} \tag{4.6}$$

$$\Gamma = \{ \text{“root node is good and shield node is good”} \} \tag{4.7}$$

Since nodes are distributed uniformly[6], E and Φ are statistically independent. $P(E) = P(\Phi) = 1 - f$ by definition of f. $\Gamma = E \wedge \Phi$. Thus, we have

$$P(\Gamma) = P(E) \times P(\Phi) \tag{4.8}$$

$$P(\Gamma) = (1 - f)^2 \tag{4.9}$$

Equation 4.8 holds since E and Φ are statistically independent. Equation 4.9 is the probability that the root and the shield are both non-adversary nodes for an arbitrary lookup. This is the minimum requirement for any lookup to succeed (note that a lookup

[5]Remember that adversary nodes only route to adversary nodes in our attacker model.

[6]This property is ensured through secure node-ID assignment which is part of our assumptions.

f	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8
$(1-f)^2$	0.81	0.64	0.49	0.36	0.25	0.16	0.09	0.04

Table 4.1: Upper Bound on Lookup Success Rate in Regular Chord

can still fail if no path to the root exists where all other nodes on the path are non-adversary nodes as well). Thus, equation 4.9 gives us an upper bound for the success rate of an arbitrary lookup in Chord:

$$P(lookupsuccess) \leq P(\Gamma) \tag{4.10}$$

$$P(lookupsuccess) \leq (1-f)^2 \tag{4.11}$$

Table 4.1 shows values for the upper bound on lookup success from equation 4.11 for $f = 0.1, \ldots, 0.8$. it can be observed that with increasing attacker rate f, the upper bound on lookup success decreases drastically. For instance, if $f = 0.5$ attackers have infiltrated the network, at most 0.25 of lookups can succeed.

The bound in equation 4.9 is specific to Chord and lower than previously published bounds for DHTs. To show this, we compare our result with the bounds from [261]. In [261], one can find the following generic bounds on the probability of lookup failure for a DHT:

$$f^i \leq P(lookupfailure) \leq \left(1-(1-f)^{\vartheta}\right)^i \tag{4.12}$$

where i is the number of independent (i.e. not having a single node in common) paths from query node to root node. In Chord $i = 1$, since any lookup has to pass the shield. Further, in Chord, $\vartheta = \frac{1}{2}\log N$ is the mean hop length of a path [164]. We can thus convert the upper bound on lookup failure in equation 4.12 into an lower bound on lookup success or Chord by taking the opposite event as follows [246]:

$$1 - \left(1-(1-f)^{\left(\frac{1}{2}\right)\log N}\right) \leq P(lookupsuccess) \tag{4.13}$$

We need to convert the upper bound on the success rate from equation 4.9 to a lower bound for failure rate in order to compare it with the bounds in equation 4.12. The opposite event in our sample space Ω of the event Γ (the upper bound for lookup success rate) is a lower bound for lookup failure rate. We have the following event for a lower bound on failure rate:

$$\Pi = \{\text{“}root\ node\ is\ bad\ or\ shield\ node\ is\ bad\text{”}\} \tag{4.14}$$

Note that Π is the opposite event of Γ in our sample space Ω. Thus, we have

$$P(\Pi) = 1 - P(\Gamma) \tag{4.15}$$

$$P(\Pi) = 1 - (1 - f)^2 \tag{4.16}$$

$$f \leq 1 - (1 - f)^2 \leq P(lookupfailure) \tag{4.17}$$

Equation 4.17 shows that the event Π has indeed a higher probability than the general lower bound for lookup failure in DHTs from [261]. Thus, the opposite event, i.e. Γ from equation 4.9, is lower than the general upper bound for lookup availability in DHTs. The left inequality of equation 4.17 holds because $f < 1$ by definition. Note that our upper bound on the success rate is only applicable to Chord whereas the bounds from [261] apply to DHTs in general.

4.3 Algorithms for Increased Lookup Availability

In this section we describe our extensions to Chord for increasing lookup availability. In principle, we combine three techniques:

1. We use the direct successor list of each node to accomplish independent (or alternate) multipath routing.

2. To overcome the shield problem we directly route to replica roots.

3. We use density checks on each iterative routing hop to detect paths that contain adversary nodes as early as possible.

Complete-knowledge iterative routing For all our techniques described in this chapter we use the following general (global) extension to Chord: Each node in the network must support iterative routing where at each routing hop the query node receives not only the next hop from the node it queried (as in regular Chord) but instead the whole routing table T_r of the queried node and its list of direct successors T_s (a similar extension to Chord has been suggested in [106]). Note that this extension only affects the size of each iterative query response. In particular, it does not affect the total number of links stored at each node because the additional information received at each iterative routing step is only stored temporarily during the lookup.

We call this extension *complete-knowledge iterative routing* because at each iterative routing hop the query node receives the complete information the hop node has about the network. All other routing techniques we introduce are solely computed at the query node (locally). Thus, it does not affect the success rate of a lookup if other nodes in the network use these techniques or not.

4.3.1 Chord Multipath Routing

In the case a lookup path has failed, we explore two techniques to let the lookup continue (we refer to this as *failover routing*)[7]:

1. by starting a new independent path at the query node (*independent restart*)
2. by starting a new path at the closest node to the key received during the previous path which has not been used in the lookup so far (*backtracking*)

For both of these techniques the query node maintains a temporary list T_m of nodes it has used in the lookup so far. In each individual path it explores during a lookup the query node only uses nodes it has not used before in this lookup, i.e. nodes not in T_m. In regular Chord the direct successor list, T_s, is only used for redundancy (i.e. in the case of node failures). We allow each node to use the list of direct successors, T_s, on every routing hop. Since we use complete-knowledge iterative routing a query node can in principle use for the next hop any node from the routing table T_r and the direct successor list T_s it received from the node on the last hop. However, for our extensions at each hop the nodes in T_s are only used in routing if all nodes from T_r have been used previously in the lookup, i.e. if all nodes from T_r are already contained in T_m. n_q (the query node) always routes greedy (as in regular Chord): It always uses the node $n_i \in T_r$ (or $n_i \in T_s$ if $T_r \subseteq T_m$) with the highest node-ID smaller than k, or the direct successor of k if n_i is the first entry in T_r (T_s, respectively) and n_i is larger than k. This assures that queries make progress towards the key. We refer to this general algorithm as *Chord Multipath Routing, CMR.*

According to our definition, backtracking explores alternate and not independent paths. Independent restart always explores independent paths. We refer to *Chord Multipath Routing* as *Alternate Multipath Routing, AMR*, if *backtracking* is used as the *failover routing* technique. We refer to *Chord Multipath Routing* as *Independent Multipath Routing, IMR,* if *independent restart* is used as the *failover routing* technique.

Figure 4.2 shows the flow chart for our algorithm of *Chord Multipath Routing (CMR).* The function `closest_node_to_key`(T_{temp}) represents regular, greedy Chord-routing applied to the nodes in T_{temp} (as described previously): It always returns the node n_i in T_{temp} with the highest node-ID smaller than k, or the direct successor of k if n_i is the first entry in T_{temp} and n_i is larger than k. Note that *smaller* and *larger* have a semantic according to the ring structure of Chord, i.e. using modular arithmetic. Similarly, in figure 4.2 $\leq$ and $\geq$ are intended to include modular arithmetic so that routing across the 0 in the node-ID space is possible.

It can be observed that T_m serves as a memory list of nodes the query node n_q has already visited in the lookup. For *independent restart*, after a path has failed the lookup continues (i.e. *restarts*) from the query node. Not resetting T_m at this stage ensures that each path in the lookup is indeed *independent* for *independent restart.*

[7]Remember that in our model a lookup consists of several individual paths.

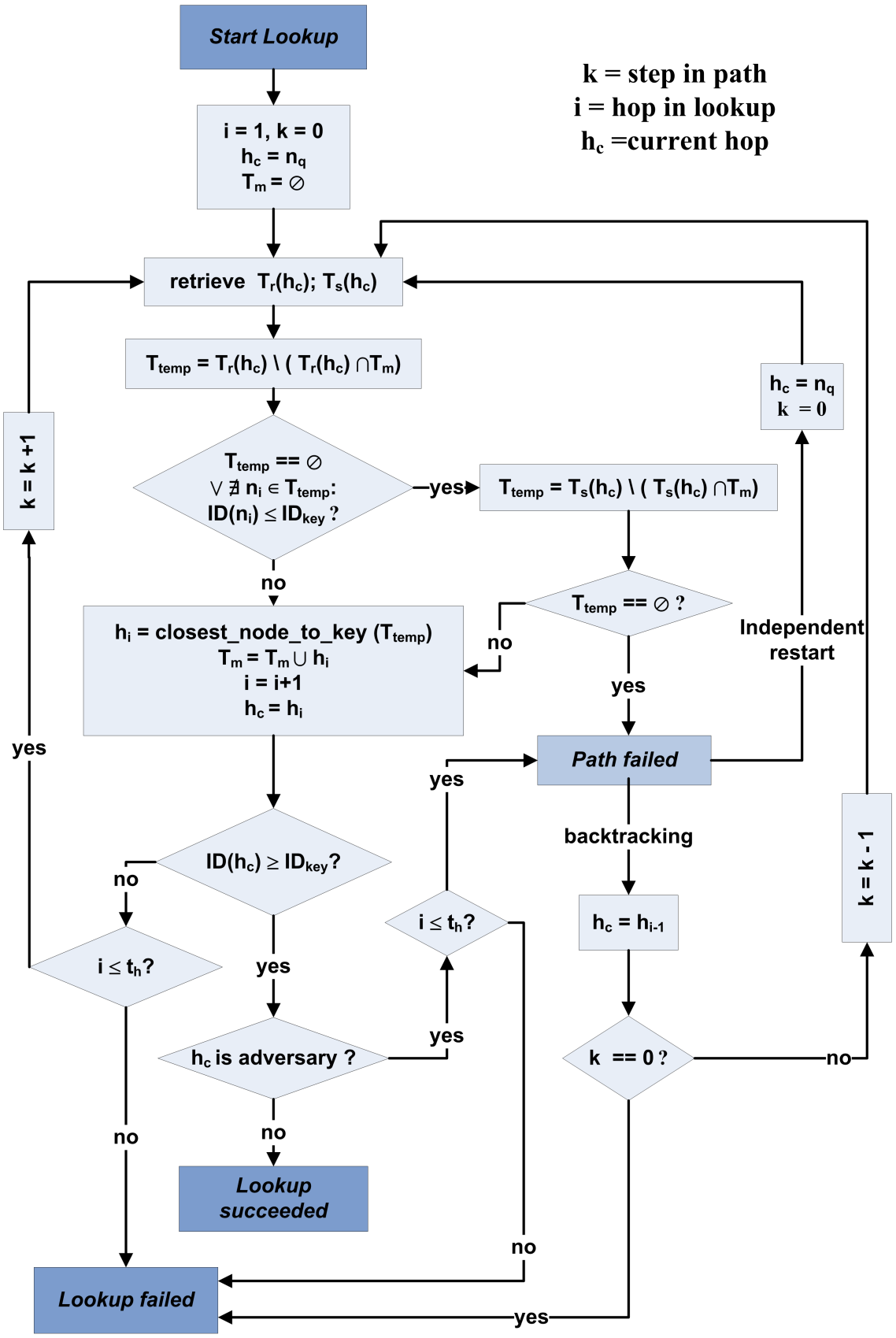

Figure 4.2: Flow Chart for Chord Multipath Routing

Further, note that CMR fails if the root node for the key, $root_k$, is adversarial. CMR can thus help if intermediate nodes on the DHT routing path are adversary (in which case regular Chord fails), but not in the case where the root node is adversary itself. To enable successful lookup also in this case, i.e. where the root node is adversary, we refer to Direct Replica Routing (DRR) in 4.3.2.

With unidirectional greedy routing independent paths converge towards the root [158]. Using T_s allows a path to continue if at some hop all entries in T_r smaller than the key k are already in T_m. For *independent restart*, using T_m guarantees that all paths in a lookup are independent. Further, independent restart allows for up to s (the number of entries in every T_s) independent paths because this is the maximum number of independent paths that can converge on the penultimate hop before reaching the root. Because with *backtracking* a new path does not necessarily start at the query node, n_q, this technique only explores alternate (but not independent) paths.

In our model, adversary nodes suppress non-adversary nodes in the routing tables T_r and T_s they return. This implies for that once a path has reached an adversary node, only adversary nodes will be added to T_m on this path. However, node-ID suppression attacks do not prevent a query node from subsequently exploring a path with only non-adversary nodes on every hop.

4.3.2 Direct Replica Routing

To tackle the situation where $root_k$, $shield_k$, or both are malicious we allow to route directly to the replica roots of k. Chord replicates content at $r-1$ replica roots which are the $r-1$ nodes directly succeeding $root_k$ in the ring. However, in regular Chord the replica roots are only used for redundancy (i.e. node failure of $root_k$).

We extend Chord in a way that routing to the replica roots of a key k is possible without passing $shield_k$ or $root_k$. We refer to the overall set of r replica roots for key k as REP_k, which includes $root_k$ and is defined as follows:

$$REP_k = \left\{root_k^1, \ldots, root_k^r\right\} \tag{4.18}$$

where $root_k^1$ is $root_k$ and $root_k^x$ refers to the $(x-1)$-th replica root directly succeeding $root_k$ in the Chord ring. We allow *direct* routing to a node $n_i \in REP_k$ if $n_i \in T_s$ (we refer to this as *Direct Replica Routing, DRR*). Because at every hop T_s contains s direct successors in the ring, the query node n_q can check if some of these nodes are in REP_k: n_q simply has to verify if $\exists n_j \in T_s | k \leq n_j \leq root_k^r$ (i.e. n_q has to check if some nodes in T_s have a node-ID greater than k which can be at maximum r nodes in T_s).

If all replica roots retrieved at some hop have been queried without success, a failover (backtracking or independent restart) is pursued. Using direct replica routing results in each key k having effectively s shield nodes (the s direct predecessors of $root_k$) which we denote with $shield_k, \ldots, shield_k^s = SHI_k$. By setting $s = 2r$ (globally) in the system,

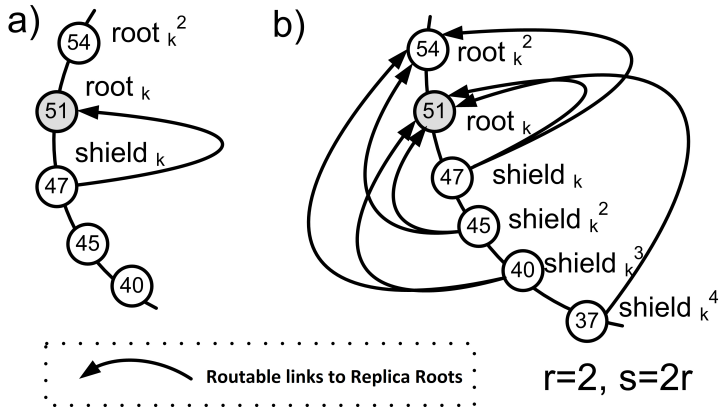

Figure 4.3: Reaching Replica Roots with Direct Replica Routing

any of the $r + 1$ closest shield nodes to a particular key k can route directly to any of the r replica root nodes for k. In general, setting $s \geq r$ ensures that the last replica root $root_k^r$ is accessible from $s - r + 1$ shield nodes.

Figure 4.3 exemplifies how replica roots can be reached through more than one node with *DRR* (b), compared to regular Chord (a). For both cases, the figure shows all routable links from shield nodes to replica roots in a network without attackers. In an infiltrated network, any T_s the query node n_q will receive from an adversary node will only contain the next s adversary nodes in the ring. However, by setting $s = 2r$ we guarantee that reaching one non-adversary shield node of the $r + 1$ closest shield nodes to k is enough to reach one non-adversary replica root in REP_k (if existing). Note, for instance, that in the example node $shield_k^4$ can only reach $root_k$ (and not $root_k^2$) because it is more than $r + 1$ directly succeeding nodes away from $root_k$.

Figure 4.4 shows the flow chart of the algorithm for Direct Replica Routing, in conjunction with Multipath Routing. As in figure 4.2, $\leq$ and $\geq$ have a semantic according to the ring structure of Chord, i.e. using modular arithmetic that allows for routing across the 0 point in the node-ID space. We refer to the combination of DRR and CMR as MRR for *Multipath Replica Routing*. Essentially, MRR uses DRR to check at each routing hop h_c if there are links in $T_s(h_c)$ which are succeeding the key k. If this is not the case, the current hop node h_c is too far away from $root_k$ and regular CMR routing continues. If there are indeed nodes in $T_s(h_c)$ succeeding the ID of the key, these nodes must be replica roots. DRR then checks each of these nodes consecutively to retrieve the key. As long as any one of these nodes (i.e. nodes in $T_s(h_c)$ succeeding the key) is non-adversary, the lookup is successful. Otherwise, the lookup can still continue using *independent restart* or *backtracking*, respectively.

4.3.3 Detecting Node-ID Suppression Attacks

Recall that in our attacker model a network of good nodes is infiltrated and routing tables in Chord are constrained (and therefore protected against routing table poisoning). Thus,

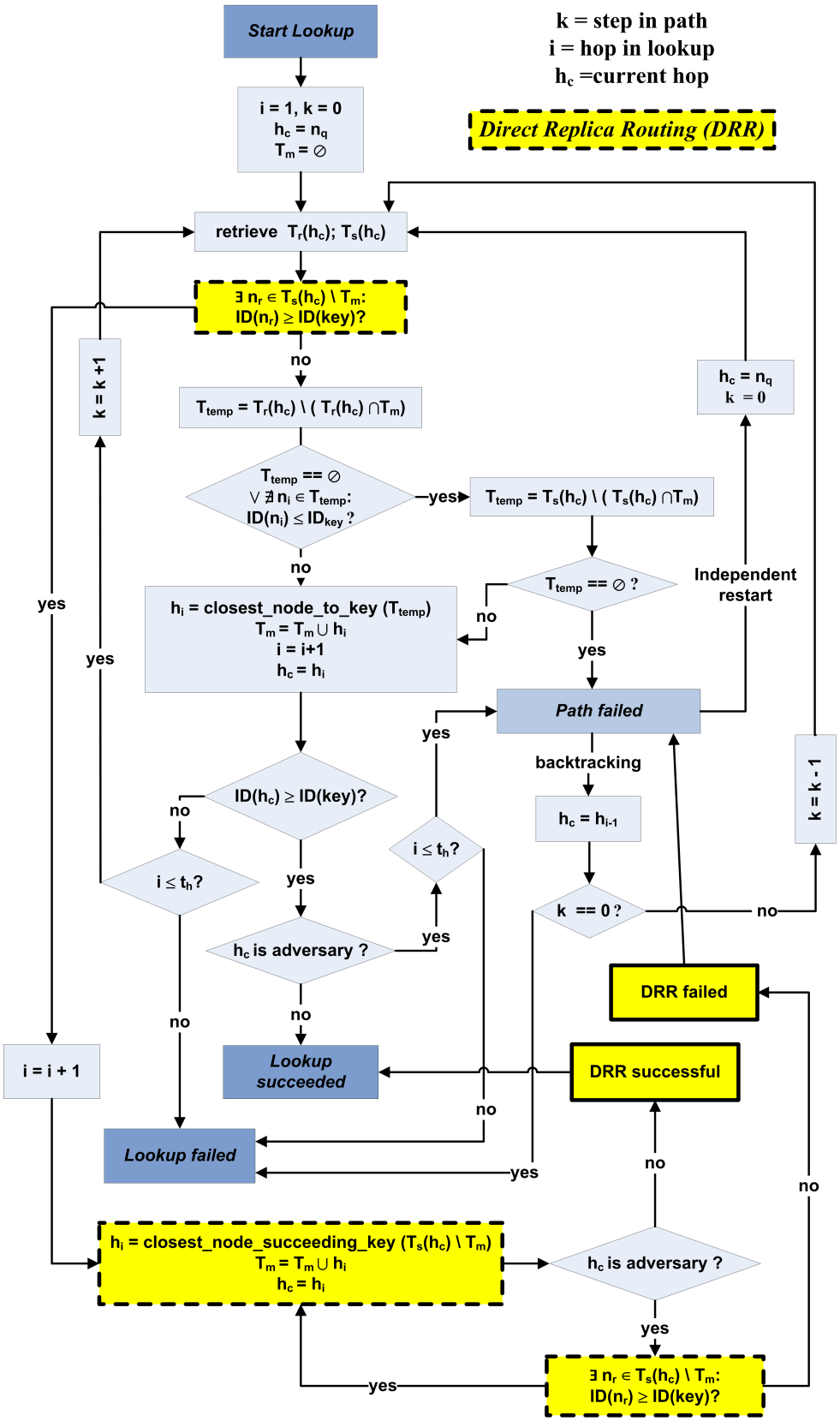

Figure 4.4: Flow Chart for Direct Replica Routing in Conjunction with Multipath Routing

good nodes have (with high probability) $f \times d$ adversary nodes and $(1-f) \times d$ good nodes in their routing table T_r as well as $f \times s$ adversary nodes and $(1-f) \times s$ good nodes in T_s. Adversary nodes suppress good nodes in the routing tables they return. This enables them to attack lookup availability even if complete-knowledge iterative routing is used by the query node.

We can detect these attacks by using *density checks*: the query node n_q calculates the average distance $\alpha(n_q)$ between nodes in its direct successor list T_s as follows:

$$\alpha(n_q) = \frac{lastentry(T_s(n_q)) - firstentry(T_s(n_q)}{s} \tag{4.19}$$

where $lastentry$ is the last entry in $T_s(n_q)$ and $firstentry$ is the first entry in $T_s(n_q)$. From any routing table $T_s(n_i)$ that n_q receives from a node n_i, n_q can compute $\alpha(n_i)$ (analog to equation 4.19) and compare it with its own average distance $\alpha(n_q)$ by computing

$$\delta = \frac{\alpha(n_i)}{\alpha(n_q)} \tag{4.20}$$

where δ is the density-ratio of the direct successor list between n_q and n_i. If $\delta \geq t_d(n_q)$ (where $t_d(n_q)$ is the density threshold of the query node), n_q considers n_i to be an adversary node.

An adversary node n_a can only decrease its $\alpha(n_a)$ by either creating artificial entries in $T_s(n_a)$ (which will be detected on the next hop if such an entry is chosen by n_q) or limit suppression of good nodes in $T_s(n_a)$ (which would give n_q access to good nodes). With a low density threshold t_d there is a risk of falsely estimating good nodes as adversary ones. However, this only affects $f_{route}()$ of n_q locally.

4.4 Assessment of the Proposed Algorithms

4.4.1 Theoretical Analysis

Our proposed extensions to Chord provide several independent paths between n_q and $root_k$, and route directly to the replica roots of a key k so that not a single root node can control all access to data items for a key k. Thus, there exist at most s shield nodes (one on the penultimate hop of every independent path) denoted $shield_k, \ldots, shield_k^s$ and for every key k there are r routable replica roots, denoted $root_k, \ldots, root_k^r$.

We now extend the theoretical results for regular Chord to this case. Analytically, we use a sample space Ω for a random lookup. Ω samples all shields and all replica roots for an arbitrary key and determines for each shield and replica root node if it is an adversary node. We are interested in the following events in our sample space:

$$A = \{\text{“at least one shield node is non-adversary”}\} \tag{4.21}$$

$$B = \{\text{“at least one replica root is non-adversary”}\} \tag{4.22}$$

$$E = \{\text{“at least one replica root and one shield node are not adversary nodes”}\} \tag{4.23}$$

Event E states an upper bound on the success rate for an arbitrary lookup because this event is the minimum requirement for any lookup to succeed (a lookup can still fail under this event if all paths explored contain at least one adversary node). We now derive the probability for event E for the case that we have precisely s shield nodes and r routable replica roots for any key k:

$$P(A) = 1 - f^s \tag{4.24}$$

$$P(B) = 1 - f^r \tag{4.25}$$

$$P(lookupsuccess) \leq P(E) = P(A) \times P(B)) \tag{4.26}$$

Inserting equation 4.24 and 4.25 in Equation 4.26 we get

$$P(lookupsuccess) \leq (1 - f^s) \times (1 - f^r) \tag{4.27}$$

Equation 4.27 thus provides an upper bound on lookup success for our proposed extensions (i.e. Multipath Replica Routing). Note that it is possible to multiply $P(A)$ and $P(B)$ because these events are statistically independent in our model. Adapting the lower bound from equation 4.13 to s independent paths we get [261]

$$1 - \left(1 - (1 - f)^{\left(\frac{1}{2}\right)\log N}\right)^s \leq P(lookupsuccess) \tag{4.28}$$

With our extensions, there exist at most s independent paths and exactly r replica roots. Since equation 4.27 provides an upper bound, it holds for our extensions even though some lookups might explore less than s independent paths. However, the lower bound in equation 4.28 does not apply to our extensions. Still, it indicates analytically that as more independent paths are explored (which is the effect of our multipath-extensions to Chord) the lower bound on the success rate increases. In any case, we are interested in the maximum success rate (and thus the upper bound) that our extensions can theoretically achieve. The upper bound enables us to assess our algorithms compared to theoretical limits on lookup success.

Table 4.2 shows the upper bound of equation 4.27 for *MRR* with $s = 16$ and $r = 8$. It can be observed that for attacker rates less than $f = 0.7$, the upper bound on lookup success is very close to 1. Even for attacker rates of $f = 0.7$ (or $f = 0.8$), the upper bound on lookup success is higher than 0.93 (or higher than 0.80 for $f = 0.8$, respectively).

f	$(1-f^s)\times(1-f^r)$
0.1	0.99999999
0.2	0.99999744
0.3	0.99993439
0.4	0.99934421
0.5	0.99607855
0.6	0.98292647
0.7	0.93922028
0.8	0.80880271

Table 4.2: Upper Bound on Lookup Success with Multipath Replica Routing (s=16, r=8)

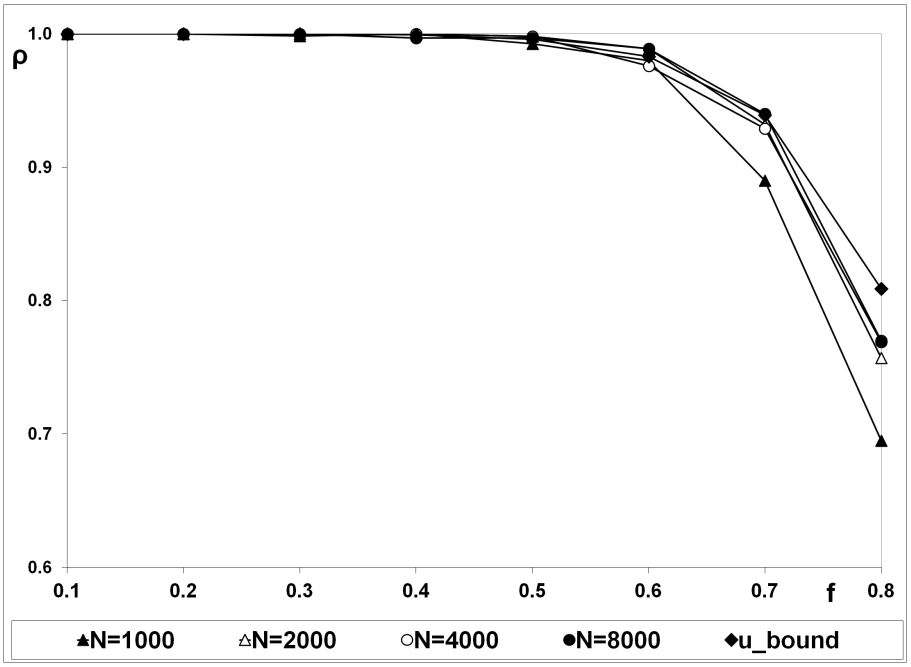

Figure 4.5: *MRR-r* with $t_h = \infty$ Compared to Theoretical Upper Bound ($N = 1000, 2000, 4000, 8000$)

4.4.2 Simulation Results

We implemented our algorithms and Chord routing extensions in the *planetsim* DHT simulation framework [15]. This implementation allows us to conduct various simulations to evaluate the effectiveness of our proposed algorithms. In all our experiments we simulated a total of 1000 lookups in 10 random Chord networks (i.e. 100 lookups in each of the 10 randomly created networks) with $|I| = 2^{32}$ and adversary nodes behaving according to our attacker model. We set $r = 8$ and $s = 16$ in all the experiments we present here. These values seem acceptable for nodes to maintain links in $T_s()$ and for data replication. Further, Table 4.2 shows that in theory these values should suffice to achieve a high lookup success rate. In our simulations, we only consider lookups where $T_s(n_q) \cap REP_k = \oslash$, i.e. lookups where no replica root is contained in the direct successor list of the query node. For details about our simulator and experiments, as well as more

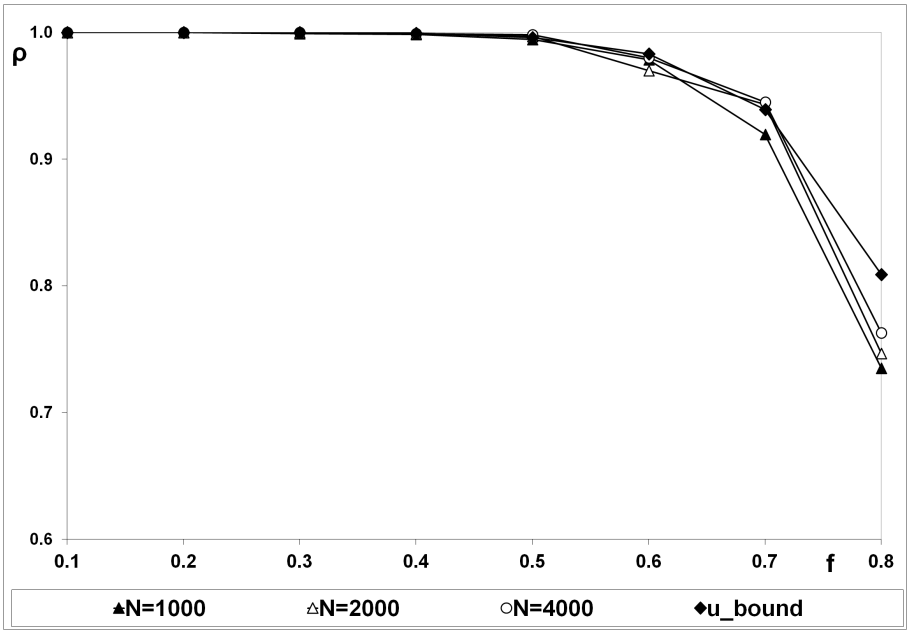

Figure 4.6: *MRR-b* with $t_h = \infty$ compared to theoretical upper bound ($N = 1000, 2000, 4000$)

detailed results of the experiments we conducted we refer to Appendix B.1.

Comparing algorithms to analytical upper bound We first simulated Chord Multipath Routing (*CMR*) combined with Direct Replica Routing (*DRR*) (i.e. *MRR, Multipath Replica Routing*) to see how close our algorithms come to theoretical limits. We simulated MRR with independent restart (*MRR-r*) and backtracking (*MMR-b*) for various network sizes, N, and attacker rates, f, and compared it to the upper bound on ρ from equation 4.27. Figure 4.5 shows results for independent restart for different network sizes. Figure 4.6 displays the results for backtracking. It can be observed that our algorithms come very close to the upper bound (*u_bound*) on lookup success in equation 4.27, almost reaching theoretical limits even for high attacker rates. We noticed, however, that with $t_h = \infty$, the average hop count χ can get quite high with increasing levels of network infiltration. For instance, for $f = 0.6$ and $N = 2000$, we obtained a success rate of 98.90%. But at the same time we measured an average of 436.26 hops per lookup (the average hop count values corresponding to the lookup success rates in Figures 4.5 and 4.6 can be found in Appendix B.1).

Results with hop count threshold In the type of DHT applications we consider in this thesis (i.e. decentralised service location and in particular real-time communication session establishment) the time it takes for a lookup to succeed is crucial. To reflect this requirement and investigate the effectiveness of our algorithms with a timing constraint, we conducted simulations with a hop threshold t_h. Figure 4.7 displays *MRR* with backtracking (*-b*) and independent restart (*-r*) compared to regular Chord with independent restart (*RC*) for $t_h = 50$ in a network of $N = 4000$ nodes. Additionally, the figure shows

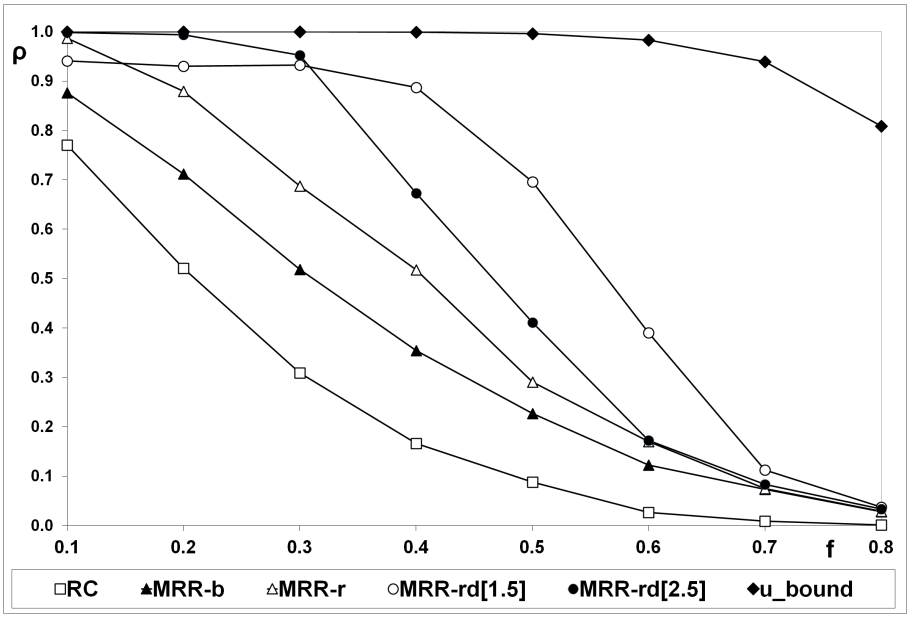

Figure 4.7: Success Rate for *MRR* with Hop Count Threshold Compared to Regular Chord and Upper Bound ($t_h = 50$, $N = 4000$)

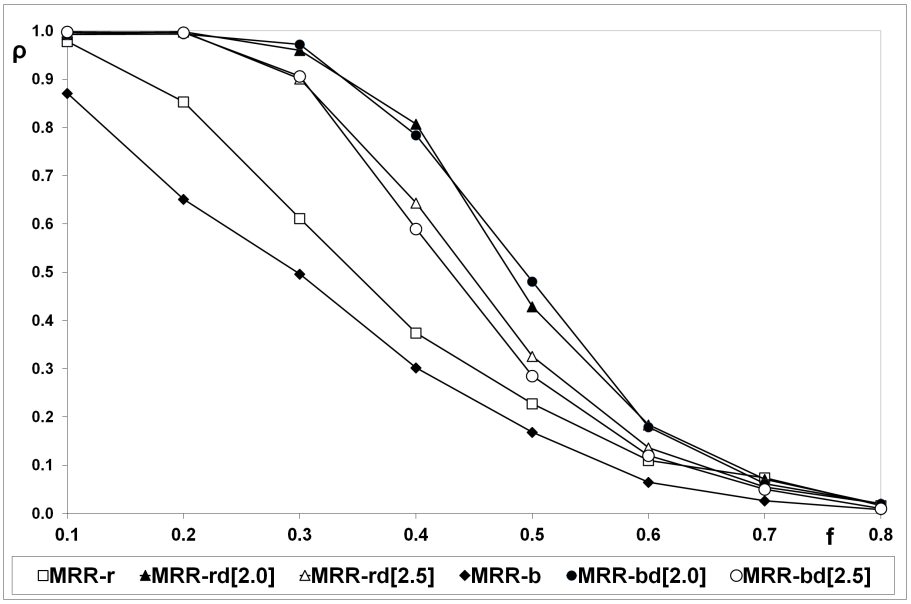

Figure 4.8: Success Rate for *MRR-r* and *MRR-b* with and without Hop Count Threshold ($t_h = 50$, $N = 8000$)

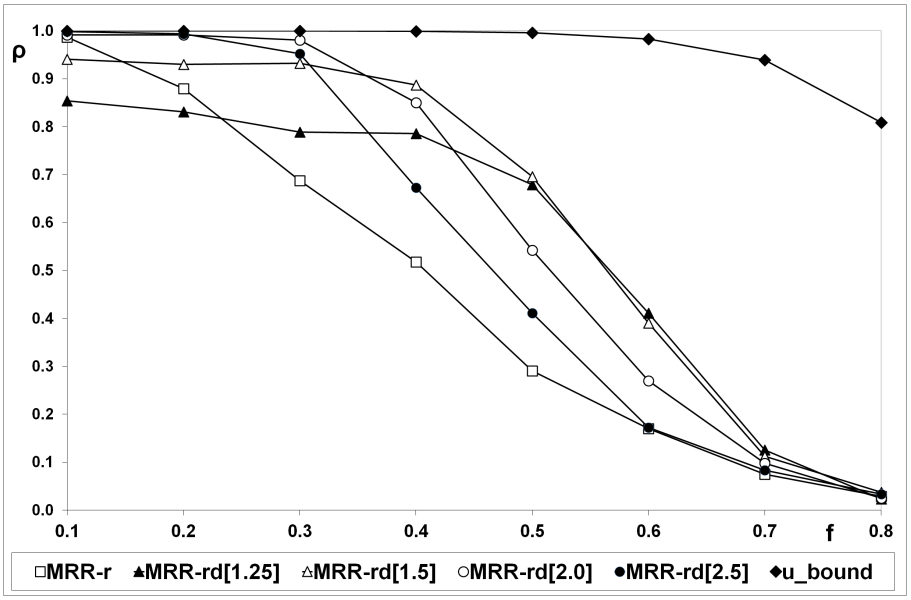

Figure 4.9: Success Rate for *MRR* with Independent Restart (*MRR-r*) and Different Density Thresholds t_d ($t_h = 50$, $N = 4000$)

the success rate for MRR-r with density checks (*MRR-rd*) for $t_d = [1.5, 2.5]$.

It can be noticed that independent restart performs better than backtracking for attacker rates up to $f = 0.7$. Further, the detection of node-ID suppression attacks with density checks on every hop increases lookup availability perceptibly. The results show that only with density checks it is possible to achieve a lookup success rate of more than 90% for attacker rates up to $f = 0.3$. One can see that a higher threshold t_d is better suited for low attacker rates, whereas a lower threshold results in better performance for high attacker rates. Precisely, in Figure 4.7, $t_d = 1.5$ performs better than $t_d = 2.5$ for attacker rates higher than $f = 0.3$. In general, it is advisable to set $t_d < \frac{1}{f}$ because the range of an attacker's successor list increases reciprocally to f with node-ID suppression attacks.

Figure 4.8 displays the success rate for *MRR-r* and *MRR-b* without and with different density thresholds ($t_d = [2.0, 2.5]$) for $t_h = 50$ in a network of $N = 8000$ nodes. Also these results show that only with density checks it is possible to achieve a lookup success rate of more than 90% for attacker rates up to $f = 0.3$. In particular, with a density threshold of $t_d = [2.0]$ or $t_d = [2.5]$ the lookup success rate is 100% for attacker rates up to $f = 0.2$, and with a density threshold of $t_d = [2.0]$ the lookup success rate is 96% or more for $f = 0.3$, independently of using iterative restart or backtracking.

Varying the density threshold Next, we investigated variations of the density threshold, t_d, in more detail. Figure 4.9 exemplarily shows results for *MRR-r* with different values of t_d in a network of 4000 nodes and a hop threshold set to 50. We obtained comparable results for backtracking and for other network sizes. These results and figures similar to Figure 4.9 for different network sizes and backtracking as well as independent

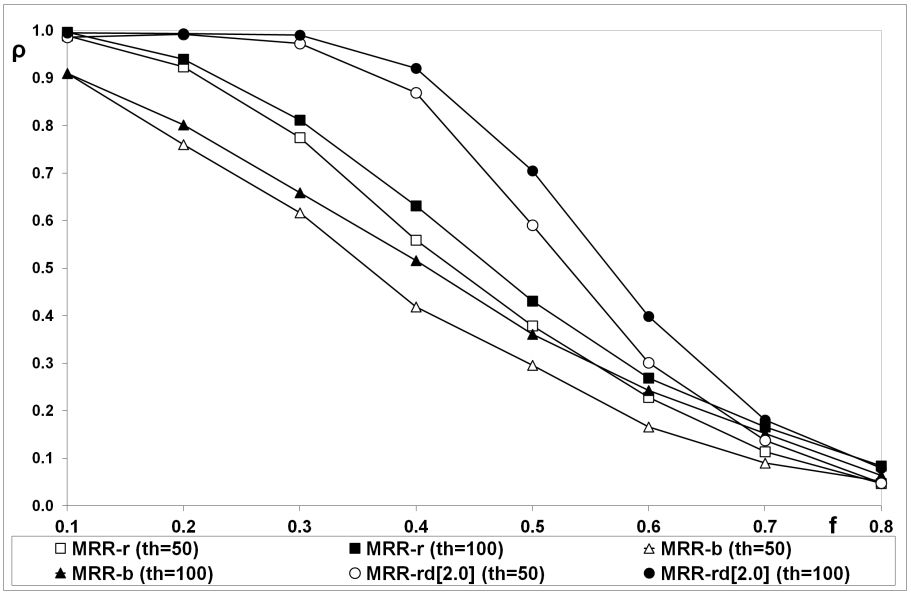

Figure 4.10: Effect of Increased Hop Threshold t_h for Different Algorithms ($N = 2000$)

	t_h	t_d	ρ	χ
MRR-r	50	∞	0.56	30.52
MRR-r	100	∞	0.63	49.90
MRR-rd	50	2.0	0.87	16.85
MRR-rd	100	2.0	0.92	21.40
MRR-b	50	∞	0.42	33.97
MRR-b	100	∞	0.52	58.69

Table 4.3: ρ and χ Compared for $t_h = 50$ and $t_h = 100$ for Different Algorithms ($f = 0.4$, $N = 2000$)

restart can be found in Appendix B.1. The results in Figure 4.9 illustrate that a density threshold of $t_d = 2.0$ can achieve nearly 100% lookup success for attacker rates of $f \leq 0.3$ (we confirmed this result for backtracking and other network sizes as well, see Appendix B.1). Moreover, observe that $t_d = 2.0$ and $t_d = 2.5$ provide better results for $f \leq 0.3$, whereas $t_d = 1.5$ yields better results for $f \geq 0.4$. A very tight density threshold, i.e. $t_d = 1.25$, performs even slightly better for high attacker rates ($f \geq 0.6$), but very poorly for low attacker rates.

In reality, the attacker rate f might not be known to nodes. For moderately infiltrated networks, $t_d = 2.0$ performs best in all our experiments. For highly infiltrated networks, achieving a high lookup success rate seems in any case unrealistic, and $t_d = 2.0$ performs only slightly worse than $t_d = 1.5$ in this case. Thus, when the exact attacker rate is unknown, 2.0 seems a reasonable recommended setting for t_d.

	f	t_d	ρ	χ
MRR-r	0.5	∞	0.29	40.72
MRR-rd	0.5	1.5	0.7	26.03
MRR-rd	0.5	2.0	0.54	31.75
MRR-r	0.3	∞	0.69	25.33
MRR-rd	0.3	1.5	0.94	11.89
MRR-rd	0.3	2.0	0.98	9.39

Table 4.4: ρ and χ for *MRR-r* with and without Density Checks ($f = 0.3$ and $f = 0.5$, $t_h = 50$, $N = 4000$)

Increasing the hop count threshold and keeping the average hop count low Figure 4.10 visualises the effect of an increased hop count threshold, t_h, of 100 in comparison to $t_h = 50$. The figure shows results for *MRR-r*, *MRR-b*, and *MRR-r* with the best density threshold for moderate attacker rates, $t_d = 2.0$, in a network of $N = 2000$ nodes. Not surprisingly, for each of these algorithms an increased hop count threshold results in an increased success rate. However, the gain is rather small. Most importantly, for attacker rates up to $f = 0.3$, the gain of an increased hop count threshold for *MRR-rd[2.0]* (i.e. the best algorithm for $t_h = 50$) is negligible (mostly because the success rate for $t_h = 50$ is already very close to 100% in these cases). Further, the gain in lookup success rate has to be weighted to the potential increase in the hop count. Table 4.3 shows the average hop count corresponding to the results in Figure 4.10 for $f = 0.4$. For regular *MRR-r* and *MRR-b* (i.e. without a density threshold), an increase of at most 0.1 in lookup success comes at the price of an increase of more than 60% in the average hop count. We obtained very similar results for $N = 1000$ and $N = 4000$ (see Appendix B.1).

In addition to increasing the success rate, density checks also significantly decrease the average hop count χ. Table 4.4 exemplarily illustrates this by showing ρ and χ for *MRR-r* (with and without density checks) for $f = 0.3$ and $f = 0.5$, $t_h = 50$, and $N = 4000$. For $f = 0.5$, $t_d = 1.5$ has clearly the best success rate. But in addition, the average hop count is 36% (or 14 hops on average) lower than for *MRR-r* without using density checks. For $f = 0.3$, $t_d = 2.0$ achieves a lookup success rate of 0.98 with less than 10 hops on average. The lookup success is 29 percent-points higher than without using density checks at all; at the same time the hop count is almost 16 hops less on average.

For $N = 8000$ and an attacker rate of $f = 0.3$, we obtained an average hop count of less than 10 hops for *MRR-bd* and settings of $t_h = 50$ and $t_d = 2.0$, while also achieving a lookup success rate of $\rho = 0.97$ (see Appendix B.1). This is a similar hop count and almost the same success rate as for *MRR-rd* and $N = 4000$ ($\rho = 0.98$) with the same settings (compare Table 4.4). Overall, our algorithms can thus achieve (in the network sizes we simulated) a lookup success rate of $\rho = 0.97$ (or even $\rho = 0.98$ in networks with up to 4000 nodes) for networks with up to 30% attackers when a hop count threshold of $t_h = 50$ is applied as a limit for lookups to succeed. At the same time, the average hop count is less than 10 hops for such medium infiltrated networks (i.e. $f = 0.3$).

Note that on the Internet, the average *Round Trip Time (RTT)* can very roughly be

estimated as $200ms$ [7]). For iterative DHT routing (as in our algorithms), one DHT hop results in one round trip from query node to hop node and back. For an Internet-scale DHT, our hop count results can thus very roughly be mapped to the same amount of Internet-scale RTTs. Our results thus imply an overall lookup time in the order of a few seconds which is acceptable for service location (e.g. $2s$ in case of 10 hops). For instance, the ITU-T regards less than $8s$ to be acceptable for phone call setup delay [30]. Also, for smaller-scale DHTs (e.g. where all—or most—nodes are located within a nationwide ISP), shorter RTTs are likely to be expected. We hence consider our results as satisfactory with respect to fulfilling the real-time requirements of Decentralised Service Location in the presence of adversary nodes.

4.5 Related Work

There exists quite some work on various DHT security challenges (see also 2.1.4). Here we survey previous work with focus on DHT lookup availability (the scope of our work in this chapter). In particular, we examine existing approaches for increasing lookup availability in Chord, identify their weaknesses and methodological flaws, and highlight the differences of these approaches in relation to the algorithms we presented.

In addition, we present a survey of approaches to increase lookup availability in DHTs other than Chord. Due to DHT routing specifics, these approaches are not directly applicable to Chord and therefore not easy to compare with our proposed solutions. Nevertheless, we regard these works to be important related work in the field of DHT security.

4.5.1 Approaches for Lookup Availability in Chord

4.5.1.1 Existing Proposals

Marti et al. : Marti et al. use an external, existing social network among users (e.g. *Facebook*) to increase the probability that a lookup path contains only non-adversary nodes in Chord [167]. The underlying assumption is that the probability of a DHT-node n_i being malicious is proportional to the number of hops between the query node's user and n_i's user in a social network. The authors present an extension to Chord called *SPROUT (Social Path ROUTing)*. With SPROUT, nodes in the DHT have additional routing links between each other if their users are connected (and online) in a given social network. Essentially, these additional social network based links are preferred in DHT routing as much as possible to increase the probability that a path contains solely non-malicious nodes. If no social link exists, the query proceeds with regular Chord routing. In [167], SPROUT is evaluated through simulation with random lookups. The metrics for assessing the benefit of the proposed algorithm are a) average path length, and b) path reliability. Path reliability is computed using a linear trust function and gives the probability that a random path contains only non-malicious nodes with respect

to a specific trust metric which is assigned to social links. It is shown that SPROUT can outperform regular Chord in both metrics. However, in their evaluation of SPROUT, the authors do not consider key replication nor lookups which consists of multiple paths.

Danezis et al. : Danezis et al. use a weak form of a social network, the so-called *bootstrap graph*, to improve lookup performance in a Chord network under attack [106]. In addition to regular Chord routing tables, each node in the DHT maintains a network introduction graph, i.e. the *bootstrap graph*. Two nodes n_i, n_j are connected in this graph if one node joined the DHT via the other node (i.e. n_i joined the DHT by connecting first to n_j). The bootstrap graph is typically a tree for DHTs [106]. Each node builds its own local bootstrap graph as a side effect of normal lookup operations and stores for each node in its routing tables T_r, T_s also the path on the bootstrap graph to these nodes [184]. Further, Chord is extended such that on each hop, the queried node returns its direct successor, its direct predecessor, all entries in its routing table T_r, and all direct connections it has in the bootstrap graph. In addition, on each hop the queried node returns for all these links the paths to these nodes on the bootstrap graph. The query node can thus select from alternatives when choosing the next routing hop.

Based on the bootstrap graph, several backtracking routing techniques are investigated, each exploring *alternate* routing paths. The goal of all strategies is to balance how many times a single node can be on different alternate routing paths for a single lookup (referred to as *diversity routing*). In other words, diversity routing tries not to put too much *trust* on a single node so that nodes are not disproportional often represented on the paths of the queries [184]. During a lookup, the query node maintains a histogram (called *trust profile*) of how many time each node was *traversed on the bootstrap graph* during the lookup so far. Based on these histograms, on each hop precisely the next hop node is selected which would increase only small trust profiles among all nodes in the histogram. The rationale behind this strategy is that adversary nodes which have bootstrapped many other adversary nodes will gain high values in the *trust profile* because they will be on many bootstrap graph paths. To ensure that queries make progress, two ways of combining *diversity routing* with regular Chord routing are explored[8]. With *mixed routing*, on each hop the next node is selected based on a weighted sum between closeness to the key on the DHT ring and diversity as described above. With *zigzag routing*, regular Chord routing and diversity routing are alternated on each routing. In summary, *zigzag* seems to be the best strategy with respect to the number of routing hops.

Artigas et al. : Artigas et al. present an approach for hierarchical DHTs called *Cyclone* [48]. In Cyclone, a single ID-space is used to connect several individual DHTs to one hierarchical meta DHT. The ID space consists of a prefix and a suffix, where the prefix denotes the ID for each node within its particular DHT and the suffix denotes the ID of the particular DHT the node belongs to (i.e. the *cluster-ID*). Suffix IDs are ordered bitwise right to left to form a hierarchy among DHTs. In this way, Cyclone combines

[8]For an in-depth description and thorough analysis of this approach, the reader is referred to [184].

a hierarchical approach with a flat DHT routing design: Several single DHTs can be combined by assigning a suffix ID (cluster-ID) for each DHT (according to the desired hierarchy) and maintaining the local DHT IDs as prefix IDs.

In [49], Artigas et al. apply Cyclone to increase lookup availability by enhancing Chord with multipath routing. A regular Chord DHT gets split into multiple disjoint DHTs using so-called *equivalence classes*. Two nodes n_i, n_j are in the same of e equivalence classes (where e is a system parameter) if the remainder of their node-ID modulo e is the same. A DHT gets hence split into e disjoint clusters, and the rightmost $log_2 e$ bits of each node ID form the Cyclone suffix ID. Except on the first and last routing hop, nodes only route within their equivalence class. The query node can start independent routing paths by selecting a node of each individual equivalence class $\in 0, \ldots, e-1$ from its list of direct successors T_s. At the last routing hop, again the direct successor list T_s is used to find a node n_e where $\text{suffix}(n_e) = \text{suffix}(k)$, i.e. to find $root_k$. In [49], the authors show by simulation that Cyclone on top of Chord can increase lookup success significantly compared to regular Chord. However, in their simulations they do not consider the case where $root_k$ is malicious; any path which reaches $root_k$ is considered successful. Note also that e independent routing paths can only be guaranteed in the case where the node identifier space is fully utilised; otherwise Cyclone can only achieve *at most* e (the number of equivalence classes) independent routing paths (see 4.5.1.2 for details).

Kapadia and Triandopoulos: *Halo* [146] is a probabilistic approach which does not change regular Chord-routing at all. Instead, multiple (not necessarily independent) lookups are started at the query node for the so-called *k-knuckles* for a key k. These *knuckle* nodes, if non-malicious, are nodes that should have a link to $root_k$ in their routing table T_r. All these knuckle nodes are then queried for the first node in their routing table T_r that succeeds k to obtain several candidates for $root_k$. The lookup succeeds if at least one of the knuckle nodes is not an adversary. By directly routing to candidate root nodes, this approach from Kapadia and Triandopoulos avoids routing pass the shield node. However, the case that a root node itself is malicious is not considered. Simulation results under these assumptions (i.e. the root node not being malicious) show that *Halo* can achieve a 90% success rate in networks with an attacker rate up to $f = 0.3$ [146].

Other approaches: *Myrmic* is a routing security extension to Chord [282]. It features a *root verification protocol* such that any node can verify if a particular node is the correct root node for a given key. However, *Myrmic* relies on a central authority called *Neighbourhood Authority (NA)* to assert that nodes are responsible for certain parts of the keyspace. Thus, *Myrmic* cannot be considered a fully decentralised approach.

Harvesf and Blough present a replica placement scheme for Chord which can provide independent lookup paths to different replicas roots [128]. However, they show analytically that their approach can guarantee a certain number of disjoint paths only for the case that a network has a fully utilised ID-space, which is a very unrealistic scenario (see further 4.5.1.2). Further, simulation results are only provided for small networks (i.e. 1024 nodes) with a rather small cardinality of the ID-space (i.e. $I = 2^{20}$) which implies a

rather small ID-space utilisation in the experiments.

Jaing et al. present *Bi-Chord*, a modification of Chord's unidirectional routing scheme to a multidirectional routing [142]. Essentially, *Bi-Chord* adds to regular Chord a second, counter-clockwise routing path for each lookup. As such, *Bi-Chord* merely provides two independent lookup paths for a given key from each node in the network. It can hence not increase lookup availability significantly.

Needels and Kwon present extensions to Chord that are similar to ours [186]. Their enhanced Chord routing uses backtracking and a routing table density check regarding T_r on every hop. Only premature simulations on small networks (with a fixed size of $N = 1000$ nodes in all experiments) have been conducted. Further, node lookup and not key lookup has been simulated (even worse: lookups that reach a malicious root node are considered *successful*). These facts make the results published in [186] not very useful; it is questionable to what extent the presented routing extensions can provide lookup availability in realistic scenarios.

4.5.1.2 Methodology Flaws in Existing Approaches

During our examination of existing research proposals for DHT lookup availability in Chord, we discovered several methodological flaws. Here we detail these findings and discoveries, highlighting why the specific simulation methodology used leads to misleading or inaccurate results[9].

Simulating only fully utilised ID-space To demonstrate the effectiveness of Cyclone for increasing Chord lookup availability, in [49] simulation results are presented which are based on a very strong (and completely unrealistic) assumption: a fully utilised node-ID space I. Without a fully utilised node-ID space I, Cyclone cannot guarantee that the query node n_q has a link to a node of each individual equivalence class $\in 0, \ldots, e-1$ in its list of direct successors $T_s(n_q)$. Similarly, at the last routing hop, there may not be a node n_e in the direct successor list T_s where $\text{suffix}(n_e) = \text{suffix}(k)$. Thus, in the case the node-ID space is not fully utilised, Cyclone cannot guarantee the existence of e independent paths, but only potentially achieve at a maximum e independent paths for each lookup.

Note that for DHTs usually $m = 160$ and $|I| = 2^m$. Hence, a DHT node-ID space is normally envisioned to have a size of $|I| = 2^{160}$ (if for instance *SHA-1* is used as the hash function). The number of nodes in the DHT is bound to the node-ID space but most likely significantly smaller. Today, P2P networks in the range of 1,000,000 nodes are considered very large. Considering $2^{20} = 1,048,576$, even such a large P2P network

[9] In addition to these methodological flaws, we found several implementation faults in the simulator used to obtain the results published in [106]: First, there is no use of a pseudo-random-number generator to initialize different simulation runs with a seed; second, key algorithms have been implemented with some bugs, making the obtained simulation results questionable. For a detailed analysis regarding these faults the reader is referred to [184].

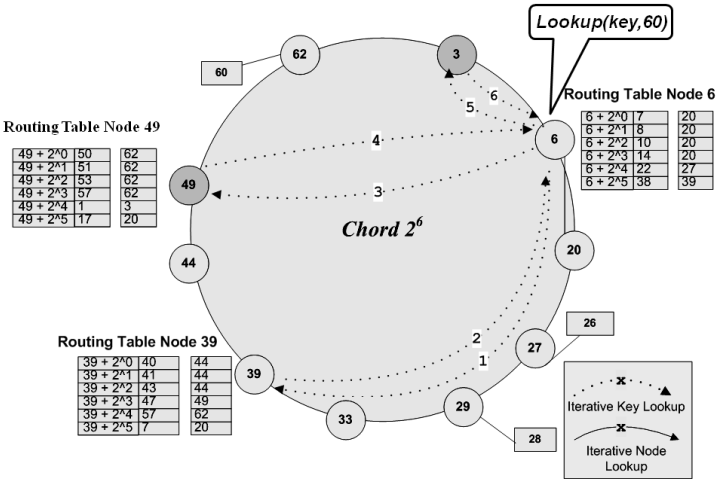

Figure 4.11: Key Lookup in Chord

would have a utilisation degree of $\frac{2^{20}}{2^{160}} = \frac{1}{2^{140}}$ which is very low. We conclude that for most DHT applications, $N << |I|$.

By simulating exactly the unlikely setting of a fully utilised ID-space, the results obtained in [49] are not applicable to the general case of an arbitrarily utilised node-ID space. More importantly, the simulated scenario is a very specific and peculiar case, in which, however, Cyclone can achieve best results for DHT lookup availability. The results published in [49] can therefore be regarded as very misleading. Similarly, an instance of this methodological flaw has been applied in [128], where the proposed replica placement scheme has only been proven to produce a given number of independent paths in case of a fully utilised ID-space.

Node-ID lookup vs. key-ID lookup We note the difference between simulating *node lookup* and *key lookup*. Simulating node lookup means that a query node always knows precisely the destination node when starting a lookup. In other words, not a lookup for a key is simulated, but simply the sending of a message from one node to another via DHT routing. On the other hand, key lookup means that the query node only knows the key when initiating a lookup. Clearly, simulating key lookup models the reality of a DHT better than node lookup. In reality, a querying node only knows a key but not the root node for a key. In addition, for key lookup the shield problem comes into play. By contrast, a node lookup can succeed without passing the shield: a node on the path which has the destination node in its routing table can route directly to the destination. For key lookup, this is not possible because it is *not* known by a node on the path which node is the root for a given key except for the shield. Thus, the success rate for *key lookup* is the metric we suggest (and consequentially use) for comparing different lookup algorithms.

Example 4.2: *As an example comparing key lookup and node lookup, consider figures*

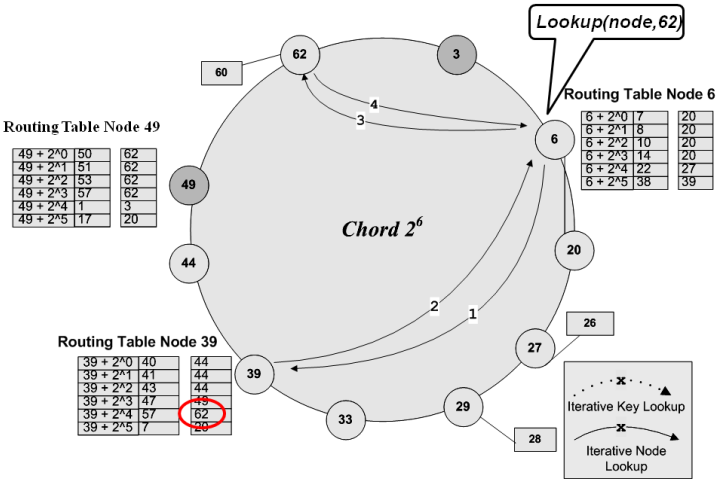

Figure 4.12: Node Lookup in Chord

4.11 and 4.12. Figure 4.11 shows a Chord network with regular key lookup. Figure 4.12 shows the same network with node lookup. In the example, n_6 *looks up key* 60 *(key lookup, figure 4.11), or* n_{62} *(node lookup, figure 4.12), respectively. It is obvious that node lookup does not have to pass* $shield_{60}$*. This is because* node lookup *can route directly to the destination node once a node has a link to the destination node in its routing table (node* 39 *in the example). In contrast,* key lookup *must pass* $shield_{60}$ *(*n_{49} *in the example) on the path to key* 60*.*

In [106], *node lookup* has been simulated, essentially meaning that not regular DHT lookups have been modelled. For key lookup, the most effective algorithm proposed in [106], *zigzag* routing, will not always succeed. This is even true if there is always a *good* path in the bootstrap graph between any two good nodes. The reason for this is the *shield problem*: Even if there is a path consisting of only good nodes in the bootstrap graph, key lookup has to pass the shield. Hence, a lookup may still fail if the shield node is adversarial. We verified this fact through simulations, showing that *zigzag* routing indeed cannot effectively circumvent the shield problem and thus not increase lookup availability significantly compared to regular Chord routing (see 4.5.1.3 and [246]).

In summary, it is unrealistic to simulate node lookup because a query node does not know $root_k$ for a lookup of a given key k. Thus, node lookup gives simplified results when showing the effectiveness of the *zigzag* routing algorithm. We emphasize that the results published in [106] are correct under the assumptions made. Also, our simulations [246] agree that this is an effective algorithm. However, this algorithm is only effective in decreasing the hop count instead of increasing the success rate. Regarding lookup availability, the simulations in [106] using only node lookup miss one important aspect of the Chord protocol, namely the shield problem.

Varying network size in simulations In a Chord network the size of the node-ID space I is 2^m; every node maintains m routing table entries. Although every node

maintains links to m nodes, out of these m nodes there are usually only $d \leq m$ distinct nodes in each routing table (compare with Figure 2.7). It can be shown that in a network of N nodes each node only has about $d = \log N$ distinct routing table entries [164]. Thus, in a network of N nodes, each node knows only about $\log N$ other nodes. As the network size increases, the ratio between distinct nodes in routing tables and network size ($\frac{d}{N}$) gets smaller. Hence, with increasing network size, the knowledge of a single node in relation to network size decreases. This is the reason why the average path length ϑ grows as N increases. The probability of all nodes on a path being good is $(1-f)^{\vartheta}$ [84]. Thus, with increasing N the probability of a path consisting of only good nodes decreases. Therefore, it should be harder to find such a path and consequently the success rate must decrease when the network size increases.

Consequently, any proposed solution for increasing DHT lookup availability should be analysed and simulated with varying network sizes. In the publications of Danezis et al. [106] and Artigas et al. [49] only results for one particular combination of network size and attacker rate f are presented. Additionally, some of the networks simulated in [106] are very small (100–500 nodes).

Contrary to network size, the size of the ID-space does not affect simulation results. Even if the ID-space is not fully utilised ($N < |I|$), d only depends on N. Thus, the ratio between distinct links and network size stays the same for constant N, independent of variations in $|I|$.

4.5.1.3 Progress with Respect to State of the Art

The approach closest to ours is Cyclone [48] [49], because it is an extension to Chord which tries to enhance Chord with independent multipath routing. Indeed, Cyclone can guarantee multiple independent paths in Chord, but only in the very special case where the ID-space is fully utilised. Also, the model used to generate the success rate results published in [49] does not consider the possibility of root nodes being adversary: each time a lookup has reached a root node, the lookup is considered successful. Compared to Cyclone, our solutions are beneficial in any network, independent of ID-space utilisation. In addition, our work differs from the one in [48] [49] as we use iterative routing (which allows the detection of node-ID suppression attacks), and our solution allows to directly route to replicated content in case the root node is adversary.

Contrary to our work, none of the attacker models in [146], [106], [167], or [49] consider the case where the node responsible for storing content, i.e. the root node, itself is adversary. Further, the attacker models in [106] and [167] also do not consider the case where the root node's predecessor, i.e. the shield node, is an adversary node. *Halo* [146] elegantly routes passed the shield node, but it cannot route to replica roots and hence fails if the root node for a given key is malicious. Consequently, state-of-the-art solutions cannot significantly increase lookup availability in Chord under realistic assumptions. We explicitly consider the cases where the root node, the shield node, or both are adversary nodes in our model. Moreover, we provide techniques to mitigate lookup failures in these situations based on realistic assumptions.

Unlike our approach, the approaches in [106] and [167] demand the existence of some sort of a social network to increase lookup availability. Essentially, this implies that query nodes rely on additional information about routing hop nodes—which is provided by other nodes—to make routing decisions. Instead, we consider a distributed system where nodes only rely on themselves for DHT routing decisions. With our approach, query nodes only use regular DHT routing information to select other nodes as routing hops. Nonetheless, approaches based on social networks can be considered complementary to our approach and such techniques can potentially be used as add-ons to our algorithms (given the existence of a social network).

Finally, in contrast to other Chord extensions, our approach enables the detection of node-ID suppression attacks on every routing hop in Chord. The advantage of this technique is to identify attacker nodes early on a path, which prevents the query node from spending resources on exploring such paths.

A quantitative comparison of results between our work and other proposals is difficult for several reasons. First, a different attacker model than ours has been used in most other publications. Second, often different scenarios or lookup algorithms (e.g. node lookup) have been simulated. Moreover, the various parameters and designs of individual algorithms make it hard to agree on comparable settings. Finally, some of the existing work uses other metrics than lookup success rate to evaluate the presented approach. We argue that a quantitative comparison with our work is not needed, as we have shown qualitatively that the main competing approaches are flawed and cannot solve the shield problem in realistic scenarios. However, in the cases of *Cyclone* [49] and *zigzag* [106]—which are both close to our work in certain aspects—comparing the effectiveness with our approach quantitatively is possible to some degree.

The quantitative results for Cyclone's lookup success rate published in [49] refer to the ideal, best imaginable case for Cyclone (a fully utilised ID-space). For our algorithms, we simulated a different scenario with a sparsely utilised ID-space. Further, the results published in [49] are not directly comparable to our results because a different attacker model underlies these experiments: In [49], root nodes are always considered as non-adversarial. Nonetheless, a somewhat imprecise comparison with our algorithms regarding the achievable lookup success rates can be done by multiplying the lookup success rates published in [49] with $1 - f$, to analytically account for the fact that the root node can be adversary with probability f in our model. Table 4.5 compares[10] the best results from [49] (for the largest network size, $N = 2048$) with our simulation results for *MRR-rd[2.0]* (in a network of $N = 2000$ nodes). The results show that with less overall hops, our solution can achieve a much higher lookup success rate for a comparable network size. We emphasise that this comparison is not exact nor precise due to the differences in the underlying assumptions, attacker models, and experiment scenarios used in our simulations and the ones conducted by Artigas et al. in [49]. However, the results for Cyclone in Table 4.5

[10]The results published in [49] have been obtained by executing eight lookup paths, each with a hop count limit of five. The overall hop count t_h for lookups has thus been 40 in the experiments in [49]. The graphs in [49] do not enable to read off the exact values obtained with the experiments. The values for ρ for Cyclone with potentially adversary root nodes in Table 4.5 have been calculated by reading off approximate results from the graphs in [49] and multiplying this value with $1 - f$.

	N	f	t_h	ρ
Cyclone	2048	0.2	40	≈0.77
MRR-rd	2000	0.2	25	0.99
Cyclone	2048	0.3	40	≈0.56
MRR-rd	2000	0.3	25	0.96
Cyclone	2048	0.4	40	≈0.32
MRR-rd	2000	0.4	25	0.79

Table 4.5: Lookup Success Rate for *Cyclone* Compared to *MRR-rd[2.0]* for Selected Attacker Rates

refer to the optimal scenario for Cyclone (a fully utilised ID-space); in realistic scenarios, the obtained success rate is likely to be even lower.

We have re-implemented the algorithms from Danezis et al. (published in [106]) in our simulator. This enables to evaluate the effectiveness of the best algorithm according to the results from [106], *zigzag* routing, under *key lookup*. Further, it enables a fair comparison with our algorithms under the same scenario and attacker model. Simulating *zigzag* routing for key lookup, we verified that—due to the shield problem—*zigzag* routing only achieves a marginally higher lookup success than regular Chord. Thus, *zigzag* is not much help against the shield problem in Chord. Figure 4.13 exemplarily shows ρ for various attacker rates, comparing *zigzag* with *MRR-r* for different network sizes. These results demonstrate that *zigzag* cannot achieve a reasonable lookup success rate. More results for *zigzag* with key lookup that we have obtained have been published in [246] and [184].

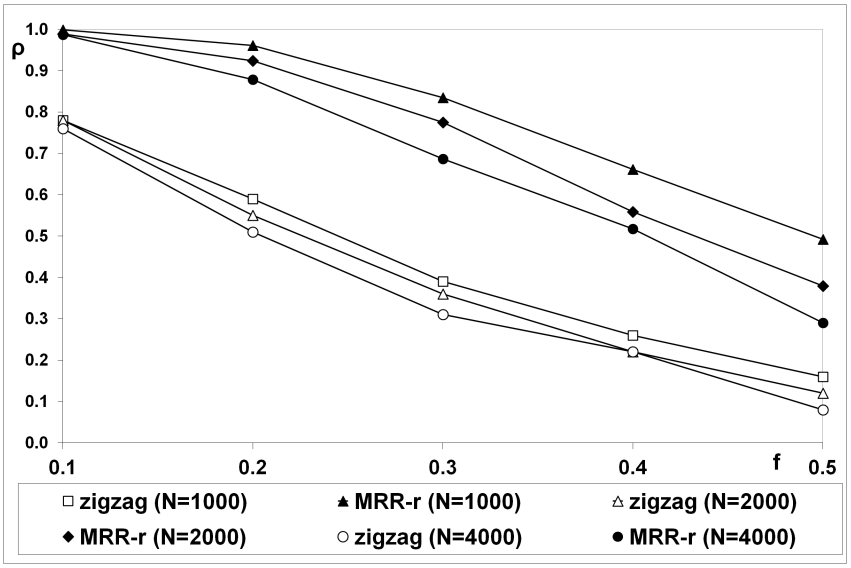

Figure 4.13: Comparing the Success Rate of *Zigzag* Routing with *MRR-r* for Various Attacker Rates and Different Network Sizes

4.5.2 Approaches for Lookup Availability in Other DHTs

Each DHT has its unique routing structure and algorithms. It is thus difficult to compare secure routing extensions that have each been proposed for a particular DHT among each other. This is especially the case since each DHT has its own security weakness and the extensions usually target the mitigation of these particular weaknesses [278]. Below we provide a summary of the main works regarding lookup availability that have been proposed for DHTs other than Chord. If possible, we highlight the main differences of these approaches (other than not being applicable directly to Chord) compared to our work.

Several works propose extensions to Pastry (and the related DHT Tapestry), trying to mitigate attacks on DHT routing, thus trying to increase lookup availability. Castro et al. propose secure routing techniques for Pastry [84]. They suggest constrained routing tables against routing table poisoning. In addition, they design a recursive multipath routing technique which explores alternate (but not necessarily independent) routing paths. Finally, they propose a density check technique specifically designed for recursive routing in Pastry where a candidate root node's neighbour list density is compared to the query node's neighbour list density. Since Pastry is a multidimensional DHT and in several other aspects different from Chord [158], these techniques are not directly applicable to Chord. Our work is fundamentally different in that we propose *iterative* routing and techniques that ensure *independent* paths. Further, we specifically target the shield problem which is unique to Chord. Finally, our density check technique for Chord is different because it is used at every iterative routing hop and compares the average distance among nodes in T_s between the current hop node and the query node[11].

Hildrum and Kubiatowicz propose iterative routing techniques for Tapestry and Pastry [130]. Their solution relies on a service that measures network proximity on the underlying network. At each routing hop, several nodes which are closest from this network proximity perspective are chosen. The underlying assumption is that adversary nodes cannot easily forge network distance on the underlying network, and thus cannot force routing to adversary nodes. The algorithms introduced in [130] essentially make Pastry's routing tables more constrained (based on network proximity). Chord features constrained routing tables per default. The proposed solution is thus somewhat specific to DHTs with unconstrained routing tables like Tapestry and Pastry.

Ganesh and Zhao investigate what they refer to as *identity theft* attacks for Pastry and Tapestry [118]. With such an attack an adversary node tries to spoof the identity of the root node for a given key. Such attacks are possible in Pastry and Tapestry because key responsibility is merely based on closeness in the ID-space and not on other constraints. As a countermeasure to such attacks, Ganesh and Zhao propose a technique called *Existence Proofs*, where special *proof manager* nodes sign certificates which certify the existence of nodes in some range of the node-ID space. Root identity spoofing attacks

[11] Castro et al. briefly mention a similar density check technique for iterative routing in [84]. However, they suggest comparing the average distance between nodes in the query node's neighbour set with the average distance between routing table entries at an intermediate hop to a *constrained* point p in the node-ID space.

are harder in Chord, as the root node has to have an identifier not only close to the key, but also succeeding the key in the Chord ring. In our work, we assume that the DHT application can detect such attacks through self-certifying data items. In addition, our *Direct Replica Routing* algorithms provide Chord with the ability to route to replica roots in the face of such attacks.

Baumgart and Mies propose extensions to Kademlia that extend Kademlia's routing algorithm such that multiple paths are independent [65]. Note that although Kademlia has a unidimensional routing structure [158], it uses multiple parallel routing paths per default. The extensions introduced in [65] hence merely extend Kademlia such that d disjoint (i.e. independent) routing paths are used. Although the goals are similar to our work, the proposed solution is very specific to Kademlia and not applicable to Chord. Also, the obtained lookup success rates are hard to compare with ours since the results are measured based on the number of disjoint paths and not on the total number of routing hops. More importantly, Kademlia's routing is fundamentally very different from Chord (e.g. routing table membership is symmetric in Kademlia and each routing table entry contains multiple links [65]) so that a comparison of the results seems unjust.

4.6 Summary and Contribution

The work presented in this chapter concerns lookup availability in Chord networks where adversary nodes are present. Our main contributions are the following: We presented a detailed analysis of previously existing approaches for increasing lookup availability in Chord, detailing the individual insufficiencies of these solutions. As a consequence, we proposed novel algorithms to address the problem of lookup availability in Chord. Further, we evaluated the effectiveness of these new solutions analytically and through simulations, showing that they can significantly increase lookup availability in Chord under realistic assumptions.

We have examined existing approaches for lookup availability in Chord, which are mainly the works of Danezis et al. [106], Artigas et al. [48] [49], and Marti et al. [167]. We discovered several flaws in evaluation methodology of these approaches: In the approach of Danezis at el. [106], *node lookup* has been simulated which does not consider the *shield problem* in Chord. Moreover, the results presented in [106] do not vary the network size. The approach proposed by Artigas et al. [49] has only been evaluated with a fully utilised node-ID space which is an unrealistic scenario for most DHT applications.

Further, the attacker models of these existing works do not consider the case where the root node itself is adversary. In addition, only the work of Artigas et al. [48] [49] provides a solution for the shield problem; however, the proposed solution can only address this problem in extremely unrealistic scenarios. Consequently, existing approaches cannot satisfactorily increase lookup availability in Chord when considering an attacker model where root node or shield node can be adversary nodes.

In summary, our analysis of prior work revealed that several of these approaches

contain methodological flaws. Further, none of the existing approaches can sufficiently—i.e. under realistic conditions and assumptions—address the shield problem in Chord or the case where the root node is adversary. These solutions can thus not achieve high lookup availability. This was our motivation to design novel algorithms for addressing the problem of lookup availability in Chord without the aforementioned deficiencies.

Based on analytical observations, we presented algorithms which enhance Chord to increase lookup availability. Our proposed algorithms extent Chord routing with multipath routing (exploring either independent or alternate routing paths), direct routing to replica roots, and mechanisms for detecting node-ID suppression attacks. Together, these algorithms can provide reasonable resilience of the DHT's lookup service against attacks on the DHT-routing layer.

We evaluated our proposed algorithms analytically by deriving upper bounds, to demonstrate that they can in principle achieve very high lookup availability in the presence of attackers. In addition, we showed through simulations that our algorithms can come very close to these theoretical limits. For instance, we can actually achieve a lookup success rate of 98 % in a network with 60 % adversary nodes. Further, we studied the effectiveness of our algorithms for different network sizes, with different hop thresholds, and with variations of the density threshold.

In summary, our results show that *Multipath Replica Routing* (*MRR*) can achieve a higher lookup success rate with independent restart than with backtracking in most cases. Further, using density checks on every hop can significantly increase lookup success as well as reduce the average hop count. For moderately infiltrated networks (i.e. $f \leq 0.3$), a density threshold of $t_d = 2.0$ yields the best results. In the network sizes we simulated, *MRR* using such a density threshold of $t_d = 2.0$ can virtually achieve 100 % lookup success for $f \leq 0.3$ (i.e. at least 97 % on average for $f = 0.3$) when a hop count threshold of $t_h = 50$ is applied as a limit for lookups to succeed. At the same time, the average hop count is less than 10 hops in the network sizes we simulated (i.e. in networks with up to 8000 nodes). We consider these hop counts acceptable for Decentralised Service Location (e.g. real-time communication session establishment) even for an Internet-scale DHT. Overall, we regard these results as satisfactory for providing lookup availability under timing constraints for low to medium attacker rates.

A quantitative comparison of our results with results from other publications is difficult due to different attacker models, diverse scenarios or lookup algorithms, and sometimes dissimilar evaluation metrics. Also, comparable settings are hard to agree on for the various algorithms. We provided two exemplary comparisons of lookup success rates where it was possible to some degree, showing that our approach can achieve higher lookup success rates than the works from Artigas et al. [49] and Danezis et al. [106] under our attacker model and a realistic scenario.

We have discovered several methodological flaws in previous work on lookup availability in Chord through a detailed analysis of existing approaches. Our theoretical analysis of Chord and the many simulations we conducted have further contributed in revealing potential pitfalls when simulating Chord with adversary nodes. We summarize our find-

ings in several propositions on characteristics of a proper Chord simulation methodology:

- *Do not simulate a fully utilised node-ID space*: In most DHT applications, it can be assumed that the number of participating nodes is several orders of magnitude lower than the size of the node-ID space, i.e. $N << |I|$. For generally applicable results, one must not simulate a fully utilised node-ID space.
- *Simulate key lookup to estimate the success rate*: The main primitive of a DHT is to lookup a key. Due to its unidirectional routing structure, the predecessor of the root node for a specific key can deny access to the key's data item in Chord. To show the effectiveness of a DHT routing algorithm in the presence of adversaries one must simulate *key lookup* instead of *node lookup*.
- *Use variations of the network size in simulations*: The probability that a path consists of only non-adversary nodes decreases with increasing network size. The network size must be varied in simulations to analyse the effectiveness of a DHT routing algorithm in the presence of adversaries.

We have conducted all simulations and obtained results according to these propositions. The reader is referred to Appendix B.1 for a description of the simulation methodology used for obtaining the results presented in this chapter and in Appendix B.1 itself.

Chapter 5

A Decentralised Mechanism for Integrity Protection of Location-Bindings

Contents

Due to the lack of a central authority, the verification of the integrity of content stored in a P2P network is a non-trivial task. However, without the possibility to verify the integrity of content stored in the network the service offered by such a network is of little use: Nodes cannot verify that messages they receive from the overlay have not been altered on some (overlay-) hop by an adversary node.

In this chapter, we present a decentralised approach to verify the integrity of data stored in a DHT[1]. We consider a P2PSIP network (see Chapter 3) as a prototypical example of DHT-based service location. We focus on attacks on the content stored in the DHT. More specifically, we look at the location bindings stored in P2PSIP networks and how to protect the integrity of these bindings. Our solution uses self-certifying data as content. This solution makes man-in-the-middle attacks on content stored in the network infeasible. Further, the authenticity of location-bindings can be verified by any entity in the network without relying on any kind of a public key infrastructure. To accomplish this, we introduce *self-certifying SIP-URIs*. We show how message-integrity can be achieved for P2PSIP with our solution. Finally, we discuss benefits and deficiencies of this approach.

The rest of this chapter is structured as follows: Section 5.1 presents the rationale for the approach and solution presented in this chapter. The proposed scheme is presented in detail in Section 5.2. Section 5.3 discusses the deficiencies and benefits of the chosen approach. Section 5.4 presents related work. The chapter concludes in Section 5.5 with a summary and a reflection on the contributions.

5.1 Rationale

5.1.1 Motivation and Goals

In Chapter 4 we presented techniques to provide availability of content stored in the network. However, even if the availability of content can be achieved (by added security measures) other attacks on Structured Overlay Networks are possible. For instance, a message that gets delivered successfully to its recipient could have been altered by the root node or by any other intermediate entity.

As an example, Figure 5.1 [239] shows a man-in-the-middle attack on content that is stored in a P2PSIP network (the binding of SIP-URI to a location). In the example recursive routing is used. Without authentication of messages, the requesting node (node 27) cannot verify the authenticity of messages it receives from the overlay. Any intermediate node on the overlay path can change the content of the message, thereby directing a phone call that follows the location lookup to a false location. In the example, node 156 acts as an adversary and exchanges the IP-address before handing over the message to the requesting node 27. When iterative routing is used, similar attacks can be conducted by an adversary root node: In that case, the root node can alter the data item it returns for a given key.

The attack model for a DHT depends on the application the overlay is used for. We focus on service location and P2PSIP as a prototypical example. In the case of P2PSIP, the integrity of messages has to be protected (in addition to availability): Otherwise,

[1]Parts of this chapter (including figures) have originally been published in [239]. See also Appendix A.

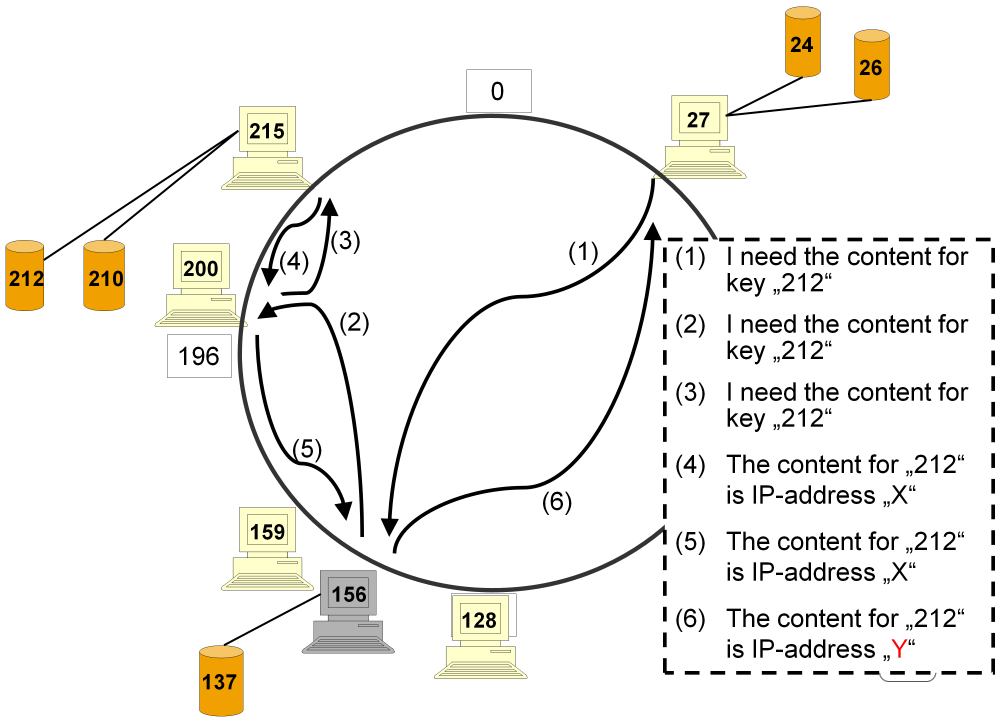

Figure 5.1: Man-in-the-Middle Attack on Content Stored in a P2P-SIP Network

attacks on the content stored in the network are possible, as shown in the previous example. The content stored in the network is the binding of SIP-URI (the identity of users in the system) and a location. Thus, it is of high importance to protect the integrity of the content in order to prevent impersonation of SIP-URIs by an attacker.

Our main goal in this chapter is to design a decentralised solution such that query nodes can verify the integrity of content they receive from the DHT. An additional objective is to have a means for query nodes to authenticate the originator of a data item stored in the DHT. We desire a decentralised solution which is secure but computationally feasible, i.e. based on cryptographic techniques which imply low computational overhead. Specifically, we intend to design a solution which fulfils the following requirements:

- *Fully decentralised and autonomous*: The verification of content's integrity shall not depend on a central authority. Further, integrity verification should not rely on the authority or opinion of other nodes, i.e. on other nodes' recommendations or assessments.

- *Secure*: Integrity verification shall be based on cryptographic techniques which can be considered secure according to state-of-the-art research.

- *Acceptable overhead*: In order to be usable for real-time communications such as P2P-VoIP, the verification process must be computationally feasible, i.e. incur low overhead for the query node.

5.1.2 Attacker Model and Integrity of Data Items

We consider the same attacker model as in the previous chapter (see 4.1.4): We assume secure node-ID assignment and a network that has been infiltrated over a certain period of time by adversary nodes. We look at the network at a certain state σ_i where $f \times N$ attacker nodes are distributed uniformly over the node-ID space I. Further, adversary nodes may collude and they may return false data items as a response to lookup requests.

We assume *store-once read-only* data items in the DHT. This assumption is reasonable as modification of data items in a DHT essentially boils down to storing new, updated data items (e.g. with newer timestamps, see 4.1.1). We thus do not consider mutable data. Further, we assume that storage of data items has happened before the network has been infiltrated by attackers; secure storage (and replication to replica root nodes) in face of adversary nodes is out of scope of our work. Note, however, that in principle also secure storage and modification of data items is possible in infiltrated networks [278].

In computer security, *integrity* can broadly be defined as "prevention of unauthorized modification of information" [121] or similarly "... that assets can be modified only by authorized parties or only in authorized ways" [202]. We consider only read-only data in our model (see above); authorized modifications are thus of no concern. We define the integrity of a data item stored in the DHT hence as follows:

> **Definition 11** (**Data Integrity**)**:** Data Integrity *is the property that a data item has not been modified since it has been stored in the DHT or during the storing process itself.*

In addition to data integrity, *data origin authentication* is a desirable property, i.e. being able "to verify the source and integrity of a message" [121]. Accordingly, we define data origin authentication as follows:

> **Definition 12** (**Data Origin Authentication**)**:** Data Origin Authentication *is a mechanism to verify the source and data integrity of a data item retrieved from a DHT.*

Note that integrity is an explicit integral part of data origin authentication. The approach we present in this chapter provides data origin authentication. It thus enables query nodes to detect if a data item retrieved from a DHT has been altered, and in addition to authenticate the source of the data item. We mostly refer to *integrity* in this chapter; if not explicitly highlighted, data origin authentication is an implicitly implied property.

5.1.3 Potential Solutions and their Drawbacks

To protect the integrity of content stored in a peer-to-peer network, one obvious solution is to add a central authority which certifies nodes (more precisely: node-IDs) in the network. Nodes could then sign content before storing it in the network. Verification of

these signatures would be possible for any node requesting content stored in the network through the added public key infrastructure.

A different approach—which has been suggested to secure P2P networks—is to use a reputation system [105] [168]. In a reputation system, so-called trust values are assigned to nodes based on prior behaviour observed by other nodes in the network. A node can decide whether to rely on information it receives from other nodes based on these values. In principle, a reputation system could be used by the query node in a DHT to determine if a root node for a given key can be considered as trustworthy (e.g. in the case that iterative routing is used and the data item for the key is retrieved directly form the root node).

We propose a different approach to authenticate content in Structured Overlay Networks: *self-certifying data*. The idea of self-certifying data is not new and has been introduced in several applications (e.g. [50], [103], [183] [1]). We suggest this approach for decentralised service location and specifically for P2PSIP because of the deficiencies we see in alternative options:

- **Trusted Authority for Certification:** A central authority which certifies identities limits scalability of the network. Also, a central authority would lower the ease of deployment of the network and add a single point of failure to the network. Thus, it diminishes much of the advocated benefits of P2P networks. Furthermore, known problems with certification infrastructures such as revocation of certificates, validation of identities to certify, and trust in the central authority would be inherited.

- **Reputation System:** Most reputation systems that have been introduced for P2P networks focus on traditional file-sharing applications, addressing the problem of selfish nodes[2] and not the prevention of malicious behaviour [168]. In addition, many existing reputation systems are not fully distributed: They rely on a central authority to store trust values. In general, reputation systems are not autonomous by their very nature. Further, they can only provide probabilistic security in the sense that it cannot be guaranteed that reputation scores are precise and correct.

5.1.4 Self-certifying Identities

The idea behind a self-certifying identity is that the ownership of the identity can be verified without relying on a trusted third party (e.g. a certificate authority in a public key infrastructure) [50]:

> **Definition 13** (**Self-certifying Identity**): *A* Self-certifying Identity *is an identity where ownership of the identity can be verified without relying on a trusted third party.*

[2] e.g. free-riding nodes which are using the services of a P2P network without delivering services to the network

This property can be achieved by representing the identity as the secure hash (using a secure hash function such as SHA-1 [28]) of a public key. Only the owner of the identity possesses the corresponding private key and can prove its ownership by signing the identity. More precisely, a self-certifying identity can be generated as follows:

1. Start with a cryptographic private/public key pair
2. Represent the identity as the hash of the public key
3. Sign the identity with the corresponding private key and append public key

Anybody can verify the signature for such an identity by

1. Checking that the hash of the public key is the identity
2. Verifying the signature with the public key

In the case of P2P-SIP, the identity stored as content in the network is the SIP-URI. Hence, for a self-certifying SIP-URI the URI must be generated as the hash of a public key. Our scheme is described in detail in the following section.

5.2 Self-certifying SIP-URIs

5.2.1 A Scheme for Protecting the Integrity of Content in P2PSIP

In order to protect the integrity of location binding updates, users can use self-certifying SIP-URIs. First, the user needs to create a public-private key pair[3]. Then she/he hashes the public-key and converts the hash into a valid SIP-URI. She/he can then sign any binding update to be stored in the network with the corresponding private key.

To check that a user location (IP-address and port) stored in the overlay indeed belongs to the specified SIP-URI, any node requesting a key lookup can verify the authenticity of the location-binding it receives from the network by doing two things: a) hashing the public key (which is sent along) to check that the public key indeed belongs to the SIP-URI and b) verifying the digital signature with the public key.

Notation: We use the following notation of public key cryptography to describe our proposed scheme: To express a digital signature on data item a with the private key of user b, $k_{priv}(b)$, we denote *"a is digitally signed with private key* $k_{priv}(b)$*"* as "$sign_{k_{priv}(b)}\{a\}$". Similarly, to express the verification of a digital signature on data item a with the public key of user b, $k_{pub}(b)$, we denote *"the digital signature on a is verified with public key*

[3]Below we use RSA [217], but in principle any asymmetric encryption system can be used.

$k_{pub}(b)$" as "verify$_{k_{pub}(b)}\{a\}$". Further, we denote storing a data item a in the DHT under key k with `Chord.Store`$(k; a)$, and the lookup of key k returning a as value with a=`Chord.Lookup`(k).

We can use this notation to describe how our scheme works in detail as follows:

1. Cryptographically Generating a SIP-URI

   (a) Assume a user intends to generate a self-certifying SIP-URI u_i. The user generates an arbitrary RSA [217] public-private key-pair, computing $k_{pub}(u_i)$ and $k_{priv}(u_i)$ as basis for the SIP-URI u_i she/he intends to create[4]:

   $$k_{pub}(u_i), k_{priv}(u_i) = \texttt{RSA.generate(seed)} \tag{5.1}$$

   (b) The user hashes the public key $k_{pub}(u_i)$ with a predefined hash function h_{sci} which is collision resistant (which implies that it is also second pre-image resistant [174]) to obtain an intermediate self-certifying identity for u_i, $sci(u_i)$:

   $$sci(u_i) = h_{sci}\left(k_{pub}(u_i)\right) \tag{5.2}$$

   (c) The user converts $sci(u_i)$ to an alphanumeric string using a predefined mapping function map_{uri} to obtain the (self-certifying) *user* component of her/his SIP-URI u_i, $user_{sci}(u_i)$:

   $$user_{sci}(u_i) = map_{uri}\left(sci(u_i)\right) \tag{5.3}$$

   (d) The user prepends the generated string $user_{sci}(u_i)$ followed by @ to her/his domain to generate her/his SIP-URI u_i:

   $$u_i = user_{sci}(u_i)@domain \tag{5.4}$$

2. Registering a Location for a SIP-URI

   (a) The user signs her/his current location (e.g. her/his IP-address and port) for u_i, $l(u_i)$, with the private key for u_i to obtain a signature for this location, $sig(l(u_i))$:

   $$sig(l(u_i)) = sign_{k_{priv}(u_i)}\{l(u_i)\} \tag{5.5}$$

   (b) Using the hash function of the DHT[5], h_{DHT}, the user computes the corresponding key-ID for u_i in the DHT, $k(u_i)$:

   $$k(u_i) = h_{DHT}(u_i) \tag{5.6}$$

   (c) The user stores the URI (u_i), the location $l(u_i)$, the signature $sig(l(u_i))$ and the public key for the URI $k_{pub}(u_i)$ in the DHT under the corresponding key-ID, $k(u_i)$:

   $$\texttt{Chord.Store}(k(u_i); \{u_i, l(u_i), sig(l(u_i)), k_{pub}(u_i)\}) \tag{5.7}$$

[4]Note that a user can have several URIs (e.g. for different roles in his personal life). This is expressed by the index $i \in \mathbb{Z}$.

[5]In principle, h_{DHT} and h_{sci} can be the same hash function, e.g. SHA-1 [28]. To avoid confusion, we differentiate the two hash functions with two different notations.

3. Verifying the Integrity of Location Data

   (a) A query node n_q requests a location for SIP-URI u_j from the overlay network. It computes the key-ID for u_j, $k(u_j)$, by applying $h_{DHT}(u_j)$ and uses this key in the DHT lookup. It receives the location binding, signature, and the corresponding public key from the DHT:

$$\{u_j, l(u_j), sig(l(u_j)), k_{pub}(u_j)\} = \texttt{Chord.Lookup}(h_{DHT}(u_j)) \tag{5.8}$$

   (b) The query node verifies that the public key, $k_{pub}(u_j)$, indeed belongs to u_j:

$$map_{uri}\left(h_{sci}\left(k_{pub}(u_j)\right)\right) @domain == u_j? \tag{5.9}$$

   (c) If the previous step results in a positive outcome (i.e. the truncated hash of u_j's public key is the user part of the SIP-URI u_j), n_q verifies the location signature with the public key:

$$\text{verify}_{k_{pub}(u_j)}\{sig(l(u_j))\} == l(u_j)? \tag{5.10}$$

   (d) If the verification is successful, $l(u_j)$ can be regarded as the correct, authenticated location for SIP-URI u_j by the query node n_q. If the verification is not successful, n_q cannot regard $l(u_j)$ as authenticated.

Note that the scheme above fulfils our goals from 5.1.1. First, the scheme is fully decentralised and autonomous as the query node can perform the verification of location-bindings it looks up in the DHT solely on its own, based on information stored in the DHT along with the actual data item for a key. Second, the scheme relies on asymmetric cryptography and a secure hash function. The security thus depends on the proper choice for these algorithms and the corresponding cryptographic strength these choices offer. One potential weakness with respect to the security that the scheme can offer is the conversion of the self-certifying identity to an alphanumeric string (i.e. the function $map_{uri}()$ in equation 5.3). We will discuss this issue below in 5.2.2.

Finally, our scheme has low overhead: For each lookup the query node has to perform one hash operation, one string-mapping, and one digital signature verification in addition to regular Chord operations. Similarly, a node storing a location binding needs to perform one hash operation, one string-mapping, and the computation of one digital signature. All these operations can be computed within milliseconds on today's hardware (even on devices such as VoIP hardphones or mobile phones) and thus make our scheme applicable to real-time communications such as P2P-VoIP.

5.2.2 Generating a Self-certifying SIP-URI

In our scheme, a private key is used to sign the binding of an IP-address to a SIP-URI (equation 5.5). To prove that this private key indeed belongs to the (owner of the) SIP-URI in the binding, the hash of the corresponding public key (equation 5.2) is encoded

bit-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
bit-2	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
bit-3	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
bit-4	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
bit-5	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
ASCII	a	b	b	d	e	f	g	h	i	j	k	l	m	n	o	p
bit-1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
bit-2	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
bit-3	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
bit-4	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
bit-5	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
ASCII	q	r	s	t	u	v	w	x	y	z	1	2	3	4	5	6

Table 5.1: Mapping of Hash-bits to ASCII Characters

within the SIP-URI. This conversion needs some discussion: First, the SIP-URI must be formed as specified in [222] in order to enable backwards compatibility with existing SIP-implementations. Second, the resulting SIP-URI shall not be too long, so that it can be typed into the interface of a user agent properly (even with limited user interfaces as in VoIP hardphones). Thus, a specified function is needed to convert the hash-value $sci(u_i)$ (equation 5.2) into a string (equation 5.3). This string then forms part of the SIP-URI (equation 5.4).

A SIP-URI is generally of the type `user@domain` [222]. The *user* part can consist of one or more ASCII-characters. In summary, these characters are limited to alphanumeric values (i.e. $a-z, A-Z, 0-9$) and some special reserved characters. Altogether, 79 different ASCII characters may be used for the user part of a SIP-URI (see [222], pp. 147 and [100] for details). Thus, it is possible to encode at most 6 bits in any ASCII-character of the user part of a SIP-URI ($2^6 = 64 \leq 79$).

We suggest the following algorithm to convert the hash of the public key into a string, assuming a 160-bit hash value is used (as with SHA-1, for example). This algorithm forms the mapping function $map_{uri}()$ in equation 5.3:

1. Use only the leftmost b bits of the hash value
2. From left to right, encode every 5 bits into one ASCII character according to a defined mapping table $t_{map-uri}$

We suggest using only lower-case letters and digits for the generated URIs. Although more characters are allowed in SIP, this prevents attacks—similar to common phishing websites—where users are tricked into a wrong URI because of case-sensitivity: An attacker could automatically generate public keys until she/he gets a value that hashes to a URI similar to the URI she/he wants to impersonate except for case-sensitivity. Thus, we only use 32 ASCII-characters (26 lower-case letters plus 6 digits). This results in encoding 5 bits into one character ($2^5 = 32$).

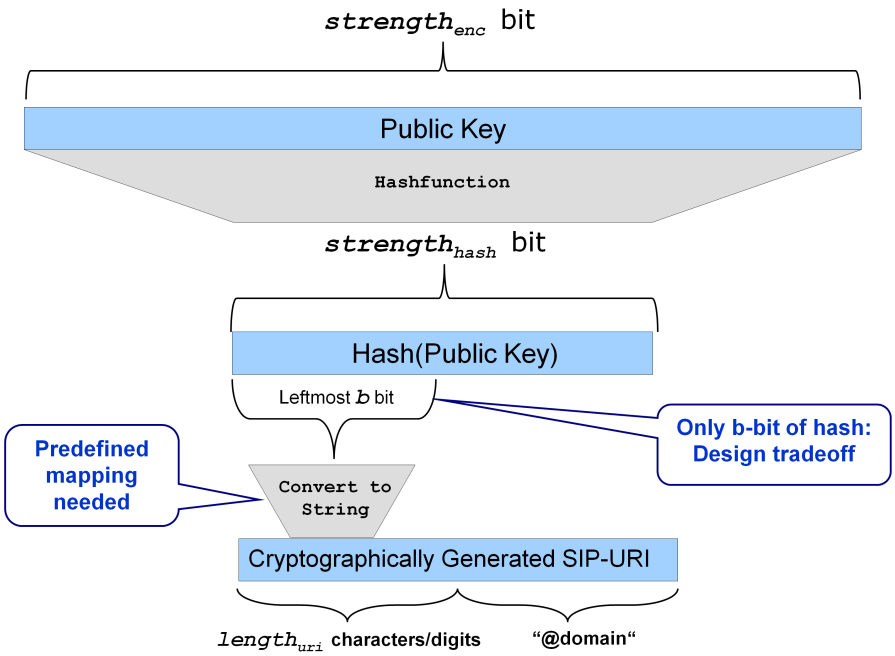

Figure 5.2: Procedure of Generating a Self-Certifying SIP-URI

Table 5.1 shows an example for a predefined table $t_{map-uri}$ which maps a hash onto ASCII characters. Figure 5.2 shows the general procedure of mapping a public key onto a self-certifying SIP-URI. First, the public key (of length $strength_{enc}$) is reduced to a $strength_{hash}$-bit hash value by a predefined hash function (h_{sci} in our scheme). Then, the function $map_{uri}()$ is applied to the leftmost b bit of the hash-value, mapping every 5 bits from left to right onto a character/digit. The resulting $length_{uri}$ characters/digits are prepended to the domain of the user, separated by the `@` sign.

In practise, one can assume a 2048-bit RSA public key ($strength_{enc}$=2048) and a hash value of 160-bit when using SHA-1 ($strength_{hash}$=160). When using, for instance, $b = 65$, the *user* part of the resulting SIP-URI is 13 characters/digits long instead of 11 characters/digits when compared to using all possible characters and encoding 6 bits into one character ($65 \div 5 = 13; 65 \div 6 \leq 11$). We feel that this is a reasonable trade off: the added security is worth the longer SIP-URI (e.g. two characters more in the case of $b = 65$).

In general, $strength_{enc}$, $strength_{hash}$, b, and the table $t_{map-uri}$ are design choices. Given a fixed table $t_{map-uri}$, $length_{uri}$ depends solely on b. The proper value for b involves a trade off: The smaller the value is, the more readable and memorable is the resulting SIP-URI for human users. For instance, with b=65, a self-certifying SIP-URI will look like `sip:k4h1gfhdh5wdd@sip-provider.de`. If, on the other extreme, $b = 160$ is chosen, the resulting SIP-URI has a *user* component of $length_{uri} = 32$ (since $160 \div 5 = 32$), and would thus look something like `sip:kdjs5ksgat2hdkfksh34nccsjaq2n1ee@sip-provider.de`.

At the same time, the smaller b is chosen, the weaker is the security provided by the

hash function against attacks. A successful second pre-image attack would enable an attacker to generate a public/private key pair where the hash of the public key would match the SIP-URI to be attacked. The larger b, the harder such attacks on (the effectively used part of) the hash function become. We discuss potential attacks on the hash function in Section 5.3.

5.3 Discussion

In this section, we evaluate our proposed scheme and the general approach. We examine imaginable attacks on the presented scheme and discuss potential countermeasures. Further, we contrast the benefits and drawbacks of our solution.

5.3.1 Potential Attacks and Countermeasures

Self-certifying SIP-URIs are susceptible to certain attacks. Below we discuss these attacks and outline potential countermeasures to prevent them.

Attacks on the hash function The security of our solution relies on some characteristics of the hash function that is used: The hash function must be collision-resistant and second pre-image resistant. Pre-image resistance is not important for our scheme because the public key gets sent along with its hash in each message. If at some time the second pre-image resistance of the hash function is broken (for example due to some new sort of attack), an attacker could generate a public/private key-pair with the same hash as the target URI of his attack. This would enable him to sign falsified messages. Since we only use the leftmost b bit of the hash, our SIP-URIs are possibly susceptible to second pre-image attacks. To circumvent this shortcoming, hash extension techniques can be used [52] [51] [50]: A second hash-value is used, where the m leftmost bits must be 0. Using this technique, the effective hash length can be incremented by m bits (see further Section 5.4).

Denial-of-Service attacks In our scheme, any node verifying the authenticity of a message must perform cryptographic primitives which consume resources at the node. Thus, in principle the network will be more susceptible to Denial-of-Service (DoS) attacks: An attacker can exploit this fact by inserting bogus messages into the network to attack nodes' availability. However, computing a hash function and verifying a digital signature is not a significant cryptographic computation. We thus regard the enhanced susceptibility to DoS attacks as minor. Moreover, our technique also prevents some DoS-attacks because each node in the network can detect messages with forged content and drop such messages immediately instead of forwarding them further (see 5.3.2).

A potential countermeasure against DoS attacks with bogus messages would be to use

a so-called *cookie* mechanism similar as in the *Internet Key Exchange Protocol (IKEv2)* ([148], pp. 17-19). When receiving a `Chord.Store`$(k(u_i); \{u_i, l(u_i), sig(l(u_i)), k_{pub}(u_i)\})$ request (equation 5.7), the root node, $root_{k(u_i)}$, would first send a *cookie* (i.e. a small, randomly generated token) to the IP-address and port, $l(u_i)$, before verifying $sig(l(u_i))$. Only after receiving an answer to the cookie message, $root_{k(u_i)}$ would begin to verify $sig(l(u_i)$. This would prevent DoS attacks where an attacker would use other IP-addresses than under her/his control in location-binding store requests. Still, an attacker could send a large amount of bogus `Store` requests from IP-addresses under her/his control, answering every *cookie* request from the root node correctly. To counter such attacks, a root node could employ a certain threshold of location-bindings it accepts from a unique location $l(u_i)$ or IP-address per time unit. Further, instead of sending a simple cookie, $root_{k(u_i)}$ could challenge the origin of a location-binding store request with a computational puzzle, demanding resource consumption per storage request at the sender (e.g. similar to the mechanism suggested in [138]).

Replay attacks Our scheme as presented in Section 5.2 is prone to *replay attacks*. An attacker can re-use an *old* location-binding, $\{u_i, l^{old}(u_i), sig(l^{old}(u_i)), k_{pub}(u_i)\}$, which she/he captured to direct calls to a location $l^{old}(u_i)$ which in fact is not valid anymore for the URI u_i. Since the signature verification for such bindings will be correct, any root node will accept such *replayed* `Store` requests. Similarly, query nodes will assume such bindings received from the DHT as correct since the signature verification will not fail. In case the attacker is currently in control of $l(u_i)$, he can direct calls to himself, resulting in an *impersonation attack* on u_i (the URI contained in the location-binding). Alternatively, calls are directed to a wrong location $l(u_i)$, resulting in a *DoS attack* on the URI u_i with respect to receiving calls.

Normally, if a user changes the location for URI u_i, $l(u_i)$, his node will update its location-binding in the DHT by sending out a fresh DHT `Store` message for the new location, $l^{new}(u_i)$:

$$\texttt{Chord.Store}(k(u_i); \{u_i, l^{new}(u_i), sig(l^{new}(u_i)), k_{pub}(u_i)\}) \tag{5.11}$$

In addition, the node will delete the old location binding by sending out a DHT Remove message:

$$\texttt{Chord.Remove}(k(u_i); \{u_i, l^{old}(u_i), sig(l^{old}(u_i)), k_{pub}(u_i)\}) \tag{5.12}$$

Note that—prior to its removal—an attacker could have easily retrieved the correctly signed location-binding $\{u_i, l^{old}(u_i), sig(l^{old}(u_i)), k_{pub}(u_i)\}$ for the old location, $l^{old}(u_i)$, from the DHT. In fact, an attacker can *harvest* location-bindings for any key $k(u_i)$ using regular DHT lookup primitives. To execute a replay attack on a URI u_i, the attacker will send a DHT `Remove` message as in (5.12) but with the correctly signed location-binding for the current location of u_i, $l^{new}(u_i)$ (which the attacker can easily fetch from the DHT). Then, the attacker will send out a DHT `Store` message as in (5.11) but with a correctly signed old location-binding for u_i, $l^{old}(u_i)$, assuming the attacker previously retrieved such a location-binding from the DHT.

Similar replay attacks on location-bindings are known for the *DNS Security Extensions (DNSSEC)* [45] [47] [46] where attackers can re-use signatures which have not expired [289]. The solution used by DNSSEC against these kinds of replay attacks is to use limited signature lifetimes [289] [47]. DNSSEC uses an *absolute time* to express the signature expiration (see [47] pp.8 for details). This limits the replaying of signatures by attackers to within the signature lifetime.

As countermeasures against replay attacks on self-certifying signatures, we suggest limited signature lifetimes as in DNSSEC. Note that such a mechanism implies the necessity of time synchronisation among entities in the system. We assume that nodes can synchronize their clocks[6], e.g. by using the *Network Time Protocol (NTP)* [177]. When signing a location-binding (compare equation 5.5), additionally an absolute time-stamp for the expiration of the signature, $exp(l(u_i))$, is signed:

$$sig(l(u_i)) = sign_{k_{priv}(u_i)} \{l(u_i)|exp(l(u_i))\} \tag{5.13}$$

where | denotes bitwise concatenation. When storing the location-binding in the DHT (compare with equation 5.7), additionally the expiration time for the signature, $exp(l(u_i))$, is stored:

$$\texttt{Chord.Store}(k(u_i); \{u_i, l(u_i), exp(l(u_i)), sig(l(u_i)), k_{pub}(u_i)\}) \tag{5.14}$$

Any node can verify the integrity of the expiration time using the signature $sig(l(u_i))$ and the public key $k_{pub}(u_i)$. Nodes should disregard expired location-bindings. This bounds replay-attacks to a limited period up to the expiration time.

Note, however, that in a fully decentralised system as P2PSIP without trust relationships among nodes, any node could *remove* a valid, existing location-binding for a URI u_i by issuing a DHT `Remove` message, as in (5.12). The `Remove` request would contain the current, correct location-binding for u_i which the attacker node can lookup in the DHT. A simple solution to prevent such attacks with forged `Remove` requests is to include the type of the DHT request (i.e. `Remove` or `Store`), req-type, in the signature of the location:

$$sig(l(u_i)) = sign_{k_{priv}(u_i)} \{l(u_i)|exp(l(u_i))|\text{req-type}\} \tag{5.15}$$

Generating a signature as in equation 5.15 limits replay attacks to the same DHT request type: An attacker could *not remove* a location-binding (prior to its expiration time), if she/he previously captured a signed `Chord.Store` message for u_i.

Another mechanism we suggest against replay attacks—in addition to limited signature lifetimes—is the use of nonces to authenticate the originator of `Store` and `Remove` messages, or in general to verify ownership of a SIP-URI. Consider a root node for key $k(u_i)$, $root_{k(u_i)}$. The root node can challenge any DHT `Store` or `Remove` request it receives by sending a randomly generated nonce, $nonce(l(u_i))$, directly (i.e. without DHT routing) to the location contained in the specific request, $l(u_i)$. Only if a message with a combined signature on $l(u_i)$, $exp(l(u_i))$, req-type, and $nonce(l(u_i))$, $sig_{nonce}(l(u_i))$, is returned and the signature is verified, $root_{k(u_i)}$ will regard the request as authenticated

[6] Attacks on time synchronisation are outside the scope of our work.

and accept it. The combined signature on location, expiration time, request-type, and nonce can be generated as follows:

$$sig_{nonce}(l(u_i)) = sign_{k_{priv}(u_i)}\{l(u_i)|exp(l(u_i))|\text{req-type}|nonce(l(u_i))\} \tag{5.16}$$

Signing the concatenation (and not just the nonce) ensures that an attacker could only reuse such a signature for the same combination of location, expiration date, DHT request type, and randomly generated nonce.

In case of recursive routing, in principle not only the root node but any intermediate node on the routing path could challenge *Store* and *Remove* requests before forwarding them. We regard such per-hop challenging as not efficient and too much overhead. It suffices if such replay attacks within the signature lifetime can be detected by the root node. Note, however, that even without challenging the sender, self-certifying identities still enable intermediate nodes to detect forged location-bindings and replay attacks with expired location-bindings.

If a root node is an adversary, it may accept any replayed `Remove` or `Store` request without challenge. In the case of `Remove` and subsequent `Store`, limited signature lifetimes (as introduced in equation 5.13 and 5.14) limit such attacks to the signature expiration date. Moreover, it requires an attacker which is in possession of a previous, correctly signed location-binding for the URI to be attacked. In the case of only `Remove`, this behaviour by the root node is not different from it not returning a data item for a key in lookup requests. It can thus be detected by a query node (see also Chapter 4).

Apart from integrity checks on DHT requests, any query node n_q can challenge the callee at a location n_q received from the DHT, $l(u_i)$. n_q sends a nonce, $nonce(l(u_i))$, to $l(u_i)$ before initiating the actual communication (e.g. sending a SIP `Invite` message). Only after n_q receives a message with $sig_{nonce}(l(u_i))$ (as in equation 5.16) which it can correctly verify, n_q starts the session establishment with $l(u_i)$. This way, a query node n_q can itself challenge the location for a URI, $l(u_i)$, with a nonce to counter replay attacks and to handle the case of an adversary root node (see above).

5.3.2 Notable Properties

Generally speaking, our scheme is independent of any network property and can thus be used as a security add-on in almost any scenario. Specifically, our proposed solution has the following benefits:

No central authority Most notably, our solution enables authentication of messages without the use of a central authority. Thus, it retains the spirit of peer-to-peer computing (no server), and scalability is not a problem.

Verification of message-integrity possible at all routing hops The verification-procedure can be done at any routing hop. Thus, not only the requesting node can verify

message-integrity: Any node in the network can verify the integrity of messages it receives and detect (and possibly drop) falsified messages. For instance, with recursive routing any node routing a storage or a lookup request could immediately drop a message which contains a forged location binding. Also—independent of recursive or iterative routing being used—a root node could decide not to replicate content at replica roots in case it receives a forged location binding for a key. Dropping messages with forged content can prevent Denial-of-Service attacks where attackers try to flood the network with bogus messages (see 5.3.1).

Independence of overlay and routing strategy The scheme works independent of the concrete DHT algorithm used. Chord has been chosen as the DHT algorithm for P2P-SIP [140], but there may be alternative implementations based on other DHTs. Therefore, it is important that any mechanism for protecting content integrity can also be used with a different DHT/overlay protocol. Also, our scheme can be used with iterative, recursive, or more sophisticated routing strategies [106].

Backwards compatibility Because the added cryptography is encoded within the SIP-URI, our scheme works with any existing SIP implementation and most likely with any future specification for P2P-SIP. No new headers or components are added. Hence, the scheme can also be used on top of any existing (Client-Server) SIP deployment.

5.3.3 Drawbacks

The proposed scheme prevents man-in-the-middle attacks on content in P2P-SIP networks: Using this scheme, any node in the network can verify the authenticity of a SIP-URI/location binding message it receives. However, the presented solution introduces some new—mostly practical—hurdles. In this subsection we list and discuss these issues.

Readability of SIP-URIs One of the most obvious disadvantages of the presented solution is the (un)readability of self-certifying SIP-URIs. As they are computed by converting the hash of a public key into a string, URIs will look cryptic to the user (e.g. `sip:k4h1gfhdh5wdd@sip-provider.de` for a design choice of $b = 65$).

Key revocation With self-certifying identities, a public/private key pair and an identity are securely bound to each other. If the private key for a self-certifying identity is compromised, key revocation essentially means URI revocation. Generating a new self-certifying SIP-URI is not problematic. However, circulating a new URI to contacts and informing them personally that the old URI is not valid anymore may be cumbersome to users.

Performance Because cryptographic functions are added to the nodes participating in the network, such a network may loose some of its performance. As highlighted previously, the cryptographic primitives necessary to verify the authenticity of content in our scheme demand very little computational resources from nodes. We therefore consider the performance degradation due to the introduced cryptographic operations as negligible.

Associating a SIP-URI with a user Because URIs are cryptic, there must be a way to reliably associate a user with a self-certifying SIP-URI. We suggest using existing authentication infrastructure available to users for this purpose: A certified web service using TLS [107] can publish an "online phone book", mapping users to SIP-URIs. The certificate of the service can be verified by users in their web-browser. In addition, users may use business cards or existing communication channels (such as encrypted email) to obtain a trustworthy association of an actual person with a SIP-URI. Note that without self-certifying SIP-URIs in regular Client-Server SIP the same problem exists: How can a user be sure that a SIP-URI indeed belongs to the person he intends to speak to, especially if the person is previously unknown to the user?

5.4 Related Work

Self-certifying IPv6 addresses (CGAs) One of the main applications of self-certifying identities has been network layer address generation for IPv6 [131]. Aura proposed to generate self-certifying IPv6 addresses in [50]. With this proposal, the *interface identifier* of an IPv6 address, i.e. the 64 least significant bits of an IPv6 address, are generated based on a cryptographic hash of a public key. This approach is generally referred to as *Cryptographically Generated Addresses (CGA)*. So-called *hash extension* techniques—as originally proposed by Aura and Roe [52]—enable to increase the difficulty of a brute force *pre-image* or *second pre-image* attack on the hash function while keeping the length of the hash output constant: An additional hash function, $h_2()$ is repeatedly computed on the public key, k_{pub}, with a changing, concatenated modifier, r, until the first m bits of this hash function's output are equal to zero. Then, the actual hash of the public key is computed, $h_1(k_{pub})$, but concatenating $r_{success}$, i.e. the value for r which resulted in an m-bit zero output of $h_2()$ (see [52] pp.7 for details). For successful verification, the first m bits of $h_2(k_{pub}|r_{success})$ must be zero and $h_1(k_{pub}|r_{success})$ must result in the self-certifying identity. Essentially, this technique increases the strength of the hash function by m bits. Earlier proposals for self-certifying IPv6 addresses [192] [181] lack this feature of being able to increase the cryptographic strength of the hash function at a constant bit-length and can thus only provide security equivalent to a 64-bit hash output (the length of the interface identifier of an IPv6 address).

A concrete mechanism for generating CGAs including hash extension techniques has been standardised [51] and is used by several Internet protocols such as *SEcure Neighbor Discovery (SEND)* [136], Shim6 [190], or *Mobility Support in IPv6* [144]. A security analysis of the significance of collision attacks on SHA-1 (e.g. by Wang et al. [284] [230]

[229]) for CGAs is provided by Bagnulo and Arkko in [56]. They conclude that not collision attacks, but rather second pre-image attacks on the hash function are the main security concern for CGAs. Bos et al. provide a detailed security analysis of CGAs and propose several improvements they refer to as CGA++ [73].

Self-certifying *YURLs* and *httpsy* scheme Close proposes self-certifying web-URLs called *YURLs* [95] [137]. A `YURL` is a URL as specified in [67] which contains the hash of a public key. Close also defines the `httpsy` scheme [96] for `YURL`s. Essentially, `httpsy` uses a self-certifying `YURL` as basis for authenticating a website (instead of a DNS hostname as in `https`): only a server which can authenticate itself with a private key that corresponds to the public key which is intrinsically encoded in the `YURL`, is accepted by the `httpsy` client.

In addition, Close proposes the use of so-called *petnames* [176] [97] to map cryptographic hashes as in a `YURL` to a human readable format. The user's browsers maintains a mapping of user-chosen *petnames* to `YURL`s. The browser uses the `YURL`/`httpsy` concept to authenticate web servers internally, but it displays these websites with *petnames* to the user.

TOR Hidden Services *TOR* is an anonymity network using *Onion Routing* [108]. TOR provides so-called *Hidden Services* which enable to offer any kind of service running over TCP (e.g. a web-server) without revealing the location (i.e. the IP-address) of the service [21]. TOR Hidden Services can only be accessed via the TOR network, and only by TOR clients. A pseudo top-level domain `.onion` is used to access TOR Hidden Services. Self-certifying hostnames are generated by hashing and truncating a public key onto 16 alphanumeric characters and appending the `.onion` domain [22] [21]. Any hidden service generates such a hostname, signs it with the corresponding private key, and publishes it at centralised TOR directory servers[7] [193].

To locate a hidden service in TOR, a TOR client needs to retrieve the self-certifying hostname of that service out-of-band (e.g. by using a special website). Then, it looks up this hostname (e.g. `fjsusjfl4lq9hdfj.onion`) at the TOR directory service. As a result of the lookup, the client retrieves not the location of the desired service, but the address of an *introduction point*. The client can verify the authenticity of this address due to the self-certifying property of the service's hostname. The introduction point is a TOR relay server chosen by the hidden service. The TOR client requests a connection with the hidden service at the introduction point, notifying it about its *rendezvous point*, a TOR relay server chosen by the client. Communication with the actual hidden service is established via the rendezvous point: Both the client and the hidden service establish a TOR circuit (i.e. a set of encrypted connections through relays on the network) with the rendezvous point individually.

[7] In principle, any key-value lookup service (e.g. also a decentralised one such a DHT) could be used [108]. However, such an external lookup service would allow clients to access services anonymously only if the lookup service itself is protected by TOR (which would not be the case for a DHT) [108].

Other applications of self-certifying identities The *Host Identity Protocol (HIP)* [183] uses the hash of a public key as the so called *Host Identity Tag (HIT)*. This HIT is a self-certifying identity in HIP. Since the HIT is the identifier used for host identification on HIP, the self-certifying property enables hosts to authenticate each other based on their identifier in HIP.

So-called *strong names* provide a form of self-certifying identities to software *assemblies* (i.e. compiled code libraries) in Microsoft's *.NET* platform [1]. In the .NET framework, an assembly's name consists of several parts, such as a filename and a version number. If the assembly name also contains a public key token, it is called a strong name [1]. Such a public key token is obtained by hashing an associated public key. By containing this token as part of itself, the assembly name becomes self-certifying: it contains the hash of an associated public key. The owner of the code can sign the assembly with the corresponding private key to protect the integrity of the software assembly which has a strong name.

Mazieres et al. present *SFS*, a file system which introduces *self-certifying pathnames* [172]. In SFS, file systems are addresses by a path `Location:HostID`, where `HostID` is the cryptographic hash of a public key. These self-certifying pathnames enable clients to verify the authenticity of servers' public keys solely based on the path of a file-system. Dabek et al. propose a *Cooperative File System (CFS)* in [103]. CFS is a distributed read-only file system based on the Chord DHT [267]. Files are stored in distributed blocks in the DHT. The root block of a file is signed with the owner's public key. The root block's Chord key is the hash of the owner's public key, making the DHT key for root blocks a self-certifying identity.

Baumgart proposes *P2PNS*, a decentralised naming service for P2PSIP [63]. This approach mainly addresses the problem of secure node-ID generation and node-ID assignment in a DHT. In P2PNS, a DHT `nodeID` is generated as the hash of a public key. A hash extension as proposed in [52] is applied to increase the hash strength while keeping the effective bit-length of the resulting `nodeID` reasonably small. Users can choose an arbitrary SIP-URI; a static binding between URI and `nodeID` is stored in the DHT. This URI/`nodeID` binding is signed with the private key that corresponds to the public key which hashes to the `nodeID`. In addition, the current location for a `nodeID` is stored in the DHT and also signed with the private key corresponding to the `nodeID`. In summary, this *two-stage name resolution* has a self-certifying `nodeID`, but no self-certifying SIP-URI.

Alternative approaches for integrity protection of content stored in a P2P network Tamassia and Triandopoulos present an approach for authenticating data retrieved from a P2P network [272] in which they introduce a distributed version of a *Merkle tree* [175]. A data source can cryptographically protect the integrity of multiple data items stored in a P2P network: A root data item is signed by the source, and the integrity of other data items from the same source is protected with an *authentication tree*. This approach assumes that any node in the system can authenticate the public key of a source via some external mechanism. The scheme targets applications where a single source store multiple data items in the DHT; the main benefit of the authentication tree

is that this source does not have to sign each data item.

Pathak and Iftode [197] and Palomar et al. [194] propose decentralised mechanisms for authenticating public keys and content in a P2P network based on *Byzantine agreement*. Assuming that each peer has a *trusted group* of other peers it regards as trustworthy, and assuming that the majority of peers in this *trusted group* is actually behaving in a non-adversary way, peers can authenticate public keys using the protocols from [197]. The actual mechanism requires that each peer in a peer B's *trusted group* challenges the originator of a new, unknown public key received by B. If there is no consensus among the peers in B's *trusted group*, Byzantine agreement is used to enable B to decide on authenticating the newly received public key or not. The work in [194] extends this concept to enable authentication of content received from a P2P network.

Differences to self-certifying SIP-URIs We have presented a solution to protect the integrity of location bindings stored as content in a P2P-SIP network. In our approach, SIP-URIs are self-certifying, i.e. generated based on a cryptographic public key. Prior to our work[8], no self-certifying approach for SIP-URIs has been presented. In addition, our work constitutes a detailed design to protect the integrity of data items stored in a DHT for the purpose of communication session establishment. As another difference to other work, our approach makes the *user identity* in a distributed system self-certifying.

Compared to CGAs [50] and HIP [183], our approach puts self-certifying identities at the application layer, making it suitable to protect P2P networks which route at the application layer. The identity to be authenticated is not a routing layer address or network layer identifier (as in the case of HIP), but a user's identifier at the application layer. Hash extension techniques [52] [50] [51] can easily be applied to our approach, and in fact we suggest the use of such techniques to strengthen the security of the hash function against attacks while keeping a SIP-URI at reasonable length (see 5.3.1).

SFS [172] and CFS [103] concern file system integrity and prevent file modification attacks. Self-certifying SIP-URIs are designed to protect the integrity of location-bindings in real-time communications. They prevent impersonation attacks on a user's identifier.

The work of Tamassia and Triandopoulos [272] seems to target the same goal as our work: authenticating data retrieved from a P2P network. However, there are important differences. In P2P-SIP (i.e. in decentralised service location), only a single location-binding is stored for each URI. Thus, there is no need for an authentication tree and the added complexity. Further, self-certifying SIP-URIs provide the property that the public key belonging to a *source* (i.e. to a SIP-URI) can be authenticated without external mechanisms. Finally, Tamassia and Triandopoulos list protection against replay attacks as a main benefit of their solution compared to other approached. We have presented techniques to prevent replay attacks for self-certifying SIP-URIs, making our solution comparable in this respect.

Authentication approaches based on Byzantine agreement [197] [194] differ from our

[8] originally published in [239]

approach as they assume for each peer in the network a set of peers it initially regards as trustworthy. Additionally, these approaches result in significant communication overhead among peers and demand additional computational resources at each peer. Further, approaches using Byzantine agreement can only guarantee correct authentication up to a certain threshold of attackers. In contrary, self-certifying SIP-URIs do not rely on pre-established trust relationships among peers, work correctly independently of the number of attackers in the network, and require an acceptable cryptographic overhead.

P2PNS [63] and *YURLs* [95] [137] are approaches close to ours. P2PNS has in common with our approach that it aims at securing P2P-SIP networks with self-certifying identities. However, the goal of P2PNS is to provide secure node-ID generation. Self-certifying SIP-URIs protect the user-ID. With a self-certifying node-ID as in P2PNS, the integrity of a location-binding cannot be verified based on a URI. However, in reality, the caller's node is in possession of the URI of the callee, and not its node-ID. A query node cannot rely on a signed URI/node-ID binding in P2PNS unless it knows the node-ID of the callee. Another drawback of P2PNS is that nodes have static node-IDs and a static node-ID/URI mapping. This does not ensure uniform distribution of node-IDs, and hence enables DHT routing attacks, where an attacker repeatedly generates node-IDs until she/he is able to join the DHT in such a way that she/he can cut off a particular node-ID from the DHT. In addition, a static node-ID/URI mapping makes it not straightforward for a user to receive calls for a URI on a secondary or temporary device.

YURLs are similar to our approach as the self-certifying identity is a URI at the application layer, and the application type to be secured is service location. However, *YURLs* use no hash extension techniques to keep the hash output at reasonable length. Instead, *petnames* are the suggested solution to make self-certifying identities human-readable. *petnames* rely on a trustworthy software (e.g. a browser) with an adequate user interface to map a self-certifying identity onto a *petname*. They are thus, in contrary to self-certifying SPI-URIs, not backwards compatible with existing VoIP hardphones which only accept a SIP-URI as an identity. One other difference is that *YURLs* encode self-certifying identities within an http-address, while our approach takes SIP-specifics into account. In addition, one of our core contributions is the application of self-certifying identities in P2P networks, and the design of a self-certifying scheme for DHT-based service location (including protection against replay attacks). The existing work on *YURLs* targets http client-server communications. In principle, however, *YURLs* could also be applied to authenticate data items stored in a DHT (e.g. for a decentralised, DHT-based DNS approach).

Another approach very close to our one is the use of self-certifying URLs by *TOR Hidden Services* [21]. Similarly to our approach, self-certifying URLs are used to locate a service. However, with TOR hidden services, the self-certifying hostname does not actually locate the desired service. Instead, a TOR relay server (as introduction point for the hidden service) is returned. The actual location (i.e. IP-address) of the hidden service is never revealed to the TOR client. This is due to the objective of hiding services (also called *responder anonymity*). Our work does not target anonymity of services; hence the actual location of services is not protected by self-certifying identities in our approach. Further, self-certifying `.onion` hostnames are stored at centralised TOR directory servers.

Our work includes the design of distributed storage of self-certifying identities in a DHT. It is noteworthy to point out that in [108], the use of a DHT as a decentralised alternative to TOR directory servers for storing the addresses of introduction points for `.onion` hostnames is briefly mentioned. However, such a design is currently not used by TOR [22]. More importantly, such a design would not enable clients to connect to hidden services anonymously [108].

5.5 Summary and Contribution

In any SIP communication, a user is represented by a SIP-URI. We protect the integrity of messages regarding this identity by introducing self-certifying SIP-URIs. For these URIs, only the owner of the identity can sign messages. Thus, users can prove their ownership of a SIP-URI.

However, we do not solve the problem of how to bootstrap authentication: How can a caller be sure about the callee's real (physical) identity? Bootstrapping authentication without a central authority or a pre-call trust relationship seems very hard to achieve. We feel it is best to rely on existing authentication infrastructure to achieve this, e.g. by using an https website with a URI directory like a phone book. Even though existing authentication infrastructure (like https web-pages) uses a central authority (and has some problems), this central authority is neither part of the P2P network nor used to authenticate the SIP identity.

With self-certifying SIP-URIs, users cannot choose the characters in their SIP-URI. Essentially, this is the price being paid for the ability to prove ownership of a SIP-URI. We feel that the ability to prove ownership of a URI to any entity in the system is worth this drawback.

We have presented a solution to protect the integrity of location bindings stored as content in a P2P-SIP network. Our proposal is the first self-certifying approach for SIP-URIs. Moreover, our approach is unique as it proposes a self-certifying user identifier in DHT-based communication session establishment. We provide a scheme for using self-certifying SIP-URIs as data items for user registration and location lookup in a P2P network. The proposed scheme requires no central authority, enables verification of message-integrity on all routing hops, and is independent of the DHT or routing strategy being used. Further, it can be used on top of any existing SIP-deployment because all added cryptography is encoded within the SIP-URI.

The proposed scheme has been described in detail. We have shown how our solution prevents man-in-the-middle attacks on content stored in the network. In addition, we have discussed problems that arise when using self-certifying content in a P2P-SIP network and evaluated our approach. Finally, we have examined potential attacks on the scheme and presented countermeasures, most notably against replay attacks.

Our approach as described in this chapter is specific to SIP-based communications,

as the self-certifying property is intrinsically encoded in the SPI-URI. It can be applied to secure DHT-based as well as to client-server based SIP deployments. However, we see self-certifying identities—and in particular self-certifying URIs—as a general mechanism to protect the integrity of location-bindings in any kind of decentralised service location application. For instance, our scheme could be modified to self-certifying URIs for a decentralised Domain Name System (DNS).

Chapter 6

Decentralised Identity Assessment

Contents

In this chapter, we present a decentralised solution for protection against unsolicited incoming messages (*Spam*) in real-time communications[1]. For VoIP, the term *Spam-over-IP-Telephony (SPIT)* is often used to describe such unwanted communications from the callee's perspective. Our solution applies to any kind of multimedia communication Spam; however, for simplicity we will use VoIP as the example application in this chapter and hence the term *SPIT* to refer to unsolicited real-time communications. One core challenge in preventing SPIT attacks is to assess the trustworthiness of the caller's identity. Further, spoofing attacks must be prevented by verifying that the call has been initiated by the user belonging to the caller's identity. We propose to adapt a Web-of-Trust model to real-time communication in order to assess the trustworthiness of incoming signalling messages—e.g. VoIP calls—based on the social relationships among users. We present the design of a system which is capable of cryptographically verifying identity assertion chains associated with users in *real-time* [44], i.e. efficiently in that it induces minimal overhead during the regular processing of signalling messages. We show how our scheme can be applied to *client-server* SIP as well as to decentralised scenarios such as P2PSIP. Further, we present an evaluation of the proposed approach applied to P2PSIP, based on a prototypical implementation. Our results show that indeed *real-time* cryptographic verification of certificate chains among users is feasible for P2P-based VoIP communications and can be performed in a *decentralised* manner. Finally, we highlight the benefits of our system as well as its limitations, discuss open issues, and relate our proposed scheme to other work.

We discuss our motivation and goals as well as existing solutions and their drawbacks in Section 6.1. In Section 6.2 we present our approach and describe the proposed system in detail. Subsequently, we present the results of our prototypical implementation and discuss potential limitations of our proposal (Section 6.3). Finally, we relate our work to other approaches (Section 6.4) and conclude with a summary of our contributions (Section 6.5).

[1]Parts of this chapter (including figures) have originally been published in [244]. See also Appendix A.

6.1 Rationale

6.1.1 Motivation and Goals

As highlighted in Chapter 3, one research challenge for P2P-based VoIP systems is the prevention of *Spam-over-IP Telephony (SPIT)*. In order to prevent SPIT it is necessary to estimate if an incoming signalling message (e.g. `SIP-Invite`) is unsolicited from the receiving SIP user agent's perspective or not. One core challenge is to assess the trustworthiness of the caller's identity with respect to sending unsolicited messages. Moreover, it is important to verify that the call has been initiated by the user belonging to the caller's identity in a SIP-message (the `SIP-URI` in the `From` header). This problem is not easy to solve as VoIP-identities (e.g. the SIP-URI) can be spoofed easily. This makes the detection of social attacks like SPIT and unsolicited communications in general (i.e. any kind of unwanted incoming messages from the callee's perspective) a sophisticated problem.

Existing solutions to protect users against SPIT rely on central entities such as SIP proxies [220] [204] (see further Section 6.4). In a fully decentralised scenario, anti-SPIT protection can only reside in terminals. There is no central entity such as a proxy protecting the terminal. Some existing anti-SPIT techniques such as *whitelisting*, *blacklisting*, or simple *IQ tests* (i.e. challenging the caller) can be integrated in terminals [245]. However, existing solutions for identity assertion in SIP rely on pre-call trust relationships between SIP entities of the caller's and callee's domain. The process of issuing certificates is centralised and certificate verification is based on a public key infrastructure (PKI). In P2PSIP, no centralised SIP entities nor a centralised PKI exists. Therefore, such mechanisms cannot be applied to a completely decentralised setting.

In this chapter we present a solution for decentralised identity assertion: We propose to adapt a Web-of-Trust (WoT) model to *real-time communications*. Our approach uses the social relationships between users in order to detect if an incoming signalling message was really sent by the user belonging to the caller-identity inherent in the message or not, and further to estimate if this user/identity is trustworthy, i.e. not sending unsolicited (or even malicious) messages. In the absence of any reliable information about a caller's identity (e.g. in case a callee has not received a call before from a given caller), a fully autonomous decentralised solution is not feasible. Instead, the only decentralised approach is to rely on other nodes to judge the trustworthiness of the identity of an incoming call. This is our rationale for choosing a Web-of-Trust model.

To be applicable to real-time communications such as VoIP, the social relationships between users in a Web-of-Trust have to be known to the callee either prior to the call or must be derived at the time of the call *in real-time*, i.e. efficiently enough to still allow for a session establishment duration that is acceptable for real-time communications. Otherwise, the downloading of certificates and cryptographic verification of certificate chains is likely to delay communications (e.g. a VoIP call) too much to be acceptable to users. There exist algorithms for real-time derivation of WoT certificate chains [198]. However, existing application usage models of a Web-of-Trust are different from our

approach and do not enable *real-time verification* of certificate chains. Consider the common usage of the PGP [298] Web-of-Trust in non real-time communications such as email: PGP key-servers (e.g. [10]) merely offer storage of users' certificates. A user who receives a signed email usually has a direct relationship with the sender of the email. Otherwise, the user has to retrieve the certificate of the sender and must then verify the certificate chain between itself and the sender of the email (which most probably results in fetching more certificates). While there exist services for computing the certificate chain between sender and receiver [198], such tools are not integrated in email clients and the cryptographic verification of the certificate chain can only be done *after* receiving an email. More importantly, the necessary certificates need to be fetched subsequently by the user. This renders current usage of Web-of-Trust models infeasible for real-time communications.

One main challenge for our work is hence the adaptation of a Web-of-Trust model to real-time communications such as VoIP. Further, we aim to design a scheme for *decentralised* identity assertion, i.e. the overall system should rely on as few central entities as possible. The main goal of our work is thus to develop and evaluate mechanisms that allow *decentralised identity assertion* for *real-time communications*. In other words, we desire a system distributed among entities which enables the timely—i.e. fast enough to be useful for real-time communications—derivation of a certificate chain between a caller and callee as well as the corresponding cryptographic verification of the certificate chain.

6.1.2 Existing Solutions for Identity Assertion in Real-Time Communications

The research and standardisation communities have realised the problem of SPIT and in particular SIP identity spoofing. Below we summarize existing approaches for SIP identity assertion and point out why they are not applicable to P2P-based VoIP systems.

Many proposed mechanisms for estimating if an incoming message was sent by the user belonging to the caller-identity rely on so-called *strong identities* [276]: If an identity is signed by a centralised authority which is trusted by the receiving end, messages received from this identity are believed to be non-malicious. For SIP, the identity of the caller is the `SIP-URI` in the `From` header of a `SIP-Invite` message. RFC 4474 [201] specifies a mechanism for having this identity, i.e. the `SIP-URI` of the caller, signed by the domain of the caller. When the caller places a call, its SIP domain challenges the caller with an authentication request. Only after proper authentication the domain will sign the outgoing message. When receiving a call, the proxy of the callee verifies the signature with the public key of the caller's domain [201]. Figure 6.1 shows this approach in the context of the classic SIP trapezoid.

For verifying the binding of a retrieved public key to the domain of the sender a PKI is used. A hierarchy of Certificate Authorities (CAs) is used to establish a cryptographically verifiable certificate chain between any two SIP domains. This implies that a central authority on top of this CA-hierarchy, the Root-CA, is regarded as trustworthy by all

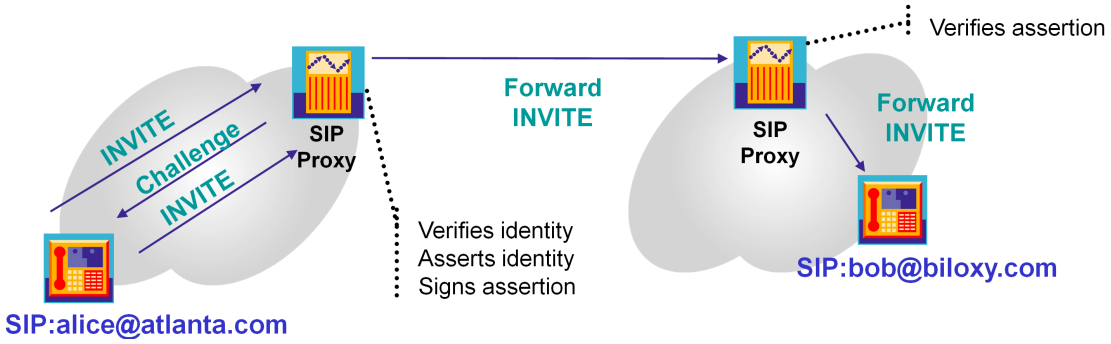

Figure 6.1: SIP Strong Identity Approach (RFC 4474)

entities in the system. The Root-CA is the basic building block for all assurance related to assertions of identities in such a system. Thus, the process of issuing certificates is centralised.

In addition, with current approaches there is no trustworthiness associated with the process of identity assertion. The cryptographic assertion of an identity is merely technical: the proxy's assertion does not regard the trustworthiness of the signed identity but instead in most cases only assesses the possession of the private part of an asymmetric cryptographic key pair. It is important to realise that an identity asserted by its domain can still misbehave, e.g. send unsolicited messages (which are correctly signed by its outgoing SIP-proxy). The reason is that it is not specified what the basis for asserting a SIP-identity is. In particular, identity assertion is not tied to the perceived trustworthiness of the asserted identity regarding certain actions (e.g. with respect to sending out unsolicited messages).

In summary, existing approaches for SIP identity assertion rely on centralised SIP servers and in addition on a centralised PKI for certificates. Further, identity assertion is merely technical and does not consider the trustworthiness of an identity regarding an expected behaviour. In contrast, our goal is to design a decentralised solution for SIP identity assertion. By using a Web-of-Trust, our approach also enables to bind identity assertion to social relationships among users.

6.2 Adapting a Web-of-Trust Model to Decentralised Real-Time Communications Identity Assertion

The main idea behind our approach is to use a Web-of-Trust to verify the binding of an unknown identity to a public key and additionally to verify the trustworthiness of this identity. The rationale behind this approach is that the social relationships among users (e.g. in a VoIP network) are very beneficial for assessing the trustworthiness of identities. Users can assess the trustworthiness of identities based on the perceived interaction in the past, e.g. through received phone calls in the case of VoIP. Using a Web-of-Trust adds the necessary cryptographic primitives so that users can express their assessment

of SIP-identities by signing other users' identities. We assume the reader to be familiar with the general scheme of a Web-of-Trust, as used for instance in PGP [298].

Compared to using a Public Key Infrastructure (PKI) for retrieving and verifying public keys of the certificate chain between caller and callee, our approach has the advantage of *decentralised identity assertion*: In a Web-of-Trust, the process of issuing and signing public keys is not centralized but distributed among nodes. Thus, there is no single root authority which has to be trusted by all participants in the system (as is the case in a PKI). Rather, the authority to assert identities is distributed among all nodes in the system.

6.2.1 Assumptions and Definitions

We assume that a Web-of-Trust (WoT) infrastructure exists (e.g. the publicly available protocols used by PGP [81]) where redundant servers (similar to PGP key-servers [10]) store certificates of users. The WoT key-servers update each other's key-database with a special protocol amongst themselves. Users can upload certificates which have been signed by other users to these servers. Each public/private key pair has a unique *key-ID* in the system. The binding between a user-identity (*user-ID*) and such a key-ID is signed by users in certificates.

A key assumption for our system is that users sign other identities in the system only if they have had positive (e.g. non-malicious or solicited) interactions with this identity in the past. This means that users can express how trustworthy they regard other users to be (with respect to sending malicious/unsolicited messages) by signing the certificates of these other users in the WoT. Note that in a PGP-like WoT, key-servers are not trusted entities; they simply store certificates. Any user can store any certificate on these servers. Specifically, the signatures of the certificates are not necessarily verified by the key-servers. This task is left to the users. Thus, to achieve real-time computation and verification of a *certificate chain* between two users this approach is not feasible per se and needs to be modified. We define a certificate chain between two user identities as follows:

> **Definition 14** (**Certificate Chain**)**:** *A* Certificate Chain *(or* transitive identity assertion path*) between two user identities u_x, u_l in a Web-of-Trust is a chain of signed certificates between users in the Web-of-Trust such that u_x* asserts *u_1* asserts *u_2 ... u_{l-2}* asserts *u_{l-1}* asserts *u_l, where* assertion *refers to signing the binding of the identity with the corresponding public key.*

We use the terms *certificate chain* and *identity assertion path* synonymously. The length l of an identity assertion path is the number of intermediate entities in the certificate chain between the two users including u_l. Note that there can be various different identity assertion paths between two users. We thus define the minimal certificate chain length as described below:

Definition 15 (**Shortest Certificate Chain Length**): *The* Shortest Certificate Chain Length l_{min} *between two user identities is the minimal certificate chain length among all certificate chains between these two users.*

The underlying assumption is that in the average case the shorter the certificate chain, the less is the probability that the transitive assertion path between two users has been infiltrated by an attacker (e.g. by deluding a legitimate user to assert an attacker's identity). Adopting the definition of an assertion path to real-time communications such as VoIP and expressing it cryptographically, we say that there is a certificate chain of length l between callee and caller, if in the WoT there is an assertion path such that

$$[keyID(callee)]\ signed\ [keyID_1, userID_1]\ signed\ [keyID_2, userID_2]$$
$$\ldots [keyID_{l-1}, userID_{l-1}]\ signed\ [keyID(caller), userID_{caller}].$$

Since the receiving party (i.e. the callee) wants to know how trustworthy the calling identity is, we are only interesting in identity assertion paths going from callee to caller (and not vice-versa).

All (or a given subset of all) certificate chains for a given WoT can be visualised in a certificate graph, which we define as follows:

Definition 16 (**Certificate Graph**): *The* Certificate Graph *of a Web-of-Trust is a directed graph $G = (V, E)$ where vertices (V) represent identities and edges (E) represent certificates issued among identities. A directed edge $v_1, v_2 \in E$ denotes that v_1 asserted the identity v_2 with a certificate.*

Example 6.1: *Figure 6.2 shows the certificate graph between two user identities in the PGP Web-of-Trust (obtained with [88]). In the example, the minimum length of a certificate chain between* `jan.seedorf@nw.neclab.eu` *and* `dwing@cisco.com` *is* 6*. Note that in the figure also the respective PGP key-IDs of the corresponding public key for each identity-key binding is displayed, e.g.* `C9E46707` *for* `jan.seedorf@nw.neclab.eu`*.*

The overall rationale is that the longer the identity assertion path, the less reliable is the (indirect) signature chain of the caller's identity to the callee. Our approach potentially enables identifying trustworthy identities (*whitelisting*) but not the detection of badly behaving identities. In other words, if no or only a long certificate chain can be found, our approach cannot assess the trustworthiness of the caller. We therefore assume that our WoT scheme is used in conjunction with other security and SPIT prevention mechanisms which either reside on central entities or on the receiving terminal.

We propose detailed schemes on how to apply our idea to a client/server-based VoIP system as well as to a P2P-based VoIP system. Our main interest is the application to a fully decentralised system such as P2PSIP. However, it is worth noting that our decentralised identity assertion scheme can be applied to a centralised client/server SIP system just as good. Furthermore, it is easier to grasp the overall scheme and message

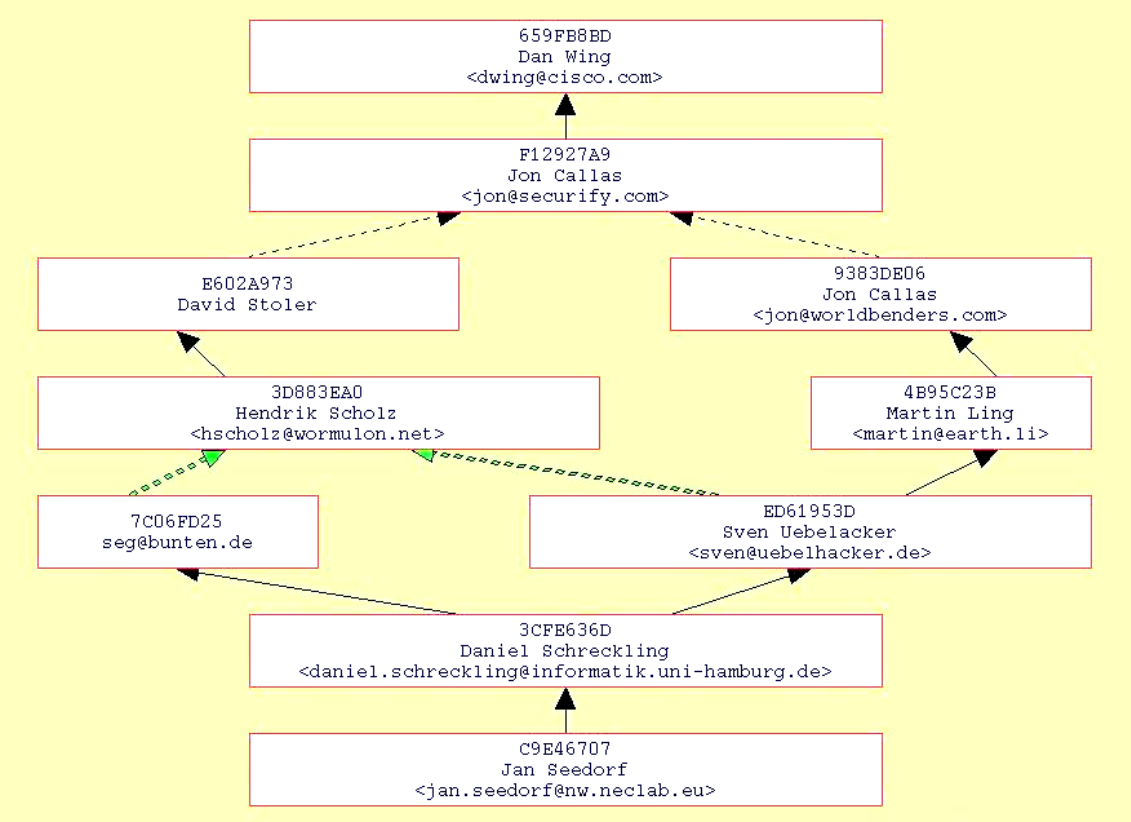

Figure 6.2: Example of a Certificate Graph

flow if explained how it can be applied to a client/server setting first. For each use case we have specific assumptions:

1. *Client/server-based SIP scenario:* Here, we assume that the callee is protected by its upstream SIP proxy which analyses incoming messages on behalf of the callee. To assess the trustworthiness of incoming calls, the callee's proxy can either invoke an assertion path calculation from a third entity or such a computation can be performed by the SIP proxy itself. Depending on the length of the shortest identity assertion path found, the proxy conducts further processing of SIP messages[2]. The result of the WoT mechanism is combined with a holistic VoIP protection solution, e.g. as presented in [204]. If a short certificate chain is found, we assume this to be an indication of the incoming call not being unsolicited. Depending on system settings, e.g. the set certificate chain path length threshold, the call might directly be forwarded to the callee, or this fact might influence an overall security score computed by the callee's SIP proxy.

2. *P2PSIP scenario:* Here, we assume that the callee's terminal itself analyses incoming messages. The reason for this assumption is that in a P2PSIP scenario the callee cannot rely on any upstream entity for protection. Thus, to assess the trustworthiness of incoming calls, the callee's terminal must in some way do the assertion

[2]e.g. forwarding to the callee for very short path lengths, conducting further security tests for medium length paths, and potentially directly forwarding the call to the mailbox of the callee for long paths

path calculation itself. Since there are existing anti-SPIT mechanisms which can be deployed at the receiving terminal [245], we account for other security checks of incoming messages conducted by the receiving terminal which can be combined with identity assertion into an overall rating. The assessment of the incoming SIP message based on identity assertion (potentially combined with other mechanisms) results in a SIP-response sent to the caller. Depending on the overall rating, such a response can either reject the phone call, impose some additional challenges to be solved by the caller, redirect the call to some outsourced protection entity, or accept the call.

6.2.2 Real-time Derivation and Verification of Certificate Chains

Our approach relies on the computation of transitive identity assertion paths between callee and caller. In principle, an attacker can fake such paths by uploading a certificate to a key-server which binds its user-ID to its key-ID, but with an invalid signature apparently from a legitimate user. This results in falsely marking the attacker as trustworthy. To prevent such assertion path forging, any certificate chain in a WoT must be cryptographically verified.

6.2.2.1 Verifying Key Server

Our approach enables to find the identity assertion path between two identities as well as cryptographically verifying the complete certificate chain *in real-time*. To achieve *derivation* and *cryptographic verification* of an identity assertion path in real-time, we introduce a specialized WoT key-server as part of the key-server federation which operates normally in the key-server network but additionally does the following:

1. For each certificate it receives either from a user or from another server it verifies the signatures in the certificate. The server can do this because as a key-server it is in possession of the corresponding public keys to the private keys which signed the certificate. In case the key-server is missing a public key necessary to verify a newly received certificate, it tries to retrieve this key from other servers in the key-server network. If it cannot obtain the necessary certificate, the corresponding signature is considered as unverified by the key-server.

2. It periodically publishes a file which contains in a compressed, machine-readable format the certificate graph (i.e. the identity assertion relationships) between all the certificates it stores of which it has verified the signatures. These identity assertion relationships are verified by the server in the sense that all cryptographic signatures responding to the certificate graph inherent in the file have been verified. The file published by the server is structured in such a way that certificate paths between any two identities can be computed efficiently using this file. We refer to such a file as a *Verified Compressed Certificate Graph (VCCG)*.

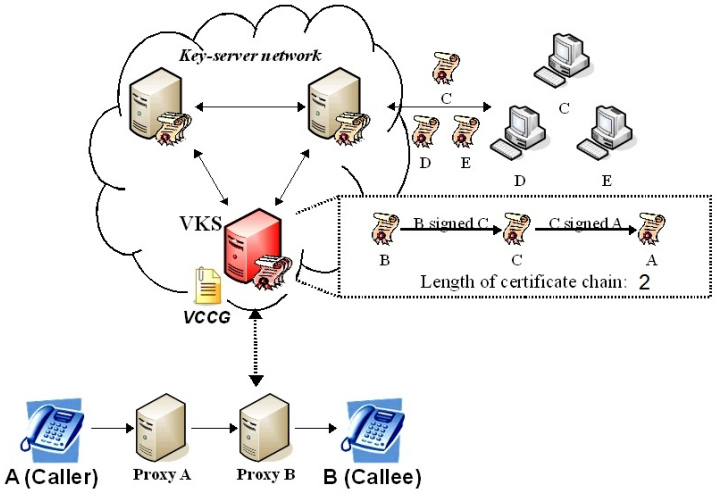

Figure 6.3: System Architecture of Proposed Approach

Note that there already exists a file format for publishing a certificate graph in a compressed way, the so-called *.wot*-file format [87]. Such files are created by certain PGP key servers. We demand that all signatures have been verified before creating such certificate graph files. In case the corresponding signatures for a certificate graph have not been verified, we refer to such a compressed data structure (with unverified certificate chains) as an *Unverified Compressed Certificate Graph (UCCG)*.

The special *Verifying Key-Server (VKS)* is a lightweight central entity in the system. Its function is to to correctly perform certificate verification and to correctly compute the certificate graph as outlined above. It is trusted to perform these operations correctly by the receiving party, i.e. either the callee's SIP-proxy (in the case of client/server SIP), or the callee's terminal (in the case of P2PSIP). It is important to understand that the VKS is not a trusted authority with respect to issuing and signing certificates. This functionality is decentralised in our scheme based on the Web-of-Trust model. The authorities which sign certificates (i.e. the users) are distributed over the system. The VKS fetches publicly available certificates from other key-servers and verifies the signatures. The callee (or its proxy) merely relies on the fact that the VKS performs this signature verification correctly. Most notably, the VKS is not necessarily needed for the establishment of phone calls: the receiving party can use the VCCG published by the VKS to do the certificate chain calculation itself. Indeed we study such an approach when applying the scheme to P2PSIP networks.

6.2.2.2 Efficient Certificate Chain Computation

The VCCG published or used by the VKS itself contains the identity assertion relationships among all nodes in the WoT in a compressed form. Identity assertion paths derived

from the VCCG can be considered verified because all certificates corresponding to the certificate graph have been verified a priori by the VKS. We now consider the problem of deriving the certificate chain from the VCCG efficiently.

For each identity in the WoT, it can be obtained from the VCCG which other identities it asserted. Also, all the identities which asserted a specific identity (and its certificate) can be found directly using the VCCG. A simple approach for finding an identity assertion path is to start at the callee's identity and conduct a Breadth-First Search (BFS) on the VCCG until the caller is found. However, BFS has a time-complexity of $O(b^{d+1})$ where b is the branching factor and d is the depth of the graph [225].

Given that from the VCCG it can be inferred directly for each identity i not only the other identities which i asserted, but also the identities which signed i's certificate, it is more efficient to start a search from both directions, i.e. in parallel going from the caller to the callee and vice versa. The time-complexity of this search is $O(2b^{\frac{d}{2}})$, because essentially two BFS algorithms are executed with half the depth d [225]. Note that the time-complexity of such a *double-sided* Breadth-First Search is less than the $O(b^{d+1})$ for standard BFS (assuming $b \geq 2$). A sketch of an algorithm for computing a certificate chain in a WoT efficiently based on the above rationale has been presented in [198]. We refer to such an algorithm as $dBFS$ for *double-sided Breadth First Search*. In the following, we assume that a $dBFS$ (or similar) algorithm is used for fast certificate chain derivation.

6.2.3 A Scheme for Decentralised Identity Assertion in Real-Time Communications

Before we detail how the VKS and a WoT can be used for identity assertion in a P2P-based VoIP system, we present a scheme for using our decentralised identity assertion approach in a client/server SIP setting.

6.2.3.1 System Architecture

The general procedure of our approach is as follows: A user signs a signalling message (e.g. `SIP-Invite`) with its private key and appends (e.g. via S/MIME [208]) a self-signed certificate containing the corresponding public key[3]. The proxy of the callee receives the message with the attached self-signed certificate of the caller. To check that the message was really sent by a user who is in possession of the corresponding private key, the callee's proxy verifies the message-signature using the public key from the attached certificate. To further check that the message was sent by the user belonging to the identity inherent in the message and to check that the identity of the caller is trustworthy, the proxy checks that the SIP-URI in the *From-header* of the message corresponds to the SIP-URI in the attached certificate and invokes an algorithm which delivers a certificate chain between

[3]The concrete way of signing a message is orthogonal to our method as long as the signature is unique for every message to prevent replay attacks (RFC 4474 [201] specifies such signatures for SIP messages).

the caller and the callee. The input to this algorithm is the certificate attached to the received SIP-message as well as the receiver's certificate. Since the certificate attached to the message binds the caller's user-ID (e.g. his/her SIP-URI) to its key-ID, attackers cannot spoof SIP identities as long as they are not in possession of the corresponding private key. Depending on the length of the identity assertion path between caller and callee, the proxy may invoke further steps to check the trustworthiness of the message, e.g. by applying other tests on the message [204]. The computation of identity assertion paths is either done by the callee's proxy or by a third party.

Figure 6.3 shows the general architecture of the proposed solution. As described previously, the VKS in the figure is a specialized key-server which not only stores keys/certificates but additionally verifies the signatures. Also, it periodically computes a VCCG file which contains the identity assertion relationships between all verified certificates by the server. Using this file, it can offer the service of computing a certificate chain between two identities. As an alternative, it may also publish this VCCG file to be used by others. In the example, however, a proxy which receives a message from a A to B uses the services offered by the VKS to find out the minimum certificate chain length l_{min} between B (callee) and A (caller). The server detects that B has asserted an identity C which in turn has asserted an identity A. Since all signatures in the corresponding certificates of A, B, and C have been verified by the VKS a priori during their upload, the certificate chain is not only computed but verified as well.

6.2.3.2 Detailed Scheme and Message Flow

We now describe the cryptographic procedures and message flows of our proposed scheme in detail. We describe our scheme for the case that the VKS is used by the callee's SIP proxy on every call to compute a verified shortest certificate chain length l_{min}. In principle, the callee could also perform the certificate verification process itself for each call. However, this would result in longer call setup time because the callee would need to fetch the corresponding certificates from a key server. We postpone the investigation of this trade-off, i.e. between depending on the VKS for certificate verification and call setup duration, to Subsection 6.2.4 where we discuss applying our scheme to P2PSIP networks.

In the following, we use the notation for digitally signing and verifying messages introduced in Section 5.2. Hence, we denote a self-signed certificate for identity s as $\text{sign}_{k_{priv}(s)}\{k_{pub}(s), s\}$, meaning that the signature cryptographically protects the binding of identity and corresponding public key (e.g. by signing a hash of the concatenation of both elements). Assume a caller (with SIP-URI s) is trying to establish a SIP-based VoIP call with a callee (with SIP-URI r). Assume further that the callee is protected by its proxy P_r, and the proxy is in possession of the key-ID for the certificate of r, $k_{ID}(r)$. The proxy uses a special, trusted WoT key-server *VKS* which offers real-time derivation of pre-verified certificate chains. Then the following steps are executed:

1. P_r receives a `SIP-Invite` message m which is signed by s with its private key

$k_{priv}(s)$, attached is a self-signed certificate from s:

$$s \rightarrow P_r : \ \text{sign}_{k_{priv}(s)} \{m\}, \text{sign}_{k_{priv}(s)} \{k_{pub}(s), s\} \tag{6.1}$$

2. To protect r, P_r needs to find out if the certificate attached to m really belongs to the SIP-URI in the From-header of m (i.e. s). Therefore P_r first verifies the signature using the public key from the certificate:

$$P_r : \ \text{verify}_{k_{pub}(s)} \{m\} \tag{6.2}$$

If the signature is valid, P_r knows that whoever sent m was in possession of the private key $k_{priv}(s)$. Otherwise (i.e. if the signature is not valid), P_r rejects m.

3. Additionally, P_r wants to verify that the attached public key really belongs to identity s and to know how trustworthy the identity s is. Thus, P_r computes the key-ID for s, $k_{ID}(s)$, by hashing the certificate of s with a specified cryptographic hash function h_{KID}, and then sends this key-ID as well as the key-ID of the callee, $k_{ID}(r)$, to the VKS:

$$P_r : \ k_{ID}(s) = h_{KID} \left(sign_{k_{priv}(s)} \{k_{pub}(s), s\} \right) \tag{6.3}$$

$$P_r \rightarrow \text{VKS} : \ k_{ID}(s), k_{ID}(r) \tag{6.4}$$

4. The VKS computes the length of the minimal certificate chain length in the WoT between r and s, l_{min}. To find the shortest certificate path efficiently, it can use a double-sided breadth first search ($dBFS$) algorithm to find the minimum certificate chain from r to s. If there is no certificate chain, l_{min} is 0. VKS returns l_{min} to P_r:

$$\text{VKS} : l_{min} = \text{dBFS} \{k_{ID}(r), k_{ID}(s)\} \tag{6.5}$$

$$\text{VKS} \rightarrow P_r : \ l_{min} \tag{6.6}$$

5. Depending on l_{min}, P_r conducts further processing of m.

Figure 6.4 shows an example for a possible message flow using the proposed scheme for establishing a VoIP call. Here, a message M is sent from caller s to callee r. The callee's SIP proxy server P_r uses an external service provided by a special WoT VKS. It is also possible that the proxy does the identity assertion path computation itself. In this case P_r and VKS could either be located in the same device/entity, or P_r would frequently fetch a VCCG file published by the VKS. Depending on l_{min} (and potentially other mechanisms) the proxy may take different actions on M, e.g. forward M to the callee if an overall score is below a certain threshold.

As an alternative (shown in Figure 6.5), the caller may only append its key-ID (instead of its self-signed certificate) to the message. In this case, the proxy of the callee passes the

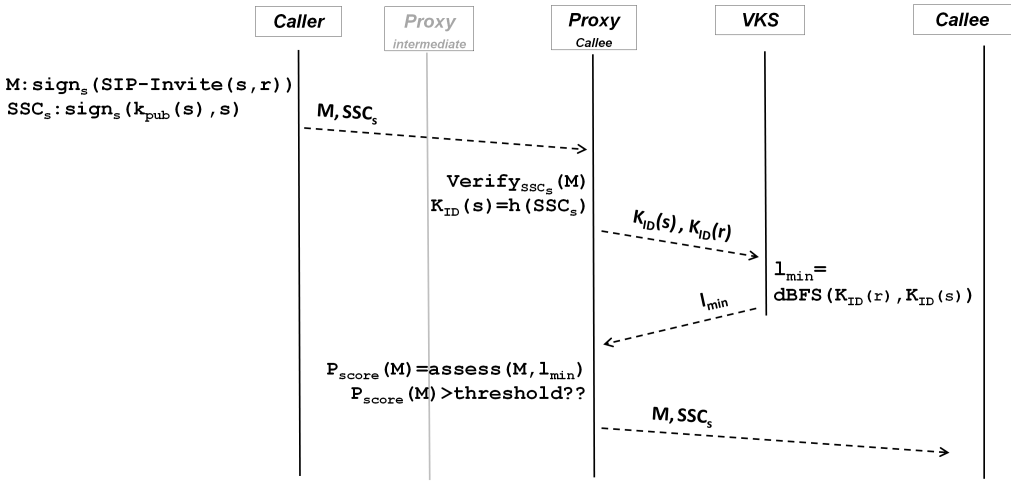

Figure 6.4: Scheme for Decentralised Identity Assertion (Caller Attaching Self-signed Certificate)

key-ID on to the key-server which returns not only the length of the certificate chain but also the certificate of the caller[4]. This variation has the advantage that the `SIP-Invite` message does not increase much compared to a regular SIP message (except for the key-ID in a new SIP-header). Appending the certificate with each `SIP-Invite` message would increase each such message by the size of a certificate (e.g. up to $4kb$ with 4096-bit RSA keys). On the other hand, as a disadvantage in this case (i.e. only appending the key-ID), it is not possible for the proxy to verify the signature of the message instantly (i.e. before passing on information to the key-server). Instead, the proxy can only verify the signature *after* having received l_{min} and the certificate of s from the key-server. This makes the scheme more susceptible to Denial-of-Service (DoS) attacks: An attacker could send bogus messages with invalid signatures in order to stress the computational power of the proxy or the VKS. Note that replay attacks are not possible in any case since the signature is unique for every SIP message.

Note that our definition of the WoT key-ID (equation 6.3) deviates from the specification of a key-ID in PGP. In PGP, the key-ID is defined as the leftmost 64-bit of the hash of the corresponding public key [81]. In particular, PGP key-IDs are not pre-image nor second pre-image resistant. In other words, it can be possible for an attacker to generate a public key which matches an existing key-ID of some other user. In contrary, our definition of a key-ID assumes that a secure hash function is applied to the certificate which binds the user's identity to its public key. Specifically, we assume the hash function to be second pre-image resistant. Hence, if the caller sends its key-ID along with the SIP-Invite message, no identity spoofing attacks are possible: In case an attacker is creating a fake key and correctly signs the SIP-Invite message with this key and appends the corresponding key-ID, there will be no identity assertion path to the key-ID in the

[4]Delivering the corresponding certificate for a certain key-ID is the basic primitive provided by any regular WoT key server.

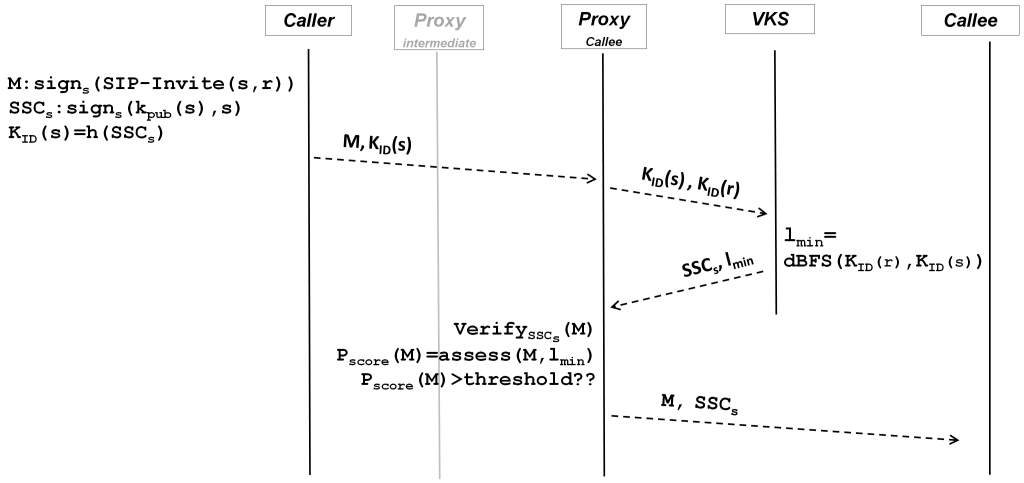

Figure 6.5: Scheme for Decentralised Identity Assertion (Caller Attaching Key-ID)

WoT. On the other hand, an attacker can append a correct key-ID of a legitimate user but is unable to sign the SIP-Invite because he is not in possession of the corresponding private key.

6.2.4 Applying the Scheme to P2PSIP Networks

We now describe how the scheme presented in the previous subsection can be adopted to a decentralised scenario such as P2PSIP.

6.2.4.1 System Architecture

Since our goal is to design decentralised security solutions, we do not consider the case where the callee's terminal contacts the VKS on every call. Such an approach would put too much dependency on a centralised entity. Therefore, we envision the callee's terminal to periodically fetch the VCCG file published by the VKS, in an out-of-band fashion with respect to receiving phone calls. The VKS pre-verifies certificates it uses in the published certificate graph VCCG. Thus, the callee's terminal simply has to compute the minimal certificate chain itself. It can assume that the certificate chain is valid. Figure 6.6 shows the corresponding system architecture: The callee directly fetches the VCCG file from the VKS.

If a self-signed certificate is attached to the SIP-Invite by the caller, the callee's terminal can use it directly to verify the integrity of the message. In case that only the key-ID is attached to the message, the callee's terminal needs to retrieve the corresponding public key for verifying the signature of the SIP-Invite message. It can fetch the caller's certificate from any regular key server (i.e. not necessarily from the VKS): Since the

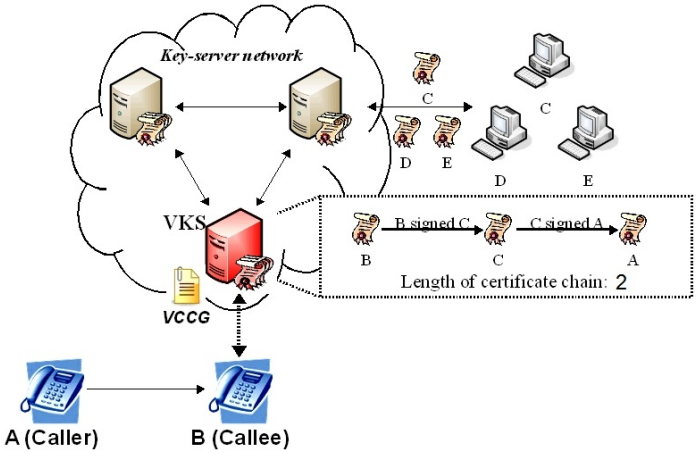

Figure 6.6: System Architecture in a Decentralised Scenario

key-ID is computed via a secure hash function of the certificate in our approach (which comprises the identity and the public key), a secure link to the certificate chain can be verified by the callee by hashing the retrieved certificate and comparing the result with the key-ID sent by the caller.

6.2.4.2 Detailed Scheme and Message Flow

For simplicity, we only present the message flow for the case that the caller appends its self-signed certificate to the `SIP-Invite` message. In case the caller only appends its key-ID to the `SIP-Invite` message, the corresponding certificate needs to be fetched from any key-server by the callee. Prior to receiving calls, r has downloaded a verified certificate graph (VCCG) from a trustworthy VKS:

$$\text{VKS} \rightarrow r: \ \text{VCCG}_{VKS} \tag{6.7}$$

In a P2PSIP scenario, there is no proxy protecting the callee. The callee, r, thus directly receives a `SIP-Invite` message m which is signed by the caller, s, with its private key $k_{priv}(s)$. To confirm that the certificate attached to an incoming message, m, really belongs to the SIP-URI in the From-header of m, r first verifies the signature using the public key from the certificate. The first two steps are thus similar to the client/server case (6.1, 6.2), except that r, the callee, is the entity performing the signature verification of the incoming message:

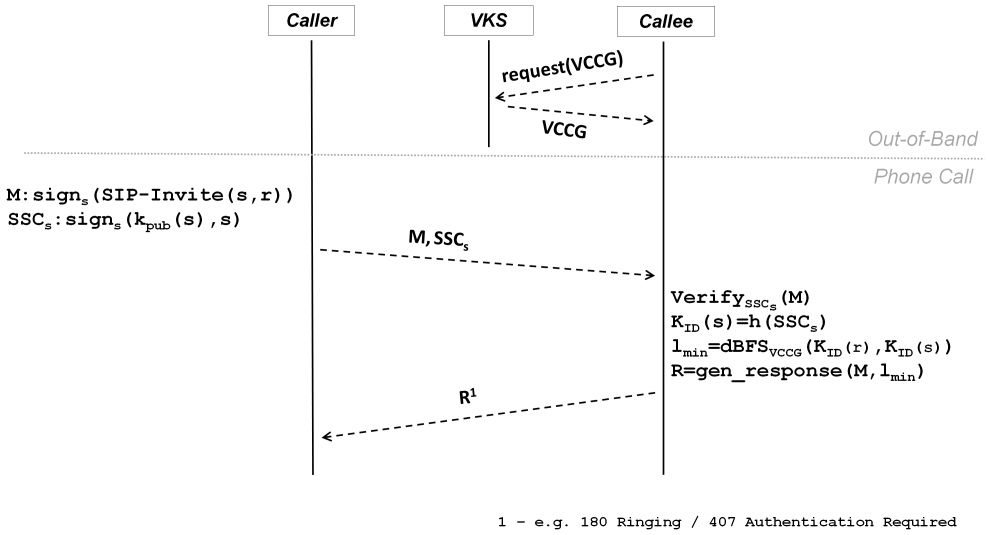

Figure 6.7: Scheme Applied to P2PSIP

1. r receives a message m from s, attached is a self-signed certificate from s:

$$s \rightarrow r: \ \text{sign}_{k_{priv}(s)} \{m\}, \text{sign}_{k_{priv}(s)} \{k_{pub}(s), s\} \tag{6.8}$$

2. r verifies the signature of m. If the signature is valid, r knows that whoever sent m was in possession of the private key $k_{priv}(s)$. Otherwise (i.e. if the signature is not valid), r rejects m :

$$r: \ \text{verify}_{k_{pub}(s)} \{m\} \tag{6.9}$$

3. To see if the public key really belongs to the identity s and to determine how trustworthy the identity s is, r computes the minimum certificate path length as follows. First, r computes the key-ID for s, $k_{ID}(s)$, by hashing the certificate of s with a specified hash function h_{KID}. Assume that r has precomputed its own key-ID $k_{ID}(r)$. Then, r executes a double-sided Breadth First Search ($dBFS$) algorithm on VCCG to find the minimum certificate chain from r to s:

$$r: \ k_{ID}(s) = h_{KID} \left(sign_{k_{pri}(s)} \{k_{pub}(s), s\} \right) \tag{6.10}$$

$$r: l_{min} = \text{dBFS} \{k_{ID}(r), k_{ID}(s))\} \tag{6.11}$$

4. Depending on l_{min}, r creates a SIP response message $R(m)$. For instance, r may return a SIP `180-ringing` message if it accepts the call from s, or r may return a SIP `407-authentication required` message to signal to s that it cannot (sufficiently) authenticate the binding of s and $k_{pub}(s)$.

The message flow is depicted in Figure 6.7 for a message M sent by a caller. The response generated by r may not only depend on l_{min}. Rather, r can use l_{min} as one

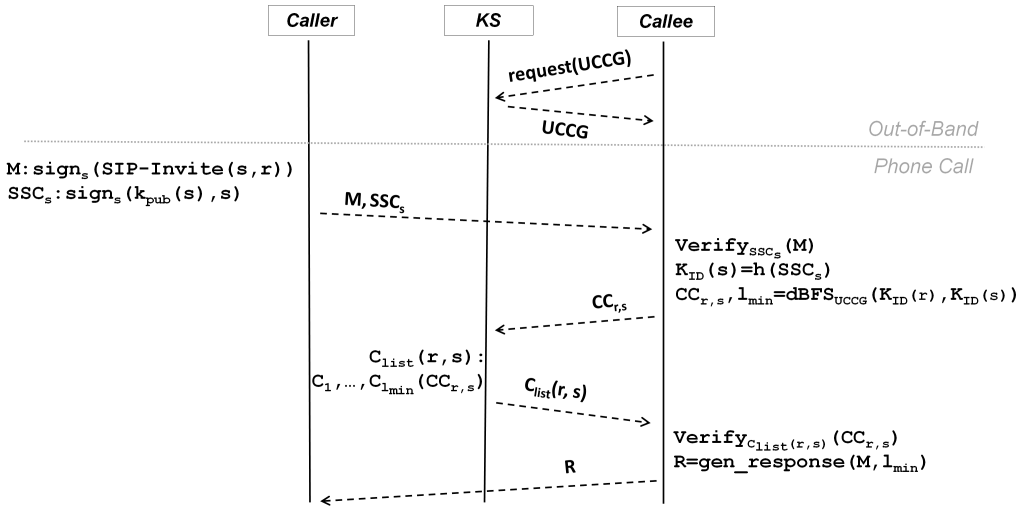

Figure 6.8: Using an Unverified Certificate Graph by the Callee

component (out of many) for judging the trustworthiness of s and for assessing M. Further, r can use a more or less restrictive policy in accepting calls. It may immediately reject calls which do not pass certain checks or it can further challenge the caller in such cases to account for false positives (as outlined e.g. in [204]). For instance, in case r does not immediately accept an incoming message, it may challenge the caller with a computational puzzle [138].

6.2.5 Trade-offs for Higher Degrees of Decentralisation

A potential drawback—from a decentralisation perspective—of the scheme outlined in 6.2.4 is the fact that the callee relies—though not on every call—on a trustworthy VKS. In principle, any untrusted key-server KS could be used which publishes an unverified certificate graph $UCCG$ periodically. However, the retrieval of the certificates corresponding to an identity assertion path must then be done by the either the caller or the callee's receiving terminal. In addition, the callee must verify all certificates in the chain efficiently itself, prior to accepting a call. In this subsection we analyse such adaptations of the scheme which allow a higher degree of decentralisation. However, such decentralisation comes at the price of higher workload for the callee or the caller and results in longer call establishment times. We postpone a quantitative analysis of the different options to 6.3 where we evaluate them based on our prototypical implementation.

6.2.5.1 Using Unverified Certificate Graphs

Using Unverified Certificate Graphs allows for higher decentralisation because the callee does not depend on a trustworthy VKS. However, the callee cannot trust the certificate relationships in an unverified certificate graph, $UCCG$, because an attacker could have

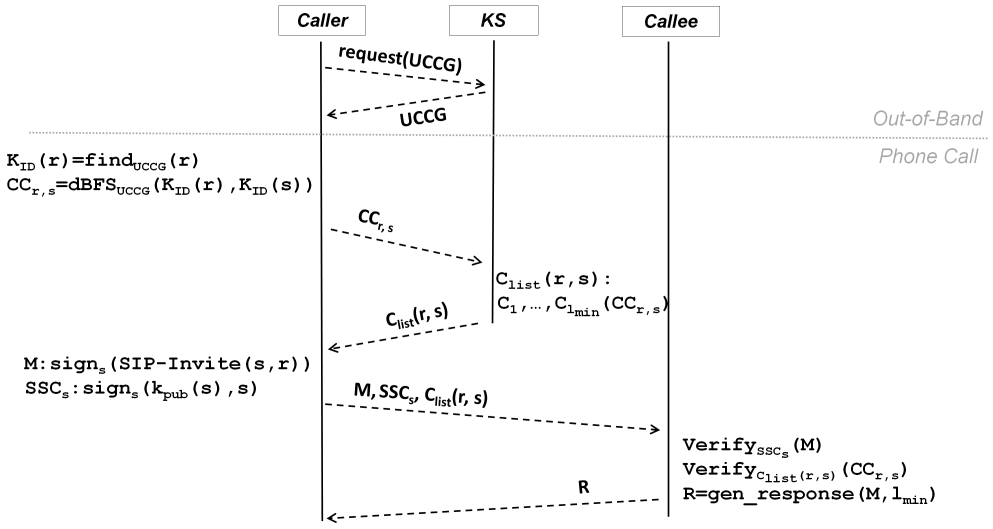

Figure 6.9: Offloading Certificate Chain Computation and Certificate Retrieval to Caller

uploaded an incorrect certificate to a key server. Thus, the *UCCG* can merely be used for computing a *potentially* cryptographically verifiable identity assertion path; additionally, the callee's terminal must have the corresponding certificates at its disposal and verify each certificate in the chain. In principle, path computation and certificate retrieval for a potential certificate chain can be done by the caller or the callee. We discuss both options below. Certificate verification must be done by the callee.

Verification of certificate chain by receiving terminal A trivial usage of an UCCG is as follows: When receiving a call, the callee can use an unverified certificate graph (which is periodically published by any regular key server) to compute the shortest certificate chain length l_{min} itself. It then fetches the necessary certificates from any regular key server and verifies the certificate chain.

Figure 6.8 displays the corresponding message flow. It can be observed that this option puts high load on the callee's terminal. It needs to verify the signature on the SIP-Invite message, compute a certificate chain from r to s, $CC_{r,s}$, fetch all the corresponding certificates, $C(1), \ldots, C_{l_{min}}$ $(CC_{r,s})$, and finally verify each certificate in the chain. Thus, the scheme would make the callee highly susceptible to Denial-of-Service attacks since any incoming bogus message results in several operations at the callee.

Offloading path computation and key retrieval to caller The computation of identity assertion paths and key retrieval can be offloaded to the caller. This avoids the aforementioned drawbacks that the callee needs to perform many operations on every incoming call. Figure 6.9 shows the message flow for this option. The caller, s, periodically downloads an *UCCG* from any WoT key-server. s searches this file for the WoT key-ID corresponding to the callee's identity r, $K_{ID}(r)$. It uses the *UCCG* to compute the shortest identity assertion path, $CC_{r,s}$ between the callee and itself. It then retrieves

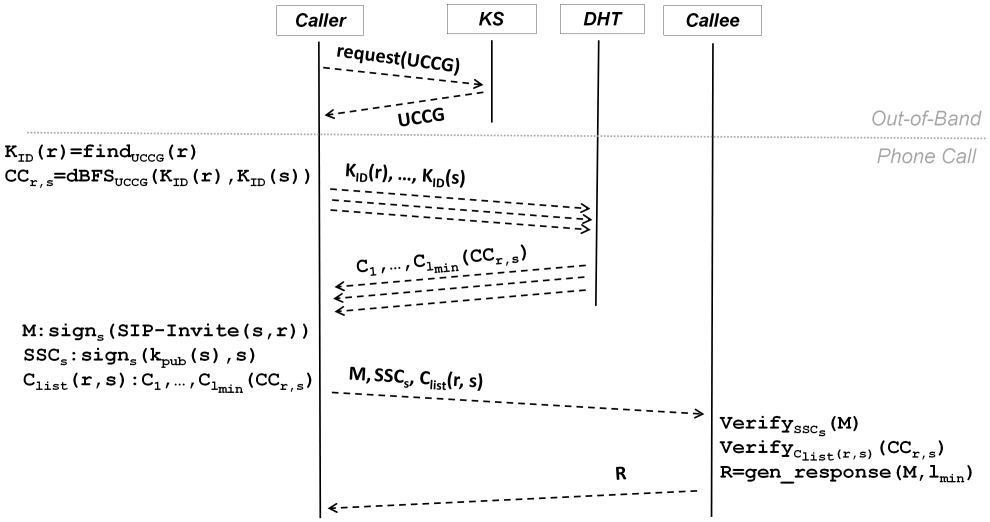

Figure 6.10: Storing Certificates in DHT

the certificates for the path, $C_{list}(r, s)$, from any WoT key-server, and sends them along with the signed `SIP-Invite` message. The callee, r, just needs to verify the signature of the Invite-message and the certificates. Based on the verification and l_{min} it sends back a response, R, to the caller. Optionally, to exclude the possibility of retrieving wrong certificates from key servers, the caller may also verify the signatures of the certificate chain prior to signing and sending out the message.

Note, however, that the mapping from r to $k_{ID}(r)$ is not secure. An attacker could have uploaded a forged certificate which binds the identity r to an incorrect key-ID $k_{ID}(\widehat{r})$. Such forged certificates would result in a Denial-of-Service on s because r might reject (or treat differently) the call if it cannot verify the certificate chain. Using self-certifying SIP-URIs (as proposed in Chapter 5) would prevent such attacks, as the binding between r's public key and r could be verified by s in this case (see further 6.2.6). If no self-certifying SIP-URIs are used and there are multiple certificates for an identity r, s cannot determine which is the correct key-ID $k_{ID}(r)$ to use. We therefore assume the use of self-certifying identities with this scheme and postpone an associated discussion to 6.2.6.

It can be observed in Figure 6.9 that this options puts much load on the caller. The callee only needs to perform several signature verification operations. In particular, the callee does not need to download any certificates. Hence, this option is preferable from the perspective of Denial-of-Service attacks on the callee.

6.2.5.2 Decentralised Certificate Storage

The message flows depicted in figures 6.8 and 6.9 still rely on the availability of a WoT key server for certificate retrieval *on every call*. To further decentralise the scheme, certificates can be distributedly stored in and retrieved from a DHT, i.e. the P2PSIP DHT in which

nodes participate in for SIP service location. Figure 6.10 shows a corresponding scheme with certificate retrieval from a DHT. The message flow is the same as in figure 6.9, except that the caller fetches the certificates which are needed to verify the certificate chain $CC_{r,s}$ from the DHT.

As proposed in [253], certificates can be stored in the DHT by using $h(certificate : [SIP - URI])$ as the DHT key-ID for retrieving the certificate corresponding to a particular SIP-URI. As an alternative, depicted in figure 6.10, the WoT key-ID (equation 6.3) can be used directly as the DHT key-ID. In both cases, the integrity of certificates retrieved from the DHT cannot be verified by the caller unless self-certifying SIP-URIs are used. Similarly, the mapping from r to $k_{ID}(r)$ is not secure if self-certifying SIP-URIs are not used (compare 6.2.5.1). We thus assume that self-certifying SIP-URIs are used when certificates are stored in the DHT and refer to a discussion thereof to 6.2.6.

It is worth noting that in the scheme as described above, merely certificate retrieval is decentralised in a DHT. The computation of a UCCG-file is performed by WoT key servers, i.e. distributed but not completely decentralised entities. We thus assume that each time nodes store a certificate in the DHT, in parallel they store the certificate also in the WoT key server network. It is an open research problem how to compute a certificate graph in a decentralised way, e.g. when keys are stored in a P2P network[5]. Hence, the approach we present above constitutes the highest degree of decentralisation possible with state-of-the-art research.

6.2.6 Simplifications when Using Self-Certifying SIP-URIs

In principle, the use of self-certifying SIP-URIs (as presented in Chapter 5) is independent from the identity assessment schemes we present in this chapter. Self-certifying SIP-URIs enable the *caller* to verify the location of a desired *callee*. Identity assertion, on the other hand, enables a *callee* to assess the identity of a *caller*. Hence, the former scheme mostly secures the initiation of real-time communication, while the later scheme protects the receiver of such a communication attempt.

Nevertheless, there are several instances where a combination of the two techniques provides some benefit. For instance, if self-certifying SIP-URIs are used, P_r would know more about the caller s in the client-server scheme with attached self-signed certificate by the sender (see Figure 6.4). Instead of only knowing that the sender was in possession of the private key which corresponds to the self-signed certificate, P_r could verify that the signature indeed was done by the owner of identity s. Further, if the identity of s is self-certifying, s would not need to attach a self-signed certificate, SSC_s. It would suffice to attach the public key $k_{pub}(s)$ and the key-ID $K_{ID}(s)$. The self-certifying property would enable an intermediate entity (or the callee) to verify that $k_{pub}(s)$ is the correct

[5]In [182], an approach for computing a certificate graph in a decentralised way has been presented. However, the work in [182] uses a probabilistic algorithm for finding certificates and certificate chains in a decentralised fashion. Thus, in contrary to a DHT, it cannot guarantee that a certificate is found. Further, the approach introduces a high number of messages to be sent (i.e. in the range of 100 messages each with several hops) for finding a certificate chain.

public key for identity s.

For a decentralised P2PSIP scenario, self certifying SIP-URIs are useful in case a *UCCG* is used and certificate retrieval is offloaded to the caller (see figures 6.9 and 6.10). In such scenarios, the caller s needs to determine the key-ID of the callee r, $K_{ID}(r)$. This opens the door for attacks with forged certificate bindings between an identity r and an incorrect key-ID $k_{ID}(\widehat{r})$ (see 6.2.5.1). If the identity r is self-certifying, such attacks are not possible because the caller can exploit the self-certifying property to verify that the certificate retrieved, $C(r)$, is correct for r. The caller can thus verify the binding of identity and public key contained in each certificate it (eventually) retrieves for a key-ID if self-certifying SIP-URIs are used.

6.3 Evaluation and Analysis

In this section, we evaluate and analyse our proposed scheme. We present a prototypical implementation and performance results of decentralised identity assertion we obtained with this prototype. Based on these measurements, we compare the different approaches for integrating our approach into P2PSIP presented in the previous section quantitatively. Finally, we discuss general advantages and drawbacks of our approach. More details on our experiments and our prototype can be found in Appendix B.2.

6.3.1 Prototype Implementation

6.3.1.1 Prototypical Implementation of Decentralised Identity Assertion

We implemented our approach in a prototype [244] which uses the existing *PGP* [298] WoT and the corresponding infrastructure (i.e. key-servers [10], protocols [81], etc.). Using the PGP WoT allows us to use the largest publicly available and cryptographically secured WoT for our experiments. Currently the PGP WoT is used to bind email addresses as identities to public keys. In principle, however, this infrastructure would also allow to bind SIP-URIs (which are very similar to email addresses) to public keys. To evaluate our approach with a real WoT, we treat the identities in the existing PGP WoT as SIP-URIs instead of email addresses and we regard the corresponding signatures in the WoT as according to our scheme (i.e. not only binding an identity to a public key but also expressing trustworthiness with respect to that identity not sending unsolicited or malicious messages).

Our prototype sends signed `SIP-Invite` messages to a modified SIP proxy which then verifies the signatures and calculates the identity assertion path length. For signing the messages we follow the procedure depicted in Figure 6.5 (see 6.2.3.2), i.e. the caller attaches its key-ID to the `SIP-Invite` message. We compute the signature for each message as specified in RFC 4474 [201] and also use the SIP `Identity` header for the signature and the SIP `Identity-info` header for the key-ID as defined in [201]. Note,

Name	# identities	# signatures	Date of Snapshot
WOT25k	25,487	230,445	25-Feb-2005
WOT33k	33,050	328,912	01-Jun-2006
WOT40k	40,480	405,289	15-Nov-2008

Table 6.1: WoT Snapshots Used for Analysis

however, that we only use the syntax from [201] and that our approach is different: Instead of transmitting the identity of a PKI certificate authority we convey a Web-of-Trust key-ID (as defined in equation 6.3) in the `Identity-info` header. In order to verify a signature, the proxy has to be in possession of the sender's public key. In our implementation, if this key does not exist in the proxy's local cache, the key is downloaded from a PGP key server.

To evaluate the path search performance, we implemented a *double-sided Breadth First Search (dBFS)* algorithm in our SIP proxy. As input graph for the search algorithm we use snapshots of the PGP WoT in the *.wot* file format [87]. These files[6] contain all key-IDs and the identity assertion relationships (who signed whom) between the PGP users of the largest cluster – the so called *strong set*. The restriction to the strong set implies that between any two identities a certificate chain *always* exists. To analyse the influence of different WoT sizes on the path search performance we used different snapshots: the oldest .wot file we found contains approximately 25,000 identities ($\approx$ 230 thousand signatures); the newest one (at the time of the evaluation) contains approximately 40,000 identities ($\approx$ 400 thousand signatures). Finally we chose a third *.wot* file containing 33,000 identities ($\approx$ 328 thousand signatures). The different WoT snapshots are summarized in Table 6.1.

6.3.1.2 Applying the Prototype to Evaluate Various Use Cases

Our prototype (as described above in 6.3.1.1) is generic in the sense that it enables an evaluation of our WoT approach for client-server SIP as well as for P2PSIP. Our implementation comprises a SIP proxy which is able to

1. verify certificates,
2. fetch certificates from a key server if not locally available,
3. and compute a *double-sided Breadth First Search (dBFS)* to find the shortest certificate chain between to identities.

Executing a large number of experiments, our implementation is able to measure the average time needed for all the three steps above. Figure 6.11 shows a screenshot of the graphical user interface of our prototype implementation, visualising how an identity assertion path between callee and caller is found.

[6]We assume that for each identity assertion path in the .wot file the signatures have been pre-verified. Specifically, we treat .wot files downloaded from real PGP key-servers as a *VCCG* file (as described in Section 6.2), i.e. as if they were published from a trustworthy VKS.

Evaluating client-server SIP with the prototype For a client-server SIP evaluation of our decentralised identity assertion approach, our implementation in principle mimics a SIP proxy (which protects the callee) with a co-located (or integrated) VKS. However, the integrated VKS does not contain all certificates. If a certificate is not cached locally (because it has been retrieved for previous communications), the proxy fetches it from a key server. Comparing with figure 6.5, it can be observed that our implementation emulates the steps $VKS : l_{min} = \text{dBFS}\{k_{ID}(r), k_{ID}(s)\}$ (compare also with eq. 6.5) and Proxy : $\text{verify}_{SSC_s}\{M\}$ (compare also with eq. 6.2). Note that our prototype can also be used to evaluate the option where the caller attaches a self-signed certificate (instead of the key-ID) since all key functionality is emulated (compare with Figure 6.4).

Evaluating P2PSIP with the prototype For an evaluation of our decentralised identity assertion approach in a P2PSIP network, our implementation mimics precisely a P2PSIP callee which has been enhanced with the necessary functionality we introduce for our approach. In particular, our prototype implementation has fetched a VCCG file (i.e. a *.wot* file), and can emulate the following steps (compare with Figure 6.7): Callee : $\text{verify}_{SSC_s}\{M\}$ (compare also eq. 6.9) and Callee : $l_{min} = \text{dBFS}_{VCCG}\{k_{ID}(r), k_{ID}(s)\}$ (compare also eq. 6.11). For higher degrees of decentralisation (see Subsection 6.2.5), our prototype also emulates the main functions computed by the callee (compare Figures 6.8, 6.9, and 6.10), including the fetching of necessary certificates in case an unverified certificate graph (UCCG) is used by the callee (compare Figure 6.8). Note, though, that our prototype only verifies the certificate of the caller, since it assumes the use of a VCCG. In cases an unverified certificate graph (UCCG) is used (see Figures 6.8, 6.9, and 6.10), the callee also has to verify the complete certificate chain. This is not emulated in our prototype. However, we measure the time needed for the verification of one signature (the caller's signature). It is reasonable to assume that the time necessary for the verification of more signatures, i.e. l_{min} signatures when verifying the complete certificate chain, scales linearly with the number of signatures. Our implementation can thus also be used to evaluate use cases where a UCCG is used by the callee.

6.3.1.3 Performance Results

For each WoT snapshot (see Table 6.1) we randomly selected 1.200.000 source-destination key pairs and executed the path search algorithm for each of those pairs. Figure 6.12 shows the distribution of the path lengths found: The distribution is very similar for the different WoT sizes. The average identity assertion path length is nearly identical for the three WoT snapshots (5.99, 6.01, and 5.97 for WOT25k, WOT33k and WOT40k, respectively). Furthermore, independent of the WoT size, 90% of all paths found in our experiments have a length of eight hops or less; 99% of the paths consist of at most twelve hops. Figure 6.13 visualises this by showing the cumulative distribution function for the identity assertion path length in different WoT snapshots.

Figure 6.14 shows the time our implementation needs to find a path of a certain length as well as the 95% confidence intervall. As one expects for a Breadth First Search

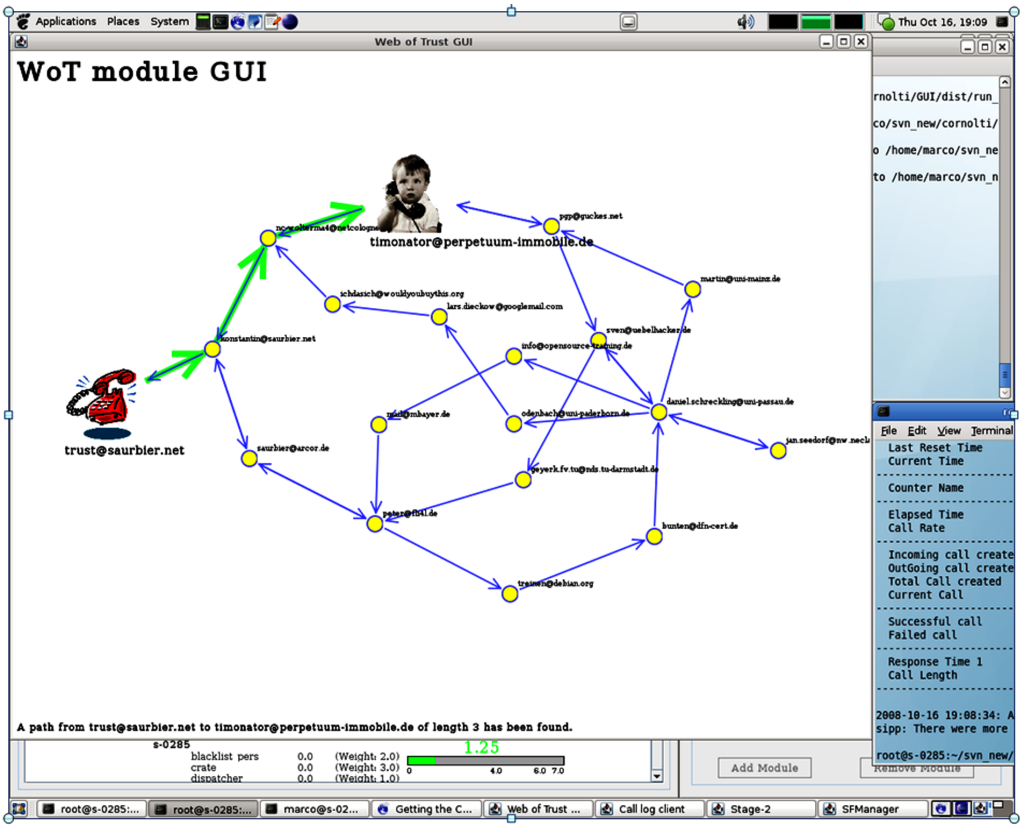

Figure 6.11: Screenshot of WoT Prototype Implementation in SIP Proxy

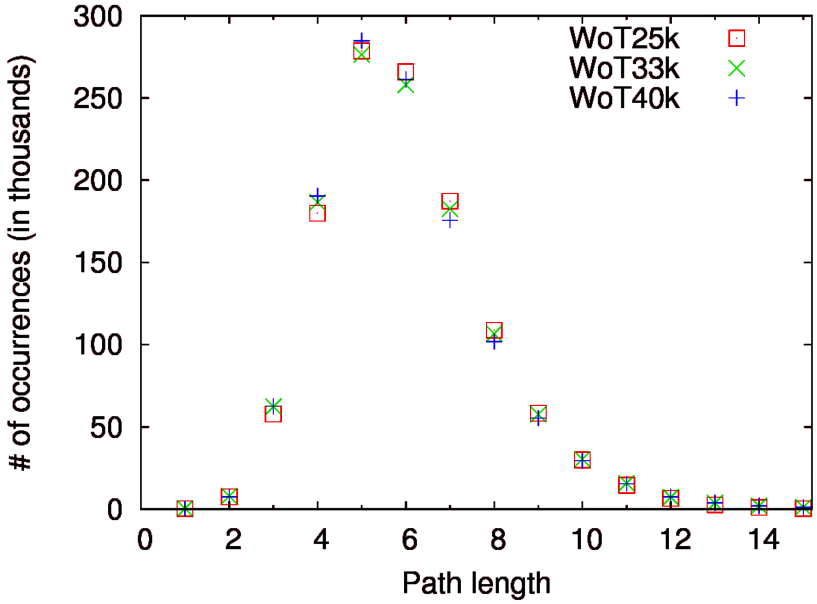

Figure 6.12: Distribution of Identity Assertion Path Lengths for Different WoT Sizes

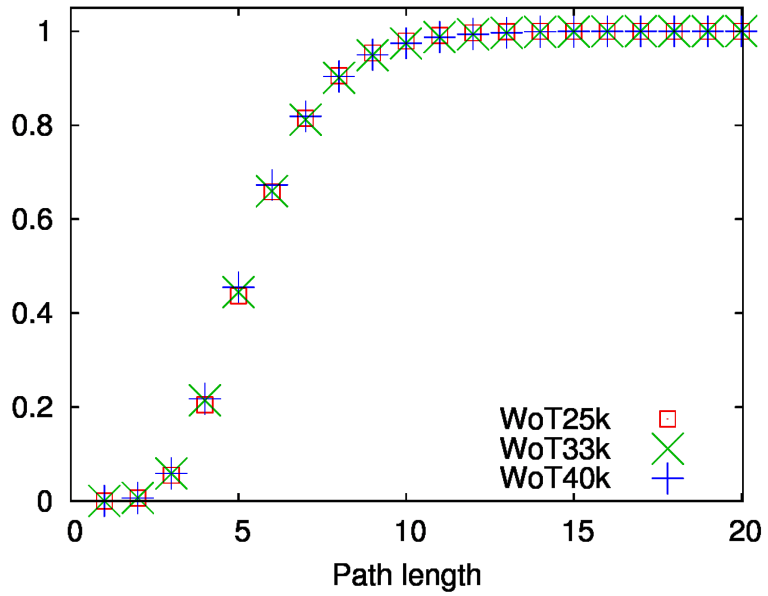

Figure 6.13: Cumulative Distribution Function of Identity Assertion Path Lengths for Different WoT Sizes

algorithm, the time increases exponentially with the path length. At the same time, the results suggest that (at least for the WoT sizes we evaluated) the search time increases linearly with the number of signatures in the WoT. Recall from 6.2.2.2 that the time-complexity of a double-sided BFS algorithm is $O(2b^{\frac{d}{2}})$, where b is the branching factor and d is the depth of the search. In our case, d is the path length and b is the maximum number of identities a member of the WoT has signed. Our results suggest that b is linearly dependent on the overall WoT size.

	Cache Check	Certificate Retrieval	Signature Verification
$\mu[ms]$	0.62ms	98.39ms	3.19ms
σ	0.08ms	47.09ms	0.40ms

Table 6.2: Prototype Measurements (15,000 Repetitions) on Average Time Needed for Cache Checking, Certificate Retrieval, and Signature Verification (Average Time μ, Standard Deviation σ)

Our experiments revealed that our system needs in average $0.6ms$ to check if a key is already present in the local key cache. If the key does not exist in the cache, it is downloaded within $100ms$ on average. After the download, the signature verification is performed in $3.2ms$ on average. The exact measurement results are shown in Table 6.2.

6.3.2 Quantitative Analysis of Decentralisation Trade-offs

In a real world application, the weight a callee puts on a certificate chain—when judging incoming messages—decreases with the length of the certificate chain. In other words,

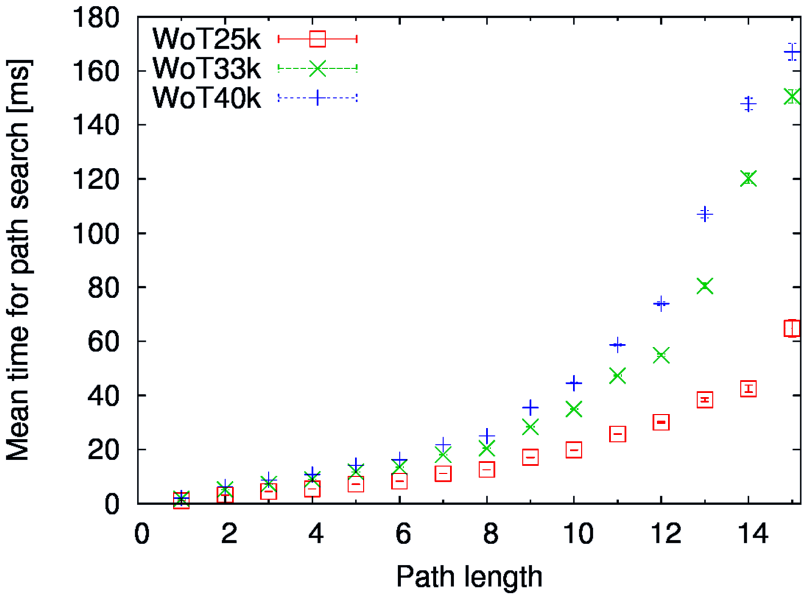

Figure 6.14: Duration for Path Finding Depending on Path Lengths

from the user's point of view, a long identity assertion path does not imply much trustworthiness for messages from the corresponding identity. Thus, it is reasonable to have the path search algorithm only examine identity assertion paths up to a certain length[7]. Our results show that for short path lengths, identity assertion paths can be found in a reasonably short amount of time (compare with Figure 6.14), e.g. around $10ms$ on average for path lengths up to 5.

When comparing the times needed for certificate download, t_{cr} ($\sim 100ms$), signature verification, t_{sv} ($\sim 3ms$), and path search, $t_{dbfs}(l_{min})$ (5-$10ms$ for short path lengths), the time for certificate download, t_{cr}, is the largest one by at least one order of magnitude. Analysing our results for P2PSIP, we conclude that the time needed for identity path calculation, $t_{dbfs}(l_{min})$, is acceptable for real-time communications. Path search needs to be performed only once by the callee for each incoming call, and can be performed on average in about $10ms$ for short identity assertion path lengths. Further, the number of signature verifications computed by the callee can be neglected since it takes less than $5ms$ per signature, so that even for medium path lengths (e.g. 10) the time needed for signature verification of the whole identity assertion path by the callee is acceptable.

The most time-sensitive operation occurs when a certificate has to be downloaded by the callee ($\sim 100ms$ on average). Thus, attention has to be paid to the number of key retrievals needed at the callee. We can differentiate two different cases, depending if the callee uses a VCCG or a UCCG file. The former case requires less time-consuming certificate retrievals, while the later case allows for higher decentralisation (at the price of multiple certificate retrievals):

- In case a VCCG file is used by the callee (compare Figure 6.7), at most one certifi-

[7]Determining the concrete threshold up to which identity assertion paths should be computed is outside the scope of our work.

cate retrieval operation is needed at the callee: In case the caller attaches his/her certificate to the SIP Invite message, no certificate retrieval is needed at all; in case the caller attaches his/her WoT key-ID, the callee only needs to retrieve the certificate corresponding to the caller's key-ID. All other (*intermediate*) certificates are not needed because a VCCG file implies that all intermediate identity assertions have been pre-verified by the issuing `VKS`.

- In case a a UCCG file is used by the callee (compare Figures 6.8, 6.9, and 6.10) all certificates of the identity assertion path are needed by the callee. The time for multiple certificate retrievals can be offloaded to the caller (as in the schemes shown in Figures 6.9 and 6.10). In that way, the callee has to perform no time-consuming certificate retrieval at all. As a drawback, attaching several certificates to the SIP signalling message by the caller implies a larger messages size, demanding more bandwidth. If the callee retrieves each certificate itself (as in the scheme displayed in Figure 6.8), the download of multiple certificates from a `KS` can potentially be performed in parallel to speed up the process.

The overall time needed to perform all necessary operations, t_{dia}, for the different decentralisation options we propose can be estimated as follows:

$$t_{dia} = \alpha \times t_{cr} + \beta \times t_{sv} + t_{dbfs}(l_{min}) + t_{sg} \tag{6.12}$$

where α is the number of times a certificate has to be retrieved, β is the number of times a signature has to be verified, and t_{sg} is the time needed to generate a signature (by the caller on the SIP Invite message)[8]. Table 6.3 compares the overall time penalty our different approaches introduce at the caller and the callee. Note that in the schemes where the certificate chain computation and certificate retrieval is offloaded to the caller, we assume that the caller is attaching its self-signed certificate to the SIP-Invite message. Attaching the key-ID would require an additional, unnecessary key retrieval at the callee in these schemes.

In summary, our results show that when the callee is using a certificate graph with pre-verified certificates (i.e. a VCCG file) and regards identity assertion path lengths up to 5 as useful, the additional operations we introduce will delay an incoming phone call on average by less than $150ms$ ($t_{dia}(Callee) = t_{cr}+t_{sv}+t_{dbfs}(l_{min})) \approx 100ms+3ms+10ms \approx 113ms$). We regard this an acceptable overhead for processing incoming messages by the callee. Moreover, if the caller is attaching its certificate and a VCCG file is used by the callee, the additional operations we introduce at the callee can be performed on average in less than $15ms$ ($t_{dia}(Callee) = t_{sv} + t_{dbfs}(l_{min})) \approx 3ms + 10ms \approx 13ms$).

To decentralise our scheme further—by not relying on a trustworthy key-server—we proposed several schemes where the callee is using an unverified certificate graph (a UCCG file). Table 6.3 reveals that with this option, it is meaningful to offload certificate

[8]We neglect the time needed for hashing the self-signed certificate of the caller onto a key-ID, i.e. the computation of $k_{ID}(s) = h_{KID}\left(sign_{k_{priv}(s)}\{k_{pub}(s), s\}\right)$ by the caller or the callee (depending on the scheme). We regard the time needed for computation of a hash function negligible on today's hardware.

chain computation and certificate retrieval to the caller. In these cases, the callee only has to verify the l_{min} signatures of the identity assertion path ($t_{dia}(Callee) = l_{min} \times t_{sv} \approx 5 \times 3ms \approx 15ms$ for an identity assertion path length of 5). The caller on the other hand needs to retrieve all certificates of the complete identity assertion path in addition to computing the certificate chain ($t_{dia}(Caller) = t_{sg} + (l_{min} - 1) \times t_{cr} + t_{dbfs}(l_{min}) \approx 3ms + 4 \times 100ms + 10ms \approx 413ms$ on average for a certificate chain length of $l_{min} = 5$)[9]. It is debatable if such an additional call setup delay is acceptable on top of the DHT operations P2PSIP requires for finding the location of the callee.

Decentralised Identity Assertion Scheme	**Type of Certificate Graph used**	**Certificate Graphs and Certificates retrieved from**	α, β	t_{dia} **(Caller** s**, Callee** r**)**	**Fig.**
Caller attaching key-ID	VCCG	VKS	1, 1	s: t_{sg} r: $t_{cr} + t_{sv} + t_{dbfs}(l_{min})$	—
Caller attaching self-signed certificate	VCCG	VKS	0, 1	s: t_{sg} r: $t_{sv} + t_{dbfs}(l_{min})$	6.7
Caller attaching key-ID	UCCG	KS	l_{min}, l_{min}	s: t_{sg} r: $l_{min} \times t_{cr} + l_{min} \times t_{sv} + t_{dbfs}(l_{min})$	—
Caller attaching self-signed certificate	UCCG	KS	$l_{min} - 1$, l_{min}	s: t_{sg} r: $(l_{min} - 1) \times t_{cr} + l_{min} \times t_{sv} + t_{dbfs}(l_{min})$	6.8
Offloading certificate chain computation and certificate retrieval to caller	UCCG	KS	$l_{min} - 1$, l_{min}	s: $t_{sg} + (l_{min} - 1) \times t_{cr} + t_{dbfs}(l_{min})$ r: $l_{min} \times t_{sv}$	6.9
Offloading certificate chain computation and certificate retrieval to caller, storing certificates in DHT	UCCG	KS + DHT	$l_{min} - 1$, l_{min}	s: $t_{sg} + (l_{min} - 1) \times t_{cr} + t_{dbfs}(l_{min})$ r: $l_{min} \times t_{sv}$	6.10

Table 6.3: Estimated Time t_{dia} for Caller and Callee for Different Decentralised Identity Assertion Schemes Applied to P2PSIP

Moreover, in case a DHT is used for certificate retrieval, t_{cr} is additionally increased by the time it takes to look up a root node for a certificate in the DHT[10]. However, note that the certificate retrieval as well as the necessary DHT lookups prior to retrieval from DHT root nodes may be performed in parallel, resulting in less time needed for the

[9]Even though we did not measure the time needed for signature generation at the callee, t_{sg}, it is reasonable to assume this value to be similar to the time needed for signature verification, t_{sv}.

[10]How many hops are needed for a DHT lookup depends on the size of the DHT and if or to what extent it is infiltrated by attackers (see further Chapter 4). The time it takes for each DHT hop depends on the distribution of DHT nodes over the underlying network and the specifics of this underlying network. It is therefore hard to estimate the time a DHT lookup takes on average for the general case.

overall retrieval of all certificates. In fact, in case certificates are retrieved from the DHT, a query node could not only start parallel lookups for the necessary certificates, but start these certificate lookups already in parallel to the regular DHT lookup for the location of the callee. Since certificates are stored in the same DHT as location-bindings, these parallel lookups should consume—on average—comparable time, so that the root nodes for certificates should roughly be found in the same time as the lookup for the location of a SIP-URI takes.

Call setup times in the order of seconds are accepted by users for phone calls, e.g. ITU-T regards a call setup delay of $8s$ as acceptable for international ISDN calls (so-called *Post-selection delay* [30]). Therefore, even extending call setup up to $500ms$ (or possibly even more in case of time-demanding DHT lookups) may very well be acceptable.

6.3.3 Limitations of the Proposed Approach

We propose to adopt a Web-of-Trust model to real-time communication where users can assert identities cryptographically. Exploiting the social relationships among users, we enable decentralised identity assertion without relying on a trusted third party for certifying identity assertions. Our scheme can be used to assess the trustworthiness of a sender of a message, e.g. a VoIP caller, with respect to sending non-malicious or non-solicited messages. Using asymmetric cryptography, our proposed approach ensures that identity assertions can be securely verified by the callee. If the signature on the SIP-`Invite` is unique for every SIP message (e.g. by including the SIP `Call-ID` and the SIP `Date` header field in the signature as specified in RFC 4474 [201]), our scheme is protected against replay attacks.

Our results show that a simple double-sided Breadth First (*dBFS*) path search algorithm is fast enough to calculate the identity assertion path length in a reasonable amount of time for short certificate chains. We could not perform experiments on larger real world WoTs since, to our knowledge, the PGP WoT we used is the largest one available. On the other hand, we regard the limitation to short identity assertion paths to be acceptable since the benefit to the callee decreases with the path length, rendering long identity assertion paths less useful. In summary, our measurement results show that our overall scheme implies an acceptable overhead—with respect to the time needed to compute the additional operations we introduce—to be used in decentralised real-time applications (such as P2PSIP).

To work in practice, our approach relies on users behaving correctly. If users are not careful in signing other identities, the overall system becomes less useful because it can be infiltrated by attackers who trick careless users into signing their identities. In general, we envision two options for users to express that an identity from which they received a call is trustworthy: Either explicitly, e.g. by pressing a special button on the callee's phone, or implicitly, e.g. automatically based on statistics like the combination of call duration and call frequency.

Our system uses pre-verification of certificate chains. The disadvantage of such pre-

computation is, however, that the key database will always be slightly out of date. This implies that recently uploaded certificates will not necessarily be considered for assessing incoming calls. We regard this not to be a significant problem because it can potentially only result in further message processing by the callee (or its proxy) but not in any kind of attack.

Related to this timeliness of .wot-files is signature revocation. Revocation of certificates and signatures is a challenge in any large system which relies on asymmetric cryptography. For our WoT approach, we assume that users are able to revoke signatures. Further, we assume that such revocations of signatures can be uploaded to key-servers and are taken into account by a VKS when computing a VCCG file (i.e. if a signature has been revoked the VKS does not consider it anymore in the certificate graph). Thus, if a formerly trustworthy identity suddenly starts to send malicious messages (e.g. because a VoIP terminal has been infiltrated by malicious software), users which have signed this identity are assumed to revoke their signatures for this identity as soon as they realize that the identity is not trustworthy anymore. But as long as not all signatures for such an infiltrated identity have been removed, this identity is still connected to the WoT and may have a short assertion path to certain callees.

However, experience with email-worms shows that such malicious software usually tries to spread using the address book of infected identities [285]. Here our approach has advantages: Presumably exactly the identities in the address book of a user are the ones which have social relationships and can thus revoke certificates. In addition, short-lived signatures could limit the effect of infected hosts.

In general, we believe that other protection mechanisms (such a blacklisting or a holistic approach as suggested in [204]) should be used in conjunction with our proposed WoT approach in order to protect against infiltrated certificate chains, as may be the case with sophisticated threats such as botnets. Research has shown that in order to prevent against unsolicited communications, a combination of several mechanism is useful to compensate for false positives and false negatives of single mechanisms [204] [245].

We showed that a double-sided Breadth First path search algorithm performs well for efficient identity assertion path computation. We expect this to be true due to the small world properties of the PGP WoT networks we used. However, we only considered relatively small networks (with respect to real-world VoIP networks) with medium path lengths in our experiments as these are the largest WoT networks publicly available today. The scalability of the approach to very large WoTs still has to be investigated. A lot of research has been done in the area of (heuristic) path computations in social networks [74]. Although we did not discuss any advanced path search algorithm or heuristic, such research is likely to help in increasing the maximum number of identities and signatures which can be handled by a single server. Additionally, we believe that analysing load balancing and data distribution of certificates is interesting future research regarding the overall scalability of our approach.

6.4 Related Work

Many works consider the prevention of unsolicited communications in VoIP systems, so-called *SPIT (Spam-over-IP-Telephony)* prevention. In addition, researchers have analysed the existing PGP Web-of-Trust in the context of email communications. Below we provide an overview of existing work in both areas: SPIT prevention and Web-of-Trust research. Following this overview of related work in both areas, we then identity the differences and advances of our approach compared to state-of-the-art.

6.4.1 SPIT Prevention Mechanisms

Due to the different nature of the medium, many means very effective in blocking email spam cannot be directly applied to mitigate the SPIT threat [220]. Thus, researchers are investigating new protection techniques specifically targeted at SPIT [58]. An overview of state-of-the-art in SPIT prevention is provided in [220] and [204]. The potential legal issues involved when filtering traffic for SPIT have been outlined and discussed in [259]. In principle, one can distinguish three kinds of methods:

- *Non-intrusive methods* are solely based on the exchange and analysis of signalling messages. These methods do not create inconvenience for the caller and do not disturb the callee (if they successfully block SPIT calls). This category includes blacklisting, whitelisting, detections of call rates, call patterns, reverse lookup of caller DNS entries, context awareness (e.g. time of the day, bundling with calendar entries), and similar solutions [245] [220] [204].

- *Caller-side interaction methods* require the caller to pass a checking procedure for the call which is typically based on question and answer [205] [275] or more general on action and reaction [138] [245]. However, these methods create additional inconvenience for the caller-side (e.g. delays in establishing communication).

- *Callee interaction methods* exchange information with the callee on a per-call basis. An example is asking the callee before accepting a certain call (e.g. *consent-based* communications [219], feedback from callee [188]). These methods are problematic with respect to immediate prevention, as they miss the general goal of protecting the callee from disruptions. However, they can be useful for subsequent protection of the callee. An example for a useful callee interaction method is receiving feedback from the callee on false negatives. With this feedback, an adaptive SPIT prevention system can avoid letting a SPIT message pass the system twice.

Existing proposals for preventing SPIT are addressing the threat from different viewpoints. On a high level, one can classify these approaches as architectural proposals [187] [276] [228], protocol extension proposals [287] [188], and detailed mechanism/algorithm proposals (e.g. [205] [156] [275] [212] [138] [219] [260]).

Using PGP for SPIT prevention PGP certificates had been envisioned for signing messages in the original SIP specification [126] (now deprecated in RFC 3261 [222]). However, PGP was only envisioned as the certificate format for SIP and not for using a distributed WoT model. Zimmermann proposed ZRTP [297] as a decentralized solution for user authentication and key exchange over RTP streams. In contrary to our work, ZRTP does not consider identity assertion paths for signalling messages but only direct authentication of audio streams between caller and callee after the call has been established.

SPIT prevention in P2PSIP systems There exists very few work on SPIT prevention for P2PSIP. One exception is the approach proposed by Heikkila and Gurtov [129] which has some similarities with our proposal[11]. As in our approach, using a Web-of-Trust is proposed to protect P2PSIP users against SPIT. However, the approach in [129] simply introduces a trusted third party for certificate chain computation which is involved during every call. Moreover, this entity (called *pathfinder*) is accepted by caller and callee as trustworthy. The caller uses the external *pathfinder* service for identity assertion path computation. The *pathfinder* service returns a signed token with the path length which the caller can present to the callee. As such, the approach in [129] is merely a re-implementation of existing, stand-alone WoT identity assertion path computation (*pathfinder*) services (e.g. [88]) which gets contacted by the caller for each call, and returns a token signed by the *pathfinder* service.

Our presented approach differs in several important aspects: Most importantly, the *external pathfinder* service in [129] needs to be involved in every call setup. With our approach, the identity path computation is *integrated* into the callee (or caller), allowing the callee to assess incoming calls itself. Thus, our approach decentralises identity assertion as much as possible. We showed through prototypical implementation that this approach is technically feasible with low overhead. In addition, in our scheme only the callee needs to rely on a trustworthy key-server which is merely used offline. Furthermore, we investigate decentralised schemes with unverified certificate graphs (including certificate storage within the P2PSIP DHT) and the corresponding trade-offs with respect to performance.

6.4.2 Web-of-Trust Research

Most Web-of-Trust research focuses on the existing PGP Web-of-Trust and on the analysis of the structure of the PGP certificate graph. Capkun et al. [279] as well as Penning [199] provide a graph analysis of the PGP network. Bidder et al. propose a new method for synchronization among key-servers in a WoT [68].

Morselli et al. propose *KeyChains*, a decentralised system for computing certificate chains [182]. The approach uses a probabilistic algorithm for finding certificates and

[11]Note that the initial disclosure of our ideas and scheme [189] (later on published in [244]) has been prior to the publication of [129].

certificate chains in a decentralised fashion. However, it cannot be guaranteed that an existing certificate is found. Moreover, the approach introduces a high number of messages to be sent (i.e. in the range of 100 messages each with several hops) for finding a single certificate chain.

Capkun et al. present a decentralised public key authentication scheme for mobile ad-hoc networks [83]. In their scheme, key authentication is performed via chains of public-key certificates. Certificates are stored and exchanged among nodes in a decentralised way, i.e. no trusted, central authority is involved. Their solution focuses on mobile ad-hoc networks and on the distributed generation of certificate graphs in scenarios where nodes have limited capabilities and limited reachability among each other.

6.4.3 Progress with Respect to State of the Art

We adapt existing algorithms to derive a certificate chain between a VoIP caller and callee (or more general between real-time communication users) in *real-time*[12] and additionally propose novel techniques to achieve *decentralised* cryptographic verification of the certificate chain. Our approach thus integrates a Web-of-Trust into real-time communications such as VoIP. In addition, our scheme provides a SPIT prevention mechanism specifically suitable for P2PSIP. In particular, a core contribution of our work is the investigation and implementation of adapting a Web-of-Trust model for securing decentralised VoIP systems in a *decentralised* fashion, i.e. with as little involvement of centralised entities as possible during call setup. As such, our work is related to other work in the area of SPIT prevention but novel as it exploits the social relationships among users in a *decentralised* cryptographic scheme for assessing trustworthiness of signalling messages.

6.5 Summary and Contribution

We applied a Web-of-Trust model to real-time communications in order to secure applications such as VoIP. We showed how our approach can be applied to decentralised real-time communication systems such as P2PSIP. Our approach exploits the social relationships between end-users for detecting *solicited* real-time communications. Further, it can prevent identity spoofing attacks in a decentralised way. In summary, our system enables decentralised cryptographic identity assertion based on a Web-of-Trust model for the assessment of the trustworthiness of user identities in real-time communications.

A core contribution of our approach is the integration of a Web-of-Trust into real-time communications such as VoIP. In addition, we contribute the investigation and implementation of adapting a Web-of-Trust model for securing decentralised VoIP systems in a *decentralised* fashion, i.e. with as little involvement of centralised entities as possible during call setup.

[12] more precisely in *soft* real-time, as defined in [44]

We evaluated our proposal with a prototype implementation using real world WoT graphs. Our results demonstrate that, for short path lengths, a double-sided Breadth First path search algorithm integrated into a SIP proxy performs well enough for deriving identity assertion paths in a real-time communication scenario. To enable fast assessment of incoming calls by the callee, we introduce a specialised *Verifying Key Server (VKS)* which pre-verifies WoT certificate chains. To prevent time-consuming downloading of certificates for key-IDs, the caller can append his/her public key in a self-signed certificate to signalling messages.

We investigated the different trade-offs between decentralisation, i.e. reliance on a trustworthy key server for pre-verification of certificate chains and certificate retrieval, and additional call setup delay at the callee and caller in a P2PSIP scenario. In summary, our results show that if the callee is using a certificate graph with pre-verified certificates and regards identity assertion path lengths up to five as useful, the additional operations we introduce will delay an incoming phone call on average by less than $150ms$. If the callee is using an unverified certificate graph and certificate chain computation as well as certificate retrieval is offloaded to the caller, the callee only has to verify the signatures of the identity assertion path which can be done in less than $20ms$ for short paths. However, such schemes increase message size and may introduce a call setup delay of up to $500ms$ (potentially even more in cases where certificates are retrieved from the DHT) at the caller.

Throughout this chapter and in our prototypical implementation we exemplified our proposed scheme for applying a WoT model to real-time communications with SIP [222] and PGP [298]. Our approach is, however, general in nature and in principle applicable to any kind of signalling protocol for setting up and managing real-time communication sessions and any WoT infrastructure. Our approach only requires a (cryptographic) WoT among users which is instantiated through protocols and key-servers.

Finally, it is worth noting that our scheme for decentralised identity assertion cannot only be used for SPIT prevention or the identification of non-malicious incoming messages. Our scheme enables the detection of spoofed identities. This is a key technique to fight Spam (i.e. SPIT in the case of VoIP) because identity spoofing is used by spammers to circumvent identity blacklisting/whitelisting and other anti-Spam techniques. However, the detection of identity assertion can also help in preventing other types of attacks which utilise identity spoofing. For instance, our scheme can also be used to protect against Denial-of-Service attacks. If an attacker sends bogus `SIP-Invite` messages with spoofed SIP-URIs in the SIP `From` header, no certificate chain to that identity will be found, or the attacker will not be able to correctly sign messages for the spoofed identity. The callee can exploit this fact by not accepting calls from such identities without further challenging the caller. In summary, our scheme can be used against SPIT, but moreover it is generally applicable to detect identity spoofing attacks.

Chapter 7

Lawful Interception in P2PSIP

Contents

Lawful Interception (LI) is the process of legally authorised wiretapping of communications carried out by law enforcement organisations. It is conducted without the intercepted parties being aware of it [209]. LI has been carried out in the PSTN and GSM networks successfully for quite some time. However, the characteristics of highly decentralised P2P-systems impose unique challenges for LI to be conducted in such systems.

On a high level, a Lawful Interception activity in communication networks is initiated by an authorised legal entity which has identified a certain target identity (e.g. a mobile

phone number) of a suspect user. This target identity gets securely transmitted to the responsible service provider with an authoritative interception request for the identity. The service provider than has the technical challenge of identifying the current location of the target identity, and further—depending on the location of that identity—to determine a suitable location and entity in the network for interception. For instance, in the PSTN, the location for a given phone number (target identity) is fixed, and a central point for interception can thus easily be determined for a given phone number. In mobile networks (e.g. *GSM*), the location for a given mobile phone number may change dynamically. Still, these networks are designed in such a way that for each phone number a central interception entity can be identified by the service provider. Having identified a suitable point for interception in the network, the actual interception then consists of passively logging any kind of communication for the target identity at the interception point, and transmitting this information securely to the legal authority that triggered the Lawful Interception activity.

With P2P-based real-time communication, the identification of an interception point in the network for a given target identity becomes more complex. Not only may the target identity change its location dynamically. More importantly, the mapping of identities to dynamically changing locations is not stored at central entities but in the *itself* dynamically changing P2P overlay. It is thus not possible to identify a static point in the network that is suitable for intercepting all signalling messages for a given target identity.

This chapter provides a case study of carrying out Lawful Interception in P2P-based real-time communication systems[1]. We use VoIP, and accordingly the terms *caller* and *callee*, as an example application for P2PSIP to investigate Lawful Interception. However, our study analyses the session establishment process in P2PSIP with respect to Lawful Interception. It is thus general in nature and applies to any multimedia communication session being set up with P2PSIP, and not only VoIP. As a prerequisite, Section 7.1 gives a general introduction to the field of Lawful Interception. In addition, the challenges of applying LI in IP networks and for (client/server-based) VoIP communications in particular are examined.

There is a lot of controversy surrounding the topic of Lawful Interception in VoIP networks regarding social, political, and economic issues (see e.g. [66] [38]). However, in this chapter (and in this thesis) we analyse the problem of applying Lawful Interception to P2P-based real-time communication systems (exemplified with Voice-over-IP) solely technically. The characteristic properties of P2P-based approaches to multimedia session establishment and the corresponding implications that complicate Lawful Interception will be examined in detail in Section 7.2. Further, we inspect potential solutions for implementing Lawful Interception in a P2PSIP system in general and discuss advantages and drawbacks of such solutions (Section 7.3). Section 7.4 summarises this case study and highlights contributions.

Throughout this chapter we will exemplify technical details with Chord-figures. However, the characteristic properties and corresponding challenges for LI we highlight in

[1] A significant part of this chapter (including figures) has originally been published in [240]. See also Appendix A.

this chapter are mostly independent of Chord or the DHT's routing structure. Thus, the analysis we present is of general nature and applies to all DHT-based VoIP systems.

7.1 Introduction to Lawful Interception

Lawful Interception can provide crucial information for criminal and security investigations. LI may be used to collect information that can be used as evidence in court afterwards as well as to gather information regarding criminal or terrorist activities in order to even prevent crimes or attacks. Many countries have legislation in place which allows a *Law Enforcement Agency (LEA)* to request a communication provider (such as an Internet Service Provider or a VoIP Service Provider) to carry out a LI-operation.

Lawful Interception of *Voice-over-IP (VoIP)* communications is technically more challenging than in the *Public Switched Telephone Network (PSTN)*. One reason for this is the mobility of users, as enabled by SIP. In practise, user mobility implies that the current location of an identity cannot always be determined prior to a call. Currently, Lawful Interception is being standardised for VoIP and IP networks in general in ETSI [35] and other standards organisations [32] [34] [57][2].

In general, the more central components are on the signalling path, the more manageable LI becomes. In current approaches, service architectures are considered where central entities are on the signalling path (as this is the way VoIP and real-time communications are being deployed today). However, future types of VoIP service architectures may be characterised by a higher degree of decentralisation. For instance, if a P2P-network is used for VoIP signalling (i.e. as with P2PSIP) there is no central entity in the network through which signalling will pass. In principle, this means that not only the users, but also signalling and location servers are mobile and highly dynamic, rendering LI even more challenging.

This section introduces common Lawful Interception terminology. In addition, an overview on how Lawful Interception is generally implemented in IP networks is given. Further, some examples of applying LI to Voice-over-IP communications will be presented.

7.1.1 Terminology and Reference Model for IP Networks

A Lawful Interception activity gets triggered by a *Law Enforcement Agency (LEA)* which authorises a Network Operator, Access Provider, or Service Provider to intercept traffic for a target identity. Generally, two different types of interception data can be distinguished:

- *Intercept Related Information (IRI)* denotes the signalling data identifying the com-

[2]The IETF, however, has published a statement arguing that *wiretapping* solutions will not be standardised within its body [134].

munication. This data may comprise the source identity, the destination identity, call duration, and other signalling information.

- The *Content of Communication (CC)* denotes the actual payload being transmitted. For VoIP, this refers to the audio content of the call, i.e. the RTP-packets transferred from/to the subject which is the target of the LI operation.

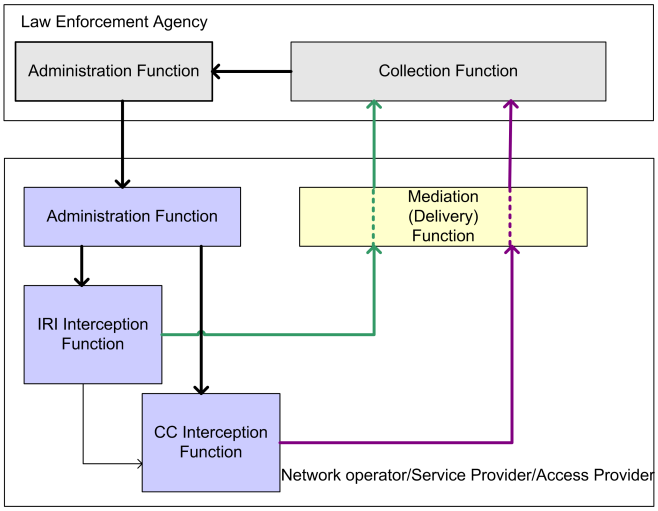

Figure 7.1: Generic Model for Lawful Interception in IP Networks

In general, when analysing various standards for Lawful Interception in IP networks (e.g. [32] [34] [35] [57]), one can extract the generic model depicted in Figure 7.1 [209]. This model shows the core functions necessary for Lawful Interception. An administration function serves for the operator to securely receive Lawful Interception requests authenticated by the LEA. This information is used by the operator to trigger the interception function for the IRI and the CC, respectively. The data collected by these interception functions is mediated and then collected by the LEA for analysis.

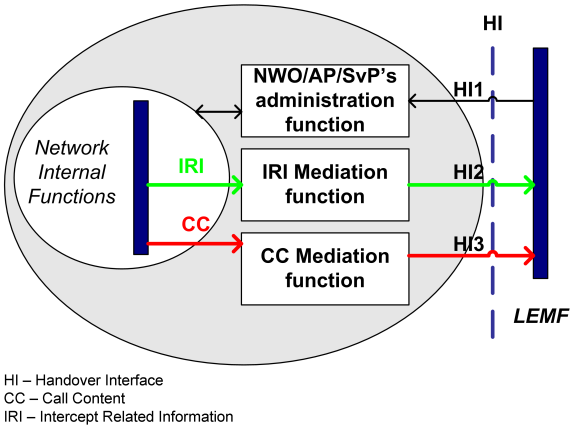

Figure 7.2: ETSI Reference Model for Lawful Interception in IP Networks

Figure 7.2 shows the reference model for Lawful Interception in IP networks as standardised by ETSI [35]. Three handover interfaces are defined (and standardised by ETSI) between the operator that carries out the LI and the LEA that authorises the LI for a specific target: an interface for the administration function *(HI1)*, an interface for the IRI mediation function *(HI2)*, and an interface for the CC mediation function *(HI3)*.

7.1.2 Lawful Interception of Multimedia Communications in Server-based Systems

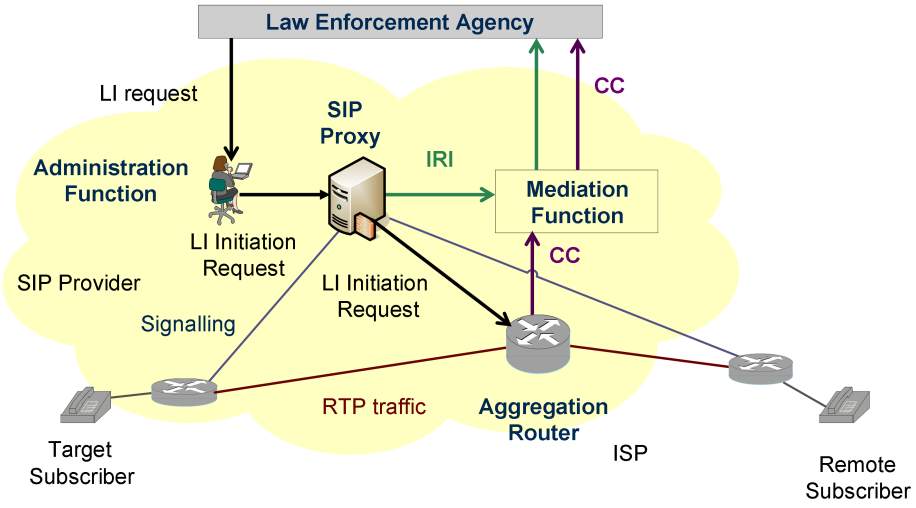

Figure 7.3: Lawful Interception of VoIP Communications

If multimedia communications such as voice are transmitted over IP networks, some fundamental differences in signalling and media transport make LI challenging in such systems. For Lawful Interception in SIP networks, the SIP-URI is the target identity of a LI operation. Some of the characteristics of SIP-signalling complicate LI. First, signalling and media traffic take different routes in the network (see Subsection 2.2.1). Second, the network provider can be different from the multimedia communication service provider (e.g. the SIP provider). Third, because SIP offers mobility to users, the network provider used by a specific target identity can change frequently. Thus, the following fundamental problem arises for LI: How to enable interception of the CC *in real-time*[3] if the network providers of the call participants are not known prior to the call?

Figure 7.3 shows how Lawful Interception can technically be implemented in a setting were the network provider is also the SIP service provider [209]. In this case, the IP-address of the target identity can be extracted from the SIP signalling messages intercepted at a SIP server of the target identity. This data is then used to trigger the interception of the CC (the media) at the corresponding aggregation router.

[3]as defined in[44]

Figure 7.4 displays a different setting where the target identity is located in a different access network than the SIP service provider [209]. However, in this setting a so-called *Session Border Controller (SBC)* is used to force signalling and media (i.e. the IRI and CC data) to traverse a single central entity. As depicted in the figure, in such a setting the SBC becomes the central element to carry out LI requests received by the SIP Service Provider. Essentially, Lawful Interception of VoIP communications is not much different from the PSTN-case in this scenario.

In a setting where no SBC is deployed and the network provider of the target identity is different from the network provider of the SIP service provider, it is necessary to extract the IP-address of either the caller or the callee from the IRI (i.e. SIP signalling messages) and then send an LI request to the corresponding network provider (as determined from the IP-address) to trigger interception of the corresponding CC (i.e. RTP packets). All of this has to be done in *hard real-time* in order to start intercepting the CC immediately and not miss parts of the conversation, which is technically very challenging. Further, it assumes a way for the SIP service provider to send an authenticated IP-traffic interception request to the network provider in *hard real-time* (which then still has to determine the corresponding aggregation router for the interception), most probably indirectly via the LEA.

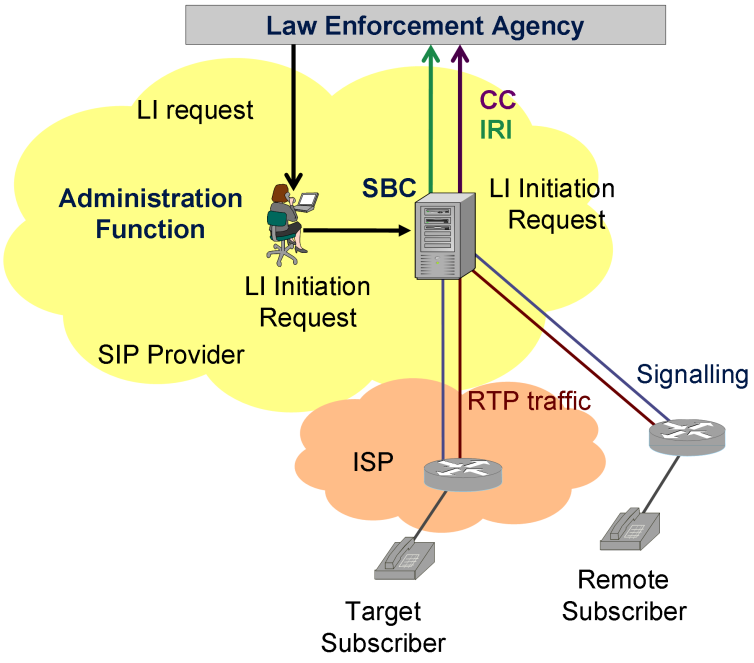

Figure 7.4: Lawful Interception of VoIP Communications with a Session Border Controller

In summary, Lawful Interception in client-server SIP systems is technically challenging. The concrete technical challenges depend to large extent on the type of service architecture being considered.

7.2 Challenges for Lawful Interception in P2PSIP Systems

Using a P2P-network implies a significant paradigm shift for real-time communications signalling, in particular with respect to Lawful Interception. First, there are no centralised components on the signalling path. Second, the network is dynamic and there are no static routing paths between entities. From a Lawful Interception perspective, these characteristics have serious implications, rendering current Lawful Interception practices used in client-server SIP systems infeasible. In this section we analyse the specific properties of P2PSIP that make Lawful Interception a challenge in such systems and highlight what the implications of these characteristics are for LI.

P2PSIP inherits (from client-server SIP with respect to LI) the problem that signalling and media not only take different routes in the network but also that the media path cannot be determined prior to a call. We therefore focus on the signalling differences in P2PSIP (compared to client-server SIP) and the consequences for Lawful Interception in our analysis. In addition to these signalling differences, any P2PSIP LI solution still has to solve the problems of deriving the corresponding point for interception of media in the network (e.g. the aggregation router of a network provider) and intercepting the corresponding CC in real-time, just as LI solutions for VoIP systems today.

7.2.1 Lack of a Central Entity for Interception

The most fundamental problem for Lawful Interception in a P2PSIP system is the lack of a central entity on the signalling path. This has two important implications: First, there is technically no single point in the network to intercept all outgoing call establishment attempts for a specific target identity. Second, there is legally no body with whom a Law Enforcement Agency can have a trustworthy and legally binding relationship for sending authorised interception requests to.

No server involved in call-setup Since there is no server involved in call-setup, for a specific target identity it cannot be determined prior to a call which nodes will be on the signalling path for outgoing calls. In contrary, for client-server SIP the first signalling hop for outgoing calls can be determined prior to a call based on the domain of the target's SIP-URI. Thus, for Lawful Interception the problem arises where to intercept signalling traffic for a target identity *at all* in the network.

No service provider to receive interception requests from LEA Lawful Interception as specified within the ETSI reference model [35] assumes an operator (e.g. the Network Operator, Access Provider, or Application Service Provider such as a VoIP Service Provider) in order for the LEA to be able to trigger a Lawful Interception activity for a specific target identity through the administration function provided by such an opera-

tor (compare with Figure 7.2). Further, it is assumed that there will be legal agreements or requirements forcing the provider to cooperate with Law Enforcement Agencies. With P2PSIP, the role of an operator can only vaguely be estimated and depends to a large extent on the use case and actual deployment of P2PSIP. As envisioned by the IETF P2PSIP working group [13], it may be the case that an operator merely fulfils the role of secure node-ID assignment, i.e. enrolment in the system. In this case, the LEA has no legal agreement with nodes involved in routing signalling messages through the network.

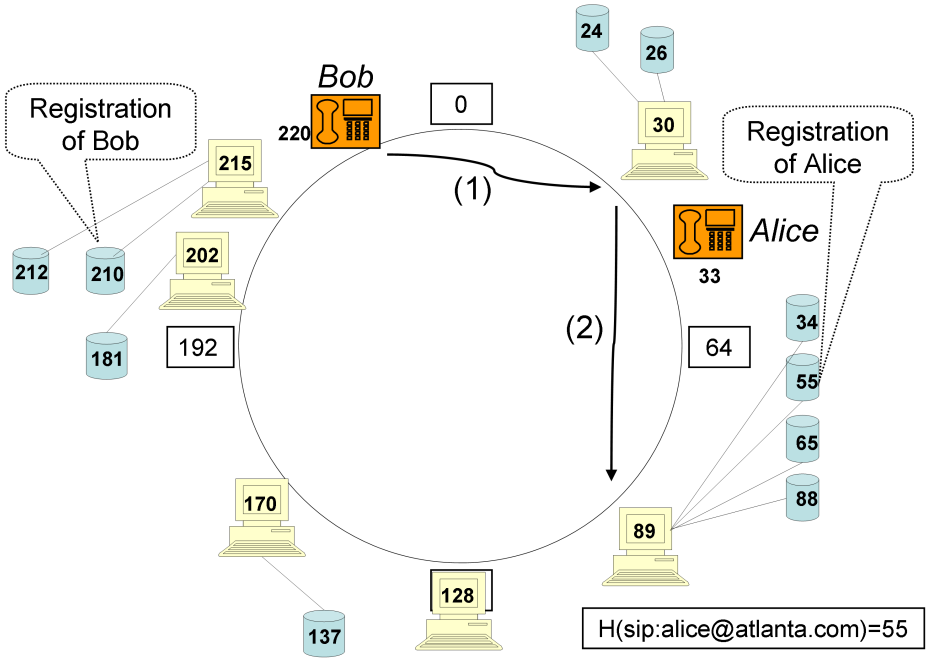

Figure 7.5: Incoming Signalling Messages Take a Different Routing Path

7.2.2 P2P-Routing

Routing in P2PSIP differs from routing in client-server SIP drastically. Essentially, for every call there is a unique routing path for signalling messages.

Inbound and outbound signalling messages take different paths In contrary to client-server SIP, with P2PSIP the last signalling hop of an incoming call is usually different from the first signalling hop of an outgoing call for a specific target identity. As an example, Figure 7.5 shows the example-DHT from Figure 2.13. However, this time Bob tries to call Alice, assuming that Alice's key-ID (i.e. the hash of her SIP-URI) is 55 and that she has stored the corresponding location previously in the network. It can be observed that in this case routing traverses a completely disjoint set of nodes than when Alice called Bob (compare to Figure 2.13). Note that the value of Alice's key-ID determines the signalling routing when Bob tries to call her, not the location of her node

(i.e. her node-ID as the hash of her current IP-address). For LI this implies that in order to intercept outgoing and incoming calls there needs to be more than one interception point for signalling messages for *each* target identity.

Different outgoing signalling node for different callee With client-server SIP, there is a static relationship between a target identity and the first signalling hop for outgoing calls. With P2PSIP, the SIP-URI of the callee determines the first signalling hop. For instance, in the example displayed in Figure 2.13 if Alice would call a different callee, say Carol whose key-ID is `88`, there would be a single routing hop to node `89` which stores the current location for key-ID `88`. Thus, there is potentially a different first signalling hop for every callee. For LI this means that even for outgoing messages there is no single point in the network where all signalling messages for a target identity can be intercepted.

These radical differences in routing compared to client-server SIP (where a single SIP server can be used for intercepting many target identities of a single domain) demand a *per target identity* solution and possibly a *per callee* solution for LI in P2PSIP.

7.2.3 Dynamic Nature of P2P Systems

Because P2P systems are highly dynamic, network membership and routing paths change as the network state (i.e. the nodes in the network, the routing links between them, and data items stored in the network) changes. Overall, this characteristic makes the previously analysed properties even worse for LI because the network state can change at any time, resulting in different routing paths and key-ID responsibility.

Joining and leaving of nodes In a P2P network, participating nodes join and leave the network frequently. As a consequence, the signalling routing path between a specific caller and a specific callee cannot be determined prior to call-setup time because it changes frequently over time. Thus, any LI attempt must derive the first signalling hop for an outgoing call attempt of a target identity dynamically in real-time.

Figure 7.6 displays again the example network from Figure 2.13. However, in this example a node with node-ID `164` has joined the network. As a consequence, if Alice calls Bob again her first lookup message would be sent to the newly joined node `164`. Also, node `170` which was previously on the routing path between Alice and Bob does not receive routing messages anymore if Alice calls Bob.

Figure 7.7 shows the same DHT as in Figure 7.6 after node `202` and node `170` have left the network (e.g. because their users went offline). Note that in this case node `215` takes over the responsibility for key-ID `181` from node `202`. Also, routing changes and node `164` directly routes to node `215` (2).

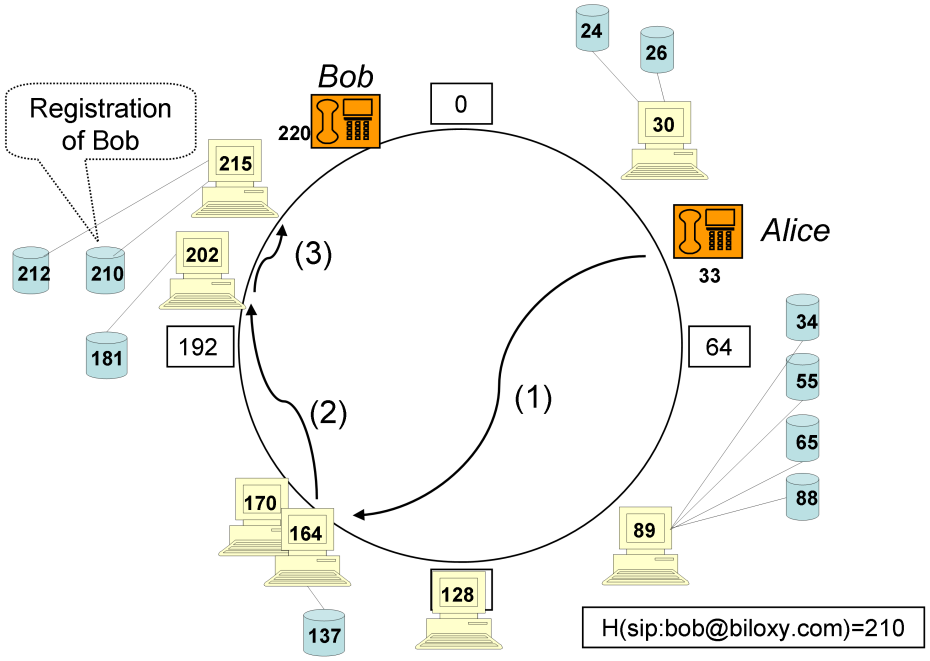

Figure 7.6: Node Join in a P2PSIP Network

Responsibility for user registrations changes frequently The node storing the location binding for a particular SIP-URI changes frequently (if a new node is responsible for a particular SIP-URI the location/identifier-binding is transferred to this node). For instance, in Figures 7.6 and 7.7 it can be observed that by joining the network, node `164` has also taken over the responsibility for storing the SIP-registration for key-ID `137`. Similarly, when node `202` left the network, data responsibility for key-ID `181` was taken over by node `215`[4].

Hence, joining and leaving of nodes affects the signalling routing path between a caller and a callee dynamically as the network state changes as well as the responsibility among nodes for certain key-IDs and corresponding data items. For LI this implies that even if there would be a relationship (or legal agreement) between the LEA and the node storing the registration data in the DHT this would be of very limited value because at any time the registration responsibility may get transferred to another node due to nodes joining and leaving the network.

7.2.4 P2P Nodes are not Trustworthy

In general, it cannot be guaranteed that P2P nodes follow the DHT operations properly. In any large network with regular hosts or even VoIP terminals it must be considered that

[4]These changes are handled by regular DHT maintenance operations to cope with the case when a node fails or leaves the network without transferring data items previously.

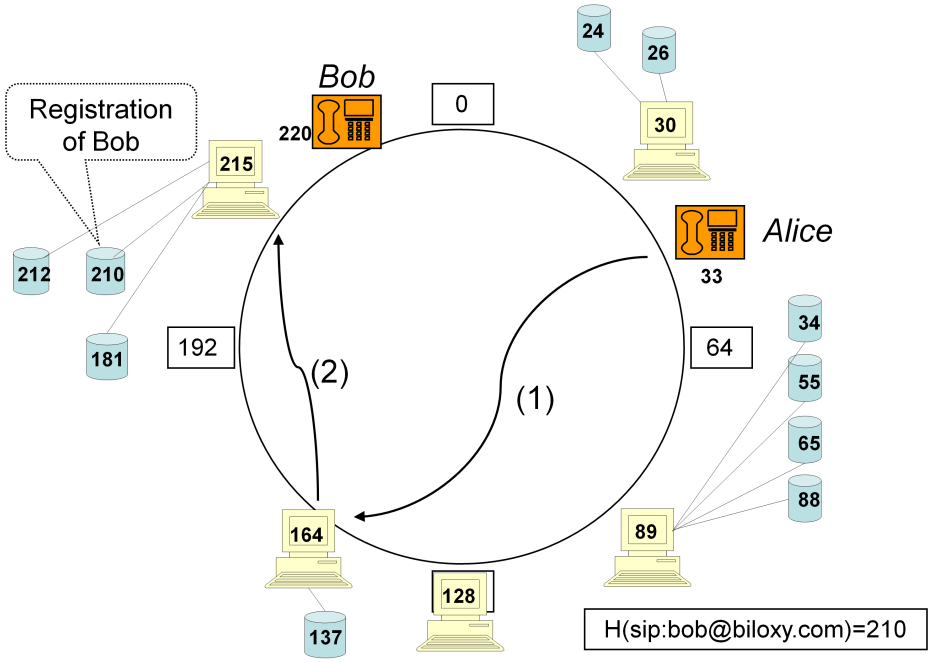

Figure 7.7: Node Leave in a P2PSIP Network

some nodes have been compromised or act maliciously. This complicates LI because with P2PSIP such potentially compromised nodes are part of signalling and store location-to-URI bindings in the DHT.

User registrations are stored at peers/terminals With P2PSIP, location bindings of users (i.e. SIP-registrations that bind the current location of a user to a SIP-URI) are distributedly stored at all participating nodes in the network, potentially at users' terminals. However, P2P nodes cannot be considered trustworthy. Also, it is possible that attackers launch a *chosen-location attack* in the DHT, trying to place a node under their control at a specific location in the virtual DHT routing structure (e.g. the Chord routing ring). Because user registrations are stored at potentially compromised and generally non-trustworthy nodes or terminals, the integrity of user registrations stored in the network cannot be guaranteed [239]. Consider the example in Figure 2.13: Bob has no means of preventing node `215` to forge the location stored for his SIP-URI.

Difficult to authenticate user registrations Besides not being trustworthy, a node storing registrations for some SIP-URI has no means of authenticating the corresponding data item due to the lack of a shared secret or trust relationship with the owner of a SIP-URI. Consider again the example in Figure 2.13: Node `215` has no means of authenticating a store-message for Bob's SIP-URI in order to detect attacks by intermediate nodes that route the store-message and replace Bob's location with a forged one.

Thus, unless cryptographic add-ons are used (e.g. self-certifying SIP-URIs, compare Chapter 5), user registrations stored in a P2PSIP network can be forged, possibly misleading Lawful Interception operations.

7.3 Potential Solutions

As highlighted in the previous section, the paradigm of P2P computing imposes significant challenges for Lawful Interception of real-time communications signalling and media that seem almost unachievable technically. Nevertheless, there are some approaches to incorporate LI even in highly decentralised systems like P2PSIP. In this section, we discuss these potential solutions on a general level, pointing out the advantages and disadvantages of each potential approach for Lawful Interception in a P2P-based SIP system.

The primary problem for Lawful Interception in P2PSIP systems is the lack of a centralised entity in the network through which all signalling traffic traverses. Thus, to still render Lawful Interception feasible, any solution needs to solve the problem of where to intercept communications in a highly dynamic system. Considering the dynamic network structure and routing in P2P systems described previously, it seems an option to move the interception point in the network for Lawful Interception from the network core towards the edge. In the discussion of possible P2PSIP LI solutions, we will therefore inspect solutions at the network edge first and then look at solutions that intercept closer to the network core.

7.3.1 Footprint in Devices

One solution that has been suggested for Lawful Interception of IP-traffic and VoIP communication in server-based systems is to implant a *footprint*—i.e. software that is secretly running on the user's hardware and controlled by the LEA[5]—in devices. Such a footprint could intercept all outgoing traffic at the source and would also have access to all incoming traffic for the target identity. Currently, many countries are considering this option. For instance, in Germany there is a discussion about trojan horses developed by government agencies to be secretly installed on hosts of target identities [38]. Of course, such footprints can also be installed prior to deployment of devices.

Advantages Since signalling and media are correlated in terminals, one key advantage of a footprint in devices would be that there is no need to trigger the interception of media traffic (at another location in the network) from the intercepted signalling, hence coping with user-mobility. For P2PSIP, this solution could also handle the constantly changing structure of a DHT. In general, having a footprint in the device of the target

[5]Technically, this approach essentially means that the LEA installs a *Trojan Horse* on the device of the user.

identity would solve the problem where to intercept in the network and also mitigate the problem of a dynamic network structure and routing.

Disadvantages The core problem with this approach is forcing terminals to incorporate such a footprint in their code. With hardphones, it might be feasible to enforce this legally prior to deployment. Still, hackers might find solutions to break firmware and thus circumvent LI. But even worse, with softphones and open standards like SIP (or potentially P2PSIP once it is standardised by the IETF) almost anyone can change the behaviour of terminals or even write a new P2PSIP application. Thus, it seems impossible to enforce a mandatory LI footprint in open systems with open standards. Future work in this direction might explore trusted computing and smartcards as hardened platforms that could protect the integrity of P2PSIP applications including a pre-installed footprint (still not preventing self-written applications).

7.3.2 Intercepting at IP-Layer

Because P2PSIP essentially replaces SIP servers with a highly dynamic and unpredictable interconnection of P2P-nodes, a possible solution for LI could be to intercept all traffic of the target identity at the IP-layer. This would assume stateful packet inspection on the IP-layer, filtering SIP messages, and then extracting signalling information in order to trigger interception of the CC. Further, it would assume that the LEA knows the network operator of the target in order to authorise LI through the administration handover interface.

Advantages In a setting where the target identity always uses the same network operator this may be a feasible approach. In such a setting, the target identity might get assigned a different IP-address frequently (as is common for home users connected via an ISP) which would result in a new DHT position on the signalling layer (remember that most DHTs compute a node-ID by hashing the node's current IP-address). Since the network operator usually authenticates the target identity on the IP-layer, it seems more practical to intercept at the IP-layer in this setting as described above.

Disadvantages In a scenario where the target of a Lawful Interception operation changes its network operator frequently (due to mobility on the IP-layer as specifically offered by SIP), this approach becomes very challenging. First, in order to intercept on the IP-layer in this case the LEA would need to inform *virtually all* network operators in its legislative domain and provide them with the target identity (i.e. the SIP-URI of the target). Second, all network providers must have the technical ability to correlate CC interception with SIP signalling extracted from filtered SIP messages.

Overall, it seems that this approach is hard to deploy in practise. If users change the IP access provider (e.g. by using wireless networking hotspots) frequently, this approach

demands for a completely correlated surveillance of users on the IP-layer among network operators. Such a comprehensive Lawful Interception system would be very expensive, possibly imposing large costs to network/access providers.

7.3.3 Infiltrating the Peer-to-Peer Network

In order to make LI feasible not at edge nodes but in the network core, the only option seems to infiltrate the network with nodes under control of the LEA. Note that this would imply an important paradigm shift to LI carried out today where the LEA does not actively conduct LI but rather sends authorised LI requests to operators which are legally bound to carry them out.

It is well known that this approach is pursued by the music industry in order to track illegal uploading of music in P2P file-sharing networks. However, there is an important difference to Lawful Interception: the goal of the music industry is to find *some* illegal file-sharing activity, whereas the goal of Lawful Interception is to intercept *all* traffic for a *specific* target identity. Since even the control of a very large amount of nodes cannot guarantee being on every possible signalling path, effectively the LEA would need to place itself at specific locations (depending on the DHT structure and properties of the particular P2PSIP network) in relation to the target identity's node-ID and key-ID in the network. Essentially this meant that the LEA becomes a kind of attacker carrying out *chosen-location attacks* for the target identity.

Intercepting incoming signalling messages by guarding the key-ID For instance, with an unidirectional DHT such as Chord [267], the LEA could control all access to a specific key-ID by controlling a node[6] with a node-ID marginally smaller then the key-ID (i.e. very close in the unidirectional DHT routing structure). For other DHT protocols (e.g. CAN [210] or Pastry [223]), the DHT structure determines *closeness* to a key, respectively. As an example, Figure 7.8 shows the DHT from the previous figures after a new node (assumed to be under the control of the LEA) has joined with node-ID `209`. By becoming the direct predecessor of the node responsible for storing data for key-ID `210`, due to routing properties specific to Chord (i.e. unidirectional greedy routing) [247] the LEA can control all queries for key-ID `210` (i.e. all incoming signalling messages for Bob's SIP-URI). This would enable the LEA to intercept incoming signalling messages for a specific target identity. Because the key-ID depends on the SIP-URI of the target, such an approach is relatively independent of network dynamics as long as the target does not change its SIP-URI.

Intercepting outgoing signalling messages by infiltrating routing tables To intercept all outgoing messages from a target identity on the signalling layer, the LEA would need to infiltrate all DHT routing table entries of the target identity's node in the DHT. Since the target identity gets assigned a new node-ID whenever the target

[6] e.g. by joining the network with such a node-ID

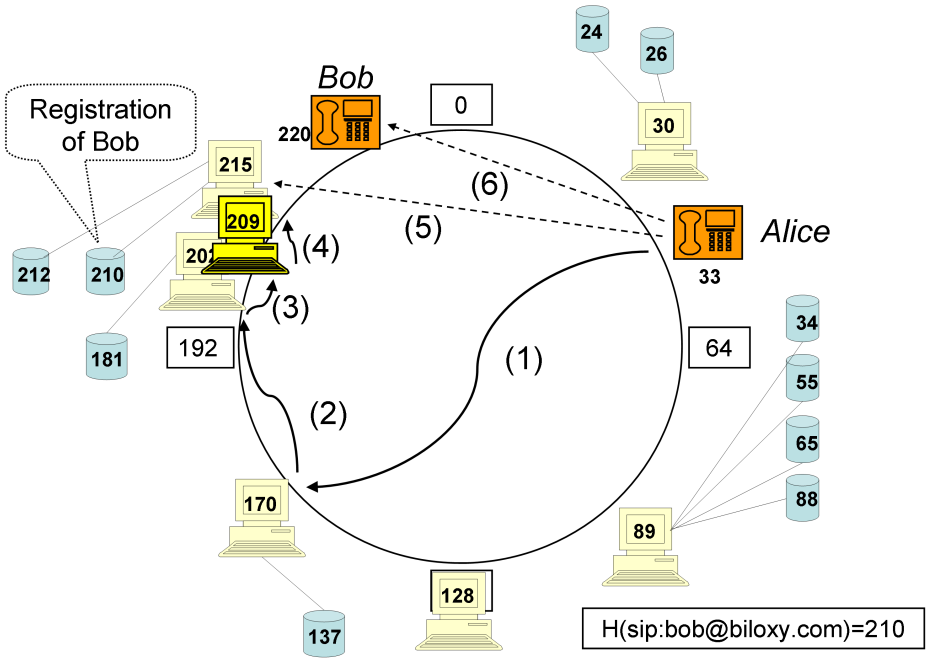

Figure 7.8: A Node Under the Control of a Law Enforcement Agency Joining a DHT in order to Intercept All Incoming Signalling Messages for a Target Identity

obtains a new IP-address, this seems unjustifiable effort. Additionally, for some DHTs such *routing table poisoning attacks* [84] are hard to conduct due to *constrained routing tables* [251] (see Chapter 3). Hence, due to the rapidly changing network structure in P2P-systems, it is almost impossible for a LEA to infiltrate all routing table entries of a target identity's DHT-node.

Using an enrolment server of the operator If an enrolment server is operated by an operator which has a legal agreement with a LEA, joining the network at some specific location and infiltrating routing table entries of a target identity may become feasible for the LEA. An enrolment server could take out the effects of network dynamics by assigning the same node-ID to a target identity even in the case of mobility on the IP-layer (in [63] an approach for such a special DHT enrolment-service is presented), assuming participants authenticate themselves at the enrolment server. Potentially, this could enable a LEA to intercept all outgoing calls for an identity even in a highly dynamic network because a particular target identity would have the same node-ID in the DHT even if it changed its IP-address. Therefore, using an enrolment server could mitigate some of the problems in infiltrating a P2P network mentioned previously. This opens an opportunity for *bootstrapping* Lawful Interception with specific nodes that infiltrate the network at specific locations as demanded by the LEA (with respect to the target identity).

7.4 Summary and Contribution

This chapter provided a technical analysis of Lawful Interception in P2P-based real-time communication systems. The key properties of P2P systems that impose challenges for Lawful Interception have been inspected in detail: The lack of a central entity, changing participants and varying data responsibility, P2P-routing characteristics, and the non-trustworthiness of nodes. Further, potential solutions to these problems have been presented and examined, including a Lawful Interception footprint in devices, intercepting all communication on the IP-layer, and actively infiltrating the P2P network. The presented case study contributes a detailed technical analysis of Lawful Interception in P2P-based multimedia communication systems.

Existing Lawful Interception architectures have been designed with traditional telecommunication networks in mind. This model maps to IP-based networks, as long as there is a service provider with central signalling entities that know about the locations of its users. With P2P systems, this model may change. Potentially, the P2P network will not be controlled by an operator. Instead, the P2P overlay may form itself, and change highly dynamically all the time.

None of the presented solutions seems to be a very promising and satisfactory approach for implementing Lawful Interception in P2PSIP networks at large, each having its own drawbacks as well as practical issues. Therefore, at the current state, providing a general solution for Lawful Interception in P2P-based real-time communication systems must be regarded as technically highly challenging and it is an open research problem how to circumvent the problems inspected in this chapter.

The preferable approach for Lawful Interception in a P2PSIP network largely depends on the exact properties and use case of the system. It remains to be seen which type of service architecture will prevail for P2PSIP and to what extent a potential operator will be involved in signalling at all. If a P2PSIP network is *operated* by a provider which is in charge of enrolment and node-ID assignment, there is an opportunity for Law Enforcement Agencies to bootstrap Lawful Interception from this procedure. Further research needs to be done in order to investigate precise requirements for such an approach and its feasibility.

Chapter 8

Proof-of-Concept Prototype of Security-Enhanced P2PSIP System

Contents

In this chapter, we present a prototype implementation of a P2PSIP system which has been enhanced by some of our proposed security mechanisms[1]. We implemented several of our secure DHT routing algorithms and investigated through experiments their effect on the success rate of lookups and the number of DHT routing hops in an infiltrated P2PSIP network. In particular, our proof-of-concept prototype enables the use of Chord DHT routing enhancements such as *backtracking* and *Direct Replica Routing (DRR)* (as presented in Chapter 4). Further, we implemented *self-certifying SIP-URIs* (as presented in Chapter 5) and the corresponding verification of data items by P2PSIP nodes. In addition to these previously presented techniques, our prototype enables the evaluation of an application-intrinsic social network as a countermeasure to adversary nodes on lookup paths.

[1]Parts of this chapter (including figures) have originally been published in [248]. See also Appendix A.

There exist other P2PSIP prototypes [18] [59] as well as general simulation studies of DHT-based P2P networks under attack [84] [106] [247]. Different from these works, we present a P2PSIP system that can *emulate* malicious nodes. Our prototype constitutes a P2PSIP implementation that enables to emulate adversary nodes in an automated way. Further, our emulation setup allows the injection of large amounts of automatically generated SIP traffic (i.e. SIP-Invite requests) into the network. This enables the measurement of the corresponding success rate of reaching the desired callee as well as estimating the average call-setup time while the network is under attack in an actual P2P network.

Overall, the prototype architecture we present provides the ability to study a P2PSIP network during a Denial-of-Service (DoS) attack on the DHT-routing layer and the effect of corresponding security techniques in a real environment. As a general benefit, our system provides the ability to analyse attacker behaviour as well as novel security techniques in an actual P2PSIP environment with comparably low effort.

Section 8.1 presents the rationale that led to our emulated system and an overview of our implementation. The security techniques we implemented to test their effects on lookup success in an actual P2PSIP network are described in Section 8.2. The experiments we conducted and the corresponding results are detailed in Section 8.3. Section 8.4 discusses advantages and limitations of our prototype architecture and concludes this chapter with a summary of our prototype implementation work and the experimental results we obtained.

8.1 Design and Implementation

8.1.1 Design Considerations and Requirements

The goal of our prototype is to evaluate security techniques. Thus, in contrary to other approaches [18] [59], we do not put special focus on SIP-protocol details or current standardisation proposals for P2PSIP and DHT messages [140]. Instead, we envision a simple P2PSIP implementation which follows the generic architecture depicted in Figure 8.1: To use a DHT for locating SIP users, the SIP-URI is the key and the corresponding data item stored in the network is the current location of the corresponding user.

The example in Figure 8.1 displays a Chord DHT in a schematic view where a user Alice wants to call another user Bob whose key-ID (i.e. the hash of his SIP-URI) is `206`; Alice's and Bob's user agents have joined the network with node ID `33` and `231`, respectively[2]. Note that in such a general architecture, DHT routing (step 3 in Figure 8.1) and retrieving a SIP registration from the DHT root node (step 4 in Figure 8.1) are independent from SIP session establishment with the callee (step 5 in Figure 8.1). This enables to implement DHT routing independently from SIP messages in our implementation. Most importantly, we do not assume any specific protocol for the DHT routing layer, nor do we impose any specific way of how DHT routing interacts with the SIP-layer of the

[2]See Subsection 2.3.3 for more details on P2PSIP.

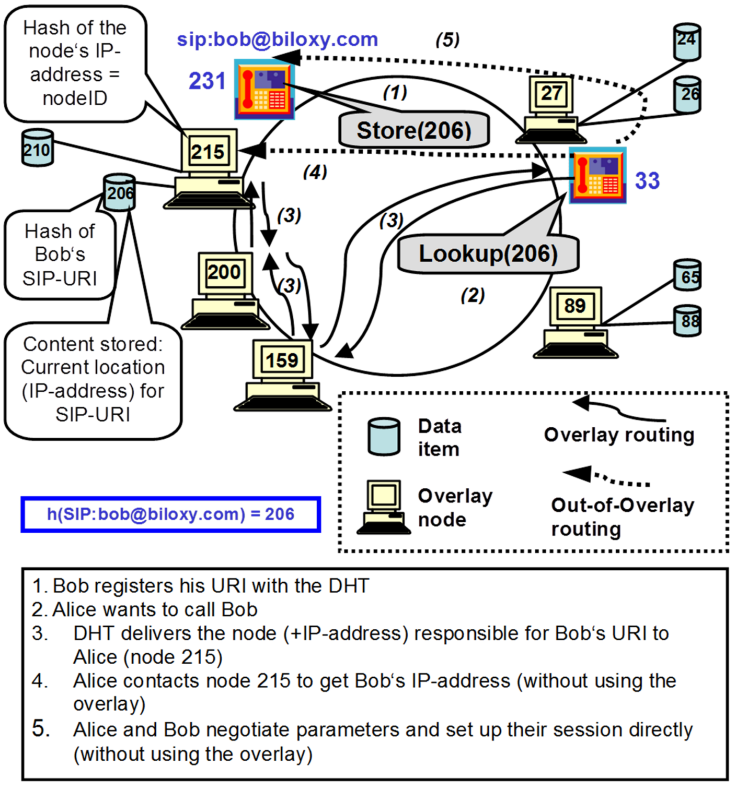

Figure 8.1: Generic P2PSIP Architecture Followed by Prototype

caller. Therefore, our results are of general nature and apply to any P2PSIP approach that follows the schematic signalling flow depicted in Figure 8.1.

To study P2PSIP security in a generic way, we envision an architecture where every node in the DHT represents a single SIP-client (i.e. SIP user agent) with a single SIP-URI. This setting enables us to generate uniformly distributed session establishment attempts and to statistically measure lookup success. This is not a limitation because other architectures where each DHT-node represents more than one SIP user agent can be seen as an extension of this setting.

To assure proper functioning of the prototype, the establishment of a SIP session and corresponding phone call between two regular SIP clients must be possible and tested. However, in order to run experiments with a large number of phone calls (to measure the average success rate of lookups), it is not feasible to start calls by hand. Thus, a key requirement is that a large amount of SIP call establishment attempts can be generated automatically. Further, to have realistic non-biased experiments, it is necessary that generated calls are triggered distributed uniformly over all participating nodes (as origin) and all participating keys (as target) in the network. While it is questionable if P2P IP Telephony calls are distributed uniformly among users, this enables to measure the success rate as the average lookup success rate in relation to all nodes in the network.

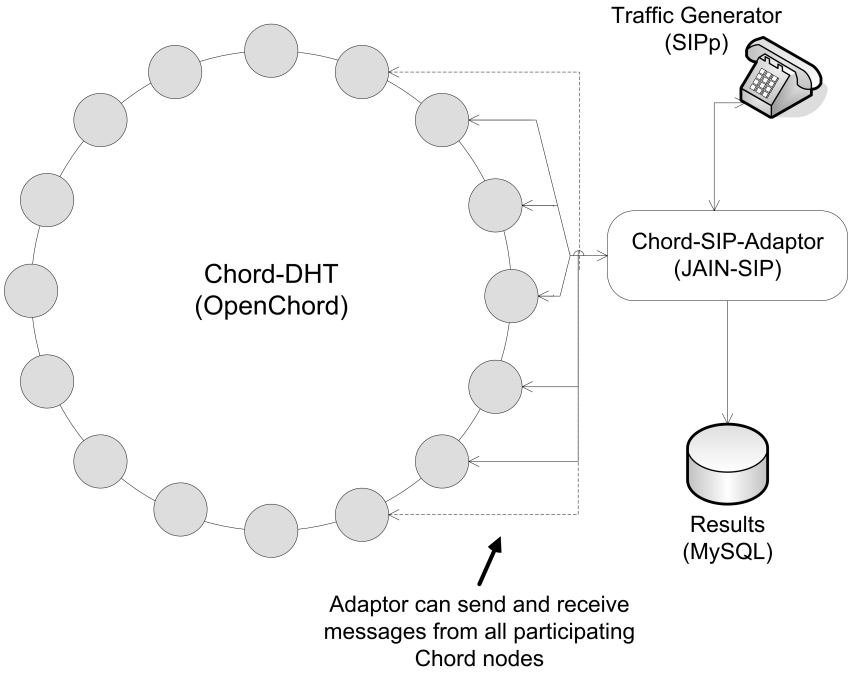

Figure 8.2: Architecture of P2PSIP Security Testing System

8.1.2 Implementation

Similar to P2PSIP standardisation [140], our implementation is based on Chord [267]. We used *OpenChord* [12] as the core DHT and implemented a DHT/SIP-Adaptor with *Jain-SIP* [9]. Figure 8.2 displays the overall architecture of our system. As depicted, the single DHT/SIP-Adaptor can send and receive overlay messages to/from all nodes currently in the network. Thus, the adaptor *controls* an experiment which consists of a multitude of SIP-requests from various callers to various callees. Messages are generated with *SIPp* [24] and sent to the adaptor which triggers the corresponding DHT request. The result of a DHT lookup as well as statistical data is stored in a MySQL database by the adaptor before it passes the corresponding SIP message on to the SIP user agent client (SIPp) which initiated the location lookup.

The adaptor model allows for testing of new security techniques without changing the SIP-stack or firmware of phones[3]. Additionally, with this setup the single adaptor can measure results and statistical data for analysis of the behaviour of the system. We also tested several regular SIP hard-/softphones with this setup and could successfully establish calls, demonstrating the overall functionality of the system.

Figure 8.3 shows the message flow for a SIP Invite request and the interaction between SIPp and the DHT-Adaptor. When the adaptor receives a SIP `Invite` request, it fetches the callee's SIP-URI from the `To-header`, hashes it to determine the corresponding DHT key-ID, and starts a DHT-lookup at a node randomly selected among the non-adversary

[3]After having initially proposed SIP as the DHT-routing protocol, the IETF P2PSIP working group also has agreed on not having SIP as the DHT message format [140], as in our architecture.

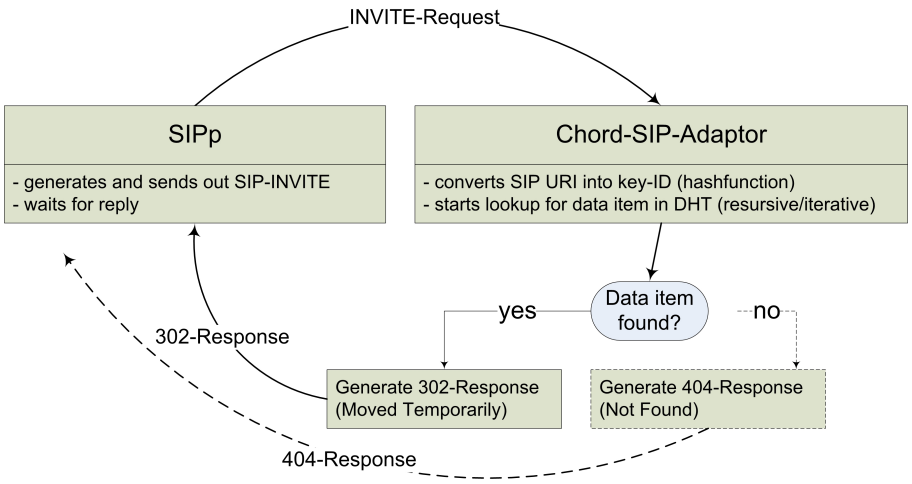

Figure 8.3: Message Flow between Message Generator and SIP/DHT-Adaptor

DHT-nodes. If the DHT successfully returns the node responsible for storing the callee's location, the adaptor retrieves the location of the callee and signals this location to the caller (SIPp in our setup) via a SIP `302-moved temporarily` response code. Using a `302-moved temporarily` response code ensures compatibility with existing SIP-clients and at the same time cleanly encapsulates the DHT-lookup service for measurements. If the lookup does not succeed, a `404-not found` response code gets returned to the caller. In both cases, success/failure of the lookup, the total time needed, and the number of hops used get stored in a MySQL database by the adaptor. More details on the architecture of the prototype and its implementation can be found in Appendix B.3.

8.2 Security Techniques Evaluated

To evaluate how potential countermeasures can positively affect lookups in a real P2PSIP network under attack, we implemented some of our security enhancements to Chord as well as self-certifying SIP-URIs as a cryptographic solution for protecting the integrity of SIP registrations in the SIP/DHT-adaptor. In addition, we implemented a variant of *Social Path Routing (SPROUT)* [167] that exploits the application-intrinsic social network among P2PSIP users. These techniques can be turned on/off during individual experiments to evaluate each technique's merit individually.

The main goal of our prototype is to investigate attacks on DHT routing. We are interested in the availability of the lookup service under real-time constraints, i.e. the ability for a caller to receive the current location for a particular callee during a Denial-of-Service attack on routing on the DHT layer by participating nodes in the network. In addition, our prototype intends to confirm the concept of self-certifying SIP-URIs and to show that this concept incurs very low performance overhead.

Attacker model We assume a P2PSIP network where *secure node-ID assignment* is deployed against Sybil-attacks [109] (e.g. where a single attacker tries to generate multiple virtual node IDs) and against so-called chosen location attacks (in which attackers deliberately choose a location in the DHT to cut off traffic from specific target nodes). This property can be achieved through a central authority like an enrolment server. As an alternative, there exists formal work to achieve this property [114] [53] as well as other more practical solutions [98] (compare Chapter 4).

Secure node-ID assignment prevents some important attacks on DHTs. However, it is important to acknowledge that even under this assumption adversary nodes can degrade the availability of a DHTs lookup service significantly [247]. For instance, consider the case were nodes that had been previously authenticated by an enrolment server are infected with malicious software such as a trojan horse or as part of a bot-net. In this case, a single attacker can launch a Denial-of-Service attack on the DHT's lookup service by having all nodes under his control block legitimate lookup requests in the network. In this scenario, secure node-ID assignment merely assures that attacker nodes are *distributed uniformly* in the network [98]. In general, an enrolment server (or similar solutions) prevents an attacker from joining the network with a large amount of nodes or at some specific location but it does not prevent nodes that have successfully joined the network from acting maliciously.

Under these assumptions, we consider the following attacker model: A P2PSIP network of N nodes has been infiltrated by $f \times N$ ($f < 1$) malicious nodes (e.g. because they have been compromised) which drop messages instead of routing them to their destination. Malicious nodes not only drop messages but also do not return data items (for keys under their responsibility) upon request. Adversary nodes may be controlled by a single external attacker, but they do not actively collude with each other during DHT operation. Attacks on the IP-layer are outside the scope of our implementation; we concentrate solely on the DHT-routing layer.

Iterative routing and backtracking Iterative routing enables the query node to verify that queries make progress towards the key and is thus preferable from a security perspective [235]. We implemented an extension of iterative routing where at each routing step the whole routing table and direct successor list (instead of just a single next-hop node) of the hop node gets returned to the query node. We refer to this extension as *Complete-knowledge iterative routing*, as described in Section 4.3. This technique has the advantage that at each routing step the query node can decide on the next hop based on an advanced secure routing strategy (e.g. use *Direct Replica Routing* [247] or *Social Path Routing* [167], see below).

Further, we enhanced iterative Chord routing with backtracking. When a node does not reply to a routing request after a certain time-out, it is assumed to be malicious (in our prototype). In this case, the lookup continues by choosing the second best node from the previous routing hop according to the current routing strategy.

Direct Replica Routing In Chord, data items for keys are replicated at $r-1$ nodes directly succeeding the node responsible for the key-ID in the ring structure. However, in regular Chord this replication is used only for reliability in the case that the node responsible for the data item fails or does not leave the network properly. In particular, if the responsible node for a key is malicious it can deny any access to the corresponding data item. Even worse, due to Chord's unidirectional greedy routing not only the node responsible for storing some content but also its direct predecessor in the Chord-ring can deny access to data items[4] [246].

To circumvent these problems we have proposed *Direct Replica Routing (DRR)* [247] (see Chapter 4). With this extension, nodes can route directly to nodes storing replicated data items without depending on the node responsible for a particular data item. This is achieved by using the direct successor list nodes maintain for reliability in routing. If during an iterative routing hop the direct successor list received contains a node larger than the key-ID, this node might store replicated content (depending on the replication parameter r) for the key and can thus be contacted directly by the query node to get the corresponding data item.

Self-Certifying SIP-URIs In a P2PSIP network with untrustworthy nodes it is crucial that the integrity of content retrieved from the network (i.e. a SIP-URI/location binding) can be verified. Otherwise, adversary nodes on the routing path or responsible for storing data items can manipulate content. In P2PSIP this means that attackers could redirect phone calls to hosts under their control by replacing the IP-address which is returned as a result of a lookup.

We have proposed *self-certifying SIP-URIs* [239] as a decentralised solution to ensure the integrity of content in P2PSIP (see Chapter 5). With this approach, SIP-URIs are cryptographically generated by hashing a public key. This enables the owner of a SIP-URI to cryptographically sign location-bindings stored in the DHT with the corresponding private key. By appending the public key to registrations stored in the DHT, any node that retrieves data items can verify that the public key belongs to the SIP-URI requested (by hashing the public key) and then use the public key to verify the location stored for that SIP-URI.

In our prototype system all participating nodes use self-certifying SIP-URIs and are capable of verifying content retrieved from the network cryptographically. Only signed registration bindings are stored in the DHT. This ensures that the integrity of SIP registrations stored in the P2P network can be cryptographically verified by any node, rendering man-in-the-middle attacks on data items stored in the DHT infeasible.

Application-intrinsic social network routing Marti et al. have suggested to use an external social network in conjunction with DHT-routing [167]. The assumption is that nodes with whom the query node has a transitive link in the social network can be assumed (at least with higher probability than the average DHT node) not to be

[4]We refer to this as the *Shield Problem* in Chord, extensively discussed in Chapter 4.

malicious and behave according to the DHT protocol. With this approach (called *Social Path ROUTing*, *SPROUT*) lookups try to route along nodes with whom the query node has a transitive relationship in the social network, as these hop nodes are considered to be non-adversary.

Instead of an *external* (in the sense that the social network application is independent of the DHT application) social network (such as *Facebook* [25] or *Xing* [26]), we assume that P2PSIP nodes build an *application-intrinsic* social network over time. Often times social relationships exist between real-time communication partners. For instance, some VoIP users may frequently call known friends or family members. This enables to create social links based on the interaction among users *within* the P2P application itself (e.g. VoIP calls). These social links could either be created manually by users (e.g. by pressing a button during a phone call with a known and trustworthy callee) or automatically (e.g. based on certain criteria such as exceeding a minimum calling time or a certain calling frequency for a callee). In either case, the social network is created based on transactions *within* the application. We use the term *application-intrinsic* social network for these kinds of *application-interaction* based social links among nodes. Figure 8.4 exemplarily shows social relationships among users that use a P2P network[5]. Note that users B and C have a direct link on the social layer, but not on the DHT layer (they are connected by node D on the DHT layer).

In our prototype, we implemented two variations of an *application-intrinsic* P2PSIP social network. The underlying DHT-routing of both schemes is based on the SPROUT algorithm published in [167]: Nodes route preferably to the social link which is closest to the key-ID on the DHT layer. If such a node does not exist (i.e. there is no social link node closer to the key), regular Chord routing is resumed. First, we implemented application-intrinsic social network routing where at each experiment a uniformly distributed social network is generated on top of the DHT. Each non-adversary node maintains links to ϕ randomly chosen non-adversary nodes in the social network, maintained in a routing table $T_{soc}()$ at each non-adversary node. These social links are preferred in routing as long as queries make progress towards the key (i.e. we preserve a unidirectional DHT routing structure). We refer to this as *static* application-intrinsic social network routing (*AISNR-s*).

One underlying assumption for static AISNR is that nodes can maintain connections to their ϕ social links despite leaving and re-joining the DHT by these nodes. To maintain social links despite churn, nodes could frequently lookup the corresponding SIP-URIs in the DHT. Churn is not emulated in our prototype, so the ϕ links in $T_{soc}()$ at each non-adversary node remain valid during the course of one experimental run.

Second, we enhanced *AISNR-s* such that nodes add other nodes to their routing table $T_{soc}()$ over time. We refer to this as *dynamic* AISNR (*AISNR-d*). With *AISNR-d*, after each successful DHT lookup, the query node, n_q, adds the callee's DHT node, n_{callee}, as well as the root node (or replica root node in case of DRR), $root_k(callee)$ (which returned a correct data item for the callee's key, k_{callee}), to its social routing table $T_{soc}(n_q)$. Further, also the callee's DHT node (i.e. n_{callee}) adds the caller's DHT node (i.e. n_q) to

[5]Figure 8.4 is a modified version of the original figure published in [241].

its social routing table $T_{soc}(n_{callee})$. This simulates a successful call establishment where both parties (either manually or automatically triggered) add each other to their social routing table. In addition, the caller adds the DHT root node (or replica root node in case of DRR) to its social routing table, to account for the fact that this node returned a correct data item for the lookup.

Note that in case a malicious node is inadvertently added to $T_{soc}()$ not much harm is done to future lookups. First, such a node can merely delay lookups (by a single hop in the attacker model of our prototype). Second, in case a node $n_m \in T_{soc}(n_q)$ does not answer to messages or returns falsified data items (i.e. turns out to be malicious), a query node n_q can delete it from $T_{soc}(n_q)$ at any time. In our prototype implementation, only non-adversary nodes are added to $T_{soc}()$, as only calls to non-adversary nodes are emulated and only non-adversary nodes return correct data items to DHT requests.

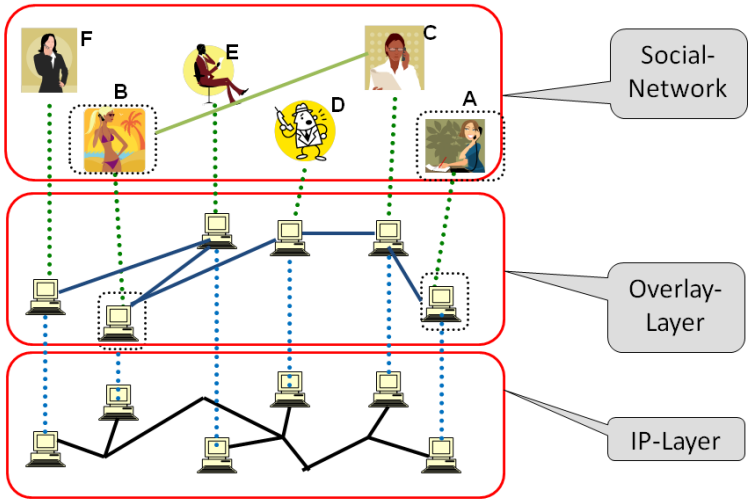

Figure 8.4: Application-Intrinsic Social Network in P2PSIP

8.3 Experiments and Results

Using the emulated P2PSIP system described previously, we conducted a multitude of experiments with various attacker rates. We investigated the success rate of regular Chord routing as well as different combinations of security techniques as add-ons to iterative routing. Additionally, we looked at the effect of the implemented techniques on the average hop count for lookups. Below we present aggregated results that visualise and summarise the effects of our algorithms that we observed. Additional and more detailed results can be found in Appendix B.3.

8.3.1 Experimental Setup

The emulated system we designed can in principle be distributed on several physical machines with many DHT-nodes running on different ports on each host. All our experiments were conducted with all nodes running on one or two Windows-XP 64-bit machines[6]. Node-IDs were computed by hashing the combination of IP-address, port, and a random number. By using a random number we emulated secure node-ID assignment and thus uniformly distributed node-IDs in our system.

In our experiments a network consisting of N ($N = 200, 400, 600$) random nodes was generated with non-adversary nodes and adversary nodes distributed uniformly over the node-ID space. As each node in the DHT represents a SIP user agent in our model, additionally N arbitrary self-certifying SIP-URIs were generated, each of these was associated with a DHT-node, and then stored in the network (i.e. the IP-address/port of the corresponding DHT-node was stored for each SIP-URI). After the network had been built and keys had been stored, a multitude of arbitrary key lookups (i.e. `SIP-Invite` requests with a key randomly selected from the previously stored values) were generated and sent to the adaptor. These `Invite`-messages were only sent out from SIP-URIs belonging to non-adversary nodes and contained an existing (self-certifying) SIP-URIs as the callee. For each setting, we conducted 10 experiments (each with a randomly generated network and SIP-URIs) consisting of at least 100 random lookups (we conducted 100 lookups to evaluate *AISNR-s* and 500 lookups to evaluate *AISNR-d*).

The following settings were used in all experiments (our implementation allows changing each of these parameters): We set $r = s$ and $m = 160$ (using SHA-1 as the hash function). All our experimental runs were conducted using *key lookup* and with a scarcely utilised key space (compare with our recommendations for Chord simulations in Section 4.6). We varied the following parametrisations in our experiments:

- s, the size of the successor list used for Direct Replica Routing
- N, the total number of nodes in the network
- f, the attacker rate
- ϕ, the number of application intrinsic social links each node has initially in the network

Our prototype logs the total time needed for each lookup. However, since in our experimental setup a DHT consisting of several hundred nodes is emulated on just a few hosts connected on a local area network, the absolute time we measured is unrealistic to assume for wide area network deployments. Our time measurements therefore cannot be used to estimate real-world lookup times for DHTs where nodes are distributed over a wide area network or even the Internet. Instead, we use the number of hops a lookup takes to succeed (also measured by our prototype) as a metric for lookup time. Depending

[6]See Appendix B.3 for more details regarding our experimental setup.

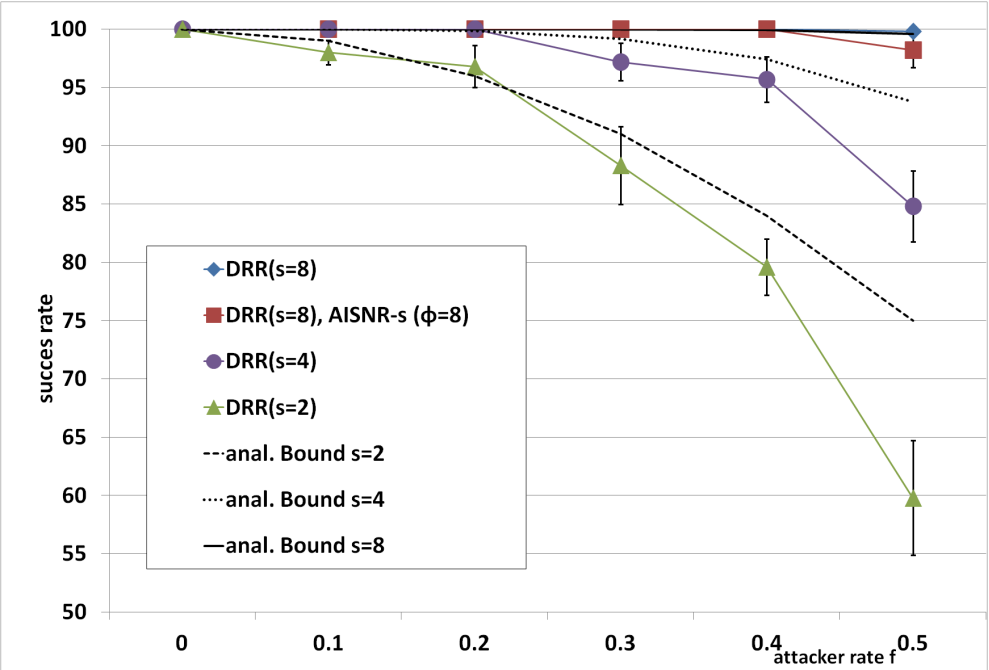

Figure 8.5: Success Rate for *DRR* and *DRR* Combined with *AISNR-s* for Different Values of s ($N = 200$)

on the properties and node distribution of an actual DHT, our hop count can roughly be mapped to lookup time by multiplying it with the estimated average Round Trip Time (RTT) in a given setting. For instance, for an Internet-scale DHT, one DHT hop can very roughly be estimated with $200ms$ (i.e. the average Round Trip Time on the Internet [7]). Thus, an average hop count of less than 10 hops (as in all our experiments where $f \leq 0.4$, see Appendix B.3 for detailed results) is acceptable for real-time communication session establishment even on an Internet scale (as it could very roughly be estimated with a session establishment time of $2s$, i.e. less than $8s$ as regarded to be acceptable by the ITU-T for phone call setup delay [30]).

In each experiment we conducted, self-certifying SIP-URIs have been used and the corresponding cryptographic operations (i.e. signature generation before storing data items in the DHT and verification of signatures after retrieval of data items from the DHT) have been performed by nodes. Our experiments thus confirm the practicality of this concept in a running system. The time needed for the cryptographic operations was measured in initial experiments and turned out to be very negligible. Further, these operations do not depend on any of the parametrisations we varied in our experiments. We thus did not log the time needed for self-certifying operations individually for the experiments we conducted.

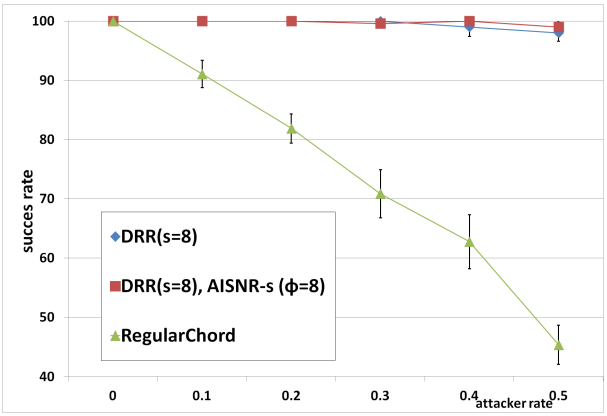

Figure 8.6: Success Rate for *DRR* and *DRR* Combined with *AISNR-s* Compared to Regular Chord ($s = 8, N = 400$)

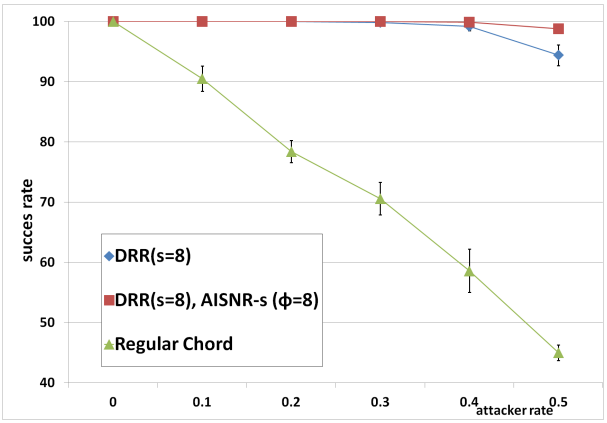

Figure 8.7: Success Rate for *DRR* and *DRR* Combined with *AISNR-s* Compared to Regular Chord ($s = 8, N = 600$)

8.3.2 Results

Figure 8.5 shows the average lookup success rate we measured for *DRR* and *DRR* combined with *AISNR-s* in a network of 200 nodes, for different settings of s and different attacker rates (f). Further, the figure displays the analytical bound for different values of s (compare equation 4.27; the analytical bound for $s = 8$ can hardly be seen in the figure because the value is essentially 100% for $f = 0.1, \ldots, 0.5$). It can be observed that our algorithms come close to analytical bounds in our emulation prototype. It is noteworthy to point out that the average lookup success rate we measured is slighty higher than the analytical bound for $s = 2$ and $f = 0.2$. Since the analytical bound concerns the average lookup success rate this slight deviation seems tolerable. The success rate is essentially 100% for $s = 8$ and attacker rates of $f \leq 0.4$.

Figures 8.6 and 8.7 show similar results for networks of size $N = 400$ and $N = 600$ where $s = 8$, comparing *Regular Chord* (enhanced with iterative backtracking), *DRR*, and *DRR* combined with *AISNR-s*. Again, it can be observed that a combination of *DRR* and *AISNR* can achieve 100% lookup success for $f \leq 0.4$. Further, note that regular Chord performs poorly in the presence of adversary nodes.

Reducing the hop count with *AISNR* The results in Figures 8.5, 8.6, and 8.7 show that *DRR* with $s = 8$ suffices to achieve a very high lookup success rate (essentially 100% or nearly 100%) for attacker rates up to $f = 0.4$. Application-intrinsic social network routing can be used in conbination with *DRR* in order to achieve a similar success rate with a lesser hop count. The results in Figure 8.8 demonstrate this effect of *AISNR-s* on the average hop count. The figure displays the hop count distribution for *DRR* compared to *DRR* combined with *AISNR-s* in a network with 400 nodes and $f = 0.3$. Notice that with *AISNR-s* and $\phi = 8$ lookups succeed with less hops compared to just using *DRR*. For *DRR* combined with *AISNR-s*($\phi = 8$), the 95-percentile for the hop

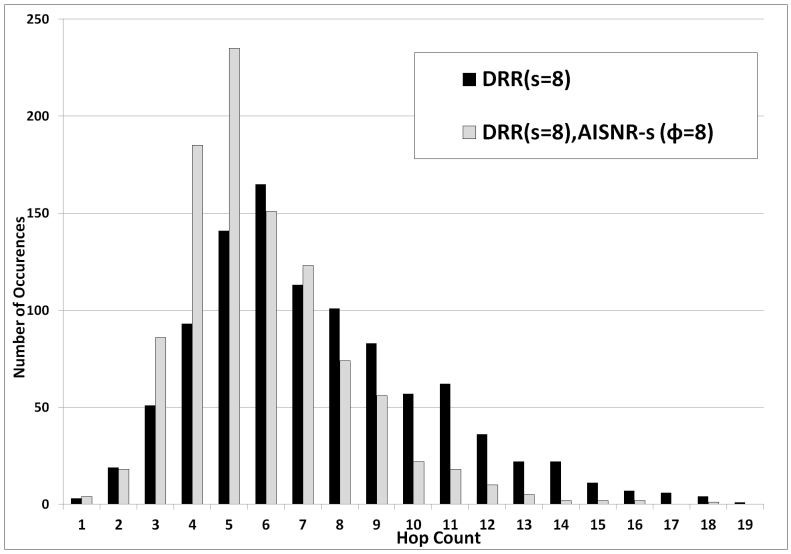

Figure 8.8: Hop Count Distribution for *DRR* and *DRR* Combined with *AISNR-s* in a Network with 400 Total Nodes of which 30% are Adversary Nodes ($f = 0.3$, $s = 8, N = 400$)

count is 10, whereas for *DRR* only, the 95-percentile for the hop count is 14 in the same scenario ($N = 400$, $f = 0.3$). Similar results have been obtained for different network sizes ($N = 200$, $N = 600$) and other attacker rates. The reader is referred to Appendix B.3 for those results.

To investigate the effect of ϕ, the number of initial (and constant in the case of *AISNR-s*) social links each node has, on the hop count distribution, we ran experiments with different values of ϕ. Figure 8.9 shows the results for $f = 0.3$ and $N = 200$ (results regarding the average hop count for different attacker rates are contained in Appendix B.3). The higher ϕ, the less hops are needed for lookups to succeed. This is an intuitive result: The more social links each node has, the higher is the probability that it can make good progress towards a key in each routing hop. Notice, however, that the gain of $\phi = 12$ compared to $\phi = 8$ is not significant. From these results we can infer that for the network sizes we emulated, $\phi = 8$ a sufficient for hop count reduction.

The effect of dynamic *AISNR* over time So far, we have shown that *static AISNR*, i.e. a fixed number of social links used by each node, can reduce the hop count for lookups. To investigate *dynamic AISNR* (*AISNR-d*), we ran experiments where nodes add links to their routing table $T_{soc}()$ during the experiment (as described in Section 8.2). We set the initial number of social links to 0 ($\phi = 0$). This setting emulates the complete evolution of an application-intrinsic social network over time, where initially each node has no social links. We executed 500 lookups during each run, and studied the hop count distribution for different intervals of lookups in the run. Figure 8.10 visualises the effect of *AISNR-d* on the hop count distribution over time in a network of $N = 200$ nodes and $f = 0.4$ (more results for *AISNR-d* for different attacker rates can be found in Appendix B.3). It can be observed that during the first 100 lookups, lookups tend to need more hops to

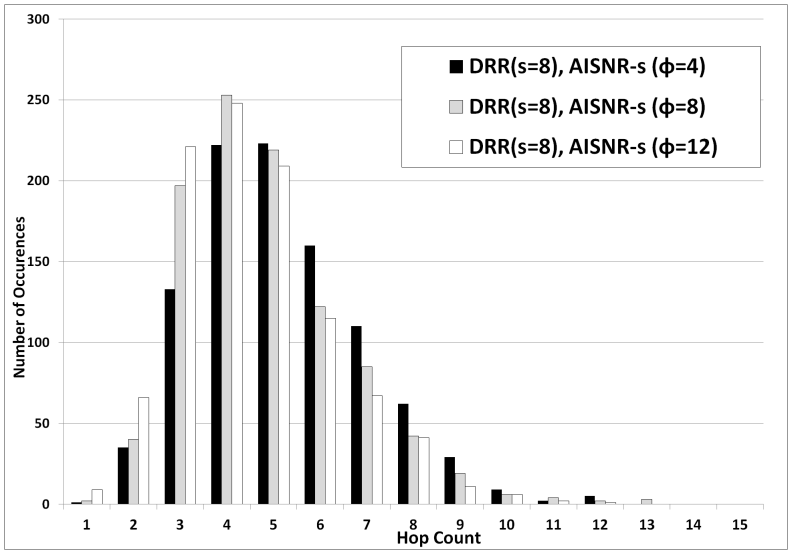

Figure 8.9: Hop Count Distribution for *DRR* Combined with *AISNR-s* for Different Values of ϕ in a Network with 200 Total Nodes of which 30% are Adversary Nodes ($f = 0.3$, $s = 8, N = 200$)

succeed than in the last 100 lookups of the experiment. As nodes add more and more social links to their routing table $T_{soc}()$, the application-intrinsic social network builds itself *over time*, thereby more and more reducing the hop count.

8.4 Discussion and Summary

We have presented a P2PSIP prototype system which can emulate adversary nodes. This system enabled us to study a real implementation of a P2PSIP network during Denial-of-Service attacks on the DHT-routing layer. We analysed the effect of several secure routing techniques on the lookup success rate as well as on the estimated hop count (which in the end amounts to call-setup time) under different settings.

Our experiments demonstrate that without the use of countermeasures (i.e. when using regular Chord routing), attackers can significantly degrade the service of locating a callee. More importantly, our results verify the effectiveness of our mechanisms for securing P2PSIP networks. In summary, our algorithms enable a success rate of largely 100% for $s = 8$ in networks with up to $N = 600$ nodes and up to 40% adversary nodes. Moreover, our experiments demonstrate that an application-intrinsic social network can be exploited within DHT routing to reduce the average hop count. This enables the timely establishment of real-time communication settings. Moreover, such an application-intrinsic social network can evolute as part of regular P2PSIP operations, thereby making the overall network more and more secure over time.

The results we presented in this chapter have been obtained from the emulation of rather moderately-sized P2PSIP networks, consisting of less than 1000 nodes. Note that

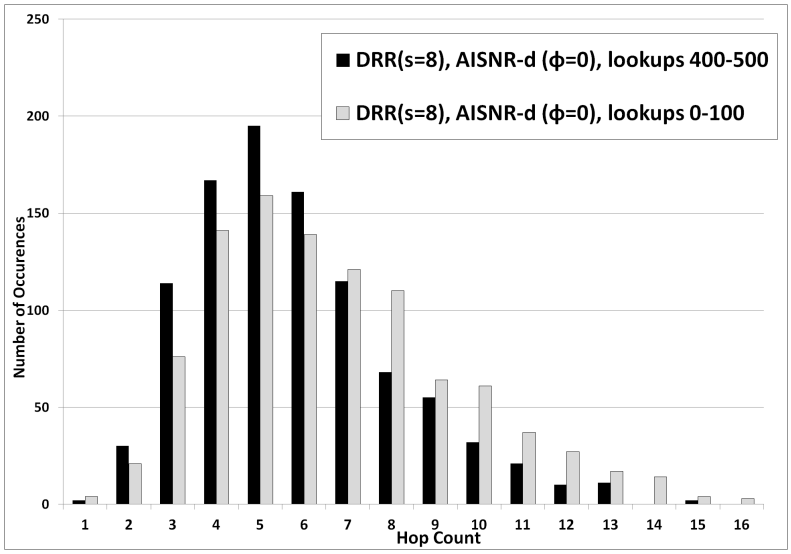

Figure 8.10: The Effect of *AISNR-d* on the Hop Count Distribution Over Time ($f = 0.4$, $s = 8, N = 200$)

some of the P2PSIP use-cases envision application scenarios with network sizes in this range (e.g. "Emergency First Responder Networks" or "Ad-Hoc and Ephemeral Groups" [77]). For such scenarios our results are directly applicable and valid. In addition, our system could easily be run on multiple hosts with bigger memory, emulating networks in the range of several 1000 nodes. Our architecture could hence be used to emulate larger networks.

Another potential drawback of our experiments is the fact that our results originate from a local setup. This prevents to determine realistic lookup delays for Internet scale scenarios. To mitigate this drawback, we used the hop count which can be used to estimate Internet-scale round trip times and thus realistic call setup delays. Finally, a condition of our experiments is that we emulated a static network without nodes joining or leaving the network (so-called *churn*) during experimental runs.

While admittedly these limitations but some constraints on the transferability of our results to large Internet-scale P2PSIP networks, it is important to realise that our results provide an upper bound on lookup success for any larger network: Analytically it can be shown that as network size increases the average lookup success rate may decrease in networks under attack due to longer path lookup lengths (see equation 4.13 and [246]), possibly even worsened by churn. Thus, any algorithm that provides unsatisfactory outcome in small networks in our system must be regarded as not useful in larger networks.

In summary, our system provides the ability to study P2PSIP-security in a real environment. It can easily be extended to test new secure routing algorithms. Thus, our prototype architecture enables to analyse different attacker models and to study future novel security techniques in an actual P2PSIP network *with relatively small effort*. By *emulating random traffic*, our system offers short-term estimations regarding the usefulness of algorithms in an actual network without the need to deploy nodes Internet-wide, e.g. via *planetlab* [14].

Chapter 9

Conclusion

Contents

This chapter concludes this thesis. We summarise our work and highlight our contributions. Further, we assess our contributions by contrasting them with our initial goals. Finally, we discuss open issues and provide an outlook on future work.

9.1 Discussion and Assessment of Contributions

9.1.1 Main Contributions in Summary

Our security analysis for Decentralised Service Location—exemplified with P2P-based real-time communication session establishment (*P2PSIP*)—identified several security requirements. Our study of prior art revealed that the following security challenges have not been sufficiently addressed by existing work: On the DHT Routing Layer, the *availability* of the lookup service needs to be maintained. Further, the *integrity* of location-bindings stored as data items in the DHT needs to be protected. On the application layer—considering specifically session establishment for real-time communication—*assessment of unknown identities* is necessary for decentralised protection against unsolicited communication (Spam). For each of these challenges, we have developed decentralised mechanisms to address them. Further, *Lawful Interception* is a legal requirement in many

countries for multimedia communication session establishment. We provided a detailed technical investigation for this challenge.

Availability of the DHT lookup service We have proposed algorithms to increase the success rate of DHT lookups (see Chapter 4 for details). Our algorithms extend Chord [266] in order to maintain availability of the lookup service in the presence of attackers in the DHT. We enhanced Chord with *complete-knowledge iterative routing*. In addition, we designed essentially three techniques as enhancement to regular Chord routing. *Independent/Alternate Multipath Routing (CMR)* enables to cope with failed lookup paths by either restarting an independent path or by using backtracking. *Direct Replica Routing (DRR)* ensures that data items can be retrieved even if the root node for a given key or its predecessor are adversary nodes. *Density Checks* can detect attacks on node-ID suppression in routing tables. Our algorithms are truly decentralised: Each node makes its own routing decision autonomously, without relying on any central entity.

We have analysed the proposed algorithms analytically and evaluated them through simulation. Our experiments show that our combined algorithms can come very close to theoretical bounds on the DHT lookup success rate. For instance, our solution can in principle achieve 99.9% success rate in the case of 40% adversary nodes in the network. To account for the fact that lookups are expected to succeed within a certain amount of time, we conducted additional experiments with varying hop thresholds. Our results show that even with a limited hop threshold our algorithms are still effective and can provide availability of the DHT lookup service.

Our contribution is the design and evaluation of algorithms that constitute effective and efficient measures to enable DHT lookup availability for Chord. Moreover, we have discovered several methodology flaws in the evaluation of prior work and hence shown the insufficiency of such solutions. In contrast, we have shown that our algorithms are effective under realistic and appropriate conditions. Therefore, we consider our work an important and considerable contribution.

Integrity of location-bindings We have designed a mechanism using *self-certifying SIP-URIs* to protect the integrity of location-bindings stored in a DHT (see Chapter 5 for details). With this approach, any node in the network can verify the integrity of location-bindings it receives (or forwards), without relying on a central authority. We have presented detailed schemes on how to generate a self-certifying SIP-URI and how to apply this approach to P2PSIP. Further, we have discussed potential attacks on our scheme and how to mitigate them.

We have assessed this approach qualitatively by discussing its benefits and drawbacks. As advantages, our approach does not rely on a central certification authority, can be used by any node in the network to check the integrity of arbitrary messages, is independent of the DHT used or any particular routing strategy, and is compatible with regular SIP-URI's as specified in RFC 3261 [222]. However, our approach also entails some disadvantages. First, self-certifying SIP-URIs are not human-readable. Second, key-

revocation is cumbersome because essentially revoking a cryptographic key implies the generation of a new SIP-URI. Further, the cryptographic primitives we introduce result in minor computational overhead. Finally, our solution does not solve the problem of associating a physical entity (i.e. a user) with a SIP-URI.

Compared to prior art, our contribution is the design of a self-certifying scheme for SIP-URIs in conjunction with P2P-based session establishment. We believe the advantages of our approach and its truly decentralised nature outweigh its drawbacks. In summary, our solution is cryptographically secure, efficient, and decentralised.

Assessment of unknown identities Existing mechanisms to assess incoming VoIP calls based in the identity of the caller are based on centralised entities which assert identities. A consequence of Decentralised Service Location is that such solutions are not applicable. Hence, any decentralised mechanism to detect unsolicited communication (e.g. *Spam-over-IP-Telephony*, *SPIT*) must reside on terminals. We have designed a decentralised alternative based on a Web-of-Trust model (see Chapter 6 for details). Our approach exploits the social relationships between end-users to detect solicited incoming messages. Users can cryptographically sign the identity of another user, based on the perceived past behaviour regarding such an identity. Our solution enables to derive and cryptographically verify in real-time a certificate chain from a callee to a given caller. The length of the derived certificate chain helps the callee to judge the trustworthiness of an incoming message.

We have evaluated this approach through prototype implementation using a modified SIP proxy and the existing PGP Web-of-Trust. Our implementation enables to evaluate our approach for client-server SIP as well as for P2PSIP. We have analysed various scenarios and degrees of decentralisation of applying our approach to P2PSIP. Our results show that our approach is feasible for real-time communication setup establishment. Depending on the concrete setting, the additional call setup delay is in the order of several $100ms$. We regard this an acceptable performance penalty for the gained ability to assess incoming messages.

Our approach relies on centralised entities (Web-of-Trust key servers) for storing—verified or unverified—certificate chain graphs. However, these servers only need to be contacted infrequently by P2PSIP nodes. Further, the identity assertion process itself is completely decentralised. For verification of certificate chains or incoming calls the callee does not rely on nor need to trust a centralised entity. Our approach contributes the integration of a Web-of-Trust into real-time communications such as VoIP. In addition, our scheme can be used as a decentralised SPIT prevention mechanism suitable for P2PSIP. Thus, our contribution is the design and implementation of adapting a Web-of-Trust model for securing decentralised real-time communication systems in a *decentralised* fashion, i.e. with as little involvement of centralised entities as possible during session establishment.

Lawful Interception Our technical analysis of Lawful Interception in P2PSIP networks (see Chapter 7 for details) revealed that several key characteristics of P2P networks make targeted interception of communications very challenging: First, P2P systems lack a central entity for interception. Second, routing paths are unique for every call. This fact is even worsened by the frequently changing membership of nodes, rendering the routing structure and key responsibility (on which communication setup is based) highly dynamic. Finally, P2P nodes must be regarded as arbitrary, non-trustworthy entities. They can thus not be relied upon following any protocol or mechanism correctly.

Our main contribution is the precise and detailed analysis of technical challenges for Lawful Interception in DHT-based service location. In addition, we have outlined and discussed technical difficulties for potential solutions. Our study contributes a detailed technical analysis of applying Lawful Interception in P2PSIP networks.

9.1.2 Revisiting Initial Objectives and Research Questions

The objectives of this thesis have been threefold: First, to provide a security analysis for Decentralised Service Location; second, to contrast the derived security requirements with existing, decentralised solutions and their applicability; and third, to design and evaluate innovative, decentralised mechanisms that properly address the identified security challenges.

We have analysed the security requirements and research challenges for Decentralised Service Location (exemplified with multimedia communication session establishment) in detail (see Chapter 3). In addition, we have examined existing work and derived the remaining research gap as challenges for designing innovative solutions (see also Chapter 3). We thus consider our first two objectives as achieved.

Our main contribution is the design of individual, decentralised solutions to secure Decentralised Service Location. The solutions we have developed each tackle one of the identified security challenges. The solutions are decentralised in nature. Moreover, we have evaluated them extensively and shown their effectiveness. We have thus achieved our initial aim of devising appropriate decentralised solutions and securing Decentralised Service Location.

Furthermore, our individual solutions can be used in conjunction for a holistic protection of Decentralised Service Location. Our proof-of-concept prototype implementation demonstrates this (see Chapter 8 for details). It enables to concatenate some of our algorithms for lookup availability with self-certifying SIP-URIs to protect the integrity of location bindings. These techniques secure the availability and the integrity of a DHTs lookup service. For instance, it enables a P2PSIP caller to securely *retrieve* the location of a callee. We have demonstrated the practical feasibility of these solutions combined in a running system. In addition, our Web-of-Trust approach for decentralised identity assessment enables to protect against unsolicited *incoming* communication requests. It can therefore be used independently of—or in conjunction with—our other solutions.

In summary, all of our initial goals have been satisfactorily attained. During the course of our work, we have also been able to answer our initial research questions (compare with 1.2). In principle, all our solutions are decentralised in the sense that their *security* does not depend on a central authority. Therefore we can conclude that it is indeed possible to secure Decentralised Service Location with decentralised mechanisms. Moreover, our algorithms for maintaining DHT lookup availability are truly decentralised: Each node makes its routing decision solely based on information retrieved from other P2P nodes and no entity has a special role in the network.

However, some of our approaches involve—in a very limited way, but nonetheless—central components. Self-certifying SIP-URIs enable fully decentralised authentication of location-bindings, i.e. without relying on any central authority. But in order to be useful for establishing real-time communication among users, a mapping between a URI and a physical identity must be available. As an example, we have suggested using an https based website for such a mapping, i.e. a server. Strictly speaking, the functionality of providing such a physical-identity-to-URI mapping is outside the scope of our work: Service Location assumes that a name for a service has been acquired previously to finding the location for a given service. Nevertheless, we acknowledge the need for such a mapping. We did not develop a decentralised solution to this problem. Providing a fully decentralised mapping of physical identities to SIP-URIs has not been answered by our work.

Similarly, our Web-of-Trust based approach for decentralised identity assessment relies loosely on central entities. We assume the presence of key-servers to compute and provide certificate chain graphs. However, it is important to acknowledge that these entities need to be contacted only infrequently and are not needed for every session establishment. Also, these entities are not necessarily required to verify certificate chains. Most importantly, these key-servers do not act as central authorities (i.e. certificate authorities) in our system; the process of asserting identities is truly decentralised among P2P nodes.

In conclusion, we have been able to design fully decentralised solutions for preserving the availability of the DHT lookup service and for integrity protection of location-bindings. We acknowledge that our integrity protection approach probably requires a centralised service for securely mapping a physical user onto a SIP identity[1]. Our decentralised identity assessment approach requires central entities to compute and provide certificate chain graphs. We have thus not in all cases been able to design completely decentralised solutions. In those cases where this was not possible, we still regard the degree of decentralisation as rather high because the centralised entities we assume a) do not act as authorities, i.e. the security of our approaches does not depend on them, and b) are not needed for every session establishment but rather infrequently.

For our lookup availability algorithms we have studied (by means of simulation) the effect of various parametrisations on the attainable lookup success rate and lookup delay (measured in DHT hops). We have hence comprehensively answered the research question what application performance is possible with these algorithms. Our results show

[1]Since this functionality is outside the scope of Decentralised Service Location we did not investigate finding a decentralised solution further.

that for small to medium attacker rates our algorithms can provide high lookup success rates and acceptable lookup delay. Our proof-of-concept implementation confirms these insights. Further, our prototype experiments verified that our proposed approach using self-certifying SIP-URIs delays session establishment insignificantly.

For our proposed decentralised identity assertion approach, we have extensively analysed the trade-offs involved between performance and decentralisation. We analysed different scenarios quantitatively regarding their reliance on a trustworthy key server for pre-verification of identity assertion paths and the corresponding call setup delay at the callee and caller in a P2PSIP scenario (see 6.3.2 for details). Our conclusion has been that a higher degree of decentralisation implies less reliance on central entities but a *performance penalty* in the form of an additional, but still acceptable call setup delay.

9.2 Open Issues and Future Work

The outcome of our work are effective, decentralised security approaches for Decentralised Service Location. However, some open issues exist. Future work may possibly address some of these issues or the minor shortcomings of our solutions.

As highlighted previously, not all our proposed solutions are fully decentralised in the sense that they do not require any central components at all. In particular, it is an open research issue whether it is possible to compute a Web-of-Trust certificate chain graph deterministically in a completely decentralised fashion. Also, further research may investigate the question how to bind a physical entity onto a communication identity without using trusted, central entities.

Further, it is still an open research question whether it is possible to design secure DHT routing algorithms (i.e. ones that provide protection against attacks on availability of the lookup service) that scale *logarithmically*. We have shown that density checks enable our algorithms to scale with increasing network size. However, our algorithms do not scale logarithmically as the network size, and consequently the average path length, increases.

For our Web-of-Trust based approach to assert SIP identities, we have proposed to use the *length* of the smallest identity assertion path as a metric to judge incoming messages. However, we did not investigate what a proper threshold would be to consider an incoming message as trustworthy. In addition, more sophisticated metrics, such as the number of existing assertion paths to the caller's identity where the length is below a certain threshold, could prove more useful and more robust against certificate chain infiltration by attackers. Further, we did not study how to combine our approach with other, existing Spam-prevention mechanisms. Finally, due to the lack of an existing large-scale Web-of-Trust, we could not verify our approach for networks in the range of several 100.000 users. Further research is needed to investigate these open issues. However, it is likely that proper answering of these research questions would require real datasets of users using our approach, or at least *call data records (CDRs)* from existing VoIP traffic

which include clearly marked Spam messages. At this point in time, such datasets are hard to obtain, largely due to the fact that Spam-over-IP-Telephony is not a widespread phenomenon yet.

Within the scope of our P2PSIP prototype implementation, we have proposed the novel idea of using an *application-intrinsic* social network for DHT-layer security. However, we have only investigated this approach in principle and shown its effectiveness rudimentarily. Further research could study this approach in more depth. For instance, interesting research would be to explore the applicability of this approach in conjunction with realistic data about the application-intrinsic social relationships among VoIP users (i.e. using centroid and clustering models obtained from VoIP CDRs) and considering realistic call patterns and online time of users (also obtained from CDRs).

Finally, we have only touched the surface of finding a proper solution for Lawful Interception of P2PSIP traffic. We have studied potential approaches incipiently and not in detail. However, our analysis of the problem has clearly revealed that it is technically very challenging to solve and most existing mechanisms are not applicable. Existing Lawful Interception architectures have been designed with a service provider in mind, which is trustworthy and knows the locations of users. With P2PSIP, this service provider is essentially replaced with a dynamically changing overlay network. Thus, the mapping of identities to locations is stored at untrustworthy—and potentially at each point in time different—nodes. We therefore regard it an interesting open research problem to find a scalable, efficient, and real-time solution for Lawful Interception in P2PSIP networks.

Bibliography

[1] "Assembly Names," last visited on June 5th, 2012. [Online]. Available: http://msdn.microsoft.com/en-us/library/k8xx4k69(v=vs.71)

[2] "BBC iPlayer - iPlayer TV Home," last visited on June 4th, 2012. [Online]. Available: http://www.bbc.co.uk/iplayer/tv

[3] "BitTorrent - Weltweite Bereitstellung von Inhalten," last visited on June 4th, 2012. [Online]. Available: http://www.bittorrent.com/intl/de/

[4] "BitTorrent Live," last visited on June 4th, 2012. [Online]. Available: http://live.bittorrent.com/

[5] "Disk2vhd," last visited on June 4th, 2012. [Online]. Available: http://technet.microsoft.com/en-us/sysinternals/ee656415

[6] "Free Skype internet calls and cheap calls to phones online - Skype," last visited on June 4th, 2012. [Online]. Available: http://www.skype.com/intl/en/home

[7] "Frequently Asked Questions (FAQ) /// Internet Traffic Report," last visited on August 25th, 2012. [Online]. Available: http://www.internettrafficreport.com/faq.htm

[8] "GnuTLS - GNU Project - Free Software Foundation," last visited on June 4th, 2012. [Online]. Available: http://www.gnu.org/software/gnutls/gnutls.html

[9] "JAIN-SIP: JAVA API for SIP Signaling," last visited on June 4th, 2012. [Online]. Available: http://java.net/projects/jain-sip/

[10] "MIT PGP Key Server," last visited on June 4th, 2012. [Online]. Available: http://wwwkeys.pgp.net/

[11] "Oracle VM VirtualBox," last visited on June 4th, 2012. [Online]. Available: https://www.virtualbox.org/

[12] "Otto-Friedrich-Universität Bamberg: OpenChord," last visited on June 4th, 2012. [Online]. Available: http://www.uni-bamberg.de/en/pi/bereich/research/software-projects/openchord/

[13] "P2PSIP Status Pages - Peer-to-Peer Session Initiation Protocol (Active WG)," last visited on June 4th, 2012. [Online]. Available: http://tools.ietf.org/wg/p2psip/

[14] “PlanetLab - An open platform for developing, deploying, and accessing planetary-scale services,” last visited on June 4th, 2012. [Online]. Available: http://www.planet-lab.org/

[15] “PlanetSim: Object Oriented Simulation Framework for Overlay Networks,” last visited on June 4th, 2012. [Online]. Available: http://projects-deim.urv.cat/trac/planetsim/

[16] “PPLive - Download,” last visited on June 4th, 2012. [Online]. Available: http://pplive.en.softonic.com/

[17] “RC4 Page,” last visited on June 4th, 2012. [Online]. Available: http://www.wisdom.weizmann.ac.il/~itsik/RC4/rc4.html

[18] “SIPDHT,” last visited on June 4th, 2012. [Online]. Available: http://sipdht.sourceforge.net

[19] “sipsak homepage,” last visited on June 4th, 2012. [Online]. Available: http://sipsak.org/

[20] “Statistics - Most Cited Articles in Computer Science,” (generated from documents in the CiteSeerx database as of May 20, 2012), last visited on June 4th, 2012. [Online]. Available: http://citeseer.ist.psu.edu/stats/articles

[21] “Tor: Hidden Service Protocol,” last visited on June 4th, 2012. [Online]. Available: https://www.torproject.org/docs/hidden-services.html.en

[22] “Tor Project: Hidden Service Configuration Instructions,” last visited on June 4th, 2012. [Online]. Available: https://www.torproject.org/docs/tor-hidden-service.html.en

[23] “W3C Glossary - Search results,” last visited on June 4th, 2012. [Online]. Available: http://www.w3.org/2003/glossary/alpha/S/60

[24] “Welcome to SIPp,” last visited on June 4th, 2012. [Online]. Available: http://sipp.sourceforge.net/

[25] “Willkommen bei Facebook - anmelden, registrieren oder mehr erfahren,” last visited on June 4th, 2012. [Online]. Available: http://www.facebook.com/

[26] “XING - Das professionelle Netzwerk XING,” last visited on June 4th, 2012. [Online]. Available: http://www.xing.com/de

[27] “YaCy - Freie Suchmaschinensoftware und dezentrale Websuche,” last visited on June 4th, 2012. [Online]. Available: http://yacy.net/de/

[28] “Secure Hash Standard,” US National Institute of Standards and Technology (NIST), Federal Information Processing Standards Publication 180-1, Apr. 1995, last visited on June 4th, 2012. [Online]. Available: http://www.itl.nist.gov/fipspubs/fip180-1.htm

[29] "One-Way Transmission Time," International Telecommunication Union Telecommunication Standardization Sector (ITU-T), Recommendation G.114, 1996, last visited on June 4th, 2012. [Online]. Available: http://www.itu.int/rec/T-REC-G.114-200305-I

[30] "Network Grade of Service Parameters and Target Values for Circuit-Switched Services in the Evolving ISDN," International Telecommunication Union Telecommunication Standardization Sector (ITU-T), Recommendation E.721, May 1999, last visited on June 4th, 2012. [Online]. Available: http://www.itu.int/rec/T-REC-E.721-199905-I

[31] "IP Multimedia Subsystem (IMS) (Stage 2) - Release 5," 3rd Generation Partnership Project (3GPP), Technical Specification TS 23.228, 2002.

[32] "Lawfully Authorized Electronic Surveillance (LAES) for Voice over Packet Technologies in Wireline Telecommunications Networks," Alliance for Telecommunications Industry Solutions (ATIS), Tech. Rep. ATIS-1000678.200X (ANS T1.678), Version 2. Draft, (proposed), 2003.

[33] "Multiple Vulnerabilities in Implementations of the Session Initiation Protocol (SIP)," Carnegie Mellon University CERT, Tech. Rep. CERT Advisory CA-2003-06, 2003, last visited on June 4th, 2012. [Online]. Available: http://www.cert.org/advisories/CA-2003-06.html

[34] "PacketCable Electronic Surveillance Specification," Cable Television Laboratories, Tech. Rep. PKT-SP-ESP-104-040723, Jul. 2004.

[35] "Lawful Interception (LI): Interception Domain Architecture for IP networks," European Telecommunications Standards Institute (ETSI), Tech. Rep. ETSI TS 102 528, v1.1.1, Nov. 2006.

[36] "Telecommunications and Internet Converged Services and Protocols for Advanced Networking (TISPAN) - NGN Release 1," European Telecommunications Standards Institute (ETSI), Tech. Rep. ETSI TR 180 001, v1.1.1, 2006.

[37] "Factsheet - Root server attack on 6 February 2007," ICANN (Internet Corporation for Assigned Names and Numbers), Tech. Rep., 2007, last visited on June 4th, 2012. [Online]. Available: http://www.icann.org/en/news/announcements/announcement-08mar07-en.htm

[38] "German minister defends 'Trojan horse' spy tactic as needed to fight terror," 2007, last visited on June 4th, 2012. [Online]. Available: http://www.tmcnet.com/usubmit/2007/08/31/2903041.htm

[39] "Packet-Based Multimedia Communications Systems," International Telecommunication Union Telecommunication Standardization Sector (ITU-T), Recommendation H.323 (v.7), Dec. 2009, last visited on June 4th, 2012. [Online]. Available: http://www.itu.int/rec/T-REC-H.323-200912-I

[40] H. Abdelnur, R. State, and O. Festor, "Kiph: A Stateful SIP Fuzzer," in *Principles, Systems and Applications of IP Telecommunications (IPTComm)*. iptcomm.org, 2007, last visited on June 4th, 2012. [Online]. Available: http://iptcomm.org/iptcomm2007/index.html

[41] B. Aboba, D. Thaler, and L. Esibov, "Link-local Multicast Name Resolution (LLMNR)," IETF, RFC 4795, Jan. 2007.

[42] A. Adelsbach, A. Alkassar, K.-H. Garbe, M. Luzaic, M. Manulis, E. Scherer, J. Schwenk, and E. Siemens, "VoIPSEC - Studie zur Sicherheit von Voice over Internet Protocol," Bundesamt für Sicherheit in der Informationstechnik, Tech. Rep., 2005.

[43] S. Androutsellis-Theotokis and D. Spinellis, "A Survey of Peer-to-Peer Content Distribution Technologies," *ACM Computing Surveys*, vol. 36, no. 4, pp. 335–371, Dec. 2004.

[44] C. Aras, J. Kurose, D. Reeves, and H. Schulzrinne, "Real-Time Communication in Packet-Switched Networks," *Proceedings of the IEEE*, vol. 82, no. 1, pp. 122–139, Jan. 1994.

[45] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose, "DNS Security Introduction and Requirements," IETF, RFC 4033, Mar. 2005.

[46] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose, "Protocol Modifications for the DNS Security Extensions," IETF, RFC 4035, Mar. 2005.

[47] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose, "Resource Records for the DNS Security Extensions," IETF, RFC 4034, Mar. 2005.

[48] M. S. Artigas, P. G. López, and J. P. Ahulló, "Cyclone: A Novel Design Schema for Hierarchical DHTs," in *IEEE International Conference on Peer-to-Peer Computing*. IEEE Computer Society, Washington DC, USA, Aug. 2005, pp. 49–56.

[49] M. S. Artigas, P. G. López, and A. F. Gómez-Skarmeta, "A Novel Methodology for Constructing Secure Multipath Overlays," *IEEE Internet Computing*, vol. 9, no. 6, pp. 50–57, Nov. 2005.

[50] T. Aura, "Cryptographically Generated Addresses (CGA)," in *Information Security Conference (ISC)*, ser. LNCS, no. 2851. Springer-Verlag, Heidelberg, Germany, Oct. 2003, pp. 29–43.

[51] T. Aura, "Cryptographically Generated Addresses (CGA)," IETF, RFC 3972, Mar. 2005.

[52] T. Aura and M. Roe, "Strengthening Short Hash Values," Tech. Rep., last visited on June 4th, 2012. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.145.7681

[53] B. Awerbuch and C. Scheideler, "Towards Scalable and Robust Overlay Networks," in *International Workshop on Peer-to-Peer Systems (IPTPS)*. iptps.org, 2007, last visited on June 4th, 2012. [Online]. Available: http://www.iptps.org/papers.html#2007

[54] B. Awerbuch, "Robust Distributed Name Service," in *International Workshop on Peer-to-Peer Systems (IPTPS)*. iptps.org, 2004, last visited on June 4th, 2012. [Online]. Available: http://www.iptps.org/papers.html#2004

[55] B. Awerbuch and C. Scheideler, "Towards a Scalable and Robust DHT," in *Eighteenth annual ACM symposium on Parallelism in algorithms and architectures (SPAA)*. ACM, New York, USA, 2006.

[56] M. Bagnulo and J. Arkko, "Support for Multiple Hash Algorithms in Cryptographically Generated Addresses (CGAs)," IETF, RFC 4982, Jul. 2007.

[57] F. Baker, B. Foster, and C. Sharp, "Cisco Architecture for Lawful Intercept in IP Networks," IETF, RFC 3924, Oct. 2004.

[58] V. A. Balasubramaniyan, M. Ahamad, and H. Park, "CallRank: Combating SPIT Using Call Duration, Social Networks and Global Reputation," in *Conference on Email and AntiSpam (CEAS)*, 2007, last visited on June 4th, 2012. [Online]. Available: http://ceas.cc/2007/

[59] S. Baset, "OpenVoIP: An Open Peer-to-Peer VoIP and IM System," last visited on June 4th, 2012. [Online]. Available: http://www.cs.columbia.edu/~salman/peer/index.html

[60] S. Baset, H. Schulzrinne, and M. Matuszewski, "Peer-to-Peer Protocol (P2PP)," IETF, Internet Draft draft-baset-p2psip-p2pp-01, Nov. 2007, expired, last visited on June 4th, 2012. [Online]. Available: http://tools.ietf.org/html/draft-baset-p2psip-p2pp-01

[61] S. Baset, G. Gupta, and H. Schulzrinne, "OpenVoIP: An Open Peer-to-Peer VoIP and IM System," in *Proceedings of the ACM SIGCOMM 2008 Conference*. ACM, New York, NY, USA, Aug. 2008, demonstration.

[62] S. Baset and H. Schulzrinne, "An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol," in *25th IEEE International Conference on Computer Communications (INFOCOM)*. IEEE Press, Piscataway, NJ, USA, 2006.

[63] I. Baumgart, "P2PNS: A Secure Distributed Name Service for P2PSIP," in *Proceedings of the Sixth Annual IEEE International Conference on Pervasive Computing and Communications (PerCom)*. IEEE Computer Society, Washington DC, USA, Mar. 2008, pp. 480–485.

[64] I. Baumgart, B. Heep, and S. Krause, "A P2PSIP Demonstrator Powered by OverSim," in *IEEE International Conference on Peer-to-Peer Computing*. IEEE Computer Society, Washington DC, USA, 2007, poster.

[65] I. Baumgart and S. Mies, "S/Kademlia: A Practicable Approach Towards Secure Key-Based Routing," in *Proceedings of the 13th International Conference on Parallel and Distributed Systems - Volume 02*. IEEE Computer Society, Washington DC, USA, 2007.

[66] S. M. Bellovin, M. Blaze, E. Brickell, C. Brooks, V. Cerf, W. Diffie, S. Landau, J. Peterson, and J. Treichler, "Security Implications of Applying the Communications Assistance to Law Enforcement Act to Voice over IP," Tech. Rep., 2006, last visited on June 4th, 2012. [Online]. Available: https://www.cs.columbia.edu/~smb/papers/CALEAVOIPreport.pdf

[67] T. Berners-Lee, R. Fielding, and L. Masinter, "Uniform Resource Identifiers (URI): Generic Syntax," IETF, RFC 2396, Aug. 1998.

[68] A. v. Bidder and N. Weiler, "Key Exchange (KX) - A Next Generation Protocol to Synchronise PGP Keyservers," in *Proceedings of the Twelfth International Workshop on Enabling Technologies (WETICE '03)*. IEEE Computer Society, Washington DC, USA, 2003.

[69] P. Biondi and F. Desclaux, "Silver Needle in the Skype," BlackHat Europe, Tech. Rep., 2006, last visited on June 4th, 2012. [Online]. Available: http://www.blackhat.com/presentations/bh-europe-06/bh-eu-06-biondi/bh-eu-06-biondi-up.pdf

[70] D. Bonfiglio, M. Mellia, M. Meo, D. Rossi, and P. Tofanelli, "Revealing Skype Traffic: When Randomness Plays with You," in *Proceedings of the ACM SIGCOMM 2007 Conference*. ACM, New York, NY, USA, 2007.

[71] D. Bonfiglio, M. Mellia, M. Meo, N. Ritacca, and D. Rossi, "Tracking Down Skype Traffic," in *27th IEEE International Conference on Computer Communications (INFOCOM)*. IEEE Press, Piscataway, NJ, USA, 2008.

[72] N. Borisov, "Computational Puzzles as Sybil Defenses," in *IEEE International Conference on Peer-to-Peer Computing*. IEEE Computer Society, Washington DC, USA, 2006.

[73] J. W. Bos, O. Ozen, and J.-P. Hubaux, "Analysis and Optimization of Cryptographically Generated Addresses," in *Information Security Conference*, ser. LNCS, no. 5735. Springer-Verlag, Heidelberg, Germany, 2009, pp. 17–32.

[74] U. Brandes, "On Variants of Shortest-Path Betweenness Centrality and their Generic Computation," *Elsevier Social Networks*, vol. 30, no. 2, pp. 136–145, May 2008.

[75] D. Bryan, B. Lowekamp, and C. Jennings, "A P2P Approach to SIP Registration and Resource Location," IETF, Internet Draft draft-bryan-sipping-p2p-03, Oct. 2006, expired, last visited on June 4th, 2012. [Online]. Available: http://tools.ietf.org/html/draft-bryan-sipping-p2p-03

[76] D. Bryan, B. Lowekamp, and M. Zangrilli, "The Design of a Versatile, Secure P2PSIP Communications Architecture for the Public Internet," in *IEEE international Parallel and Distributed Processing Symposium (IPDPS)*. IEEE Computer Society, Washington DC, USA, 2008.

[77] D. Bryan, E. Shim, and B. Lowekamp, "Use Cases for Peer-to-Peer Session Initiation Protocol (P2P SIP)," IETF, Internet Draft draft-bryan-p2psip-usecases-00, 2007, expired, last visited on June 4th, 2012. [Online]. Available: http://tools.ietf.org/html/draft-bryan-p2psip-usecases-00

[78] D. Bryan, B. Lowekamp, and C. Jennings, "SOSIMPLE: A Serverless, Standards-based, P2P SIP Communication System," in *Proceedings of the First International Workshop on Advanced Architectures and Algorithms for Internet Delivery and Applications*. IEEE Computer Society, Washington DC, USA, 2005, pp. 42–49.

[79] D. Bryan and B. Rosen, "Peer-to-Peer Session Initiation Protocol (p2psip) Charter," last visited on June 4th, 2012. [Online]. Available: http://tools.ietf.org/wg/p2psip/charters

[80] S. Burresi, C. Canali, M. E. Renda, and P. Santi, "MeshChord: A Location-Aware, Cross-Layer Specialization of Chord for Wireless Mesh Networks," in *Proceedings of the 6th Annual IEEE International Conference on Pervasive Computing and Communications (Percom)*. IEEE Computer Society, Washington DC, USA, Mar. 2008, pp. 206–212.

[81] J. Callas, L. Donnerhacke, H. Finney, D. Shaw, and R. Thayer, "OpenPGP Message Format," IETF, RFC 4880, Nov. 2007.

[82] G. Camarillo and M.-A. Garca-Martn, *The 3G IP Multimedia Subsystem (IMS): Merging the Internet and the Cellular Worlds*, 1st ed. John Wiley and Sons Ltd., Chichester, England, 2004.

[83] S. Capkun, L. Buttyan, and J.-P. Hubaux, "Self-Organized Public-Key Management for Mobile Ad Hoc Networks," in *IEEE Transactions on Mobile Computing*, vol. 2, no. 1, Jan. 2003, pp. 52–64.

[84] M. Castro, P. Druschel, A. Ganesh, A. Rowstron, and D. Wallach, "Secure Routing for Structured Peer-to-Peer Overlay Networks," in *Proceedings of the 5th Symposium on Operating systems design and implementation (OSDI)*. ACM, New York, NY, USA, Dec. 2002.

[85] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, "Splitstream: High-Bandwidth Multicast in Cooperative Environments," in *Proceedings of the nineteenth ACM symposium on Operating systems principles (SOSP)*. ACM, New York, NY, USA, Oct. 2003.

[86] M. Castro, P. Druschel, Y. C. Hu, and A. Rowstron, "Proximity Neighbor Selection in Tree-Based Structured Peer-to-Peer Overlays," Microsoft Research, Tech. Rep. MSR-TR-2003-52, Sep. 2003.

[87] J. Cederlof, "The Web of Trust .wot file format," Nov. 2004, last visited on June 4th, 2012. [Online]. Available: http://www.lysator.liu.se/~jc/wotsap/wotfileformat.txt

[88] J. Cederlof, "Wotsap," Dec. 2004, last visited on June 4th, 2012. [Online]. Available: http://www.lysator.liu.se/~jc/wotsap/

[89] K.-T. Chen, C.-Y. Huang, P. Huang, and C.-L. Lei, "Quantifying Skype User Satisfaction," in *Proceedings of the ACM SIGCOMM 2006 Conference.* ACM, New York, NY, USA, 2006.

[90] C.-M. Cheng, S.-L. Tsao, and J.-C. Chou, "Unstructured Peer-to-Peer Session Initiation Protocol for Mobile Environment," in *18th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC).* IEEE Press, Piscataway, NJ, USA, 2007.

[91] S. Cheshire and M. Krochmal, "DNS-Based Service Discovery," IETF, Internet Draft draft-cheshire-dnsext-dns-sd-11, Dec. 2011, expired, last visited on August 27th, 2012. [Online]. Available: http://tools.ietf.org/html/draft-cheshire-dnsext-dns-sd-11

[92] S. Cheshire and M. Krochmal, "Multicast DNS," IETF, Internet Draft draft-cheshire-dnsext-multicastdns-15, Dec. 2011, expired, last visited on August 27th, 2012. [Online]. Available: http://tools.ietf.org/html/draft-cheshire-dnsext-multicastdns-15

[93] D. Chopra, H. Schulzrinne, E. Marocco, and E. Ivov, "Peer-to-Peer Overlays for Real-Time Communication: Security Issues and Solutions," *IEEE Communications Surveys & Tutorials*, vol. 11, no. 1, pp. 4–12, 2009.

[94] L. D. Cicco, S. Mascolo, and V. Palmisano, "Skype Video Responsiveness to Bandwidth Variations," in *Proceedings of the 18th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV).* ACM, New York, NY, USA, 2008.

[95] T. Close, "What Does the 'y' Refer to?" 2003, last visited on June 4th, 2012. [Online]. Available: http://www.waterken.com/dev/YURL/Definition/

[96] T. Close, "Authentication of TLS Upgrade Within HTTP/1.1," 2004, last visited on June 4th, 2012. [Online]. Available: http://www.waterken.com/dev/YURL/httpsy/

[97] T. Close, "Trust Management for Humans," 2004, last visited on June 4th, 2012. [Online]. Available: http://www.waterken.com/dev/YURL/Name/#PNML_f

[98] T. Condie, V. Kacholia, S. Sankararaman, J. M. Hellerstein, and P. Maniatis, "Induced Churn as Shelter from Routing-Table Poisoning," in *Annual Network and Distributed System Security Symposium (NDSS)*, 2006, last visited on June 4th, 2012. [Online]. Available: http://www.isoc.org/isoc/conferences/ndss/06/proceedings/

[99] R. Cox, A. Muthitacharoen, and R. T. Morris, "Serving DNS Using a Peer-to-Peer Lookup Service," in *International Workshop on Peer-to-Peer Systems (IPTPS)*. iptps.org, 2002, last visited on June 4th, 2012. [Online]. Available: http://www.iptps.org/papers.html#2002

[100] D. Crocker and P. Overell, "Augmented BNF for Syntax Specifications: ABNF," IETF, RFC 2234, Nov. 1997.

[101] R. Cuevas, A. Cuevas, M. Uruena, and A. Banchs, "Applying Low Discrepancy Sequences for Node-ID Assignment in P2PSIP," *IEEE Communications Letters*, vol. 15, no. 2, pp. 256 – 258, Feb. 2011.

[102] R. Cuevas, M. Uruena, and A. Banchs, "Routing Fairness in Chord: Analysis and Enhancement," in *28th IEEE International Conference on Computer Communications (INFOCOM)*. IEEE Press, Piscataway, NJ, USA, 2009, pp. 1449–1457.

[103] F. Dabek, M. Kaashoek, D. Karger, R. Morris, and I. Stoica, "Wide-Area Cooperative Storage with CFS," in *Proceedings of the seventeenth ACM symposium on Operating systems principles (SOSP)*. ACM, New York, NY, USA, Oct. 2001.

[104] F. Dabek, R. Cox, F. Kaashoek, and R. Morris, "Vivaldi: A Decentralized Network Coordinate System," in *Proceedings of the ACM SIGCOMM 2004 Conference*. ACM, New York, NY, USA, 2004, pp. 15–26.

[105] E. Damiani, S. D. C. di Vimercati, S. Paraboschi, P. Samarati, and F. Violante, "A Reputation-Based Approach for Choosing Reliable Resources in Peer-to-Peer Networks," in *Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS)*. ACM. New York, NY, USA, 2002.

[106] G. Danezis, C. Lesniewski-laas, M. F. Kaashoek, and R. Anderson, "Sybil-Resistant DHT Routing," in *9th European Symposium On Research in Computer Security (ESORICS)*, ser. LNCS, no. 3193. Springer-Verlag, Heidelberg, Germany, 2004, pp. 305–318.

[107] T. Dierks and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2," IETF, RFC 5246, Aug. 2008.

[108] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: the second-generation onion router," in *13th USENIX Security Symposium*. USENIX Association, Berkeley, CA, USA, 2004.

[109] J. R. Douceur, "The Sybil Attack," in *Revised Papers from the First International Workshop on Peer-to-Peer Systems*, ser. LNCS, no. 2429. Springer-Verlag, Heidelberg, Germany, Mar. 2002.

[110] P. Druschel and A. Rowstron, "PAST: A large-scale, persistent peer-to-peer storage utility," in *Proceedings of the Eighth Workshop on Hot Topics in Operating Systems*. IEEE Computer Society, Washington DC, USA, May 2001.

[111] E. Eren and K.-O. Detken, *VoIP Security Konzepte und Lösungen für sichere VoIP-Kommunikation*, 1st ed. Hanser, Munich, Germany, 2007.

[112] A. Fessi, N. Evans, H. Niedermayer, and R. Holz, "Pr2-P2PSIP: Privacy Preserving P2P Signaling for VoIP and IM," in *Principles, Systems and Applications of IP Telecommunications (IPTComm)*. iptcomm.org, 2010, last visited on June 4th, 2012. [Online]. Available: http://iptcomm.org/iptcomm2010/IPTComm2010-Proceedings.pdf

[113] A. Fessi, H. Niedermayer, H. Kinkelin, and G. Carle, "A cooperative SIP Infrastructure for Highly Reliable Telecommunication Services," in *Principles, Systems and Applications of IP Telecommunications (IPTComm)*. iptcomm.org, 2007, last visited on June 4th, 2012. [Online]. Available: http://iptcomm.org/iptcomm2007/index.html

[114] A. Fiat, J. Saia, and M. Young, "Making Chord Robust to Byzantine Attacks," in *13th Annual European Symposium on Algorithms (ESA)*, ser. LNCS, no. 3669. Springer-Verlag, Heidelberg, Germany, 2005.

[115] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, "Hypertext Transfer Protocol – HTTP/1.1," IETF, RFC 2616, Jun. 1999.

[116] P. Fischer and P. Hofer, *Lexikon der Informatik*, 15th ed. Springer-Verlag, Heidelberg, Germany, 2010.

[117] M. J. Freedman, E. Freudenthal, and D. Mazieres, "Democratizing Content Publication with Coral," in *Proceedings of the 1st USENIX Symposium on Networked Systems Design and Implementation (NSDI)*. USENIX Association, Berkeley, CA, USA, 2004.

[118] L. Ganesh and B. Y. Zhao, "Identity Theft Protection in Structured Overlays," in *Proceedings of the First international conference on Secure network protocols (NPSec)*. IEEE Computer Society, Washington DC, USA, 2005.

[119] M. Garcia-Martin, "Input 3rd-Generation Partnership Project (3GPP) Release 5 Requirements on the Session Initiation Protocol (SIP)," IETF, RFC 4083, May 2005.

[120] Y. Y. Goland, T. Cai, P. Leach, Y. Gu, and S. Albright, "Simple Service Discovery Protocol/1.0 - Operating without an Arbiter," IETF, Internet Draft draft-cai-ssdp-v1-03, Oct. 1999, expired, last visited on June 4th, 2012. [Online]. Available: http://tools.ietf.org/html/draft-cai-ssdp-v1-03

[121] D. Gollmann, *Computer Security*, 2nd ed. John Wiley and Sons Ltd., Chichester, England, 2005.

[122] B. Goode, "Voice over Internet Protocol (VoIP)," *Proceedings of the IEEE*, vol. 90, no. 9, pp. 1495–1517, Sep. 2002.

[123] S. Guha and N. Daswani, "An Experimental Study of the Skype Peer-to-Peer VoIP System," in *International Workshop on Peer-to-Peer Systems (IPTPS)*. iptps.org, 2006, last visited on June 4th, 2012. [Online]. Available: http://www.iptps.org/papers.html#2006

[124] E. Guttman, C. Perkins, J. Veizades, and M. Day, "Service Location Protocol, Version 2," IETF, RFC 2608, Jun. 1999.

[125] M. Handley, V. Jacobson, and C. Perkins, "SDP: Session Description Protocol," IETF, RFC 4566, Jul. 2006.

[126] M. Handley, H. Schulzrinne, E. Schooler, and J. Rosenberg, "SIP: Session Initiation Protocol," IETF, RFC 2543, Mar. 1999.

[127] T. Hardie, A. Newton, H. Schulzrinne, and H. Tschofenig, "LoST: A Location-to-Service Translation Protocol," IETF, RFC 5222, Aug. 2008.

[128] C. Harvesf and D. M. Blough, "The Effect of Replica Placement on Routing Robustness in Distributed Hash Tables," in *IEEE International Conference on Peer-to-Peer Computing*. IEEE Computer Society, Washington DC, USA, 2006.

[129] J. Heikkilä and A. Gurtov, "Filtering SPAM in P2PSIP Communities with Web of Trust," in *First International ICST Conference on Security and Privacy in Mobile Information and Communication Systems (MobiSec)*, ser. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering (LNICST), no. 17. Springer-Verlag, Heidelberg, Germany, 2009.

[130] K. Hildrum and J. Kubiatowicz, "Asymptotically Efficient Approaches to Fault-Tolerance in Peer-to-Peer Networks," in *17th International Symposium on Distributed Computing*, ser. LNCS, no. 2848. Springer-Verlag, Heidelberg, Germany, 2003.

[131] R. Hinden and S. Deering, "IP Version 6 Addressing Architecture," IETF, RFC 4291, Feb. 2006.

[132] T. Holz, M. Steiner, F. Dahl, E. Biersack, and F. Freiling, "Measurements and Mitigation of Peer-to-Peer-based Botnets: A Case Study on Storm Worm," in *1st USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET)*, 2008, last visited on August 27th, 2012. [Online]. Available: https://www.usenix.org/conference/leet-08/measurements-and-mitigation-peer-peer-based-botnets-case-study-storm-worm

[133] T.-Y. Huang, K.-T. Chen, and P. Huang, "Tuning Skype's Redundancy Control Algorithm for User Satisfaction," in *28th IEEE International Conference on Computer Communications (INFOCOM)*. IEEE Press, Piscataway, NJ, USA, 2009.

[134] IAB and IESG, "IETF Policy on Wiretapping," IETF, RFC 2804, May 2000.

[135] Iptel.org, "SIP Messages," last visited on August 27th, 2012. [Online]. Available: http://www.iptel.org/sip/intro/messages

[136] J. Arkko (Editor), J. Kempf, B. Zill, and P. Nikander, "SEcure Neighbor Discovery (SEND)," IETF, RFC 3971, Mar. 2005.

[137] C. Jackson and A. Barth, "Beware of Finer-Grained Origins," in *Proceedings of the Web 2.0 Security and Privacy Conference (W2SP)*, 2008, last visited on June 4th, 2012. [Online]. Available: http://w2spconf.com/2008/

[138] C. Jennings, "Computational Puzzles for SPAM Reduction in SIP," IETF, Internet Draft draft-jennings-sip-hashcash-06, Jul. 2007, expired, last visited on June 4th, 2012. [Online]. Available: http://tools.ietf.org/html/draft-jennings-sip-hashcash-06

[139] C. Jennings, B. Lowekamp, E. Rescorla, S. Baset, and H. Schulzrinne, "A SIP Usage for RELOAD," IETF, Internet Draft draft-ietf-p2psip-sip-07, Jan. 2012, expired, last visited on August 27th, 2012. [Online]. Available: http://tools.ietf.org/html/draft-ietf-p2psip-sip-07

[140] C. Jennings, B. Lowekamp, E. Rescorla, S. Baset, and H. Schulzrinne, "REsource LOcation And Discovery (RELOAD) Base Protocol," IETF, Internet Draft draft-ietf-p2psip-base-22, Jul. 2012, work in progress, last visited on August 27th, 2012. [Online]. Available: http://tools.ietf.org/html/draft-ietf-p2psip-base-22

[141] C. Jennings, J. Peterson, and M. Watson, "Private Extensions to the Session Initiation Protocol (SIP) for Asserted Identity within Trusted Networks," IETF, RFC 3325, Nov. 2002.

[142] J. Jiang, R. Pan, C. Liang, and W. Wang, "BiChord: An Improved Approach for Lookup Routing in Chord," in *Advances in Databases and Information Systems*, ser. LNCS, no. 3631. Springer-Verlag, Heidelberg, Germany, 2005, 2005.

[143] X. Jiang, H. Zheng, C. Macian, and V. Pascual, "Service Extensible P2P Peer Protocol," IETF, Internet Draft draft-jiang-p2psip-sep-01, Feb. 2008, expired, last visited on June 4th, 2012. [Online]. Available: http://tools.ietf.org/html/draft-jiang-p2psip-sep-01

[144] D. Johnson, C. Perkins, and J. Arkko, "Mobility Support in IPv6," IETF, RFC 3775, Jun. 2004.

[145] A. Johnston and H. Sinnreich, "SIP, P2P, and Internet Communication," IETF, Internet Draft draft-johnston-sipping-p2p-ipcom-02, Mar. 2006, expired, last visited on June 4th, 2012. [Online]. Available: http://tools.ietf.org/html/draft-johnston-sipping-p2p-ipcom-02

[146] A. Kapadia and N. Triandopoulos, "Halo: High-Assurance Locate for Distributed Hash Tables," in *Annual Network and Distributed System Security Symposium (NDSS)*, Feb. 2008, pp. 61–79, last visited on June 4th, 2012. [Online]. Available: http://www.isoc.org/isoc/conferences/ndss/08/proceedings.shtml

[147] D. Karger, E. Lehman, T. Leighton, R. Panigrahy, M. Levine, and D. Lewin, "Consistent Hashing and Random Trees: Distributed Caching Protocols for Relieving Hot Spots on the World Wide Web," in *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*. ACM, New York, NY, USA, 1997.

[148] C. Kaufman, "Internet Key Exchange (IKEv2) Protocol," IETF, RFC 4306, Dec. 2005.

[149] S. Kent and K. Seo, "Security Architecture for the Internet Protocol," IETF, RFC 4301, Dec. 2005.

[150] W. Kho, S. Baset, and H. Schulzrinne, "Skype Relay Calls: Measurements and Experiments," in *11th IEEE Global Internet Symposium*. IEEE Press, Piscataway, NJ, USA, 2008.

[151] J. Koskela, K. Karvonen, T. Kilinkaridis, and A. Gurtov, "Secure and Usable P2P VoIP for Mobile Devices," in *Proceedings of the 12th international conference on Human computer interaction with mobile devices and services*. ACM, New York, NY, USA, Sep. 2010.

[152] D. R. Kuhn, T. J. Walsh, and S. Fries, "Security Considerations for Voice over IP Systems - Recommendations of the National Institute of Standards and Technology," US National Institute of Standards and Technology (NIST), Special Publication 800-58, Jan. 2005, last visited on June 4th, 2012. [Online]. Available: http://csrc.nist.gov/publications/nistpubs/800-58/SP800-58-final.pdf

[153] J. F. Kurose and K. W. Ross, *Computer Networking: A Top-Down Approach Featuring the Internet*, 1st ed. Addison-Wesley Longman, Amsterdam, Netherlands, 2000.

[154] J. Li, J. Stribling, R. Morris, and M. F. Kaashoek, "Bandwidth-Efficient Management of DHT Routing Tables," in *Proceedings of the 2nd USENIX Symposium on Networked Systems Design and Implementation (NSDI)*. USENIX Association, Berkeley, CA, USA, 2005.

[155] D. Liben-Novell, H. Balakrishnan, and D. Karger, "Analysis of the Evolution of Peer-to-Peer Systems," in *Proceedings of the 21st Annual Symposium on Principles of Distributed Computing (PODC)*. ACM, New York, NY, USA, 2002.

[156] J. Lindqvist and M. Komu, "Cure for Spam over Internet Telephony," in *4th IEEE Consumer Communications and Networking Conference (CCNC)*. IEEE Press, Piscataway, NJ, USA, 2007.

[157] M. Linser and R. Barnes, "Emergency Context Resolution with Internet Technologies (ecrit) - Description of Working Group (Charter)," last visited on June 4th, 2012. [Online]. Available: http://datatracker.ietf.org/wg/ecrit/charter/

[158] E. K. Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim, "A Survey and Comparison of Peer-to-Peer Overlay Network Schemes," *IEEE Communications Surveys & Tutorials*, vol. 7, no. 2, pp. 72–93, 2005.

[159] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker, "Search and Replication in Unstructured Peer-to-Peer Networks," in *Proceedings of the 16th international conference on Supercomputing (ICS)*. ACM New York, NY, USA, 2002, pp. 84–95.

[160] C. M. MacKenzie, K. Laskey, F. McCabe, P. F. Brown, and R. Metz, "OASIS Reference Model for Service Oriented Architecture 1.0," OASIS (Organization for the Advancement of Structured Information Standards), Official OASIS Standard, Oct. 2006, last visited on June 4th, 2012. [Online]. Available: http://docs.oasis-open.org/soa-rm/v1.0/soa-rm.pdf

[161] J. Maenpaa and G. Camarillo, "Study on Maintenance Operations in a Chord-Based Peer-to-Peer Session Initiation Protocol Overlay Network," in *IEEE International Symposium on Parallel & Distributed Processing (IPDPS)*. IEEE Computer Society, Washington DC, USA, May 2009, pp. 1–9.

[162] J. Maenpaa and G. Camarillo, "Service Discovery Usage for REsource LOcation And Discovery (RELOAD)," IETF, Internet Draft draft-ietf-p2psip-service-discovery-05, Apr. 2012, work in progress, last visited on August 27th, 2012. [Online]. Available: http://tools.ietf.org/html/draft-ietf-p2psip-service-discovery-05

[163] J. Maenpaa, G. Camarillo, and J. Hautakorpi, "A Self-tuning Distributed Hash Table (DHT) for REsource LOcation And Discovery (RELOAD)," IETF, Internet Draft draft-ietf-p2psip-self-tuning-06, Jul. 2012, work in progress, last visited on August 27th, 2012. [Online]. Available: http://tools.ietf.org/html/draft-ietf-p2psip-self-tuning-06

[164] P. Mahlmann and C. Schindelhauer, *Peer-to-Peer-Netzwerke - Algorithmen und Methoden*, 1st ed. Spinger, Heidelberg, Germany, 2007.

[165] D. Malkhi, M. Naor, and D. Ratajczak, "Viceroy: A Scalable and Dynamic Emulation of the Butterfly," in *Proceedings of the 21st Annual Symposium on Principles of Distributed Computing (PODC)*. ACM, New York, NY, USA, 2002.

[166] E. Marocco and E. Ivov, "Extensible Peer Protocol (XPP)," IETF, Internet Draft draft-marocco-p2psip-xpp-01, Nov. 2007, expired, last visited on June 4th, 2012. [Online]. Available: http://tools.ietf.org/html/draft-marocco-p2psip-xpp-01

[167] S. Marti, P. Ganesan, and H. Garcia-Molina, "DHT Routing Using Social Links," in *International Workshop on Peer-to-Peer Systems (IPTPS)*. iptps.org, 2004, last visited on June 4th, 2012. [Online]. Available: http://www.iptps.org/papers.html#2004

[168] S. Marti and H. Garcia-Molina, "Taxonomy of Trust: Categorizing P2P Reputation Systems," *Elsevier Computer Networks*, vol. 50, no. 4, pp. 472–484, Mar. 2006.

[169] I. Martinez-Yelmo, A. Bikfalvi, R. Cuevas, C. Guerrero, and J. Garcia, "H-P2PSIP: Interconnection of P2PSIP Domain for Global Multimedia Services Based on a Hierarchical DHT Overlay Network," *Elsevier Computer Networks*, vol. 53, no. 4, pp. 556–568, Mar 2009.

[170] M. Matuszewski and E. Kokkonen, "Mobile P2PSIP - Peer-to-Peer SIP Communication in Mobile Communities," in *5th IEEE Consumer Communications and Networking Conference (CCNC)*. IEEE Press, Piscataway, NJ, USA, 2008, pp. 1159–1165.

[171] P. Maymounkov and D. Mazieres, "Kademlia: A Peer-to-peer Information System Based on the XOR Metric," in *Revised Papers from the First International Workshop on Peer-to-Peer Systems*, ser. LNCS, no. 2429. Springer-Verlag, Heidelberg, Germany, Mar. 2002.

[172] D. Mazieres, M. Kaminsky, M. F. Kaashoek, and E. Witchel, "Separating Key Management from File System Security," in *Proceedings of the seventeenth ACM symposium on Operating systems principles (SOSP)*. ACM, New York, NY, USA, 1999.

[173] J. McLachlan, A. Tran, N. Hopper, and Y. Kim, "Scalable Onion Routing with Torsk," in *Proceedings of the 16th ACM Conference on Computer and Communications Security (CCS)*. ACM. New York, NY, USA, 2009.

[174] A. Menezes, P. van Oorschot, and S. Vanstone, *Handbook of Applied Cryptography (Discrete Mathematics and Its Applications)*, 1st ed. CRC Press, Boca Raton, FL, USA, 1996.

[175] R. C. Merkle, "A Certified Digital Signature," in *Proceedings on Advances in Cryptology (CRYPTO)*. Springer-Verlag New York, NY, USA, 1989, pp. 218–238.

[176] M. Miller, "Lambda for Humans – The Pet Name Markup Language," last visited on June 4th, 2012. [Online]. Available: http://www.erights.org/elib/capability/pnml.html

[177] D. Mills, U. Delaware, J. Martin, J. Burbank, and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification," IETF, RFC 5905, 2010.

[178] A. Mislove, A. Post, C. Reis, P. Willmann, P. Druschel, D. S. Wallach, X. Bonnaire, P. Sens, J.-M. Busca, and L. Arantes-Bezerra, "POST: A Secure, Resilient, Cooperative Messaging System," in *Proceedings of the 9th conference on Hot Topics in Operating Systems*. USENIX Association, Berkeley, CA, USA, 2003.

[179] P. V. Mockapetris, "Domain names – concepts and facilities," IETF, RFC 1034, Nov. 1987.

[180] P. V. Mockapetris, "Domain names – implementation and specification," IETF, RFC 1035, Nov. 1987.

[181] G. Montenegro and C. Castelluccia, "Statistically Unique and Cryptographically Verifiable (SUCV) Identifiers and Addresses," in *Annual Network and Distributed System Security Symposium (NDSS)*, 2002, last visited on June 4th, 2012. [Online]. Available: http://www.isoc.org/isoc/conferences/ndss/02/proceedings.shtml

[182] R. Morselli, B. Bhattacharjee, J. Katz, and M. Marsh, "Keychains: A Decentralized Public-Key Infrastructure," University of Maryland (Department of Computer Science), Tech. Rep. CS-TR-4788, 2006.

[183] R. Moskowitz, P. Nikander, P. J. (Editor), and T. Henderson, "Host Identity Protocol," IETF, RFC 5201, Apr. 2008.

[184] C. Muus, "Availability in DHT-based Structured Overlay Networks Considering Chord as an Example," University of Hamburg, Germany (Department of Informatics), Diploma Thesis, Nov. 2007.

[185] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," Tech. Rep., 2009, last visited on June 4th, 2012. [Online]. Available: http://bitcoin.org/bitcoin.pdf

[186] K. Needels and M. Kwon, "Secure Routing in Peer-to-Peer Distributed Hash Tables," in *ACM Symposium on Applied Computing (SAC)*. ACM, New York, NY, USA, 2009.

[187] S. Niccolini and J. Quittek, "Signaling TO Prevent SPIT (SPITSTOP) Reference Scenario," IETF, Internet Draft draft-niccolini-sipping-spitstop-01, Feb. 2007, expired, last visited on June 4th, 2012. [Online]. Available: http://tools.ietf.org/html/draft-niccolini-sipping-spitstop-01

[188] S. Niccolini, S. Tartarelli, M. Stiemerling, and S. Srivastava, "SIP Extensions for SPIT identification," IETF, Internet Draft draft-niccolini-sipping-feedback-spit-03, Feb. 2007, expired, last visited on June 4th, 2012. [Online]. Available: http://tools.ietf.org/html/draft-niccolini-sipping-feedback-spit-03

[189] S. Niccolini, J. Seedorf, and N. d'Heureuse, "Method and System for Verifying the Identity of a Communication Partner," Patent PCT/EP08/03 508, 2008, patent application (grant pending).

[190] E. Nordmark and M. Bagnulo, "Shim6: Level 3 Multihoming Shim Protocol for IPv6," IETF, RFC 5533, Jun. 2009.

[191] C. O'Donnell and V. Vaikuntanathan, "Information Leak in the Chord Lookup Protocol," in *IEEE International Conference on Peer-to-Peer Computing*. IEEE Computer Society, Washington DC, USA, Aug. 2004, pp. 28–35.

[192] G. O'Shea and M. Roe, "Child-proof Authentication for MIPv6 (CAM)," in *Computer Communication Review*, vol. 31, no. 2, 2001, pp. 4–8.

[193] L. Overlier and P. Syverson, "Locating Hidden Servers," in *IEEE Symposium on Security and Privacy*. IEEE Computer Society, Washington DC, USA, 2006.

[194] E. Palomar, J. M. Estevez-Tapiador, J. C. Hernandez-Castro, and A. Ribagorda, "A P2P Content Authentication Protocol Based on Byzantine Agreement," in *International Conference on Emerging Trends in Information and Communication Security (ETRICS)*, ser. LNCS, no. 3995. Spinger, Heidelberg, Germany, 2006, pp. 60–72.

[195] A. Panchenko, S. Richter, and A. Rache, "NISAN: Network Information Service for Anonymization Networks," in *Proceedings of the 16th ACM Conference on Computer and Communications Security (CCS)*. ACM, New York, NY, USA, 2009.

[196] V. Pappas, D. Massey, A. Terzis, and L. Zhang, "A Comparative Study of the DNS Design with DHT-Based Alternatives," in *25th IEEE International Conference on Computer Communications (INFOCOM)*. IEEE Press, Piscataway, NJ, USA, 2006.

[197] V. Pathak and L. Iftode, "Byzantine Fault Tolerant Public Key Authentication in Peer-to-Peer Systems," *Elsevier Computer Networks*, vol. 50, no. 4, pp. 579–596, Mar. 2006.

[198] H. P. Penning, "Computing shortest paths in WOTs," Aug. 2004, last visited on June 4th, 2012. [Online]. Available: http://pgp.cs.uu.nl/doc/shortest-paths-in-wots.php

[199] H. P. Penning, "Analysis of the strong set in the PGP web of trust," Mar. 2009, last visited on June 4th, 2012. [Online]. Available: http://pgp.cs.uu.nl/plot/

[200] J. Peterson, "A Privacy Mechanism for the Session Initiation Protocol (SIP)," IETF, RFC 3323, Nov. 2002.

[201] J. Peterson and C. Jennings, "Enhancements for Authenticated Identity Management in the Session Initiation Protocol (SIP)," IETF, RFC 4474, Aug. 2006.

[202] C. P. Pfleeger, *Security in Computing*, 2nd ed. Prentice Hall International, Upper Saddle River, NJ, USA, 1997.

[203] J. Posegga and J. Seedorf, "Voice over IP: Unsafe at any Bandwidth?" in *Proceedings of Eurescom Summit 2005 Ubiquitous Services and Applications*. VDE Verlag, Berlin/Offenbach, Germany, Apr. 2005, pp. 305–314.

[204] J. Quittek, S. Niccolini, S. Tartarelli, and R. Schlegel, "On Spam over Internet Telephony (SPIT) Prevention," *IEEE Communications*, vol. 22, no. 5, 2008.

[205] J. Quittek, S. Niccolini, S. Tartarelli, M. Stiemerling, M. Brunner, and T. Ewald, "Detecting SPIT Calls by Checking Human Communication Patterns," in *Proceedings of the IEEE International Conference on Communications (ICC)*. IEEE Press, Piscataway, NJ, USA, Jun. 2007, pp. 1979–1984.

[206] R. Sparks (Editor), A. Hawrylyshen, A. Johnston, J. Rosenberg, and H. Schulzrinne, "Session Initiation Protocol (SIP) Torture Test Messages," IETF, RFC 4475, May 2006.

[207] V. Ramasubramanian and E. G. Sirer, "The Design and Implementation of a Next Generation Name Service for the Internet," in *Proceedings of the ACM SIGCOMM 2004 Conference*. ACM, New York, NY, USA, 2004.

[208] B. Ramsdell and S. Turner, "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.2 Message Specification," IETF, RFC 5751, Jan. 2010.

[209] I. Rappu, "Lawful Interception of VoIP in SIP-based Networks," Technical-University Hamburg-Harburg (TUHH), Germany, Project Work ("Studienarbeit"), May 2007.

[210] S. Ratnasamy, P. Francis, M. Handley, R. M. Karp, and S. Shenker, "A Scalable Content-Addressable Network," in *Proceedings of the ACM SIGCOMM 2001 Conference*. ACM, New York, NY, USA, 2001, pp. 161–172.

[211] S. Ratnasamy, M. Handley, R. M. Karp, and S. Shenker, "Application-Level Multicast Using Content-Addressable Networks," in *Proceedings of the Third International COST264 Workshop on Networked Group Communication*. Springer-Verlag, London, UK, 2001.

[212] Y. Rebahi and D. Sisalem, "Sip Service Providers and the Spam Problem," in *2nd Workshop on Securing Voice over IP*, Washington DC, USA, Jun. 2005, last visited on June 4th, 2012. [Online]. Available: http://www.iptel.org/~dor/papers/Rebahi0605_SIP.pdf

[213] M. G. Reed, P. F. Syverson, and D. M. Goldschlag, "Proxies for Anonymous Routing," in *Proceedings of the 12th Annual Computer Security Applications Conference (ACSAC)*. IEEE Computer Society, Washington, DC, USA, 1996.

[214] E. Rescorla, "HTTP Over TLS," IETF, RFC 2818, May 2000.

[215] S. Rhea, P. Eaton, D. Geels, H. Weatherspoon, B. Zhao, and J. Kubiatowicz, "Pond: The OceanStore Prototype," in *Proceedings of the 2nd USENIX conference on File and storage technologies*. USENIX Association, Berkeley, CA, USA, Mar. 2003.

[216] S. Rhea, D. Geels, T. Roscoe, and J. Kubiatowicz, "Handling Churn in a DHT," in *Proceedings of the USENIX Annual Technical Conference*. USENIX Association, Berkeley, CA, USA, 2004.

[217] R. Rivest, A. Shamir, and L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," *Communications of the ACM*, vol. 21, no. 2, Feb. 1978.

[218] J. Rosenberg, "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols," IETF, RFC 5245, Apr. 2010.

[219] J. Rosenberg, G. Camarillo, and D. Willis, "A Framework for Consent-Based Communications in the Session Initiation Protocol (SIP)," IETF, RFC 5360, Oct. 2008.

[220] J. Rosenberg and C. Jennings, "The Session Initiation Protocol (SIP) and Spam," IETF, RFC 5039, Jan. 2008.

[221] J. Rosenberg and H. Schulzrinne, "Session Initiation Protocol (SIP): Locating SIP Servers," IETF, RFC 3263, Jun. 2002.

[222] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler, "SIP: Session Initiation Protocol," IETF, RFC 3261, Jun. 2002.

[223] A. Rowstron and P. 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*. Springer-Verlag, London, UK, Nov. 2001.

[224] A. Rowstron, A.-M. Kermarrec, M. Castro, and P. Druschel, "SCRIBE: The Design of a Large-Scale Event Notification Infrastructure," in *Proceedings of the Third International COST264 Workshop on Networked Group Communication*. Springer-Verlag, London, UK, 2001.

[225] S. J. Russell and P. Norvig, *Artificial Intelligence - A Modern Approach*, 2nd ed. Pearson Education, Upper Saddle River, NJ, USA, 2003.

[226] S. Saroiu, P. K. Gummadi, and S. D. Gribble, "Measuring and Analyzing the Characteristics of Napster and Gnutella Hosts," *Springer Multimedia Systems*, vol. 9, no. 2, pp. 170–184, Aug. 2003.

[227] C. Scheideler, "How to Spread Adversarial Nodes?: Rotate!" in *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*. ACM, New York, NY, USA, 2005.

[228] R. Schlegel, S. Niccolini, S. Tartarelli, and M. Brunner, "SPam over Internet Telephony (SPIT) Prevention Framework," in *IEEE Global Telecommunications Conference (GLOBECOM)*. IEEE Press, Piscataway, NJ, USA, 2006.

[229] B. Schneier, "Schneier on Security: Cryptanalysis of SHA-1," Feb. 2005, last visited on June 4th, 2012. [Online]. Available: http://www.schneier.com/blog/archives/2005/02/cryptanalysis_o.html

[230] B. Schneier, "Schneier on Security: SHA-1 Broken," Feb. 2005, last visited on June 4th, 2012. [Online]. Available: http://www.schneier.com/blog/archives/2005/02/sha1_broken.html

[231] H. Schulzrinne, "A Uniform Resource Name (URN) for Emergency and Other Well-Known Services," IETF, RFC 5031, Jan. 2008.

[232] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications," IETF, RFC 3550, Jul. 2003.

[233] H. Schulzrinne and R. M. (Editor), "Requirements for Emergency Context Resolution with Internet Technologies," IETF, RFC 5012, Jan. 2008.

[234] H. Schulzrinne, A. Rao, and R. Lanphier, "Real Time Streaming Protocol (RTSP)," IETF, RFC 2326, Apr. 1998.

[235] J. Seedorf, "Security Challenges for Peer-to-Peer SIP," *IEEE Network*, vol. 20, no. 5, pp. 38–45, Sep. 2006.

[236] J. Seedorf and E. Burger, "Application-Layer Traffic Optimization (ALTO) Problem Statement," IETF, RFC 5693, Oct. 2009.

[237] J. Seedorf, S. Niccolini, E. Chen, and H. Scholz, "Session Peering for Multimedia Interconnect (SPEERMINT) Security Threats and Suggested Countermeasures," IETF, RFC 6404, Nov. 2011.

[238] J. Seedorf, "SIP Security - Status Quo and Future Issues," in *Proceedings of 23rd Chaos Communication Congress*, Nov. 2006, last visited on June 4th, 2012. [Online]. Available: http://events.ccc.de/congress/2006/Fahrplan/events/1459.en.html

[239] J. Seedorf, "Using Cryptographically Generated SIP-URIs to Protect the Integrity of Content in P2P-SIP," in *3rd Annual VoIP Security Workshop*, Jun. 2006, last visited on June 4th, 2012. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.89.3742&rep=rep1&type=pdf

[240] J. Seedorf, "Lawful Interception in P2P-Based VoIP Systems," in *Principles, Systems and Applications of IP Telecommunications (IPTComm)*, ser. LNCS, no. 5310. Springer-Verlag, Heidelberg, Germany, 2008, pp. 217–235.

[241] J. Seedorf, "Security Issues for P2P-Based Voice- and Video-Streaming Applications," in *Open Research Problems in Network Security (iNetSec)*, ser. International Federation for Information Processing (IFIP) AICT, no. 309. Springer-Verlag, Heidelberg, Germany, 2009, pp. 95–110.

[242] J. Seedorf, K. Beckers, and F. Huici, "Testing Dialog-Verification of SIP Phones with Single-Message Denial-of-Service Attacks," in *4th International Conference on Global E-Security*, ser. Communications in Computer and Information Science (CCIS), no. 12. Springer-Verlag, Heidelberg, Germany, 2008, pp. 61–64.

[243] J. Seedorf, K. Beckers, and F. Huici, "Single-Message Denial-of-Service Attacks Against Voice-over-Internet Protocol Terminals," *Inderscience International Journal of Electronic Security and Digital Forensics*, vol. 2, no. 1, pp. 29–34, Mar. 2009.

[244] J. Seedorf, N. d'Heureuse, S. Niccolini, and M. Cornolti, "Detecting Trustworthy Real-Time Communications using a Web-of-Trust," in *IEEE Global Telecommunications Conference (GLOBECOM)*. IEEE Press, Piscataway, NJ, USA, 2009.

[245] J. Seedorf, N. dHeureuse, S. Niccolini, and T. Ewald, "VoIP SEAL: A Research Prototype for Protecting Voice-over-IP Networks and Users," in *Konferenzband der 4. Jahrestagung des Fachbereichs Sicherheit der Gesellschaft für Informatik e.V. (GI) - Sicherheit 2008: Sicherheit, Schutz und Zuverlässigkeit*, ser. GI-Edition: Lecture Notes in Informatics (LNI), no. 128. Bonner Köllen Verlag, Bonn, Germany, 2008.

[246] J. Seedorf and C. Muus, "Availability for Structured Overlay Networks: Considerations for Simulation and a New Bound on Lookup Success," in *Proceedings of the 12th Nordic Workshop on Secure IT-Systems (Nordsec)*. Reykjavik University, Reykjavik, Iceland, Oct. 2007, pp. 23–34, (Reykjavik University Technical Report, ISBN: 978-9979948346).

[247] J. Seedorf and C. Muus, "Availability for DHT-Based Overlay Networks with Unidirectional Routing," in *Proceedings of the 2nd IFIP WG 11.2 international Conference on Information Security Theory and Practices: Smart Devices, Convergence and Next Generation Networks (WISTP)*, ser. LNCS, no. 5019. Springer, Heidelberg, Germany, 2008, pp. 78–91.

[248] J. Seedorf, F. Ruwolt, M. Stiemerling, and S. Niccolini, "Evaluating P2PSIP under Attack: An Emulative Study," in *IEEE Global Telecommunications Conference (GLOBECOM)*. IEEE Press, Piscataway, NJ, USA, 2008.

[249] J. Shi, Y. Wang, L. Gu, L. Li, W. Lin, Y. Li, Y. Ji, and P. Zhang, "A Hierarchical Peer-to-Peer SIP System for Heterogeneous Overlays Interworking," in *IEEE Global Telecommunications Conference (GLOBECOM)*. IEEE Press, Piscataway, NJ, USA, 2007.

[250] C. Shirky, "What is p2p... and what isn't," 2000, last visited on June 4th, 2012. [Online]. Available: http://www.openp2p.com/pub/a/p2p/2000/11/24/shirky1-whatisp2p.html

[251] A. Singh, M. Castro, P. Druschel, and A. Rowstron, "Defending Against Eclipse Attacks on Overlay Networks," in *Proceedings of the 11th workshop on ACM SIGOPS European workshop*. ACM, New York, NY, USA, 2004.

[252] K. Singh and H. Schulzrinne, "Peer-to-Peer Internet Telephony using SIP," in *Proceedings of the 15th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*. ACM, New York, NY, USA, 2005, pp. 63–68.

[253] K. Singh, "Reliable, Scalable and Interoperable Internet Telephony," Ph.D. dissertation, Columbia University (Computer Science Department), New York, USA, 2006.

[254] K. Singh and H. G. Schulzrinne, "Using an External DHT as a SIP Location Service," Columbia University (Computer Science Department), New York, USA, Tech. Rep., 2006, last visited on June 4th, 2012. [Online]. Available: http://academiccommons.columbia.edu/catalog/ac:110503

[255] H. Sinnreich and A. B. Johnston, *Internet Communications Using SIP - Delivering VoIP and Multimedia Services with Session Initiation Protocol*, 2nd ed. Wiley Publishing, Indianapolis, IN, USA, 2006.

[256] E. Sit and R. Morris, "Security Considerations for Peer-to-Peer Distributed Hash Tables," in *International Workshop on Peer-to-Peer Systems (IPTPS)*. iptps.org, 2002, last visited on June 4th, 2012. [Online]. Available: http://www.iptps.org/papers.html#2002

[257] E. Sit, R. Morris, and M. F. Kaashoek, "UsenetDHT: A Low-Overhead Design for Usenet," in *Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*. USENIX Association Berkeley, CA, US, 2008.

[258] H. Song, X. Jiang, R. Even, and D. Bryan, "P2PSIP Overlay Diagnostics," IETF, Internet Draft draft-ietf-p2psip-diagnostics-09, Aug. 2012, work in progress, last visited on August 27th, 2012. [Online]. Available: http://tools.ietf.org/html/draft-ietf-p2psip-diagnostics-09

[259] C. Sorge, S. Niccolini, and J. Seedorf, "The Legal Ramifications of Call-Filtering Solutions," *IEEE Security & Privacy*, vol. 8, no. 2, pp. 45–50, Mar. 2010.

[260] C. Sorge and J. Seedorf, "A Provider-Level Reputation System for Assessing the Quality of SPIT Mitigation Algorithms," in *Proceedings of the IEEE International Conference on Communications (ICC)*. IEEE Press, Piscataway, NJ, USA, 2009.

[261] M. Srivatsa and L. Liu, "Vulnerabilities and Security Threats in Structured Overlay Networks: A Quantitative Analysis," in *Proceedings of the 20th Annual Computer Security Applications Conference (ACSAC)*. IEEE Computer Society, Washington, DC, USA, Dec. 2004, pp. 252–261.

[262] W. Stallings, *Data and Computer Communications*, 8th ed. Prentice Hall International, Upper Saddle River, NJ, USA, 2007.

[263] M. Steiner, D. Carra, and E. W. Biersack, "Evaluating and Improving the Content Access in KAD," in *Springer Journal of Peer-to-Peer Networks and Applications*, vol. 3, no. 2, Jun. 2010, pp. 115–128.

[264] R. Steinmetz, S. Götz, and S. Rieche, *P2P Systems and Applications*, ser. LNCS. Springer-Verlag, Heidelberg, Germany, 2005, no. 3485, ch. Distributed Hash Tables, pp. 79–93 & 95–117.

[265] M. Stiemerling and M. Brunner, "A Peer-to-Peer SIP System based on Service-Aware Transport Overlays," in *IEEE International Conference on Peer-to-Peer Computing*. IEEE Computer Society, Washington DC, USA, 2008.

[266] I. Stoica, R. Morris, D. Karger, M. Kaashoek, and H. Balakrishnan, "Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications," in *Proceedings of the ACM SIGCOMM 2001 Conference*. ACM, New York, NY, USA, 2001.

[267] I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan, "Chord: A Scalable Peer-to-Peer Lookup Protocol for Internet Applications," *IEEE/ACM Transactions on Networking*, vol. 11, no. 1, Feb. 2003.

[268] J. Stribling, J. Li, I. G. Councill, M. F. Kaashoek, and R. Morris., "OverCite: A Distributed, Cooperative CiteSeer," in *Proceedings of the 3rd USENIX Symposium on Networked Systems Design and Implementation (NSDI)*. USENIX Association, Berkeley, CA, USA, 2006.

[269] K. Suh, D. R. Figueiredo, J. Kurose, and D. Towsley, "Characterizing and Detecting Relayed Traffic: A Case Study Using Skype," in *25th IEEE International Conference on Computer Communications (INFOCOM)*. IEEE Press, Piscataway, NJ, USA, 2006.

[270] S. Sutardi and J. Seedorf, "SISU - Ein Web-Service zum Testen der Sicherheit SIP-basierter Voice-over-IP Endgerte," in *Sicherheit in vernetzten Systemen: 15. DFN Workshop*. Books on Demand, Norderstedt, Germany, 2008.

[271] T. Taylor (Editor), H. Tschofenig, H. Schulzrinne, and M. Shanmugam, "Security Threats and Requirements for Emergency Call Marking and Mapping," IETF, RFC 5069, Jan. 2008.

[272] R. Tamassia and N. Triandopoulos, "Efficient Content Authentication in Peer-to-peer Networks," in *Proceedings of the 5th international conference on Applied Cryptography and Network Security (ACNS)*, ser. LNCS, no. 4521. Springer-Verlag, Heidelberg, Germany, 2007.

[273] A. S. Tanenbaum and M. van Steen, *Distributed Systems - Principles and Paradigms*, 2nd ed. Prentice Hall International, Upper Saddle River, NJ, USA, 2006.

[274] M. Thomson and J. Winterbottom, "Revised Civic Location Format for Presence Information Data Format Location Object (PIDF-LO)," IETF, RFC 5139, Feb. 2008.

[275] H. Tschofenig, E. Leppanen, S. Niccolini, and M. Arumaithurai, "Completely Automated Public Turing Test to Tell Computers and Humans Apart (CAPTCHA) based Robot Challenges for SIP," IETF, Internet Draft draft-tschofenig-sipping-captcha-01, Feb. 2008, expired, last visited on June 4th, 2012. [Online]. Available: http://tools.ietf.org/html/draft-tschofenig-sipping-captcha-01

[276] H. Tschofenig, H. Schulzrinne, D. Wing, J. Rosenberg, and D. Schwartz, "A Framework to Tackle Spam and Unwanted Communication for Internet Telephony," IETF, Internet Draft draft-tschofenig-sipping-framework-spit-reduction-04, Jul. 2008, expired, last visited on June 4th, 2012. [Online]. Available: http://tools.ietf.org/html/draft-tschofenig-sipping-framework-spit-reduction-04

[277] H. Tschofenig, D. Wing, H. Schulzrinne, T. Froment, and G. Dawirs, "A Document Format for Expressing Authorization Policies to Tackle Spam and Unwanted Communication for Internet Telephony," IETF, Internet Draft draft-tschofenig-sipping-spit-policy-03, Jul. 2008, expired, last visited on June 4th, 2012. [Online]. Available: http://tools.ietf.org/html/draft-tschofenig-sipping-spit-policy-03

[278] G. Urdaneta, G. Pierre, and M. van Steen, "A Survey of DHT Security Techniques," *ACM Computing Surveys*, vol. 43, no. 2, pp. 1–53, Jun. 2011.

[279] S. Čapkun, L. Buttyán, and J.-P. Hubaux, "Small Worlds in Security Systems: An Analysis of the PGP Certificate Graph," in *Proceedings of The ACM New Security Paradigms Workshop*. ACM. New York, NY, USA, 2002, pp. 28–35.

[280] D. S. Wallach, "A Survey of Peer-to-Peer Security Issues," in *Software Security - Theory and Systems (Mext-NSF-JSPS International Symposium)*, ser. LNCS, no. 2609. Springer-Verlag, Heidelberg, Germany, 2002.

[281] C. Wang and B. Li, "Peer-to-Peer Overlay Networks: A Survey," Hong Kong University of Science and Technology (Department of Computer Science), Hong Kong, China, Tech. Rep., 2003, last visited on June 4th, 2012. [Online]. Available: http://www.csun.edu/~andrzej/COMP529-S05/papers/TR-P2P.pdf

[282] P. Wang, I. Osipkov, N. Hopper, and Y. Kim, "Myrmic: Secure and Robust DHT Routing," University of Minnesota at Twin Cities (Digital Technology Center), MN, USA, Tech. Rep., 2007. [Online]. Available: http://www.dtc.umn.edu/publications/reports/2006_20.pdf

[283] Q. Wang and N. Borisov, "Octopus: A Secure and Anonymous DHT Lookup," in *32nd International Conference on Distributed Computing Systems (ICDCS)*. IEEE Computer Society, Washington DC, USA, 2012.

[284] X. Wang, Y. L. Yin, and H. Yu, "Collision Search Attacks on SHA1," Feb. 2005, last visited on June 4th, 2012. [Online]. Available: http://cryptome.org/sha1-attacks.htm

[285] N. Weaver, V. Paxson, S. Staniford, and R. Cunningham, "A Taxonomy of Computer Worms," in *Proceedings of the 2003 ACM workshop on Rapid malcode (WORM)*. ACM, New York, NY, USA, 2003.

[286] C. Wieser, M. Laakso, and H. Schulzrinne, "SIP Robustness Testing for Large-Scale Use," in *Testing of Component-Based Systems and Software Quality*, ser. GI-Edition: Lecture Notes in Informatics (LNI), no. 58. Bonner Kllen Verlag, Bonn, Germany, 2004, pp. 165–178.

[287] D. Wing, S. Niccolini, M. Stiemerling, and H. Tschofenig, "Spam Score for SIP," IETF, Internet Draft draft-wing-sipping-spam-score-02, Feb. 2008, expired, last visited on June 4th, 2012. [Online]. Available: http:/tools.ietf.org/html/draft-wing-sipping-spam-score

[288] H. Xie and Y. R. Yang, "A Measurement Based Study of the Skype Peer to Peer VoIP Performance," in *International Workshop on Peer-to-Peer Systems (IPTPS)*. iptps.org, 2007, last visited on June 4th, 2012. [Online]. Available: http://www.iptps.org/papers.html#2007

[289] H. Yan, E. Osterweily, J. Hajdu, J. Acres, and D. Massey, "Limiting Replay Vulnerabilities in DNSSEC," in *4th Workshop on Secure Network Protocols (NPSec)*. IEEE Computer Society, Washington DC, USA, 2008, pp. 3–8.

[290] J. Zar, D. Endler, D. Ghosal, R. Jafari, A. Karlcut, M. Kolenko, N. Nguyen, and W. Walkoe, "VoIP Security and Privacy Threat Taxonomy," Voice-over-IP Security Alliance (VOIPSA), Public Release v1.0, Oct. 2005, last visited on June 4th, 2012. [Online]. Available: http://www.voipsa.org/Activities/VOIPSA_Threat_Taxonomy_0.1.pdf

[291] H. Zhang, A. Goel, and R. Govindan, "Improving Lookup Latency in Distributed Hash Table Systems using Random Sampling," *IEEE/ACM Transactions on Networking*, vol. 13, no. 5, pp. 1121–1134, Oct. 2005.

[292] R. Zhang, X. Wang, X. Yang, and X. Jiang, "Billing Attacks on SIP-Based VoIP Systems," in *Proceedings of the first USENIX workshop on Offensive Technologies (WOOT '07)*. USENIX Association Berkeley, USA, 2007.

[293] B. Zhao, L. Huang, J. Stribling, S. Rhea, A. Joseph, and J. Kubiatowicz, "Tapestry: A Resilient Global-Scale Overlay for Service Deployment," *IEEE Journal on Selected Areas in Communications*, vol. 22, no. 1, pp. 41–53, Jan. 2004.

[294] X. Zheng and V. Oleshchuk, "Improvement of Chord Overlay for P2PSIP-based Communication Systems," *AIRCC International Journal of Computer Networks & Communications (IJCNC)*, vol. 1, no. 3, Oct. 2009, last visited on August 27th, 2012. [Online]. Available: http://airccse.org/journal/cnc/1009s13.pdf

[295] F. Zhou, L. Zhuang, B. Y. Zhao, L. Huang, A. D. Joseph, and J. Kubiatowicz, "Approximate Object Location and Spam Filtering on Peer-to-peer Systems," in *Proceedings of the ACM/IFIP/USENIX International Conference on Middleware*, ser. LNCS, no. 2672. Springer-Verlag, Heidelberg, Germany, 2003.

[296] S. Q. Zhuang, B. Y. Zhao, A. D. Joseph, R. H. Katz, and J. D. Kubiatowicz, "Bayeux: An Architecture for Scalable and Fault-Tolerant Wide-Area Data Dissemination," in *Proceedings of the 11th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*. ACM, New York, NY, USA, 2001, pp. 11–20.

[297] P. Zimmermann, A. Johnston, and J. Callas, " ZRTP: Media Path Key Agreement for Unicast Secure RTP," IETF, RFC 6189, Apr. 2011.

[298] P. Zimmermann, *The official PGP user's guide*. MIT Press, Cambridge, MA, USA, May 1995.

[299] N. Zong, X. Jiang, R. Even, and Y. Zhang, "An Extension to RELOAD to Support Direct Response Routing," IETF, Internet Draft draft-ietf-p2psip-drr-02, May 2012, work in progress, last visited on August 27th, 2012. [Online]. Available: http://tools.ietf.org/html/draft-ietf-p2psip-drr-02

[300] N. Zong, X. Jiang, R. Even, and Y. Zhang, "An Extension to RELOAD to Support Relay Peer Routing," IETF, Internet Draft draft-ietf-p2psip-rpr-02, May 2012, work in progress, last visited on August 27th, 2012. [Online]. Available: http://tools.ietf.org/html/draft-ietf-p2psip-rpr-02

Appendix A

Previously Published and Related Publications

A considerable part of this thesis including figures has been originally published in conferences, journals, or by other means. Below we list these associated publications which have been published prior to the publication of this thesis, and explain how they relate to this thesis.

A.1 Peer-reviewed Publications in Scientific Conferences or Journals

- J. Seedorf, "Using Cryptographically Generated SIP-URIs to Protect the Integrity of Content in P2P-SIP," in 3rd Annual VoIP Security Workshop, Jun. 2006, last visited on June 4th, 2012. [Online]. Available: `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.89.3742&rep=rep1&type=pdf` [239];
  This publication describes the approach of self-certifying SIP-URIs and how this concept can protect the integrity of data items in a P2PSIP DHT. The content of the paper is to large extent contained in Chapter 5 of this thesis, where self-certifying SIP-URIs and the applicability of this approach for integrity protection of data items in P2PSIP are being presented. The paper was solely written by the author of this thesis.

- J. Seedorf, "Security Challenges for Peer-to-Peer SIP," IEEE Network, vol. 20, no. 5, pp. 38–45, Sep. 2006. [235];
  This article investigates the security challenges of using a P2P network as a substrate for SIP communication. It identifies the security implications of using a structured overlay network for SIP registration and location lookup and examines possible solutions for securing P2PSIP. The content of this article is largely contained in Chapter 3 of this thesis, where a security analysis of P2PSIP is being provided and the corresponding research gap is being derived. This publication was

solely written by the author of this thesis.

- J. Seedorf and C. Muus, "Availability for Structured Overlay Networks: Considerations for Simulation and a new Bound on Lookup Success," in Proceedings of the 12th Nordic Workshop on Secure IT-Systems (Nordsec). Reykjavik University, Reykjavik, Iceland, Oct. 2007, pp. 23–34, (Reykjavik University Technical Report, ISBN: 978-9979948346). [246];
  This paper contributes analytical bounds for lookup availability in Chord, analyses the simulation methodology used in publications on previous approaches for lookup availability in Chord, and proposes considerations for simulating unidirectional Distributed Hash Tables like Chord. These contributions are contained in Chapter 4 of this thesis, where Chord is being analysed analytically and previous approaches for lookup availability in Chord are examined. The paper is the result of a Diploma thesis of the second author of the paper under the supervision of the author of this thesis. In the Diploma thesis a DHT security simulator was developed, and several Chord security extensions were developed and implemented in the simulator. The paper was written by the author of this thesis; the second author conducted the simulations for the results published in the paper.

- J. Seedorf, "Lawful Interception in P2P-Based VoIP Systems," in Principles, Systems and Applications of IP Telecommunications (IPTComm), ser. LNCS, no. 5310. Springer-Verlag, Heidelberg, Germany, 2008, pp. 217–235. [240];
  This paper provides a technical analysis of applying Lawful Interception to P2P-based Voice-over-IP systems, highlighting the characteristic properties of such an approach and the corresponding implications that complicate Lawful Interception. Further, potential solutions for implementing Lawful Interception in a P2PSIP system are examined. The content of this paper is to large extent contained in Chapter 7 of this thesis, where a technical analysis of Lawful Interception in the context of P2PSIP is provided. This publication was solely written by the author of this thesis.

- J. Seedorf and C. Muus, "Availability for DHT-Based Overlay Networks with Unidirectional Routing," in Proceedings of the 2nd IFIP WG 11.2 International Conference on Information Security Theory and Practices: Smart Devices, Convergence and Next Generation Networks (WISTP), ser. LNCS, no. 5019. Springer, Heidelberg, Germany, 2008. pp. 78–91 [247];
  This paper contributes concrete algorithms for maintaining lookup availability in a Chord DHT in the presence of adversary nodes. These contributions are contained in Chapter 4 of this thesis, where these Chord extensions are being presented and analysed in detail. The paper is the result of a Diploma thesis of the second author of the paper under the supervision of the author of this thesis. In the Diploma thesis a DHT security simulator was developed, and several Chord security extensions were developed and implemented in the simulator. The paper was written by the author of this thesis; the second author conducted the simulations for the results published in the paper.

- J. Seedorf, K. Beckers, and F. Huici, "Testing Dialog-Verification of SIP Phones with Single-Message Denial-of-Service Attacks" in 4th International Conference on

Global E-Security, ser. Communications in Computer and Information Science (CCIS), no. 12. Springer-Verlag, Heidelberg, Germany, 2008, pp. 61–64. [242];
This paper presents experimental results of testing several SIP terminals against deficient verification of SIP dialog-IDs. This research is not directly related to the core contribution of this thesis. The content of this paper is merely summarized as relevant background to our work in Subsection 2.2.3.2 (re-using one figure from [242]). The paper was written by the first and third author; the second author provided the prototypical implementation of the test environment as part of his Diploma thesis under the supervision of the author of this thesis.

- J. Seedorf, F. Ruwolt, M. Stiemerling, and S. Niccolini, "Evaluating p2psip under attack: An emulative study," in IEEE Global Telecommunications Conference (GLOBECOM). IEEE Press, Piscataway, NJ, USA, 2008. [248];
  This paper presents the prototypical implementation of a security-enhanced P2PSIP system and emulation results obtained with this prototype. The content of the paper is contained in Chapter 8, where this prototype (and corresponding results obtained with it) are being presented in this thesis. The paper is the result of the Diploma thesis of the second author. During the course of that Diploma thesis, the P2PSIP prototype described in the paper was developed under the supervision of the author of this thesis. The paper was written mostly by the author of this thesis, with advise from the third and fourth author.

- J. Seedorf, K. Beckers, and F. Huici, "Single-Message Denial-of-Service Attacks Against Voice-over-Internet Protocol Terminals," Inderscience International Journal of Electronic Security and Digital Forensics, vol. 2, no. 1, pp. 29–34, Mar. 2009. [243];
  This paper presents experimental results of testing several SIP terminals against deficient verification of SIP dialog-IDs and is an extended version of [242]. This research is not directly related to the core contribution of this thesis. The content of this paper is merely summarized as relevant background to our work in Subsection 2.2.3.2. The paper was written by the first and third author; the second author provided the prototypical implementation of the test environment as part of his Diploma thesis under the supervision of the author of this thesis.

- J. Seedorf, "Security Issues for P2P-Based Voice- and Video-Streaming Applications," in Open Research Problems in Network Security (iNetSec), ser. International Federation for Information Processing (IFIP) AICT, no. 309. Springer-Verlag, Heidelberg, Germany, 2009, pp. 95–110. [241];
  This paper discusses new, interesting security challenges imposed by P2P-based voice and video streaming systems. The paper presents a summary of existing work in the area, derives and discusses open research problems, and outlines approaches towards potential solutions for securing P2P-based voice and video streaming applications. Several figures from this publication are used in Chapter 2 in this thesis. Also, the idea of an application-intrinsic social network is being discussed in this paper and also appears in Chapter 8 of this thesis (also re-using a figure from the paper), where an application-intrinsic P2PSIP social network is being proposed. This publication was solely written by the author of this thesis.

- J. Seedorf, N. d'Heureuse, S. Niccolini, and M. Cornolti, "Detecting Trustworthy Real-Time Communications Using a Web-of-Trust," in IEEE Global Telecommunications Conference (GLOBECOM). IEEE Press, Piscataway, NJ, USA, 2009. [244];
  This paper describes a Web-of-Trust-based approach for decentralised identity assertion. Parts of this paper are contained in Chapter 6 of this thesis, where this Web-of-Trust approach is being presented in this thesis. However, the paper only covers the application of the approach in client-server systems whereas Chapter 6 also provides a detailed design and analysis of applying this approach in P2P-based systems. This application of the approach to P2PSIP was developed solely by the author of this thesis. The general Web-of-Trust approach was developed and the paper was written mostly together by the first and second author. The third author contributed advise in the design and paper writing. The fourth author contributed the prototypical implementation of the approach as part of an internship under the supervision of the first and second author of the paper.

A.2 Other Pre-published Publications

- J. Posegga and J. Seedorf, "Voice over IP: Unsafe at any Bandwidth?" in Proceedings of Eurescom Summit 2005 Ubiquitous Services and Applications. VDE Verlag, Berlin/Offenbach, Germany, Apr. 2005, pp. 305–314. [203];
  This paper provides a technical analysis of the security aspects of VoIP, focusing on the SIP protocol. The major differences and implications of VoIP, in particular compared to circuit-switched voice are being discussed. Parts of this paper are contained in Section 2.2 of this thesis, where VoIP, SIP, and the related security issues are being introduced. The paper was written together by the first and second author.

- J. Seedorf, "SIP Security - Status Quo and Future Issues," in Proceedings of 23rd Chaos Communication Congress, Nov. 2006, last visited on June 4th, 2012. [Online]. Available: `http://events.ccc.de/congress/2006/Fahrplan/events/1459.en.html` [238];
  This paper provides an overview of VoIP security issues, focusing on SIP. In addition, an overview on research activities and standardisation efforts in the field of VoIP security is given. The paper also contains a high-level overview on P2PSIP security issues. Parts of this paper are contained in Section 2.2 of this thesis, where VoIP, SIP, and the related security issues are being introduced, and in Chapter 3 of this thesis, where a security analysis of P2PSIP is being provided. This paper was written solely by the author of this thesis.

- S. Niccolini, J. Seedorf, and N. d'Heureuse, "Method and System for Verifying the Identity of a Communication Partner," Patent PCT/EP08/03 508, 2008, patent application (grant pending) [189];
  This patent application relates to the Web-of-Trust-based approach for decentralised identity assertion published in [244]. The content of this patent application is in part contained in Chapter 6 of this thesis, where this Web-of-Trust approach is being presented in this thesis.

Appendix B

Implementation and Experiment Details

B.1 Algorithms for Increased Lookup Availability: Detailed Simulation Results

B.1.1 Experimental Setup

Our simulator is an extension of the Java-based *planetsim* [15] DHT simulation framework. Java classes for modelling our attacker model (e.g. adversary nodes routing exclusively to adversary nodes) and for routing with our DHT-extensions have been implemented and added to planetsim (or existing classes have been modified). All our simulations have been carried out according to our proposed simulation methodology (see Section 4.6): We simulated *key lookup* (see 4.5.1.2) in a hardly utilised node-ID space. Further, each parametrisation has been simulated for different network sizes. All our experiments have been conducted on one of two physical hosts (AMD Phenom Quadcore, 8GB Ram; AMD Athlon II Quadcore, 16GB Ram) under Debian Linux (AMD 64-bit).

For all our experiments, we set $s = 16$ and $r = 8$ and executed 100 random lookups each in 10 randomly created Chord DHT networks (i.e. 10 experimental runs each consisting of 100 lookups). The average results for lookup success and hop count provided below and in Chapter 4 are obtained by averaging the 100-lookup average of each individual run over the 10 executed runs. The standard-deviation of the lookup success refers to the overall average among the 10 runs.

B.1.2 Additional and Detailed Results

Figure B.1 shows the success rate for *MRR-r* with a hop count threshold, t_h, of 50 and different density thresholds, t_d, in a network of $N = 1000$ nodes. Figure B.2 shows

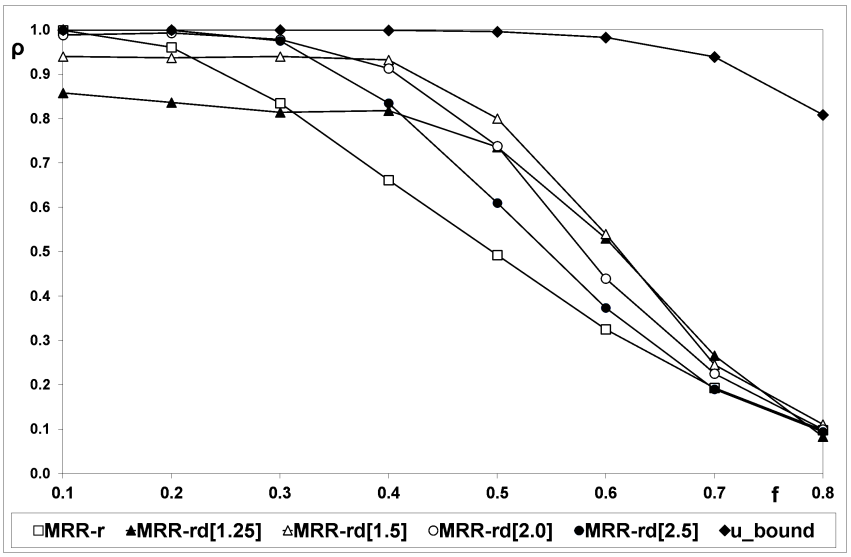

Figure B.1: Success Rate for *MRR-r* with Different Density Thresholds t_d (t_h = 50, N= 1000)

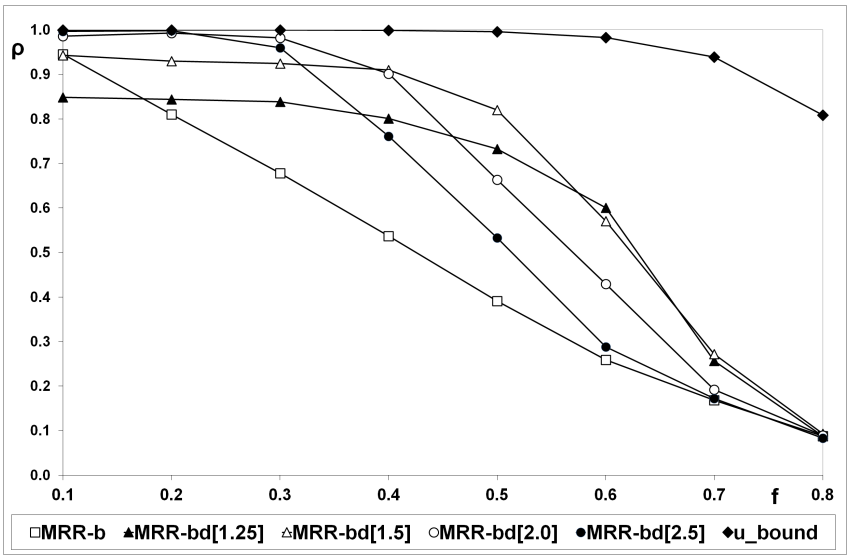

Figure B.2: Success Rate for *MRR-b* with Different Density Thresholds t_d (t_h = 50, N= 1000)

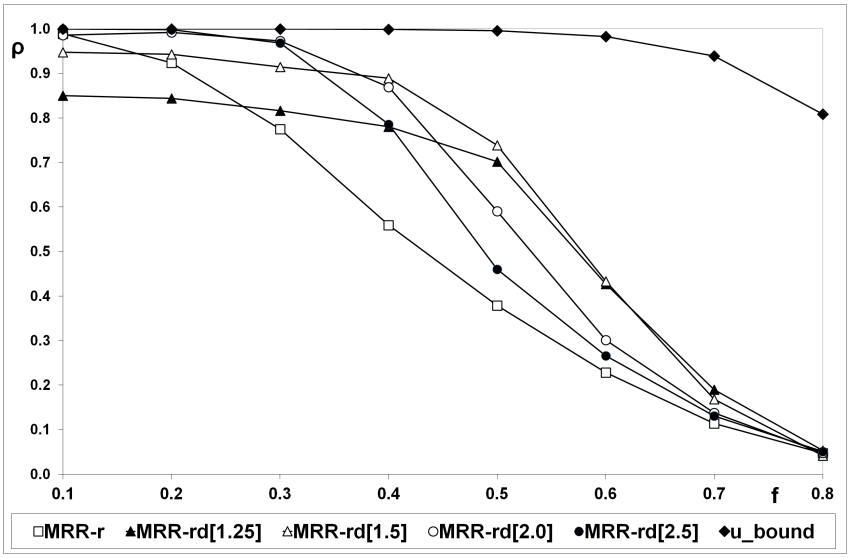

Figure B.3: Success Rate for *MRR-r* with Different Density Thresholds t_d ($t_h = 50$, N= 2000)

similar results for *MRR-b*. Figure B.3 shows the success rate for *MRR-r* with a hop count threshold, t_h, of 50 and different density thresholds, t_d, in a network of $N = 2000$ nodes. Figure B.4 shows similar results for *MRR-b*. Figure B.5 shows the success rate for *MRR-b* with a hop count threshold, t_h, of 50 and different density thresholds, t_d, in a network of $N = 4000$ nodes (a similar figure for $N = 4000$ and *MRR-r* is shown in Subsection 4.4.2).

Figure B.6 visualises the effect of an increased hop count threshold, t_h, of 100 in comparison to $t_h = 50$. The figure shows results for *MRR-r*, *MRR-b*, and *MRR-r* with the best density threshold for moderate attacker rates, $t_d = 2.0$, in a network of $N = 1000$ nodes. Figure B.7 displays similar results for *MRR-r* and *MRR-b* in a network of $N = 4000$ nodes.

Figures B.8, B.9, B.10, B.11, B.12, B.13, B.14, B.15, and B.16 show detailed results of our experiments. In particular, for each parametrisation we simulated, the average lookup success rate (and standard deviation) as well as the average hop count we obtained is provided.

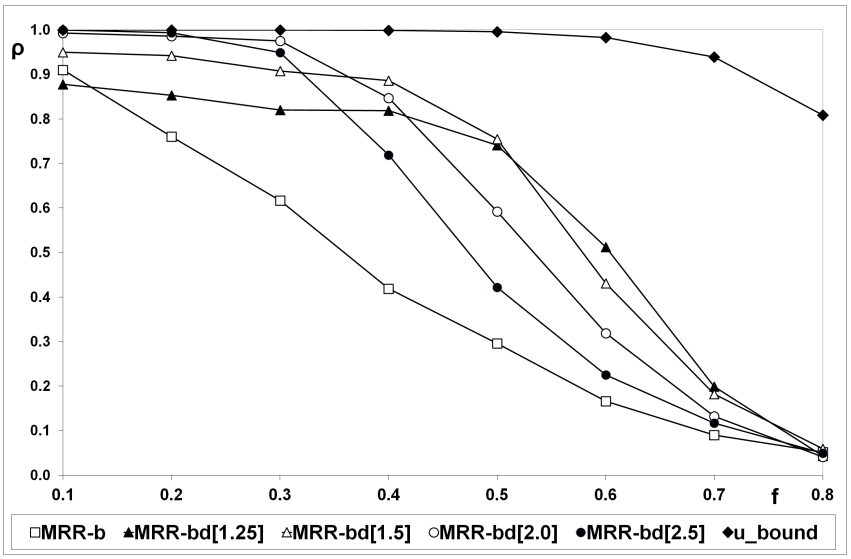

Figure B.4: Success Rate for *MRR-b* with Different Density Thresholds t_d (t_h = 50, N= 2000)

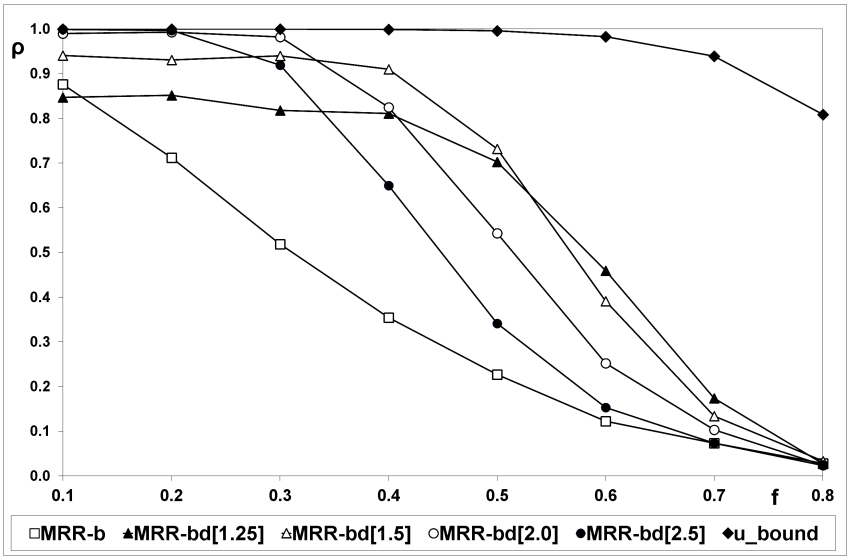

Figure B.5: Success Rate for *MRR-b* with Different Density Thresholds t_d (t_h = 50, N= 4000)

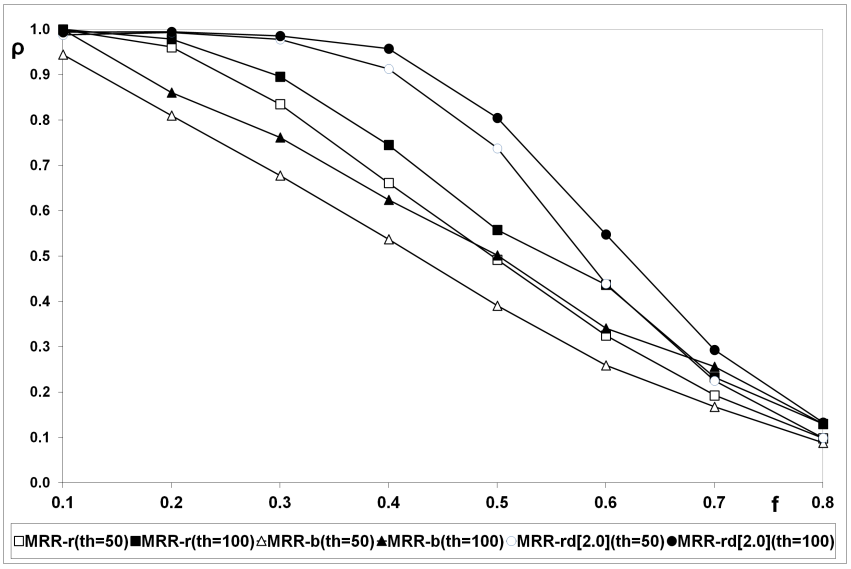

Figure B.6: Effect of Increased Hop Threshold t_h for Different Algorithms (N= 1000)

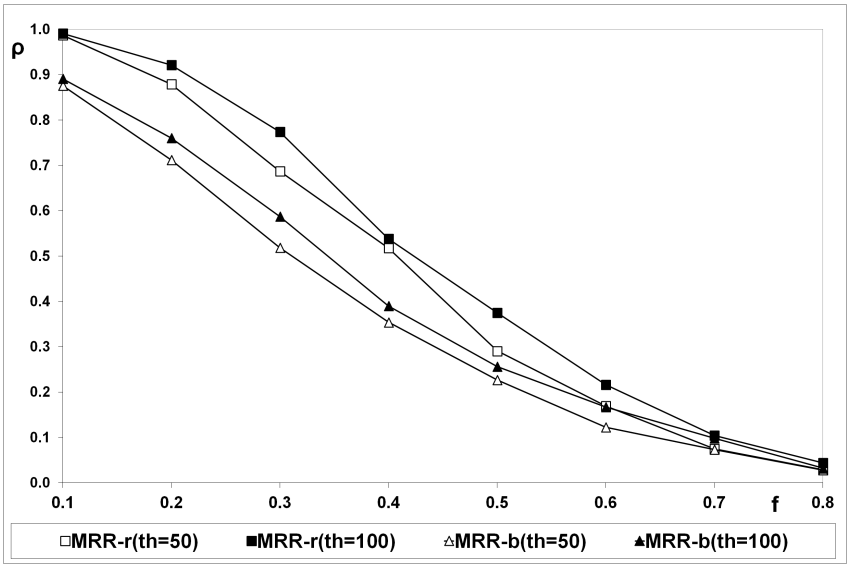

Figure B.7: Effect of Increased Hop Threshold t_h for Different Algorithms (N= 4000)

N	Algorithm	hop-threshold	density-threshold	f	Average Success Rate	stddev(Average Success Rate)	Average Hop Count
1000	MRR-b			0.1	100.00	0.00	10.38
	MRR-b			0.2	100.00	0.00	28.11
	MRR-b			0.3	99.90	0.31	65.06
	MRR-b			0.4	99.85	0.37	100.60
	MRR-b			0.5	99.45	1.00	159.03
	MRR-b			0.6	97.85	1.93	227.47
	MRR-b			0.7	91.94	3.70	326.05
	MRR-b			0.8	71.30	6.34	491.64
	MRR-b	50		0.1	94.50	1.58	8.27
	MRR-b	50		0.2	81.00	3.06	15.54
	MRR-b	50		0.3	67.80	4.80	22.97
	MRR-b	50		0.4	53.70	5.38	29.51
	MRR-b	50		0.5	39.10	4.93	35.42
	MRR-b	50		0.6	25.90	3.93	41.23
	MRR-b	50		0.7	16.80	3.71	44.15
	MRR-b	50		0.8	8.80	2.35	47.08
	MRR-b	50	1.25	0.1	84.90	5.11	12.88
	MRR-b	50	1.25	0.2	84.40	3.37	13.13
	MRR-b	50	1.25	0.3	83.90	4.43	13.94
	MRR-b	50	1.25	0.4	80.10	6.15	16.64
	MRR-b	50	1.25	0.5	73.30	7.23	20.82
	MRR-b	50	1.25	0.6	60.10	5.36	28.49
	MRR-b	50	1.25	0.7	25.70	3.92	42.40
	MRR-b	50	1.25	0.8	8.80	4.10	47.20
	MRR-b	50	1.50	0.1	94.30	2.98	7.75
	MRR-b	50	1.50	0.2	93.00	2.58	8.78
	MRR-b	50	1.50	0.3	92.50	2.92	9.33
	MRR-b	50	1.50	0.4	91.00	2.62	11.25
	MRR-b	50	1.50	0.5	82.00	4.45	17.82
	MRR-b	50	1.50	0.6	57.10	6.71	29.94
	MRR-b	50	1.50	0.7	27.20	8.13	41.31
	MRR-b	50	1.50	0.8	9.30	2.31	47.12
	MRR-b	50	2.00	0.1	98.60	1.50	5.45
	MRR-b	50	2.00	0.2	99.30	1.17	5.47
	MRR-b	50	2.00	0.3	98.25	1.41	6.94
	MRR-b	50	2.00	0.4	90.15	3.67	12.94
	MRR-b	50	2.00	0.5	66.35	5.28	25.33
	MRR-b	50	2.00	0.6	42.95	6.62	35.39
	MRR-b	50	2.00	0.7	19.25	4.15	43.86
	MRR-b	50	2.00	0.8	8.95	2.89	47.25
	MRR-b	50	2.50	0.1	99.70	0.48	4.93
	MRR-b	50	2.50	0.2	99.80	0.42	5.15
	MRR-b	50	2.50	0.3	96.00	1.89	8.76
	MRR-b	50	2.50	0.4	76.10	5.30	19.97
	MRR-b	50	2.50	0.5	53.30	6.67	30.36
	MRR-b	50	2.50	0.6	28.80	6.29	40.40
	MRR-b	50	2.50	0.7	17.20	4.21	44.45
	MRR-b	50	2.50	0.8	8.30	3.33	47.51
	MRR-b	100		0.1	100.00	0.00	10.71
	MRR-b	100		0.2	86.10	2.08	22.51
	MRR-b	100		0.3	76.20	5.18	35.53
	MRR-b	100		0.4	62.40	4.40	51.53
	MRR-b	100		0.5	50.20	5.12	62.01
	MRR-b	100		0.6	34.10	5.55	75.69
	MRR-b	100		0.7	25.60	4.45	83.45
	MRR-b	100		0.8	13.00	3.80	92.59

Figure B.8: Detailed Simulation Results (MRR-b, $N = 1000$)

N	Algorithm	hop-threshold	density-threshold	f	Average Success Rate	stddev(Average Success Rate)	Average Hop Count
1000	MRR-r			0.1	100.00	0.00	5.98
	MRR-r			0.2	100.00	0.00	13.71
	MRR-r			0.3	99.85	0.37	35.59
	MRR-r			0.4	100.00	0.00	69.39
	MRR-r			0.5	99.25	1.16	136.77
	MRR-r			0.6	98.00	1.65	209.51
	MRR-r			0.7	89.00	4.83	341.01
	MRR-r			0.8	69.50	5.60	498.73
	MRR-r	50		0.1	99.90	0.32	5.89
	MRR-r	50		0.2	96.10	1.29	10.18
	MRR-r	50		0.3	83.50	3.37	17.29
	MRR-r	50		0.4	66.10	5.99	26.02
	MRR-r	50		0.5	49.20	8.07	33.52
	MRR-r	50		0.6	32.50	3.89	39.56
	MRR-r	50		0.7	19.30	5.23	44.06
	MRR-r	50		0.8	9.80	4.29	46.80
	MRR-r	50	1.25	0.1	85.80	5.55	12.23
	MRR-r	50	1.25	0.2	83.60	4.84	14.07
	MRR-r	50	1.25	0.3	81.40	3.53	15.51
	MRR-r	50	1.25	0.4	81.80	5.59	16.23
	MRR-r	50	1.25	0.5	73.60	6.72	22.37
	MRR-r	50	1.25	0.6	53.00	4.29	32.64
	MRR-r	50	1.25	0.7	26.60	6.40	41.97
	MRR-r	50	1.25	0.8	8.30	2.16	47.44
	MRR-r	50	1.50	0.1	94.00	2.75	8.10
	MRR-r	50	1.50	0.2	93.70	2.95	8.88
	MRR-r	50	1.50	0.3	94.00	3.20	9.53
	MRR-r	50	1.50	0.4	93.20	1.23	11.83
	MRR-r	50	1.50	0.5	80.00	5.70	20.41
	MRR-r	50	1.50	0.6	54.00	4.94	32.26
	MRR-r	50	1.50	0.7	24.50	4.79	42.19
	MRR-r	50	1.50	0.8	11.10	3.87	46.35
	MRR-r	50	2.00	0.1	98.80	1.40	5.51
	MRR-r	50	2.00	0.2	99.30	1.06	6.07
	MRR-r	50	2.00	0.3	97.80	1.32	7.72
	MRR-r	50	2.00	0.4	91.30	2.98	14.17
	MRR-r	50	2.00	0.5	73.80	2.49	24.12
	MRR-r	50	2.00	0.6	43.90	4.63	35.45
	MRR-r	50	2.00	0.7	22.50	2.68	42.84
	MRR-r	50	2.00	0.8	9.90	2.56	46.50
	MRR-r	50	2.50	0.1	99.90	0.31	5.05
	MRR-r	50	2.50	0.2	99.90	0.45	5.60
	MRR-r	50	2.50	0.3	97.50	1.32	9.07
	MRR-r	50	2.50	0.4	83.50	3.83	18.21
	MRR-r	50	2.50	0.5	60.95	4.87	29.18
	MRR-r	50	2.50	0.6	37.30	4.13	37.78
	MRR-r	50	2.50	0.7	18.95	3.35	43.97
	MRR-r	50	2.50	0.8	9.35	2.52	47.13
	MRR-r	100		0.1	100.00	0.00	5.86
	MRR-r	100		0.2	97.90	1.91	12.05
	MRR-r	100		0.3	89.60	3.89	24.01
	MRR-r	100		0.4	74.50	4.97	40.21
	MRR-r	100		0.5	55.80	5.47	58.69
	MRR-r	100		0.6	43.70	5.33	69.36
	MRR-r	100		0.7	23.30	3.62	84.77
	MRR-r	100		0.8	13.00	2.87	91.39
	MRR-r	100	1.50	0.1	95.18	2.29	10.62
	MRR-r	100	1.50	0.2	94.25	2.53	11.86
	MRR-r	100	1.50	0.3	93.48	3.11	13.42
	MRR-r	100	1.50	0.4	91.80	3.54	17.33
	MRR-r	100	1.50	0.5	86.00	4.31	28.21
	MRR-r	100	1.50	0.6	65.35	5.58	51.99
	MRR-r	100	1.50	0.7	35.38	5.07	77.27
	MRR-r	100	1.50	0.8	13.31	3.60	91.66
	MRR-r	100	2.00	0.1	99.37	0.93	5.68
	MRR-r	100	2.00	0.2	99.47	0.78	6.08
	MRR-r	100	2.00	0.3	98.57	1.25	8.46
	MRR-r	100	2.00	0.4	95.73	1.80	16.26
	MRR-r	100	2.00	0.5	80.50	3.49	37.03
	MRR-r	100	2.00	0.6	54.80	6.74	61.92
	MRR-r	100	2.00	0.7	29.30	5.44	81.39
	MRR-r	100	2.00	0.8	13.29	3.38	91.49
	MRR-r	100	2.50	0.1	99.70	0.47	5.35
	MRR-r	100	2.50	0.2	99.85	0.49	5.85
	MRR-r	100	2.50	0.3	99.20	0.70	9.55
	MRR-r	100	2.50	0.4	89.35	2.89	25.44
	MRR-r	100	2.50	0.5	69.90	4.14	47.26
	MRR-r	100	2.50	0.6	46.80	6.07	67.08
	MRR-r	100	2.50	0.7	27.65	4.12	82.02
	MRR-r	100	2.50	0.8	12.95	3.47	91.77

Figure B.9: Detailed Simulation Results (MRR-r, $N = 1000$)

N	Algorithm	hop-threshold	density-threshold	f	Average Success Rate	stddev(Average Success Rate)	Average Hop Count
2000	MRR-b			0.1	100.00	0.00	17.45
	MRR-b			0.2	100.00	0.00	55.29
	MRR-b			0.3	100.00	0.00	126.18
	MRR-b			0.4	99.90	0.32	192.24
	MRR-b			0.5	99.70	0.48	316.25
	MRR-b			0.6	97.00	1.56	483.94
	MRR-b			0.7	94.30	3.20	629.24
	MRR-b			0.8	74.30	5.50	991.83
	MRR-b	50		0.1	91.00	3.50	10.51
	MRR-b	50		0.2	76.00	7.62	18.47
	MRR-b	50		0.3	61.70	3.80	25.86
	MRR-b	50		0.4	41.90	3.84	33.97
	MRR-b	50		0.5	29.60	4.77	39.31
	MRR-b	50		0.6	16.60	4.45	44.17
	MRR-b	50		0.7	9.00	3.09	47.03
	MRR-b	50		0.8	5.20	2.15	48.24
	MRR-b	50	1.25	0.1	87.80	3.22	12.08
	MRR-b	50	1.25	0.2	85.30	4.16	13.51
	MRR-b	50	1.25	0.3	82.00	4.35	15.48
	MRR-b	50	1.25	0.4	81.90	3.87	15.99
	MRR-b	50	1.25	0.5	74.10	4.58	21.53
	MRR-b	50	1.25	0.6	51.20	5.85	32.98
	MRR-b	50	1.25	0.7	19.90	3.38	43.97
	MRR-b	50	1.25	0.8	4.30	2.75	48.67
	MRR-b	50	1.50	0.1	95.00	2.05	8.07
	MRR-b	50	1.50	0.2	94.20	2.66	8.91
	MRR-b	50	1.50	0.3	90.80	6.16	10.71
	MRR-b	50	1.50	0.4	88.60	3.92	13.32
	MRR-b	50	1.50	0.5	75.50	3.44	21.10
	MRR-b	50	1.50	0.6	43.10	7.78	35.95
	MRR-b	50	1.50	0.7	18.20	3.74	44.14
	MRR-b	50	1.50	0.8	5.90	2.47	48.25
	MRR-b	50	2.00	0.1	99.30	1.08	5.67
	MRR-b	50	2.00	0.2	98.60	1.39	6.36
	MRR-b	50	2.00	0.3	97.55	1.64	7.88
	MRR-b	50	2.00	0.4	84.70	4.16	16.16
	MRR-b	50	2.00	0.5	59.20	4.47	28.11
	MRR-b	50	2.00	0.6	31.90	5.86	39.45
	MRR-b	50	2.00	0.7	13.20	4.69	45.93
	MRR-b	50	2.00	0.8	4.10	2.08	48.64
	MRR-b	50	2.50	0.1	100.00	0.00	5.27
	MRR-b	50	2.50	0.2	99.40	0.97	6.02
	MRR-b	50	2.50	0.3	94.90	1.73	10.03
	MRR-b	50	2.50	0.4	71.90	4.33	21.70
	MRR-b	50	2.50	0.5	42.20	5.96	34.77
	MRR-b	50	2.50	0.6	22.50	3.87	42.30
	MRR-b	50	2.50	0.7	11.70	2.45	46.11
	MRR-b	50	2.50	0.8	4.90	2.51	48.35
	MRR-b	100		0.1	91.00	1.76	15.66
	MRR-b	100		0.2	80.20	4.05	28.74
	MRR-b	100		0.3	65.90	5.76	44.28
	MRR-b	100		0.4	51.60	5.83	58.69
	MRR-b	100		0.5	36.10	5.30	73.18
	MRR-b	100		0.6	24.30	5.25	82.93
	MRR-b	100		0.7	15.20	4.85	90.52
	MRR-b	100		0.8	6.40	3.63	95.77

Figure B.10: Detailed Simulation Results (*MRR-b*, $N = 2000$)

N	Algorithm	hop-threshold	density-threshold	f	Average Success Rate	stddev(Average Success Rate)	Average Hop Count
2000	MRR-r			0.1	100.00	0.00	8.01
	MRR-r			0.2	100.00	0.00	23.39
	MRR-r			0.3	100.00	0.00	70.57
	MRR-r			0.4	100.00	0.00	155.90
	MRR-r			0.5	99.80	0.42	272.93
	MRR-r			0.6	98.90	1.20	436.26
	MRR-r			0.7	93.20	1.75	655.29
	MRR-r			0.8	75.71	3.77	993.36
	MRR-r	25	2.00	0.1	98.70	0.95	5.86
	MRR-r	25	2.00	0.2	98.90	1.29	6.55
	MRR-r	25	2.00	0.3	95.90	2.56	8.14
	MRR-r	25	2.00	0.4	78.60	3.27	12.31
	MRR-r	25	2.00	0.5	47.10	5.47	18.11
	MRR-r	25	2.00	0.6	22.80	5.39	21.80
	MRR-r	25	2.00	0.7	11.10	4.82	23.48
	MRR-r	25	2.00	0.8	3.60	2.07	24.47
	MRR-r	50		0.1	98.90	0.99	7.33
	MRR-r	50		0.2	92.40	2.50	12.74
	MRR-r	50		0.3	77.50	3.57	20.92
	MRR-r	50		0.4	55.90	4.01	30.52
	MRR-r	50		0.5	37.90	3.00	37.65
	MRR-r	50		0.6	22.80	6.41	43.10
	MRR-r	50		0.7	11.40	2.59	46.59
	MRR-r	50		0.8	4.70	1.77	48.46
	MRR-r	50	1.25	0.1	85.00	2.94	13.21
	MRR-r	50	1.25	0.2	84.40	3.95	13.96
	MRR-r	50	1.25	0.3	81.60	3.03	16.02
	MRR-r	50	1.25	0.4	78.00	5.12	18.83
	MRR-r	50	1.25	0.5	70.20	4.73	24.46
	MRR-r	50	1.25	0.6	42.70	7.47	36.18
	MRR-r	50	1.25	0.7	19.00	4.03	44.40
	MRR-r	50	1.25	0.8	5.30	1.70	48.22
	MRR-r	50	1.50	0.1	94.80	2.10	8.53
	MRR-r	50	1.50	0.2	94.30	2.98	9.29
	MRR-r	50	1.50	0.3	91.50	1.65	11.45
	MRR-r	50	1.50	0.4	88.90	3.63	14.95
	MRR-r	50	1.50	0.5	73.90	6.92	23.65
	MRR-r	50	1.50	0.6	43.30	4.47	36.02
	MRR-r	50	1.50	0.7	16.80	5.55	45.16
	MRR-r	50	1.50	0.8	4.20	1.69	48.70
	MRR-r	50	2.00	0.1	98.60	0.97	6.37
	MRR-r	50	2.00	0.2	99.20	0.92	6.88
	MRR-r	50	2.00	0.3	97.30	1.89	8.90
	MRR-r	50	2.00	0.4	86.90	3.00	16.85
	MRR-r	50	2.00	0.5	59.10	4.01	29.74
	MRR-r	50	2.00	0.6	30.10	6.28	40.72
	MRR-r	50	2.00	0.7	13.80	2.82	45.91
	MRR-r	50	2.00	0.8	4.80	2.53	48.67
	MRR-r	50	2.50	0.1	100.00	0.00	5.55
	MRR-r	50	2.50	0.2	99.80	0.52	6.53
	MRR-r	50	2.50	0.3	96.85	1.79	10.31
	MRR-r	50	2.50	0.4	78.60	3.53	20.84
	MRR-r	50	2.50	0.5	46.00	5.36	34.98
	MRR-r	50	2.50	0.6	26.56	4.77	41.58
	MRR-r	50	2.50	0.7	13.10	2.81	46.10
	MRR-r	50	2.50	0.8	5.10	1.91	48.45
	MRR-r	100		0.1	99.70	0.48	7.35
	MRR-r	100		0.2	94.00	2.00	16.82
	MRR-r	100		0.3	81.20	5.59	32.00
	MRR-r	100		0.4	63.10	3.51	49.90
	MRR-r	100		0.5	43.10	5.84	68.01
	MRR-r	100		0.6	26.90	4.72	81.41
	MRR-r	100		0.7	16.60	3.24	89.18
	MRR-r	100		0.8	8.40	2.27	94.69
	MRR-r	100	1.50	0.1	94.63	2.58	11.39
	MRR-r	100	1.50	0.2	94.63	2.33	12.23
	MRR-r	100	1.50	0.3	93.50	3.32	14.14
	MRR-r	100	1.50	0.4	91.30	3.03	19.18
	MRR-r	100	1.50	0.5	80.45	4.07	34.75
	MRR-r	100	1.50	0.6	54.40	3.62	61.48
	MRR-r	100	1.50	0.7	25.85	5.43	83.03
	MRR-r	100	1.50	0.8	7.70	2.79	95.25
	MRR-r	100	2.00	0.1	99.50	0.83	6.13
	MRR-r	100	2.00	0.2	99.40	0.75	6.87
	MRR-r	100	2.00	0.3	99.05	1.19	9.01
	MRR-r	100	2.00	0.4	92.10	2.86	21.40
	MRR-r	100	2.00	0.5	70.50	6.44	45.19
	MRR-r	100	2.00	0.6	39.85	4.90	71.73
	MRR-r	100	2.00	0.7	18.05	3.75	87.93
	MRR-r	100	2.00	0.8	7.95	3.98	94.83

Figure B.11: Detailed Simulation Results (MRR-r, $N = 2000$)

N	Algorithm	hop-threshold	density-threshold	f	Average Success Rate	stddev(Average Success Rate)	Average Hop Count
2000	MRR-r	100	2.50	0.1	99.85	0.37	5.67
	MRR-r	100	2.50	0.2	99.90	0.31	6.38
	MRR-r	100	2.50	0.3	98.15	1.63	11.71
	MRR-r	100	2.50	0.4	84.80	4.12	30.32
	MRR-r	100	2.50	0.5	58.95	4.11	55.31
	MRR-r	100	2.50	0.6	34.15	4.65	76.27
	MRR-r	100	2.50	0.7	15.35	3.84	89.82
	MRR-r	100	2.50	0.8	7.85	2.01	95.00

Figure B.12: Detailed Simulation Results (MRR-r, $N = 2000$)

N	Algorithm	hop-threshold	density-threshold	f	Average Success Rate	stddev(Average Success Rate)	Average Hop Count
4000	MRR-b			0.1	100.00	0.00	31.04
	MRR-b			0.2	100.00	0.00	125.06
	MRR-b			0.3	100.00	0.00	256.54
	MRR-b			0.4	99.90	0.32	461.27
	MRR-b			0.5	99.80	0.42	664.07
	MRR-b			0.6	97.27	1.79	962.93
	MRR-b			0.7	94.50	3.50	1,259.07
	MRR-b			0.8	76.30	9.86	1,931.95
	MRR-b	50		0.1	87.60	3.86	12.49
	MRR-b	50		0.2	71.20	4.21	20.95
	MRR-b	50		0.3	51.83	6.48	29.81
	MRR-b	50		0.4	35.40	6.15	36.87
	MRR-b	50		0.5	22.65	4.15	42.00
	MRR-b	50		0.6	12.25	3.89	45.67
	MRR-b	50		0.7	7.30	3.16	47.47
	MRR-b	50		0.8	2.80	2.35	48.97
	MRR-b	50	1.25	0.1	84.70	3.62	14.07
	MRR-b	50	1.25	0.2	85.20	4.05	14.24
	MRR-b	50	1.25	0.3	81.80	4.10	15.51
	MRR-b	50	1.25	0.4	81.10	3.00	17.26
	MRR-b	50	1.25	0.5	70.30	4.42	23.69
	MRR-b	50	1.25	0.6	45.90	5.26	35.22
	MRR-b	50	1.25	0.7	17.40	4.58	45.13
	MRR-b	50	1.25	0.8	2.70	1.64	49.07
	MRR-b	50	1.50	0.1	94.10	2.92	8.96
	MRR-b	50	1.50	0.2	93.10	2.64	9.95
	MRR-b	50	1.50	0.3	94.00	2.62	10.23
	MRR-b	50	1.50	0.4	91.00	2.05	12.76
	MRR-b	50	1.50	0.5	73.20	4.44	23.18
	MRR-b	50	1.50	0.6	39.10	4.12	37.43
	MRR-b	50	1.50	0.7	13.40	3.44	45.85
	MRR-b	50	1.50	0.8	3.30	1.16	48.97
	MRR-b	50	2.00	0.1	99.00	1.21	6.38
	MRR-b	50	2.00	0.2	99.30	0.73	6.70
	MRR-b	50	2.00	0.3	98.20	1.42	8.40
	MRR-b	50	2.00	0.4	82.50	4.28	17.68
	MRR-b	50	2.00	0.5	54.30	6.90	30.63
	MRR-b	50	2.00	0.6	25.20	3.22	41.88
	MRR-b	50	2.00	0.7	10.30	3.71	46.70
	MRR-b	50	2.00	0.8	2.40	1.43	49.29
	MRR-b	50	2.50	0.1	99.90	0.32	5.90
	MRR-b	50	2.50	0.2	99.70	0.48	6.55
	MRR-b	50	2.50	0.3	91.90	2.13	12.25
	MRR-b	50	2.50	0.4	65.00	3.89	25.02
	MRR-b	50	2.50	0.5	34.10	5.84	37.77
	MRR-b	50	2.50	0.6	15.30	4.64	44.97
	MRR-b	50	2.50	0.7	7.30	2.45	47.66
	MRR-b	50	2.50	0.8	2.30	1.89	49.28
	MRR-b	100		0.1	89.10	3.57	17.99
	MRR-b	100		0.2	76.00	6.34	33.06
	MRR-b	100		0.3	58.70	4.55	51.96
	MRR-b	100		0.4	39.00	5.16	68.98
	MRR-b	100		0.5	25.60	4.97	81.06
	MRR-b	100		0.6	16.70	2.75	88.83
	MRR-b	100		0.7	9.80	3.46	94.28
	MRR-b	100		0.8	3.20	1.93	97.98

Figure B.13: Detailed Simulation Results (*MRR-b*, $N = 4000$)

N	Algorithm	hop-threshold	density-threshold	f	Average Success Rate	stddev(Average Success Rate)	Average Hop Count
4000	MRR-r			0.1	100.00	0.00	9.90
	MRR-r			0.2	100.00	0.00	48.78
	MRR-r			0.3	99.90	0.32	136.29
	MRR-r			0.4	100.00	0.00	304.28
	MRR-r			0.5	99.80	0.42	576.58
	MRR-r			0.6	97.60	1.26	927.67
	MRR-r			0.7	92.90	1.66	1,358.69
	MRR-r			0.8	76.90	4.86	2,025.85
	MRR-r	50		0.1	98.70	0.95	8.11
	MRR-r	50		0.2	87.90	3.95	15.80
	MRR-r	50		0.3	68.70	5.61	25.33
	MRR-r	50		0.4	51.75	4.90	32.52
	MRR-r	50		0.5	29.05	4.86	40.72
	MRR-r	50		0.6	16.95	3.52	44.77
	MRR-r	50		0.7	7.50	2.42	47.79
	MRR-r	50		0.8	2.85	1.87	49.04
	MRR-r	50		0.9	0.50	0.53	49.84
	MRR-r	50	1.25	0.1	85.40	3.08	13.69
	MRR-r	50	1.25	0.2	83.10	3.45	15.49
	MRR-r	50	1.25	0.3	78.86	6.10	17.95
	MRR-r	50	1.25	0.4	78.60	4.50	19.51
	MRR-r	50	1.25	0.5	67.90	6.81	26.18
	MRR-r	50	1.25	0.6	41.10	9.55	37.57
	MRR-r	50	1.25	0.7	12.50	2.92	46.28
	MRR-r	50	1.25	0.8	2.30	1.89	49.33
	MRR-r	50	1.50	0.1	94.10	1.45	9.13
	MRR-r	50	1.50	0.2	93.00	1.83	10.62
	MRR-r	50	1.50	0.3	93.20	1.55	11.89
	MRR-r	50	1.50	0.4	88.70	4.22	15.49
	MRR-r	50	1.50	0.5	69.60	4.90	26.03
	MRR-r	50	1.50	0.6	39.00	4.42	38.12
	MRR-r	50	1.50	0.7	11.20	4.02	46.76
	MRR-r	50	1.50	0.8	3.70	1.70	48.71
	MRR-r	50	2.00	0.1	99.15	0.93	6.63
	MRR-r	50	2.00	0.2	99.17	1.11	7.63
	MRR-r	50	2.00	0.3	98.10	0.99	9.39
	MRR-r	50	2.00	0.4	85.00	4.27	18.55
	MRR-r	50	2.00	0.5	54.20	5.55	31.75
	MRR-r	50	2.00	0.6	27.00	5.23	41.71
	MRR-r	50	2.00	0.7	9.80	2.94	47.22
	MRR-r	50	2.00	0.8	2.50	1.08	49.17
	MRR-r	50	2.50	0.1	99.80	0.42	6.21
	MRR-r	50	2.50	0.2	99.40	0.97	7.38
	MRR-r	50	2.50	0.3	95.20	2.44	12.03
	MRR-r	50	2.50	0.4	67.30	2.67	26.13
	MRR-r	50	2.50	0.5	41.10	3.00	36.92
	MRR-r	50	2.50	0.6	17.20	3.65	44.83
	MRR-r	50	2.50	0.7	8.30	2.67	47.27
	MRR-r	50	2.50	0.8	3.30	1.64	48.76
	MRR-r	100		0.1	99.10	1.10	9.09
	MRR-r	100		0.2	92.10	3.54	20.18
	MRR-r	100		0.3	77.40	4.74	36.18
	MRR-r	100		0.4	53.80	3.74	57.73
	MRR-r	100		0.5	37.50	5.17	72.62
	MRR-r	100		0.6	21.60	5.85	85.26
	MRR-r	100		0.7	10.40	3.20	92.90
	MRR-r	100		0.8	4.40	1.96	97.28

Figure B.14: Detailed Simulation Results (MRR-r, $N = 4000$)

N	Algorithm	hop-threshold	density-threshold	f	Average Success Rate	stddev(Average Success Rate)	Average Hop Count
8000	MRR-b	50		0.1	87.10	2.92	13.17
	MRR-b	50		0.2	65.10	6.40	24.03
	MRR-b	50		0.3	49.60	5.50	31.02
	MRR-b	50		0.4	30.20	5.14	39.28
	MRR-b	50		0.5	16.80	3.58	43.80
	MRR-b	50		0.6	6.50	2.46	47.61
	MRR-b	50		0.7	2.60	1.71	49.30
	MRR-b	50		0.8	0.80	1.32	49.74
	MRR-b	50	2.00	0.1	99.30	0.82	6.80
	MRR-b	50	2.00	0.2	99.40	0.70	7.19
	MRR-b	50	2.00	0.3	97.20	1.81	9.60
	MRR-b	50	2.00	0.4	78.40	4.58	19.74
	MRR-b	50	2.00	0.5	48.10	2.38	33.26
	MRR-b	50	2.00	0.6	17.90	3.14	44.42
	MRR-b	50	2.00	0.7	6.20	2.70	48.12
	MRR-b	50	2.00	0.8	1.30	1.16	49.65
	MRR-b	50	2.50	0.1	99.80	0.42	6.48
	MRR-b	50	2.50	0.2	99.70	0.48	7.25
	MRR-b	50	2.50	0.3	90.60	2.46	13.33
	MRR-b	50	2.50	0.4	58.80	3.36	27.67
	MRR-b	50	2.50	0.5	28.63	4.64	40.24
	MRR-b	50	2.50	0.6	11.87	4.60	45.87
	MRR-b	50	2.50	0.7	5.17	2.29	48.29
	MRR-b	50	2.50	0.8	1.20	1.00	49.63
	MRR-b	100		0.1	86.80	2.82	20.81
	MRR-b	100		0.2	69.10	4.04	39.52
	MRR-b	100		0.3	46.00	5.94	60.88
	MRR-b	100		0.4	33.00	3.94	73.95
	MRR-b	100		0.5	20.70	4.60	84.01
	MRR-b	100		0.6	10.00	3.74	92.90
	MRR-b	100		0.7	5.60	2.55	96.18
	MRR-b	100		0.8	2.30	1.25	98.42

Figure B.15: Detailed Simulation Results (*MRR-b*, $N = 8000$)

N	Algorithm	hop-threshold	density-threshold	f	Average Success Rate	stddev(Average Success Rate)	Average Hop Count
8000	MRR-r			0.1	100.00	0.00	13.55
	MRR-r			0.2	100.00	0.00	92.23
	MRR-r			0.3	100.00	0.00	300.72
	MRR-r			0.4	99.70	0.67	654.61
	MRR-r			0.5	99.70	0.48	1,175.73
	MRR-r			0.6	98.90	0.88	1,865.61
	MRR-r			0.7	93.50	3.03	2,716.67
	MRR-r			0.8	76.67	4.50	4,001.78
	MRR-r	50		0.1	97.85	1.21	9.89
	MRR-r	50		0.2	85.30	3.30	18.27
	MRR-r	50		0.3	61.10	4.09	28.70
	MRR-r	50		0.4	37.40	6.24	37.63
	MRR-r	50		0.5	22.59	4.73	42.92
	MRR-r	50		0.6	11.54	2.13	46.53
	MRR-r	50		0.7	4.36	2.60	48.66
	MRR-r	50		0.8	1.43	1.22	49.54
	MRR-r	50	2.00	0.1	99.32	0.95	7.02
	MRR-r	50	2.00	0.2	99.80	0.42	7.84
	MRR-r	50	2.00	0.3	96.00	2.71	11.63
	MRR-r	50	2.00	0.4	80.70	3.40	21.05
	MRR-r	50	2.00	0.5	42.90	4.72	35.81
	MRR-r	50	2.00	0.6	18.40	3.27	44.43
	MRR-r	50	2.00	0.7	7.10	1.73	47.91
	MRR-r	50	2.00	0.8	1.90	0.74	49.43
	MRR-r	50	2.50	0.1	99.70	0.67	6.66
	MRR-r	50	2.50	0.2	99.80	0.63	7.70
	MRR-r	50	2.50	0.3	90.10	2.08	15.07
	MRR-r	50	2.50	0.4	64.40	6.96	27.53
	MRR-r	50	2.50	0.5	32.60	6.57	39.74
	MRR-r	50	2.50	0.6	13.60	2.67	46.04
	MRR-r	50	2.50	0.7	5.50	2.17	48.33
	MRR-r	50	2.50	0.8	1.75	1.14	49.50

Figure B.16: Detailed Simulation Results (MRR-r, $N = 8000$)

B.2 Web-of-Trust Prototype Implementation and Experiment Details

Prototype details and experimental setup In our WoT prototype implementation, the *OpenCDK* library [8] is used for all PGP related functions. All experiments were performed on an AMD Athlon 2800 XP system with 2GB of RAM running Linux 2.6. All software libraries and algorithms were written in C/C++ in order to assess a fast and realistic implementation. As SIP proxy, the *VoIP SEAL* prototype [245] was used and extended with the functionality as described in Subsection 6.3.1. SIP messages have been generated with *SIPp* [24] and then been sent to the SIP proxy according to *SIPp* XML-files which specify the performance measurement scenarios we executed.

B.3 Implementations Details of P2PSIP Security Prototype and Detailed Results of Emulation Experiments

B.3.1 Experimental Setup

Our experiments were conducted on either a physical host (AMD Phenom Quadcore, 8GB Ram) running Windows-XP 64-bit or in virtual machines (created by cloning the physical host with Microsoft's *disk2vhd* tool [5]) which were run inside Oracle's Virtual Box [11] with Debian Linux (AMD 64-bit) as host system on a second physical host (AMD Athlon II Quadcore, 16GB Ram). All experiments with $N = 200$ were run on a single machine (physical host or virtual machine); all experiments with $N = 400, 600$ were run on two connected virtual machines.

We executed 100 random lookups each in 10 randomly created P2PSIP networks (i.e. 10 experimental runs each consisting of 100 lookups). For the dynamic *AISNR* experiments, 500 lookups were executed in each run. The average lookup success and hop count results provided below and in Chapter 8 are obtained by averaging the 100-lookup average of each individual run over the 10 executed runs. The standard-deviation of the lookup success refers to the overall average among the 10 runs. In addition, we provide the average over the 10 runs of the standard-deviation of the hop count average obtained in each individual run.

B.3.2 Additional and Detailed Results

Figure B.17 shows the hop count distribution for *DRR* compared to *DRR* combined with *AISNR-s* in a network with 200 nodes and $f = 0.3$. Figure B.18 shows results for the same settings but in a network of 600 nodes.

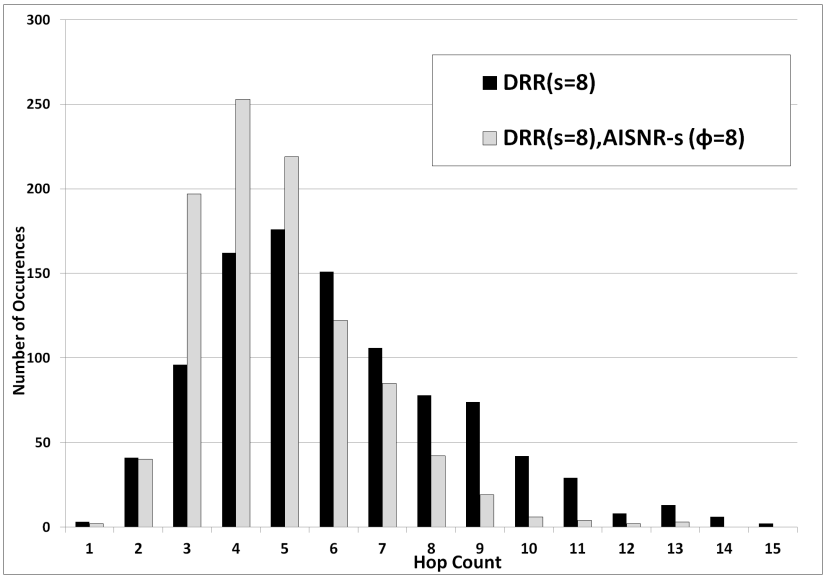

Figure B.17: Hop Count Distribution for *DRR* and *DRR* Combined with *AISNR-s* ($f = 0.3$, $s = 8, N = 200$)

Figures B.19, B.20, B.21, B.22, and B.23 show hop count distributions for our experiments of dynamic *AISNR* (*AISNR-d*) in a network with 200 nodes for different attacker rates f ($s = 8$ and $\phi = 0$). In particular, each figure shows for a given attacker rate f the hop count distribution for the first 100 lookups, lookups 200-300, and lookups 400-500.

Figures B.24 and B.25 provide detailed results of all our experiments. Specifically, for each combination of algorithm and parameter setting, the average lookup success rate and the average hop count we measured in our experiments is given. Additionally, standard-deviations of these averages are provided. Note that also for *AISNR-d*, the hop count results in Figures B.24 and B.25 refer to the average of 100 lookups, i.e. only to the first 100 lookups in the case of *AISNR-d*.

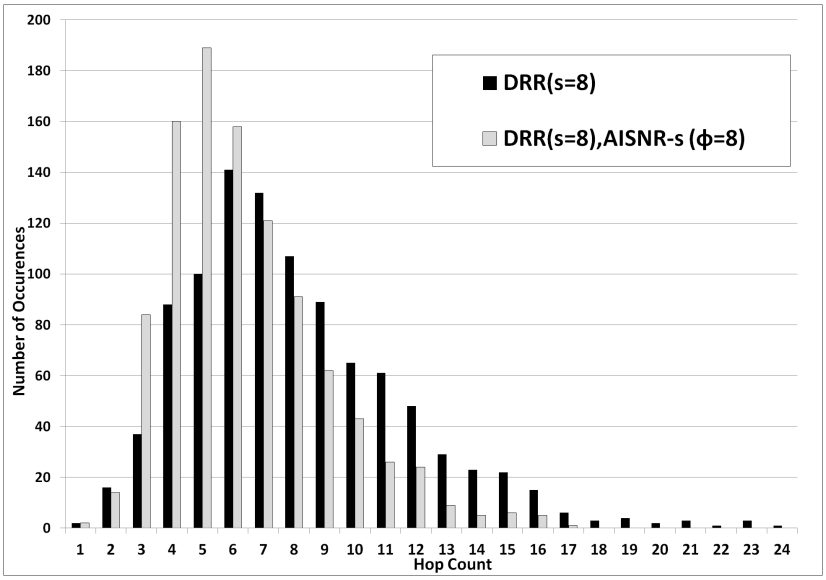

Figure B.18: Hop Count Distribution for *DRR* and *DRR* Combined with *AISNR-s* ($f = 0.3$, $s = 8, N = 600$)

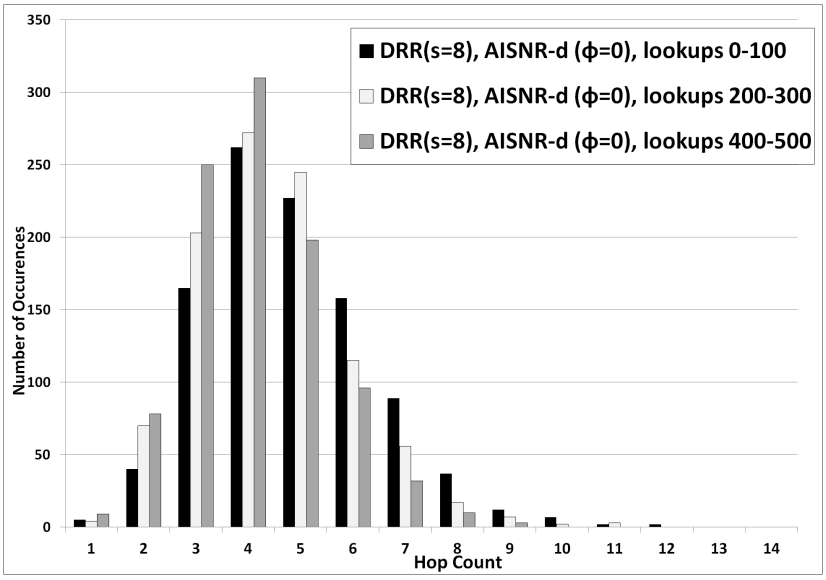

Figure B.19: The Effect of Dynamic *AISNR* on the Hop Count Distribution Over Time ($f = 0.1$, $N = 200$)

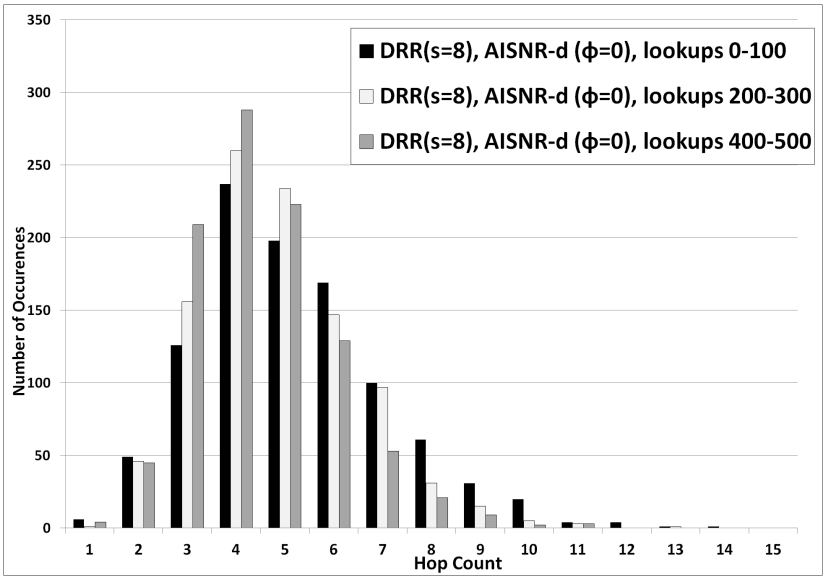

Figure B.20: The Effect of Dynamic *AISNR* on the Hop Count Distribution Over Time ($f = 0.2$, $N = 200$)

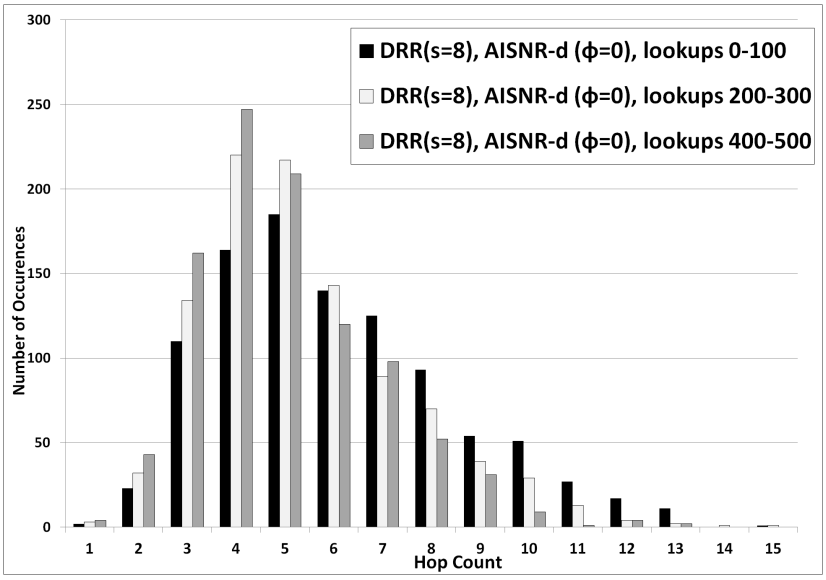

Figure B.21: The Effect of Dynamic *AISNR* on the Hop Count Distribution Over Time ($f = 0.3$, $N = 200$)

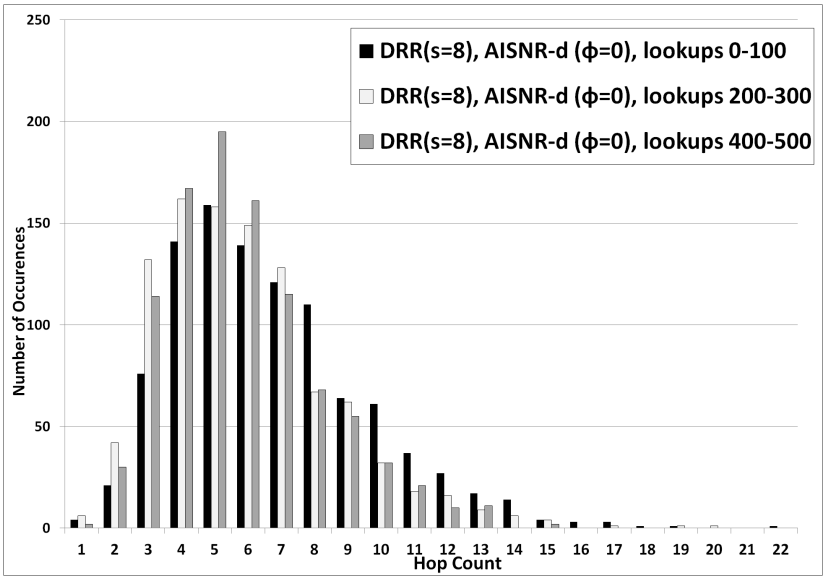

Figure B.22: The Effect of Dynamic *AISNR* on the Hop Count Distribution Over Time ($f = 0.4$, $N = 200$)

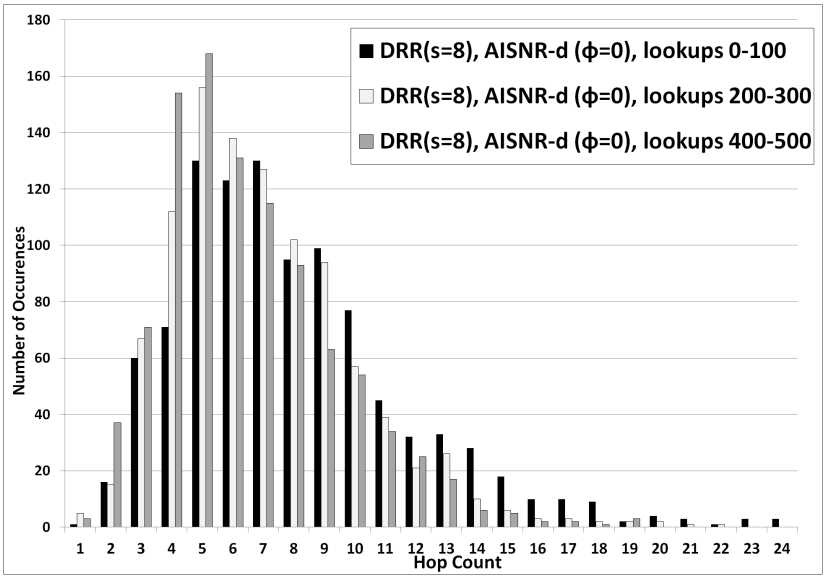

Figure B.23: The Effect of Dynamic *AISNR* on the Hop Count Distribution Over Time ($f = 0.5$, $N = 200$)

N	Algorithm	s	phi	f	Average Lookup Success Rate	stddev(Average Lookup Success Rate)	Average Hop Count	Average(stddev Hop Count)
200	DRR	2		0	100.00	0.00	5.03	1.53
	DRR	2		0.1	98.00	2.11	5.51	1.87
	DRR	2		0.2	96.80	3.61	6.12	2.43
	DRR	2		0.3	88.30	6.68	6.56	2.65
	DRR	2		0.4	79.60	4.86	7.33	3.02
	DRR	2		0.5	59.80	9.82	7.89	3.68
	DRR	4		0	100.00	0.00	4.76	1.34
	DRR	4		0.1	100.00	0.00	5.20	1.74
	DRR	4		0.2	100.00	0.00	5.63	2.09
	DRR	4		0.3	97.20	3.19	6.24	2.36
	DRR	4		0.4	95.70	3.89	7.35	3.11
	DRR	4		0.5	84.80	6.11	8.18	3.61
	DRR	8		0	100.00	0.00	4.27	1.19
	DRR	8		0.1	100.00	0.00	4.86	1.57
	DRR	8		0.2	100.00	0.00	5.37	2.02
	DRR	8		0.3	100.00	0.00	6.16	2.71
	DRR	8		0.4	100.00	0.00	7.07	3.12
	DRR	8		0.5	99.80	0.63	8.37	3.85
	DRR, AISNR-d	8	0	0	100.00	0.00	4.27	1.27
	DRR, AISNR-d	8	0	0.1	100.00	0.00	4.81	1.65
	DRR, AISNR-d	8	0	0.2	100.00	0.00	5.17	1.96
	DRR, AISNR-d	8	0	0.3	100.00	0.00	6.06	2.49
	DRR, AISNR-d	8	0	0.4	100.00	0.00	6.67	2.95
	DRR, AISNR-d	8	0	0.5	99.70	0.95	7.92	3.67
	DRR, AISNR-s	8	4	0	100.00	0.00	4.04	1.16
	DRR, AISNR-s	8	4	0.1	100.00	0.00	4.37	1.41
	DRR, AISNR-s	8	4	0.2	100.00	0.00	4.84	1.76
	DRR, AISNR-s	8	4	0.3	100.00	0.00	5.13	1.84
	DRR, AISNR-s	8	4	0.4	100.00	0.00	5.99	2.53
	DRR, AISNR-s	8	4	0.5	98.20	3.55	6.73	3.23
	DRR, AISNR-s	8	8	0	200.00	0.00	3.74	1.09
	DRR, AISNR-s	8	8	0.1	100.00	0.00	4.09	1.28
	DRR, AISNR-s	8	8	0.2	100.00	0.00	4.43	1.61
	DRR, AISNR-s	8	8	0.3	100.00	0.00	4.78	1.78
	DRR, AISNR-s	8	8	0.4	100.00	0.00	5.21	1.95
	DRR, AISNR-s	8	8	0.5	98.20	3.01	6.01	2.71
	DRR, AISNR-s	8	12	0	100.00	0.00	3.60	1.03
	DRR, AISNR-s	8	12	0.1	100.00	0.00	3.76	1.13
	DRR, AISNR-s	8	12	0.2	100.00	0.00	4.21	1.37
	DRR, AISNR-s	8	12	0.3	100.00	0.00	4.52	1.68
	DRR, AISNR-s	8	12	0.4	98.70	2.75	4.99	1.98
	DRR, AISNR-s	8	12	0.5	100.00	0.00	5.62	2.36

Figure B.24: Detailed P2PSIP Prototype Results ($N = 200$)

N	Algorithm	s	phi	f	Average Lookup Success Rate	stddev(Average Lookup Success Rate)	Average Hop Count	Average(stddev Hop Count)
400	DRR	8		0	100.00	0.00	4.73	1.30
	DRR	8		0.1	100.00	0.00	5.33	1.66
	DRR	8		0.2	100.00	0.00	6.11	2.29
	DRR	8		0.3	100.00	0.00	7.38	3.17
	DRR	8		0.4	99.00	3.16	9.35	4.59
	DRR	8		0.5	98.00	2.75	11.31	5.71
	DRR, AISNR-s	8	8	0	100.00	0.00	4.27	1.18
	DRR, AISNR-s	8	8	0.1	100.00	0.00	4.51	1.39
	DRR, AISNR-s	8	8	0.2	100.00	0.00	4.97	1.80
	DRR, AISNR-s	8	8	0.3	99.60	1.26	5.81	2.26
	DRR, AISNR-s	8	8	0.4	100.00	0.00	6.36	2.77
	DRR, AISNR-s	8	8	0.5	99.00	1.76	7.26	3.31
	Regular Chord			0	100.00	0.00	4.75	1.29
	Regular Chord			0.1	91.10	4.61	5.33	1.73
	Regular Chord			0.2	81.90	4.91	5.91	2.23
	Regular Chord			0.3	70.90	8.14	7.08	3.19
	Regular Chord			0.4	62.80	9.11	8.32	4.06
	Regular Chord			0.5	45.40	6.60	10.86	5.74
600	DRR	8		0	100.00	0.00	4.98	1.27
	DRR	8		0.1	100.00	0.00	5.57	1.83
	DRR	8		0.2	100.00	0.00	6.74	2.58
	DRR	8		0.3	99.80	0.63	8.06	3.59
	DRR	8		0.4	99.20	1.48	9.80	4.52
	DRR	8		0.5	94.40	3.44	12.10	5.82
	DRR, AISNR-s	8	8	0	100.00	0.00	4.46	1.26
	DRR, AISNR-s	8	8	0.1	100.00	0.00	4.91	1.57
	DRR, AISNR-s	8	8	0.2	100.00	0.00	5.57	2.02
	DRR, AISNR-s	8	8	0.3	100.00	0.00	6.33	2.59
	DRR, AISNR-s	8	8	0.4	99.90	0.32	7.04	3.20
	DRR, AISNR-s	8	8	0.5	98.80	1.32	8.42	3.93
	Regular Chord			0	100.00	0.00	4.83	1.36
	Regular Chord			0.1	90.50	4.22	5.52	1.78
	Regular Chord			0.2	78.40	3.69	6.46	2.49
	Regular Chord			0.3	70.60	5.36	7.67	3.45
	Regular Chord			0.4	58.60	7.17	8.98	4.59
	Regular Chord			0.5	45.00	2.58	11.87	6.21

Figure B.25: Detailed P2PSIP Prototype Results ($N = 400, 600$)