P2PSIP over MANETs: Approaches to a Secure Implementation

by

Alexandre Cormier

A thesis submitted to the Faculty of Graduate and Postdoctoral Affairs in partial fulfillment of the requirements for the degree of

Master of Applied Science

in

Electrical and Computer Engineering

Carleton University

Ottawa, Ontario

Abstract

This work examines the security of voice over IP (VoIP) in peer-to-peer (P2P) networks. We do so by focusing on a specific use case, namely that of private (e.g. military) mobile ad hoc networks (MANETs), rather than taking a more general approach to P2P security. This allows for security measures that, while not necessarily applicable to all P2P networks, elegantly solve issues in the given context that more general state-of-the-art solutions cannot solve. We propose two different approaches to implement the P2P version of the Session Initiation Protocol (SIP) in such networks. Both inspired by state-of-the-art approaches, one uses a flooding-based name resolution mechanism while the other instead makes use of a distributed hash table (DHT). We then provide a security solution built on top of both approaches using an offline public key infrastructure (PKI) and redundancy to ensure data integrity and availability. This is similar to one existing solution in particular — the RELOAD standard — but we also tackle issues that affect it notably by being careful with the redundancy scheme. We then present results from performing experiments in a simulator for both of our approaches, demonstrating that they do not suffer from some of the issues affecting other state-of-the-art solutions including RELOAD.

Acknowledgements

First and foremost, I would like to express my gratitude and appreciation to my supervisors, Prof. Babak Esfandiari and Prof. François Gagnon, for their guidance and assistance in all stages of this research. Without you, this thesis would not have been possible.

Special thanks also go to everyone else who was involved in this peer-to-peer SIP (P2PSIP) project: Prof. Thomas Kunz, Dr. Alan Davoust, Ngozi Silas Echegini, Walid Abdel Gelil, Frank Ockenfeld, Rania Darweesh Saleh. Thank you for the feedback, discussions and help throughout the project.

On a personal note, I would like to thank my parents for their love and support throughout my life and the last couple of years, in particular. *Merci maman; merci papa.*

Friends who showed interest in my research, I wish to thank you as well for providing motivation and encouragement for me to bring it to completion.

Finally, I would like to acknowledge organisations that provided financial support: Carleton University, the Ontario Graduate Scholarship program and the US Army. The Army was also important in coming up with the project idea in the first place.

The research was sponsored by the Army Research Laboratory/US Army RDECOM-Americas and was accomplished under Cooperative Agreement Number W911NF-16-1-0345. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory/US Army RDECOM-Americas or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

Contents

List of Tables

List of Figures

Glossary

Omnet++ . . . . . Simulation library and framework, mainly for network simulations.

OverSim . . . . . Peer-to-peer and overlay network simulation framework for *Omnet++*.

Acronyms

(D)TLS . . . . . (Datagram) Transport Layer Security

AODV . . . . . Ad hoc On-Demand Distance Vector

AOR . . . . . Address-of-Record

CA . . . . . certificate authority

CHR . . . . . Cell Hash Routing

DHT . . . . . distributed hash table

DNS . . . . . Domain Name System

DoS . . . . . denial of service

DTLS . . . . . Datagram Transport Layer Security

GPSR . . . . . Greedy Perimeter Stateless Routing

HOLSR . . . . . Hierarchical OLSR

ICE . . . . . Interactive Connectivity Establishment

IETF . . . . . Internet Engineering Task Force

MANET . . . . . mobile ad hoc network

MPR . . . . . multi-point relay

NAT . . . . . Network Address Translation

OLSR . . . . . Optimized Link State Routing

P2P . . . . . peer-to-peer

P2PSIP . . . . . peer-to-peer SIP

PKI . . . . . public key infrastructure

RELOAD . . . . . REsource LOcation And Discovery

SHA-1 . . . . . Secure Hash Algorithm 1

SIP . . . . . Session Initiation Protocol

SMON . . . . . Structured Mesh Overlay Network

TC . . . . . topology control

TLS . . . . . Transport Layer Security

URI . . . . . Uniform Resource Identifier

VoIP . . . . . voice over IP

VRR . . . . . Virtual Ring Routing

Chapter 1

Introduction

Voice over IP (VoIP) is a convenient way to establish a communication channel because it does not rely on costly traditional telephony infrastructure but rather on the now almost ubiquitous Internet infrastructure. There are scenarios, however, where access to such infrastructure may not be available, making communications a challenge despite being crucial in many cases. Examples of such scenarios include post-disaster emergency response and military deployments. In these cases, a mobile ad hoc network (MANET) can be quickly set up and VoIP becomes even more convenient.

Security, in these kinds of scenarios, is also paramount. Communications need to be protected from hostile interference. Emergency responders or soldiers, for example, need to be confident that whomever they called is who they are speaking with.

VoIP sessions are typically established using the Session Initiation Protocol (SIP). For this reason, this thesis focuses on securing a P2PSIP implementation running over a MANET.

This chapter details the contributions of this thesis, lists publications it has lead to and then provides an overview of the structure used for the rest of the thesis.

1.1 Contributions

The main contributions of this thesis are as follows:

- A threat model is defined for P2PSIP in the MANET context. We detail attacks which a malicious actor could employ in an attempt to disrupt or compromise the integrity of the service. This is put into the context of privately owned MANETs.

- We delve into the REsource LOcation And Discovery (RELOAD) protocol [1, 2], standardized by the Internet Engineering Task Force (IETF), and identify security issues that were overlooked by its authors.

- This thesis defines two distinct approaches to P2PSIP over MANETs that are highly based on existing solutions, but with the important difference that they are more secure. One operates at the network layer while the other makes use of a distributed hash table (DHT), and both provide complete data integrity protection, which is not the case for other state-of-the-art approaches. Notably, they are both immune to the security issues identified in RELOAD. Both approaches also provide service availability protection.

- We compare both of our solutions through experimentation and demonstrate their effectiveness. Additionally, we compare them with RELOAD through simulation where possible, showing that our solutions are indeed immune to attacks that do affect RELOAD. While not the main focus, we also compare the performance of the two approaches in terms of network utilization.

1.2 Publications

The work that has been done for this thesis has also lead to two peer-reviewed publications:

- [3] Alan Davoust, François Gagnon, Babak Esfandiari, Thomas Kunz, and Alexandre Cormier. Towards securing peer-to-peer sip in the manet context: Existing work and perspectives. In *Proceedings of the 10th IEEE International Conference on Cyber, Physical and Social Computing (CPSCom)*, pages 223–229. IEEE, 2017

- [4] Alexandre Cormier, François Gagnon, Babak Esfandiari, and Thomas Kunz. Toward testing security attacks and defense mechanisms for p2psip in manets with a simulator. In *Proceedings of the 14th International Joint Conference on e-Business and Telecommunications - Volume 3: DCNET, (ICETE 2017)*, pages 43–54. INSTICC, SciTePress, 2017

Furthermore, an extended version of [4] has been submitted and accepted for publication. It will be published by Springer in the *Communications in Computer and Information Science* series under the title *Approaches to Securing P2PSIP in MANETs*.

1.3 Structure

The remainder of this thesis is organized as follows. Chapter 2 covers some background about the key concepts discussed throughout this thesis: MANETs, SIP and peer-to-peer (P2P). Chapter 3 then establishes the threat model that we are focusing on, formally stating the problem we are attempting to solve by detailing the context in

which our P2PSIP solution for MANETs must operate and the attacks to defend against. Chapter 4 presents the state of the art of the different aspects of this problem by examining different approaches found in literature. Our own security solutions for P2PSIP over MANETs, improving on the state of the art, are then presented in chapter 5. Chapter 6 details experiments performed with these security mechanisms in a simulator and their results. Finally, chapter 7 closes with a summary of our findings and contributions, as well as a discussion of future work.

Chapter 2

Background

The key concepts used throughout this thesis consist of MANETs, SIP and P2P. They are covered in this section, where a short description of each is provided as background.

2.1 Mobile Ad Hoc Networks

MANETs [5] are infrastructure-less wireless networks in which every node acts as both an end device and a router. If a node needs to send a packet to another node that is not within communication range, a multi-hop path is created between those two nodes and the packet is routed to its destination wirelessly through intermediate nodes on this path. An example of a message being routed in such a network is shown in figure 2.1. In this specific transmission, the packet is sent from A to G, which both act as end devices while B, D and E act as routers. The exact way a packet is routed in a MANET, as with any network, depends on the specific routing protocol that is used.

MANETs thus do not require any kind of central administration and allow for dynamic topologies that can be changing constantly. Nodes are devices that can be

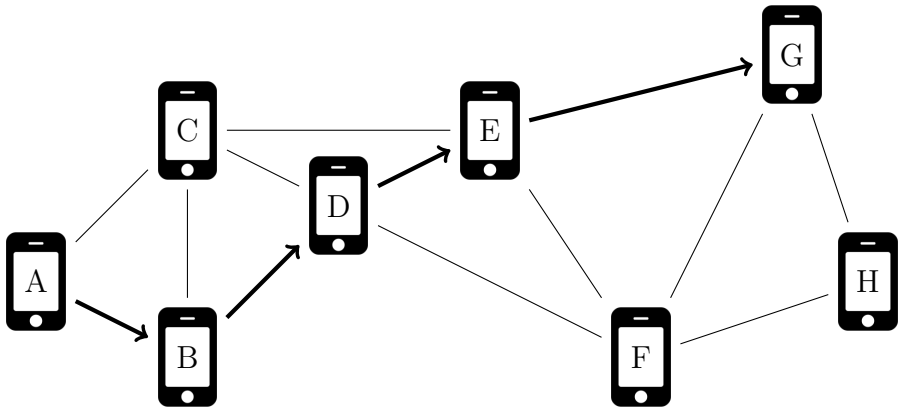


Figure 2.1: Example of Routing in a MANET

carried by their users or even vehicles. These characteristics make MANETs ideal for scenarios where infrastructure may not be available or cannot be relied upon, such as emergency response and military networks. Even in the most hostile of environments, users can carry mobile devices and create a network in order to communicate with each other.

2.1.1 OLSR

Optimized Link State Routing (OLSR) is a proactive routing protocol for MANETs. As the name implies it is an optimization of a classic link state routing protocol specifically for usage in MANETs.

It works by having all nodes periodically advertise their entire one-hop neighborhood to their immediate neighbors. These advertisements are called Hello messages. Using the Hello messages that they receive, nodes learn about their one- and two-hop neighbors.

The key concept in OLSR is that of multi-point relays (MPRs). Each node chooses a set of its immediate neighbors such that it covers routes to all of its two-hop

neighbors and elects them as MPRs. MPRs are responsible for declaring link state information about themselves and nodes that selected them as MPR. This is done through topology control (TC) messages. They are also the only nodes responsible for forwarding TC messages intended for diffusion to the entire network. These TC messages allow nodes to collect information about the topology of the network, to calculate routes and construct their routing tables. Each node keeps links to their neighbors, MPRs and nodes that selected them as MPR.

The advantage of a proactive routing protocol such as OLSR is a low latency. If a node needs to send data to another one and a route exists between them, then a route is already known and data can be sent immediately. The inconvenience is the overhead caused by control packets used to maintain these routes.

2.1.2 AODV

Ad hoc On-Demand Distance Vector (AODV) is an example of a reactive routing protocol for MANETs, as opposed to OLSR being proactive. It works by discovering routes only when needed, rather than maintaining them all the time.

Concretely, when a node needs to send data to another one, it first sends a route request that gets flooded through the network and allows intermediary nodes to set up reverse links towards the source. When the destination receives the route request, it responds with a route reply that travels along the reverse path created by the route request. It also allows intermediary nodes to set up the forward links towards the destination. When the route reply reaches the source node, a bidirectional route has been established between the source and destination and data can be sent along it.

The advantage of a reactive routing protocol like AODV is that it has very low overhead due to few control packets, because routes do not have to be maintained.

The inconvenience is a relatively high latency: unused routes are never kept and a new route has to be discovered before data can be sent between nodes that rarely communicate with one another.

2.2 Session Initiation Protocol

SIP [6] is an IETF-standardized protocol used to initiate a session between two users' devices. The main point of SIP is that this is not done using just IP addresses, but rather, users are identified by a human-readable identifier called a SIP Uniform Resource Identifier (URI). This is similar, both conceptually and in format, to an email address (e.g. sip:alice@example.com).

The heart of the protocol is thus mapping SIP URIs to a contact address, which is the current network location of the SIP client or, in other words, its IP address. This mapping of a SIP URI to a contact address is called an Address-of-Record (AOR). AORs are stored by registrars, whose role is to keep a registry of clients' location and perform the mapping from SIP URI to IP address at session initiation time. SIP being based on a client-server architecture, a number of SIP servers assume the role of registrars. Larger organizations or Internet service providers generally manage those.

Putting all the pieces together, this means that a client that needs to establish a connection with another one needs to first contact its own SIP server, which in turn needs to locate the destination client's SIP server. This is done using the Domain Name System (DNS), based on the domain part of the destination client's SIP URI. This SIP server has the destination client's AOR and can thus contact it to establish the session between the two clients. Figure 2.2 illustrates this mechanism with an example in which a user calls another user with SIP URI sip:bob@example.com. Message 1

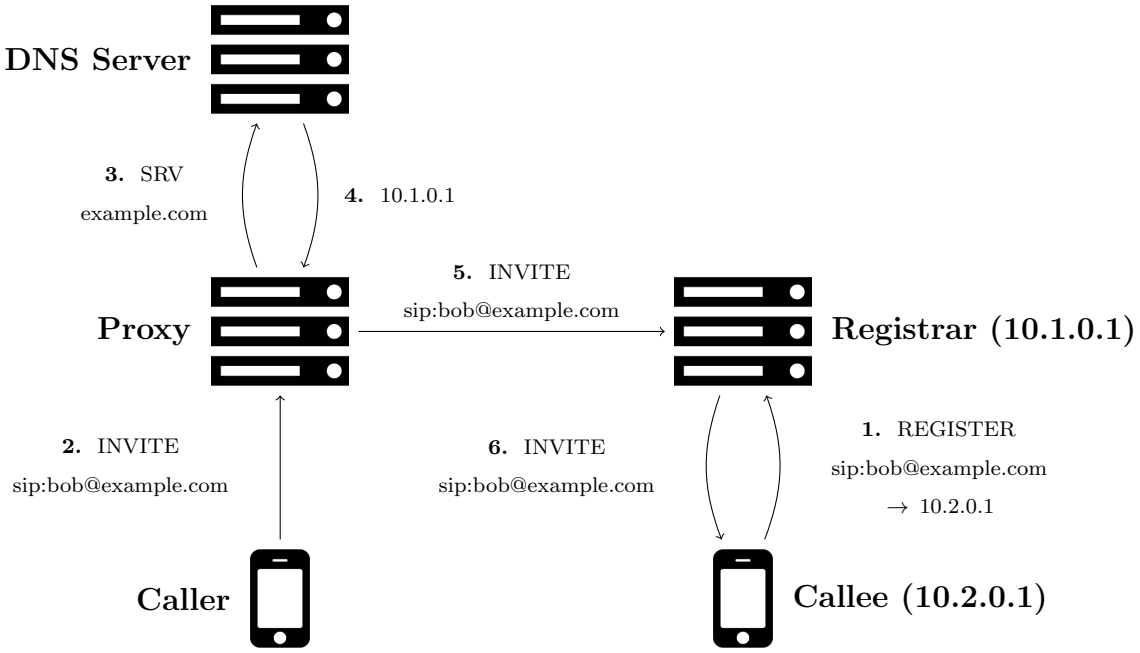


Figure 2.2: High-level Overview of the User Location Mechanism in SIP

shows the callee (Bob), registering their AOR with their SIP registrar. Messages 2, 5 and 6 show the routing of the INVITE message, used to ring the callee's phone, through the caller's SIP proxy and the callee's SIP registrar. Messages 3 and 4 show the DNS query used to locate the callee's SIP registrar, which in turn can locate the callee.

2.3 P2P

As explained in section 2.2, SIP is based on a client-server architecture, which means that it is not appropriate for MANETs. Centralized SIP servers, which serve as registrars, need to be replaced with a distributed solution. One way to do this is to build a P2P overlay network over the MANET, for example using a DHT, to store and retrieve AORs. The RELOAD protocol [1], standardized by the IETF, is such an

approach.

A P2P network [7] is a distributed network in which peers work together by sharing a part of their resources in order to provide a certain service or content. There is no need for central intermediary entities for traffic to pass through, as participants can access each other directly. A *pure* P2P network is one in which peers can leave and rejoin without affecting the service being provided by the network as a whole, which means that it is completely decentralized. P2P is an ideal choice for MANETs because of those characteristics.

A DHT is one way to structure a P2P network. It stores $(key, value)$ pairs and provides peers with an easy and efficient way to retrieve the value associated with a given key as well as to store new pairs. To achieve this, nodes that form the DHT and keys need to share the same identifier space. A common way to assign node identifiers is to compute a hash of the node's IP address. The same hashing function can then be used to compute keys from meaningful names related to the values that need to be stored. The value is then stored on the node closest to the resulting key, for some definition of closeness.

2.3.1 Chord

One of the original DHT protocols, and one of the most well known, is Chord [8]. As suggested above, it assigns an m-bit identifiers to nodes and keys. A Secure Hash Algorithm 1 (SHA-1) hash truncated to m bits is used to generate these identifiers, by hashing the nodes' IP addresses and the keys themselves. m is chosen such that it is large enough to make the probability of collisions negligible. The terms *node* and *key* denote both the node and key themselves as well as their identifier, without ambiguity.

Keys are assigned to nodes using consistent hashing. This entails that nodes are distributed around an identifier circle modulo 2^m. Key k is then assigned to the node whose identifier is equal to that of k, if such a node exists, or to the first node whose identifier follows k's on the circle. Consider the Chord ring in figure 2.3, for example, where filled-in identifiers correspond to nodes and colors indicate which keys each node is responsible for. Keys 0 to 4, as well as 15, are assigned to node 4, for instance. Node 4 is thus called the *successor* of these keys. Similarly, the *predecessor* of a node or key is defined as the first node whose identifier is less than or equal to the identifier of the node or key in question. For example, $predecessor(4)$ would be node 14 in figure 2.3.

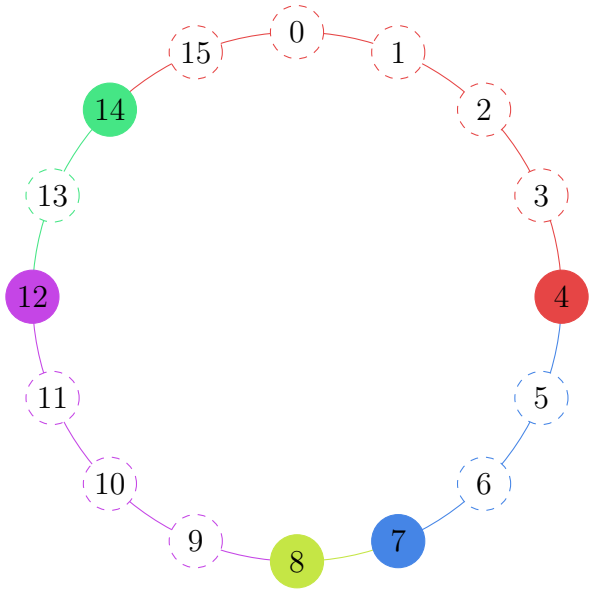


Figure 2.3: Chord Ring for $m = 4$

Each node keeps track of their predecessor and successor. This is used to move data around when a node joins or leaves the DHT. When a node n joins, it inherits responsibility of some keys from its successor: data associated to keys $predecessor(n)$ (exclusive) to n (inclusive) is transferred from $successor(n)$ to n. The opposite happens

when a node leaves: all of the data it was responsible for it transferred to its successor.

To provide efficient queries, each node also maintains what is called a *finger table*. If m is the length of identifiers in bits, as before, this table contains m entries. The (zero-indexed) i^{th} entry in node n's finger table corresponds to $successor(n + 2^i)$. Conceptually, this means that each node keeps track of nodes succeeding it by half the identifier circle, a quarter, an eighth and so on to $\frac{1}{2^m}$ of the ring. This is illustrated in figure 2.4, where arrows point to a node's fingers.

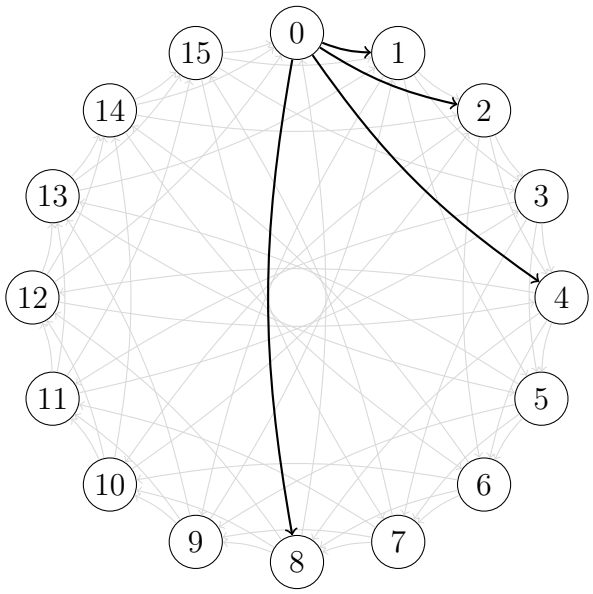


Figure 2.4: Finger Tables in a Fully Populated Chord Ring

To locate key k's successor, node n then contacts its finger whose identifier is the largest while still being smaller than k and asks whose identifier is closest to k. This process is repeated until n contacts a node that knows $successor(k)$. This scheme allows Chord to provide insertions and lookups by contacting only $O(\log N)$ nodes with high probability, where N is the number of nodes in the network. Figure 2.5 illustrates this, with node 0 locating $successor(11)$ and thick arrows indicating the nodes that are contacted.

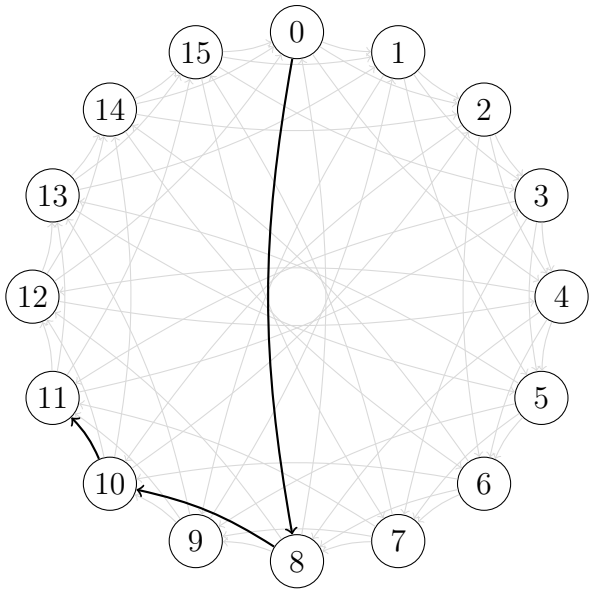


Figure 2.5: Example of Key Location in Chord

2.4 Conclusion

In this chapter, we gave a description of the key concepts that may be used to establish communication channels in an infrastructure-less network: MANETs, SIP and P2P. We also gave examples for each and hinted at how they can all work together. With these concepts explained, we can now move to the main focus of this thesis: the security of a system that puts these pieces together to provide a P2PSIP implementation over a MANET. Chapter 3, coming next, presents the threats that such a network needs to be protected against.

Chapter 3

Threat Model

Before we can discuss attacks to which the network is vulnerable, we need to detail the assumptions that we make with regard to the context in which it is used as well as the security properties that are important in this context. We describe those in section 3.1 and section 3.1.1, respectively. Then, section 3.2 formally defines and extends the notation used in chapter 5 and throughout this thesis. Finally, we present attacks on which the attacker can rely in sections 5.1.1 and 5.2.1.

3.1 Context

This thesis considers a MANET used to run SIP (to initiate VoIP sessions). In such a setting, the network provides a resolution service replacing traditional SIP registrars, mapping users' SIP URIs to their current contact address. It allows nodes to both update their own contact address as well as locate other users' by retrieving their contact address from their SIP URI. The focus of our work is to protect this service.

Furthermore, we are making the following assumptions/observations regarding the context in which this network be used:

- The network is private, in the sense that it is controlled and operated by a given entity (e.g. the army) that can assert who should and who should not be part of it. This is different from the usual P2P contexts which are usually designed to be as open as possible.

- Application level identifiers (SIP URI) can be authorized by the central authority when a node is granted access to the network. We assume that this is done offline, prior to the deployment of the network. Thus, there is no need to worry about ID generation and potential collisions. Those IDs will be human readable (e.g. firstName.LastNameNumber@example.com).

- The registry does not need to be fully persistent. Once a user y leaves the network, it is not meaningful to keep the entry mapping y to its network address up-to-date.

- For simplicity purposes, we assume that each device (each P2P node) is associated with a single user. This is reasonable for personal devices (e.g. smartphone-like gear) but would not be for larger devices (e.g. vehicles). We also assume that every user has a single device.

3.1.1 Security Properties

Several security objectives can be defined for such a service:

- Data integrity: an attacker should not be able to fool a legitimate node with regard to the current location of another node.

- Availability/resilience: an attacker should also not be able to prevent a legitimate node from locating another node.

- Confidentiality: contact addresses retrieved from the service should not be exposed to intermediary nodes. Only the querying node should access the content.

- Anonymity: when a node queries the service to resolve a SIP URI, the node providing the information should not be able to trace back where the query originated from.

In this context, we are interested in protecting data integrity and service availability. A successful attack affecting these two properties also corresponds to the attacker's two objectives, with data integrity being the primary target and service availability the secondary. To achieve their goal, we consider that malicious nodes have the following capabilities:

- They can collude together.

- They can communicate together via a dedicated channel to coordinate their attack.

- Each one controls one legitimate device with one valid SIP URI and the associated certificate. This allows them to impersonate this ID, but should not allow them to impersonate others.

We do not consider confidentiality nor anonymity, but they have been studied in P2P literature. For example, Tarzan [9], MorphMix [10] and Octopus [11] study confidentiality and [12] additionally considers anonymity.

3.2 Notation

The notation used throughout this thesis is that of [4], which is as follows:

- Q stands for a legitimate node querying the SIP service.
- R stands for a legitimate node to communicate with.
- M stands for a malicious node.
- S stands for a generic node.
- x and y stand for users associated with nodes Q and R, respectively.
- z stands a generic users.
- A stands for a contact address.
- $P(x)$ denotes the node(s) responsible for storing x's AOR.

When describing attack scenarios, we assume the following context: node Q is querying the P2P network to get the current location of user y, which is node R's IP address. This query needs to be routed to $P(y)$. For flooding-based SIP resolution, $P(y)$ is simply R itself. Using the DHT-based approach, however, $P(y)$ could be any node, or even multiple nodes if redundant storage is used.

3.3 Generic Attacks and Attack Classes

Some attacks, or classes of attacks, are generic enough that they do not depend on the specific implementation of P2PSIP that is used. These are described here using an adapted version of the classification established by Urdaneta et al. in their work on DHT security [13]. Attacks depending on implementation details will be presented along with the P2PSIP solutions for which they are relevant in chapters 4 and 5.

It is important to note that we focus strictly on attacks targeting P2PSIP and assume that the underlying layers are as reliable as they can be in terms of protecting

availability. For example, radio jamming could be used to attempt to disrupt the network, but is not the subject of this thesis. The same applies for a denial of service (DoS) attack where malicious nodes would drop all packets they receive regardless of their type or content, for instance.

3.3.1 Impersonation Attacks

Attacks in this section do not by themselves compromise the network's operation. However, they allow an attacker to gain more control over the network and to do more damage when performing other attacks that do compromise the normal operation of the network.

3.3.1.1 Sybil Attack

The Sybil attack [14] comes from the inherent openness of P2P systems and consists of one malicious peer being able to act as multiple different logical nodes in the system (i.e. controlling multiple IDs in the DHT). By itself, this kind of attack does not compromise the resilience nor the integrity of the network. However, the ability for a malicious entity to easily control a large number of nodes greatly enhances its ability to perform attacks. For instance, if an entity can setup several malicious nodes in a network, its chances of disrupting the network through a DoS attack are vastly superior than if it controls a single one.

3.3.1.2 Eclipse Attack

An Eclipse attack [15] creates a situation where a malicious entity controls all the nodes "surrounding" its target. This can happen at the network layer, when all nodes physically neighboring the victim are malicious. It can also happen at the DHT layer,

when all of the victim's logical neighbors (e.g. fingers in Chord) are malicious. The target peer thus has a compromised view of the logical network and all of its queries pass through a malicious node which can then manipulate the data as it sees fit. This attack is generally much easier to pull off for a malicious entity if it can easily control multiple identities by the means of a Sybil attack.

3.3.2 Attacks on Data Integrity and Service Availability

This section's attacks, contrary to the those of section 3.3.1, do compromise the operation of the network. They directly target the security properties outlined in section 3.1.1.

3.3.2.1 Storage and Retrieval Attacks

This type of attack consists of a node misbehaving when storing data, AORs in this case, or when asked to provide the data. It could done by providing incorrect information, to target data integrity, or by refusing to serve the data, to target service availability.

3.3.2.2 Routing Attacks

Routing attacks, as the name suggests, target the routing of messages that are used to store and retrieve the information. By itself, such an attack cannot target data integrity, but it can be used to facilitate a storage and retrieval attack by routing towards a node that will perform that attack. It can also target service availability directly, by refusing to route a message.

3.3.2.3 Replay Attack

Another class of attack that is common to any implementation of P2PSIP is the replay attack. It consists in M intercepting a message from a legitimate node and reusing it at a later time when the information in the message is outdated. This is not as powerful as other attacks presented above, but it is more difficult to counter. It can be executed at the different stages of network's operation as well: both when storing and when retrieving data.

3.4 Conclusion

In this chapter, we covered the context in which our P2PSIP implementation operates, established some notation to be used for the rest of this thesis and presented attacks and attack classes to which the network may be subject. This provides us with a clear objective — securing P2PSIP implementation from these attacks in the given context — and the necessary tools to examine existing solutions for a secure MANET used for P2PSIP. The next chapter does just that, by presenting the main state-of-the-art approaches to attain this objective.

Chapter 4

Related Work

This chapter covers the state of the art of research related to all aspects of securely implementing P2PSIP over MANETs. This obviously includes existing proposals for P2PSIP over MANETs (section 4.1) but also, because some solutions make use of a DHT, works on implementing a DHT over a MANET (section 4.2) and research on securing DHTs (section 4.3). Finally, we also include a review of an important standard for P2PSIP over the Internet (section 4.4).

4.1 P2PSIP over MANETs

As discussed in chapter 2, the main problem to solve when implementing SIP over a MANET is the handling of the registration service in the absence of conventional SIP servers. In other words, a scheme must be devised for the storage and retrieval of AORs in a distributed system. Stuedi's analysis of the problem [16] shows that solutions to store information in such a system range from local information storage and lookups based on broadcast to full replication and local retrieval. For a network of N nodes, the former provides $O(1)$ insertions and $O(N)$ lookups, while the latter

has $O(N)$ insertions but $O(1)$ lookups. A DHT lands in the middle of the scale and usually provides insertions and lookups both in $O(\log N)$ *logical* hops.

We can use this range to categorize existing P2PSIP solutions for MANETs according to their registration scheme as follows.

4.1.1 Local Storage

TacMAN [17] and a proposal by Banerjee et al. identified as "Loosely Coupled" [18] use local storage, meaning that each peer stores only its own information. A remote peer that needs to retrieve this information then broadcasts a request to locate it using the underlying routing protocol. TacMAN uses Hierarchical OLSR (HOLSR) [19] while the solution by Banerjee et al. uses AODV.

4.1.2 Network Subset Storage

Some solutions instead stay closer to traditional SIP and select a subset of the network's peers to act as registrars. This is the case of AdSIP [20], another approach by Banerjee et al. identified as "Tightly Coupled" [21], MANETSip [22] and a solution by Aburumman et al. [23]. In all of these, a subset of the network's nodes are selected as registrars. In the first two, in particular, the registrar nodes form a dominating set of the network graph.

In all of these solutions, nodes register their AOR only with the closest registrar(s). Registrars in [21] and [23] are heads of clusters and each node registers with their own cluster's head. In AdSIP, on the other hand, nodes register with all their neighbors which are registrars. In [23], one backup server node per cluster also copies data from the cluster's head in order to provide load balancing and help with hand-off when nodes leave.

4.1.3 DHT Storage

Four unnamed proposals, by Wongsaardsakul [24], O'Driscoll et al. [25], Bryan et al. [26] and Seedorf [27], as well as P2PNS [28] implement a DHT over the MANET. This DHT is used to store AORs in order to replace registrars.

References [27] and [26] are actually not designed specifically for MANETs, but could be used over one. The first simply implements Chord to store AORs. An interesting aspect is a focus on security, with self-certifying SIP URIs to ensure the integrity of registrations. To achieve this, each node generates a key pair for itself. The user part of the SIP URI is then replaced with a hash of the node's public key, and messages can be signed to provide authentication and integrity. However, this means that the SIP URI cannot be, as is often desired, a meaningful name like the user's first and last names. Reference [26], on the other hand, ultimately lead to the RELOAD protocol, standardized by the IETF, which we cover in section 4.4.

P2PNS also simply uses a very typical DHT, Kademlia [29], to store and retrieve AORs. It puts a focus on security though, by assigning certificates to all nodes and signing messages. Certificates are self signed, but to prevent a Sybil attack, it relies on crypto-puzzles. If this is not enough, [28] also mentions the possibility of using a central certificate authority (CA). As far as we can tell, P2PNS does not protect against replay attacks at all, however.

Wongsaardsakul's solution uses a DHT-like overlay, although without key-based routing. It is a cross-layer design called Structured Mesh Overlay Network (SMON) and is based on the earlier CrossROAD [30]. As opposed to a more traditional DHT, SMON does not have any particular structured topology. Instead, nodes have full knowledge of each other and their identifiers, which they obtain from the underlying MANET routing protocol (a modified OLSR). Outside of P2PSIP literature, a similar

DHT implementation was proposed called OneHopOverlay4MANET [31].

The proposal by O'Driscoll et al. is based on a hierarchical DHT with a Chord backbone. Nodes form clusters, and the head of each cluster joins the Chord DHT. Each cluster can be a structured or unstructured network. This solution however has no implementation or evaluation, and it is unclear how routing should work between clusters.

As mentioned previously, the RELOAD protocol [1] also uses the DHT storage approach, although it is primarily designed for the Internet rather than MANETs. We cover it independently in section 4.4.

4.1.4 SIPHoc

SIPHoc [32] is in a category of its own. It supports both local storage and full replication, as well as solutions in-between, depending on the routing protocol that is used. Nodes register their AOR with their local registrar and then exchange information with each other by piggy-backing on the routing protocol. If a proactive routing protocol is used, like OLSR, AORs are exchanged between registrars proactively. On the contrary, if an on-demand routing protocol like AODV is used, then storage stays local and AORs are only retrieved on demand (and can then be cached).

4.2 DHT over MANET

Focusing on a DHT is the norm for P2PSIP, but it is interesting to notice that among proposals specifically designed for the MANET context, very few make use of the full DHT model. Even in Wongsaardsakul's proposal, SMON is unstructured and nodes need full knowledge of the network topology, despite using a DHT-like scheme.

Outside the specific context of P2PSIP over MANETs literature, however, solutions for efficient implementations of DHTs over MANETs have been presented. These could be used to replace SIP registrars and directly implement P2PSIP. We present them briefly here.

4.2.1 MADPastry

MADPastry [33] is a variant of the Pastry DHT [34] specifically designed for MANETs, by integrating it with the AODV routing protocol to create a topology-aware overlay.

A concept known as *landmarking* is used in early work on topology-aware DHTs [35]. A few well-known reference servers, the *landmarks*, in different geographic locations are used to evaluate the relative locations of peers: nodes with similar round-trip time measurements to the landmarks are geographically close. This geographic clustering, reflecting the network topology, is then used to establish an overlay adapted to the physical location of the peers.

MADPastry takes this a step further and uses *random landmarking* [36]: a small number of well-known keys in the identifier space are selected and the nodes responsible for those keys act as landmarks. This allows a periodic reconfiguration of the overlay to maintain a logical DHT topology that's always adapted to the underlying physical topology.

4.2.2 VRR

Virtual Ring Routing (VRR) [37] is very different from most DHT protocols in that it is does not rely on another routing protocol but rather it *is* a routing protocol. It is implemented directly on top of the link layer and provides both point-to-point routing and DHT functionality. Nodes are given random identifiers which are used for routing

(instead of IP addresses) and DHT-level keys are assigned in the same identifier space.

As the name indicates, nodes form a virtual ring similar to Chord, but they don't maintain a finger table. Instead, each node keeps track of a small number r of logical neighbors in the virtual ring and all of its physical neighbors. It maintains multi-path routes to its logical neighbors, half of which are successors and half being predecessors.

To facilitate routing, each node also keeps track, in its routing table, of the paths between logical neighbors that pass through it. The number of such extra paths is proportional to the average path length and could potentially be large ($O(r.\sqrt{n})$ is suggested).

4.2.3 CHR

Cell Hash Routing (CHR) [38] is a proposal for a DHT protocol for MANETs based on geographic clustering and routing. It creates a DHT of clusters rather than individual nodes, where each cluster consists of all nodes located in a given *cell*. A cell is a section of the geographic space in which the MANET operates and all of them have the same size. If a cell is empty, the nearest non-empty cell takes responsibility for its keys.

To make this scheme efficient, CHR also uses an adapted version of the Greedy Perimeter Stateless Routing (GPSR) protocol, which provides geographic routing.

4.3 DHT Security

In this section, existing approaches for protecting from the attacks described previously are examined. Notes on how well each of them could be applied to the context of a DHT over a MANET are also covered.

4.3.1 Securing the Infrastructure of the DHT

We first cover solutions and mitigations found in the literature for the Sybil and Eclipse attacks, in particular.

4.3.1.1 Public Key Infrastructure

Castro et al. [39] argue, just like Douceur does [40], that the only way to prevent a Sybil attack is to have a central, trusted CA issue and sign identities. They suggested having a set of trusted authorities that issue certificates comprised of:

- the node's identifier, which should be assigned at random;
- the node's IP address;
- the node's public key.

The IP address is included in order to prevent an attacker that has gotten a hold of a valid certificate from using it with multiple nodes it controls, as well as to allow some optimizations.

Each node has its own certificate and private key, as well as the CA's public key. Nodes can thus sign all their messages with their private key and send their certificate along with it. The recipient can then verify the signature with the public key found in the certificate, as well as verify the certificate with the CA's public key. This allows them to ensure that the received message comes from a trusted node. If a node starts misbehaving and the anomaly is detected, a mechanism to revoke its certificate can be put in place.

The authors suggest preventing an attacker from creating multiple identities by making it hard to acquire a certificate from the CA, for example by making them expensive either money- or computationally-wise.

This approach does not help protect from the Eclipse attack specifically, although preventing the Sybil attack does remove one facilitating vector in the orchestration of the Eclipse attack.

Limitations This approach works well when the node identifiers are fixed for any and all devices, but cannot work otherwise. Same goes for the IP address; it needs to be fixed for each device. Furthermore, having a central CA that every node can go through before joining the DHT is unrealistic in some scenarios. Finally, this approach does not help against a money- or computationally-rich attacker.

Applicability to MANETs The biggest problem with this approach in the context of a MANET is that it assumes a fixed node identifier and IP address for all nodes, because it is included in the certificates. In a MANET, some routing and DHT protocols may break this assumption to get better performance even with mobility.

Another potential issue is the need for a central CA to issue certificates. It might not be available to devices in the MANET. This limitation can be worked around in some contexts, like the one described in section 3.1, by having all certificates issued by an *offline* CA before deploying the network.

Finally, certificate revocation can also be a challenge. In its most naive form, this process needs a central service to be reachable by all nodes at all times, which is not realistic for a MANET. More appropriate approaches to this problem in the context of a P2P system are however found in literature [41, 42, 43].

4.3.1.2 Node IDs Based on the IP Address

An approach to preventing attacks on the infrastructure of a DHT that has been discussed multiple times [44, 45] is to derive the identifier from the node's IP address.

This typically means that the node ID is a hash of the IP address. As mentioned in section 2.3.1, this idea is even core to the popular DHT protocol Chord [8].

If nodes validate other nodes' identifiers before communicating with them, an attacker can only create as many identities in the DHT as IP addresses it controls. It also has no control on the identifier that gets assigned and thus cannot intentionally occupy an advantageous location in the DHT.

Limitations This approach works under the assumption that an attacker cannot choose its IP address and cannot control many IP addresses.

If an attacker could control its IP address, then it could also have some control over its node ID, thus potentially its logical location in the DHT. For example, with Chord, a malicious peer could try and get an IP address that hashes to an identifier that makes it its victim's finger. This would allow it to control some of the victim's requests.

If an attacker had control over many IP addresses (possibly choosing them as well), then it could gain control over a large part of the DHT, whether through a Sybil attack or just using multiple physical devices. It could perform an Eclipse attack by choosing its IP addresses carefully from the range of addresses it controls.

Also, if the nodes' IP addresses change often, then their identifiers will change just as often. This means that extra work needs to be done to stabilize the DHT, just as if the node had left and re-joined the network.

Applicability to MANETs Because devices in a MANET are mobile by definition, this approach could incur a performance overhead if a node's IP address depends on its location in the network topology. This is not generally the case, but if it is, stabilization of the DHT has to be performed every time a node moves and its IP

address changes.

Additionally, depending on how the MANET is managed, some nodes may have the capability to choose their own IP address. As described above, this would mean that this approach would not offer as much protection as it should in this context.

If none of these apply to the MANET, however, this approach is very sensible. This is especially conceivable in a privately controlled network.

4.3.1.3 Node IDs Based on the Location

Another approach that has been studied is to base the node identifier on the physical location of the node [46, 47, 48]. This approach typically uses hop count or round-trip time measurements to designated nodes as well as other information on the network topology as a base for the node ID. It aims to prevent a Sybil attack by only assigning a single or a fixed small number of node IDs per location. This prevents a single device from getting multiple IDs.

Limitations Similar to the IP-based node IDs presented in section 4.3.1.2, it is assumed that the characteristics on which the node ID is based, in this case the location of the node, are relatively stable. Otherwise, nodes' identifiers change frequently and the DHT needs to be stabilized very often, incurring a performance cost.

Schemes based on network topology also assume that some designated nodes can be trusted and strategically placed, in order to properly locate the nodes within this topology.

Applicability to MANETs The limitations presented previously are especially apparent in the context of a MANET. Because devices are mobile, nodes move

physically and the network topology is constantly changing. Hence, node IDs are constantly changing, adding overhead performance-wise.

In a MANET scenario, it also cannot be assumed that trusted devices can be placed in strategic locations, as is required by some of the implementations of this approach.

4.3.1.4 Persea

Persea [49] is a recent proposal by Al-Ameen et al. for a social DHT protocol. It deals with Sybil attacks in a unique way: when joining the network, a node takes responsibility for a chunk of the identifier space that belonged to its bootstrap node. The chunk assignment is then certified and the certificate is stored in the DHT itself. This can be visualized as a tree, called the bootstrap tree, where a node's parent is its bootstrap node.

This scheme ensures that an attacker that gets into the network cannot invite other peers, whether physical or sybils, to gain more control over the DHT. In fact, this attacker would only divide the ID space that is already under its control.

Limitations This approach works well under two important conditions:

- the first x nodes to join the network are honest, for a large enough value of x, and distributed uniformly in the bootstrap tree;
- it is hard for an attacker to join the network through an honest node.

Indeed, if an attacker is near the root of the bootstrap tree, because it joined the network early enough or the bootstrap tree is unbalanced, it gains control over a big chunk of the ID space regardless of its (in)ability to create sybil identities.

Furthermore, if it is easy for an attacker to join the network, then it can simply join multiple times, from different bootstrap nodes. In such a case, it could even control an enormous portion of the ID space by taking all the available chunks.

Another limitation is the fact that this proposition does not include a strategy to eliminate malicious nodes from the network once detected and there is no obvious way to do so. This means that if an attacker has gained enough control to be bothersome, it could be very hard to get it out.

Applicability to MANETs Persea is not designed specifically for MANETs, so while it can be implemented over a MANET, it is not necessarily optimally efficient.

The second assumption listed above can also be a problem. Namely, making it hard for an attacker – but easy for honest nodes – to join the network could be a challenge. This can however be dealt with using other solutions, such as those presented in section 4.3.1.1.

4.3.2 Protecting the Integrity of the DHT

This section covers existing approaches found in literature to prevent or mitigate attacks on the integrity of the DHT, such as storage and retrieval attacks as well as routing attacks.

4.3.2.1 Octopus

Octopus [11] is a DHT that aims to provide a secure and anonymous lookup mechanism. It achieves this by issuing a valid certificate to all joining peers and revoking certificates for nodes that are found to be malicious or otherwise misbehaving.

The detection of malicious peers works as follow: each node maintains a predecessor

list and a successor list and periodically asks its predecessors for their successor lists to make sure that it is present in it. If the node discovers that it, itself, is not present in its predecessor's successor list, then it is implied that this predecessor manipulates its successor list and is malicious or misbehaving. To prevent the node under test from responding with a correct successor list while actually using an incorrect one for the usual routing operations, onion routing [50] is used to mask the initiator of the test request. A similar mechanism is also used to detect finger table manipulation.

It is also of note that for this to be effective, Octopus needs to query the entire routing table of a node on lookup, so that test requests are not any different than a normal lookup. This also has benefits from an anonymity perspective: nodes on the lookup path do not know the destination of the lookup.

Limitations Octopus incurs a bandwidth overhead to provide its security and anonymity benefits. Extra bandwidth is used for integrity checks as well as requesting nodes' entire routing tables on lookup despite really only needing one entry. It remains reasonable though according to the authors' simulations, at a few kilobits per second.

Applicability to MANETs The most serious limitation of Octopus for use over a MANET is the need for a central CA to be available to any joining peer. In the specific context described in section 3.1 though, static offline CA could be used.

But Octopus also requires the CA to be available to any peer willing to report a malicious node or to query the revocation status of a node's certificate. For these situations, distributed approaches can be found in literature [41, 42, 43], as mentioned in section 4.3.1.1.

Finally, Octopus is not designed with MANETs in mind, meaning performance in this scenario may not be optimal. The bandwidth overhead may have a more

noticeable impact on the performance of the DHT when used over a MANET.

4.3.2.2 Redundant Storage

Redundant storage is a recurring approach to mitigate attacks against the integrity of a DHT. It is the idea of storing data in multiple nodes and assuming that, upon retrieval, the majority of those nodes will be honest and correct.

It can be divided in two categories: data replication and erasure coding.

The most common and preferred approach is data replication [13]. It is simple to implement, as it consists of storing copies of the data in multiple nodes and retrieving it from all the same nodes, using a majority voting system to determine the correct value. Replicas can be stored either close to each other (used in Kademlia [29], Pastry [34], Myrmic [51] and others [39, 52, 53, 54]) or spread – evenly or randomly – over the ID space (Tapestry [55], Persea [49] and others [56]). Some approaches also combine both techniques [57, 58].

In an extreme case, data could be stored on every node. Then when retrieving a value from the DHT, every node would be queried for it. The majority voting mechanism means that this would be fully effective in scenarios with up to half of the nodes being malicious, as long as they only perform pure storage and retrieval attacks as opposed to routing attacks. Of course, such an extreme is impractical, as full replication can be implemented much more simply and efficiently without a DHT.

The second approach, erasure coding, consists in dividing the data in multiple parts, encoding each part along with some redundant data pieces and storing each part on a different node. This allows the node performing a lookup to decode and re-assemble the data even when one or a few parts cannot be retrieved or are incorrect. It provides storage savings as well as theoretical bandwidth savings when compared

to replicated data. This technique was used by a few researchers [53, 59, 60], but has been mostly abandoned for reasons covered below.

Redundant storage mainly protects against storage and retrieval attacks, where the node responsible for the data is malicious and tampers with it. It can also indirectly help against routing attacks — where malicious nodes can prevent the request from reaching the node that is responsible for the data — by creating different paths to the different nodes storing the data.

Limitations The obvious limitations to data replication are the increased storage space and bandwidth. This is exactly what erasure coding aims to fix, at the cost of a more complex system. In practice, however, it has been shown to not provide any improvement over data replication for bandwidth, because maintaining the appropriate redundancy levels incurs an increase in bandwidth [61]. Furthermore, erasure coding can only be effective with large enough data chunks.

Applicability to MANETs Data replication can be applied to MANETs, but the extra bandwidth costs may become apparent or troublesome. This is also true for erasure coding, with the additional limitations detailed above.

4.3.2.3 Redundant Routing

Redundant routing is the idea of taking multiple paths to reach a node. This allows to reliably locate nodes responsible for a given key. For example, it can prevent a single malicious node from making a request fail because it is on the single path to the destination node.

As with redundant storage, this can be divided in two techniques: multiple paths [39, 62, 56] and wide paths [52, 53, 54, 51, 29]. Some researchers have also

combined both approaches and used multiple wide paths. [57, 58]

Multiple paths are just what they sound like: several independent paths between the same two nodes. Wide paths, on the other hand, consists in finding multiple physical paths for each step in a logical path.

Limitations As with redundant storage, the obvious limitation of redundant routing is bandwidth. Multiple paths and wide paths generate more traffic than a single (not wide) path.

Applicability to MANETs Again, the bandwidth overhead may be noticeable when redundant routing is implemented in a MANET context. This appears to be even more of a problem than with previous approaches, however, because a single logical path at the DHT level does not necessarily map one for one to the network layer. While a DHT provides routing in $O(\log n)$ logical hops, the path is usually longer in physical hops.

4.4 RELOAD

While DHT security only relates to P2PSIP security indirectly, an example of a DHT-based protocol that can be used for P2PSIP with security considerations is RELOAD [1]. It is a signaling protocol for use on the Internet. It was designed mainly to support a P2PSIP implementation, but was generalized to also support other applications with similar requirements. This entails that RELOAD allows one to locate other members of the RELOAD network and establish a connection for an arbitrary application with them. The P2PSIP implementation built atop RELOAD is defined in RFC 7904 [2]. While not designed specifically for MANETs, it can be used

with one.

We place a special emphasis on RELOAD for two reasons. First, it includes a full *standardized* specification for P2PSIP. Being recognized by the IETF in that way gives it a certain importance, having gone through the rigorous standardization process. Second, the standard includes an extensive security model to account for potential security in the presence of malicious actors. This corresponds very closely to the topic of this thesis: a *secure* P2PSIP implementation, that can be used with MANETs.

This section describes the high-level functioning of both in section 4.4.1 with a focus on its security in section 4.4.2, all within the context of P2PSIP.

4.4.1 Functioning of P2PSIP over RELOAD

The core of RELOAD is a P2P overlay. The standard is topology-independent and provides a way to define topology plugins, which implement different P2P overlay topologies and can easily be interchanged. Only one such plugin is defined in the RELOAD standard: a modified version of Chord [8]. Rather than using the simple GET and PUT operations of a DHT, however, RELOAD defines its own message types and operations to be routed by the overlay using key-based routing.

RELOAD offers a generic storage service, defining messages for storage requests and answers as well as retrieval requests and answers. These can be used to store any *kind* of data. Each data kind has an associated *type* among SINGLE, ARRAY and DICTIONARY. It also offers a signaling mechanism using so-called AppAttach messages, which contain an application ID. Those are requests to establish a direct connection for the given application.

Included in the RELOAD specification is also the usage of the Interactive Connectivity Establishment (ICE) protocol for Network Address Translation (NAT) traversal.

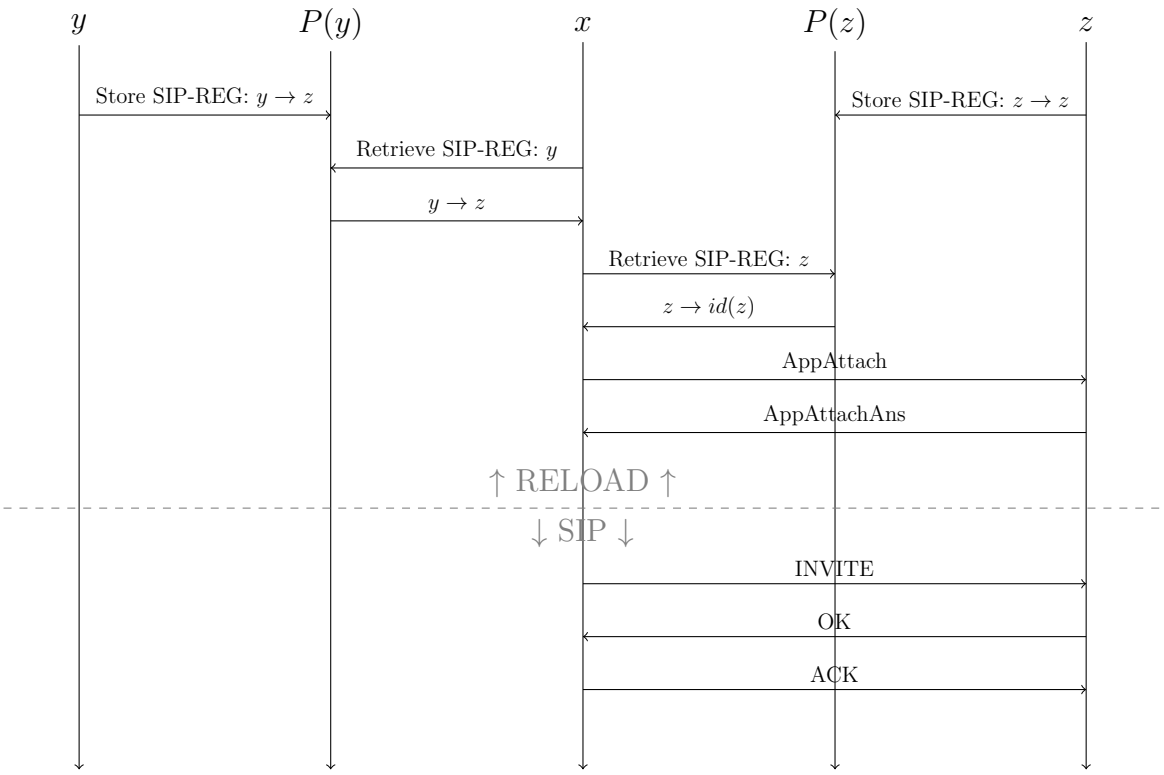


Figure 4.1: P2PSIP Connection Establishment with RELOAD with Call Forwarding

This is irrelevant for MANETs, however, so we ignore it for the purpose of simplicity.

To use P2PSIP over RELOAD, the SIP-REGISTRATION data kind is defined. It is of type DICTIONARY and a resource of this kind contains mappings from the user's name to AORs. An AOR, in this context, is defined as either the user's node identifier or another user's name, to whom calls should be forwarded. Application IDs are also defined for unencrypted and encrypted SIP. Figure 4.1 illustrates the establishment of a SIP connection using this scheme in a situation where user x calls y, who is forwarding their calls to z.

4.4.2 Security of P2PSIP over RELOAD

RELOAD features a three-level security model based around public key certificates, which are typically assigned by a central authority but could also be self-signed if used in a closed network. For this document, the focus is put on the first approach, as it is the most secure. In this scenario, the central authority is also responsible for assigning node identifiers, which are certified along with the users' names. The certificates are then stored in the RELOAD overlay, using the pre-defined CERTIFICATE_BY_NODE and CERTIFICATE_BY_NAME data kinds, to avoid nodes being responsible of their own data.

The three levels at which communications security is enforced in RELOAD are defined as the *connection*, *message* and *object* levels. Concretely, this means that all communications between RELOAD nodes are performed over Transport Layer Security (TLS) or Datagram Transport Layer Security (DTLS) and that both RELOAD messages and stored objects are signed by the node creating them.

On top of this, RELOAD provides an access control mechanism taking advantage of object signatures to authorize (or reject) operations. An admission control feature based on a pre-shared secret is also defined as optional, mostly to be used along with a self-signed password to prevent anyone from joining the network.

4.4.2.1 Connection Security

All communications between RELOAD nodes are performed over (D)TLS, with both client and server authentication. Thanks to this, only nodes with a valid certificate, thus approved by the central authority, can even join the network. This eliminates the possibility of a Sybil attack.

This also allows that they are communicating exactly with whom they intended.

In the case of a P2PSIP session establishment for example, this allows the caller to validate that they have reached the actual user that they called or the correct other user, in case the call was forwarded. The callee can also verify the caller's identity.

4.4.2.2 Signature-based Access Control

RELOAD defines an access control scheme, built upon its public key infrastructure (PKI), to control authorization of various operations. Each data kind has an access control policy associated to it, among the four defined by RELOAD: USER-MATCH, NODE-MATCH, USER-NODE-MATCH and NODE-MULTIPLE. When a storage request is received, it must be honored if and only if the access control validation according to the policy of the data kind being stored succeeds.

USER-MATCH means that the Resource-ID of the resource being stored needs to be a hash of the user name found in the certificate of the signer of the resource. Similarly, a NODE-MATCH policy implies that the Resource-ID must be a hash of the certificate's node identifier. USER-NODE-MATCH is essentially a USER-MATCH that can only be applied to dictionary types, with the additional constraint that the key in the dictionary needs to be the certificate's node identifier. Finally, NODE-MULTIPLE is similar to NODE-MATCH, but allows a small integer to be concatenated to the certificate's node identifier before it is hashed and compared to the Resource-ID.

In the case of P2PSIP, the SIP-REGISTRATION data kind is given the USER-NODE-MATCH policy. This means that nodes can only store mappings for themselves, mapping only their own node identifier to an AOR.

In addition to access control policies, RELOAD includes a timestamp in all stored resources. A node receiving a storage request for a given resource must first validate

that the timestamp in the incoming resource is more recent than the one on the existing stored resource for this Resource-ID. If this validation fails, it must reject the storage request to prevent replay attacks.

4.4.2.3 Resilience

The RELOAD specification suggests storing replicas of all stored data and defines some rules related to access control as to how these replicas should be handled. The exact way the replication is performed is up to the topology plugin in use in the network. There is also no mention of if or when a node retrieving data from the overlay should query replicas.

4.4.2.4 Issues

In the specific case of using RELOAD for P2PSIP, some security issues remain. Most importantly, if user y initially trusts user z and forwards their calls to them but later y stops trusting z, a door is open for a replay attack compromising data integrity. Such an attack, if successful, would lead an unknowing node x calling y to reach z instead and believe z to still be trustworthy.

In such a scenario, y would first store an AOR containing $y \rightarrow z$, which $P(y)$ would accept. A user x calling y at this point would instead reach z (or any user z is forwarding their calls to). Validating the signature of the AOR would confirm that this is y's decision and z's identity can also be verified when establishing the (D)TLS connection. If y stops trusting z — say because z lost their device and y knows it — y will update their AOR to remove this forwarding to z, for example by replacing it $y \rightarrow id(y)$. A malicious $P(y)$, perhaps colluding with z, could however keep serving y's previous AOR. When calling y, x would get this old AOR which should now be

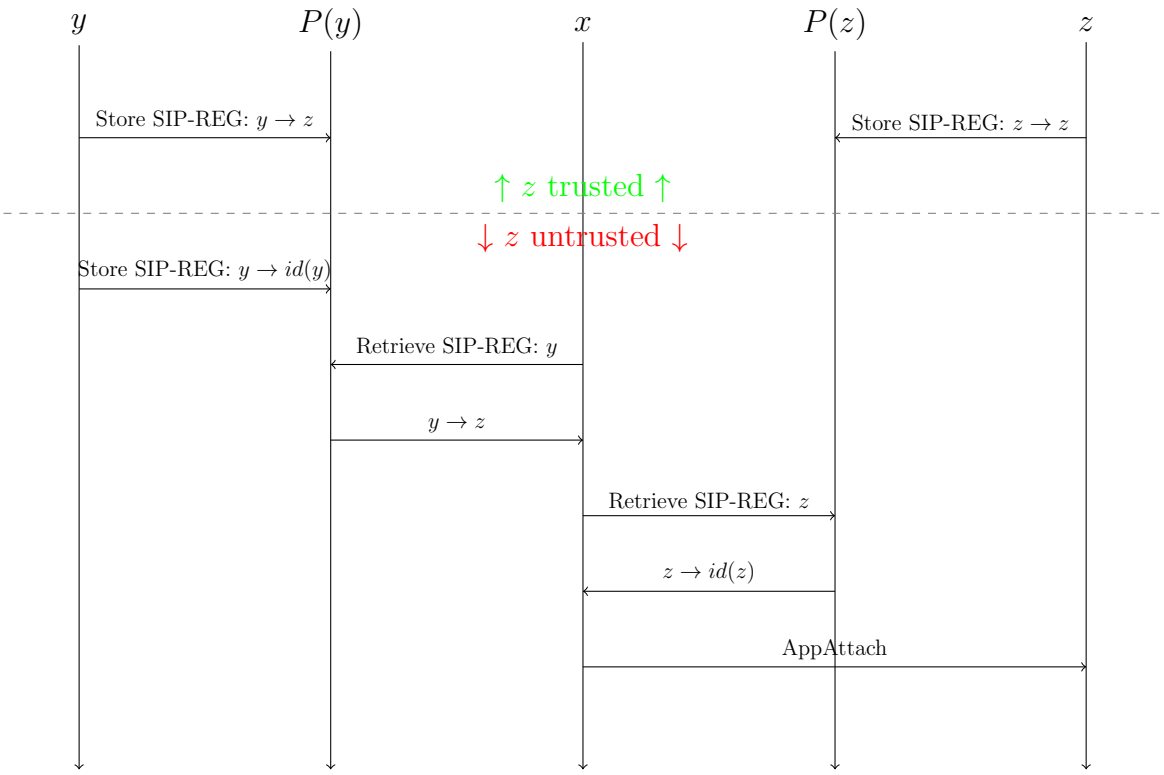


Figure 4.2: Replay Attack Against P2PSIP over RELOAD

invalid, but the signature will still be valid and so x will believe that y is willingly forwarding the call to z. x will also successfully validate z's identity and a call will be initiated, despite y not trusting z anymore. Figure 4.2 illustrates this attack. Also of note is the fact that $P(y)$ does not necessarily have to be malicious for this to be a problem: it could be a well-intentioned but malfunctioning node.

Another detail which may be considered an issue is that, if y trusts z to take their calls, z may in turn forward forward their calls to any other user, which y may not trust. Lastly, although it is considered a feature more than an issue, it is important to note that z has no say in whether y forwards their calls to them.

As for resilience, redundancy is left to the topology plugin to handle. In the only topology plugin defined in the specification, however, $P(y)$ is responsible of sending

storage requests for replicas of y's AOR. This means that redundancy is useless in case $P(y)$ is compromised or malfunctioning, which could lead to data loss and availability problems.

The RFC does include a method for the storing node to verify whether replicas have been stored, but does not require this verification. This process is also flawed, as it is based on $P(y)$ response to the storing request: $P(y)$ responds with the list of nodes where replicas will be stored and y can then query these nodes to verify. If $P(y)$ is malicious, it can send a list of other malicious nodes with which it colludes, that can then falsely confirm that the replica was stored.

All of these issues can be solved; our solutions in section 5.2.2 will address them.

4.5 Conclusion

In this chapter, we presented the main state-of-the-art solutions to the different pieces of a secure P2PSIP implementation on top of a MANET. This included different approaches to P2PSIP over MANETs, to DHTs over MANETs, to DHT security and a focus on a standardized protocol for P2PSIP with security considerations.

We conclude from this overview that none of these solutions is perfect for our specific context and objectives, detailed in chapter 3. Many proposals for SIP over MANETs, or just P2PSIP, do not consider security at all. Those that do still present issues, in some cases by oversight and in others because they are designed for a different context and either cannot make as many assumptions as we can or they are simply based on different assumptions. Some solutions like P2PNS do not consider replay attacks, for instance, and some approaches to DHT security rely on an online CA, which is unrealistic for a MANET. RELOAD comes close to offering a fully secure

P2PSIP implementation and it can be used with MANETs despite not being designed with this type of network in mind, but it also has its issues with its under-specified redundancy-scheme and its call forwarding feature as we described in section 4.4.2.4.

A better solution, more adapted to the context we gave ourselves and more secure, is thus required. In order to build this better solution, we learned from the state of the art that a PKI is the surest way to protect from Sybil attacks (and Eclipse attack indirectly). We also note, however, that having an online CA is unrealistic in our context. We also learned of the usefulness of redundancy to help with resilience as well as storage and retrieval attacks. Issues of state-of-the-art solutions with replay attacks reminded us of the importance of considering this specific attack. RELOAD considered all of these and provided us we a solid base to start with, but also taught us of the danger of leaving details unspecified and supporting extra features that may not be necessary.

The next chapter provides two variants of a solution that puts these pieces together to build a fully secure P2PSIP implementation for the context in chapter 3.

Chapter 5

Two Approaches to P2PSIP

We propose two approaches to building a secure MANET for P2PSIP. In this section, we focus on the details of the operation of the network for each approach. Both are based on general approaches seen in chapter 4. The first one, described in section 5.1, is a simple method in which network flooding is used to resolve SIP URIs to IP addresses, while the other replaces SIP registrars with a DHT, similarly to RELOAD, and is detailed in section 5.2.

5.1 Flooding-based P2PSIP

The first approach we propose is rather simplistic. It is based on a standard flooding protocol and is not expected to provide the best performance in terms of network utilization and latency, but section 5.1.2 will show that it has some security benefits: it provides fully redundant routing and thus makes it easy to protect from routing attacks.

In this approach, local storage is used, just like it was presented in section 4.1.1. Rather than having SIP registrars keeping records of other nodes' AORs, each node is

responsible for its own AOR. This means that the registration step of SIP is eliminated, as each node has knowledge of their own SIP URI and IP address.

When a node Q needs to reach another node R, it sends a resolve request to all of its neighbors, indicating R's SIP URI, as well as its own SIP URI and IP address. Any node S receiving such a request checks the SIP URI enclosed in the message and compares it to its own. If they do not match, meaning that $S \neq R$, S forwards the request to all of its neighbors. If the SIP URIs match, that is if $S = R$, R creates a response message by adding its IP address to the request and then broadcasts it to the network for it to be routed back to Q following the same mechanism. When Q receives this response, the connection is established and the two nodes can communicate using any IP-based MANET routing protocol. It is relevant to note, however, that if flooding is not used to carry this communication, it may be unsuccessful even after a successful resolve request. This is because flooding provides security advantages, as will be detailed in section 5.1.2.

For this to work well, two last steps are required: preventing messages from remaining in the network and being perpetually forwarded between nodes, as well as collision avoidance. For the former, duplicate packet detection is implemented, by adding a large random number to each message as an identifier and using it to detect and drop duplicate packets. Finally, to avoid collisions, a small random delay is added before sending messages. This avoids a situation where all nodes forward or respond to a message at the same time, potentially interfering with each other.

5.1.1 Attacks in a Flooding Context

There are two potential angles of attack for a malicious node in the flooding-based approach: attacking the resolve request messages and attacking the response messages.

The simplest possible attack is a denial of service attack that can target both by simply dropping all messages. M can perform this attack on every single resolve request happening in the network, because all nodes receive all resolve requests and responses. We will call this the *Drop Messages Attack*.

To target data integrity, M can focus on request messages and respond to any of them, even if it is not the destination of the request. M's response can include invalid data to direct Q to the wrong node, for example M itself. We have implemented this attack and called it the *Resolve to Self Attack*. It causes Q to initiate a session with the malicious node M.

M could also focus on response messages by editing them with an invalid contact address, like its own, before forwarding it to its neighbors. We call this the *Edit Responses Attack*. This attack has the advantage of potentially preventing legitimate nodes from forwarding the valid response message if M's edited response makes it to their packet cache first. Vis-à-vis the *Resolve to Self Attack*, however, it has the disadvantage that M cannot respond to the request before R, thus making it less likely for its bogus response to make it to Q first.

5.1.2 Securing Flooding-based P2PSIP

In this section, we propose a solution to secure SIP resolution in a MANET based on a simple flooding protocol as described in section 5.1. At the heart of this security solution is an offline PKI. We first tackle the data integrity issue in section 5.1.2.1 before discussing the system's resilience in section 5.1.2.2.

5.1.2.1 Data Integrity

As a mirror of the attacker's main objective of compromising data integrity, our main goal for security is to protect this data integrity. This means that malicious nodes should not have the ability to fool a legitimate node Q into believing that user y is located at node S with address A if such is not the case. Q should have the ability to detect such an attempt from malicious nodes and stop trying to communicate with y before sending any meaningful data to S.

Because the flooding-based approach does not include the registration mechanism from SIP (each node stores its own AOR), the only attack vector is at query time. A malicious node will try to respond to Q's query with invalid data as discussed in section 5.1.1.

When Q sends a query for y's current address, it may receive multiple responses: one from each of its physical neighbors. Because of the duplicate message detection, described in section 5.1, only the first one is considered. If this first response is malicious, then Q is fooled. Waiting for all responses to arrive before trusting any of them would be an improvement, but that presents an issue: if more malicious responses are received than honest ones, then Q is still fooled as to y location.

To fix this issue, Q needs a way to differentiate malicious responses from legitimate ones. To achieve that, we propose using an offline PKI to authenticate responses.

Setup An offline CA creates a key pair for itself and the associated certificate. It also issues a certificate for every SIP URI in use in the system, signed by the CA's private key. Before the network is deployed, each node is given its own certificate (userCert) and the associated key pair, as well as the CA's certificate (rootCert), which contains the CA's public key. This is illustrated in figure 5.1. The keys shown in the

figure are private keys, while the corresponding public keys are part of the certificates immediately to their left.

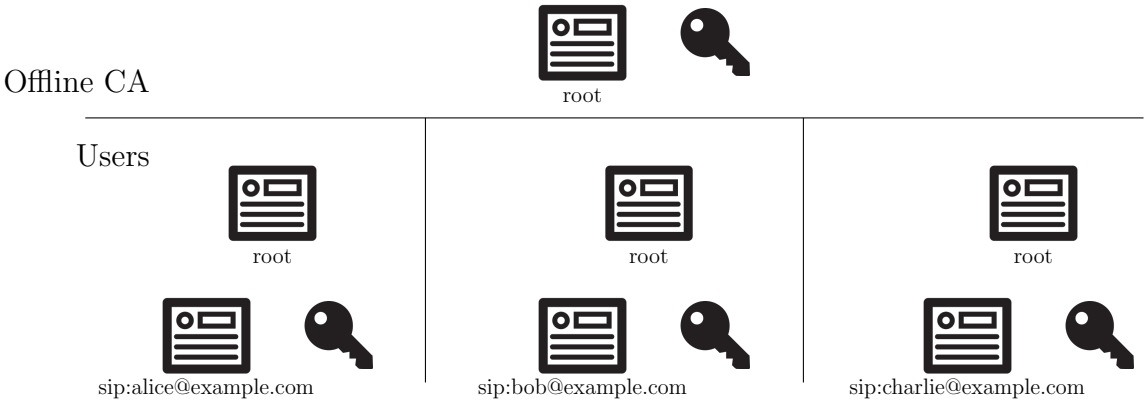


Figure 5.1: PKI Setup

In the context in which we operate, described in chapter 3, this is not a major overhead. Indeed, a central authority figure owns and is in control of the network and all nodes correspond to devices that will have to be prepared for missions before being deployed anyway.

Details With this setup in place, a node R, acting on behalf of user y, can now prove its real identity and authenticate its responses by using its private key to sign them and sending its certificate along with them. Other nodes can then validate these responses by verifying that:

- the SIP URI in the response corresponds to the one in the userCert,
- the message was indeed signed with the private key associated to the public key found in the userCert,

- the userCert was signed with the private key associated to the public key in rootCert.

If a node — Q or an intermediary node — cannot validate a response, it should drop it rather than adding it to the cache that is used for duplicate message detection. This way, a malicious node cannot forge a response, or it will simply be dropped, preventing cache poisoning attacks.

One thing that still remains to be fixed, however, is replay attacks. A malicious node cannot forge responses, but it can still use previous responses from R with an outdated contact address in order to mislead Q. To fix this, we can require that the request message's identifier, which is random, be included in the response and signed. This acts as a nonce, meaning previously signed responses from R will not have the correct request identifier and will not be considered valid.

When Q receives a response that it can successfully validate, it can trust that it came from R and the session can be established.

5.1.2.2 Resilience

Once data integrity is fixed, an attacker may instead focus on denying service to legitimate nodes, preventing them from communicating together. To do so, this malicious entity needs to prevent the resolve request from Q from making it to R or R's response from making it back to Q.

By design, this flooding approach offers fully redundant routing, meaning that to achieve this goal, a malicious actor needs to control all paths between Q and R. This would constitute an Eclipse attack. The one exception to this is if the attacker manages to get an invalid message into other nodes' message cache, thus preventing legitimate messages with the same identifier from being forwarded. A way to fix this

last issue is for nodes to differentiate messages not just by their identifier, but also by their source node, when it comes to caching. This requires that those messages be unforgeable, which can be achieved by signing them. We already explained how this is done for responses in section 5.1.2.1, but the same mechanism can be applied to requests.

5.1.2.3 Formalization

To formalize this security protocol, we need to extend the notation presented in section 3.2. We use N to denote nonces, C for for certificates, K for public keys and K^{-1} for private keys. Subscripts denote an association to a node or user, such that K_y^{-1} is user y's private key, for example. A dot is used for concatenation. The encryption of a message m with key K is indicated as $\{m\}_K$.

Given this new notation, the protocol detailed above is as shown in table 5.1. This table shows, on each row, the step in the protocol, the nodes between which the message is sent and its content. For example, step 1 is a message from Q to R containing $N_Q.y$. Also, as mentioned above, messages are sent by flooding the network and the signature in step 2 is validated by all intermediate nodes as well as Q.

1	$Q \rightarrow R : N_Q.y$
2	$R \rightarrow Q : \{N_Q.y.A_y\}_{K_y^{-1}}.C_y$

Table 5.1: Security Protocol for Flooding-based Approach

In this thesis, we focus on authentication of the callee by the caller. We quickly note, however, that the same mechanism could be used for the callee to authenticate the caller, as shown in table 5.2.

1	$Q \rightarrow R : N_Q.y$
2	$R \rightarrow Q : \{N_Q.y.A_y.N_R\}_{K_y^{-1}}.C_y$
3	$Q \rightarrow R : \{N_R.x\}_{K_x^{-1}}.C_x$

Table 5.2: Extended Security Protocol for Flooding-based Approach

5.1.3 Summary

In this section, we presented a solution to secure a P2PSIP over MANETs using local storage and flooding-based SIP URI resolution. We used a PKI with an offline CA and challenge-response mechanism with nonces to provide user authentication. Flooding-based routing also provides routing redundancy for this solution to help with resilience. Next, we will extend these concepts to a more structured approach to P2PSIP for MANETs, using a DHT as the storage scheme.

5.2 DHT-based P2PSIP

The second approach is more structured, as nodes in the network form a DHT that is used to replace SIP registrars.

Using this approach, all nodes need to register their SIP AOR with the DHT. A node R trying to do so uses a hash function to reduce its SIP URI to a value k in the key space of the DHT. It then sends a PUT request to that key with the value being its AOR. The node S responsible for this key k receives this PUT request and stores R's AOR.

When a node Q needs to reach R, it hashes R's SIP URI, which yields k. It then sends a GET request to the DHT for this key k. S receives this GET request, as it is responsible for this key, and responds with R's AOR. From the AOR, Q gets R's IP address and communication between Q and R can then be established.

Lastly, if IP addresses are not static, this scheme requires that nodes update their AOR that is stored in the DHT whenever their IP address changes.

5.2.1 Attacks in the DHT Context

When a DHT is used for P2PSIP, the attack surface gets larger. An attacker can target the DHT's routing mechanism, which is responsible of locating $P(y)$, the query mechanism or the insertion mechanism.

The simplest routing-based attack is for M to simply drop the message used to locate $P(y)$. To do this, M obviously needs to be on the path followed by this message. This attack is called a *Drop Find Node Attack* in the *OverSim* library for *Omnet++*. A more sophisticated routing attack consists in providing an incorrect response to the message. When M is asked for the next hop to reach $P(y)$, it sends Q on a false trail by answering with a random IP address or by saying that it is $P(y)$ itself, for example. *OverSim* calls these the *Invalid Nodes Attack* and *Is Sibling Attack*, respectively. The latter is illustrated in figure 5.2 and sets the table for a query-based attack, which would allow M to compromise data integrity rather than only availability.

A query-based attack is one that is performed by M when it happens to be $P(y)$ or when Q has been fooled into thinking M is $P(y)$. Again, the simplest case is a denial of service attack, by refusing to cooperate and not serving the data by not responding to the query. A more interesting case would be for M to answer the query with invalid data. *OverSim* implements this in its *Invalid Data Attack*, which causes malicious nodes to respond to DHT queries with random data. This data is not in the correct format for a P2PSIP response, so it is easily detected by Q without any security mechanism and thus only causes a denial of service. For M to target data integrity, we have implemented the *Resolve to Self Attack*. Akin to its flooding-based

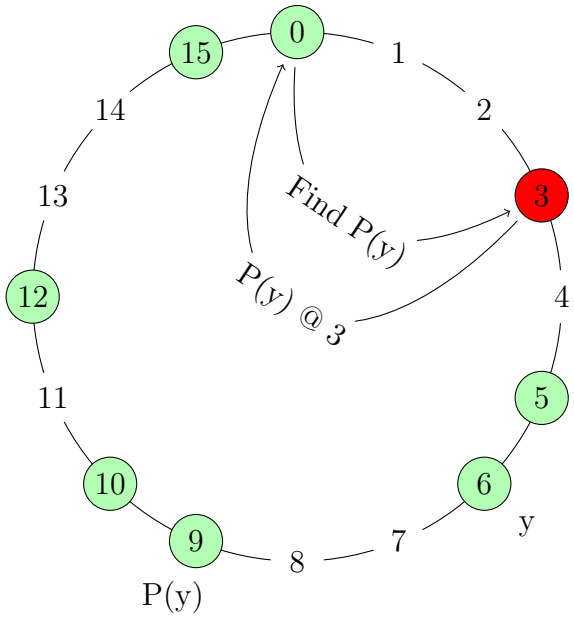


Figure 5.2: Is Sibling Attack

counterpart, it makes a malicious node M respond to Q's resolve request with its own contact address rather than R's. This is illustrated in figure 5.3.

We note that the issues related to call forwarding that affect RELOAD, detailed in section 4.4.2.4, do not affect our solution, because we do not support call forwarding. RELOAD users could obviously agree not to use the call forwarding feature to get the same result.

DHT poisoning, shown in figure 5.4, is an attack against the insertion mechanism of the DHT. In this attack, M inserts or overwrites data in the DHT with fabricated data, which is then unknowingly served to honest nodes. For example, M could insert AORs mapping other users' SIP URIs to its own IP address. For M, this has the advantage of being much easier to execute than a routing- or query-based attack.

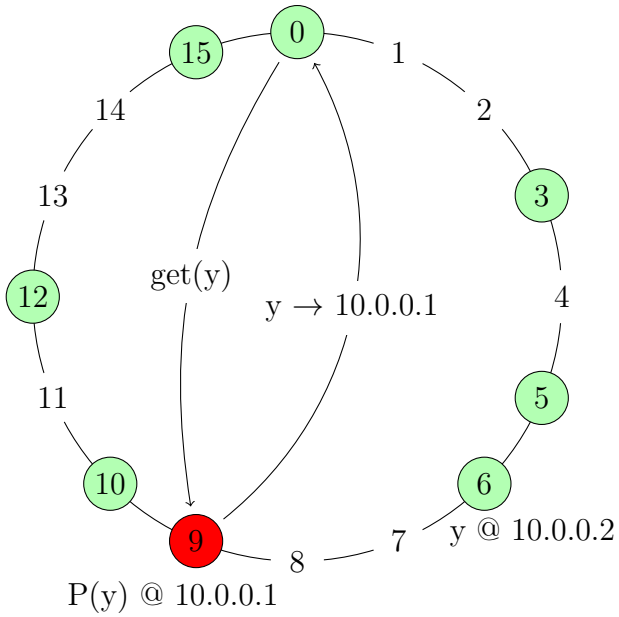


Figure 5.3: Resolve to Self Attack

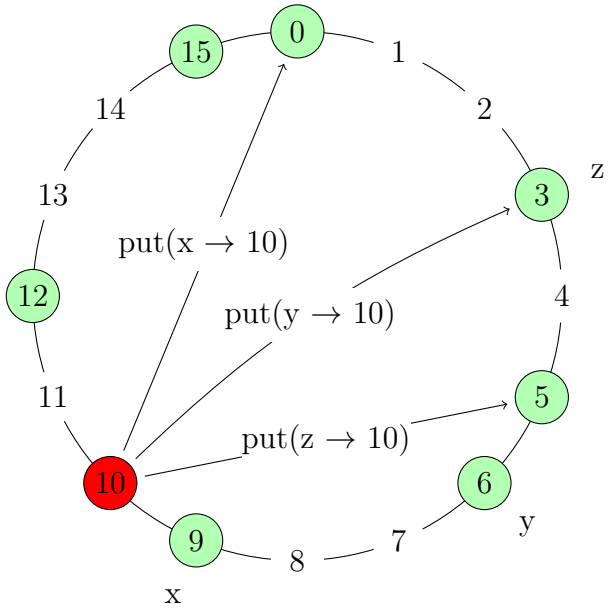


Figure 5.4: DHT Poisoning

5.2.2 Securing the DHT for P2PSIP

This section presents our multi-layer approach to securing the DHT-based SIP resolution mechanism described in section 5.2. Just like our solution to the flooding-based approach, this one is centered around an offline PKI. First, we discuss how to address data integrity protection and, then, we focus on ensuring the system is resilient.

5.2.2.1 Data Integrity

Again, our main objective is to protect data integrity, which means a malicious actor should not be able to convince Q that y is currently located at address A if this is not the case. With the DHT-based approach, there are two vectors through which an attacker could attempt this. The attacker could act:

- at query time. When Q queries the DHT for y's AOR, a malicious node M could perform a query-based attack, as discussed in section 5.2.1, possibly combined with a routing-based attack. This would allow M or an accomplice of M to respond to Q's query with false information.

- at insertion time. This means a malicious node M could insert invalid data into the DHT so that when Q requests y's AOR, $P(y)$ unknowingly responds to Q with tampered information. This corresponds to the DHT poisoning attack detailed in section 5.2.1 and is particularly important because of its simplicity and efficiency compared to query time attacks.

In the context of P2PSIP, the simplest way for Q to ensure that the data retrieved by the DHT is valid is through a cryptographic challenge mechanism. At a high level, this works by having Q take the IP address from the AOR that it received and ask

the node at this address to prove that it is indeed the intended destination user y. This counters attacks performed through both vectors above.

To achieve this, we make use of the same PKI setup described in section 5.1.2.1. This gives R, acting on behalf of y, the technical means to sign Q's challenge. Some additional considerations need to be taken in order to make sure the answer to the challenge is not replayable nor transferable.

First, to avoid a malicious node M being able to intercept the challenge response from R and reuse it later to impersonate R and the associated user y, the challenge will include a random value, or nonce. R will sign the random value and send it back as its response along with its certificate. When Q receives the challenge response, it will validate that this nonce is the same as what it sent and that it is properly signed by R's private key. This way, if M attempts to reuse an old challenge response from R, the nonce will not match. M is obviously not capable of faking R's signature with the proper nonce.

Then, we need to ensure that if M managed to have y's SIP URI resolve to its own IP address, it cannot simply transfer the challenge to R before sending it back to Q. This means that the challenge response should be specific to y's location. The challenge response should thus include y's IP address as well as the nonce, all of which is signed. Q can then verify that the address received in the challenge response correspond to the one it sent the challenge to. If this is not the case, it means that the challenge was transferred from a malicious node to R.

Only when the challenge is successful should the session be initiated. If the challenge fails, it means that the node controlling the IP address to which Q's query resolved is malicious. In such a case, Q should halt its attempt to communicate with y.

5.2.2.2 Resilience

With the challenge mechanism described in section 5.2.2.1, data integrity is protected. However, any attempt at compromising data integrity now results in a failed challenge and thus in denial of service. This means the system is not very resilient. To fix this, we will first tackle DHT poisoning and storage exhaustion attacks, followed by replay attacks and finally routing- and query-based attacks.

Preventing DHT Poisoning and Storage Exhaustion Attacks With only the challenge mechanism in place, one way that malicious nodes can cause challenges to fail is by inserting invalid data into the DHT, either to overwrite valid entries or to use up all storage resources of a node and causing it to be unable to store valid entries. To prevent this, whenever a node is asked to store an AOR, it should be able to validate it first.

To achieve this, we also rely on the PKI and require that all AORs to be inserted into the DHT be signed by the node inserting it. The node responsible for storing this AOR then needs to validate that the SIP URI in the AOR corresponds to the SIP URI of the node that signed the entry, as indicated in the certificate sent along with the insertion request.

This signature validation ensures that a malicious node cannot insert invalid data into the DHT. This thus makes it impossible for it to perform a DHT poisoning or storage exhaustion attack, reducing the attack surface available to a malicious actor.

Limiting Replay Attacks Now that a malicious node cannot insert arbitrary data into the DHT, an attacker may try a replay attack. For example, if y used to be located at an IP address A but has moved, a malicious node M may try reusing y's

previous AOR that maps its SIP URI to A and inserting it into the DHT. If successful, this would cause a node Q trying to reach user y to be unable to do so.

To fix this, when receiving an insertion request, a node should be able to validate that this is a new AOR, rather than an outdated one. We propose doing this by adding a timestamp to AORs. Because the AOR needs to be signed, it is not possible for a malicious node to forge an invalid timestamp. Then, when a node receives a request to insert an AOR, it needs to validate the timestamp.

Say S receives a request to insert user y's AOR, located at node R and address A. If S already has an AOR for user y's SIP URI, it needs to validate that the timestamp in the new AOR is indeed more recent than the timestamp in the existing, already stored AOR. If this is not the case, it means that someone is trying to insert an older AOR into the DHT and S should reject it. If S does not already have an entry for y's AOR, it is unlikely that the insertion could be a replay attack because, in the vast majority of cases, this insertion request should be y's first, which means that there are no old requests to replay. However, it is possible that y's first insertion request did not make it to $P(y)$ but that M intercepted it. In this case, however, y is already unreachable, so the replay attack is harmless: when Q tries to contact R, the challenge will fail, resulting in the session initiation attempt being dropped, just like if $P(y)$ had no AOR for R.

Limiting the Effect of Retrieval Attacks Attacks at insertion time are now taken care of, but there still remains routing- and query-based attacks. Those can be handled by adding redundancy to the storage and retrieval mechanism. The same AOR can be stored by multiple nodes and all of those nodes can be queried at session initiation time. To avoid the issue that RELOAD has with redundancy where

the node receiving the storage request is responsible for sending replicas to other nodes (presented in section 4.4.2.4), the storing node needs to handle the multiple storage requests itself. For example, it can send storage requests for key k to r nodes, uniformally distributed in the m-bit identifier space, by sending them to $P(k + \frac{im^2}{r} \pmod{m^2})$ for i from 0 (inclusive) to r (exclusive).

When retrieving an AOR, the querier also needs to handle the multiple requests itself, again to avoid the issue that affects RELOAD. Additionally, if the querier receives more than one response to their resolve request, it needs to be able to determine which one is the most trustworthy. This is simple: the correct AOR will be properly signed by the destination node's private key and it will be the latest AOR created by this node. The querier thus needs to trust the most recent properly signed AOR that it received, based on its timestamp.

To prevent the query from succeeding, a malicious node then needs to prevent all legitimate nodes from responding, be it by controlling all nodes responsible for the data or all paths to reach those nodes. Whether this malicious entity performs a typical DoS attack, for example a *Drop Find Nodes* attack, or a more elaborate attacking usually targeting data integrity like a query-based replay attack, the result will be the same. If this malicious actor manages to prevent all honest nodes from responding and itself sends outdated AOR entries for the destination node, the challenge will fail and service will be denied but data integrity will be preserved.

5.2.2.3 Formalization

To formalize the security protocol for this DHT-based approach, we need to define a last piece of notation: T denotes a timestamp. Given this, the protocol works as shown in table 5.3. In this table, we use $Q \rightarrow R$, for instance, to denote key-based

routing while $A_x \rightarrow A_y$ denotes a message sent between the same two nodes using MANET routing.

Insertion	1	$R \rightarrow P(y)$	: PUT.$\{y.A_y.T_A\}_{K_y^{-1}}.C_y$
Query	2	$Q \rightarrow P(y)$	: GET.y
	3	$P(y) \rightarrow Q$	: $\{y.A_y.T_A\}_{K_y^{-1}}.C_y$
	4	$A_x \rightarrow A_y$	: N_Q
	5	$A_y \rightarrow A_x$	: $\{N_Q.y.A_y\}_{K_y^{-1}}.C_y$

Table 5.3: Security Protocol for DHT-based Approach

,

Figure 5.5 shows a visual representation of table 5.3. Numbered arrows correspond to the numbered messages in the table.

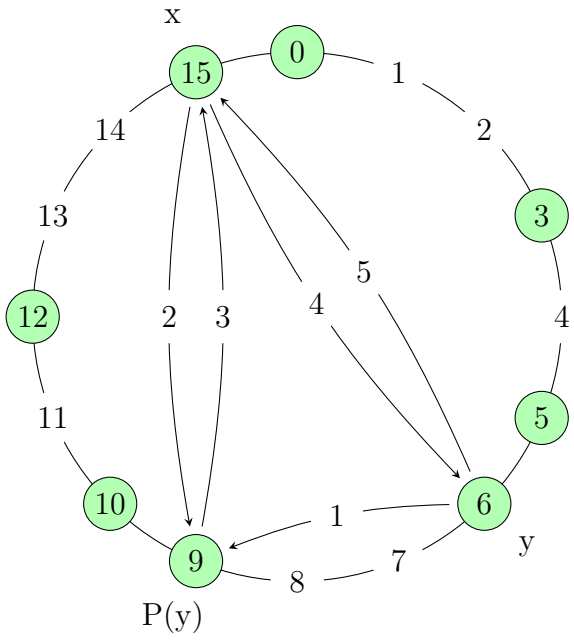


Figure 5.5: Security Protocol for DHT-based Approach

Just like we did with the flooding-based approach, we note that it would be possible to extend this protocol to include caller authentication by the callee, even if this is not the focus for this thesis. This is shown in table 5.4.

Insertion	1	$R \rightarrow P(y)$	: PUT.$\{y.A_y.T_A\}_{K_y^{-1}}.C_y$
Query	2	$Q \rightarrow P(y)$	: GET.y
	3	$P(y) \rightarrow Q$	: $\{y.A_y.T_A\}_{K_y^{-1}}.C_y$
	4	$A_x \rightarrow A_y$	: N_Q
	5	$A_y \rightarrow A_x$	: $\{N_Q.y.A_y.N_R\}_{K_y^{-1}}.C_y$
	6	$A_x \rightarrow A_y$	: $\{N_R.x.A_x\}_{K_x^{-1}}.C_x$

Table 5.4: Extended Security Protocol for DHT-based Approach

5.2.2.4 Comparison with RELOAD

Formalizing RELOAD would yield a result similar to table 5.4, at least conceptually. It would however be more complex and would show significant overhead compared to our simpler solution. Key differences would include the following:

- Certificates would be stored and retrieved separately from the AOR. The specification is unclear to us, however, about exactly how and when they are retrieved from the overlay for signature verification.
- A_y would be replaced with R in the AOR and key-based routing would be used between Q and R.
- A full (D)TLS handshake, including protocol negotiation, would be used instead of messages 4 to 6.
- NAT traversal with ICE would be added, with the associated intermediary server(s).

The other main difference is how replica storage is handled. Assuming that the key identifier for y's AOR is calculated with $h(y)$, table 5.5 highlights this difference in the insertion mechanism for a network using 4-bit identifiers and storing 2 copies of all AORs.

Our solution	1	$R \rightarrow P(h(y))$	: PUT.$\{y.A_y.T_A\}_{K_y^{-1}}.C_y$
	2	$R \rightarrow P(h(y)+8)$	: PUT.$\{y.A_y.T_A\}_{K_y^{-1}}.C_y$
RELOAD	1	$R \rightarrow P(h(y))$	: PUT.$\{y.R.T_A\}_{K_y^{-1}}.C_y$
	2	$P(h(y)) \rightarrow successor(P(h(y)))$	: PUT.$\{y.R.T_A\}_{K_y^{-1}}.C_y$

Table 5.5: Comparison with RELOAD's Insertion Mechanism

A summary of all the differences between our DHT-based solution and RELOAD is shown in table 5.6. Again, it comes down to our solution being simpler, with an improvement in security over RELOAD.

RELOAD	Our solution
Supports signaling for arbitrary applications including P2PSIP	Specific to P2PSIP
AORs map SIP URIs to node IDs	AORs map SIP URIs to IP addresses
Key-based routing for application data	IP routing for application data
Certificates stored in the DHT	Certificates sent with messages
Full (D)TLS for all connections	Simple signature scheme with challenge mechanism
NAT traversal support	No NAT traversal support
Call forwarding support (with minor security issue)	No call forwarding
Single point of failure in storage redundancy scheme	No single point of failure in storage redundancy scheme

Table 5.6: Differences between our solution and RELOAD

5.2.3 Summary

This section detailed a secure DHT-based approach to P2PSIP over MANETs. We extended the concepts presented in section 5.1 to the different storage scheme and used an offline CA and a challenge mechanism for user authentication. We used this

PKI and storage redundancy to improve resilience in the presence of malicious nodes.

We note, again, that our solution is similar to RELOAD, which also uses a DHT for storage and provides protection from the attacks listed in section 5.2.1. The main differences are the lower complexity and overhead of our solution and some security issues specific to how RELOAD is specified, as discussed in section 4.4 and section 5.2.2.3.

5.3 Conclusion

This chapter presented two approaches to a secure P2PSIP implementation over a MANET, one based on a local storage scheme with flooding-based retrieval of AORs and the other based on a DHT. Both were designed to fix issues that were identified in state-of-the-art solutions presented in chapter 4. We note that, for both solutions, the focus for data integrity is put on the caller authenticating the callee, but the same mechanisms could be used for the callee to authenticate the caller. The next chapter will present experimentation results obtained by simulating the two approaches to show the effectiveness of their design.

Chapter 6

Experiments

In this section, we describe our experiments and results, starting with our methodology in section 6.1, followed by the flooding-based solution in section 6.2 and the DHT-based solution in section 6.3. Then, in section 6.4, we discuss the results of these experiments.

6.1 Methodology

As previously mentioned, we implemented simulations using *Omnet++*, along with the *OverSim* framework to implement the DHT. We simulated a network of 25 static nodes placed such that there exists at least one path between any two nodes, assuming all honest and cooperating nodes.

For simulations of attack scenarios, each node has a given probability of spawning as malicious, as indicated on each figure. In all experiments where the effect of changing the number of malicious nodes is evaluated, it ranges from no malicious node at all to each node having a 50% chance of being malicious.

Every node periodically tries, every 30 seconds, to resolve a SIP URI among those

of all nodes. This process starts after 100 seconds, to let the network build itself and stabilize, which is important for the DHT approach. We simulated 600 seconds per simulation and each scenario was run 20 times in order to account for randomness and in order to get statistically significant results. Margins of error corresponding to 95% confidence intervals are also shown on all graphs.

Details specific to each of our two solutions will be included in their respective subsection.

6.2 Flooding-based Solution

First, we cover experiments with the approach using network flooding. We go over our methodology and then present our results.

6.2.1 Methodology

The cache we used for duplicate message detection is a simple bounded queue with a capacity of 512 packets. The random delay used to prevent collisions varies from 0 to 10 milliseconds and follows a uniform distribution.

The defense mechanism presented in section 5.1.2 has been implemented, along with the *Drop Messages*, *Resolve to Self* and *Edit Responses* attacks discussed in section 5.1.1.

6.2.2 Results

This sections shows the result of our simulations in which we evaluated the effect of different attacks on the resolve success rate as well as observed the total amount of traffic generated.

6.2.2.1 Drop Messages Attack

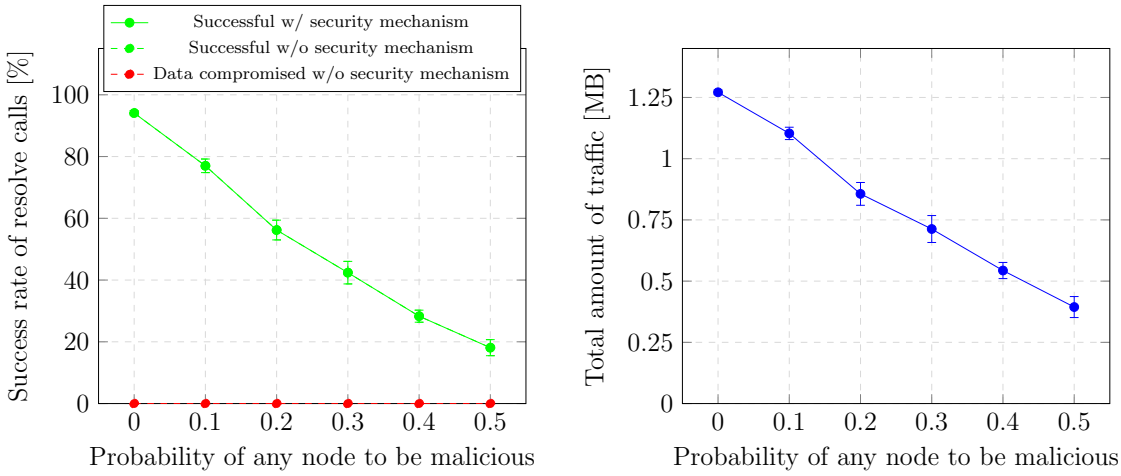
This attack is a denial of service attack for which our security solution itself is not expected to help, but the nature of the resolve mechanism, being based on flooding, is expected to provide great benefits: there needs to be a malicious node on all paths between the caller and the callee for the attack to have an effect.

Figure 6.1a shows the effect of the attack on the success rate of resolve calls. We notice that, evidently, the success rate goes down as the proportion of malicious nodes goes up. It drops faster than the increase in malicious nodes. This is because, given that a node is malicious with probability p, each resolve call fails with probability p, due to the SIP URI being resolved belonging to a malicious node. In addition, even if the end node itself is not malicious, a resolve call fails if malicious nodes cut all paths between the caller and the callee. Also, the two green lines overlap, confirming our expectation that the security mechanism does not play a role in this scenario.

Figure 6.1b shows the network utilization in this same scenario, calculated as the total amount of traffic by the MAC layer to the physical layer. It is indirectly proportional to the number of malicious nodes, as expected, because malicious nodes do not forward messages or respond to resolve requests. The amount of traffic when all nodes are honest is most interesting though, for comparison with the DHT-based solution results in similar circumstances, that follow in section 6.3.

6.2.2.2 Resolve to Self Attack

This attack targets data integrity and is thus expected to be countered by the signature and verification mechanism of our security solution. Figure 6.2a shows that this is the case: there is only a slight decrease in the success rate of resolves as the number of malicious nodes increases. This decrease is due to situations where all paths between



(a) Resolve Calls Statistics

(b) Network Utilization

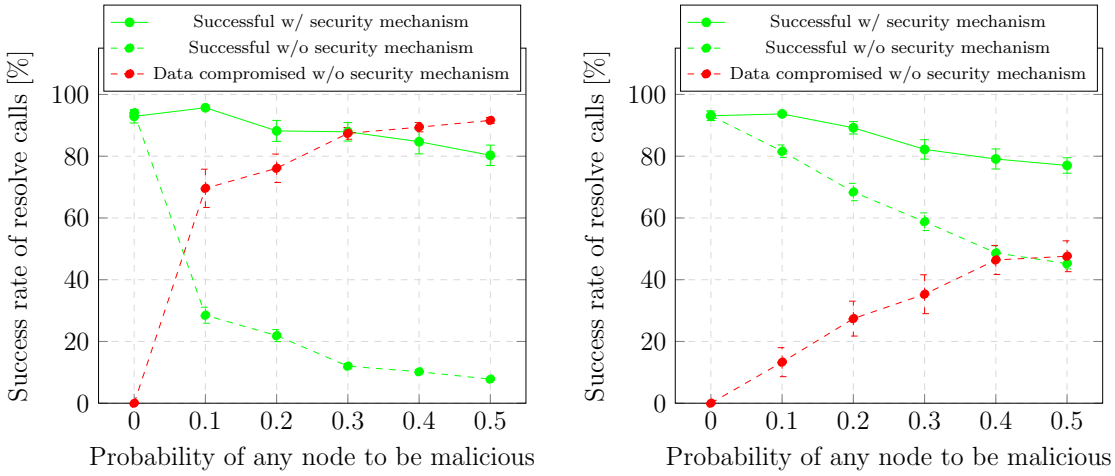
Figure 6.1: Drop Messages Attack Statistics

the calling and called nodes count at least one malicious node. In this case, the node attempting to resolve a SIP URI only receives invalid AORs, so the resolve fails, but no session is established because it detects that the AORs are invalid. Without the security mechanism and assuming the first resolve response received for a given request is always trusted, a few malicious nodes suffice to greatly affect the integrity of the network. For instance, when every node has only a 10% chance of being malicious, we observe that over 60% of requests are compromised. This is shown with dashed lines in figure 6.2a, where the red one represents cases in which the session would be established with the wrong, malicious node.

6.2.2.3 Edit Responses Attack

With the security solution in place, this attack is expected to yield the same result as the *Resolve to Self Attack*, following the same reasoning. Figure 6.2b confirms that this is indeed the case. Without the security mechanism, this attack is less effective than the *Resolve to Self Attack*, however. This is due to the fact that, in a *Resolve to*

Self Attack, malicious nodes respond to the request before the destination node does if they are closer to the querier, which is not the case with an *Edit Responses Attack*. Additionally, a response that has been edited keeps the same message identifier and thus may not be forwarded by all nodes because of the duplicate packet detection mechanism.



(a) Resolve to Self Attack

(b) Edit Responses Attack

Figure 6.2: Resolve Calls Statistics

6.3 DHT-based Solution

Now, we describe our experiments with the approach making use of a DHT, before presenting our results.

6.3.1 Methodology

These experiments were run using a Chord DHT [8] over an OLSR network [63]. After joining the DHT, nodes repeatedly try registering their AOR with the P2PSIP service

by storing it in the DHT, until they succeed and nodes only try resolving SIP URIs that have been registered.

The full security solution presented in section 5.2.2 has been implemented. Some of our experiments, however, focus on the data integrity aspect and used the PKI and cryptographic challenge mechanism in isolation. The *Resolve to Self Attack*, presented in section 5.2.1, was also implemented.

6.3.2 Results

In this section, we show the results of our simulations with the DHT-based approach. These evaluate the effect of the different attacks on the ability of nodes to join the DHT, the success rate of resolve calls and the amount of network traffic generated. These statistics are evaluated for different proportions of malicious nodes, again ranging from none to half the total number of nodes, as well as the different levels of redundancy. Once more, the margin of error for 95% confidence intervals is displayed on all graphs.

6.3.2.1 Challenge Mechanism: Attacks in *OverSim*

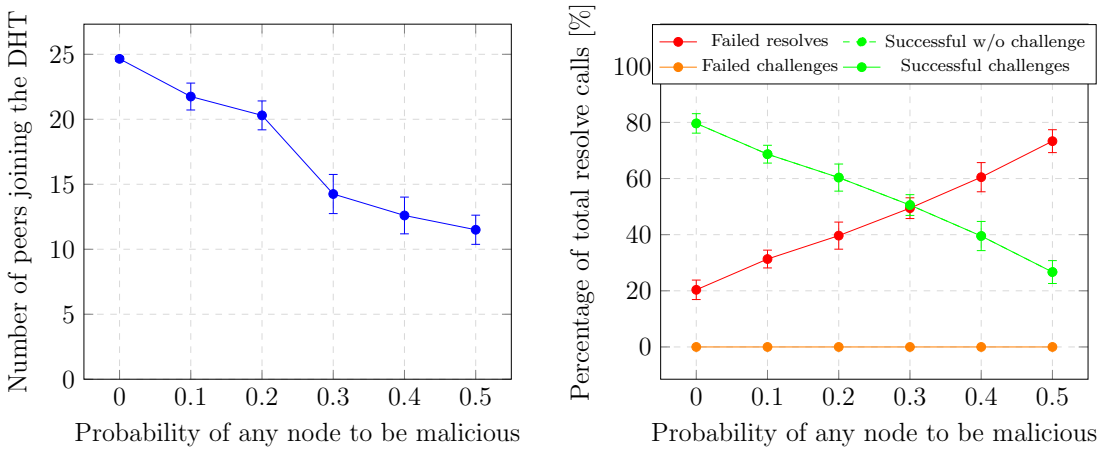
We conducted some experiments with only part of our solution implemented: the challenge mechanism. We begin with results for attacks already present in *OverSim*, for which we do not expect to observe benefits from the challenges. This is because these attacks are either DoS attacks or are detectable, in the context of P2PSIP, without any security mechanism.

Graphs for resolve calls statistics (figures 6.3b, 6.4b, 6.5 and 6.6) show four relevant statistics. The solid green line shows the success rate of resolve calls, meaning that a response was received, the challenge succeeded and communication could be established. The orange line indicates the percentage of resolve requests that yielded a properly

formatted response, but for which the challenge failed because the data had been tampered with. The dashed green line is a combination of these last two statistics and represents the percentage of resolve requests that would be considered successful without the challenge mechanism. In some cases, it overlaps with the solid green line. Finally, the red line shows the percentage of resolve requests that failed because no response was received or the response was incorrectly formatted, meaning it did not contain an IP address.

Drop Find Node and Invalid Nodes Attacks These two attacks are very similar. They both target the mechanism responsible for locating a node in the DHT. The first one does so by simply dropping the message while the second one responds with an invalid node that, in most cases, will not exist. They thus present very similar results, as shown in figures 6.3 and 6.4. In both cases, the attack affects the capacity of nodes to join the DHT, as it goes down when more malicious nodes are present. This is because the attacks affect the ability of a joining node to find its fingers. We also notice that the success rate of resolve requests goes down in the same circumstances. The challenge mechanism has no effect, as expected, because these are DoS attacks. This is why the orange line remains at 0 and the two green lines are merged.

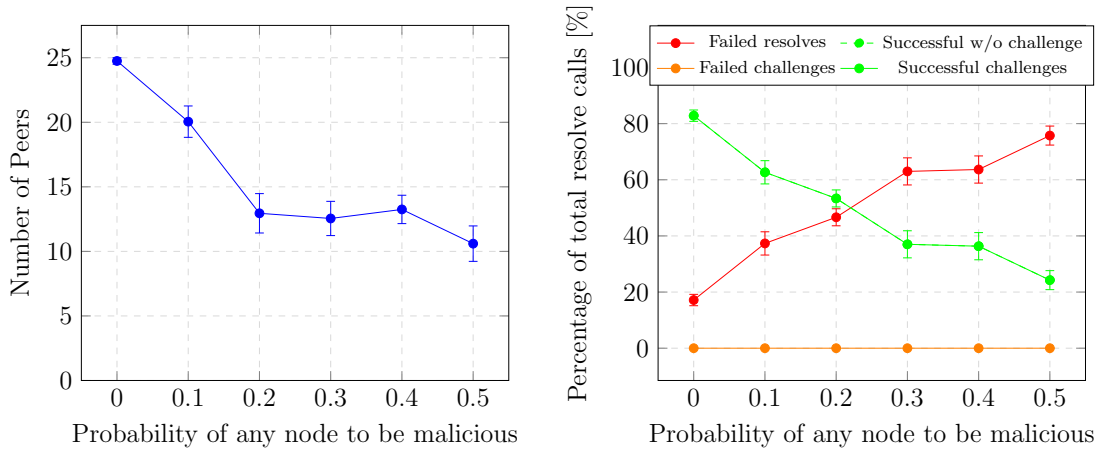
Invalid Data Attack This attack, in the general DHT case, targets data integrity. However, in a P2PSIP context, the data returned by a malicious node is not in the correct format for the expected response. This means that the attack is detected and the resolve request fails even without a security mechanism in place, turning it into a DoS attack for which the challenge mechanism is not expected to help. This is exactly what figure 6.5a shows. Resolve failures increase as the number of malicious nodes increases.



(a) Peers that Successfully Join the DHT

(b) Resolve Calls Statistics

Figure 6.3: Drop Find Node Attack Statistics

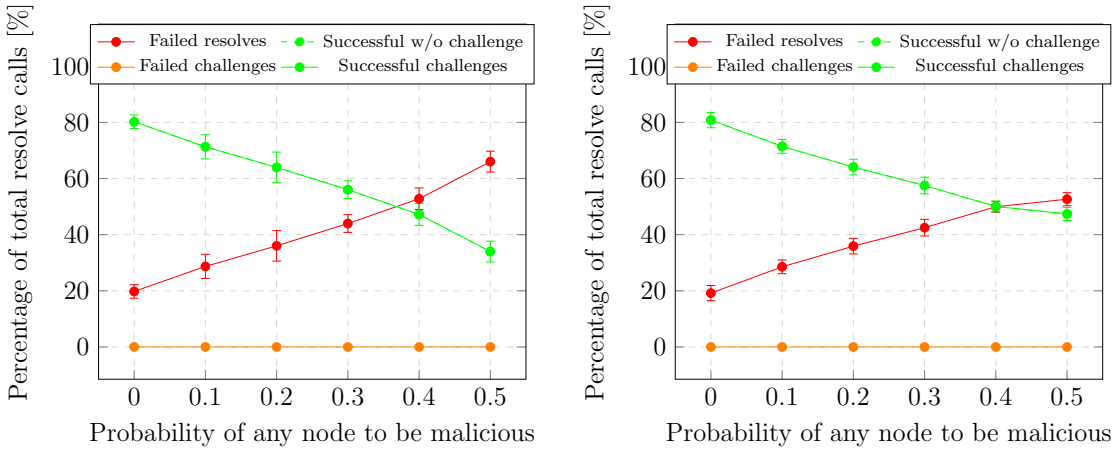


(a) Peers that Successfully Join the DHT

(b) Resolve Calls Statistics

Figure 6.4: Invalid Nodes Attack Statistics

Is Sibling Attack This attack, by itself, is a DoS attack. This means that the challenge mechanism is not expected to provide benefits and that the success rate is expected to go down with more malicious nodes. Figure 6.5b confirms that.



(a) Invalid Data Attack

(b) Is Sibling Attack

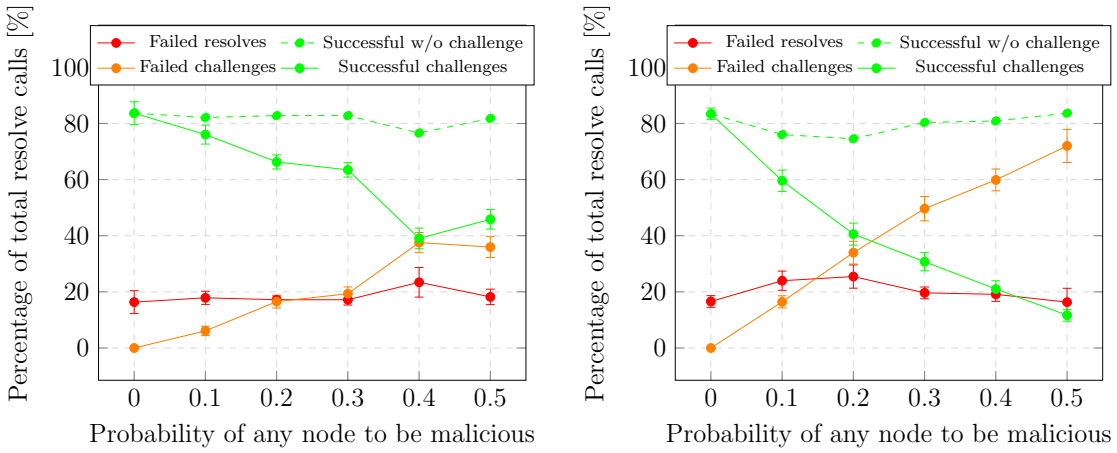
Figure 6.5: Resolve Calls Statistics

6.3.2.2 Attacks Demonstrating the Effectiveness of the Challenge Mechanism

To demonstrate the effectiveness of the challenge mechanism, we conducted some experiments with only this part of the security solution enabled. Only the *Resolve to Self Attack* is expected to have an effect because it is the only one that targets data integrity and is undetectable without a security solution in place. This attack can be amplified by combining it with the *Is Sibling Attack*, so we go over both scenarios.

Resolve to Self Attack Because the data sent by malicious nodes is in a valid format, this attack cannot be detected without implementing a security solution. If $P(y)$ is malicious, it performs the attack and tampers with the data by returning an invalid AOR resolving to its own IP address. Figure 6.6a shows this in the number of failed challenges and the difference between the amount of successful resolves with and without the challenge mechanism.

Resolve to Self Attack and Is Sibling Attack Figure 6.6b shows that this combination of attacks has the same effect as the *Resolve to Self Attack* by itself, but it is amplified. This is because, when trying to locate the node responsible for a given AOR, the first malicious node reached will resolve the query to its own IP address. If any node performing routing to $P(y)$ or $P(y)$ itself is malicious, the received response is invalid, leading to a failed challenge.



(a) Alone

(b) With Is Sibling Attack

Figure 6.6: Resolve to Self Attack

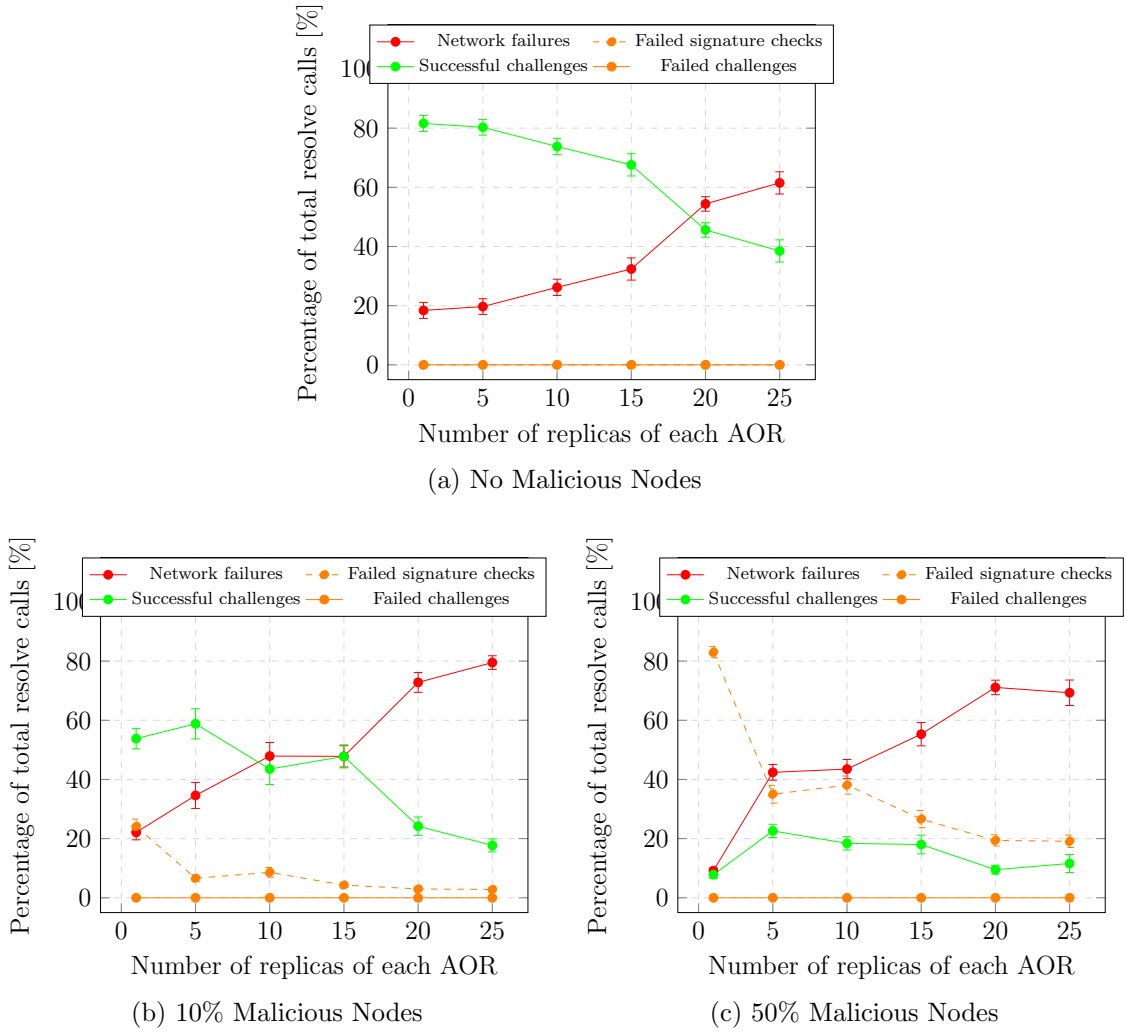
6.3.2.3 Full Security Solution

To evaluate our complete security solution for the DHT-based approach, including AOR signatures and storage redundancy in addition to the challenge mechanism, we simulated a combination of the *Resolve to Self* and *Is Sibling Attacks* with different levels of redundancy (how many replicas of each AOR) and for different probabilities of a given node being malicious. We also simulated a DHT poisoning attack.

Resolve to Self and Is Sibling Attack Figure 6.7 shows the result of the first of the aforementioned simulations. The green line shows the success rate of resolve requests while the red one is the amount of failures due to network issues or otherwise not receiving a response from the DHT. Both orange lines capture resolve failures resulting from the attack but where data integrity was preserved thanks to the defense mechanism. The dashed orange line represents cases where none of the responses received were properly signed and the solid line represents failed challenges. As explained in section 5.2.2.2, only one properly signed response is required and that, if multiple such responses are received, the most recent one according to the included timestamp will be trusted. The challenge is then sent to the contact address found in this trusted response to validate its identity and establish the connection. It is also interesting to note that, if malicious nodes performed a query-based replay attack instead of the *Resolve to Self Attack*, the two orange lines would be switched. This is because the *Resolve to Self Attack* creates improperly signed messages, causing a signature validation failure before the challenge is even sent, while a replay attack would use properly signed messages pointing to the wrong node.

The amount of network failures shown in figure 6.7, even in figure 6.7a, seems to go up quickly as more redundancy is added. This is counter-intuitive, but can be explained by the fact that only 100 seconds are reserved for the DHT to build itself before starting resolve requests and data collection. The more redundancy is used, the longer it takes for the DHT to reach a stable state.

The most important statistic to note is the rate at which resolves fail because no properly signed response was received — the dashed orange line. We note a great improvement in this regard from no redundancy to 5 replicas, but the rate of improvement slows down as more replicas are added.



(a) No Malicious Nodes

(b) 10% Malicious Nodes

(c) 50% Malicious Nodes

Figure 6.7: Statistics for Resolve to Self and Is Sibling Attacks with Different Probabilities of Each Node Being Malicious and the Full Security Solution in Place

Figure 6.8 shows the amount of traffic sent over the network during the entire simulation and per 50 successful resolve requests, for different levels of redundancy. It shows that the more redundant the network is, the more traffic is generated, including all DHT and OLSR control messages. Even without redundancy, the amount of traffic generated by the DHT-based approach is an order of magnitude higher than that of the flooding-based approach. This traffic includes DHT maintenance messages, register

messages, resolve requests and responses as well as challenges and the associated response.

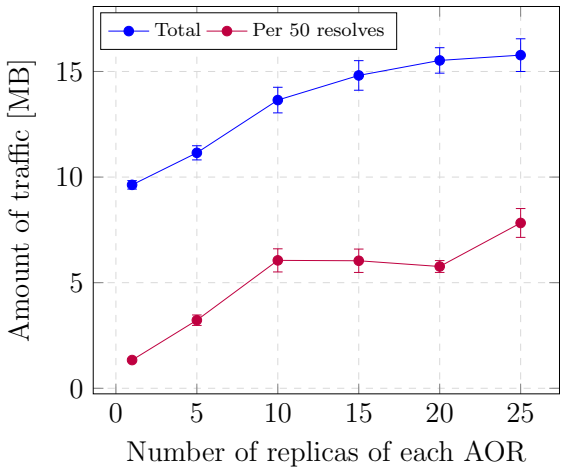
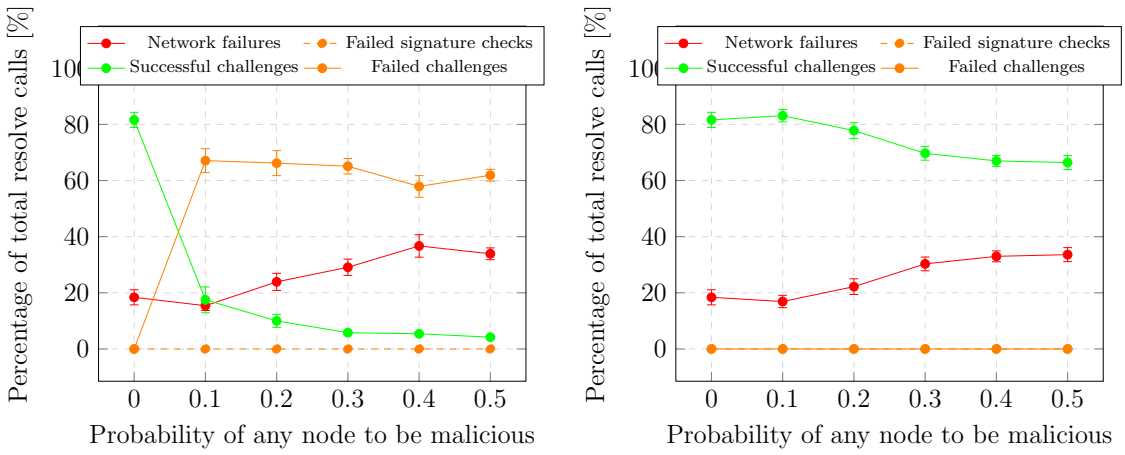


Figure 6.8: No Attack — Network Utilization

DHT Poisoning Attack Figure 6.9 shows the result of a DHT poisoning attack. This was simulated without redundancy to avoid stability issues and because redundancy would not help anyway if all replicas get poisoned. In our implementation of this attack, malicious nodes try to override all of the other nodes' AORs with their own IP address periodically, every 30 seconds.

Without signed AORs (figure 6.9a), we see a significant decrease in success rate, because valid AORs get overridden with invalid ones that malicious nodes insert. The challenge mechanism catches those and protects data integrity, if not service availability. Without challenges, data integrity would be heavily compromised. We also notice that the amount of failed challenges is relatively stable for the various proportions of malicious nodes, except when there are none, obviously. This is explained by the fact that malicious nodes try to override all valid AORs. If it's successful, even a single malicious node can compromise all of them, so more malicious nodes only create more



(a) Without Signatures

(b) With Signatures

Figure 6.9: DHT Poisoning Attack

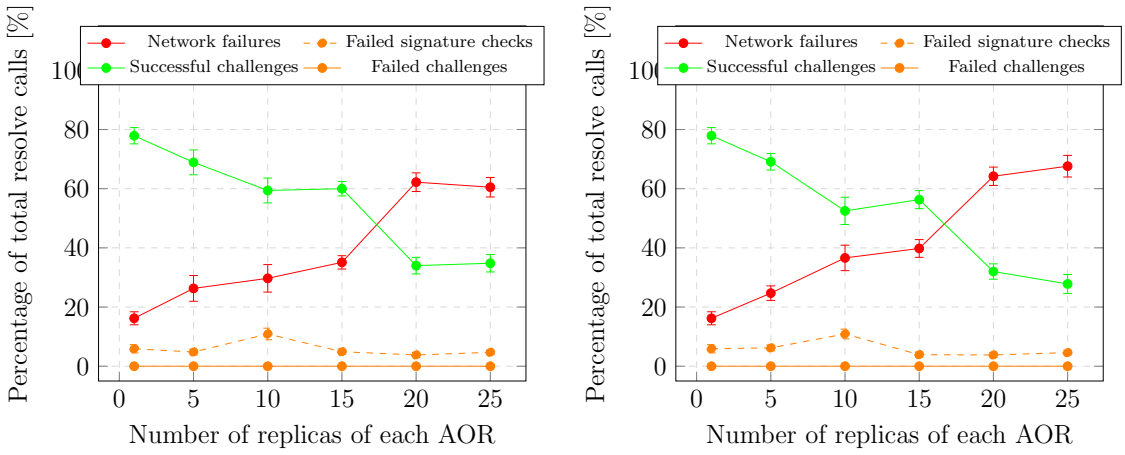
traffic.

Even with signed AORs (figure 6.9b), success rate decreases with an increase of the number of malicious nodes, albeit not nearly as significantly as without signatures. The amount of challenge failures and signature check failures remains zero though; only the number of network failures increases. This can be attributed to the extra traffic being generated by malicious nodes trying to poison the DHT. It is important to note that the signature check failures that are plotted are those done by the node performing the resolve request, on the AOR retrieved from the DHT. The signature checks that prevent this attack are those done by the storing node when receiving an insertion request. They are not shown in figure 6.9.

6.3.3 Comparison with RELOAD

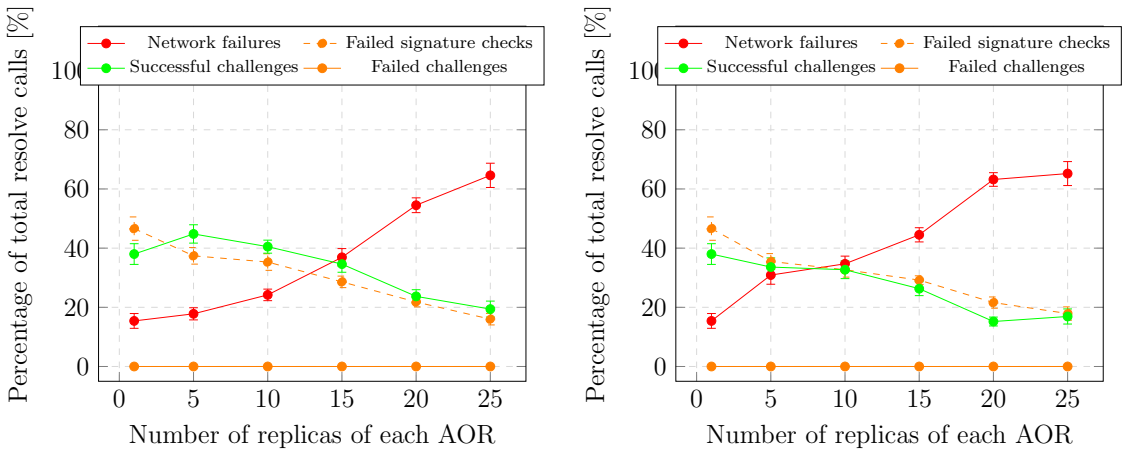
To demonstrate that our solution improves on RELOAD, we simulated the attack on redundancy affecting RELOAD described in section 4.4.2.4. Under ideal circumstances, i.e. when routing to $P(y)$ succeeds and a value can be retrieved, this attack alone

does not negatively effect the SIP service, so we combined it with the *Resolve to Self Attack*. Figures 6.10 and 6.11 show the result when each node has a 10% and 50% chance of being malicious, respectively.



(a) Our DHT Solution (b) RELOAD

Figure 6.10: Resolve to Self and Redundancy Attack against RELOAD — 10% Malicious Nodes



(a) Our DHT Solution (b) RELOAD

Figure 6.11: Resolve to Self and Redundancy Attack against RELOAD — 50% Malicious Nodes

The difference is slim, but we do notice that our solution provides a slightly higher success rate. Theory says that this attack against RELOAD, in ideal network conditions, completely defeats the point of redundancy, while our own scheme is unaffected. We cannot verify that, however, due to the instability of the network. As we just mentioned though, we do see an improvement when using our own solution.

6.4 Discussion

As expected, we have demonstrated that both of our security solutions are effective. For both solutions, we have shown that the PKI, with the signature and cryptographic challenge schemes, protect the integrity of the network and ensure that a malicious node cannot fool a legitimate one into establishing a connection with an unintended node. We have also shown that the same PKI, combined with storage and routing redundancy, helps to maintain the resilience of the network when it is subject to an attack.

For the DHT-based approach, we found that adding redundancy makes it harder for the DHT to stabilize, causing resolve request failures while the DHT has not finished building itself. The return of redundancy in terms of mitigating attacks is also great for a small number of replicas but diminishes as more are added. This means that a good balance would need to be found between DHT stability and effectiveness of the security solution in order to determine an appropriate level of redundancy for a given scenario.

We have also shown that for our configuration of 25 static nodes, maintaining a DHT brings a lot of overhead compared to flooding-based approach. Maintaining the distributed data structure, in itself, is complex and the attack surface is also higher

with the DHT, making the security solution more complex. The amount of traffic generated was also an order of magnitude higher with the DHT and the success rate was lower because of the lower stability of the network. We can conclude that, for such a small network, implementing a solution based on a DHT is not worth it compared to the much simpler approach using flooding. This may not be true for bigger networks though, as DHTs are designed to scale, with lookups in a logarithmic number of logical hops with regard to the total number of nodes, while flooding does not scale as well. It would be interesting to find the scale at which the DHT-based solution becomes more practical than the flooding-based one. Simulating larger networks requires a lot more processing power and time, however, making it impractical for us.

Finally, while the results we have shown are only for static nodes, we presume that our solutions are just as applicable to MANETs with mobile nodes. The results in terms of success rate of resolve calls would not be as good with added mobility, due to more route breakage, but our solutions themselves would work just fine. In fact, the redundancy used in our solutions, while primarily designed for scenarios where malicious nodes may be in the network, would also help reduce service availability issues when routes break. Finally, data integrity would not be affected by mobility. In both of our solutions, communications is only established if the identity callee can be properly authenticated.

6.5 Conclusion

In this chapter, we presented experimental results for both of the approaches presented in chapter 5 demonstrating their effectiveness. We also analyzed those results and hinted at some limitations of our experiments. The next chapter will summarize the

contributions of the thesis and, based on this analysis, suggest potential paths to follow future work.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

The implementation of SIP over MANETs and the security of P2P networks have both been the subject of research. Few, however, have combined both to study the security of P2PSIP over MANETs. Those who have either made little assumptions about the context in which the network is used in order to provide very general solutions or overlooked some potential issues. In any case, this led to less-than-ideal solutions for the perhaps slightly more restricted context that is of interest to us: the closed MANET, entirely controlled by a single organization.

In this thesis, we detailed this context and the entire threat model to which our P2PSIP implementation for MANETs would be subject. This included details on the characteristics of the network and the organization that controls it, the security properties that we are aiming to protect and attacks to which the network may be exposed that aim to compromise these properties.

We then presented two approaches to SIP over MANETs designed to perform

better, from a point of view of security and within the aforementioned context, than the existing solutions. One of our approaches is based on a simple local storage scheme with network flooding for SIP URI resolution; the other is based on a DHT. They both provide a strong security solution to protect both data integrity and service availability.

We finally simulated our two approaches and their respective security mechanisms, against multiple attack scenarios and presented our results. This allowed us to demonstrate, through experimentation, that our solutions are indeed effective. It also allowed us to compare both of them and to qualify their relative effectiveness for the specific network configuration that we chose of 25 static nodes. We concluded that for such a small network, the overhead of a DHT made the simpler flooding-based approach a more appropriate choice, but speculated that the roles would be reversed at a larger scale. We also directly compared our DHT-based solution to RELOAD, a similar state-of-the-art standard for P2PSIP, to demonstrate that our solution improves on its security.

7.2 Future Work

As future work, our security solutions could be tested in more diverse networks to evaluate how they both evolve under different parameters and to provide results that are closer to real life scenarios. Because our experiments were all performed with networks of 25 static nodes, this includes varying the number of nodes — increasing it, especially — and adding node mobility. This would be useful to find the kind of scale at which a DHT-based approach would be justified as opposed to using the simpler flooding-based solution.

One attack scenario that we did not experiment with through our simulations is the replay attack. Future work could include experiments with this attack, to evaluate its effect against our security solutions. This goes hand in hand with mobility, as nodes changing IP addresses is what introduces the possibility for replay attacks.

Running longer simulations would also be interesting as future work. Our results for the DHT-based solution were affected by an unstable overlay when introducing storage redundancy, which would likely be fixed by allowing more time for the DHT to build and stabilize itself. It would be interesting to confirm that this is the case and see how much better the network performs in these better circumstances.

In this thesis, as mentioned in section 3.1, we assumed a one-to-one mapping between users and devices. Relaxing this assumption in the future could provide benefits such as allowing users to replace their device, get a second one or borrow one from their colleague. The resolution mechanism would then need to be adjusted to allow storing multiple IP addresses per user and a solution would need to be developed to move the users' private keys to their new devices. The latter could most likely be accomplished in the DHT-based solution by encrypting the users' credentials with a password and storing them in the DHT.

Finally, our DHT solution is designed to work with any routing and DHT protocol. We only experimented with OLSR and Chord, but it would be very interesting to see how the performance of this solution changes when used with different protocols. In particular, using DHT protocols specifically designed for MANETs, such as those presented in section 4.2, would be extremely appropriate and would likely noticeably improve performance. The use of a hierarchical network where the routing protocol and the DHT work together in a cross-layer fashion also seems apt for this task, especially for larger networks.

Bibliography

[1] C. Jennings, B. Lowekamp, E. Rescorla, S.Baset, and H. Schulzrinne. REsource LOcation And Discovery (RELOAD) Base Protocol. RFC 6940, January 2014.

[2] Cullen Jennings, Bruce Lowekamp, Eric Rescorla, Salman Baset, Henning Schulzrinne, and Thomas C. Schmidt. A SIP Usage for REsource LOcation And Discovery (RELOAD). RFC 7904, October 2016.

[3] Alan Davoust, François Gagnon, Babak Esfandiari, Thomas Kunz, and Alexandre Cormier. Towards securing peer-to-peer sip in the manet context: Existing work and perspectives. In *Proceedings of the 10th IEEE International Conference on Cyber, Physical and Social Computing (CPSCom)*, pages 223–229. IEEE, 2017.

[4] Alexandre Cormier, François Gagnon, Babak Esfandiari, and Thomas Kunz. Toward testing security attacks and defense mechanisms for p2psip in manets with a simulator. In *Proceedings of the 14th International Joint Conference on e-Business and Telecommunications - Volume 3: DCNET, (ICETE 2017)*, pages 43–54. INSTICC, SciTePress, 2017.

[5] Silvia Giordano et al. Mobile ad hoc networks. *Handbook of wireless networks and mobile computing*, pages 325–346, 2002.

[6] Jonathan Rosenberg, Henning Schulzrinne, Gonzalo Camarillo, Alan Johnston, Jon Peterson, Robert Sparks, Mark Handley, and Eve Schooler. Sip: session initiation protocol. RFC 3261, 2002.

[7] Rüdiger Schollmeier. A definition of peer-to-peer networking for the classification of peer-to-peer architectures and applications. In *Peer-to-Peer Computing, 2001. Proceedings. First International Conference on*, pages 101–102. IEEE, 2001.

[8] Ion Stoica, Robert Morris, David Karger, M Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. *ACM SIGCOMM Computer Communication Review*, 31(4):149–160, 2001.

[9] Michael J. Freedman and Robert Morris. Tarzan: a peer-to-peer anonymizing network layer. In *Proceedings of the 9th ACM Conference on Computer and Communications Security, CCS 2002, Washington, DC, USA, November 18-22, 2002*, pages 193–206, 2002.

[10] Marc Rennhard and Bernhard Plattner. Introducing morphmix: peer-to-peer based anonymous internet usage with collusion detection. In *Proceedings of the 2002 ACM Workshop on Privacy in the Electronic Society, WPES 2002, Washington, DC, USA, November 21, 2002*, pages 91–102, 2002.

[11] Qiyan Wang and Nikita Borisov. Octopus: A secure and anonymous dht lookup. In *Distributed Computing Systems (ICDCS), 2012 IEEE 32nd International Conference on*, pages 325–334. IEEE, 2012.

[12] Maarten Fonville. Confidential peer-to-peer file-sharing using social-network sites. In *13th Twente Student Conference on IT, Jun*, volume 21, page 10, 2010.

[13] Guido Urdaneta, Guillaume Pierre, and Maarten Van Steen. A survey of dht security techniques. *ACM Computing Surveys (CSUR)*, 43(2):8, 2011.

[14] John R. Douceur. The sybil attack. In *Peer-to-Peer Systems, First International Workshop, IPTPS 2002, Cambridge, MA, USA, March 7-8, 2002, Revised Papers*, pages 251–260, 2002.

[15] Hatem Ismail, Daniel Germanus, and Neeraj Suri. Detecting and mitigating P2P eclipse attacks. In *21st IEEE International Conference on Parallel and Distributed Systems, ICPADS 2015, Melbourne, Australia, December 14-17, 2015*, pages 224–231, 2015.

[16] Patrick Stuedi. *From Theory to Practice: Fundamental Properties and Services of Mobile Ad Hoc Networks*. PhD thesis, ETH Zurich, 2008.

[17] Li Li and Louise Lamont. Support real-time interactive session applications over a tactical mobile ad hoc network. In *Military Communications Conference, 2005. MILCOM 2005. IEEE*, pages 2910–2916. IEEE, 2005.

[18] Nilanjan Banerjee, Arup Acharya, and Sajal K Das. Peer-to-peer SIP-based services over wireless ad hoc networks. In *BROADWIM: Broadband Wireless Multimedia Workshop*, 2004.

[19] Luis Villasenor-Gonzalez, Ying Ge, and Louise Lamont. HOLSR: a hierarchical proactive routing mechanism for mobile ad hoc networks. *Communications Magazine, IEEE*, 43(7):118–125, 2005.

[20] S. Yahiaoui, Y. Belhoul, N. Nouali-Taboudjemat, and H. Kheddouci. AdSIP: Decentralized SIP for mobile ad hoc networks. In *Advanced Information Networking*

and Applications Workshops (WAINA), 2012 26th International Conference on, pages 490–495, March 2012.

[21] Nilanjan Banerjee, Arup Acharya, and Sajal K Das. Enabling SIP-based session setup in ad hoc networks. In *Proceedings of INFOCOM*, 2005.

[22] S. Fudickar, K. Rebensburg, and B. Schnor. MANETSip - a dependable SIP overlay network for MANET including presentity service. In *Networking and Services, 2009. ICNS '09. Fifth International Conference on*, pages 314–319, April 2009.

[23] Ala' Aburumman, Wei Jye Seo, Christian Esposito, Aniello Castiglione, Rafiqul Islam, et al. A secure and resilient cross-domain sip solution for manets using dynamic clustering and joint spatial and temporal redundancy. *Concurrency and Computation: Practice and Experience*, 2016.

[24] Thirapon Wongsaardsakul. *P2P SIP over mobile ad hoc networks*. PhD thesis, Evry, Institut national des télécommunications, 2010.

[25] Aisling O'Driscoll, Susan Rea, and Dirk Pesch. Hierarchical clustering as an approach for supporting P2P SIP sessions in ubiquitous environments. In *9th IFIP International Conference on Mobile Wireless Communications Networks, MWCN 2007, Cork, Ireland, 19-21 September, 2007*, pages 76–80. IEEE, 2007.

[26] David A Bryan, Bruce B Lowekamp, and Marcia Zangrilli. The design of a versatile, secure p2psip communications architecture for the public internet. In *Parallel and Distributed Processing, 2008. IPDPS 2008. IEEE International Symposium on*, pages 1–8. IEEE, 2008.

[27] Jan Seedorf. Using cryptographically generated sip-uris to protect the integrity of content in p2p-sip. In *Third Annual VoIP Security Workshop*, 2006.

[28] Ingmar Baumgart. P2pns: A secure distributed name service for p2psip. In *Pervasive Computing and Communications, 2008. PerCom 2008. Sixth Annual IEEE International Conference on*, pages 480–485. IEEE, 2008.

[29] Petar Maymounkov and David Mazieres. Kademlia: A peer-to-peer information system based on the xor metric. In *International Workshop on Peer-to-Peer Systems*, pages 53–65. Springer, 2002.

[30] F. Delmastro. From pastry to crossroad: Cross-layer ring overlay for ad hoc networks. In *Pervasive Computing and Communications Workshops, 2005. PerCom 2005 Workshops. Third IEEE International Conference on*, pages 60–64, March 2005.

[31] Mohammad Al Mojamed and Mario Kolberg. Design and evaluation of a peer-to-peer manet crosslayer approach: Onehopoverlay4manet. *Peer-to-Peer Networking and Applications*, 10(1):138–155, 2017.

[32] Patrick Stuedi, Marcel Bihr, Alain Remund, and Gustavo Alonso. SIPHoc: Efficient SIP middleware for ad hoc networks. In Renato Cerqueira and Roy H. Campbell, editors, *Middleware 2007, ACM/IFIP/USENIX 8th International Middleware Conference, Newport Beach, CA, USA, November 26-30, 2007, Proceedings*, volume 4834 of *Lecture Notes in Computer Science*, pages 60–79. Springer, 2007.

[33] Thomas Zahn and Jochen Schiller. MADPastry: A DHT substrate for practicably sized MANETs. In *Proc. of ASWN*, 2005.

[34] Antony Rowstron and Peter Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *IFIP/ACM International Conference on Distributed Systems Platforms and Open Distributed Processing*, pages 329–350. Springer, 2001.

[35] Sylvia Ratnasamy, Mark Handley, Richard Karp, and Scott Shenker. Topologically-aware overlay construction and server selection. In *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 3, pages 1190–1199. IEEE, 2002.

[36] Rolf Winter, Thomas Zahn, and Jochen Schiller. Random landmarking in mobile, topology-aware peer-to-peer networks. In *Distributed Computing Systems, 2004. FTDCS 2004. Proceedings. 10th IEEE International Workshop on Future Trends of*, pages 319–324. IEEE, 2004.

[37] Matthew Caesar, Miguel Castro, Edmund B. Nightingale, Greg O'Shea, and Antony Rowstron. Virtual ring routing: Network routing inspired by dhts. In *Proceedings of the 2006 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, SIGCOMM '06, pages 351–362, New York, NY, USA, 2006. ACM.

[38] Filipe Araujo, Luís Rodrigues, Jörg Kaiser, Changling Liu, and Carlos Mitidieri. CHR: a distributed hash table for wireless ad hoc networks. In *Distributed Computing Systems Workshops, 2005. 25th IEEE International Conference on*, pages 407–413. IEEE, 2005.

[39] Miguel Castro, Peter Druschel, Ayalvadi Ganesh, Antony Rowstron, and Dan S Wallach. Secure routing for structured peer-to-peer overlay networks. *ACM SIGOPS Operating Systems Review*, 36(SI):299–314, 2002.

[40] John R Douceur. The sybil attack. In *International Workshop on Peer-to-Peer Systems*, pages 251–260. Springer, 2002.

[41] Jose L Muñoz, Jordi Forne, Oscar Esparza, and Miguel Soriano. Certificate revocation system implementation based on the merkle hash tree. *International Journal of Information Security*, 2(2):110–124, 2004.

[42] Matei Ciobanu Morogan and Sead Muftic. Certificate revocation system based on peer-to-peer crl distribution. In *Proc. of the DMS'03 Conference*, 2003.

[43] Chu Yee Liau, Stéphane Bressan, Kian-Lee Tan, Chu Yee, L Stphane, and BK lee Tan. Efficient certificate revocation: A p2p approach. In *ASIAN 2002 Workshop on Southeast Asian Computing Research*, 2002.

[44] Jochen Dinger and Hannes Hartenstein. Defending the sybil attack in p2p networks: Taxonomy, challenges, and a proposal for self-registration. In *First International Conference on Availability, Reliability and Security (ARES'06)*, pages 8–pp. IEEE, 2006.

[45] Arvid Norberg. DHT Security. URL: `http://blog.libtorrent.org/2012/12/dht-security`, December 2012.

[46] Rida A Bazzi and Goran Konjevod. On the establishment of distinct identities in overlay networks. *Distributed Computing*, 19(4):267–287, 2007.

[47] Honghao Wang, Yingwu Zhu, and Yiming Hu. An efficient and secure peer-to-peer overlay network. In *The IEEE Conference on Local Computer Networks 30th Anniversary (LCN'05) l*, pages 8–pp. IEEE, 2005.

[48] Rida A Bazzi, Young-ri Choi, and Mohamed G Gouda. Hop chains: secure routing and the establishment of distinct identities. In *International Conference On Principles Of Distributed Systems*, pages 365–379. Springer, 2006.

[49] Mahdi N Al-Ameen and Matthew Wright. Persea: a sybil-resistant social dht. In *Proceedings of the third ACM conference on Data and application security and privacy*, pages 169–172. ACM, 2013.

[50] Paul Syverson, Gene Tsudik, Michael Reed, and Carl Landwehr. Towards an analysis of onion routing security. In *Designing Privacy Enhancing Technologies*, pages 96–114. Springer, 2001.

[51] Peng Wang, Ivan Osipkov, N Hopper, and Yongdae Kim. Myrmic: Secure and robust dht routing. *U. of Minnesota, Tech. Rep*, 2006.

[52] Kirsten Hildrum and John Kubiatowicz. Asymptotically efficient approaches to fault-tolerance in peer-to-peer networks. In *International Symposium on Distributed Computing*, pages 321–336. Springer, 2003.

[53] Amos Fiat, Jared Saia, and Maxwell Young. Making chord robust to byzantine attacks. In *European Symposium on Algorithms*, pages 803–814. Springer, 2005.

[54] Moni Naor and Udi Wieder. A simple fault tolerant distributed hash table. In *International Workshop on Peer-to-Peer Systems*, pages 88–97. Springer, 2003.

[55] Ben Y Zhao, Ling Huang, Jeremy Stribling, Sean C Rhea, Anthony D Joseph, and John D Kubiatowicz. Tapestry: A resilient global-scale overlay for service deployment. *IEEE Journal on selected areas in communications*, 22(1):41–53, 2004.

[56] Cyrus Harvesf and Douglas M Blough. The effect of replica placement on routing robustness in distributed hash tables. In *Sixth IEEE International Conference on Peer-to-Peer Computing (P2P'06)*, pages 57–6. IEEE, 2006.

[57] Amos Fiat and Jared Saia. Censorship resistant peer-to-peer networks. *Theory of Computing*, 3(1):1–23, 2007.

[58] Baruch Awerbuch and Christian Scheideler. Towards a scalable and robust dht. *Theory of Computing Systems*, 45(2):234–260, 2009.

[59] Frank Dabek, Jinyang Li, Emil Sit, James Robertson, M Frans Kaashoek, and Robert Morris. Designing a dht for low latency and high throughput. In *NSDI*, volume 4, pages 85–98, 2004.

[60] Bryan N Mills and Taieb F Znati. Scar-scattering, concealing and recovering data within a dht. In *41st Annual Simulation Symposium (anss-41 2008)*, pages 35–42. IEEE, 2008.

[61] Rodrigo Rodrigues and Barbara Liskov. High availability in dhts: Erasure coding vs. replication. In *International Workshop on Peer-to-Peer Systems*, pages 226–239. Springer, 2005.

[62] Marc Sanchez Artigas, Pedro Garcia Lopez, and Antonio F Gomez Skarmeta. A novel methodology for constructing secure multipath overlays. *IEEE Internet Computing*, 9(6):50–57, 2005.

[63] Thomas Clausen and Philippe Jacquet. Optimized link state routing protocol (olsr). RFC 3626, 2003.