Approaches to Securing P2PSIP in MANETs

Alexandre Cormier[1], François Gagnon[2(✉)], Babak Esfandiari[1], and Thomas Kunz[1]

[1] Department of Systems and Computer Engineering, Carleton University, Ottawa, Canada
{alexandrecormier,babak,tkunz}@sce.carleton.ca
[2] Cybersecurity Research Lab, Cégep Sainte-Foy, Québec City, Québec, Canada
frgagnon@cegep-ste-foy.qc.ca

Abstract. This paper studies the security for Voice over IP in peer-to-peer (P2P) networks. Instead of taking a general approach to security in P2P, we focus on a specific use case, namely private (e.g. military) mobile ad hoc networks. This allows for security measures that are not necessarily applicable to general P2P networks, but elegantly solve the issues in the given context. We propose security measures for two different approaches to the P2P version of the Session Initiation Protocol in such networks, provide their implementations and present results from performing experimentations in a simulator.

Keywords: P2P · SIP · MANET · Security
Attacks and simulation experiments

1 Introduction

Telecommunication in areas without access to the Internet infrastructure can be enabled using mobile ad hoc networks (MANET). Securing such communication against hostile interference is a requirement in many domains, such as military or law enforcement. Voice over IP (VoIP) is a convenient means for such communication and, because VoIP connections are usually established using the Session Initiation Protocol (SIP), this paper focuses on securing peer-to-peer SIP (P2PSIP) in MANETs.

In [1], we presented our preliminary work toward securing a MANET used in such a scenario. This involved presenting a threat model for peer-to-peer SIP (P2PSIP) over MANETs, proposing a security solution for an approach to P2PSIP based on a distributed hash table (DHT) and providing experimental results for a partial implementation of this security solution, in *Omnet++*[1] using the P2P simulation framework *OverSim*[2]. In this paper, we extend this work with:

[1] https://omnetpp.org/ [2].
[2] http://www.oversim.org [3].

M. S. Obaidat and E. Cabello (Eds.): ICETE 2017, CCIS 990, pp. 38–62, 2019.
https://doi.org/10.1007/978-3-030-11039-0_3

– Experimental results for full implementation of the DHT-based solution.
– A second security solution, built upon a flooding-based approach to P2PSIP.
– Experimental results for this new solution, also in *Omnet++*.

The remainder of this paper is organized as follows. Section 2 covers some background about MANETs, SIP and P2P. Section 3 then presents related work from literature before Sect. 4 describes in detail the two approaches to SIP over MANETs which our work is based on. Section 5 presents the threat model for our context, then Sects. 6 and 7 present our two security solutions, respectively built upon each of the two aforementioned approaches. Section 8 details experiments performed with these security mechanisms in a simulator and their results. Finally, Sect. 9 closes with a short summary of our findings and discussion of future work.

2 Background

The key concepts used throughout this paper 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 [4] 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.

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 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.

2.2 Session Initiation Protocol

SIP [5] is an IETF-standardized protocol used to initiate a session between two users' devices. The entire 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 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 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.

2.3 P2P

As explained in Sect. 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 distributed hash table (DHT), to store and retrieve AORs. The RELOAD protocol [6], 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's 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.

3 Related Work

Implementing SIP over MANETs is not a new idea, and so a few solutions have been proposed. These can be grouped into distinct categories following the registration scheme used to replace SIP servers, as follows.

Local Storage. TacMAN [8] and a proposal by Banerjee et al. identified as "Loosely Coupled" [9] 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.

Network Subset Storage. AdSIP [10], another approach by Banerjee et al. identified as "Tightly Coupled" [11], MANETSip [12] and a solution by Aburumman et al. [13] instead stay closer to traditional SIP and select a subset of the network's peers to act as registrars. For the first two of those, in particular, the peers are chosen such that they form a dominating set of the network graph, meaning that all nodes have at least one neighbor in this subset and all lookups are at most one hop.

DHT Storage. Two unnamed proposals, by Wongsaardsakul et al. [14] and O'Driscoll et al. [15], implement a DHT over the MANET, in which AORs are stored in order to replace registrars. As mentioned in Sect. 2.3, this is also the approach used in the RELOAD protocol [6], standardized by the IETF.

For the most part, however, these solutions do not focus on security. RELOAD is the notable exception, with its three-level security model based on a central certificate authority (CA). TLS or an equivalent protocol is used for all communication between nodes, all messages are signed and stored objects are also signed by the node that creates them.

A few additional solutions also put the focus on securing a P2PSIP network: P2PNS [16] and two unnamed proposals, by Bryan et al. [17] and Seedorf [18]. The first two rely on public key cryptography to sign overlay messages. P2PNS additionally signs registration messages, while the solution by Bryan et al. involves signing the registrations themselves. Both rely on rate limiting mechanisms to make it hard for an attacker to get a valid key pair, rather than using a central CA. For the former, this mechanism is crypto-puzzles. Seedorf, on the other hand, proposes to use 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, instead of being a meaningful name like the user's first and last names, is then replaced with a hash of the node's public key. Finally, registrations are signed with the corresponding private key. A more thorough examination of security solutions for P2PSIP, with a focus on DHT security, is presented in [19].

To summarize, many proposals for SIP over MANETs simply do not consider security. Those that do, for the most part, make little assumptions about the context in which they will be used. This allows them to provide a general solution, suitable to many contexts. However, it means that these solutions are not tailored to any specific context, such as the context under focus for this paper (see Sect. 5). Consequently, it makes it more difficult or even impossible to provide the stricter security guarantees that more context awareness can provide.

4 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 Sect. 3. The first one, described in Sect. 4.1, 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 and is detailed in Sect. 4.2.

4.1 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 Sects. 5 and 6 will show that it has some benefits, security-wise.

In this approach, 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 Sect. 6.

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 can be 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 can be 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.

4.2 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 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. Our focus is on private networks, controlled and operated by a given entity (e.g. the army) that can assert who should and who should not be part of it. SIP URIs can thus be authorized by this central authority prior to the deployment of the network.

In this context, we are interested in protecting data integrity and service availability. The former means an attacker should not be able to fool a legitimate node with regard to the current location of another node, while the latter means an attacker should also not be able to prevent a legitimate node from locating another node.

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. We consider that each malicious node is in full control of one legitimate device and that all malicious nodes can collude and communicate together via a dedicated channel.

We do not consider confidentiality nor anonymity, but they have been studied in P2P literature. For example, Tarzan [20], MorphMix [21] and Octopus [22] study confidentiality and [23] additionally considers anonymity. Our previous work in [1] also describes the entire context in greater depth.

Section 5.1 formally defines and extends the notation used in Sect. 4 and throughout this paper. Then, we present attacks on which the attacker can rely in Sects. 5.2 and 5.3, focusing on storage and retrieval attacks. Again, our previous work in [1], as well as [19], describe these in a more detailed fashion, along with impersonation attacks.

5.1 Notation

The notation used throughout this paper is that of [1], which is as follows:

- Q and R stand for legitimate nodes.
- x and y stand for legitimate users.
- Q and x denote an entity querying the SIP service.
- R and y denote an entity to communicate with.
- M stands for a malicious node.
- S stands for a generic node.

– 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.

5.2 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.

5.3 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 *OverSim*. 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 Fig. 1 and sets the table for a query-based attack, which would allow M to compromise data integrity rather than only availability.

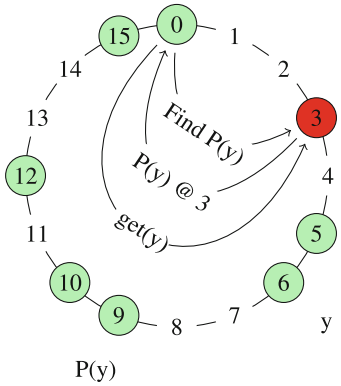


Fig. 1. Is Sibling Attack.

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 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 Fig. 2.

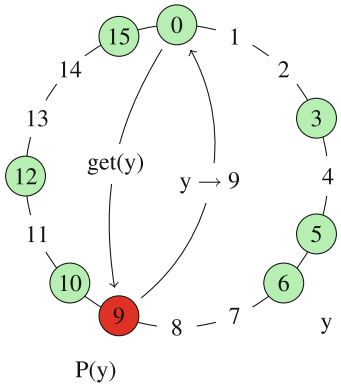


Fig. 2. Resolve to Self Attack.

DHT poisoning, shown in Fig. 3, 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.

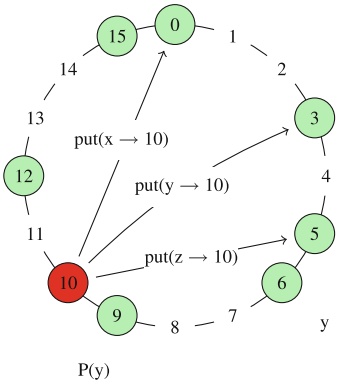


Fig. 3. DHT poisoning.

5.4 Replay Attack

A class of attack that is common to both the flooding and DHT scenarios 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 query and also, in the DHT context, at insertion time.

6 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 Sect. 4.1. At the heart of this security solution is an offline public key infrastructure (PKI). We first tackle the data integrity issue in Sect. 6.1 before discussing the system's resilience in Sect. 6.2.

6.1 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 Sect. 5.2.

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 Sect. 4.1, 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 certificate authority (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 Fig. 4 and, in the context in which we operate, described in Sect. 5, this is not a major overhead as a central authority figure owns and is in control of the network and all these nodes correspond to devices that will have to be prepared for missions before being deployed anyway.

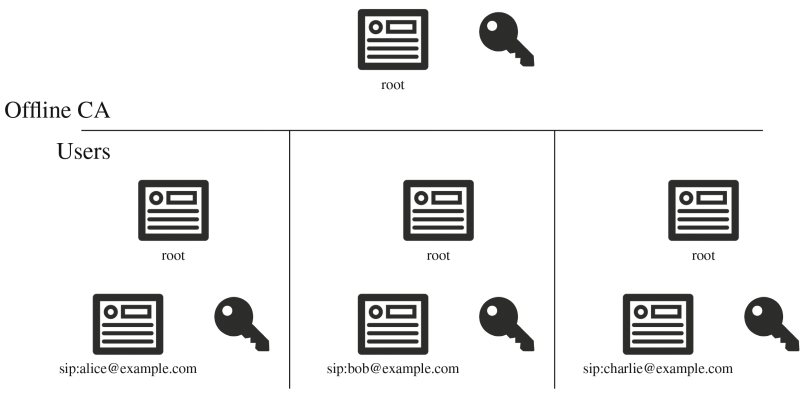


Fig. 4. PKI setup.

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.

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.

6.2 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. 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 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 Sect. 6.1, but the same mechanism can be applied to requests.

7 Securing DHT for P2PSIP

This section presents our multi-layer approach to securing the DHT-based SIP resolution mechanism described in Sect. 4.2. 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.

7.1 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 Sect. 5.3, 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 Sect. 5.3 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 Sect. 6.1. 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 signing 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 transfered 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.

7.2 Resilience

With the challenge mechanism described in Sect. 7.1, 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 resource exhaustion attacks, followed by replay attacks and finally routing- and query-based attacks.

Preventing DHT Poisoning and Resource 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 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 resource 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.

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 targetting 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.

8 Experiments

In this section, we describe our experiments and results, starting with our methodology in Sect. 8.1, followed by the flooding-based solution in Sect. 8.2 and the DHT-based solution in Sect. 8.3. Then, in Sect. 8.4, we discuss the results of these experiments.

8.1 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 to 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 s, to resolve a SIP URI among those of all nodes. This process starts after 100 s, 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.

8.2 Flooding-Based Solution

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

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 ms and follows a uniform distribution.

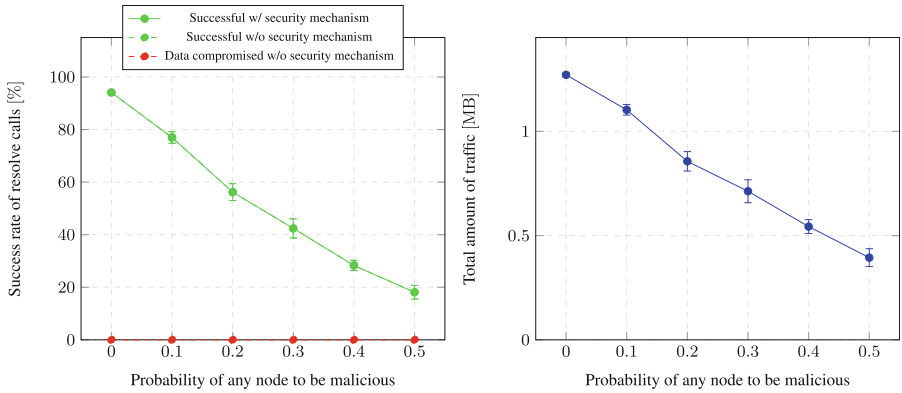
The defense mechanism presented in Sect. 6 has been implemented, along with the *Drop Messages*, *Resolve to Self* and *Edit Responses* attacks discussed in Sect. 5.2.

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.

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 5a 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, it 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 5b 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 Sect. 8.3.



(a) Resolve Calls Statistics **(b)** Network Utilization

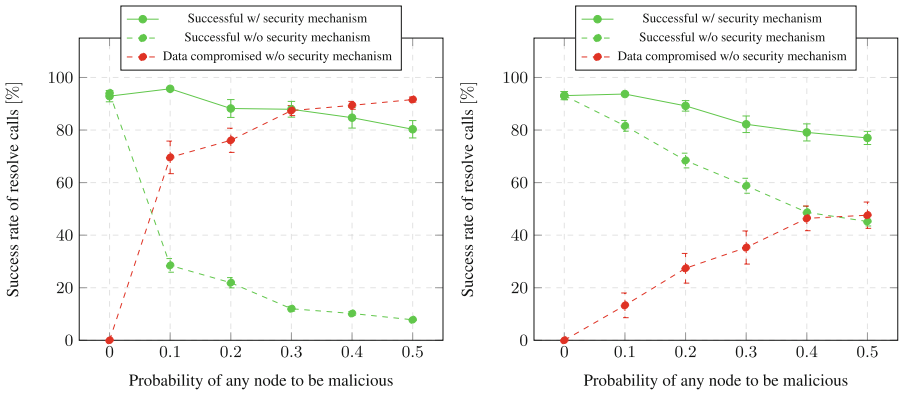
Fig. 5. Drop Messages Attack statistics.

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 6a 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 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. This is shown with dashed lines in Fig. 6a, where the red one represents cases in which the session would be established with the wrong, malicious node.

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 6b 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 packet detection mechanism.

8.3 DHT-Based Solution

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



(a) Resolve to Self Attack **(b)** Edit Responses Attack

Fig. 6. Resolve calls statistics.

Methodology. These experiments were run using a Chord DHT [24] over an OLSR network [25]. 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 Sect. 7 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 Sect. 5.3, was also implemented.

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.

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.

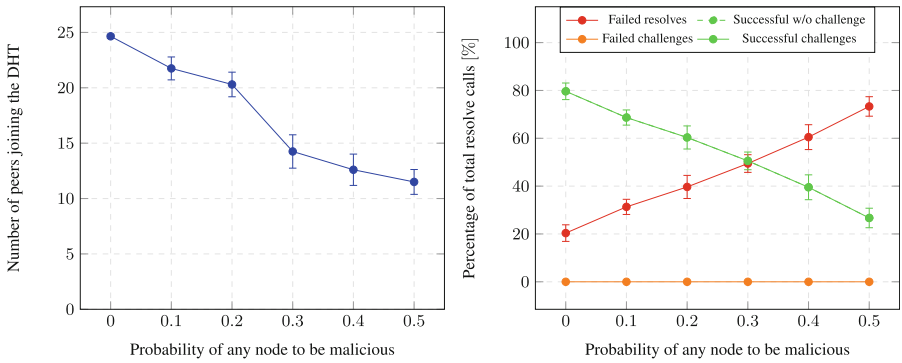
Graphs for resolve calls statistics show four relevant statistics. The solid green line shows the success rate of resolve calls, meaning that a response was received, the challenge succeeded and communication could be established. The orange line indicates the percentage of resolve requests that yielded 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 Figs. 7 and 8. In both cases, the attack affects the capacity of nodes to join the DHT, as it goes down when more malicious nodes are present. 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.

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 Fig. 9a shows. Resolve failures increase as the number of malicious nodes increases.

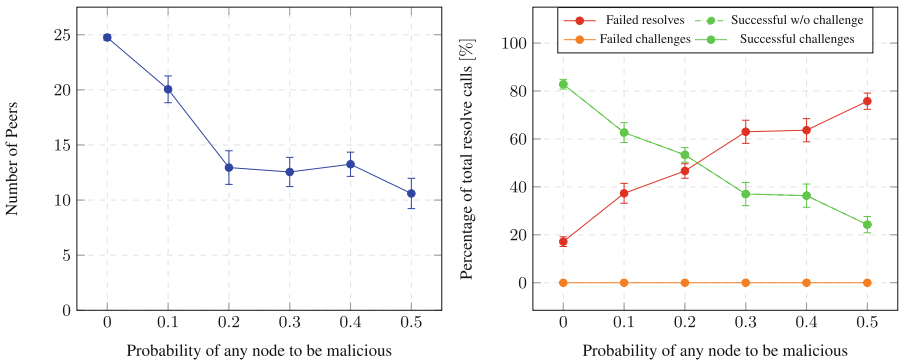
Is Sibling Attack. This attack, by itself, is a DoS attack. This means that the challenge mechanism is not expected to provide benefits and that the success rate is expected to go down with more malicious nodes. Figure 9b confirms that.



(a) Peers that Successfully Join the DHT **(b)** Resolve Calls Statistics

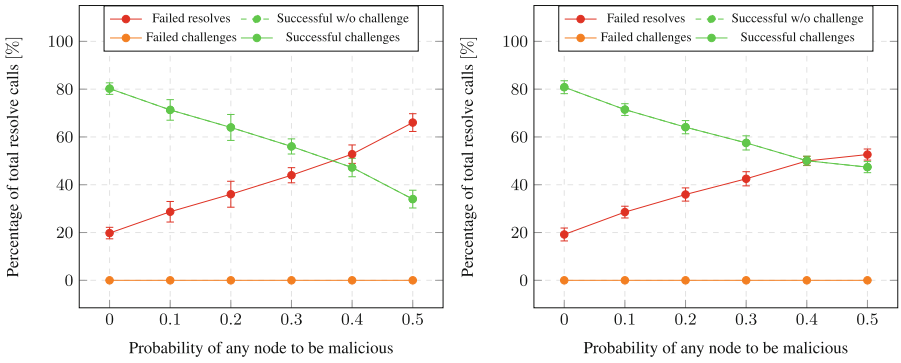
Fig. 7. Drop Find Node Attack statistics [1].

Attacks Demonstrating the Effectiveness of the Challenge Mechanism. To demonstrate the effectiveness of the challenge mechanism, we conducted some



(a) Peers that Successfully Join the DHT **(b)** Resolve Calls Statistics

Fig. 8. Invalid Nodes Attack statistics [1].



(a) Invalid Data Attack **(b)** Is Sibling Attack

Fig. 9. Resolve calls statistics [1].

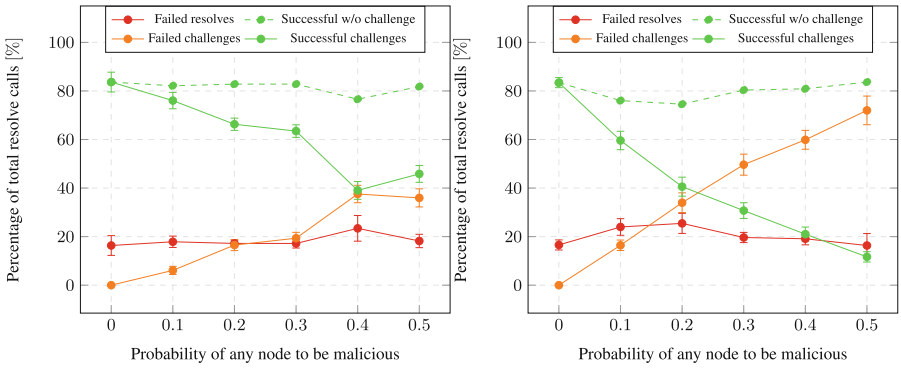


Fig. 10. Resolve to Self Attack alone (left) and with the Is Sibling Attack (right) [1].

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 (Fig. 10).

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. Figure 9 shows this in the number of failed challenges and the difference between the amount of successful resolves with and without the challenge mechanism.

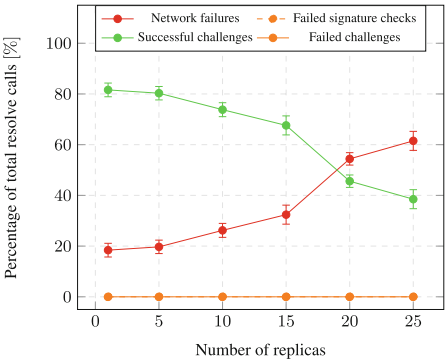
Resolve to Self Attack and Is Sibling Attack. Figure 9 shows that this combination of attacks has the same effect as the *Resolve to Self Attack* by itself, but it is amplified. This is because, when trying to locate the node responsible for a given AOR, the first malicious node reached will resolve the query to its own IP address.

Full Security Solution. To evaluate our complete security solution for the DHT-based approach, including AOR signatures and 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 and for different probabilities of a given node being malicious.

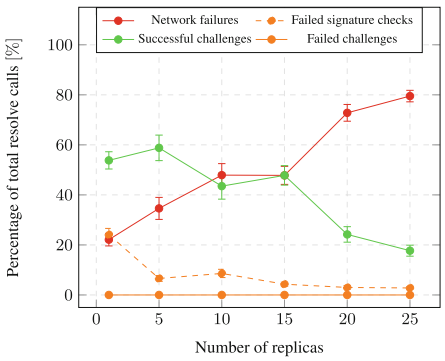
Figure 11 shows the result. The green line shows the success rate of resolve requests while the red one is the amount of failures due to network issues or otherwise not receiving a response. Both orange lines capture resolve failures resulting from the attack but where data integrity was preserved thanks 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 Sect. 7.2, 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 Fig. 11, even in Fig. 11a, 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.

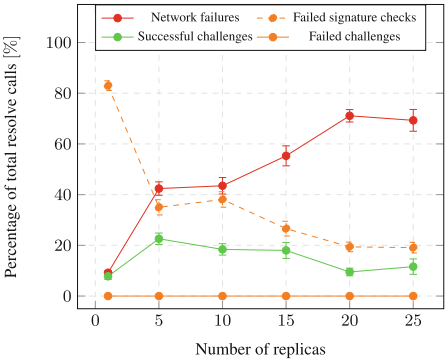
The most important statistic to note is the rate at which resolves fail because no properly signed response was received—the dashed orange line. We note a great improvement in this regard from no redundancy to 5 replicas, but the rate of improvement slows down as more replicas are added.



(a) No Malicious Nodes



(b) 10% Malicious Nodes



(c) 50% Malicious Nodes

Fig. 11. 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 12 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.

8.4 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

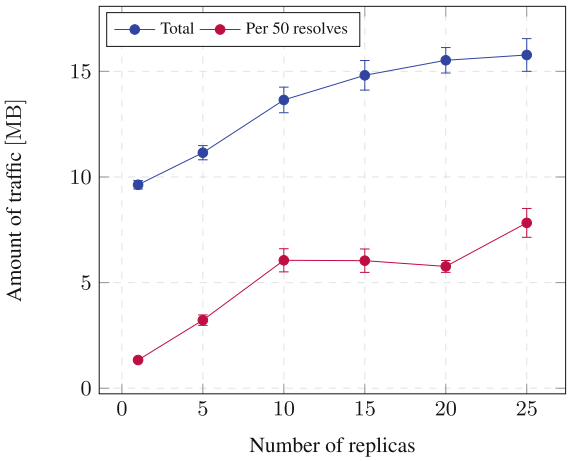


Fig. 12. No attack—network utilization.

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 also. 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.

9 Conclusion

The implementation of SIP over MANETs and the security of P2P networks have both been the subject of research. Few have combined both to study the security of P2PSIP over MANETs and those who have made little assumptions about the context in which the network is used in order to provide very general

solutions. In this paper, we considered two approaches to SIP over MANETs—one based on network flooding, the other on a DHT—and detailed the threat model to which both are subject. By making assumptions that are reasonable for a military MANET, we were able to provide a strong security solution to protect both data integrity and service availability in each of these two approaches. We then simulated the two approaches, with their respective security mechanisms, against multiple attack scenarios. This allowed us to show that both solutions are effective and to quantify their effectiveness relative to each other for a specific network configuration of 25 static nodes.

We plan, as future work, to expand this comparisons to more diverse networks, varying the number of nodes and adding mobility, to evaluate how both security solutions evolve under different parameters and to provide results that are closer to a real life scenario. This will allow us to discover at what scale a DHT-based approach may be justified. We will also run simulations with longer time periods before the start of resolve requests, to see how allowing more time for the DHT to build and stabilize itself improves its effectiveness. We will also implement more attack scenarios, for example the DHT poisoning attack and replay attacks, and evaluate their effect against our security solutions. Finally, we intend on comparing both of our solutions to existing ones to examine their differences and how it effects the network. A comparison with RELOAD, in particular, will be interesting due to its similarity to our DHT-based solution.

Acknowledgment. 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.

References

1. Cormier, A., Gagnon, F., Esfandiari, B., Kunz, T.: 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), INSTICC, pp. 43–54. SciTePress (2017)
2. Varga, A., Hornig, R.: An overview of the OMNeT++ simulation environment. In: Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), p. 60 (2008)
3. Baumgart, I., Heep, B., Krause, S.: OverSim: a flexible overlay network simulation framework. In: Proceedings of 10th IEEE Global Internet Symposium (GI 2007) in Conjunction with IEEE INFOCOM 2007, Anchorage, AK, USA, pp. 79–84 (2007)
4. Giordano, S., et al.: Mobile ad hoc networks. In: Handbook of Wireless Networks and Mobile Computing, pp. 325–346 (2002)

5. Rosenberg, J., et al.: SIP: Session Initiation Protocol. RFC 3261 (2002)
6. Jennings, C., Lowekamp, B., Rescorla, E., Baset, S., Schulzrinne, H.: REsource LOcation and Discovery (RELOAD) Base Protocol. RFC 6940 (2014)
7. Schollmeier, R.: A definition of peer-to-peer networking for the classification of peer-to-peer architectures and applications. In: First International Conference on Peer-to-Peer Computing, Proceedings, pp. 101–102. IEEE (2001)
8. Li, L., Lamont, L.: Support real-time interactive session applications over a tactical mobile ad hoc network. In: Military Communications Conference, MILCOM 2005, pp. 2910–2916. IEEE (2005)
9. Banerjee, N., Acharya, A., Das, S.K.: Peer-to-peer SIP-based services over wireless ad hoc networks. In: BROADWIM: Broadband Wireless Multimedia Workshop (2004)
10. Yahiaoui, S., Belhoul, Y., Nouali-Taboudjemat, N., Kheddouci, H.: AdSIP: decentralized SIP for mobile ad hoc networks. In: 2012 26th International Conference on Advanced Information Networking and Applications Workshops (WAINA), pp. 490–495 (2012)
11. Banerjee, N., Acharya, A., Das, S.K.: Enabling SIP-based session setup in ad hoc networks. In: Proceedings of INFOCOM (2005)
12. Fudickar, S., Rebensburg, K., Schnor, B.: MANETSip - a dependable SIP overlay network for MANET including presentity service. In: Fifth International Conference on Networking and Services, ICNS 2009, pp. 314–319 (2009)
13. Aburumman, A., Seo, W.J., Esposito, C., Castiglione, A., Islam, R., et al.: A secure and resilient cross-domain SIP solution for MANETs using dynamic clustering and joint spatial and temporal redundancy. In: Practice and Experience, Concurrency and Computation (2016)
14. Wongsaardsakul, T.: P2P SIP over mobile ad hoc networks. Ph.D. thesis, Evry, Institut national des télécommunications (2010)
15. O'Driscoll, A., Rea, S., Pesch, D.: 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, pp. 76–80. IEEE (2007)
16. Baumgart, I.: P2PNS: a secure distributed name service for P2PSIP. In: Sixth Annual IEEE International Conference on Pervasive Computing and Communications, PerCom 2008, pp. 480–485. IEEE (2008)
17. Bryan, D.A., Lowekamp, B.B., Zangrilli, M.: The design of a versatile, secure P2PSIP communications architecture for the public internet. In: IEEE International Symposium on Parallel and Distributed Processing, IPDPS 2008, pp. 1–8. IEEE (2008)
18. Seedorf, J.: Using cryptographically generated SIP-URIs to protect the integrity of content in P2P-SIP. In: Third Annual VoIP Security Workshop (2006)
19. Davoust, A., Gagnon, F., Esfandiari, B., Kunz, T., Cormier, A.: Towards securing peer-to-peer sip in the manet context: existing work and perspectives. In: 2017 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), pp. 223–229. IEEE (2017)
20. Freedman, M.J., Morris, R.: 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, 18–22 November 2002, pp. 193–206 (2002)

21. Rennhard, M., Plattner, B.: 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, 21 November 2002, pp. 91–102 (2002)
22. Wang, Q., Borisov, N.: Octopus: a secure and anonymous DHT lookup. In: 2012 IEEE 32nd International Conference on Distributed Computing Systems (ICDCS), pp. 325–334. IEEE (2012)
23. Fonville, M.: Confidential peer-to-peer file-sharing using social-network sites. In: 13th Twente Student Conference on IT, June, vol. 21, p. 10 (2010)
24. Stoica, I., Morris, R., Karger, D., Kaashoek, M.F., Balakrishnan, H.: Chord: a scalable peer-to-peer lookup service for internet applications. ACM SIGCOMM Comput. Commun. Rev. **31**, 149–160 (2001)
25. Clausen, T., Jacquet, P.: Optimized Link State Routing Protocol (OLSR). RFC 3626 (2003)