Institute of Information Security

University of Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Masterarbeit

Security Analysis of a Machine-to-Machine Messaging System

Wolfgang Kraus

Course of Study:	Softwaretechnik
Examiner:	Prof. Dr. Ralf Küsters
Supervisor:	Dipl.-Inf. Guido Schmitz
Commenced:	2017-06-14
Completed:	2017-12-14
CR-Classification:	C.2.0, C.2.2, C.2.4

Abstract

With the increasing popularity of small embedded devices, self-organizing and self-repairing networks are preferable. One such choice are peer-to-peer (P2P) systems which create an overlay network in the application layer to achieve a de-centralized and resilient communication system. Neuropil builds an encryption layer on top of a Pastry/Tapestry P2P network to provide confidentiality and integrity to a certain degree. This thesis analyzes the protocol used by Neuropil in its default implementation and explores some possibilities to improve the security.

Contents

1 Introduction

This introduction shortly states the motivation, defines the scope and goal of this thesis and explains the further structure of this thesis.

1.1 Motivation

With the increasing popularity of small embedded devices, self-organizing and self-repairing networks are preferable. One such choice are peer-to-peer (P2P) systems which create an overlay network in the application layer to achieve a de-centralized and resilient communication system. Neuropil provides an encryption layer based on a Pastry/Tapestry P2P network to provide confidentiality and integrity to a certain degree. The encryption is realized by exchanging public keys prior to admission into the Neuropil network and subsequent messages are sent encrypted.

1.2 Scope

The scope of this thesis is to analyze the Neuropil protocol, from its implementation in its initial version 0.1 from 2016-12-30, changeset b3c1c5eeee80.

Cryptographic libraries, in this case libsodium [5], are assumed as secure, putting the main focus on the protocol and not on an implementation. As such the implementation details of Neuropil are also a minor point for this thesis.

1.3 Goal

The main goal of this thesis is a protocol description of Neuropil to perform a security analysis on it and provide insight on possible changes to improve the security of it.

1.4 Structure

The remainder of this thesis has the following structure: Chapter 2 introduces some basics of P2P systems and associated security considerations. The Neuropil protocol is explained in chapter 3, which is analyzed in chapter 4. Chapter 5 concludes.

2 Related Work

This section explains the basics starting from peer to peer networks, a general understanding of packet based communication through the Internet is assumed. Afterwards it lays out relevant work in the field of associated security considerations for peer to peer networks.

2.1 Peer to Peer Networks

Peer to peer networks consist of several peers which form an overlay network on top of the underlay network, usually the Internet. Each peer has mostly the same responsibilities in the overlay network including storing data for later retrieval and forwarding requests to other peers. Peer to peer networks classified as structured or unstructured. The former assigns IDs to peers to create a structure, the latter relies solely on network metrics such as latency for placement in the network. As Neuropil is based on a structured network, we will focus on these.

Structured Peer to peer networks are usually based on a mapping of peers to keys. These keys are used to create a distributed hash table (DHT) for searching purposes, by associating an object with a key and storing the object, or a reference to its owner, on one or more peers. While joining the network, the peer is assigned a key and thus a place in the DHT. Each peer is responsible for a subset of the hash table and retains either the data itself or a pointer to a node which holds the data. Usually a peer maintains connections to several peers closest to its own DHT-Key, called leafset.

2.1.1 Tapestry

Tapestry [13, 32] is a structured peer-to-peer overlay network, implementations include the Chimera project [19], which is used in the Neuropil implementation. The DHT-key is an m-bit string, interpreted as b-bit digit, given that m is dividable by b. For example a 256-bit string interpreted as hexadecimal characters would have 64 digits from 0-f.

Using this key, a distance metric can be established by the numerically difference of the keys.

Routing

The primary routing table at each node has b columns and $\frac{m}{b}$ rows. The row index constitutes the size of the needed shared prefix. In row zero are the furthest known nodes, which share no digits with the own node. The row one contains nodes which share the first digit with the current node, and so on.

The column index is the needed digit at the next position. Thus a row with index i and column index c, contains a node which shares a prefix of at least i digits and has c as next digit. If multiple nodes fulfill the criteria, the one with the lowest network costs (latency/hops/etc...) is used. If no such node exists the entry is left as `null`.

An active connection (in case of TCP) or a regularly refreshed soft state (in case of UDP) is established with peers in the primary routing table.

A secondary routing table with alternatives is kept and filled with entries which are not used (i.e. more expensive ones) from the primary routing table. The secondary table is used if the peer from the primary table is leaving or has failed.

A list with so-called backpointers is kept with references to nodes which have the peer in their routing tables. They are observable through the active connection or regular heartbeat messages.

An example lookup is shown in figure 2.1. To forward a message to a key, the longest matching prefix between the destination and the current node is computed. The length of the prefix is used to select the row of the routing table. Starting at the column corresponding to the next digit, it cycles through as long as the entry is `null`. If no other peer with a longer prefix exists, the current node is responsible for the message. The routing algorithm requires that the `null` entries of the lowest populated row are consistent across the peers.

Joining the network

When a new node is about to join the Tapestry network, the bootstrap node sends a message addressed to the DHT-key of the joining node. The replying node n_c is the closest DHT-Key in the current network and is used to compute the shared prefix α between the joining node and the currently existing nodes. The joining node is announced to be included in the routing tables of the other nodes with an acknowledged multicast

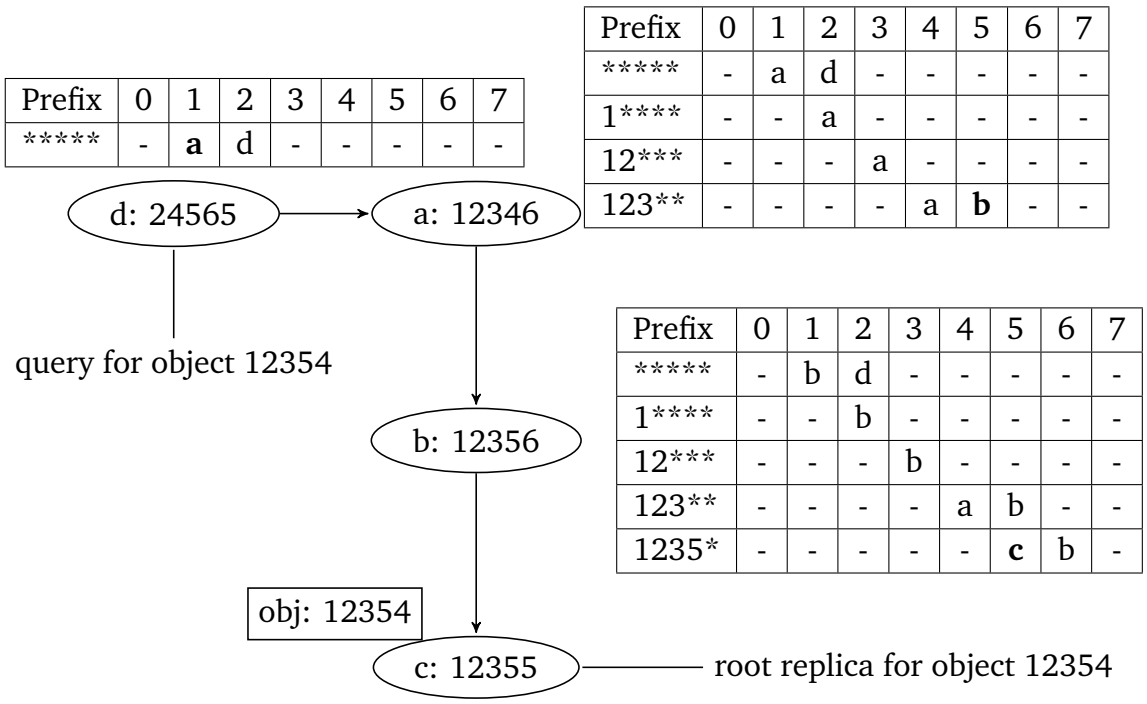


Figure 2.1: Routing example for a tapestry network. Node d looks for the object with key 12354, for which the node c is the root replica. The routing tables for the nodes d, a and b are shown in clockwise direction, starting at the top left. The entry shown in bold is the next hop.

containing (α, InsertNewNode(α, n_{new})) starting at n_c. For the acknowledged multicast n_c forwards the message to all known peers sharing the prefix α. After receiving an acknowledgment from all recipients, it sends its own. Receiving the same multicast message from a node besides the initiator, an acknowledgment is sent to that node.

After the multicast is done and the new node is establishing itself in the overlay network, it is given ownership of any stored object for which it is now responsible. As the new node is placed in the routing tables of the nodes with prefix α, it has those through the backpointer list. Using them, it can start to build its routing table through neighbor discovery or by obtaining one or more routing tables of its neighborhood.

2.2 Security in P2P Networks

By design P2P networks have several weaknesses, as arbitrary peers have responsibilities for the function of the network, including searching and storing of entries, as well as routing messages. According to Urdaneta, Pierre, and van Steen [27] the most studied attacks are Sybil [2, 3, 6, 16, 26, 27, 28], Eclipse [23, 24, 27, 31] and Content poisoning attacks [4, 9, 25, 27, 28]. Usual proposals to reduce the impact of these attacks include computational puzzles or a trusted third party for admission into the network, quorums for byzantine (arbitrary behavior) fault tolerance, redundant routing, and self-certifying data.

A computational puzzle, or crypto puzzle, require a node to compute a solution to a computationally hard problem. This is usually acquiring a nonce, so that a cryptographic hash of the network identity concatenated with the nonce fulfills a condition, for example having a certain amount of leading zeros.

2.2.1 Authentication

As stated in the RFC7435 Opportunistic Security [7],

> [p]rotection against active attacks requires authentication. The ability to authenticate any potential peer on the Internet requires an authentication mechanism that encompasses all such peers. No IETF standard for authentication scales as needed and has been deployed widely enough to meet this requirement.

Thus an open P2P network admitting everybody is currently not able to authenticate all peers. The available authentication mechanisms include (pre-) shared secrets and a trusted third party through a public key infrastructure (PKI). Using them to restrict membership in the P2P network gives an additional hurdle to pass, in terms of acquiring a device, certificate or locality.

If no authentication of the network peers is possible, the exchange of information might be done similar to encrypted subscriptions, as mentioned by [20, 29]. By publishing functions to establish whether a message matches a subscription, the message itself can be sent encrypted although an out-of-bands distribution of encryption keys is necessary.

Pre-shared secrets include factory installed ones, as used by some IoT devices. However as physical access to those devices is possible, the secret can most likely be recovered as shown by Ronen et al. for a Philips Hue smart lamp [21]. The key was recovered through side-channel attacks on the hardware and was used to sign attackable firmware to expose the device.

Another mechanism is by using a locality metric, for example the strength of a wireless signal. For wired networks, membership in a secured local subnet might be used to distribute a shared secret for later communications. This however has the drawback, that the device needs to be bootstrapped at a certain access point. In the case of physically mobile devices and communication through the Internet, the key would need to be refreshed regularly, to avoid a possibly compromised key to be used indefinitely.

The way of a trusted third party includes the Certificate Authorities (CAs) and DNS entries [14], which can be secured through DNS security (DNSSec) [1]. However the verification through a CA also has its own problems of the CA being not really trusted or responsible [10, 11, 12]. By extension these problems also exist for DNSSec, extended by having only a single issuer for the root zone and possible loss of functionality if DNSSec is not properly supported [17].

2.2.2 Sybil attack

The Sybil attack, as described by Douceur [6], entails the introduction of multiple malign peers from at least one host. By joining the network multiple times with each host, the amount of nodes controlled by the adversary rises, without the need for a one-to-one mapping of host to node. The goal is the subversion of the confidentiality or correctness of the P2P network, as the attacker has an improved chance to disrupt the function. The probability to manipulate a query is $(1 - f) * d$, for a network with a fraction of f malign nodes and average path length d, where d is usually logarithmic to the amount of nodes.

To mitigate the occurrence of Sybil nodes, the admission rate of nodes into the network can be limited. This can be done for example by increasing the cost of joining the network in terms of resources or money. Limiting by resources can be done through computational puzzles which needs to be adjusted to the resources of the weakest peer and needs to be changed regularly. If the puzzle is not changed a single adversary could still introduce an arbitrary amount of nodes. A possible random factor for the puzzle could be to include a pre-defined and mostly unpredictable external value like the Dow-Jones Index [31]. This consumes resources in any case and an attacker could introduce up to $\lfloor \frac{\text{resources attacker}}{\text{resources weakest}} \rfloor$ nodes. The monetary restriction could be done through a requirement that the nodes need to have a unique certificate issued through a CA and verifiable through the PKI.

Another approach is establishing a distinct identity for each peer. This can be done through network triangulation [2], by having each node include the distance (response time) to several network beacons. These identities can be used to restrict the amount of peers a single identity can introduce into the network. The drawbacks include the need

for (trusted) network beacons and accuracy, as clusters will exists, for example a beacon in Europe will see most peers in the United States with a similar latency.

To tolerate a small amount of Sybil nodes and still operate correctly, secure routing [3] can be used. The success rate for an operation is multiplied by the amount of unique paths to the destination, given that the destination is an honest peer. If it is used, insert and update operations should always use the secure routing algorithm, whereas a lookup can be done normally. If the lookup does not yield a result, it can be repeated with the secure routing algorithm. The drawback includes the definition of a timeout function for a lookup and the increased message overhead even in the absence of malicious nodes.

Using a social approach for use-cases where a social component exists, for example instant messaging, the underlying structures of a social network can be used. By arguing that this graph has few edges between clusters of malign (Sybil) nodes and honest nodes, Lesniewski-Laas and Kaashoek [16] claims to create an unstructured, Sybil-resistant P2P network, by using this graph as bootstrap graph. Connections between a Sybil cluster and an honest cluster are thought to be expensive, in terms of establishing a connection between a Sybil node and an honest node through social engineering or similar approaches. Thus increasing f by adding more (inexpensive) nodes, the effect is not amplified, as the connections between the node clusters requires the Sybil node to establish contact through non-automated interaction on part of the honest peer (for example adding to a friend list on the social network). The drawback of these approaches are that a bootstrap graph needs to exist, and a mapping exists as in user a and peer b are the same entity, which is part of the authentication problem.

2.2.3 Eclipse attack

The eclipse attack is described as more general than the Sybil attack [24]. By introducing colluding attacker nodes at a disproportional rate into the routing table of a target node, the attacker controls the message flow from and to the node, thus eclipsing it. This can be used to prevent service from or to the eclipsed node or to exploit its resources. In contrast to the Sybil attack it is a targeted attack and the attacker is not forced to control a large faction of the nodes, only enough to fill the routing table of that peer.

Large DHTs can be hardened against Eclipse attacks for example through redundant routing, requiring computational puzzles or employing the cuckoo rule and small scale DHTs can employ a more rigorous authentication process, thereby limiting the amount of malicious nodes in the network.

As argued by Zhang et al. [31], requiring computational puzzles with a changing initialization vector (IV) for the DHT-Key generation hinders the ability to eclipse a node

in a large scale network. Generating any valid DHT-Key takes some while and obtaining several keys close enough to the target requires a large number of valid keys in a densely populated network.

Another possible mitigation is limiting the in- and out-degree of the peers [24]. By only accepting a certain amount of peers into the backpointer list and processing only messages from these peers, a single node cannot be included in an arbitrary number of routing tables. The drawbacks include the need to regularly audit the neighbors via anonymous routing and pruning any connections which are not valid. Invalid nodes are those with too many incoming connections, or if it is a peer from its primary routing table, not being included in the backpointer list and vice versa.

To prevent a selective insertion through repeated join/leave attempts in the case of network assigned DHT-Keys, the Cuckoo rule [22] can be applied. When a node joins the network for the first time and is assigned a place in the DHT, some nodes in the neighborhood are forced to rejoin the network at a different location. This has the disadvantage of regular churn through each introduced peer, which can lead to frequent ownership changes of published content and thus an increased message overhead.

2.2.4 Content poisoning

Being the responsible peer to store certain data or by pretending to be that peer, an attacker can ignore the query, or send arbitrary data as answer. To detect these manipulations, self-certifying data can be used. By signing the hash of the data with a trusted key, a receiving node can check whether the received data is correct [4, 9]. This requires acquisition and verification of the key through some out-of-band means.

2.2.5 Tolerating adversaries

To tolerate peers which exhibit arbitrary behavior, also called byzantine failure in regard to the byzantine general problem, a distributed system can employ quorums. The quorum needs to have at least $3 * f + 1$ peers, to tolerate f byzantine peers, to ensure that there is always a majority of honest peers. Such quorums are applicable in a DHT to ensure correctness of the routing [8, 30] and thus information storage and retrieval, given that the target is honest.

For example the robust communication protocol II as described by Young et al. [30] uses distributed signatures to prevent a peer to act on its own. The distributed key is stored in shares among the peers and it takes a certain amount of shares to reconstruct the secret/create a valid signature. There also exist a set of associated public shares

which are used to verify signatures. Each quorum knows the public key of its neighbor quorums and signs those to enable a chain of trust. A sending peer first informs its own quorum Q_1 that it wants to send a message and the quorum needs to supply a signed proof that the sender is allowed to do so. One possible reason to deny the sender would be excessive sending of messages (spamming). With this proof the sender can contact the next quorum Q_2 as computed by the routing table. The response of Q_2 is the next quorum Q_3, the public key of Q_3 and a proof that Q_2 permitted the message, all signed by Q_2 and verifiable through the public key known to Q_1. This continues until the quorum Q_i which contains the message target is reached, and the sender delivers the proof of Q_{i-1} and the message to the target.

For these techniques to be effective the amount of byzantine peers, and their distribution throughout the quorums, needs to be controlled. Thus admission restrictions (for example the computational puzzles with periodically changing initialization vector for the puzzle) and either a verifiable node-id generation and/or a mechanism to prevent targeted insertion like the cuckoo rule needs to be in place.

2.3 Possible attackers

There are several classes of possible attackers in a network context with differing capabilities. The stated attacks on peer to peer networks usually fall into the category of malicious peers.

2.3.1 Passive network attacker

A passive network attacker is somebody who only reads the messages going through the underlay network, but does not interfere in the message exchange. Any information which is not encrypted can be read by this attacker. This can be the case if some part of the routing architecture is compromised or otherwise compelled to provide the information.

2.3.2 Active network attacker

An active network attacker controls (part) of the network and is able to capture, change and replay arbitrary packets. It is similar to the passive network attacker, however it actively interferes with the communication passing through it. Any non-authenticated

encryption can be subverted by this attacker, as it can act as a man-in-the-middle and forward and/or manipulate the data passing through it.

In regard to peer to peer systems it is capable to attack certain nodes as its capabilities allow it to interrupt the communication between honest peers easing for example eclipse attacks. Depending on the authentication schema it might also introduce malicious nodes into a network.

2.3.3 Malicious peer

One or more nodes which are part of the network but do not necessarily follow the protocol. This case is distinct from the active network attacker, as the malicious peer can only manipulate the packets passing through one of its nodes. They can drop, misroute or manipulate arbitrary packets passing through them and inject arbitrary packets into the network. The arbitrary behavior is usually called byzantine failure in the context of distributed systems and can appear for example if one server of a distributed system has been compromised.

Additionally it might disrupt the protocol by inserting junk messages to generate unnecessary network load (spamming) and depending how new nodes are authenticated by expanding its influence through additional Sybil nodes. Another possible attack could be convincing other peers to send unsolicited messages to a target peer, for example by faking a lookup request.

2.3.4 Honest but curious

An honest but curious node follows the protocol but wants to obtain more information than it would need to follow the protocol. Any information, for which the exchange is not authenticated, and at some point becomes known to an honest but curious node could be leaked to it.

3 Neuropil protocol

Neuropil is a Machine-to-Machine Messaging System aimed to provide encrypted communication between two or more peers. The Neuropil protocol is based on Pastry/Tapestry as distributed hash table (DHT) and uses it for routing of messages with an additional public key exchange and encryption. The used key length for the DHT-Keys are 256 bit, interpreted as 64 hexadecimal digits. The additions are both below the overlay network to encrypt the messages for the next hop, and above to encrypt the message payload separately from/to the application layer. All messages are padded and/or segmented so that each packet is 1024 bytes long. It supports two different modes for acknowledging packets, the usual destination only where the receiving peer sends the acknowledgment and an each-hop mode where every forwarding peer sends one.

To initiate an exchange of application payloads, a peer announces an intent to receive or send messages about a subject. To avoid the need to know the DHT-Key and/or network address of each other, an approach similar to topic based publish/subscribe networks is used through the DHT. The peer having the DHT-Key closest to the subject key is responsible for the message discovery. This peer collects and distributes the sender/receiver tokens of interested peers. After the initial exchange of interest tokens, sender and receiver establish a point-to-point communication directly through the overlay network (possibly through other peers).

Each node is identified through a DHT-Key of the overlay network as used by Pastry/Tapestry. The DHT-Key is obtained through hashing the identifying information of a node or message which are contained in an AAA-Token.

3.1 AAA-Token

The data structure containing the identifying information in addition to the public key and a signature of the owner are called AAA-Token, which stands for authentication, authorization and accounting token.

Identifying information includes a realm, issuer, subject, audience, valid from/to (time), public key, optional extensions and a self-generated universally unique identifier (UUID).

Realms and audiences are used for message filtering and may be empty. Currently the values for realm, issuer, subject and audience are plain text, a future change to use hashed values is intended.

The optional extensions are key/value pairs and might be used for use-case specific information by the user of the Neuropil library. Possible uses include message authentication codes (MACs) obtained through a pre-shared secret, or a signature which can be validated through a public key infrastructure (PKI). Currently the extensions are neither part of the DHT-Key nor verifiable through the signature of the token, as some intended purposes require mutable information.

The two main uses for the token are node identity and message sender/receiver intent. For the node identity the subject is the issuer, for the message intent it is the subject of the application payload.

3.2 Initialization / Handshake

The messages for joining the network are shown in figure 3.1. The left-hand side shows a new node initiating the communication, which is the case when obtaining the bootstrap node through some out-of-band means. On the right-hand side the bootstrap node knows about the joining node, which might be the case if the bootstrap node is a controller responsible for starting the other nodes.

The initial session key for symmetric encryption between nodes follows the elliptic curve Diffie–Hellman (ECDH) key agreement scheme. One of the nodes initiates the protocol with a handshake message (1) containing the public key of the sender, to which the other nodes replies with its own handshake (2). At this point, both know the public key of each other. Using its own private key and the public key of the other node, a shared secret can be computed.

After the handshake is complete, the initiating node sends a join request (3) to the other node, to be included its routing table. Now the bootstrap node may evaluate the information contained in the AAA-Token to either acknowledge or reject this request. If it is acknowledged (4), the new node is part of the network (-realm) and the bootstrap node proceeds to send a part of its routing table (5). The row with the longest shared prefix between the bootstrap and the joining node is sent, or the leafset if that row is empty. Also the bootstrap node announces the joining node to the closest peers it is currently connected to, which may establish connections to the new node through own join requests. This enables the new node to build its routing table and being included in the overlay network.

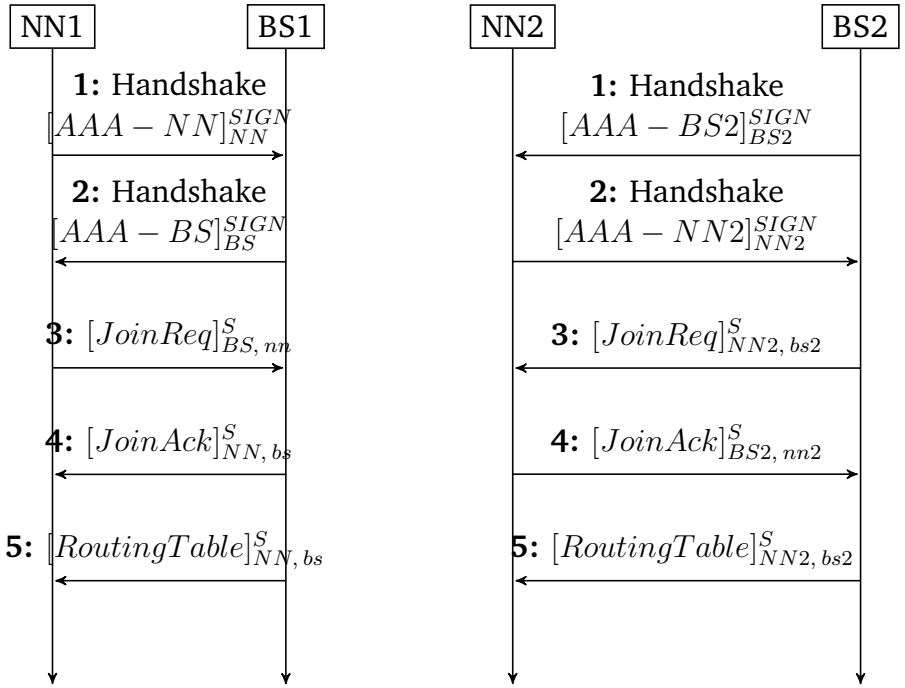


Figure 3.1: NN = New Node, BS = Bootstrap Node.
Messages for joining a Neuropil overlay network. Left-hand side is peer initiated, right-hand side is bootstrap initiated.
$[Message]^{SIGN}_{a}$ is a message signed by node a, which can be verified with the public key.
$[Message]^{S}_{a,\,b}$ is symmetrically encrypted using the ECDH derived shared secret between nodes a and b.
$AAA - node$ is the token containing the node identity and public key of the $node$,

If a realm is used, the bootstrap node in the figure is the realm master. Should some other node receive a join request it is forwarded to the realm master. The forwarding is done through sender/receiver intents with the realm name as subject and sending the AAA-Token to the realm master.

3.3 Message intent / Sender & Receiver List

The messages for a subject based message exchange (application payloads) are shown in figure 3.2. The routing key for the sender/receiver intent is obtained by hashing the subject name. The Neuropil node which is closest to this key is the intermediate

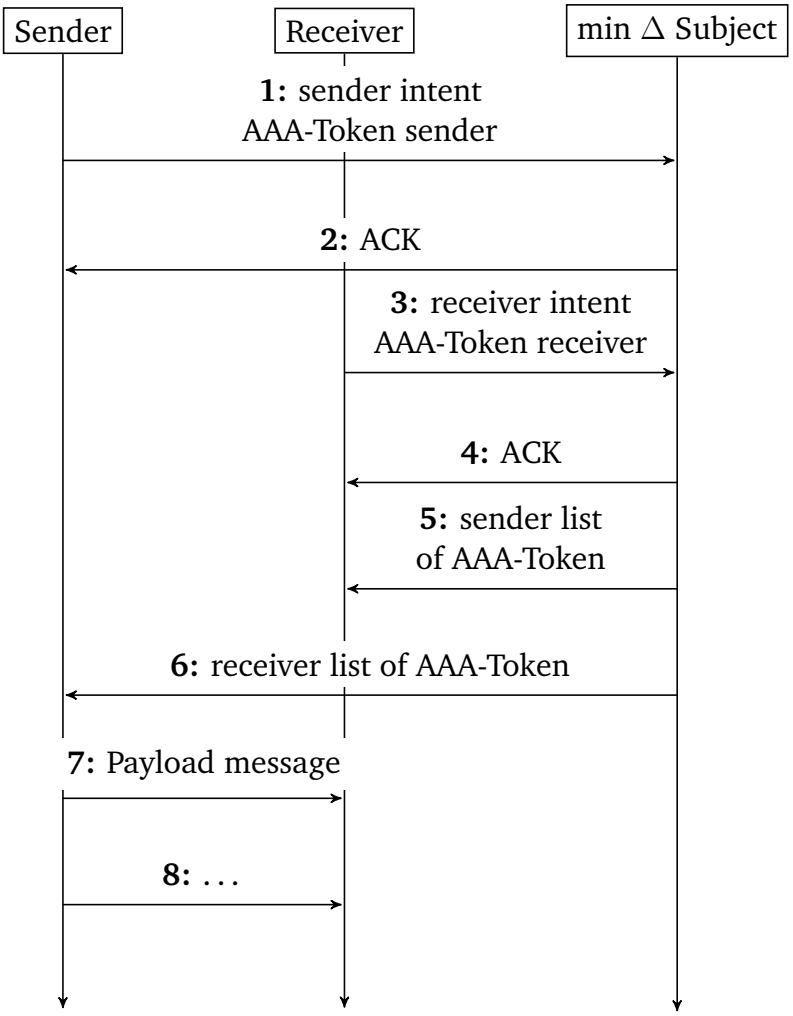


Figure 3.2: Messages for sending a message in the Neuropil overlay network. 'min Δ Subject' is the intermediate Node with the DHT-Key closest to the Routing Key generated for the subject. All messages are symmetrically encrypted using the ECDH derived shared secret between the corresponding peers.

node responsible to gather sender and receiver tokens. These tokens contain the subject, whether it is a sender or receiver, the node identity, optionally a realm as intended audience, and is signed by the creator of the token. Matching and thus filtering for possibly existing audiences contained in the tokens is done by the intermediate node before a list is sent.

3.4 Message parts

Payload message structure:

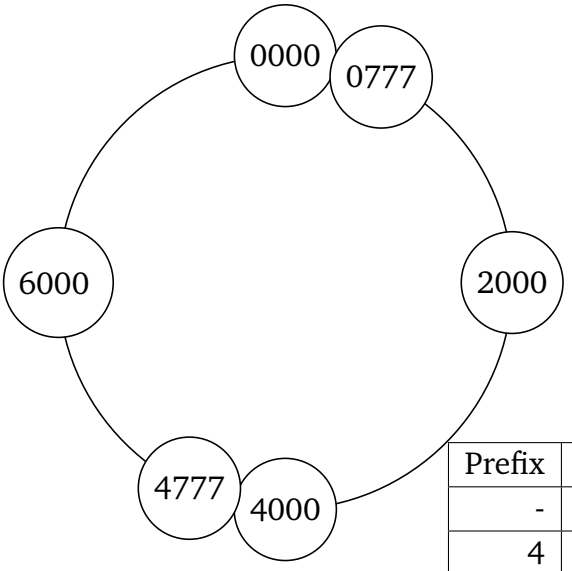
- Routing information: DHT-from, DHT-to, DHT-reply to, sent time, expiry time, subject, sequence number(monotonous increasing), a generated UUID for ACK/reply purposes
- Payload key: k^M symmetric key for the payload
- Payload
- padding to 1024b - signature length
- Signature over the message

For the payload message encryption, k^M is used for symmetric encryption. K^n is the public key of the next hop. K^r is the public key of the receiver as contained in the receiver token. Each encryption step uses a generated random nonce:

- Payload: symmetric (k_m^S).
- Payload key: for the receiver, authenticated symmetric encryption with ECDH-derived shared secret (K^r, k^s).
- Complete message: for the next hop, authenticated symmetric encryption with ECDH-derived shared secret (K^n, k^s).

Control messages, for example join requests or pings, employ only authenticated encryption to the next hop. As those messages do not contain an application layer payload additional encryption is not necessary. As a special case the ACK can travel through several nodes and contains only the UUID of the message.

3.5 Example message flows



Routing table of 4000

Prefix	0	1	2	3	4	5	6	7
-	0000	-	2000	-	4000	-	6000	-
4	4000	-	-	-	-	-	-	4777
40	4000	-	-	-	-	-	-	-

Figure 3.3: Small example network with six nodes to illustrate several message exchanges. Realistic distribution and density have been ignored to create a smaller example. To show message forwarding between nodes, it is assumed that the numerically lowest node is used, if multiple nodes could be used at one point in the routing table.

Message discovery for sender 6000 and receiver 4777 about a subject that hashes to 0772, using the network shown in figure 3.3, and visualized in figure 3.4. For this example it is assumed that the numerically smallest node is used for the routing table, if there are multiple nodes for a prefix, resulting in several indirections.

Receiver discovery of the sender (Steps 1-2):

1. 6000: Sender intent for subject 0772, generation of AAA-Token $[AAA - s]^{sign}_{6000}$
2. 6000: Sending token to nearest known node 0000

   $[target:\ 0772,\ message\ [AAA - s]^{sign}_{6000}]^{S}_{0000,6000}$
3. 0000: Receiving $[target:\ 0772,\ message\ [AAA - s]^{sign}_{6000}]^{S}_{0000,6000}$ and forwarding to 0777 as $[target:\ 0772,\ message\ [AAA - s]^{sign}_{6000}]^{S}_{0777,0000}$
4. 0777: As nearest node, matching known tokens. As only a sender intent is known, nothing needs to be done.

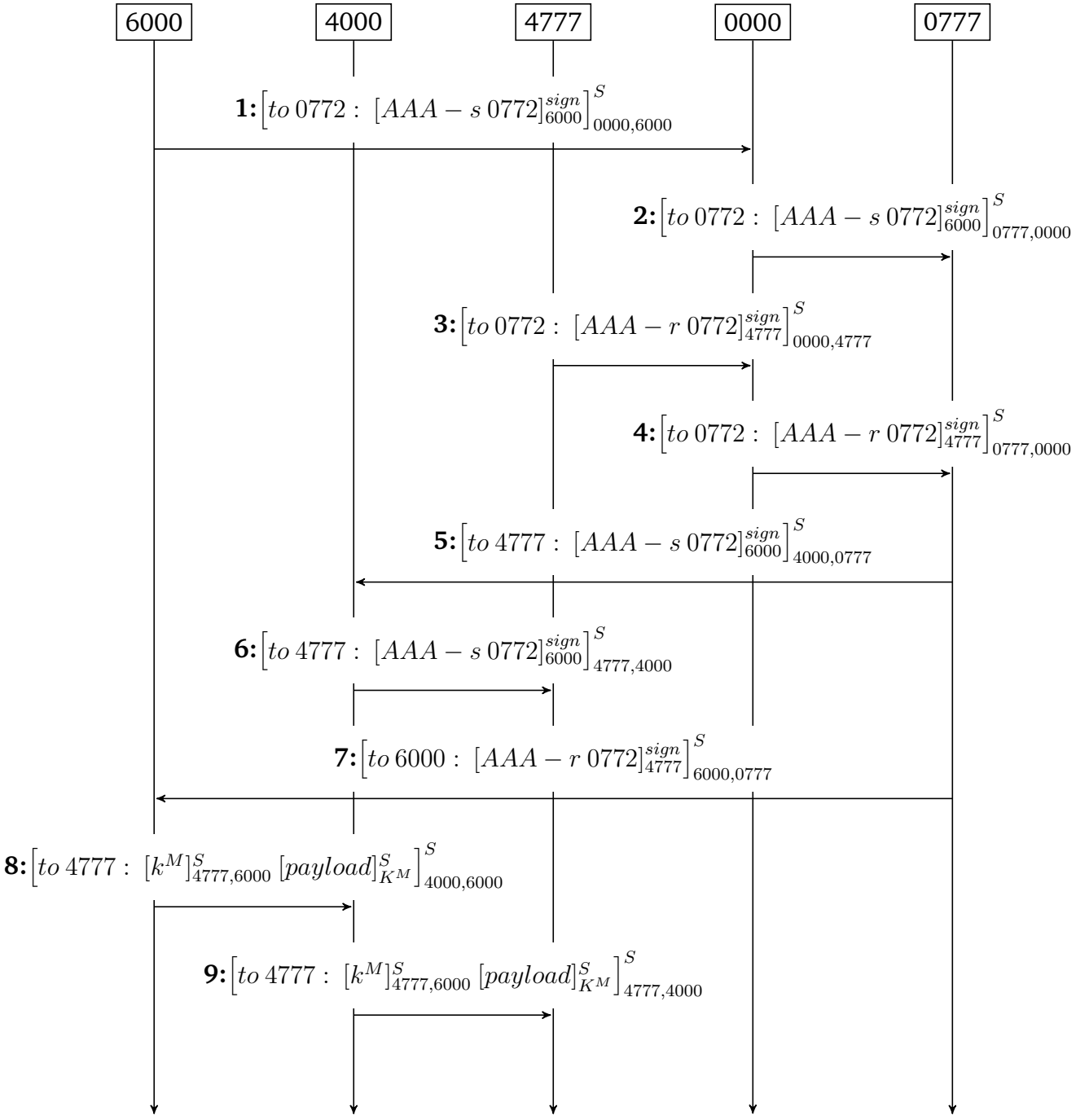


Figure 3.4: Subject exchange and transmission of payload messages. $[x]^{S}_{Y,z}$ denotes authenticated symmetric encryption of x using the ECDH derived shared secret with the public key of Y and the private key of z. k^M is a random symmetric encryption key, which is generated for each message by the sender.

Sender discovery of the receiver (Steps 3-4):

1. 4777: Receiver intent for subject 0772, generation of AAA-Token $[AAA - r]^{sign}_{4777}$
2. 4777: Sending token to nearest known node 0000

   $[target:\ 0772,\ message\ [AAA - r]^{sign}_{4777}]^{S}_{0000,4777}$
3. 0000: Receiving $[target:\ 0772,\ message\ [AAA - r]^{sign}_{4777}]^{S}_{0000,4777}$ and forwarding to 0777 as $[target:\ 0772,\ message\ [AAA - r]^{sign}_{4777}]^{S}_{0777,0000}$
4. 0777: As nearest node, matching known tokens. Distributing $[AAA - s]^{sign}_{6000}$ and $[AAA - r]^{sign}_{4777}$ to each other.

Distribution of tokens (Steps 5-7):

1. 0777: sending to 6000: $[target:\ 6000,\ message\ [AAA - r]^{sign}_{4777}]^{S}_{6000,0777}$
2. 6000: Receiving $[target:\ 6000,\ message\ [AAA - r]^{sign}_{4777}]^{S}_{6000}$, checking and saving the token, ready to send application payloads.
3. 0777: sending to nearest known node 4000 $[target:\ 4777,\ [AAA - s]^{sign}_{6000}]^{S}_{4000,0777}$
4. 4000: Receiving $[target:\ 4777,\ [AAA - s]^{sign}_{6000}]^{S}_{4000,0777}$ and forwarding to 4777 as $[target:\ 4777,\ [AAA - s]^{sign}_{6000}]^{S}_{4777,4000}$
5. 4777: Receiving $[target:\ 4777,\ [AAA - s]^{sign}_{6000}]^{S}_{4777,4000}$ checking and saving the token, ready to receive application payloads.

Exchange of application layer payloads (Steps 8-9):

1. 6000: Sending application payload to 4000

   $[target:\ 4777,\ message\ [k^M]^S_{4777,6000}\ [payload]^S_{k^M}]]^S_{4000,6000}$

2. 4000: Forwarding the message to 4777

   $[target:\ 4777,\ message\ [k^M]^S_{4777,6000}\ [payload]^S_{k^M}]]^S_{4000,6000}$

3. 4777: Receiving the message:

   $[target:\ 4777,\ message\ [k^M]^S_{4777,6000}\ [payload]^S_{k^M}]]^S_{4777,4000}$

   decrypting the symmetric key and verifying the MAC using the session key of the public key from the sender token and its own private key. If successful, decrypting and consuming the application payload.

4 Protocol Discussion

This chapter discusses how the general attacks on DHT's are applicable to Neuropil and possible mitigations against them. Most if not all mitigations incur an overhead in terms of network load, computation resources and/or administration.

4.1 Network messages and attacks

Beginning from the handshake, the following messages are sent on the wire, for a joining node A, bootstrap node B and their public keys K^A and K^B. Node identity (network location and DHT-Key) of B needs to be known through out-of-band means:

Handshake 1 [target: B, [A, $K^A]^{sign}_{k^A}$]
Handshake 2 [target: A, [B, $K^B]^{sign}_{k^B}$]

Subsequent messages on the wire are symmetrically encrypted to the next peer, with the shared secret derived by ECDH and use a new random nonce for each encryption operation. The nonce is transmitted in plaintext with the message.

For initial setup of a sender/receiver relationship, signed interest tokens are sent to the node closest to the subject key. The token itself is not encrypted as the node responsible for the subject might be unknown to the sender/receiver. Message intent of sender S:

[target: DHT-Key of subject [S, subject, $K^S]^{sign}_{k^S}]^S_{nexthop}$

The application payload messages are symmetrically encrypted and contain the encrypted symmetric key k^P in the message. This key is symmetrically encrypted and contains a MAC, using the session key with the receiver. Payload of sender S to receiver R:

[target: R, $[k^P]^{S+MAC}_{K^R,k^S}$,[payload]$^S_{k^p}]^S_{nexthop}$

4.2 Eclipse attack

If a joining node can choose its own key for the DHT, eclipsing a node, a key or certain parts/rows of the routing table becomes easier [23, 24, 25, 27, 28]. Usage of the self generated UUID for the DHT-Key allows an attacker to generate many different node identities offline and choose the best fitting to assume control of a known subject exchange or leafset of certain nodes.

Possible locations to eclipse from the example in figures 3.3 and 3.4 would be 0777 to eclipse the message exchange of the subject 0772, or 0000 and 4000 to eclipse the message flow to their neighbors 0777 and 4777 respectively. In an actual network the rows with the shorter prefixes can be eclipsed easier, by aggressively announcing itself and other (few) collaborating nodes. By filling the routing table, as well as trying to beat the other nodes in the used metric, for example network latency, the initial long jumps can be misrouted to a collaborating malicious node with a plausible DHT-Key or dropped entirely.

A possible target would be the discovery of the realm master for joining nodes, which is done through the usual sender/receiver intents without further validation. If a malicious node controls the exchange, it can present itself or another malicious node as the realm master. By subverting the actual realm master, a joining honest node can be refused for a denial of service, or forwarded to a fake network to observe what it wants to send or receive.

4.2.1 Preventing the joining peer to generate its own DHT-Key

By limiting relevant information to create the DHT-Key to verifiable data and/or having the network generate and sign the random parts for the joining node, an attacker is forced to repeatedly join and leave the network to gain a key close to its target. Verifiable data in this case includes the network address, port number and publicly available data like stock values. While restricting the data to the network address and port still allows for a broad range of possible DHT-Keys, but gives a bound to the amount. By including a publicly available and unpredictable value as IV for the DHT-Key, regular re-computation of possible values is required, limiting the useful duration of a pre-computed identity list.

If a realm master is used in Neuropil, it is responsible for authenticating any joining node for its realm. Thus it could be used to generate the UUID for the joining node and sign the resulting identity token. Other nodes can reject any join/update messages containing nodes without a signature of a trusted realm master. To include several

realms in one DHT network, a realm master could sign the identity token of another realm to allow nodes to share the network.

For a joining node the message overhead is negligible, as the realm master needs to receive the joining token in any case, and the reply with the signed identity token is constant. As the node is not yet part of the DHT, the change in its DHT-Key should cause no other updates at this point.

The space overhead to restrict cross realm communication is linear in the worst case to the amount of realms. By collecting the signatures of other realm masters in its identity token, including the realm master token along with the identity of the join/update should incur a small message overhead, if the signature is missing. If the time of the realm membership should be constrained, then a valid duration can be included in the signature, leading to the following token for an *other* realm master identity, signed by the *own* realm master:

$$\text{RealmToken}_{other,own} = [[\text{AAA-RealmMaster}]^{\text{sign}}_{\text{other}}, \text{valid from, valid util, PublicKey}_{\text{own}}]^{\text{sign}}_{\text{own}}$$

$$\text{Cross-realm Handshake:}[[\text{AAA-Joining}]^{sign}_{joining}]^{sign}_{other}, \text{RealmToken}_{other,own}]$$

This allows a node to validate the $\text{RealmToken}_{other,own}$ against the public key of its realm master and check the identity for the join/update request against the signature of the *other* realm master. Thus the request can be allowed/denied without further communication while restricting membership to authenticated peers.

4.2.2 Static key for subject

The subject name is the only part for generating the corresponding DHT-Key, thus if the nearest node is attacker controlled the subject will stay attacker-controlled. This allows the attacker to ignore any honest receivers for the subject and present itself as only receiver to potential senders.

Currently the subject in the sender/receiver token is stored as plaintext, which enables the any node in the path to the intermediate node to obtain the subject name. Without further authorization between sender and receiver it cannot be verified whether each other is supposed to send or receive messages about that subject.

Hashed subject names

The planned change of the Neuropil protocol by its author to include only hashed values of the subject name prevents the other nodes, the intermediate node and the path to it,

to obtain the name. The name could then be used as an additional shared secret for the encryption or authentication of the message payload. This could be done by sending a nonce in the message and using the result of an HMAC of the nonce together with the subject name as initialization vector for the encryption function.

In addition, the algorithm for the Time-Based One-Time Password (TOTP) [18] could be used to concatenate the subject name with a timestamp, to prevent a permanent eclipse attack against an unknown subject. The DHT-Key of the subject will move through the DHT and cannot be compromised indefinitely after acquiring the hash value while forwarding an intent. Given the ephemeral nature of the sender/receiver intent, the message overhead should be negligible, if the time frame for the TOTP is similar to the time to live of the intent. Calculating the next subject DHT-Key and introducing a node close to it requires knowledge about the subject name.

4.3 Authentication

If no realm master is used, or if it does not require any authentication, then an arbitrary amount of nodes can be introduced into the network. Additionally any peer can introduce new peers into the network without further validation through an update message. The method of authentication is left to the user of the library through usage of key:value pairs in the extensions of the identity tokens, which are sent without encryption during an handshake. Given the open nature of the extension, an authentication mechanism which does not reveal an existing shared secret is preferable for example using it with as key for a hash-based message authentication code (HMAC) [15] over the identity token containing the network address and public key of the node.

4.3.1 Sybil attack

Any node in the network can send update messages to announce new peers without additional scrutiny, a malicious or compromised node can populate the network with Sybil nodes. There exists no restriction on the amount of nodes one physical entity can introduce into the network.

If the amount should be restricted, a differentiation of nodes would be required. The stated target of embedded devices make resources constrained differentiation [6], in which hosts are tested for distinctiveness by resource consumption not feasible. Using the network address has the problem that several physical hosts might share an external IP address through network address translation (NAT) and in the case of IPv6 currently exists an overabundance of possible addresses.

In a network with n peers, average path length of $d \approx log_{16}(n)$ and a fraction of f Sybil nodes, the possibility of a message not passing through a Sybil node is $(1 - f)^d$.

The vulnerable part of the protocol is the exchange of sender/receiver intents, as those use the lookup and store operations of the DHT network without any other knowledge about the responsible node. This allows a Sybil node on the path to an honest root node for a subject to misroute, drop or assume control of that intent.

After a successful exchange of intents, the point-to-point semantic of payload messages in Neuropil and encryption based on the contained keys limits a Sybil nodes along the path to drop or corrupt the payload message. Using the each hop ACK mode of Neuropil, the traversed path and possible black holes might be seen. A missing ACK for the message can be used to initiate an additional send operation and nodes which lose payload messages regularly might be removed from the routing table of the other peers. To react as a network as opposed to a single node, in regard to missing ACKs, non-repudiation of receiving the message is needed. A Sybil node might send an ACK only to the previous node, discard the packet and blame the previous honest node.

Secure routing

Using secure routing [3], the message is sent to all entries in the fist row of the routing table. By having multiple starting points for the message routing, more unique paths to the destination node should exist. Two paths are considered unique if the forwarding chain form the source to the destination shares no peer. This improves the successful routing of the message with u unique paths to $u * (1 - f)^d$.

The increased message overhead might be undesired, especially in a network with few Sybil nodes, thus limiting the usage of the secure routing algorithm to situations with observable routing failures might be preferred.

Replication

To eliminate the single point of failure, the node responsible for the exchange of the subject can be replicated at different points in the network as outlined in several papers [25, 28]. This forces an attacker to invest more resources to completely control a subject and provides a higher tolerance against random failures. The drawbacks are a higher space consumption and message load, although the space overhead should be negligible, as the sender/receiver intent are relatively small and ephemeral. Classical positions for the replicas are either in the leafset of the responsible node (e.g. replicated

at the 5 nodes closest to the target DHT-Key) or by creating different DHT-Keys through concatenation with some replica ID prior to hashing.

Replication along the leafset has the advantage that the content is still available at the location after a node failure and the replication can be done through the responsible node. If the root replica fails, the next closer node assumes control and initiates potentially needed backups. If the responsible node is malicious, it might skip the replication to deny access to it later, or as a member of the leafset, it might lie about the replication and prevent the propagation through the leafset.

Concatenation with a replica ID has the advantage that the sending node can send the messages to the intended locations/peers and by spreading the replicas throughout the DHT more unique paths from a sender or receiver are possible. Assuming a cryptographic hash function for generating the DHT-Keys, the replicas should be spread evenly through the DHT, creating unique paths to the corresponding replica positions.

In regard to eclipsing the target space of a subject of interest with attacker nodes, the concatenation with the replica ID should be better suited as it should create multiple unique paths.

4.4 Authentication of the realm master

If the bootstrap node is not the intended realm master, or no realm master exists, the joining node cannot verify whether it joined the intended DHT network. Due to obtaining the realm master through the sender/receiver intent, any node from the bootstrap node to the node closest to the DHT-Key of the subject can stop forwarding the intent and present itself as the realm master. The bootstrap node then forwards the handshake to the node claiming to be the realm master. This allows the malicious node to reject the joining node and prevent access to the network, or having the joining node become part of a completely attacker controlled network.

Classical means to authenticate the realm master include usage of a PKI, by including a signature in the AAA-Token of the realm master which can be verified through the PKI. If domain names and associated DNS records are used to distribute information about the DHT network, the public key or a hash of the identity of the realm master can be included in the DNS record and potentially secured through DNSSec [1] similar to the DNS-Based Authentication of Named Entities (DANE) [14].

Through the DHT network itself it could replicate the message intent for the realm master and use the majority in the returned tokens. To prevent non-bootstrap nodes

from interfering with the message intents, the identity token of the realm master can be kept by each node and used to validate future receiver intents.

4.5 Node auditing

If there is some trusted node, for example a realm master, and non-repudiation, the message exchange nodes can be audited. By extending the ACK of a sender/receiver intent with a cryptographic hash of the message content along with non-mutable header fields along with a signature, non-repudiation of receiving the intent can be established.

An audit can be initiated by any sender or receiver node through an audit request, forwarding such an ACK. The trusted peer then requests the sender and receiver lists for that subject from all replicas. After matching the tokens by the trusted peer any match can be asked whether it received the correct and/or complete list. If any ACK'ed sender or receiver token is missing from the list, or a substantial amount of sender and receivers got incomplete lists, the subject node is flagged as suspicious or eventually removed from the network by no longer admitting nodes from that IP/Hostname.

Depending on the available resources of the intended devices and the trusted peer the rate of audits might be fixed to prevent overloading some node.

5 Conclusion

Neuropil is suited against passive network attackers, as only the handshake is sent in plaintext and any further communication is encrypted through ECDH derived symmetric keys. Length and amount of payload messages can only be approximated, as all packages are padded to 1024 byte and are indistinguishable from control messages.

Against active network attackers its suitability is defined by the authentication mechanism used by a potential realm master. Definition and implementation of the mechanism is currently left to the user of the Neuropil library. Without proper authentication, or in the case of an open network, it is be possible to interrupt communication and eclipse arbitrary peers.

In any case, given the peer to peer nature of the protocol, malicious peers are able to gain control over parts of it. Currently the joining peer can pick its own place in the DHT, enabling a malicious peer to pre-compute as many identities as it needs to obtain values for eclipsing a target location. Additionally, new peers are only authenticated once while joining through an honest bootstrap node and any node can introduce arbitrary peers into the network through update messages. This can lead to either denial of service of known subjects and, if no additional authentication of senders and receivers is performed, inserting of fake data and information leakage respectively.

To mitigate the effect of an eclipse attack, the DHT-Key for a subject could include a timestamp similar to the generation of a Time-Based One-Time Password (TOTP) [18] in conjunction with the planned change to include only the hash value to prevent other network nodes to learn the subject name. This enables the subject name to function as a shared secret which might be used as part of the initialization vector for the payload key. Using for example Keyed-Hashing for Message Authentication (HMAC) [15] with the subject name and a nonce and only including the nonce in the message.

To impede the ability to pre-compute a large set of possible DHT-Keys, an unpredictably changing initialization vector can be used for the DHT-Key, for example the closing number of a stock exchange [31]. If a realm master is used, the generation of the UUID for the DHT-Key and subsequent signing of the identity of a joining node should be done by the realm master, to prevent a targeted insertion at a specific place in the DHT. Additionally a method to verify the realm master should be introduced, as most of the

security is provided by the correct function and authentication of the realm master. This could be done either through out-of-band means like a PKI or DNS entries or through the network itself by using a byzantine resistant replication protocol to obtain the identity of the realm master.

The extensions in the AAA-Tokens are currently not verifiable through the signature, as some extensions might be mutable and these changes should not invalidate the token. As some extensions are relevant to the routing properties, for example the target node, any node forwarding a AAA-Token could prevent the owner from receiving any messages through it. To mitigate this, the extensions should be a split between mutable and signed extensions to ensure that routing relevant information can be verified.

To prevent malicious peers from forging the acknowledgment (ACK) for a message, a message digest signed by the receiver could be included in the ACK. This doubles as non-repudiation for receiving a message and could be used to identify misbehaving peers, which are not properly forwarding messages.

Possible future work includes testing the feasibility of the implementation for smaller/embedded devices as overhead in CPU cycles, memory and network activity are can be prohibitive for battery powered BLE devices among others. Depending on the actual use case and its requirements for peers and authentication, an evaluation should be done whether a single realm master should be responsible for one realm, or using a quorum as the realm master might be preferable.

Bibliography

[1] R. Arends, R. Austein, M. Larson, D. Massey, S. Rose. *DNS Security Introduction and Requirements*. 2005-03. DOI: 10.17487/RFC4033. URL: https://tools.ietf.org/html/rfc4033 (visited on 2017-06-10) (cit. on pp. 13, 34).

[2] R. A. Bazzi, G. Konjevod. "On the establishment of distinct identities in overlay networks." In: *Distributed Computing* 19.4 (2007-03), pp. 267–287. ISSN: 1432-0452. DOI: 10.1007/s00446-006-0012-y. URL: https://doi.org/10.1007/s00446-006-0012-y (cit. on pp. 12, 13).

[3] M. Castro, P. Druschel, A. Ganesh, A. Rowstron, D. S. Wallach. "Secure routing for structured peer-to-peer overlay networks." In: *ACM SIGOPS Operating Systems Review* 36.SI (2002), p. 299. ISSN: 01635980. DOI: 10.1145/844128.844156 (cit. on pp. 12, 14, 33).

[4] F. Dabek, M. F. Kaashoek, D. Karger, R. Morris, I. Stoica. "Wide-area cooperative storage with CFS." In: *Proceedings of the eighteenth ACM symposium on Operating systems principles - SOSP '01*. Ed. by K. Marzullo, M. Satyanarayanan. New York, New York, USA: ACM Press, 2001, p. 202. ISBN: 1581133898. DOI: 10.1145/502034.502054 (cit. on pp. 12, 15).

[5] F. Denis. *The Sodium crypto library (libsodium)*. 2017-03. URL: https://download.libsodium.org/doc/ (visited on 2017-06-10) (cit. on p. 7).

[6] J. R. Douceur. "The Sybil Attack." In: *Peer-to-Peer Systems*. Ed. by G. Goos, J. Hartmanis, J. van Leeuwen, P. Druschel, F. Kaashoek, A. Rowstron. Vol. 2429. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 251–260. ISBN: 978-3-540-44179-3. DOI: 10.1007/3-540-45748-8_24 (cit. on pp. 12, 13, 32).

[7] V. Dukhovni. *Opportunistic Security: Some Protection Most of the Time*. 2014-12. DOI: 10.17487/RFC7435. URL: https://tools.ietf.org/html/rfc7435 (visited on 2017-06-10) (cit. on p. 12).

[8] N. Fedotova, G. Orzetti, L. Veltri, A. Zaccagnini. "Byzantine agreement for reputation management in DHT-based peer-to-peer networks." In: *2008 International Conference on Telecommunications*. 2008-06, pp. 1–6. DOI: 10.1109/ICTEL.2008.4652638 (cit. on p. 15).

[9] K. Fu, M. F. Kaashoek, D. Mazières. “Fast and secure distributed read-only file system.” In: *ACM Transactions on Computer Systems* 20.1 (2002), pp. 1–24. ISSN: 07342071. DOI: 10.1145/505452.505453 (cit. on pp. 12, 15).

[10] D. Goodin. *Firefox ready to block certificate authority that threatened Web security*. 2016-09. URL: https://arstechnica.com/information-technology/2016/09/firefox-ready-to-block-certificate-authority-that-threatened-web-security/ (visited on 2017-11-16) (cit. on p. 13).

[11] D. Goodin. *Google Chrome will banish Chinese certificate authority for breach of trust*. 2015-04. URL: https://arstechnica.com/information-technology/2015/04/google-chrome-will-banish-chinese-certificate-authority-for-breach-of-trust/ (visited on 2017-11-16) (cit. on p. 13).

[12] D. Goodin. *Google takes Symantec to the woodshed for mis-issuing 30,000 HTTPS certs*. 2017-03. URL: https://arstechnica.com/information-technology/2017/03/google-takes-symantec-to-the-woodshed-for-mis-issuing-30000-https-certs/ (visited on 2017-11-16) (cit. on p. 13).

[13] K. Hildrum, J. D. Kubiatowicz, S. Rao, B. Y. Zhao. “Distributed Object Location in a Dynamic Network.” In: *Theory of Computing Systems* 37.3 (2004-05), pp. 405–440. ISSN: 1433-0490. DOI: 10.1007/s00224-004-1146-6. URL: https://doi.org/10.1007/s00224-004-1146-6 (cit. on p. 9).

[14] P. Hoffman, J. Schlyter. *The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA*. 2012-08. DOI: 10.17487/RFC6698. URL: https://tools.ietf.org/html/rfc6698 (visited on 2017-06-10) (cit. on pp. 13, 34).

[15] H. Krawczyk, M. Bellare, R. Canetti. *HMAC: Keyed-Hashing for Message Authentication*. 1997-02. DOI: 10.17487/RFC2104. URL: https://tools.ietf.org/html/rfc2104 (visited on 2017-10-10) (cit. on pp. 32, 37).

[16] C. Lesniewski-Laas, M. F. Kaashoek. “Whanau: A Sybil-proof Distributed Hash Table.” In: *Proceedings of the 7th USENIX Conference on Networked Systems Design and Implementation*. NSDI’10. San Jose, California: USENIX Association, 2010. URL: http://dl.acm.org/citation.cfm?id=1855711.1855719 (cit. on pp. 12, 14).

[17] W. Lian, E. Rescorla, H. Shacham, S. Savage. “Measuring the Practical Impact of DNSSEC Deployment.” In: *Proceedings of the 22Nd USENIX Conference on Security*. SEC’13. Washington, D.C.: USENIX Association, 2013, pp. 573–588. ISBN: 978-1-931971-03-4. URL: http://dl.acm.org/citation.cfm?id=2534766.2534816 (cit. on p. 13).

[18] D. M’Raihi, S. Machani, M. Pei, J. Rydell. *TOTP: Time-Based One-Time Password Algorithm*. 2011-05. DOI: 10.17487/RFC6238. URL: https://tools.ietf.org/html/rfc6238 (visited on 2017-06-10) (cit. on pp. 32, 37).

[19] K. Puttaswamy, G. Swamynathan, M. Allen, B. Prof. Zhao. *Chimera Project*. 2006. URL: http://current.cs.ucsb.edu/projects/chimera/index.html (visited on 2017-05-22) (cit. on p. 9).

[20] C. Raiciu, D. S. Rosenblum. "Enabling Confidentiality in Content-Based Publish/-Subscribe Infrastructures." In: *2006 Securecomm and Workshops*. 2006-08, pp. 1–11. DOI: 10.1109/SECCOMW.2006.359552 (cit. on p. 12).

[21] E. Ronen, A. Shamir, A. O. Weingarten, C. O'Flynn. "IoT Goes Nuclear: Creating a ZigBee Chain Reaction." In: *2017 IEEE Symposium on Security and Privacy (SP)*. 2017-05, pp. 195–212. DOI: 10.1109/SP.2017.14 (cit. on p. 12).

[22] S. Sen, M. J. Freedman. "Commensal Cuckoo: Secure Group Partitioning for Large-scale Services." In: *SIGOPS Oper. Syst. Rev.* 46.1 (2012-02), pp. 33–39. ISSN: 0163-5980. DOI: 10.1145/2146382.2146389. URL: http://doi.acm.org/10.1145/2146382.2146389 (cit. on p. 15).

[23] A. Singh, T. W. Ngan, P. Druschel, D. S. Wallach. "Eclipse Attacks on Overlay Networks: Threats and Defenses." In: *Proceedings IEEE INFOCOM 2006. 25TH IEEE International Conference on Computer Communications*. 2006-04, pp. 1–12. DOI: 10.1109/INFOCOM.2006.231 (cit. on pp. 12, 30).

[24] A. Singh, M. Castro, P. Druschel, A. Rowstron. "Defending Against Eclipse Attacks on Overlay Networks." In: *Proceedings of the 11th Workshop on ACM SIGOPS European Workshop*. EW 11. New York, NY, USA: ACM, 2004. DOI: 10.1145/1133572.1133613. URL: http://doi.acm.org/10.1145/1133572.1133613 (cit. on pp. 12, 14, 15, 30).

[25] E. Sit, R. Morris. "Security Considerations for Peer-to-Peer Distributed Hash Tables." In: *Peer-to-Peer Systems*. Ed. by G. Goos, J. Hartmanis, J. van Leeuwen, P. Druschel, F. Kaashoek, A. Rowstron. Vol. 2429. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 261–269. ISBN: 978-3-540-44179-3. DOI: 10.1007/3-540-45748-8_25 (cit. on pp. 12, 30, 33).

[26] Z. Trifa, M. Khemakhem. "Sybil Nodes as a Mitigation Strategy Against Sybil Attack." In: *Procedia Computer Science* 32 (2014), pp. 1135–1140. ISSN: 18770509. DOI: 10.1016/j.procs.2014.05.544. URL: http://www.sciencedirect.com/science/article/pii/S1877050914007443 (cit. on p. 12).

[27] G. Urdaneta, G. Pierre, M. van Steen. "A Survey of DHT Security Techniques." In: *ACM Comput. Surv.* 43.2 (2011), 8:1–8:49. ISSN: 0360-0300. DOI: 10.1145/1883612.1883615. URL: http://doi.acm.org/10.1145/1883612.1883615 (cit. on pp. 12, 30).

[28] D. S. Wallach. “A Survey of Peer-to-Peer Security Issues.” In: *Software Security — Theories and Systems*. Ed. by G. Goos, J. Hartmanis, J. van Leeuwen, M. Okada, B. C. Pierce, A. Scedrov, H. Tokuda, A. Yonezawa. Vol. 2609. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 42–57. ISBN: 978-3-540-00708-1. DOI: 10.1007/3-540-36532-X_4 (cit. on pp. 12, 30, 33).

[29] C. Wang, A. Carzaniga, D. Evans, A. L. Wolf. “Security issues and requirements for Internet-scale publish-subscribe systems.” In: *Proceedings of the 35th Annual Hawaii International Conference on System Sciences*. 2002-01, pp. 3940–3947. DOI: 10.1109/HICSS.2002.994531 (cit. on p. 12).

[30] M. Young, A. Kate, I. Goldberg, M. Karsten. “Practical Robust Communication in DHTs Tolerating a Byzantine Adversary.” In: *2010 IEEE 30th International Conference on Distributed Computing Systems*. 2010-06, pp. 263–272. DOI: 10.1109/ICDCS.2010.31 (cit. on p. 15).

[31] R. Zhang, J. Zhang, Y. Chen, N. Qin, B. Liu, Y. Zhang. “Making eclipse attacks computationally infeasible in large-scale DHTs.” In: *30th IEEE International Performance Computing and Communications Conference*. 2011, pp. 1–8. DOI: 10.1109/PCCC.2011.6108091 (cit. on pp. 12–14, 37).

[32] B. Y. Zhao, L. Huang, J. Stribling, S. C. Rhea, A. D. Joseph, J. D. Kubiatowicz. “Tapestry: a resilient global-scale overlay for service deployment.” In: *IEEE Journal on Selected Areas in Communications* 22.1 (2004-01), pp. 41–53. ISSN: 0733-8716. DOI: 10.1109/JSAC.2003.818784 (cit. on p. 9).

All links were last followed on 2017-12-10.

Declaration

I hereby declare that the work presented in this thesis is entirely my own and that I did not use any other sources and references than the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted copies.

---

place, date, signature