2018

Master Thesis

“Trust Management on the Smart Grid”

Petros Panagopoulos

UNIVERSITY OF PIRAEUS

Supervised by:

Christos Xenakis (Associate Professor)

Contents

Acknowledgments

This project would not have been possible without the support of many people. Many thanks to my advisors, Dr. Xenakis Crhistos and Dr. Georgios Karopoulos for their cooperation where through their valuable advices, technological directions, discussions and support this master thesis fulfilled. Also, I would like to thank my family Kostas, Georgia and Dimitra as well as all my friends who endured this long process with me.

1. Introduction

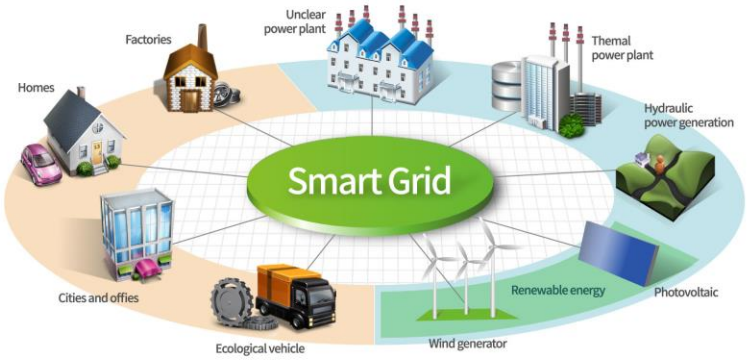

Figure 1 Smart Grid

The Smart Grid has come to modernize the traditional electric power grid with applications which contribute to more effective power distribution and management with a more reliable manner in comparison with the ageing power grids of today. According to the Department of Energy of the United States, four types of advance technology transform it into “smart”, which are the following:

1. Fully automated and integrated two way communication between the overall components of the Smart Grid.
2. Automatic Control for power distribution faults and repairs.
3. Advance Management panel, decision support software and mechanism.
4. Accurate sensing and measurement technologies.

The deployment and use of the smart grid will have great social effects and benefits since the power distribution on demand makes it more eco-friendly, saves money, assists the fault handling and mitigates the unavailability to the electric power due to grid corruptions. But since it is based on a digital networked infrastructure it should be secure and the best way to prevent any incidents is to deploy technologies, systems and networks that are designed, developed and tested to achieve secure operation.

One of the major security concerns in the smart grid is how the infrastructure will operate on a trusted basis, in order to assure its normal operation in terms of availability, integrity and confidentiality. There are numerous attacks for the smart grid which could mitigate the availability of the utility’s control systems with various consequences or to lead to incorrect billing and data theft or to affect the communication protocols that are used in the smart grid.

Let's say that a smart meter, which is an advanced meter that measures with high precision the power consumption, wants to send the data back to the utility for load balancing and billing purposes. The utility should trust the specific smart meter in order to take into account the information that receives from it, otherwise to exclude the data and take further actions for the malicious node. It becomes necessary to establish a trust management scheme on the smart grid which will facilitate the power electric utility to accumulate the data and to establish secure communications.

This master thesis deals with the problem of the key management in the smart grid in order to establish trust among the participants of the network. A standard solution of a Trust management scheme is using a Public Key Infrastructure which brings many tradeoffs due to its centralized architecture and the way that the certificates are produced and maintained. In order to maintain the operation of the system in case of a centralized PKI failure and mitigate the costs we examine the solution of the Web-of-Trust reputation concept in the smart grid which could coexist with a PKI solution. This solution is built on top of a Distributed Hash Table for efficient lookups of trust relationships and has been simulated with the Oversim P2P network simulation framework which is part of the Omnet++ simulation environment. Following, we study the concepts presented above, how they can work all together and how the solution simulated in the network simulator.

2. Security concerns on the Smart Grid

In order to spot the security issues which are associated with the smart grid, must be known the different parts of the infrastructure and how all these are interconnected. Since the smart grid is composed of computers for different purposes like smart meters and central server computers, the security issues are well-known problems in the information and communication technology domain, like buffer overflows and vulnerable implementations of cryptographic protocols as well as issues in security concepts like authentication and trust management. However the solutions of the well-known security concerns should be applicable on the resource constrained devices of the smart grid.

Smart grid incorporates two types of communication networks, Home Area Network (HAN) and Wide Area Network (WAN). A HAN connects the in-house devices with the smart meter using the ZigBee, Ethernet (wired or wireless) and Bluetooth. On the other hand, the WAN connects the smart meters, service providers and electric utility using 3G/GSM/LTE, fiber optics or WiMAX. The smart grid acts as a gateway between the two networks in order to provide the

needed information for billing and power distribution purposes. The electric utility takes the measurements and manages the power distribution within the smart grid.

The smart grid as an information infrastructure has the same security requirements as any other typical information communication system. Confidentiality, integrity and availability commonly known as the CIA Triad are the set of security principles that should be covered on the system.

- Confidentiality is the security dimension which provides secrecy on the data from different parties. It is closely linked with the privacy of the data of the end customer. Even if it is not so critical when considering grid automation systems and information it is a very important one for end consumers. Therefore, when we think about confidentiality in the smart grid we should consider privacy of consumers, power market information, etc.
- Integrity, identifies and prevents data to be modified without authorization. It is a crucial security dimension in the smart grid infrastructure for end consumers at their homes or businesses, since no unauthorized modification should happen on data related with power consumption information, or if are modified to be able to identify the modifier.
- Availability focuses in assuring the data and services are available for a precise time and specific purpose. Even if is not crucial for the smart meter, availability is very important for the smart grid power infrastructure.

Figure 2 CIA Triad

Furthermore, in the smart grid infrastructure arises the issue of the data privacy. Since smart meters will collect detailed personal data about the use of the electric devices, the end customer should be protected from any unwanted exposure of its data. With monitoring a smart meter, will be possible to determine when the apartment/building/home is in a residence and what the residents are doing. At this case, the encryption of the data transmitted by the smart meter is not adequate. Is also needed a mechanism that prevents the malicious user to understand when the home is vacant or not.

Last but not least, along with the CIA Triad, there are also two other interrelated security dimensions concerning the smart grid security which are the authentication and nonrepudiation. Both have been implicitly considered from the CIA Triad. Nonrepudiation prevents either the sender or the receiver from denying a transmitted message. Thus, when a message is sent, the receiver can prove that the alleged sender sent the message and this could be accomplished by the use of the digital signature. The authentication deals with the issue that someone or something is really who or what it claims to be. For instance, when a smart meter is requested and reports its power consumption measurements, the aggregator should be able to trust the reporting entity and to be sure that it is the one who claims to be. Authentication is one of the fundamental security concepts of the smart grid in order to build over it services that will be facilitated from the secure communication among the smart grid entities. So, this thesis studies how this could be achieved making use of security concepts like the Public Key Infrastructure and the Web of Trust mechanisms in order to mitigate any risk on trust and keep away the malicious attackers.

3. Certificate Management

3.1. Public Key Infrastructure

The Public Key Infrastructure (PKI) is a security concept which mainly provides a process that enables the users of a large and distributed setting to implement public key cryptography. It is a standard which defines a set of roles, policies and procedures to create, manage, distribute, store and restore digital certificates in order to draw an overlay where the participants are able to operate and intercommunicate on a secure and mutually trusted base.

The digital certificate identifies the information. It is like a passport which identifies a person. It is forgery resistant and since it is provided by an official and trusted agency it can be verified. What a digital certificate includes is:

- Name of certificate holder
- Serial number
- Expiration dates
- Certificate's holder public key

The digital signature is used to sign a type of information and is used to solve the problem of tampering and impersonation in digital communication. The actions that an entity should do in order to digitally sign the information are:

- Create a pair of public and private keys
- Create a hash of the information
- Encrypt the hash with its private key

Since the receiver will receive the encrypted hash, just needs the sender's public key to decrypt the hash and verify the sender.

The PKI is composed by entities with specific roles which enable services. The main services that the PKI offers are the following:

- Creates certificates associating a user's identity with a public cryptographic key
- Stores the certificates into databases
- Signs the certificates adding its credibility to the authenticity of the certificate
- Verifies the validity of the certificate
- Invalidates certificates for users who no longer are allowed access or whose private key has been exposed

With the above listed services, the PKI builds a mutually trusted network among its participants. In order to be offered such services, there are different components entitled with special roles compared with other simple nodes, which support the operation of the infrastructure.

3.1.1. Components of Public Key Infrastructure

The basic component of the PKI is the Certification Authority (CA) and optionally may exist a Registration Authority (RA).

The CA is a trusted third party with the role to authenticate entities taking part in an electronic transaction and is responsible for providing the digital certificate that contains the public key and the identity of the end entity. The final digital certificate by the CA is encrypted with the CA's private key.

The RA is an optional entity in the PKI, which supports the CA for the registration of the end entities that request a certificate. As the number of end entities increases and the location of the end entities vary a lot, the existence of multiple RAs improve the scalability of the certification request process. The RA also performs certain key management functions such as to initiate a revocation request or a key recovery on behalf of an end entity.

Let's see an example of how the certificate is issued by the CA to the end entity.

1. The end entity creates its public-private key pair.
2. Once the key pair has been created, the public key is needed to be registered with a CA.
3. Send the request of the certificate issue to the RA.
4. The RA now should check the identity of the requester and forwards the request to the CA.
5. The CA signs a certificate by using its private key and then sends the certificate to the RA.
6. The RA forwards the certificate that receives from the CA to the end entity.
7. The end entity now provides the digitally signed certificate whenever is needed to demonstrate the legitimacy of his key.

As can be seen it is not such a straightforward process which can be achieved too quickly especially if the number of the requests is too large. So, imagine the case that the trust of the smart grid is built upon a PKI solution and the CA becomes inaccessible. The certificate revocation process will cost at least a lot of time.

In order to face such a problem, the PKI could coexist with other key management schemes with no single point of failure and more flexible. For this reason OpenPGP certificates could be utilized which can be signed from multiple endorsers in comparison with PKI scheme where each certificate is signed only by the CA. Utilizing the OpenPGP standard, the certificates will be distributed among the nodes of the smart grid instead of remaining in a centralized certificate repository. With this decentralized implemented architecture, except of avoiding disruption and saving time, we avoid attacks which could cause a Denial of Service in the system.

So, let's examine the Web of Trust concept which is used in the Pretty Good Privacy encryption protocol and see how this could be integrated in our trust management scheme.

3.2. Web of Trust - Pretty Good Privacy

Pretty Good Privacy (PGP) is an open source software package for secure data communication. It provides cryptographic privacy with the use of public key cryptography and authentication with the use of digital certificates. It was invented by Phil Zimmerman in 1991 and it is the most widely used software for the encryption of e-mail. It is used for e-mail security and it is standardized with the OpenPGP standard making it an Internet standards track protocol.

PGP addresses the key distribution problem with a node's reputation scheme and shapes a ring of trust for each node. The way that the trust is built up in the PGP is not far away from the

trust in personal relationships in our everyday life. One entity can trust another entity due to a common trusted entity. This concept is also well known as Web of Trust.

OpenPGP Certificates can be digitally signed by other users who, one entity directly gives a public key to another, or the other entity fetches the first's public key from a server. Also, one entity can give a second's entity key to a third or fourth and so on. So, if the receiver is confident that an e-mail message really comes from another specific sender and has not been tampered with, the receiver will use the sender's attached public key. If the receiver trusts the sender, then may also trust the keys that gave him for other people.

In such way, can be created a chain of trust, where one entity can trust the other entity as well as the entities that the second entity trusts, as well as the others of the others and so forth. This is achieved by digital signatures where an entity signs the certificates of other entities and then can decide who is going to trust or not by the veracity of a key-and-identity combination, based on who signed the key. Moreover, PGP does not mandate a policy for establishing trust. Rather, each user is free to decide how much to trust each received key.

4. Distributed Hash Tables

4.1. Distributed Hash Table (DHT)

A Distributed Hash Table *(*DHT) is a decentralized system that provides the functionality of a hash table, i.e., insertion and retrieval of key-value pairs; a node is able to find the desired value given a key. Each node in the system stores a part of the hash table which makes him part of the storing and finding process of the information. All the nodes are connected in a structured overlay network, which enables efficient delivery of the key lookup and key insertion requests from the requestor to the node storing the key. Also, the DHT is applicable on networks with nodes arrivals and departures achieving to maintain the topology and the key-value pairs through data replication. Distributed hash tables can be applicable and effective over a large P2P network and can be utilized in a smart grid infrastructure where nodes may join and leave, the network can be really huge and depending on the particularities of the network i.e. the location of nodes, the number of nodes and their intercommunication access, different overlay protocols may be utilized in order to accomplish the most efficient overlay communication. There are different DHT protocols which specify the structure of the network and the exchange of information through node lookups like Chord, Kademlia, Pastry and many others. In our implementation we utilized the Chord algorithm in order to lookup keys in the network.

4.1.1. Chord

Chord is a distributed lookup protocol and algorithm for a peer to peer distributed hash table, which addresses the problem of efficiently locating the node that stores a particular data item. Chord specifies how the keys are stored in the hash table and the data location is achieved by associating a key with each data item and storing the key-data item at the node to which the key maps. The paper of Chord refers that the protocol supports just one operation: given a key, it maps the key onto a node.

Each Chord node needs to know routing information about only a few other nodes, which improves the scalability, and the lookups are resolved recursively or iteratively, while adapts easily in case of churn rate in the network maintaining routing information as nodes join and leave. Chord in comparison with other DHT protocols like Gnutella and Napster, avoids single points of control and failure and provides a decentralized self-organization among the nodes of the network. Also, it has a guaranteed lookup performance which does not exceed the *O(logN)* queries.

In the simulated implementation for the trust management scheme in a self-organized smart grid network the Chord selected to be utilized for the lookup operation of the digital certificate of every node through its key. But let's see how the Chord works.

4.1.1.1. Chord Protocol

As we already referred, the Chord protocol is utilized to find a value given a key. So, the basic query in the Chord protocol is to find where the desired key is located in order to retrieve the value. Every node in the Chord is responsible for some of the keys in the network and these keys are stored locally in each node. Moreover, when a node has received a request for a *key*, if has not locally any information for that key, he can forward the request to another node; its *successor*. This will lead to *O(logN)* query steps where *N* is the total number of different nodes in the network.

Chord uses consistent hashing to assign keys to the nodes and due to consistent hashing the peers join and leave the network with minimal interruption. Also, each peer gets roughly the same number of keys structuring a more balanced system and after the bootstrap phase getting to ready state each node will have information for about *O(logN)* other peers.

The successors for each node are stored in its finger table, which contains up to *m* entries, where *m* is the number of bits of the hash value. The size of *m* determines the identifier space in the chord, where an identifier refers to a specific node. All identifiers are ordered in an identifier circle modulo 2^m and each key is assigned to the first node whose identifier is equal or follows *k.* So, an identifier circle consisting of three nodes three nodes 0, 1, 2 and the value of *m is 3*, is formed as can be seen in Figure 3.

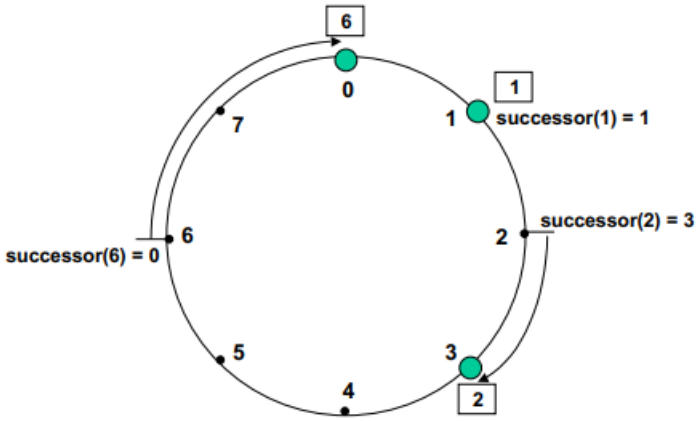

Figure 3 Identifier Circle with 3 nodes

After calculating the successors of key for each node, queries can be passed around this circle until they first encounter a node that succeeds this identifier, in case that this information cannot be retrieved locally from the current node. So, in the Figure 3 the successor of identifier 0 is the 0, of 1 is the identifier 1 and the successor of 3 is the identifier 0.

The finger table of each node is calculated with the definitions of the Table 1.

Notation	**Definition**
Finger[k].start	$(n+2^{k-1}) \bmod 2^m$, $1 \le k \le m$
.interval	*(finger[k].start, finger[k+1].start)*
.node	*First node ≥ n.finger[k].start*
Successor	*The next node on the identifier circle; finger[1].node*
Predecessor	*The previous node on the identifier circle*

Table 1 Definition of variables for node n, using m-bit identifiers

Finger[k] → the first node that succeeds the identifier

Successor → the next node from the node in question on the identifier ring

Predecessor → the previous node from the node in question on the identifier ring

For example, the finger table of node 1 points to the successor nodes of identifiers $(1+2^0) \bmod 2^3 = 2$, $(1+2^1) \bmod 2^3 = 3$, $(1+2^2) \bmod 2^3 = 5$. The same process is done for the node 1 and 3 which leads to the finger tables of the next image:

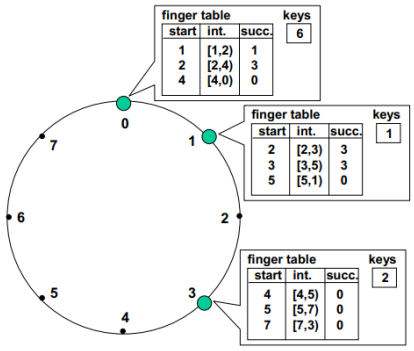

Figure 4 Finger Tables

When a new node joins the network then the successors in the finger table of each node have to be recalculated through the Chord stabilization process, in order to mirror the new status of the network and to lead to successful queries.

In order to find the *successor* node of an id:

```
// ask node n to find the successor of id
n.find_successor(id)
      if id ∈ (n, successor]
            return successor;
      else
            // forward the query around the circle
            n0 = successor.closest_preceding_node(id);
            return n0.find_successor(id);


// search the local table for the highest predecessor of id
n.closest_preceding_node(id)
      for i = m .. 1
            if (finger[i]∈(n, id))
                  return finger[i];
      return n;
```

If a node joins or leaves the network what should be preserved is the ability to locate every key in the network. To achieve this goal, Chord needs to preserve that each node's successor is correctly maintained for every key k, node successor(k) is responsible for k.

The Chord protocol is the DHT protocol among others that provides provable performance and correctness, can be applied in real time systems as the smart grid requires.

4.1.2. Kademlia

Kademlia is another distributed hash table implementation for peer to peer networks. The Kademlia network is made up of a wide range of nodes, communicating each other through User Datagram Protocol (UDP). As any other DHT algorithm, uses key-value pairs in order to store the data and the key is the identifier to find the value in the network. In comparison with Chord, uses Tree-based routing but the goal remains the same; given a data item X which is stored at some set of nodes in the network, find its location.

Each node in the network is identified by a unique number called node ID. The routing tables are structured so that the participating nodes have detailed knowledge of the other nodes close to themselves. Kademlia employs a recursive algorithm for node lookups which is the most important operation of a Kademlia participant to locate the k closest nodes to some others given a key. In order to find the "closest" nodes a bitwise XOR operation is used to measure the distance among the nodes, to route queries and locate nodes in the network; *distance (a, b) = a XOR b.*

The lookup initiator starts by picking n nodes from its closest nodes and then sends parallel and asynchronous requests to the nodes, which has the benefit that avoids timeout delays from failed nodes. Nodes that fail to respond quickly are removed from consideration until and unless they respond. The key lookup process is designed to return the k closest nodes to the target id. Because of its decentralized structure, Kademlia builds a strong defense against a denial of service attack. Its decentralized structure is equally advantageous when the nodes become flooded by messages. Nodes, files and key words deploy SHA-1 hash into a 160 bits space.

Since it uses a binary tree routing, Kademlia treats nodes as leaves of a tree. When is looking for a node, divides the binary tree into a series of subtrees that don't contain the node.

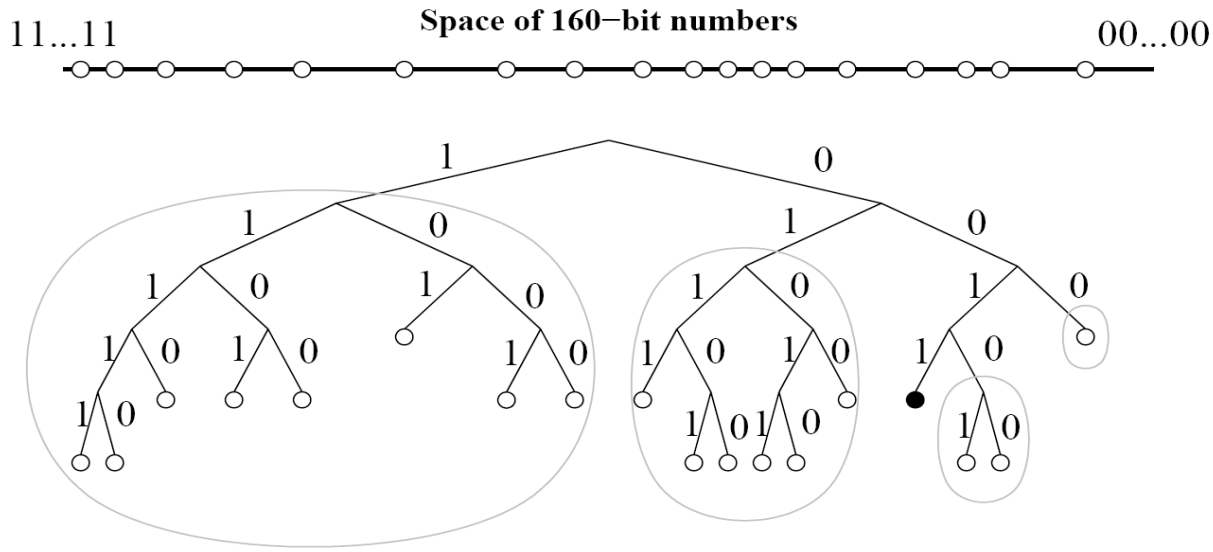

Figure 5 Kademlia routing tree

Kademlia is widely employed in file sharing networks because it makes it quite easy to search for information in file-sharing networks. This is because the keywords are used for searching file names, and each file name is divided into its basic words. Such special keywords are diced and put in network storage, along with their corresponding file hash and file name. Public networks that are based on the Kademlia network algorithm include among others the BitTorrent, Gnutella and Overnet.

4.1.3. Pastry

Pastry is another implementation of a distributed hash table, used for building a self-organizing decentralized overlay network, where each peer receives and forwards requests. Each peer in Pastry is assigned a 128-bit peer identifier which is called the NodeID.

The position of each node in the network is based on the NodeID which could be from 0 to 2^{128} - 1. When a node joins the network, is assigned randomly a NodeID which is uniformly distributed in the 128-bit space. For a network of N peers, Pastry routes to the numerically closest peer to a given key in less than $\log_B N$ steps under normal operation (where B = 2b is a configuration parameter with typical value of b = 4). The NodeIDs and keys are considered a sequence of digits with base B. Pastry routes messages to the peer whose NodeID is numerically closest to the given key. A peer normally forwards the message to a peer whose NodeIDs share with the key a prefix that is at least one digit (or b bits) longer than the prefix that the key shares with the current peer NodeID as it can be seen in the Figure 6.

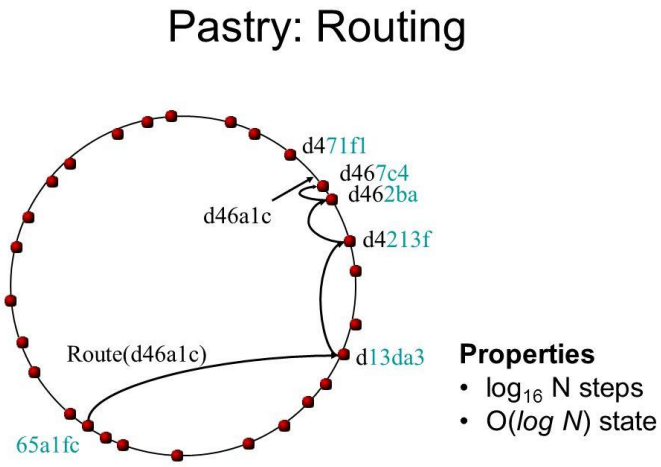

Figure 6 Pastry routing

Pastry, since it is a decentralized protocol there is no single point of failure and any single node can leave the network at any time without warning and with little or no chance of data loss. Also, the protocol utilizes external software applications for metrics such as ping or traceroute in order to determine the best routes to store in its routing table.

4.1.4. High Level DHT Algorithms Comparison

4.1.4.1. DHT Commonalities

The DHT algorithms that we have already referred (Chord, Kademlia and Pastry) have some commonalities regarding how they function in the distributed hash table environment. These commonalities are:

- They consume the same time in order to determine if a routing entry node has failed.
- The lookup process can be performed either iteratively or recursively and the lookup messages can be forwarded either sequentially or in parallel.
- The algorithms support replication of the data.
- Each peer of the network has knowledge for some of his neighbor nodes.

4.1.4.2. DHT Protocol Operations

In this section, we describe DHT operations. The major DHT operations are join, leave, insert, lookup, remove, keep-alive and replicate which a node participating in a DHT may initiate.

4.1.4.2.1. Join

For the join operation, a node initiates the join requesting from a peer already in the DHT to insert itself in the network. The mechanism to discover a peer already existed in the DHT is independent of the specific DHT algorithm. After joining the network, the information should be distributed in the rest network and the structures that help each node to find the others to be updated.

A joining node may want to build its routing table by getting a full or partial copy of its neighbors or any appropriate node's routing table. It will also need to obtain key/value pairs it will be responsible for.

Following is the list of information that will be exchanged between the newly joining node and existing peers:

- An overlay ID.
- The joining node’s ID.
- Contact information or IP address of the joining node.
- Indication whether this peer should be inserted in the p2p network thereby changing the geometry or merely stored on an existing peer.
- Full or partial routing table of an existing node.
- List of neighbors.

This information is sent by a peer of the network in response of the message it sent.

4.1.4.2.2. Leave

The leave operation is very important to be completed normally when the node leaves the network in order to maintain the stability and to mitigate failures in lookups. So the leaving node should gracefully inform the other nodes about his departure. Sequentially, the other nodes should update their pointers and take over the keys that the leaving node is responsible for. So, the node should inform about:

- Its key
- The list of key/value pairs that is responsible for
- Acknowledgement that the node has been removed from the DHT.

4.1.4.2.3. Insert

A node initiates an insert operation to a peer already in the DHT to insert a key/value pair. The insertion involves locating the node responsible for key using the lookup operation and then inserting either a reference to the key/value pair publisher or the key/value pair itself. The insert operation is different from the join operation in the sense that it does not change the DHT topology. The insert operation can also be used to update the value for an already inserted key.

- Key for the object (value) to be inserted.
- Data item for the key (or value). A sender may choose to only send the value of the key in the insert operation after the node responsible for the key has been discovered.
- A flag indicating whether the lookup should be performed recursively or iteratively.
- Publisher of the key. Multiple publishers can publish data under the same key and a node storing a key/value pair uses this field to differentiate among the publishers.
- Key/value lifetime. The time until an online peer must keep the key/value pair. The publisher of the key/value pair must refresh it before the expiration of this time.

4.1.4.2.4. Lookup

The lookup operation will be utilized by each node in order to retrieve a key/value pair from the DHT network. It locally applies DHT routing metric (Chord, Pastry or Kademlia) on its routing table to determine the peer to which it should route the message. The peer responsible for the key/value pair sends it directly back to the querying node. The value can be an IP address, or a more complex record.

The lookup message can be routed iteratively or recursively as we have already referred. A node routing a recursive query, may add its own key and IP address information in the

lookup message before forwarding it to the next hop. Following is the list of information exchanged between the nodes and the peer holding the key/value pair.

- A key to lookup.
- A flag indicating that the lookup will be performed recursively or iteratively.
- Each node will forward peer's key and IP address to the other. A node in the path of a lookup query may add its own ID and IP address to the lookup query before recursively forwarding it. This information can be used by the querying peer to possibly update its routing tables.
- A data item for the key that could not be found. If the lookup is iterative, this value also indicates whether the lookup is complete or whether the node should send the query to the next-hop peer returned in this message.

4.1.4.2.5. Remove

Each stored key/value pair has a lifetime associated with it which will expire unless refreshed by the publishing node in time. There is also the case that the publishing node may want to remove the key/value pair from the DHT before its lifetime expiration. The information that should be exchanged for the remove process is:

- Publishing node's key.
- Key for the key/value pair to be removed.
- Acknowledgment that the key has been removed.

4.1.4.2.6. Keep-alive

In the DHT there is a process that one node informs its neighbors about its existence. This is called keep alive. The two immediate neighbors do not need to send a periodic keep-alive message to each other. The peers can use various heuristics for keep-alive timer such as randomly sending a keep-alive within an interval. If a neighbor fails, a peer has to immediately find another neighbor in order to ensure lookup correctness. The actions included in the keep-alive process are:

- Sending node's key.
- Keep-alive timer expiration.

4.1.4.2.7. Replicate

As a backup process in the DHT, the node should replicate the keys that is responsible for. Heuristics such as replicate to the next k nodes can be applied for this purpose. A node may also need to replicate its keys when its neighbors are updated.

But let summarize some of the details of the already referred DHT overlay networking algorithms:

	Look up	**Store**	**Join**	**Keep-Alive**	**Update Routing Table**	**Update Neighbor Nodes**
Chord	find_successor()	NA	join()	fix_fingers()	fix_fingers()	Stabilize()
Pastry	NA	NA	Side effects of lookup	NA	On Demand	NA
Kademlia	ping	Store()	NA	ping	NA	NA

Table 1 DHT RPCs

	Key-Length	**Sequential/Parallel**	**Recursive /Iterative**	**Routing Table Name**	**Neighbor Nodes**	**Routing Data Structure**	**Routing Table row selection**
Chord	160	Both	Sequential	Finger Table	Successor List	Skip-list	Immediately succeed the interval
Pastry	128	Recursive	Sequential	Routing Table	Leaf set	Tree-Like	Any node in the interval
Kademlia	160	Iterative	Parallel	Routing Table	None	Tree-Like	Any node in the interval

Table 2 DHT details

5. Simulation

The simulation conducted in order to help us to evaluate the Web of Trust concept as applied solution on the smart grid, in order to deal with the case that a centralized infrastructure fails or is not available. The evaluation is based on metrics related with the feasibility and the performance of the solution. Feasibility and performance in matter of trust accomplishment, delays etc. For the simulation, the Oversim P2P network simulation framework was utilized in order to build a pretty good privacy like algorithm on a Chord DHT overlay network topology and build the trust among nodes in the network.

The Ovesim P2P network simulation framework is specialized in simulations for peer-to-peer networks. It provides a wide range of DHT overlay network algorithms, from which you can choose regarding the needs of the experiments. Among them there are the Chord, Kademlia, Patry, Bamboo and others. It supports implementations of multiple tiers which you can modify or to build up on them your application since Oversim was designed as a modular simulation framework. An overview of its architecture is illustrated in Figure 7 Oversim Architecture.

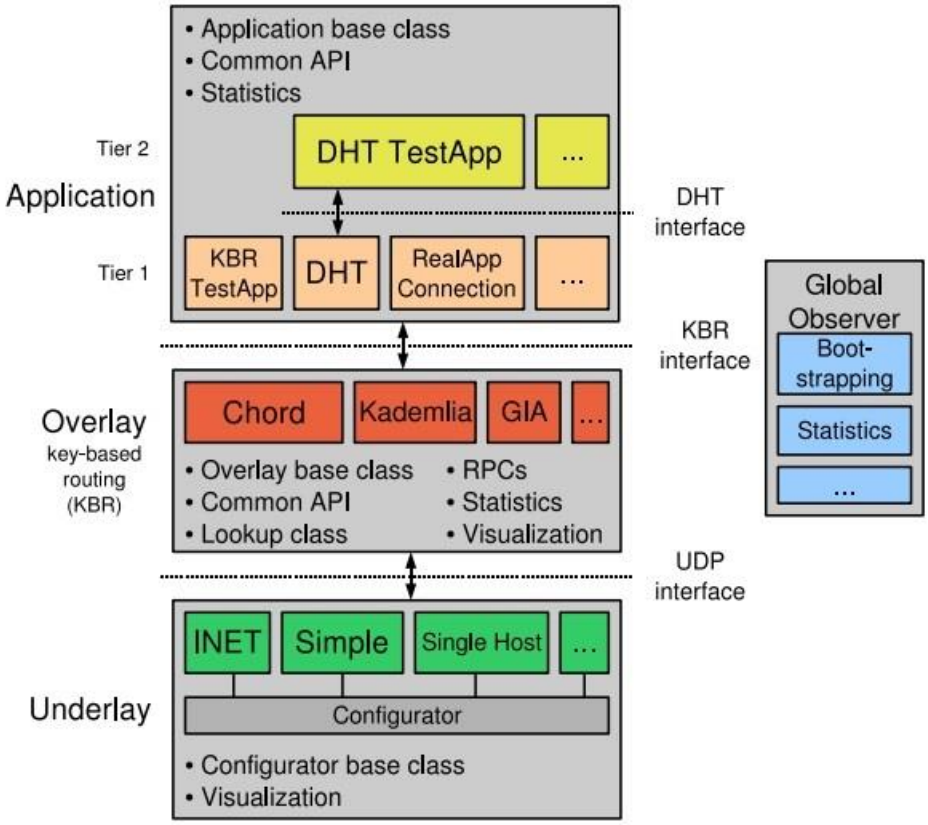

Figure 7 Oversim Architecture

- Underlay – emulates the physical network, the Internet and the ad-hoc networks. OverSim enables the simulation of network topologies with realistic bandwidths, packet delays, and packet losses. It supports three types of underlays:
    - Simple underlay – A simple 2D topology where nodes are randomly placed on a plane. Inter-node distance is based on Euclidian distance between 2 nodes and latency is proportional to distance.
    - INET underlay – Enable simulation of complex topologies with multiple Autonomous Systems (ASs) and stubs (transit-stub), link bandwidth, delay, packet loss, etc.
    - Single host – Acts as an actual node that supports TCP and UDP. Used to communicate with other nodes as a typical P2P client.
- Overlay – Structured and unstructured protocols are implemented at this layer.
- Tier 1 – Use to implement typical P2P services such as data indexing (e.g., Distributed Hash Tables (DHTs)), publisher-subscriber, caching, replication, etc.
- Tier 2 – Use to implement P2P application such as file sharing, name services, streaming, gaming, distributed data processing, etc.

Oversim uses a discrete event simulation to simulate the exchange and process of the network messages. New messages can be included to support a custom experiment. On the Application layer (Tier2) can be built a custom application and this is the layer where we developed our own algorithm, a pretty good privacy like algorithm which interfaces with the DHT module, on top of the Chord overlay network. The communication between different layers is accomplished through Remote Procedure Calls. Oversim provides configuration files where you can set values for various settings for the simulation which can affect the network performance; e.g. enabling delays, time to live of a packet, drops etc., the DHT selected protocol and the overlay topology; e.g. number of nodes, replication as well as settings of the custom application.

5.1. Architecture

The simulation is based on the architecture proposed in the paper [17] “Self-Organised Key Management for the Smart Grid” which is built on top of a Distributed Hash Table for efficient lookups of trust relationships. An abstract of the architecture can be seen in FigureFigure 8 Self Organized Key Management Architecture:

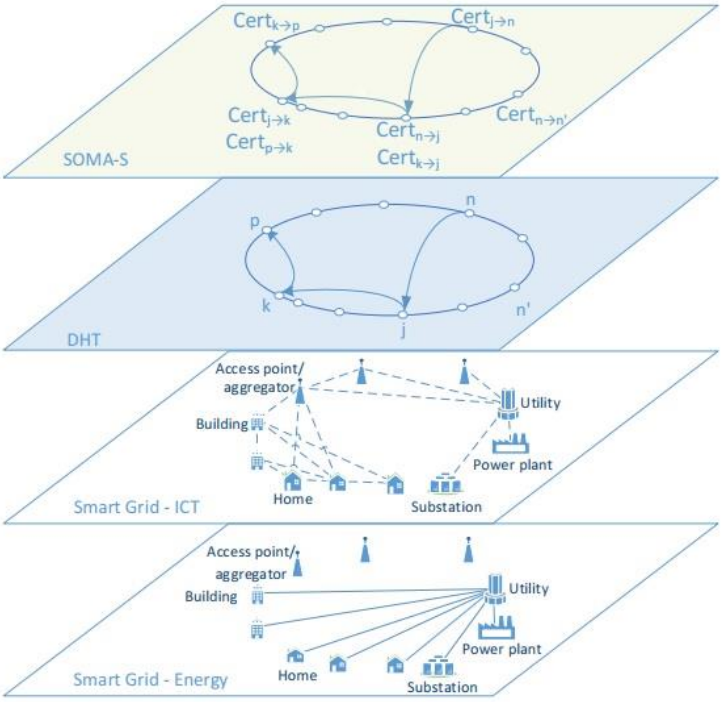

Figure 8 Self Organized Key Management Architecture

The lowest level demonstrates the physical power delivery network over which the utility company delivers power to each customers. Above of it, there is the Information and Communication Technology network which enables the bidirectional communication among the nodes of the smart grid network. There are homes, buildings, access points, aggregators, utilities most of them equipped with smart meters for remote access control. The rest two levels provide the support of the key management operations with the exchange of the certificates in order to set up trust among the nodes of the smart grid and make the system resilient in various failovers of established centralized structures and concepts in the smart grid architecture.

In more detail, every node appears in the levels above the ICT level is a participant in the smart grid. Every node is uniquely identified by an ID which is derived by a hashing its public key and a device identifier. So, node n has the ID_n where $ID_n = hash(PK_n + ID_{device})$. This ID in the DHT serves as the key through which you can find the desired value.

The Web or Trust concept is based on a reputation scheme where a node A can trust another node B if the node C trusts node B and node A trusts node C, so node A can trust node B; A and B have a common node to trust. A node of the smart grid will try to find the value of a specific key. The value will be the quantity that will make him decide if is able to trust the other party or

not. This value corresponds to the certificate of the node where each node's certificate has been signed by other nodes in the smart grid. So, each node will make requests to other nodes in order to retrieve the value and examine through the certificate's signatures if can trust the node or not.

When a node can trust another node? Let's say that the node A, in order to trust a node B, should find in B's certificate signatures the signature of A. So, at some time node A received the node's B certificate and signed it. This is a direct way of trust between A and B. Otherwise, the node A in order to decide if he can trust the node B will try to find its own signature to the certificates of the nodes that already have signed the certificate of node B, which is an indirect way of trust.

5.2. Implementation and Results

In terms of Oversim, the already presented architecture is structured by utilizing the following parts:

- Underlay – Simple Underlay topology selected where the nodes are randomly placed in the network
- Overlay – the Chord DHT algorithm selected for efficient lookups of key/values, network stabilization and effective network join procedure.
- Application tier1 – the DHT module selected which provides intercommunication with the overlay and application tier2
- Application tier2 – Our custom implementation using and extending the DHTTestApp module.

All the operations related to the Web of Trust - PGP concept are implemented on the Application tier2. This is the basement of each node from which starts to build its trust ring. Sends requests for certificates, receives and processes the responses in order to end up with its trust ring and finally to export statistics and metrics for the whole trust operation.

Firstly, when the simulation starts there is the bootstrap phase and the node join operation. The chord has an initialization phase for each node which goes through three different states:

- Init state which initializes:
    - Node's predecessor pointer
    - Node's finger table

    - Node's successor list
- Join state
    - Initiates the join process
    - Initiates join timers
    - Finds a new bootstrap node and enrolls it to the bootstrap list
- Ready state
    - All the operations of the previous states have been accomplished and the node has been characterized to be in ready state in the network

5.2.1. Node Join

The join process of a node as described above includes operations like fixing the node's finger table, the successor and predecessor list as well as actions for the bootstrap. Experiments conducted with up to 5000 nodes and the join delay of each node can be seen in Figure Figure 9 Join Delay of node in the Chord overlay network

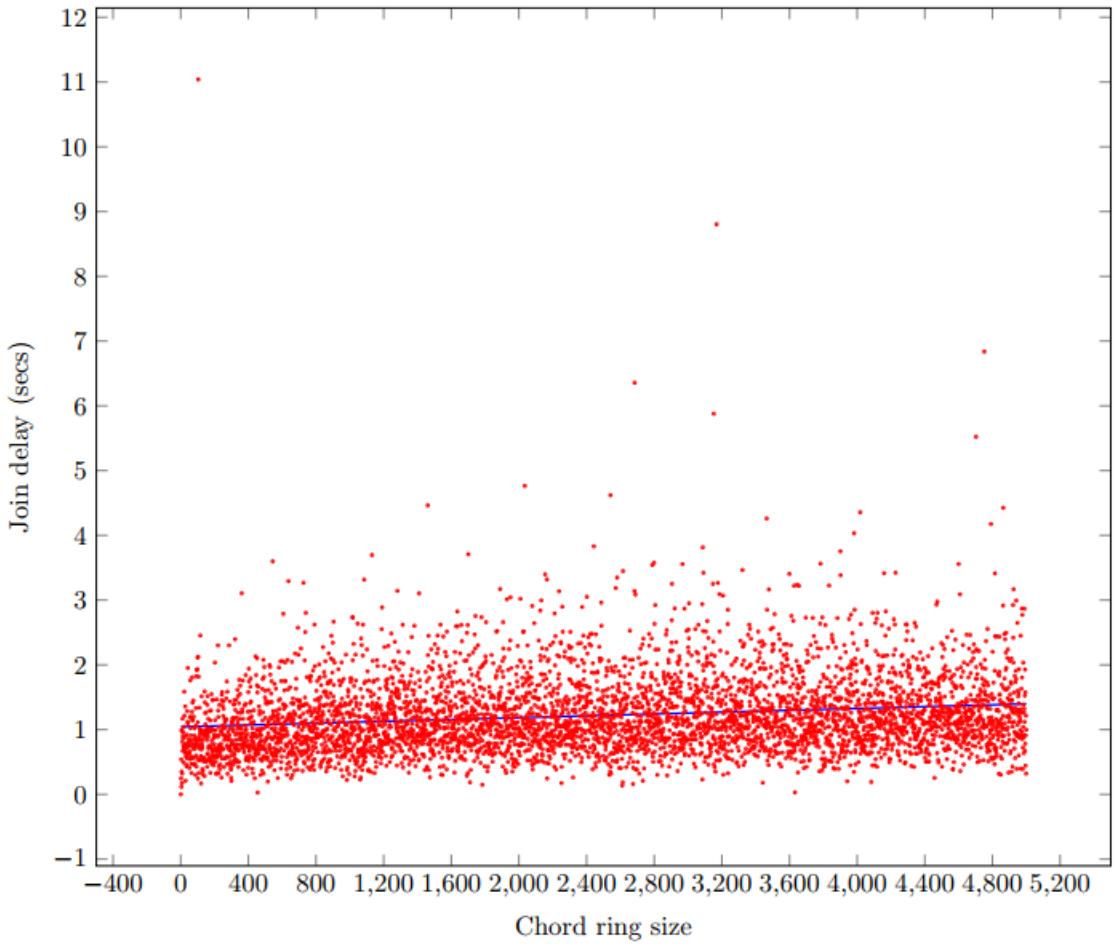

Figure 9 Join Delay of node in the Chord overlay network

The simulator is configured to introduce a node every one second. We can observe that for the node majority the join delay is kept under 2 seconds and the average is 1.23 seconds. Another observation that we could make is that as the network grows and more nodes are added the delay linearly increases but not with high rate. The performance of node join is based on Chord specific operations, since at this phase no extra operations of the custom algorithm have been added.

5.2.2. Trust of Chain

At this phase we want to build a Web of Trust scheme among the nodes of the network, where a node will trust another node either directly or indirectly. So, after the initialization phase,

each node sends to the network its key along with its certification. The DHT and the Chord are responsible to handle and store the key, along with the value, through their operations and layer intercommunication to the node that will be responsible for specific keys. The node that will receive the certificate from another node will sign it, since has not signed it already. At the end of the signing process, each node's certificate has some signatures from other nodes and this is how is built a chain of signatures for each certificate which leads to a high or low reputation depending on how many different signatures the certificate has acquired.

After acquiring digital signatures for each node's certificate, the node should start building its trust chain and determine which node can trust or not. This is accomplished by periodical requests; each node requests the certificate from other nodes and upon getting the certificate examines the signatures of this certificate. If the node will find its signature in the signature list, then he decides that the node can be trusted and inserts the node in its trust ring. At this case we have a direct type of trust and the trust chain length is equal to 1 since has not gone deeper in other nodes that have signed the initial requested node. Otherwise, if the node's signature is not in the list of signatures, then will start checking the certificates of the nodes that have signed the specific certificate, in order to find its signature at one of these certificates and determine if the initial requested node can be trusted or not. A new node will be requested for its certificate if has not been requested already.

The trust chain length grows as we go deeper in the search of nodes of node that have been initially requested. For example, if a node will not find trust at the initial requested nodes, will start requesting firstly the nodes that have signed the initial requested node's certificate, If he finds trust from these node then the trust chain length equals to 2. Otherwise the search can go deeper and start requesting the certificates of the nodes from the nodes of chain length 2 and so on. An abstract sequence diagram can be seen on FigureFigure 10 Searching for trust.

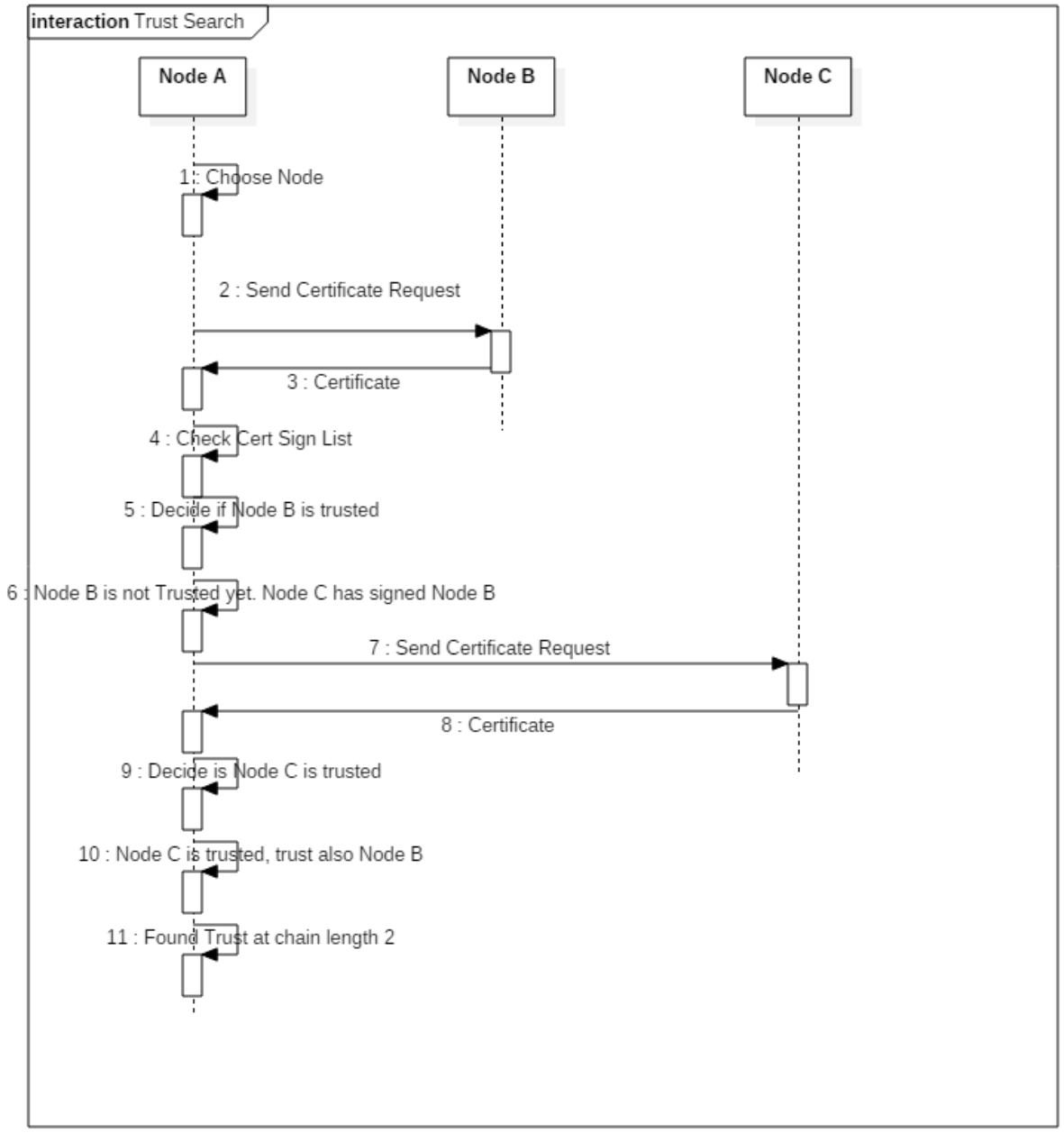

Figure 10 Searching for trust

Various experiments conducted with different number of nodes, from 100 to 5000 in order to find out the trust chain length that the nodes converge to. How deep should a node search in order to find trust for the initial requested node and how long would it take? Each node was configurable to make a number of requests to the other nodes. Since the network was

consisting by small number of nodes, each node was easy to request the certificate from each one node.

Another aspect of the problem of finding trust is to what extent a node should scan the network. If he does not have any information about where or at least at which neighborhood of nodes could find easier trust, what is the percentage of nodes of the total network that should make requests in order to find trust? In our experiments the nodes that are requested are chosen randomly. When the network consists of small number of nodes, e.g. till 500 nodes the whole network can be easily scanned and with an extent threshold of possible chain length threshold. For larger networks that we have experimented on the network simulator with number of nodes from 800 till 5000 the choice was to scan a part of the network and through measurements of delays and trust accomplishments to determine the feasibility of the possible solution. Results and evaluation are going to be presented in a related paper publication.

6. Related work

There are several propositions in the literature related with trust management among the nodes of a peer to peer and ad hoc networks. Even if the PKI solution is the most suitable for the identity management in a network, the smart grid comes with its particularities that will drive the PKI scheme to adapt to its needs or create an alternative solution concept.

Researchers have already put into a lot of interest and research about the trust establishment in mobile ad hoc networks (MANETs) which introduce concepts that someone could inspire in order to apply on the smart grid. Even if ad hoc networks have no specific structure and special situations can occur, the smart grid can also present similar characteristics, since new nodes will be being introduced in the period of time, or will be depart due to failures and power blackouts. Moreover, one could examine MANETs' trust management solutions and apply them on the smart grid applications, since the smart meters may lose connectivity with the centralized trusted authorities and continue to operate as an ad-hoc network.

In a research developed by Hubaux et al., they propose a self-organized solution for key management in ad-hoc networks similar to PGP Web of Trust, using public key certificates and issuing them to other users through a certificate graph. Each node stores the certificates of other nodes and the evaluation is done after merging the repositories with their certificates.

Another scheme is the Distributed Certificate Authority (DCA) for ad-hoc networks based on threshold cryptography. The DCA, consists of an arbitrary number of server-nodes and combiner nodes. Threshold cryptography implements computations that can be performed in a

distributed way by a trusted subset of parties. In more detail, in threshold cryptography the private key is divided to *n* parts which are shared among a group of receivers. Only authorized set of users can decrypt messages. Afterwards a group of any k participants can cooperate to reconstruct the shares. No less than k participants will be able to reconstruct it.

7. Conclusion and Future Work

The smart grid, an upcoming technology infrastructure with special characteristics and requirements should be examined also in a special way, in order to apply security concepts that would protect the infrastructure and the users. In our simulation, we experimented with the PGP Web-of-trust scheme for the trust management in order to examine the feasibility and the performance of the algorithm in small and large networks of the smart grid nodes, the trust that they succeed and the delays that are introduced. We utilized the Oversim P2P network simulator and its implemented algorithms for the DHT - Chord in order to build on that our custom implementation of a PGP Web-Of-Trust like algorithm. The Oversim as part of the Omnet++ simulator is a wide used P2P network simulator and a lot of research has been made using the simulator. Also, the implementation and results of the Chord in the Oversim has been evaluated from researchers and compared with real world implementation with very good results [20].

Regarding the future work on the trust management issue on the smart grid, could be the experimentation with DHT protocols in combination with introducing some added knowledge regarding on which set of nodes- in geographical aspect of an overlay network- would be possible to find easier trust and mitigate with random or the extensive search of the network. Another possible future work is the research regarding trust management vulnerabilities and defense mechanisms that could mitigate the risks of the trust management.

8. References

1. http://www.computerweekly.com/blog/Quocirca-Insights/Securing-the-Internet-of-Things-time-for-another-look-at-Public-Key-Infrastructure-PKI
2. https://tools.ietf.org/html/rfc5280
3. http://www.oasis-pki.org/pdfs/PKI_Basics-A_technical_perspective.pdf
4. https://www.safaribooksonline.com/library/view/understanding-pki-concepts/0672323915/ch06.html
5. Callas, J., Donnerhacke, L., Finney, H., Shaw, D., Thayer, R.: OpenPGP Message Format. RFC 4880 (Proposed Standard) (Nov 2007) http://www.ietf.org/rfc/rfc4880.txt, updated by RFC 5581
6. https://www.openpgp.org/
7. https://www.ietf.org/mail-archive/web/smartpower-interest/current/pdfW8bbGqjK7Z.pdf
8. https://www.enisa.europa.eu/topics/critical-information-infrastructures-and-services/smart-grids/smart-grids-and-smart-metering/ENISA_Annex%20II%20-%20Security%20Aspects%20of%20Smart%20Grid.pdf
9. https://smartgridawareness.org/2017/02/25/smart-meters-profile-consumers/
10. https://smartgridawareness.org/privacy-and-data-security/smart-grid-vulnerabilities-a-more-detailed-review/smart-grid-security-threats-vulnerabilities-and-solutions/
11. “International Journal of Smart Grid and Clean Energy”: http://www.ijsgce.com/
12. “Handling Churn in a DHT“ https://www.usenix.org/legacy/event/usenix04/tech/general/rhea/rhea_html/usenix-cr.html
13. Stoica, I., Morris, R., Liben-Nowell, D., Karger, D.R., Kaashoek, M.F., Dabek, F., Balakrishnan, H.: Chord: a scalable peer-to-peer lookup protocol for internet applications. IEEE/ACM Transactions on Networking 11(1), 17{32 (Feb 2003)
14. http://searchsecurity.techtarget.com/definition/digital-certificate
15. Salman Baset, Henning Schulzrinne, Eunsoo Shim, A Common Protocol for Implementing Various DHT Algorithms
16. http://www.cs.unc.edu/~jasleen/Courses/Fall09/slides/8b-DHT-Pastry.pdf
17. Foivos F. Demertzis, Georgios Karopoulos, Christos Xenakis, and Andrea Colarieti, “Self-Organised Key Management for the Smart Grid”
18. Dawoud D.S., Richard L. Gordon, Ashraph Suliman and Kasmir Raja S.V. “Trust Establishment in Mobile Ad Hoc Networks: Key Management“
19. Srdjan Capkun, Levente Buttya and Jean-Pierre Hubaux “Self-Organized Public-Key Management for Mobile Ad Hoc Networks”

20. Jamie Furness, Mario Kolberg, Marwan Fayed “An Evaluation of Chord and Pastry Models in OverSim”