SOMA-E: Self-Organised Mesh Authentication - Extended

F.F. Demertzis, C. Xenakis

University of Piraeus, Department of Digital Systems, 80 Karaoli Dimitriou Street, PC 18534, Piraeus, Greece

Abstract

Community mesh networks have emerged rapidly in every metropolis around the world, however many of the security methods applied are counter-intuitive and usually disrupt the autonomous characteristics of the mesh nodes. In SOMA we present a structured Peer-to-Peer solution providing authentication service based on a scalable, self-organized and fully distributed Web-of-Trust. Our proposal is a hybrid Public Key Infrastructure build on top of Chord, allowing each agent to place its own trust policy while keeping the autonomous characteristics the nodes intact. Our goal is to create a large-scale authentication system for mesh networks without the need of a Trusted Third Party. We leave the decision of whom to trust in each agent independently taking advantage of the overlay to alleviate the shortcomings of traditional Web-of-Trust models. This is achieved by using the overlay as a meta-structure to infer trust relationships providing a policy-based system, which is further enhanced with a Bayesian reputation based model so as to cope with the different challenges posed by the distributed nature of the system. The possible attacks and limitations of our proposal are also investigated and discussed.

Keywords: Mesh Networks, Public Key Infrastructure, Web-of-Trust, Peer-to-Peer, Autonomous Authentication

1. Introduction

Community mesh networks (CMNs) have emerged in every metropolis around the world, where the comprising nodes share resources and services in an autonomous fashion. These mesh networks are based on independent, computational powerful, wireless ad-hoc, multi-hop nodes. Such networks comprise of a decentralised infrastructure that can scale to many thousands. Similarly to traditional networking, identity management is the cornerstone for the establishment of trust and cooperation. For effective communication between the

Email addresses: fdemertz@unipi.gr (F.F. Demertzis), xenakis@unipi.gr (C. Xenakis)

nodes in a CMN a node has to bind deterministically a unique identity and a set of corresponding attributes to each of the communicating parties, and therefore mapping their names and their services correctly.

In a CMN comprising of N nodes it would require an effective mechanism for adding, publishing and generally managing the peer identities. More specifically, the nodes that wish to communicate securely, initially need to establish some form of trust between them since they are untrusted. Furthermore, this mechanism would provide secured authenticated channels to share data, publish services etc, among peers. In such networks credentials are usually exchanged in a bi-directional and symmetric fashion between two or more parties. How would a given node j, $j \in N$, know if the credentials of a node k, $k \in N$, are correct and correspond to the true identity of k? How can it tell if a malicious node hijacked k's identity or someone eavesdrops on their communication? An infrastructure such as this can take the form of a protocol that handles the distribution of the identification credentials and a trust management mechanism to ensure correctness and authenticity of the peers and their exchanged data e.g., based on a cryptographic public key scheme. Therefore, our goals are firstly to provide an infrastructure for the nodes to communicate securely and secondly to be able to reason about corresponding trust relationships in a CMN decentralised environment. Most CMNs are open in nature and each peer individually decides what services to offer in the mesh. Therefore, our authentication mechanism has to impose no barriers to entry and, in addition, keep the autonomy of the nodes intact. In this work, we focus on fully or semi-planned, long-term wireless ad-hoc networks similar to the many open community metropolitan mesh networks. The nodes use wireless multi-hop transmissions to communicate, based on IEEE 802.11 and are computationally strong enough, such as the ones found at the edges of community WMNs (e.g., AWMN [1], MIT's Roofnet [2]).

1.1. Motivation

Currently, many security models have been proposed for mesh networks, but most of them are counter-intuitive and usually disrupt the autonomous characteristics of the mesh nodes. The use of traditional Public Key Infrastructure (PKI) approaches is fundamentally incompatible with the decentralised open nature of the mesh network's topology, as they delegate trust either to a Single Certification Authority (CA) or to replicated CAs which are not viable for CMNs. This is because the former is not a scalable architecture and the latter, introduces increased security risk, by providing additional attack vectors. More specifically, an adversary may gain access to the CA's private key by compromising even a single replicated CA.

Many other solutions use empowered nodes which are involved either in the organisational aspects of the network or its trust management. Implementing such solutions for mesh networks (that scale to many thousands) would inevitably lead to inefficiency, bottlenecks and points of failure. By delegating trust decisions to an external third party, distributed or not, we would inevitably lose the individuality and autonomy of the mesh nodes. Hence, we

propose SOMA a system that has self-organization, autonomy and scalability as its principal design co-ordinates.

1.2. Our Contribution

SOMA is a certificate-based authentication infrastructure that aims to create a large-scale secure authentication system for mesh networks without the need of a Trusted Third Party (TTP). We aim at creating a self-organised, efficient and scalable authentication infrastructure, without sacrificing the autonomous characteristics of the nodes. This paper is a revised and extended version of our previous work [3]. Hence, our work focuses on building on top of a self-organised, structured Peer-to-Peer, Web-of-Trust [4] infrastructure that leaves the decision of whom to interact with and why to place trust on each of the agents, independently. It provides a policy-based system, which is further enhanced with a Bayesian reputation based model so as to cope with the different challenges posed by the distributed nature of the system. It is also designed in that way so as to be extendible to any number of relationships and metrics.

In SOMA, we make use of structured Peer-to-Peer in contrast to unstructured flooding based protocols, e.g., Gnutella [5], due to the fact that the nodes are mostly static. Therefore, using a structured Peer-to-Peer approach, such as Chord [6], a node is given access to a scalable holistic view of the mesh. In addition, building an efficient Web-of-Trust infrastructure, which exploits collective knowledge in a self-organised way, fits better to the studied autonomous network architecture. Using Chord, we gain mathematically provable scalability and great thoroughly investigated static resilience [6], which is a mandatory requirement for an identity management system. SOMA is based on a PGP-like [4] architecture, where the nodes create the public and private keys themselves. Issuing and managing of the key material is done locally and digital certificates are issued using only the collective information of the overlay routing architecture. Through certificate exchanges each node builds each keyring and stores it locally, in accordance to PGP Web-of-Trust. In contrast to hierarchical PGP, we do not use neither a central nor a distributed CA, and we avoid completely delegation of trust to a TTP. Hence, each of the agents uses its keyring independently, placing their trust depending on the identification credentials gathered. A node, after assessing the certificates on its keyring and evaluating the identity of the communicating parties, will use these credentials to establish a secure communication channel.

In the next section we will examine some of the related work and background that has been put into distributing PKIs. Section 3 describes our authentication system called SOMA. Section 4 analyses our Bayesian framework and how it interacts with the rest of the system. Section 5 presents our evaluation including discussion of the security issues and closely examines possible attack scenarios. Finally, section 6 summarises important points and outlines the conclusions drawn.

2. Related Work

The notion of a distributed PKI to provide authentication services for ad-hoc networks has been coined around for many years. A lot of interest and research has been put into creating efficient scalable infrastructures, that do not sacrifice the autonomous characteristics of the comprising nodes. Research has mainly been put into mobile ad-hoc networks (MANETs), where mobility is a limiting factor, as well as in wired systems, where the central PKI is distributed using threshold cryptographic schemes [7]. In this section, we summarize the most important literature that has been proposed for tackling public key management in ad-hoc networks. We also do a critical appraisal on issues that we will be investigating through SOMA.

In 1999 L. Zhou and Z. Haas [8] proposed the first Distributed Certification Authority (DCA) for ad-hoc networks, based on threshold cryptography. We refer to this work because it was the foundation for many DCA implementations that followed and hence, have similar limitations abated to an extent depending on the specifics of each approach. The architecture proposed by Zhou and Haas has an arbitrary number of server-nodes and combiner nodes that constitute the DCA. Without delving into technical details, these server-nodes would divide among them the private key of the CA. The server-nodes would follow a (n, t+1) scheme, needing t+1 out of n shares to recover the secret parts of the divided private key, as per Shamir's secret sharing scheme [7]. All the nodes in the system would also have a public key/secret key pair that the DCA would be responsible to certify. Subsequently, the server-nodes involved would sign with their share of the private key a partial signature and submit these pieces to a combiner node that produces the final signature.

There are several limitations with this approach, most importantly, it does not scale since it is semi-distributed. Furthermore, it uses empowered nodes, which are not self-organized, but rather depend on an off-line authority to empower them. The nodes also require an off-line assignment of Unique Universal Identifiers (UUIDs). This system does not provide a solution in case of connectivity problems between the nodes, since it was out of its scope. Large-scale changes in topology have a detrimental effect both on the system security and its performance. This is due to the fact that security comes with a trade-off to availability and hence, a method needs to be devised to re-organise the number of shares, required to construct the secret key.

Similar in concept to Zhou and Z. Haas work is proposed by Kong et al. [9], which alleviates the availability issues, by not using combiner nodes, but exposes the system to Sibyl attacks [10]. More work in similar grounds has been done in [11] by Zhou et al. for use on the Internet. Furthermore, Yi and Kravets [12][13][14] propose a DCA based on threshold cryptography. The main points of their proposal are: the nodes requiring certification contact t+1 nodes and make use an efficient β-unicast method for dissemination instead of flooding.

All the aforementioned proposals do not scale well and are not fully self-organized, hindering the autonomous characteristics of a mesh network. One different approach that uses certificate chaining similarly to the PGP Web-of-

Trust was proposed by Hubaux et al. [15][16][17] for MANETs. They developed a self-organised solution for key management in ad-hoc networks, where each user has authority and issues public-key certificates to other users. Key authentication is done via the exchange of certificate graphs. Each user stores all the certificates from other users and evaluation of valid certificate chains is done through merging their individual certificate repositories. The certificate dissemination that takes place is done in ad-hoc flooding basis, taking into account the mobility of the users, that is integral for the convergence of the certificate graph. This introduces delays in the initial stages of the network and may cause problems in cases there are insufficient trust relationships.

Other related works that focus on different requirements can be found in identity based encryption schemes in [18], [19]. In addition, overlay DCAs have been proposed that use DHT (Distributed Hash Table) to provide the primitives needed to create a distributed storage of the key material and the underlying management infrastructure. In SOMA, on the other hand, we do not use storage of a DHT as a core component. A DHT solution can be found on [20], in which the authors propose a statistical quorum-based mechanism to probabilistically get certificates of the overlay and decide on their authenticity. Moreover, on [21] they propose the use of threshold cryptography to distribute the CA functionality. They derive and delegate trust by distributing the certificate directory and the private key of the CA in many nodes, divided into segments of trust. In both these structured Peer-to-Peer solutions DHT provides a distributed storage for the key material and the underlying management infrastructure, distributing the directory and the private key of the CA. In our scheme, on the other hand, we focus on the autonomy of the nodes, each acting independently using a Web-of-Trust.

3. SOMA

In this section we present our proposal and the rationale behind our design choices. Furthermore, we proceed in analysing the steps taken for the formation of a mesh network and the authentication protocol specifics that will be followed by the nodes.

3.1. Overview

Our goal is to build an authentication infrastructure in the presence of active adversaries, so that the nodes can verify the public keys and their authenticity via chains of trust. Our system is build on top of Stoica's Chord [6]. Chord is a structured overlay protocol, that provides a look up interface based on consistent hashing over a circular (modulo) identifier space. Using an identification key, Chord maps it onto a node. Chord is normally used as the basis for providing DHT functionality. On a DHT that is based on Chord, we can normally store (key,value) pairs as in every other locally stored hash table and retrieve the value back, based on the key. Consistent hashing uses a one-way function (e.g., SHA1, SHA2) to map both the keys and the node IDs (e.g., IP-addresses) uniformly

distributed to the same identifier space. A key look-up in Chord returns the related IP address. Chord is scalable, taking O(log N) communication hops and keeps O(log N) state per node, where N is the total number of nodes in the system.

The benefits inherited from Chord can be summarised:

- A simple scalable lookup algorithm based on consistent hashing.
- An overlay that is robust on node joins and failures.
- A stabilisation protocol.

Our proposal focuses on networks that display transitive trust relationships between the comprising nodes. This can be extrapolated further saying that as a community WMN, the nodes will build a PGP Web-of-Trust and exhibit characteristics, similar to Small-World graphs [22]. Moreover, even-though the nodes themselves will not often be neighbours of one another, most nodes will be reachable from every other by a significantly smaller number of hops, compared to the network size. This is usually achieved by short-cuts between the nodes.

We use the overlay network as a meta-structure that efficiently creates the transitive relationships, so that the nodes can deduce the chains of trust between them in a self-organised and scalable manner. Hence, when the Chord protocol is used as the basis of a Web-of-Trust model and certificates are stored in local keyrings, we can safely reach the conjecture that a node would be able to find a chain of certificates in O(log N) time and in O(log N) number of certificates for any trust path given. Ideally, a peer requiring a secure authenticated service to another peer would be able to deduce a valid chain of trust, with the same complexity as would a simple look-up do in Chord. Therefore, SOMA builds a self-organised distributed authentication service, based on the aforementioned idea and tailors it to the needs of mesh networks.

The SOMA architecture comprises the mesh nodes, forming a virtual ring overlay. Each node has a unique ID that encompasses its physical and logical address. A node wishing to join SOMA requires to know one node that is already in the ring (e.g., its bootstrapping node to the mesh network). Subsequently, each node is able to exchange certificates and establish trust relationships with the rest of the nodes in SOMA. The overlay structure forms the framework for the transitive trust relationships and each node exchanges certificates with the nodes that is directly responsible for routing. When a node requests a PGP chain of certificates to another node, following the overlay routing, will result to an efficient trust path discovery.

The nodes in the mesh form logical ring. The main components of a node in SOMA are listed below:

- A logical ID that is directly correlated to its physical ID.
- A PGP keyring.
- A finger table providing efficient lookups and captures the inter-node certificate relationships.

- A list of successors and a cache of IP addresses that provide resilience against attacks and failures.

In SOMA, we assume that the nodes will use TCP and are assigned static IP addresses. The network is assumed vulnerable by adversaries at all times. The TCP communication messages are reinforced with authenticated Diffie-Hellman session keys. The certificate exchange between two parties is done using 2-pass strong authentication. The PGP messages and certificate format details follow the OpenPGP Message Format [23] or any compliant format.

3.2. Architecture Characteristics

The basic operations and goals of our solution stem from the fact that we wish to have no empowered nodes. From a security design point of view, our goals are similar to the ones found in most PKIs and public-key schemes providing an infrastructure for authentication, data integrity, confidentiality and non-repudiation. We take advantage of the consistent hashing, employed by many structured Peer-to-Peer overlays and we use the overlay look-up protocol to infer the trust relationships between the autonomous peers and therefore construct the needed chains of trust between them. SOMA will be evaluated on the following desirable system properties:

- Fault tolerance: Capability to maintain correct operation under stress, high node churn-rate, faulty nodes etc.
- Security: Assuming smart active adversaries, our architecture must provide the means for authentication, secrecy and integrity between principals. Attacks on security include failure for a user to verify a principal making an authentication request. Other attacks include, violation of integrity, such as, tampering with data from an unauthorised party, and finally, violations of secrecy, where we have unauthorised reading of captured data by third parties.
- Availability: Even though fault tolerance and availability are related, we need our infrastructure to be resilient to denial of service attacks. An attacker should not be able (or at least without incurring high cost in resources) to make the service unavailable to legitimate users.

3.3. The Chord Lookup

Chord in its essence has a single operation, given a key, find a node ID in the network that is responsible for storing that key. Node IDs and keys are both hashed and arranged in the same modulo space, having no more than *2m* nodes (where m is the number of bits that identifies a given node). If node identifiers are represented as a circle of numbers from 0 to 2^{m-1}, the *successor(key)* of a node, is the node that is responsible for that key moving clockwise in the ring.

Each node holds a small routing table with at most m entries, this is a list of pointers to node IDs in the overlay and is called the finger table. The finger

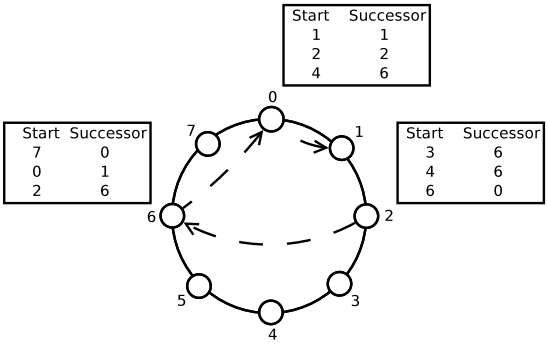

Figure 1: Chord lookup, N=8 with online nodes being 0,1,2 and 6.

table *n.finger(i)* is the i^{th} entry of the finger table for the node n and will hold the $n + 2^{i-1}$ node ID that succeeds n on the circular identifier space, where $1 \leqslant i \leqslant m$. When a node is looking for the successor of a key it will sent a find successor request to the closest finger preceding k. For instance, at Fig. 1 we have a stable network and the corresponding finger tables for each of the nodes. With $m = 3$ when node 2 initiates a find successor for key 1 it would start with the closest finger preceding key 1 which is node 6, node 6 in return would proceed with another find successor call starting at node 0 finding finally that the successor of key 1 is node 1. In practice Chord deals with concurrent node leaves and joins, therefore the finger tables would need to be updated. This is where a stabilisation protocol comes in; every node runs stabilize() periodically and this is how the finger tables get updated and new nodes join concurrently.

Now that we have briefly introduced Chord, we will proceed with SOMA detailing any additions to the original protocol and using the original notation. For more detailed description on the original protocol, node joins, stabilisation(), findsuccessor(), fixfingers() and notify(), please see Chord's protocol from [6].

3.4. Initialization and Bootstrapping

When a new node n wishes to join the network and use the authentication service, it must first create a public/secret key pair Pk_n/Sk_n . Furthermore, n will produce a self-signed certificate $Cert_n$, which is done locally and without any CA involvement. Then, it bootstraps to the network, by contacting an one hop distance network node n'. This serves as a trust-anchor and short-cut in the overlay ring. Both n and n' are considered initially trusted, since usually in CMNs a node to join the network has to contact another peer directly for bootstrapping, address resolution, antenna alignment etc. Therefore, we can safely assume that the node n can verify the identity of n', its introducer, as authentic at least to the extent of connecting the peer n to the mesh.

To extrapolate further on the above, the side channel (e.g., physical contact or other direct channels) between two peers performs the role of an off-line CA. n' being the bootstrapping node of n will sign the authenticity of the key

Pk_n and n will sign the authenticity of $Pk_{n'}$. The certificate will follow the PGP format details and n' will then put in its keyring the certificate $Cert_{n \Rightarrow n'}$. Additionally, n will hold in its keyring $Cert_{n' \Rightarrow n}$ as in the PGP Web-of-Trust. Hence, at the beginning we have a direct one-hop relationship which are bi-directionally verified by n and its introducing nodes.

To join a SOMA ring, a node needs to generate a new unique ID. The unique ID of node n is ID_n and will be derived by hashing the concatenation of the node's n public key and its network address $ID_n = h(Pk_n + IP_{address\ of} n)$. This is acceptable in CMNs since they are at least semi-planned: in other cases different unique identifiers can be used for hashing without any issues except different implementation specifics. This identifier ID_n will be the key that uniquely identifies node n in SOMA's circular identifier space. Node n will then use the produced ID_n to connect to the overlay. The authenticated introducer in SOMA is in fact what the initial contact link is to the Chord ring. Node n' will learn node's n successor by looking up ID_n and n will proceed with building its finger table, as described in the following section.

3.5. Node Join and Stabilisation

For n to join SOMA it will need to set-up its finger table. A finger table, as we mentioned before is a list of pointers to node IDs in the overlay. Instead of holding a single pointer to the next node, $n.finger = ID_{next\_node}$, a list of m nodes is maintained, *n.finger(m)*, with their logical inter-node distance increasing exponentially. This provides the efficient look up mechanism. Typically, on each of the entries, associations and additional information will be stored. For instance, each ID in the finger table will be associated with a corresponding certificate from the local keyring and a physical address. Moreover, the nodes will need to take past transactions into account, therefore, for each of the fingers n will record a trustworthiness metric and a cache of IP addresses encountered thus far. The finger table of a node can be viewed as a map holding the IDs that a node can use as certificate paths in SOMA. Therefore, the finger table will comprise all nodes that the node has exchanged certificates with and is responsible for their correct routing. The *n.finger(i)* is the i^{th} entry of the finger table for the node n and will hold the $n + 2^{i-1}$ ID and physical address that succeeds n, where $1 \leqslant i \leqslant 160$ (e.g., for SHA1 with arithmetic modulo 2^{160}).

Hence, when n joins the ring, each SOMA node has incomplete network picture holding O(log N) state in each finger table, for the network to holistically capture the new entry more nodes in SOMA need to propagate the change. Similarly to the original Chord look up, for a certificate path discovery to be successful, only the successor pointers of the finger table need to be correct. The finger table mechanism serves only in lookup efficiency, not correctness. Moreover, the finger table needs to be kept up to date with new entries so that certificate look ups scale logarithmically.

When node n joins, the steps taken are:

1. Initialization of the predecessor and the finger table entries of n by establishing authenticated relations through a chain of trust.
2. Update the fingers and predecessors of the existing nodes to reflect the addition of n.

The first step for the joining node n will be to initialize its finger table and find its virtual position in the ring. Node n calls $n.join(ID_{n'})$, where node n' is its introducer. This will make n' to do a Chord look-up of ID_n in the ring and return back to n the key/address of its successor s. The successor s is the first finger in the finger-table of n, $n.finger[1] = s$. In addition, n' will provide n with a PGP certificate $Cert_{n' \Rightarrow n}$ so that it can associate its credentials with s. Finally, node n will set $n.successor \rightarrow s$

The join mechanism of SOMA changes the periodic stabilisation protocol in Chord. The procedures $stabilize()$, $notify()$ and $fix\_fingers()$ run periodically to notify the overlay nodes for the new entry, accomplishing the steps 1 and 2. After finding its successor, n will proceed with $n.stabilize()$ so that it can exchange its credentials with its successor and propagate the information on the new join. We may think of $stabilize()$ as a protocol working on linked list, that uses certificates for authentication and updates the linked list when there is a new entry.

The stabilization runs periodically in the background by all nodes. The first step for $n.stabilize()$ is for n to notify node s that $n.successor \rightarrow s$ using the certificate provided by n'. Node s will proceed with checking the validity of the supplied credentials and call $s.findCertChain(n')$ (more details in section 3.6). The validation mechanism includes checking the public key, the address, the signatures, the timestamps and if any of the certificates in the chain is revoked. If s finds that all are correct, it sets $s.predecessor \rightarrow n$ else will request for another valid PGP certificate. Moreover, when the original predecessor of node s, node y runs $stabilize()$ asking for $s.predecessor$, s will return to y that n is its new successor. Node y in turn will $notify()$ n, which will reply back providing the certificate chain by n'. Node y will similarly check the supplied chain leading to n. After the above stabilizations finish, n will set $n.predecessor \rightarrow y$ and y will have $y.successor \rightarrow n$.

After the establishment of correct associations with the successor and predecessor, $fixFingers()$ which also runs periodically by the nodes, refreshes the finger table entries. The procedure is repeated for each of the non-trivial ring intervals in the finger table, which are the ones containing non-virtual distinct nodes. The procedure of $n.fixFingers()$ is as follows: $For\ each\ i \in m$: $finger[i] = findCertChain(finger[i])$, where m=number of bits used as index (e.g. SHA1 m=160 bits). The procedure to find a certificate chain to another node is done through the $findCertChain(target\_node)$ and follows original Chord $find\_successor()$ algorithm, as described in the following section.

```
Join(targetid)
 predecessor = null
 n gets successor n.finger[1]=s via n'
 n provides n with CERTn'->n

Stabilize() //on n.stabilise()
 n notifies s that is its new predecessor with CERTn'->n
 s.predecessor = n after evaluation of CERTn'->n

Stabilize() //on y.stabilise() the original predecessor
 y asks s if s.predecessor changed
 y.finger[1]=n after evaluation of CERTn'->n
```

Figure 2: Psedocode for n.Join(n') and subsequent n,y Stabilize() calls

3.6. Certification

When n wishes to authenticate with a peer, it will require a valid chain of trust. In an established SOMA with N number of peers, n will hold all the routing information that is needed on its finger table, similarly to Chord. In addition, n will hold a keyring with all the public keys and information that will allow it to make decisions, based on a trust metric. When two nodes need to mutually authenticate, they construct a certification request and exchange their credentials using the following:

1. $A \longmapsto B : Cert_A, r_1, t_1, ID_B, data_1, (r_1, t_1, ID_B, data_1)Sk_A$
2. $B \longmapsto A : Cert_B, r_2, t_2, ID_A, data_2, (r_2, t_2, ID_A, data_2)Sk_B$

$Cert_{A,B}$ are the respective node certificates. $r_{1,2}$ are nonces and $t_{1,2}$ are expiration timestamps used to ensure freshness and that they were received in a valid time-frame. The $data_{1,2}$ may include a session key encrypted with the receiver's public key and also other information related with the origin of data and implementation specifics.

In case that n wishes to verify the credentials of a node p, it will have to get a trust chain leading to p. Therefore, n will first check if p is already in its finger table or if p is an introducer for n. If any of the two holds true, then n will have a chain of certificates from the stabilisation protocol leading to p, which can be directly verified for its validity. If ID_p is not included in its finger table, then node n makes a certificate chain lookup to find the PGP chain to p.

Following the above, n with ID_n has introducer ID_n' and wants to authenticate the certificate of p (Fig. 3). Node n finds a certificate chain to ID_p, by sending a *findCertChain()* request to an intermediate node j, *j.findCertChain(p)*. Node j is chosen because ID_j is the closest finger that precedes ID_p in n's finger table and hence in the circular identifier space. Node n has already exchanged credentials with node j at join/stabilisation. Upon receiving the request by n, j will look in its finger table finding that the closest preceding finger is k and will

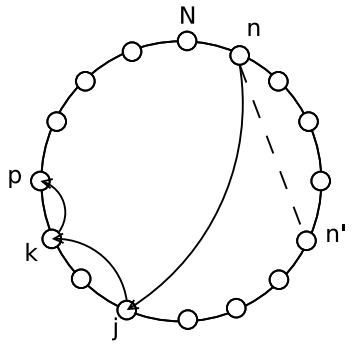

Figure 3: An example of the logical representation of SOMA ring with N nodes.

reply with this information back to n providing also the certificate containing $Cert_{j \Rightarrow k}$. Finally, node n will ask $k.findCertChain(ID_p)$ similarly returning to n the final link $Cert_{k \Rightarrow p}$. As a result, n receives on each query a certificate that will allow it to build a chain of trust up to ID_p, each time reducing the distance half-way through. Node n logarithmically gets closer to the target ID_p, building the needed trust path.

```
findCertChain(targetid)
 look in local finger table for
  the highest node n satisfying myid < n < targetid
  & having valid certificate details
 if n exists
  call findCertChain(targetid) on node n
 else
  return myid.successor
```

Figure 4: Psedocode for *findCertChain()* from a node to another

We can use the above certificate path-building method in addition to a reputation framework which will be introduced in Section 4 to include ratings from all the experiences between principals. On every transaction between a node and a finger, an outcome will be recorded and its reputation score calculated. We do not wish to claim specific parameter values as accurate ratings other than the positive and negative outcomes between events. For instance, in our previous example in Fig.3 supposing node j was malicious and was misbehaving in routing, node n could undershoot in its finger table and therefore avoiding the problematic node as if it was faulty. After a while, intentional routing misbehaviour by specific nodes would be represented in the rest ring effectively skipping it in their finger tables. This approach results in a reputation based path ranking that is similar to the discrete ranking of PGP Web-of-Trust[23] but also allowing for further flexibility and extendibility allowing more complex representation of social interactions and structures.

3.7. Revocation

In every PKI, dealing with the revocation of certificates can be a technical challenge. In SOMA a node can implicitly or explicitly revoke its published certificate. An implicit revocation by a node takes place when a certificate is allowed to natural ly expire. Each issued certificate has a predetermined lifetime. When its expiration time passes, it will no longer pass any validation tests taken by the nodes and hence, the certificate will be considered implicitly revoked. SOMA nodes will update their certificates while still having the old self-signatures, creating an updated additional self-signature with the same ID/key updating the time. The latest self signed signature takes precedence, so all peers will be updated. On the other hand, if a node is aware that its private key has been compromised, it can explicitly revoke the certificate for the public key in question. This is done by using a revocation certificate. Each node, creates and stores safely (e.g., in external media) a revocation certificate as described in the OpenPGP RFC[23] that will be used as insurance in case of a key compromise. As with most Web-Of-Trust solutions if a principal looses its private key it can not retrieve that identity and can only revoke it. In SOMA a node has to bootstrap again with a new key, the revocation certificate will be used to effectively separate the ID/node relationship for that node in the ring.

The node that revoked its certificate does not need to send the revocation request certificate to all the nodes in SOMA. The only interested parties in the revocation scheme are the ones that point in their finger table to the revoked key. Therefore, the node only needs to provide the revocation certificate to its predecessor and exchange it through the stabilisation protocol with all the nodes that update their finger table to the node itself. In this way, the Certificate Revocation List (CLR) is build conjointly with the keyring for each node and will propagate to all the nodes that require it. Furthermore, when a node requires to check if a certificate is currently revoked, it only needs to proceed with the normal lookup operation. The nodes along the SOMA ring will provide the revocation certificate if has been published.

4. Reasoning about trust

The intrinsic difficulty in the design of large scale Peer-to-Peer systems stems from the fact that the nodes are opportunistically connected and at any given time they can exhibit erratic behaviour. In our distributed infrastructure for instance, a disruptive/malicious node could misbehave each time a transaction takes place or create multiple Sibyl identities to attempt to eclipse part of the network. Therefore, the interacting nodes always face an inherent risk (as with every PKI) and since no de-/centralised regulatory authority exists, the nodes should be provided with the means to make informed decisions without sacrificing their autonomy.

There are two principal archetypes in trust modelling: certification and reputation based. In our system there is complete absence of a regulatory structure and hence, the certification based approach even though it provides hard

security assurances, it would require uncoerced cooperation. Reputation modelling on the other hand does not provide the assurances that certification based approaches do and therefore, the specifics depend mainly on the underlying infrastructure, usually following a global or local based trust model. Global trust between nodes involves a universally accessible trust metric that peers iteratively compute between them (e.g., EigenTrust [24]) through feedback in form of direct experiences and recommendations. Local trust on the other hand focuses on direct experiences and nodes receive limited neighbour feedback in the decision making process engendering trust in unregulated and dynamic scenarios (e.g., PeerTrust [25]). The most evident difference between local trust and global is that in local the nodes themselves usually select the members they will be taking feedback from.

SOMA uses a local reputation model where each peer calculates reputation scores using a Bayesian framework. The peers are able to place a rating on the behaviour of other principals and their confidence about the outcome of future transactions. For any given transaction we consider binary outcomes defined in terms of positive and negative experience.

$$o \in \{Pos, Neg\}$$

After i transactions we will have a number of outcomes:

$$o = \{o_0, o_1, ..., o_i\}$$

Ratings are generated and updated with our random variable following the Standard Beta probability density function(pdf).

$$f(x|p,q) = x^{p-1}(1-x)^{q-1}\frac{1}{B(p,q)} \; for \; 0 \leq x \leq 1; p,q > 0 \tag{1}$$

Where p and q are the function shape parameters and $B(p,q)$ is the beta function.

$$B(p,q) = \int_0^1 t^{p-1}(1-t)^{q-1}\,dt \; where \; Re(p), Re(q) > 0$$

Using Bayesian inference for x of eq.(1) for o number of observations:

$$E[x|o] = \frac{p(o)+1}{p(o)+q(o)+2} \tag{2}$$

With Bayes' theorem we update the reputation score for each peer through observations of successes and failures. This is done by combining the prior probability distribution for x with the *a posteriori* distribution. In other words, the posterior probability comes from an initial assumption and gets combined with the observations for the amount of positive $p(o)$ and negative $q(o)$ outcomes that a node has from a predefined number of previous interactions with the principal in question.

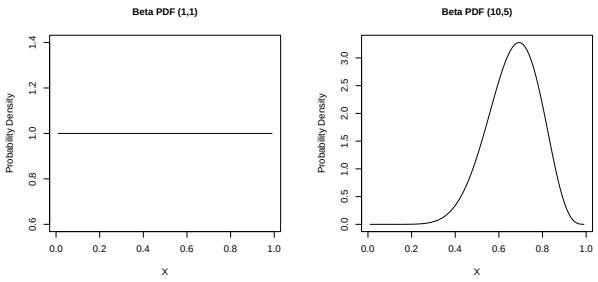

Figure 5: Beta PDF for (a) $f(x|1,1)$ and (b) $f(x|10,5)$ respectively.

Initially a newly joined node n will have a blank state about the reputation of the nodes in its finger table. Therefore, the reputation for each of the finger entries will be uniform across the probability distribution domain and for each finger we assume *a priori* the uniform distribution $f(x|1,1)$ shown in Fig.4(a). After node n continues transacting with other peers it will record the outcomes of each transaction and calculate the score for each principal. For instance, node n after a successful exchange of credentials with $n.finger[5]$ will have accumulated 9 positive experiences and 4 negative ones, then the Beta pdf for $n.finger[5]$ will be $f(x|10,5)$ in Fig.4(b) and its $E[x]$ computed through eq.(2).

We can use the previous framework to include all the experiences between a principal and its respective fingers. For every transaction between a node and a finger its outcome will be recorded and its reputation score calculated. Every node will record for each finger a vector $o = \{o_0, o_1, ..., o_i\}$, o_i being the last recorded outcome of their transactions. To minimise the impact of people building reputation and then misbehaving, we will include a decaying factor λ for the aggregated outcomes. Looking back at eq.(2) *p(o)* and *q(o)* will become $p(o)_\lambda$ and $q(o)_\lambda$ and take the form of $\sum_{n=0}^{i} \lambda^n o_i$ *where* $0 < \lambda < 1$.

5. Evaluation

5.1. Security Analysis

The design characteristics of SOMA focus on decentralisation through self-organisation, scalability and robustness against malicious behaviour. In this section we analyse the possible attack scenarios derived from the characteristics of our architecture and the desirable system properties identified in 3.2. Attacks range from the certificate exchange mechanism, the control and use of key material and, finally, the overlay routing itself.

Node Join: When a node joins the ring, a malicious bootstrapping node could attempt to provide false credentials to the rest of the SOMA nodes. This attack would be negated in our case, since the node's public key and physical IP address is connected with its logical ID on the ring. Any false credentials provided by the bootstrapping node would require a valid certification chain, which it can not forge due to the certificate digital signatures.

Certificate Chain: A node wanting to find a certificate chain to another node, needs to authenticate first a chain of intermediate nodes. These intermediate nodes have valid IDs and the digital certificates provide authentication and non-repudiation for each node in the certificate path. The consistent hashing is a pre-image resistant mechanism between the physical and logical address which, provides a simple defence against impersonation and Sibyl attacks. If one of the nodes misbehaves or simply creates multiple identities, it will be trivially detected, since the certificates are bound to its address. Therefore, this node can simply be ignored and move on the previous node preceding the target node in the finger table. As long as a single node in the finger table follows the protocol, the authentication can proceed. Moreover, with our reputation extension we can have a threshold for misbehaviour tolerance. After this threshold a malicious node can be blacklisted or be dealt accordingly to predefined rules.

Denial of Service: If a node joins a SOMA ring, where the majority of the nodes are malicious, then its identity even though could not be forged, the authentication service for that node could be potentially disrupted through denial of service (DoS). Against DoS attacks, SOMA is resilient by using ratings and hashing. With ratings as a defence mechanism peers can blacklist and exclude malicious nodes that sent fraudulent messages after a threshold since their identities are detectable. In combination to the reputation consistent hashing is used to distribute the logical identities of the nodes. Therefore, malicious peers would require a large majority to eclipse a node(cut off his ingoing and outgoing links). This is due to the fact that the IDs are mapped using consistent hashing, where the standard hardness assumptions for the chosen hash function apply.

Credential Exchange: An attack directly on the certificate exchange is thwarted by the use of nonces and timestamps, which ensure freshness, and prevent man in the middle attacks (MITM). Additionally, the inclusion of origin and target data safeguards against certificate hijacking and all forms of impersonation.

Overlay attack: Attacks on the routing protocol, itself, can be hard to avoid if the majority of nodes are malicious. Even though impersonation is averted through the logical-physical address relationship of the public key certificates, a DoS attack could potentially disrupt the overlay as a whole. In such a case, our reputation model provides the necessary insight to marginalise or expel malicious nodes. Attacks on the network infrastructure itself could include churn attacks and potential network failures from the malicious nodes. Against such attacks, SOMA is resilient by requiring at least one correct node in the finger table for correct routing. In addition, instead of using a single successor, employing successor lists will provide additional routes and mitigate the effects of overlay attacks.

5.2. Critical Appraisal

To guarantee sound decision making, we expanded our policy based model with a Bayesian based framework. This model provides additional insight and better informed reasoning that is based on direct experiences. Our system is designed so that it can incorporate a recommendation extension that can

support any form of reputation context. Such an extension can be plugged into our Bayesian model describing observations, recommendations, confidence, trustworthiness of the recomenders etc. The integration of such a model on top of SOMA to achieve a different type of service will be investigated in future work.

SOMA is a hybrid solution based on structured Peer-to-Peer to tackle the limitations that arise in threshold cryptography, random walks and pure Web-of-Trust solutions. It utilises the collective knowledge gained by structured Peer-to-Peer overlays and combines it with a certificate chaining solution to create an autonomous self-organised authentication system suitable for our goals. SOMA and its extended version follows up on the the mathematical properties of the Chord look up, therefore path discovery for the chain of trust building becomes scalable with bounded path traversal latency of O(log N).

In SOMA, there are no centralised authorities and no empowered nodes because this would limit the autonomy of the nodes and disrupt the nature of a mesh network. We do not adopt any form of threshold cryptography, since it would introduce an unnecessary trade-off between security and scalability. To extrapolate further on the (n, t+1) schemes, an open mesh network has many thousands of nodes and hence, we can not assume that an adversary would not compromise t+1 nodes in any given time frame. Moreover, we would need appropriate adaptive control mechanisms to calculate the thresholds, depending on the dynamics of the network topology, churn and number of nodes in the system. Therefore, we avoid the risk that threshold based schemes introduce.

6. Conclusions

This paper has proposed a large-scale authentication system for mesh networks without the need of a TTP. Our proposal, allows each node to decide its own trust policy while keeping its autonomy intact. We have shown how the overlay can be used as a meta-structure to infer trust relationships and not as the means to provide distributed directory storage. Wireless ad-hoc networks have inherent vulnerabilities, the nodes can always be considered susceptible to security attacks and to physical capture. Thus, in creating SOMA we took into account the security of the network as a whole and provided the building blocks for establishing trust without relying on centralised authorities.

Acknowledgement: The work described in this paper was partially supported by PeerAssist (AAL-2009-2-137), a reasearch project funded in the context of the European Commission's Ambient Assisted Living (AAL) Joint Programme.

References

[1] AWMN: Athens Wireless Metropolitan Network, `www.awmn.gr`

[2] Bicket, J., Aguayo, D., Biswas, S., Morris, R.: Architecture and Evaluation of an Unplanned 802.11b Mesh Network. In: 11th Annual International Cconference on Mobile Computing and Networking (2005)

[3] Demertzis, F. F., Xenakis, C. : SOMA: Self-Organized Mesh Authentication. In: Proc. 7th European Workshop on Public Key Services, Applications and Infrastructures (EuroPKI′10), Athens, Greece, Sept. (2010)

[4] Zimmermann, P.: The Official PGP Users Guide. The MIT Press, Cambridge, MA (1995)

[5] Gnutella RFC v0.4, `http://rfc-gnutella.sourceforge.net/index.html`

[6] Stoica, I., Morris, R., Karger, D., Kaashoek, M. F., Balakrishnan, H.: Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications. In: ACM SIGCOMM Technical Conference (2001)

[7] Shamir, A.: How to Share a Secret, ACM, vol. 22, Issue 11,pp. 612 - 613, (1979)

[8] Zhou, L., Haas, Z.: Securing Ad-hoc Networks. IEEE Network, vol. 13, no 6, pp. 2430, November/December (1999)

[9] Kong, J., Zerfos, P., Luo, H., Lu, S., Zhang, L.: Providing Robust and Ubiquitous Security Support for Mobile Ad-hoc Networks. In: 9th International Conference on Network Protocols(ICNP), November (2001)

[10] Douceur, J. R.: The Sybil Attack. Springer Berlin / Heidelberg, Microsoft Research, One Microsoft Way, Redmond, WA, 98052-6399, USA (2002)

[11] Zhou, L., Schneider, F., Renesse, R.: COCA: A Secure Distributed Online Certification Aauthority. ACM Transactions on Computer Systems (TOCS), vol. 20, Issue 4, pp. 329 - 368, November (2002)

[12] Yi, S., Kravets, R.: Practical PKI for Ad-hoc Wireless Networks. Department of Computer Science, University of Illinois, USA (2001)

[13] Yi, S., Kravets, R.: Key Management for Heterogeneous Ad-hoc Wireless Networks. In: 10th IEEE International Conference on Network Protocols (ICNP), (2002)

[14] Yi, S., Kravets, R.: MOCA: Mobile Certicate Authority for Wireless Ad-hoc Networks. In: 2nd Annual PKI Research Workshop, (2003)

[15] Capkun, S., Hubaux, J., Buttyn, L.: Mobility Helps Security in Ad-hoc Networks. In: Mobile Ad Hoc Networking and Computing(MobiHoc), (2003)

[16] Capkun, S., Buttyn, L., Hubaux, J.: Self-organized Public-key Management for Mobile Ad-hoc Networks. IEEE Transactions on Mobile Computing, Vol. 2, Issue 1, pp. 52 - 64, January- March (2003)

[17] Capkun, S., Hubaux, J., Buttyn, L.: Mobility Helps Peer-to-Peer Security. IEEE Transactions on Mobile Computing , vol. 5, Issue 1, pp. 43-51, (2006)

[18] Boneh, D., Franklin, M.: Identity-based Encryption from the Weil Pairing Advances in Cryptology. In: CRYPTO 01, (2001)

[19] Bobba, R. B., Eschenauer, L., Gligor, V., Arbaugh, W.: Global Telecommunications Conference. GLOBECOM '03. IEEE Publication Date 1-5, December (2003)

[20] Aberer, K., Datta, A., Hauswirth, M.: A Decentralised Public Key Infrastructure for Customer-to-Customer E-commerce. International Journal of Business Process Integration and Management, vol. 1, No. 1, pp. 26-33, (2005)

[21] Avramidis, A., Kotzanikolaou, P., Douligeris, C.: Chord-PKI: Embedding a Public Key Infrastructure into the Chord Overlay Network. Springer Berlin / Heidelberg, vol. 4582/2007, ISSN: 0302-9743, pp. 354-361, (2007)

[22] Watts, D.J., Strogatz, S. H.: Collective Dynamics of 'small-world' Networks. Nature, vol. 393, doi: 10.1038/30918, 440442, June (1998)

[23] Open PGP November 2007/RFC 4880, `http://tools.ietf.org/html/rfc4880`

[24] Kamvar, S., Schlosser, M., Garcia-Molina, H. : The EigenTrust Algorithm for Reputation Management in P2P Networks. Proceedings of the Twelfth International World Wide Web Conference, (2003)

[25] Xiong, L., Liu, L. : PeerTrust: Supporting Reputation-Based Trust in Peer-to-Peer Communities. IEEE Transactions on Knowledge and Data Engineering (TKDE), Special Issue on Peer-to-Peer Based Data Management, 16(7), July, (2004)