SecP2PSIP: A Distributed Overlay Architecture for Secure P2PSIP

Rasib Khan and Ragib Hasan
SECRETLab, Department of Computer and Information Sciences
University of Alabama at Birmingham, AL, USA
Email: {rasib, ragib}@cis.uab.edu

ABSTRACT

Although Peer-to-Peer (P2P) protocols are known to be highly insecure, they have proven significant improvements over traditional client-server models for ad-hoc deployments. P2PSIP is an architecture for deploying Session Initiation Protocol (SIP) services over a P2P network overlay, and thus leverages the limitations of client-server architectures for SIP. However, vanilla P2PSIP is highly insecure and is a well-known issue. Proposed secure versions of P2PSIP implements the security features mostly as a centralized operation or security through obfuscation, and imposes a complex deployment scenario, a single point of failure, limited usability, and a severe bottleneck. In this paper, we present SecP2PSIP – a novel scheme for a distributed overlay architecture for secure P2PSIP. Our model allows an ad-hoc deployment with a scalable design for a completely distributed security infrastructure, which allows attack containment for at least k out of n nodes. The proposed overlay architecture is fully adaptable with standard off-the-shelf SIP clients. We present a detailed design, feature, architecture, and security analysis of SecP2PSIP to illustrate its applicability. We have also implemented a proof-of-concept prototype for the proposed SecP2PSIP framework. The paper presents simulation and experimental evaluation for small to medium sized networks to validate the feasibility of the proposed scheme.

I INTRODUCTION

Peer-to-Peer (P2P) networks introduce many advantages over legacy client-server models. A lot of research proposals on different P2P algorithms and architectures were made to overcome the constraints imposed by P2P overlays in terms of content search [1]. Additionally, the Session Initiation Protocol (SIP) is a signaling protocol for establishing media sessions. The traditional client-server model for SIP has been widely used for VoIP services and a variety of other IP based multimedia applications using multiple types of SIP messages [2, 3].

However, a server-oriented deployment is a significant bottleneck in environments where formal servers and persistent network support cannot be established. This requirement has given birth to the concept of SIP services over P2P network overlays (P2PSIP). P2PSIP replaces the client-server model by removing the centralized SIP server. P2PSIP enables fast set-up, ad-hoc formation, easy deployment, and robustness against failures for SIP-based applications [4, 5].

Unfortunately, security in P2PSIP suffers from many shortcomings in its current implementations, including identity enforcement, management, and verification. So far, multiple solutions have been put forward to mitigate the security issues in P2PSIP [6, 7]. Seedorf *et al.* in [8] and [9] demonstrates the possible attacks on P2P overlays for SIP. The decentralized architecture in P2PSIP reduces the management and hence introduces multiple security threats. The existence of malicious peers in the P2P overlay creates a significant issue in ensuring a secure environment. Thus, a malicious peer will be able to perform man-in-the-middle attacks by dropping, tampering, and re-routing SIP messages between two other peers. Certain architectures, such as RELOAD, apply the public key certificates with a centralized public key infrastructure to ensure mutually authenticated peers for SIP sessions and guarantee confidentiality and authenticity of SIP packets [10, 11].

Research proposals on secure system enforced public key infrastructures and security through obfuscation. However, solutions for secure P2PSIP systems lacked any distributed architecture for the security mechanisms. As a result, they introduced a bottleneck in the architecture and mandated external network connectivity, which removed the feature of ad-hoc deployments for P2PSIP systems. The essence of a pure P2P architecture is therefore gone, with the management and enforcement made at a single point. Additionally, such architectures significantly reduced the usability for users.

In this paper, we present the design for a distributed overlay architecture for Secure P2PSIP (SecP2PSIP) session establishment. The proposed architecture for SecP2PSIP operates on a separate layer, and uses standard SIP interface for off-the-shelf SIP clients to communicate with the adapter overlay. We initially discuss the background and evolution of the related technologies for P2PSIP. We highlight the current security challenges in P2PSIP, and present the model and schematic design for a distributed overlay network architecture for SecP2PSIP. We present

a detailed design, feature, architecture, and security analysis of our proposed model to illustrate its applicability. Finally, we illustrate the feasibility of the design using a proof-of-concept prototype implementation. The paper includes experimental results on simulations created using our prototype implementation for small to medium sized SecP2PSIP networks.

Contributions: Our contributions in this paper are:

1. We introduce a novel distributed architecture as an overlay network for secure SecP2PSIP sessions.
2. We present a detailed analysis of our design to illustrate the feasibility, usability, and security of the scheme.
3. We have implemented a proof-of-concept prototype of the proposed scheme, and present experimental results to validate the feasibility of SecP2PSIP for small to medium sized networks.

The paper is organized as follows. We discuss the background technologies in section II. We illustrate the model for designing a security architecture in section III. The proposed scheme for SecP2PSIP is explained in section IV. Section V provides an extensive analysis of the design space for the proposed protocol. We present details for the proof-of-concept prototype implementation and experimental results in section VI. Section VII presents the related works in this area of research. Finally, we provide the conclusion and future works in section VIII.

II CHORD AND P2PSIP

Among the many Distributed Hash Table (DHT) algorithms used for maintaining information on a P2P overlay, the best known are Kademlia [12], Pastry [13], CAN [14] and Chord [15, 16]. The Chord protocol is the most widely used resource distribution architecture for P2P overlay networks. The performance comparison of the different protocols have been extensively studied by Lua *et al.* and Martinez-Yelmo *et al.* in [17] and [18] respectively.

In Chord, peers and resources are organized and indexed sequentially on the P2P overlay network. Resources are stored on the overlay network by mapping the content into a linear space, where the peers storing the resources are assigned a position in the overlay by means of a hash function. The three main operational steps in a Chord P2P overlay are (a) mapping and storage of content on nodes in the overlay by means of a hash function, (b) routing to a [*Key:Value*] pair, by starting the lookup at an arbitrary node of the overlay, and forwarding it until it finds the right peer, and (c) making a direct connection and retrieving the content [15, 16]. Establishing a session in P2PSIP is based on resource retrieval from the DHT-based P2P overlay network, and is described as follows:

- Alice is calling Bob, and calculates the hash of the Bob's username `hash(bob@example.com)` to produce a resource identifier.
- Alice locates the peer that is responsible for the calculated resource identifier as the key in the Chord overlay, i.e., the peer which is storing Bob's location. This lookup process will imply a maximum of *O(log N)* messages in the Chord overlay [15, 16].
- Once the node is found, the resource is sent back to Alice, which in this case is Bob's location information.
- After Alice receives Bob's location information, a direct connection is established between the peers. Once the connection is set up, Alice sends a SIP `INVITE` request to Bob, with the subsequent `200 OK` message from Bob and `ACK` from Alice.

III MODELING A SECURITY ARCHITECTURE

In this section, we present the models required for defining a distributed security architecture. We describe the security challenges and the corresponding system model considered while designing the proposed scheme for a distributed security architecture for SIP sessions over a P2P overlay (SecP2PSIP).

1 SECURITY CHALLENGES

P2P networks are considered inherently insecure [19, 20]. In a P2PSIP system, given that clients provide and request information at the same time, security issues are much more difficult to resolve. A malicious node may refuse to provide the data and create a Denial-of-Service (DoS) attack, or perform any unwanted operation as a man-in-the-middle (MITM). The MITM attack can also also be extended to create network partitioning by limiting resources within the compromised node. Additionally, P2PSIP systems are more vulnerable to multiple identity replications and Sybil attacks [21]. Other attacks such as worms and trojans may exist, and replicates very fast within the P2PSIP overlay.

For our architecture, we define security as the verifiability, authenticity, and integrity of the source and destination of a SIP session. The asset in this model is described as the desired destination, and the expressed identity of the source while initiating a SIP session. Therefore, no caller should be able to misrepresent his identity as another user. Additionally, there should not be any scenario where a calling party calls another user, and the user with whom the SIP session is established is not the actual desired destination. An attacker present within the overlay can try to

take control of an identity which is already present within the P2PSIP network. The attacker may also refuse to serve as a resource sharing node, thus creating a DoS attack on the P2PSIP services. Additionally, the identity management and enforcement system may be compromised, putting all the users within the overlay at risk of identity spoofing. There can also be DoS attacks on the identity server, thus overloading it with requests to obtain the public keys of users for validating source and destinations of SIP sessions. Considering the identity management system is maintained within the overlay network, the malicious node may refuse to serve lookup operations by other users within the overlay network.

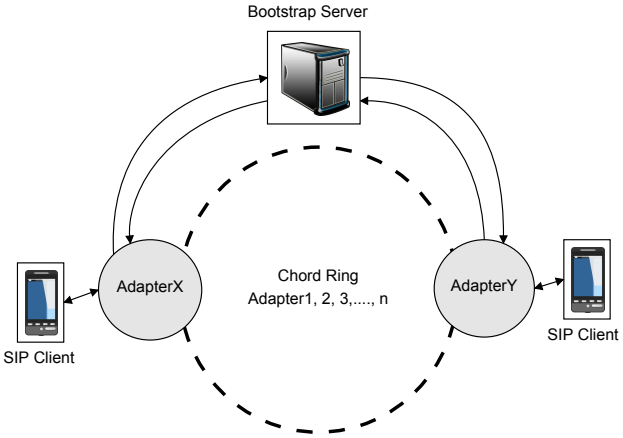

Figure 1: SecP2PSIP Architecture Overview

2 SYSTEM MODEL

We have defined the architecture for SecP2PSIP using standard elements in SIP functionality. The entities in the proposed architecture are described as follows:

- **Chord Overlay Network**: This is a ring of nodes, which forms the SecP2PSIP Chord overlay network. All nodes participating in the Chord ring are distributed in nature, with their individual resources and identities.
- **Bootstrap Server**: The bootstrap server is the only central entity in this architecture. The Chord overlay is initially created by the bootstrap server. Any node which wants to join the SecP2PSIP Chord ring is required to contact the bootstrap server to obtain the details for joining the overlay.
- **SecP2PSIP Adapter**: The SecP2PSIP adapter is a local service running on the user device. The service acts as a bridge between the user and the overlay network to provide secure services for SIP call establishment.
- **SIP Client**: This is an off-the-shelf SIP client which implements the RFC-3261 standards [2]. A user runs the SIP client on his device, and registers the client with the local SecP2PSIP adapter.

We suggest two possible options for enforcement of user identities. In a fully ad-hoc scenario, without the requirement to validate identity ownerships, the system can use self signed certificates. This is an optional feature, which can be the relaxed mode of operation for SecP2PSIP. However, in a stricter environment, the system can enforce users to have pre-owned certificates, which have already been signed by a trusted certification authority. Thus, the user will use the already signed certificate while registering with the SecP2PSIP overlay. The SecP2PSIP adapter needs to know the bootstrap server information beforehand. The joining node can obtain the bootstrapping information in certain possible ways, such as fixed/persistent nodes, network caches, or other broadcast mechanisms over the base network. However, a malicious bootstrapping node can conveniently spoof the joining peer into a false overlay network. This is a significant challenge in such ad-hoc P2P networks, and is assumed to be a trusted node. However, we suggest the use of multiple bootstrapping nodes to avoid DoS attacks and bottlenecks in the process of joining the SecP2PSIP overlay.

IV THE SECP2PSIP SCHEME

Till date, there have been multiple designs and proposals put forward to resolve the security issues in P2PSIP. However, in solutions developed so far, the distributed feature for P2PSIP has not been the primary concern while implementing the secure architectures [6, 7, 22, 23]. In this section, we present SecP2PSIP, a simple and scalable architecture for distributed security management for SIP over a P2P Chord overlay network.

1 SYSTEM OVERVIEW

We illustrate the overview of the system in figure 1. The diagram shows that each node is a standard SIP client, and is connected to a SecP2PSIP adapter. Standard SIP operates by setting up the session parameters, and then going into the media session. Our proposed architecture works on top of the SIP layer, where the SecP2PSIP adapters form an overlay network, and allows a secured identity lookup and management for the SIP session establishment.

Initially, the SIP client registers with the local SecP2PSIP adapter. The adapter then communicates with the bootstrap server to obtain the Chord ring details, and joins the P2P overlay network of secured adapters. The adapters maintain the overlay network, and store the node lookup and public key information within the Chord DHT. When a client is willing to initiate a SIP session, he sends a SIP `INVITE` to its local adapter. The local adapter then coordinates with the SecP2PSIP adapter overlay to se-

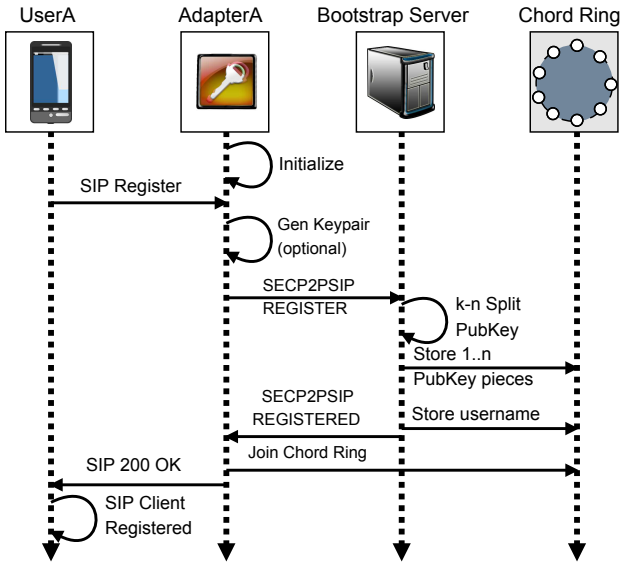

Figure 2: Sequence Diagram for SecP2PSIP Registration

curely retrieve the information required to initiate a standard SIP session. Once all the information is verified successfully, the local adapter sends a `SIP/2.0 302 MOVED TEMPORARILY` message to the local SIP client to redirect the client directly to the callee's SIP client. The following sections describe the protocol in further details.

2 SECP2PSIP REGISTRATION

The registration process in SecP2PSIP involves two phases. At the beginning, the SIP client tries to register with the local SecP2PSIP adapter. Next, the adapter registers with the bootstrap server, joins the Chord overlay, and finally sends a `200 OK` registration confirmation to the local SIP client. The sequence diagram for the registration process is illustrated in figure 2, and are explained as follows.

Adapter Initialization: First, the adapter initializes itself on the local device. The adapter is provided the IP address of the bootstrap server which the adapter uses to send the registration request. Alternatively, the adapter can obtain the bootstrap server information from the broadcast message in the base network. The adapter then awaits a standard registration request [2] from the SIP client.

SIP Client Started: The SIP client is configured with the local adapter as its registration server. As the SIP client is started, it sends a SIP `REGISTER` request to the SecP2PSIP adapter.

Key-Pair Generation: Once the adapter receives the registration request, it generates a RSA public private key-pair, and saves the key-pair on the device [24]. This is an optional step for the adapter, which can only be used when the ad-hoc deployment has a rather relaxed identity enforcement policy. In case of an environment with a higher security requirement, the system may require a certification authority signed certificate from the user. The user will then be expected to already have the signed certificate on the device.

SECP2PSIP REGISTER
Name: <SIP.Username>
IP: <Local.IP.Address>:<Adapter.Port>
PublicKey: <RSA.PublicKey.BigInteger>

Table 1: SecP2PSIP Register Message Format

SecP2PSIP Registration Request: Next, the SecP2PSIP adapter sends a SecP2PSIP registration request to the bootstrap server. To avoid constriction in the SecP2PSIP overlay, we suggest that there can be multiple bootstrap servers, each handling a subset of registration requests from the SecP2PSIP adapters. The SecP2PSIP registration request is shown in table 1.

Bootstrap Verification: The bootstrap server receives the `SECP2PSIP REGISTER` request from the adapter. The bootstrap server then verifies the registration request for the uniqueness of the SIP username and the validity of the public key. This is the only point in the sequence of operation which may require external network connectivity for verifying the certification authority signature on the public key.

Public Key Distribution: After that, the bootstrap server then executes the Shamir's Key Sharing Algorithm [25] to divide the public key for the user into *n* pieces (*PubPeyPiece*), with at least *k* pieces required for reconstructing the public key. Here, the value of *k* and *n* are flexible, and can be chosen according to the preferred security level. At this point, the bootstrap server stores the *n PubPeyPieces* in the DHT Chord overlay, on the nodes currently present within the Chord ring. Each *PubPeyPiece* is stored on the Chord overlay with a [*key:value*] pair, where the contents are as follows:

`key` : $Username.i$
`value` : $PubPeyPiece_i$; and $1 \leqslant i \leqslant n$

Lookup Information Distribution: Next, the bootstrap server stores the adapter lookup information in the Chord overlay network. The bootstrap server saves a [*key:value*] pair on the Chord ring, where the contents are as follows:

`key` : $SIP.UserName$
`value` : $Local.IP.Address : Adapter.Port$

Registration Success Response: The bootstrap server then responds to the registering SecP2PSIP adapter with a `SECP2PSIP REGISTERED` message after all the above steps are successfully completed. Additionally, the bootstrap server also includes the Chord overlay network in-

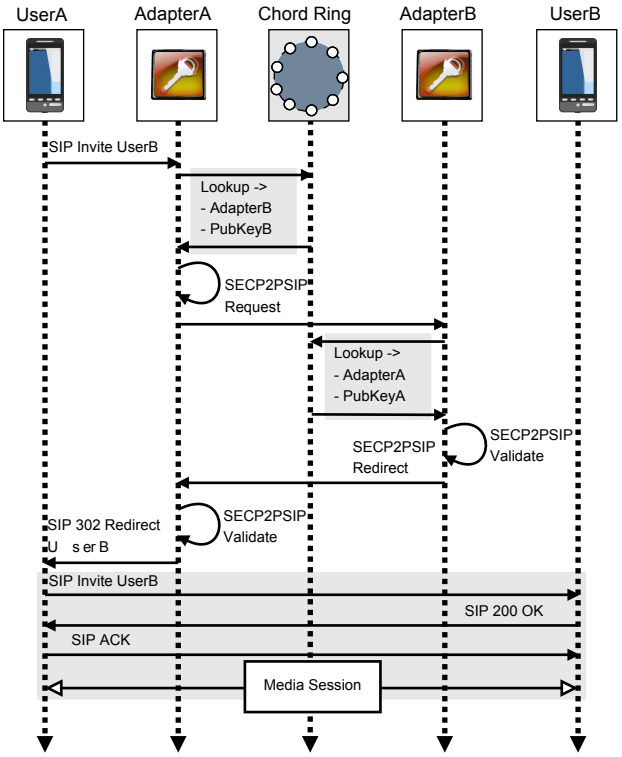

Figure 3: Sequence Diagram for SecP2PSIP Session Establishment

formation with the registration response.

SIP Client Registered: Upon receiving the success response from the bootstrap server, the adapter joins the Chord overlay network. The SecP2PSIP adapter then sends a standard `200 OK` response [2] to the SIP client in response to the SIP `REGISTER` request. The SIP client has thus completed its registration process with the local SecP2PSIP adapter.

3 SECP2PSIP SESSION ESTABLISHMENT

The process of establishing a SIP session in the SecP2PSIP architecture can be considered having two separate phases. In the first phase, we verify the source and the destination of the SIP call. Upon successful verification, we then redirect the SIP client to initiate a SIP session based on the securely negotiated parameters.

The sequence diagram for *userA* establishing a SIP session with *userB* over SecP2PSIP is illustrated in figure 3. The individual operations for establishing a session are explained as follows.

SIP Client INVITE Request: At the beginning, *userA* uses his SIP client to makes a standard SIP INVITE [2] request to *userB*. The IP and port information for the call is actually *userA*'s local SecP2PSIP adapter, in the form:

SECP2PSIP INVITE Name: userA SignedHash: $E_{userA.pvt}$(Hash(SIP Request)) SIP Request: <Original SIP INVITE>

Table 2: SecP2PSIP Invite Message Format

<sip:userB@userA.ipaddr:adapterA.port>

Contact Lookup: The local adapter for *userA* receives the SIP `INVITE` and initiates the SecP2PSIP operations from this point. *AdapterA* obtains the identity for *userB* from the SIP `INVITE`, and performs a Chord DHT table lookup for *userB* in the overlay. If successful, *AdapterA* obtains the contact information for *userB*'s adapter.

Public Key Reconstruction: Once *userB*'s contact information is retrieved, *AdapterA* then queries *k* random *PubKeyPieces* for *userB*'s public key. *AdapterA* therefore performs the following sequence of actions:

1. Execute query on Chord overlay:
   $\Rightarrow$ Query: [*key(userB) : value?*]
   $\Rightarrow$ Retrieve: *value = userB.IP:AdapterB.Port*
2. For count = *k* number of *PubKeyPieces*
   $\Rightarrow$ Randomly choose *j* ($1 \leqslant j \leqslant n$)
   $\Rightarrow$ Execute query on Chord overlay:
   $\Longrightarrow$ Query: [*key(userB.j) : value?*]
   $\Longrightarrow$ Retrieve: *value =* $PubKeyPiece_j$
3. Reconstruct public key for userB with *k* out of *n PubKeyPieces* using Shamir's Key Sharing Algorithm [25].

SecP2PSIP INVITE Request: Once *userB*'s public key has been reconstructed, *AdapterA* then sends *AdapterB* a `SECP2PSIP INVITE` message. This message can be sent in two different modes: 'signed' and 'protected'. In 'signed' mode, *AdapterA* only includes a signature using his own private key. This mode can be used when there is relatively less concern of anonymity and usage tracking for the SIP users. In 'protected' mode, in addition to the signature from *userA*, the `SECP2PSIP INVITE` message is encrypted using *userB*'s public key. The basic structure of the message is shown in table 2. Here, the *SignedHash* is created as $E_{userA.pvt}$*(Hash(SIP Request))*, using SHA-256 hash function on the original SIP `INVITE` request, and *userA*'s private key as the signature. Additionally, the original SIP `INVITE` request is appended with the `SECP2PSIP INVITE` message.

SecP2PSIP INVITE Verification: *AdapterB* receives the `SECP2PSIP INVITE` message, and validates the SIP request. At first, *AdapterB* constructs *userA*'s public key in a similar manner as shown earlier for *AdapterA*. Once the public key for *userA* has been reconstructed, *AdapterB* verifies the authenticity of the `SECP2PSIP INVITE` by decrypting the *SignedHash* parameter. The decrypted hash

is then compared to a SHA-256 hash of the <Original SIP INVITE> section created by *AdapterB* himself.

SecP2PSIP Redirection Request: *AdapterB* verifies the `SECP2PSIP INVITE` request and responds to *AdapterA* with a `SECP2PSIP REDIRECT` message. The structure of the SecP2PSIP message is shown in table 3. A standard `SIP/2.0 302 MOVED TEMPORARILY` request is created by *AdapterB* [2]. The redirection message is created such that the SIP call can be directly redirected to *userB*'s SIP client. Therefore, the `Contact` header is created using the local information, and the information from the previous SIP `INVITE` request, such that to redirect the SIP call directly to *userB*'s SIP client. Additionally, *AdapterB* creates a hash of the redirection message. The *SignedHash* is created as $E_{userB.pvt}$*(Hash(SIP Request))*, using SHA-256 hash function on the standard SIP redirection message, and signed using *userB*'s private key. All the parameters are then used together to form the `SECP2PSIP REDIRECT` request. As mentioned before, *AdapterB* can send the `SECP2PSIP REDIRECT` message in two modes, 'signed' and 'protected', which have already been discussed earlier.

SecP2PSIP Redirection Verification: *AdapterA* receives and verifies the `SECP2PSIP REDIRECT` request. It uses the previously reconstructed public key for *userB* to decrypt the *SignedHash*. The SIP request section from the `SECP2PSIP REDIRECT` message is hashed, and compared to the decrypted hash sent from *AdapterB*. In 'protected' mode, the encrypted `SECP2PSIP REDIRECT` request from *AdapterB* is first decrypted using *AdapterA*'s private key.

SIP Client Redirection: Once the authenticity and integrity of the `SECP2PSIP REDIRECT` is successfully verified, *AdapterA* sends the *SIP Request* section to *userA*'s SIP client, and the task of the SecP2PSIP overlay has successfully completed. Both the source and the destination have been verified, and the redirection parameters have also been asserted for its integrity and authenticity.

SIP Session Establishment: The SIP client for *userA* receives the `SIP/2.0 302 MOVED TEMPORARILY` request from *AdapterA*. The SIP client then redirects the call directly to *userB*'s SIP client. The redirection of the SIP client and the subsequent behaviors occur according to standard SIP definition [2], without the interference of the Chord overlay in the process.

V DESIGN ANALYSIS

In this section, we present the analysis and evaluation for the design, feature, architecture, and security of SecP2PSIP, with respect to the security challenges, system model, design goals, and characteristics of the system.

SECP2PSIP REDIRECT
Name: userB
SignedHash: $E_{userB.pvt}$(Hash(SIP Request))
SIP Request:
SIP/2.0 302 MOVED TEMPORARILY
To: sip:userB@userA.ip:adapterA.port
From: userA@userA.ip:adapterA.port
Call-ID: <from SIP INVITE>
CSeq: <from SIP INVITE>
Contact: sip:userB@userB.ip:client.port
Content-Length: 0

Table 3: SecP2PSIP Redirect Message Format

1. **Adjustable Security Mode**: The SecP2PSIP overlay allows 'signed' and 'protected' modes of operation. More security directly implies more computation, and hence more overhead. In some cases, it may not be required to have confidentiality for the exchanged messages. Thus, the 'signed' mode suffices the purpose of securing the SecP2PSIP architecture. In case of protected communication, the 'protected' mode offers the benefits of the 'signed' mode plus the confidentiality. This allows the architecture to be flexible and deployable in a wider variety of scenarios.
2. **Adjustable Key-Pair Requirement**: We previously mentioned that the key-pair can be generated on the SecP2PSIP adapter on an ad-hoc basis. This fulfills the requirement of verifying the source and the destination with our proposed scheme. However, a strict enforcement of identities for the users on the P2PSIP overlay cannot be imposed. In this case, we recommend that the system requires the users to pre-own public key certificates, which have been signed by a certification authority. The trusted signature is verified during the registration process and serves the purpose of an enforced and asserted identity for the users. This allows the system to be rather flexible and reduce complexity of deployment where it is not required.
3. **Enhanced Usability**: We use human-readable SIP URIs for identifying users on the SecP2PSIP overlay. The caller can place a call using only the username of the callee, and not requiring to know the IP information. The SIP `INVITE` is sent to the SecP2PSIP adapter, which coordinates with other adapters on the overlay network to securely find and establish the SIP session. In comparison to the other proposals for using cryptographic identifiers, our features of directing requests to the local adapter and using human-readable SIP URIs offer significantly enhanced usability.
4. **No Single Point of Failure**: The proposed SecP2PSIP architecture allows a distributed model for managing and enforcing secure verification within a P2P network overlay. It is discussed in section VII that most works

on securing P2PSIP overlays include some form of centralized identity server. Alternatively, there are proposals for using cryptographic identifiers or password based authentication using emails. However, the centralized server model suffers from a single point of failure. In that regard, our scheme uses Shamir's secret sharing [25] to divide and distribute the public key over the Chord DHT overlay. As a result, SecP2PSIP allows complete decentralization in the mode of operation for enforcing the security architecture. Additionally, this ensures that there is no centralized server for key management and thus no single point of failure, which preserves the true essence of P2P systems.

5. **Attack Containment**: SecP2PSIP uses Shamir's secret sharing to divide the public key of the users into *n PubPeyPieces*. Later, at least *k PubPeyPieces* are required for reconstructing the public key. Additionally, the *k PubPeyPieces* are chosen at random from the available *n* parts. Thus, the scheme offers two particular advantages. First, an attacker should be in control of at least *k* out of *n PubPeyPieces* to spoof a target identity. Second, the randomized selection of *k PubPeyPieces* ensures that there is no stable target for an attacker to be probabilistically successful in taking control of a node and spoofing an identity. Furthermore, we suggest that the values of *k* and *n* can be varied according to the desired level of security. Increasing the values of *k* and *n* implies increased security, but at the cost of reduced performance.
6. **Anonymity in Protected Mode**: Previous research suggested using a distributed naming service with cryptographic identifiers for anonymization. Other proposals include the use of pseudonymity services within the P2P overlay network to introduce anonymity in the communication between the calling and called parties. However, cryptographic identifiers completely remove human-readability and significantly reduce the usability. They also mandate the deployment of a directory lookup process for the users, which therefore complicates the ad-hoc deployment of the system. Furthermore, even though the pseudonymity services provide anonymity to the users, the service itself poses as a gold mine for attackers. Once an attacker gets control of such a service on the overlay, he can look into and exploit all communications between the SIP peers. In our proposed scheme, operating in 'protected' mode keeps all messages confidential. Thus, as a P2P network does not usually allocate IP addresses statically, tracking a user's activity is not possible knowing only the IP headers in the sent/received messages.
7. **Verification of SIP Parties**: SecP2PSIP ensures that the SIP parties are each who they claim to be. When a callee SecP2PSIP adapter receives a request, it verifies the identity of the calling party from the `SecP2PSIP INVITE` request. The calling party also verifies the `SecP2PSIP REDIRECT` message to ensure that it has reached the desired destination for the SIP session. This mutual verification ensures resistance against any man-in-the-middle attacks for identity spoofing.
8. **User Mobility**: The proposed scheme intends on preserving the true essence of P2P networking and ad-hoc deployment. The lookup information for the SecP2PSIP adapters are stored on the Chord DHT overlay. Additionally, the actual SIP communication between the SIP clients are separated from the SecP2PSIP overlay, which allows the user to enjoy mobility with his network attachment point. In a mobile environment, the user can run the SecP2PSIP adapter on a device which is guaranteed to maintain network connectivity at all times. The user can then use standard SIP clients on his other devices, and register the clients with the (persistent) adapter. Thus, this would allow the user to be able to securely send/receive calls from multiple devices, which is one of the key features of SIP telephony.
9. **Compatibility with Standard SIP**: SecP2PSIP works as a layer on top of the SIP functionality in P2P networks. Thus, the operation of SecP2PSIP is separate from the standardized SIP messages. This allows the architecture to be deployable using off-the-shelf SIP clients and other SIP-based services.

VI IMPLEMENTATION AND EVALUATION

In this section, we present the details of the prototype implementation for the proposed SecP2PSIP architecture. We also present extensive experimental measurements on the operation of the protocol to validate the feasibility of the protocol.

1 PROTOTYPE IMPLEMENTATION

We implemented a proof-of-concept prototype to validate the functionality of the proposed scheme. The SecP2PSIP adapter was implemented on standard Java platform. We used Open Chord, the open-source Java Chord DHT implementation [26]. We also used the open-source Java implementation of Shamir's Secret Sharing algorithm to create our own set of interfaces to serve our purpose for SecP2PSIP functionality [27]. The mentioned Java open-source packages were used to build the P2P overlay network for the SecP2PSIP adapters. Additionally, we used standard off-the-shelf SIP clients (KPhone[1], SJPhone[2]) to securely establish SIP sessions using the SecP2PSIP framework.

[1] KPhone, http://sourceforge.net/projects/kphone/
[2] SJPhone, http://www.sjphone.org/

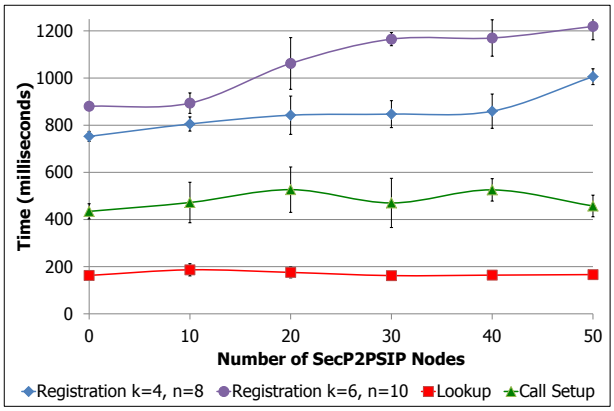

Figure 4: Performance Measurements for SecP2PSIP

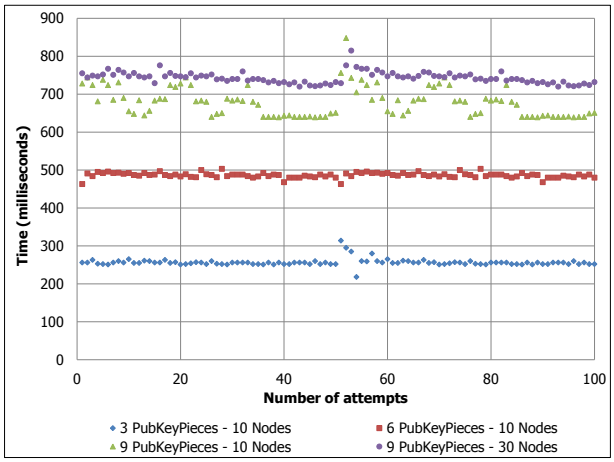

Figure 5: Performance Measurements for *PubKeyPieces* Reconstruction

The implemented prototype was tested on a local network. The testbed consisted of micro virtual machine (VM) instances with minimum system requirements. Each VM instance acted as individual users, running its own local adapter and a SIP client. For the testbed, we used a single bootstrap server, running on an individual VM instance. For each of the users, we used 256-bit RSA key-pairs, which were generated at run-time during the process of registration. However, in real-time practice, the users may also obtain their own signed certificates from any trusted authority.

2 PERFORMANCE EVALUATION

The prototype was deployed on the testbed for performing timing measurements for the protocol. We performed extensive experimental measurements on three different phases of the protocol to signify the percentage of time required for each phase. We measured each operation for 100 iterations, and for a total of six different cases, with 0 (initial node), 10, 20, 30, 40, and 50 running SecP2PSIP nodes. The results are illustrated in figure 4.

Initially, we measured the time required for a SIP peer to bootstrap and register with the SecP2PSIP overlay. We provided the bootstrap server contact information as a configuration parameter for initializing the adapter. The registration process includes dividing the public key into *PubKeyPieces* and storing them on the SecP2PSIP peers. We performed measurements for two different values of k and n: *k=4, n=8*, and *k=6, n=10*. It can be seen in figure 4, that the average time required for registration for *k=4, n=8* was lesser than that for *k=6, n=10*. This is natural, as larger values for k and n requires more time for processing and storing. It is also observed that there is a gradual increase in the required time, as the number of running SecP2PSIP nodes were increased. This is because the distribution of the *PubKeyPieces* are better achieved with more nodes running in the overlay network. The average time required with 0 nodes and *k=4, n=8* was 752.48 milliseconds, with 21.18 standard deviation. For the same case, the average time required with 50 nodes was 1005.80 milliseconds, with 33.66 standard deviation, which is only a 252.32 milliseconds increase from the previous value. For the same number of nodes, the average time required for *k=6, n=10* was 1219.66 milliseconds, which is 213.86 milliseconds longer, compared to the case for *k=4, n=8*. Additionally, the time required in the worst case is still acceptable in terms of feasibility and applicability of operation for the model.

After all the peers were registered, we measured the time required for the peer lookup process from the Chord DHT overlay. This is a prior process for the calling party to locate the destination node on the DHT overlay. As seen from figure 4, the time required for the lookup operation did not vary with the number of registered SecP2PSIP nodes. Considering all the cases, the average time required for the Chord lookup operation was 169.53 milliseconds, with an average standard deviation of 12.17. This value is significantly small and does not impose too much overhead, as because the lookup operation involves retrieval of the contact information from only one node, and the time measurement is not affected by more peers joining the network.

Next, we measured the total time taken for a client to negotiate the parameters for a SecP2PSIP session with another peer. The measurement was taken from after a `SIP INVITE` is sent by the SIP client, till when it receives the `SIP/2.0 302 MOVED TEMPORARILY` message from the SecP2PSIP adapter. As seen in figure 4, the time required for the call setup was fairly consistent for all the cases, as we increased the number of nodes to 50. The overall average recorded time the was 481.03 milliseconds, with 68.80 average standard deviation.

Finally, we performed experimental measurements on the

PubKeyPieces retrieval and key reconstruction operation. We recorded the times for three different cases of reconstruction, using *k=3, n=8* and *k=6, n=10*, with 10 running nodes, and using *k=9, n=10*, with 10 and 30 running nodes respectively. Figure 5 shows the recorded times for all the cases. The average time required for *k=3, n=8* and *k=6, n=10* was 256.48 milliseconds and 486.40 milliseconds respectively. In the case of *k=9, n=10*, the time required for 10 and 30 running nodes were 678.79 milliseconds and 743.45 milliseconds. Therefore, as shown in figure 5, increasing the value of *k* has a linear increase in time required for *PubKeyPieces* retrieval, while the number of running nodes does not impose too much constriction in the operation of retrieving the *PubKeyPieces*.

The given values show that the most time is required during the registration phase, and the value increases gradually when more peers are present in the network. The registration phase requires more time, primarily because the adapter sends the public key to the bootstrap server in *BigInteger* format. However, the registration is supposed to be done only once, and does not impose too much on the user in terms of time required. The times show that the lookup operation is approximately half of the total call setup time. Additionally, the performance of the public key reconstruction depends on the connectivity between the nodes, and the availability of the *PubKeyPieces*. Thus, the efficiency of the Chord DHT is a major concern for the given architecture. We used the open-source Java implementation of Open Chord. Therefore, the choice of the DHT overlay should emphasize performance efficiency and scalability for commercial deployments.

3 AVERAGE CALL OVERHEAD

The time required for the SecP2PSIP registration is a one-time process, and thus does not incur any overhead for subsequent SIP sessions for a user. However, we investigated the percentage of overhead that SecP2PSIP adds to an average length SIP call. Average Call Duration (ACD) refers to a well-known terminology and metrics used in network infrastructure monitoring, call volume forecasting, and traffic demand analysis. ACD is measured from the moment a call is answered till the time the call is ended. In 1996, ACD for telephone calls had a duration of 3 minutes [28]. With time and technology, the duration of calls increased. In 2006, it was seen that the median duration of a Skype call, the most popular VoIP application, lasted for 2 minutes 50 seconds. The ACD was actually seen to be way higher at 12 minutes 53 seconds [29]. The given numbers can be used to assume that current ACD for VoIP sessions last at least equal to or more than 12 minutes.

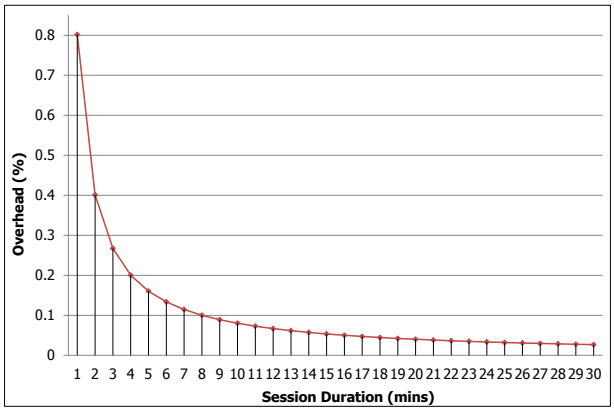

Figure 6: Overhead Percentage for Varying Call Durations

From the previous measurements, we found that the average time required for the SecP2PSIP framework to negotiate a SIP session was 481.03 milliseconds. Thus, from the above statistics, considering an ACD of 12 minutes for SIP calls, our framework will introduce less than 0.07% overhead in terms of the time required for the whole session. The percentage of relative overhead falls to values close to zero with increasing call durations, as displayed by the asymptotic shape of the graph in figure 6. Given that a call lasts for 1 minute, the relative overhead percentage will be 0.8%. The overhead decreases with longer call sessions, and drops below 0.1% for calls longer than 6 minutes, as shown in figure 6.

VII RELATED WORKS

There are many security issues with the general architecture of P2PSIP. The security challenges in P2PSIP have been analyzed by Seedorf *et al.* in [8] and [9]. Scheideler [30] proposed a model focusing the network availability. It gives a strict lower bound on the number of messages for the routing protocol which guarantees availability even in the presence of malicious nodes. There can also be man-in-the-middle attacks on the overlay routing of the P2PSIP network. Iterative routing strategies are useful to avoid such attacks. Danezis *et al.* in [31] proposed a variation of iterative routing, where the complete routing table is returned by all nodes instead of only the node closest to the key. The paper also introduced a trust based diversified routing to avoid routing security bottlenecks in the overlay. Our protocol is different in this regard, as our work does not exactly focus on securing the routing strategy of Chord or any other structured P2P overlays.

P2PSIP overlays assume that the SIP peers obtain random identities to join the P2PSIP overlay. Singh *et al.* in [32] proposed using a separate P2P network domain for each domain. Additionally, they also suggest password based authentication using emails before a peer joins a

P2P overlay. However, such mandatory processes introduce an operational constriction in the protocol. In our proposed protocol, we suggest that users can obtain their own signed certificates/identities from any trusted certification authority beforehand. As a result, we believe that this removes any significant constraint in run-time for the peer to join the overlay network. The signed certificate is verified by the bootstrapping peer during the SecP2PSIP secured registration process.

Other risks on P2PSIP systems include non-cooperative peers and violation of anonymity for the existing peers. Non-cooperating peers can be handled in the same way as for man-in-the-middle attacks, using iterative routing schemes [30, 31]. Preservation of anonymity can be enforced using distributed naming services [6]. The distributed naming service allows cryptographically generated SIP URIs, which are used to authenticate nodes when joining an overlay. Seedorf *et al.* proposed another scheme to use cryptographically generate SIP URIs for protecting the integrity of SIP messages [33]. Pseudonymity services can also be used to implement anonymous trusted message delivery [34]. This acts as a back-to-back user agent within the overlay to provide anonymity services for the peers in the network by replacing the headers for exchanged messages. Zheng *et al.* in [7] proposed a proxy-based architecture using Chord Secure Proxys (CSP). The CSP is used to join and perform subsequent operations of session initiation over a P2P overlay using a logical trust model for peer verification. However, the proposed scheme requires multiple messages before a session is established and uses a probabilistic model for randomly selecting a request for routing through a CSP. In our proposed protocol, we allow two modes of operation: 'signed' and 'protected'. In 'signed' mode, all contents are signed by the sender. In protected mode, all messages are encrypted using the public key of the receiver, and thus preventing the violation of privacy and usage tracking of the calling party over the SecP2PSIP network. Moreover, we can enforce additional security on the call establishment of the peers using provenance of the request packets, using similar approaches for location proofs proposed by Khan *et al.* in [35, 36].

VIII CONCLUSION AND FUTURE WORKS

P2P networks are always popular for distributed functionality and ad-hoc systems. We showed how P2P networks with DHT based overlays have initiated the research for P2PSIP. Unfortunately, the security in such P2PSIP overlay networks has always been a challenge. However, proposed solutions either reduce the usability for users, or introduce a centralized model to enforce the security.

In this paper, we have proposed a novel solution for a distributed security architecture, SecP2PSIP, which works as a separate overlay on top of the SIP signaling session. The proposed architecture ensures the design goals discussed in section V. In addition to the points mentioned, the scheme is designed in an easy-to-deploy manner. As a result, it can be ideal for ad-hoc use cases, such as in conferences and symposiums, to assist messaging for aid workers, and for internal communication systems in small and medium sized enterprises. The presence of an unethical personnel may exist in any environment. Thus, in case any user on the SecP2PSIP network is exposed by an attacker, the risks are still contained within a certain limit. The distributed nature of the model preserves the true essence of P2P networks, and ensures no single point of failure without limiting usability for the users. We have presented a detailed design analysis, followed by a proof-of-concept prototype implementation.

The primary target of the proposed scheme was to ensure a mutually authenticated SIP session in a distributed fashion. Thus, the performance analysis of the Chord DHT overlay is outside the scope of this work. There are improved versions of Chord DHT, such as F-Chord [37] and PChord [38], which can be used instead of the vanilla Chord as discussed in the implementation section. The implementation of a proof-of-concept prototype illustrates the feasibility of the scheme. Furthermore, the measurements from different phases of the the proposed were used to signify the importance of the different technologies used in the implementation. Hence, our proposed design can be implemented using any other underlying technology, which serves the purpose, and improves the efficiency of the targeted system.

Currently, identity enforcement for the optional ad-hoc generation of certificates is absolutely not secure. We will therefore extend the model with a trust based metrics to generate chains of trusted relations. This information will be handled in a distributed fashion, as per the requirements of a distributed security architecture. We are also working on a secure and automated procedure for bootstrapping the SecP2PSIP nodes. The secure automation will thus greatly improve the usability as well as the security of the proposed scheme.

ACKNOWLEDGMENT

This research was supported by a Google Faculty Research Award, NSF Grant #0937060 to the Computing Research Association for the CIFellows Project, and the DHS Grant #FA8750-12-2- 0254.

References

[1] D. Schoder, K. Fischbach, and C. Schmitt, *Core Concepts in Peer-to-Peer Networking*. Idea Group Publishing, Jun 2005.

[2] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler, "RFC 3261: SIP - Session Initiation Protocol," *Internet Engineering Task Force (IETF)*, 2002.

[3] H. Sinnreich and A. B. Johnston, *Internet communications using SIP: delivering VoIP and multimedia services with Session Initiation Protocol*. New York, USA: John Wiley & Sons, Inc., ISBN 0-471-41399-2, 2001.

[4] D. Bryan, E. Shim, and B. Lowekamp, "Use Cases for Peer-to-Peer Session Initiation Protocol (P2P SIP)," *Internet draft, http://www.p2psip.org/drafts/draft-bryan-sipping-p2p-usecases-00.html*, Nov 2005.

[5] P2PSIP IETF Working Group, http://www.ietf.org/html.charters/p2psip-charter.html, Last accessed May 2001.

[6] I. Baumgart, "P2PNS: A Secure Distributed Name Service for P2PSIP," in *Proceedings of The 6th Annual IEEE International Conference on Pervasive Computing and Communications*, ser. PerCom '08, Mar 2008, pp. 480–485.

[7] X. Zheng and V. Oleshchuk, "The Design of Secure and Efficient P2PSIP Communication Systems," in *Information Security Theory and Practices: Security and Privacy of Pervasive Systems and Smart Devices*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2010, vol. 6033, pp. 253–260.

[8] J. Seedorf, "Security challenges for peer-to-peer SIP," *Network, IEEE*, vol. Vol. 20, no. No. 5, pp. 38–45, Sept 2006.

[9] J. Seedorf, F. Ruwolt, M. Stiemerling, and S. Niccolini, "Evaluating P2PSIP under Attack: An Emulative Study," in *Proceedings of The Global Telecommunications Conference*, ser. IEEE GLOBECOM '08. IEEE, Nov 2008, pp. 1–6.

[10] J. Hautakorpi and G. Schultz, "A Feasibility Study of an Arbitrary Search in Structured Peer-to-Peer Networks," in *Proceedings of 19th International Conference on Computer Communications and Networks*, ser. ICCCN '10, Sept 2010, pp. 1–8.

[11] C. Jennings, B. Lowekamp, E. Rescorla, S. Baset, and H. Schulzrinne, "Draft, REsource LOcation And Discovery (RELOAD) Base Protocol," *Internet Engineering Task Force (IETF)*, Mar 2010.

[12] P. Maymounkov and D. Mazières, "Kademlia: A Peer-to-Peer Information System Based on the XOR Metric," in *Revised Papers from the First International Workshop on Peer-to-Peer Systems*, ser. IPTPS '01. London, UK: Springer-Verlag, 2002, pp. 53–65. [Online]. Available: http://dl.acm.org/citation.cfm?id=646334.687801

[13] A. Rowstron and P. Druschel, "Pastry: Scalable Decentralized Object Location and Routing for Large-Scale Peer-to-Peer Systems," in *Proceedings of The IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, Heidelberg, Germany, Nov 2001, pp. 329–350.

[14] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A scalable content-addressable network," in *Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, ser. SIGCOMM '01. New York, NY, USA: ACM, 2001, pp. 161–172. [Online]. Available: http://doi.acm.org/10.1145/383059.383072

[15] I. Stoica, R. Morris, D. Liben-Nowell, D. Karger, M. Kaashoek, F. Dabek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup protocol for Internet applications, Version 2," *IEEE/ACM Transactions on Networking*, vol. 11, no. 1, pp. 17–32, Feb 2003.

[16] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications, Version 1," in *Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, ser. SIGCOMM '01. New York, NY, USA: ACM, 2001, pp. 149–160. [Online]. Available: http://doi.acm.org/10.1145/383059.383071

[17] K. Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim, "A Survey and Comparison of Peer-to-Peer Overlay Network Schemes," *IEEE Communications Surveys & Tutorials*, vol. 7, no. 2, pp. 72–93, 2005.

[18] I. Martinez-Yelmo, R. Cuevas, C. Guerrero, and A. Mauthe, "Routing Performance in a Hierarchical DHT-based Overlay Network," in *Proceedings of The Euromicro Conference on Parallel, Distributed, and Network-Based Processing*. Los Alamitos, CA, USA: IEEE Computer Society, 2008, pp. 508–515.

[19] M. Engle and J. I. Khan, "Vulnerabilities of p2p systems and a critical look at their solutions," Kent State University, Tech. Rep., 2006.

[20] D. S. Wallach, "A survey of peer-to-peer security issues," in *Software Security - Theories and Systems*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2003, vol. 2609, pp. 42–57. [Online]. Available: http://dx.doi.org/10.1007/3-540-36532-X_4

[21] J. Douceur, "The Sybil attack," *Peer-to-peer Systems*, pp. 251–260, 2002.

[22] X. Zheng and V. Oleshchuk, "A secure architecture for P2PSIP-based communication systems," in *Proceedings of The 2nd International Conference on Security of Information and Networks*, ser. SIN '09. New York, NY, USA: ACM, 2009, pp. 75–82. [Online]. Available: http://doi.acm.org/10.1145/1626195.1626216

[23] ——, "Trust enhancement of p2psip communication systems," *International Journal of Internet Technology and Secured Transactions*, vol. Vol. 3, no. No. 2, pp. 121–133, April 2011. [Online]. Available: http://dx.doi.org/10.1504/IJITST.2011.039773

[24] J. Jonsson and B. Kaliski, "RFC 3447: Public-Key Cryptography Standards (PKCS)# 1: RSA Cryptography Specifications Version 2.1," *Internet Engineering Task Force (IETF)*, 2003.

[25] A. Shamir, "How to share a secret," *Communications of the ACM*, vol. Vol. 2, no. Issue 11, pp. 612–613, Nov 1979. [Online]. Available: http://doi.acm-org/10.1145/359168.359176

[26] Open Chord, http://open-chord.sourceforge.net/. Last accessed June 2013.

[27] T. Tiemens, "Shamir Secret Sharing in Java," http://sourceforge.net/projects/secretsharejava/.

[28] A. M. Noll, "Cybernetwork technology: issues and uncertainties," *Communications of the ACM*, vol. Vol. 39, no. No, 12, pp. 27–31, Dec 1996. [Online]. Available: http://doi.acm.org/10.1145/240483.240488

[29] S. Guha, N. Daswani, and R. Jain, "An Experimental Study of the Skype Peer-to-Peer VoIP System," in *Proceedings of The 5th International Workshop on Peer-to-Peer Systems*, ser. IPTPS '06. Santa Barbara, CA, USA: USENIX, Feb 2006.

[30] C. Scheideler, "How to spread adversarial nodes? Rotate!" in *Proceedings of The 37th Annual ACM Symposium on Theory of Computing*, ser. STOC '05. New York, NY, USA: ACM, 2005, pp. 704–713. [Online]. Available: http://doi.acm.org/10.1145/1060590.1060694

[31] G. Danezis, C. Lesniewski-Laas, M. F. Kaashoek, and R. Anderson, "Sybil-resistant DHT routing," in *Proceedings of the 10th European conference on Research in Computer Security*, ser. ESORICS'05. Berlin, Heidelberg: Springer-Verlag, 2005, pp. 305–318. [Online]. Available: http://dx.doi.org/10.1007/11555827_18

[32] K. Singh and H. Schulzrinne, "Peer-to-peer internet telephony using SIP," in *Proceedings of the International Workshop on Network and Operating Systems Support for Digital Audio and Video*, ser. NOSSDAV '05. New York, NY, USA: ACM, 2005, pp. 63–68. [Online]. Available: http://doi.acm.org/10.1145/1065983.1065999

[33] J. Seedorf, "Using Cryptographically Generated SIP-URIs to Protect the Integrity of Content in P2P-SIP," in *Proceedings of The 3rd Annual VoIP Security Workshop, Berlin, Germany*, ser. VSW '06, Jun 2006.

[34] J. Posegga and J. Seedorf, "Voice over IP: Unsafe at any Bandwidth?" in *Proceedings of Eurescom Summit 2005 - Ubiquitous Services and Applications*, 2005, pp. 305–314.

[35] R. Khan, S. Zawoad, M. Haque, and R. Hasan, "OTIT: Towards secure provenance modeling for location proofs," in *Proceedings of the 9th ACM Symposium on Information, Computer, and Communications Security*, ser. ASIACCS '14. ACM, June 2014.

[36] R. Khan, S. Zawoad, M. M. Haque, and R. Hasan, "Who, When, and Where? Location Proof Assertion for Mobile Devices," in *Proceedings of the 28th Annual IFIP WG 11.3 Working Conference on Data and Applications Security and Privacy*, ser. DBSec. IFIP, July 2014.

[37] G. Cordasco, L. Gargano, A. Negro, V. Scarano, and M. Hammar, "F-Chord: Improved uniform routing on Chord," *ACM Networks, Wiley-Interscience, New York, NY, USA*, vol. Vol. 52, no. No. 4, pp. 325–332, Dec 2008. [Online]. Available: http://dx.doi.org/10.1002/net.v52:4

[38] F. Hong, M. Li, J. Yu, and Y. Wang, "PChord: improvement on chord to achieve better routing efficiency by exploiting proximity," in *Proceedings of The 25th IEEE International Conference on Distributed Computing Systems Workshops*, 2005, pp. 806–811.