

# A Lightweight Scheme for Securely and Reliably Locating SIP Users

Lei Kong, Vijay Arvind Balasubramaniyan and Mustaque Ahamad  
 College of Computing  
 Georgia Institute of Technology  
 Atlanta, Georgia 30332  
 {konglei,vijayab,mustaq}@cc.gatech.edu

## Abstract

*Voice over Internet Protocol (VOIP) is a fast growing technology believed to be the future replacement for traditional Public Switched Telephone Network (PSTN) networks. VOIP offers many benefits over PSTN, but there are still lots of issues when it comes to service reliability and security.*

*One of the problems faced is how to locate communicating parties on the Internet in a secure and reliable manner. The security mechanisms included in RFC 3261 Session Initiation Protocol (SIP) are either weak or expensive to deploy. In this paper, we propose a light weight scheme that can protect the integrity of SIP contact addresses. This scheme also achieves high availability of SIP services through replication. End user public key distributed in this scheme can also be used for end-to-end user authentication and media session key exchange.*

## 1 Introduction

A VOIP media session is, by nature, a peer-to-peer connection, i.e. a terminal device/software should be able to contact another terminal without intermediaries involved. On the other hand, the called party needs to be located on the Internet before a media session can be established. Static setup or manual configuration works fine for small groups, but for large scale public use, some infrastructure support is needed in order to discover/locate VOIP communication endpoints. Determination of user location is one of the major functions of SIP protocols.

In SIP, proxy/redirect servers, registrar and location service are the logical entities responsible for the discovery process. Terminal device/software registers its current address to registrars, which store address bindings in location services. Proxy services query location services to retrieve current address bindings.

The integrity of mapping from SIP URI to contact address is critical to the security and reliability of VOIP communication. Incorrect contact address might lead to denial of service or even user impersonation. The services mentioned above need to be highly available, tolerant to hardware/software failures, host compromises, and resistant to denial-of-service attacks. In addition to locating callees, it is also desirable to distribute information such as user public keys to enable mutual authentication and voice data encryption between caller and callee.

In this paper, we will present a light weight scheme that can be used to locate SIP users securely using signatures, and achieve high dependability of SIP locating functionality by replication. Compared with other solutions, our scheme effectively makes use of the existing SIP structure, imposes lower performance overhead and offers better manageability. We do assume future endpoints will have reasonable amount of resources needed for computing user signatures and to encrypt media sessions when confidentiality is a concern.

## 2 SIP structure

Figure 1 shows how different logical entities interact with each other in SIP protocols. A SIP phone (terminal device or softphone) registers its contact address (address of the host where it is located) to registrar servers, which then stores the address binding (the association between SIP URI and contact address) with location services.

During a call setup, the calling party sends its invitation request to his local outbound proxy. The outbound proxy finds out the proxy that is responsible for the destination domain through a particular type of DNS lookup [7], and forwards the request to it. The destination domain proxy will query its location service for the called party's current contact address and then forward the request to that address. The callee's response message goes through the same channel in the reverse direction back to the caller.

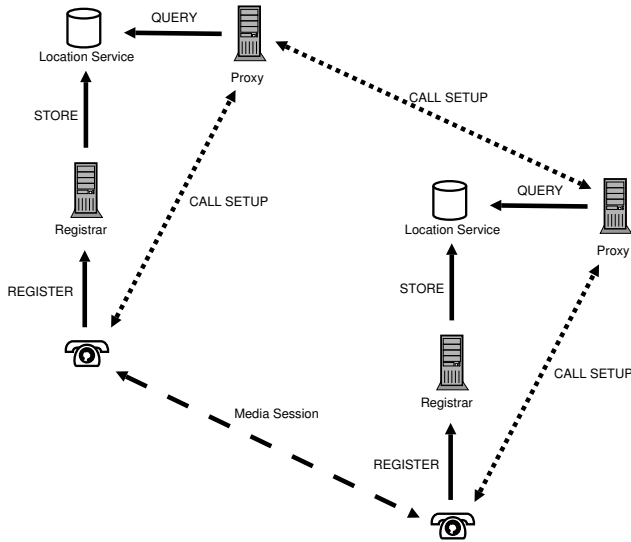


Figure 1. SIP Interactions

### 3 SIP User Contact Address Integrity

As one can see, it is desirable that caller can establish the integrity of the callee's current contact address, especially when the caller is trying to reach someone of importance, whom he does not know well, such as the customer service desk of the caller's bank. Plain SIP provides little or very weak protection on the contact address. By tampering with callee's contact address during call setup, attackers can direct calls to places under their control and impersonate the real callee, or simply cause denial of service.

#### 3.1 Existing Solutions

HTTP digest authentication is used in SIP to authenticate the originator of SIP messages, but it does not provide integrity protection on SIP messages. It can not provide end-to-end protection, and it is difficult to expand beyond a single administration domain since it is based on shared user passwords.

One possible solution is to let SIP registrar services sign SIP user contact address or address bindings. Similar techniques have been proposed to use digital signatures from a trusted authority to protect domain name system (DNS) resource records in DNS security extension [2]. Generally, it is desirable to keep the signer offline to ensure security. Compromise of the signer host usually leads to exposure of the signing private key. Some have suggested using threshold signature or simply multiple signatures to address this problem [10, 3]. Such approaches are computationally too expensive when the signer needs to serve a large number of users who generate requests constantly. There are also diffi-

culties associated with key management that is required for such a signature scheme.

In SIP, an address binding usually expires in a matter of hours. When it does, a new registration process is needed to renew address bindings. Also, mobile users needs to register their new address when their locations change. A SIP domain can have a large number of clients (hardphones or softphones), which impose a constant load on the registrar service. If the registrar service needs to compute a signature for each request, it could easily be overloaded by expensive signing operations, making it more vulnerable to denial-of-service attacks.

SIP over SSL (SIPS) can provide end-to-end protection on SIP request/response messages, so it can be used to protect SIP contact addresses. But it requires end user certificates in place. We agree with J. Rosenberg et. al. on the statement "Any mechanisms depending on the existence of end-user certificates are seriously limited in that there is virtually no consolidated authority today that provides certificates for end-user applications." [8]. After so many years, public key infrastructures are still limited to a small scale or closed domains. As in HTTPS, usually only servers have public key certificate signed by certificate authorities in public domain. The issues associated with managing a high volume of user certificates still pose a hurdle to large scale deployment of public key infrastructure. Besides, using SIPS increases the workload of SIP proxy servers significantly.

A new authentication service is proposed to authenticate SIP message origin in [13]. The authentication service signs client SIP messages using its domain certificate before forwarding these messages. Thus, these messages can be verified outside the sender's home domain without requiring end-user certificate. However, the authentication service approach has scalability issues since what it authenticates are SIP messages, which are very frequent, and the signing operations are computationally intensive. Our scheme presented in the next section also addresses the end-user certificate problem, but does not incur such performance loss.

#### 3.2 Our Scheme

Instead of burdening registrar services, we propose to let SIP phone devices sign their own contact address/address bindings on behalf of their users, thus their integrity can be verified through users' public keys. This imposes lower workload on registrar servers, compared with the approach in which a registrar signs address binding/contact address. Our approach does not rely on end-user certificates. Instead, we make use of SIP architecture itself to distribute user public keys.

We assume all involved SIP servers have certificates issued by some well-known public authority, and SIP servers

in the caller domain and the callee domain trust each other to correctly establish contact identity and address binding for their own domains. Thus, we do not require end-user certificates that can be checked by service outside of the domain of the end-user. In this section, all servers are assumed to be non-malicious. We will relax this assumption when dealing with fault/intrusion tolerance in section 4. For simplicity, it is also assumed in this paper that an instance of SIP proxy co-locates with an instance of SIP registrar and SIP location service on a single host. The clock on all hosts are assumed to be loosely synchronized. In standard SIP, the called party sends its contact address in its response to an invitation request. In our scheme, both the called party and the proxy in its local domain can send the callee's contact address back to the calling party, because the proxy needs to check the callee's contact address with the location service in order to reach the callee anyway. The same applies to a user public key.

### 3.2.1 User Signature

Each user generates its own public key/private key pair, stores its private key securely in its SIP phone and registers its public key with its local registrar. We do not explicitly address the issue of authentication here but the registration can initially use some mechanism to ensure that the public key belongs to the entity that register it. When doing SIP address registration, a client signs its address binding and sends it to its registrar. The registrar can check the registration request by verifying client's signature since it has the client's public key. Generally, verifying a signature is much faster than generating a signature for most popular public key algorithms. This reduces the workload of registrar considerably compared to a registrar signing approach. During SIP message exchange, users exchange contact addresses signed by their private keys, instead of responding to an invitation request with a plain contact address. If the caller and the callee have each other's public key (section 3.2.2 details how this is implemented), the caller can verify the contact address in the response. The same signature protection can be used for other message fields. Such a scheme imposes minimal additional workload on proxy servers compared to plain SIP.

Another benefit of user signing is that we do not need to worry about the compromise of the signer host, since that is the SIP phone client host itself. If the client is compromised, VOIP communication will be compromised anyway. In the case of registrar signing, compromise of signer will affect the whole domain that the signer is responsible. Periodically, clients re-generate their public key/private key pair and register the new public key with its registrar service to limit the damage of private key exposure in case such exposure did happen. The first time a client registers

its public key, it can go through some out-of-band channel or have some out-of-band verification. Once that is done, future public keys can be registered with the registrar in a secure way by using currently registered public keys.

### 3.2.2 Distributing Client Public Key Across SIP Domains

The integrity protection on SIP address binding/contact address by user signature rely on the assumption that user public key is known to the verifier. Inside the same SIP domain, this can be easily implemented, but some mechanism is needed to pass a user public key across SIP domains. We are avoiding using end-user certificates, since we do not want to rely on public key infrastructure to authenticate user public keys. Passing self-signed user certificates or plain user public key is vulnerable to man-in-the-middle attack since user public key is not authenticated. There are ways to detect such attacks in certain situations [14], but not to prevent them.

Our solution to this problem is to distribute user public key (not signed) through secure and trusted channels across SIP domains. User public key will not be used for non-repudiation purpose in our scheme, so it is acceptable not to sign user public key.

We explain this process in the following example. User Alice in domain 1 tries to call user Bob in domain 2. The first time Alice calls Bob, she does not have Bob's public key. So, she indicates in the invitation request that she needs Bob's public key. Alice's SIP phone will establish an SSL/TLS channel to a proxy of domain 1 using both her public key and the public key of her outbound proxy in domain 1. Similarly, there can be an SSL/TLS channel between Bob's phone and his proxy server. According to our assumption, proxies in domain 1 and domain 2 can find a trust path through a chain of certificates signed by certificate authorities in public domain, then they can also set up an SSL/TLS channel using their public keys. Chained together, there is a secure channel from Alice's phone to Bob's phone.

Through this secure channel, Alice sends her public key to Bob, and gets Bob's public key from the contacted proxy in domain 2. The contacted proxy queries its location service to retrieve Bob's registered public key. An alternative is that Alice gets Bob's public key directly from Bob through the chained secure channel. Public keys obtained in this way can be trusted since each proxy has the authority to link a public key to a user in its domain.

After receiving each other's public key, Alice and Bob will store them in their local key ring. With Bob's public key in hand, Alice can verify Bob's contact address since it is signed using Bob's public key. Only after Bob refreshes its public/private key pair, Alice needs to re-run the above

process to get Bob's new public key. In the meantime, Alice does not need to rely on the SSL/TLS channel chain mentioned above to ensure the integrity of address binding. That is to say Alice can run "plain" SIP to setup future calls to Bob after getting Bob's public key. This means less overhead on proxy servers in both domains than always using SSL/TLS to set up calls as in SIPS, because the chained SSL/TLS channel is only needed when new user public key needs to be retrieved. Typically, a user's public key expires much less frequently than the user's address binding.

### 3.2.3 Intra-Domain Authentication

In the above scheme, users register their public keys with the registration service in their home domain. The authentication of a user to its home domain registrar/proxy services is based on the user's public key. But, our scheme does not require that a user registers its public key with its home domain registrar for intra-domain user authentication. Username/password or some symmetric key based authentication can still be used to authenticate users to their home domain. As long as SIP services have certificates, trusted path for cross domain user public key exchange can be established through multi-hop TLS/SSL session. If user public keys are not registered with the registrar, the use of user public key will be completely transparent to proxy servers.

### 3.2.4 User Public Keys Can Serve Other Purposes

With public keys distributed in the way mentioned above, SIP users can actually do mutual authentication, protect integrity and confidentiality of both SIP messages and media session when necessary. Examples are CANCEL and BYE messages, which can be easily used to launch denial-of-service attack when not protected. These public keys are managed by client hosts, imposing little performance and management overhead on servers, which makes this scheme light-weight for the servers that must handle large number of requests.

## 3.3 SIP Extension

In SIP, the *Contact* header field specifies the current address for the caller as part of the INVITE message and the callee as part of 200 OK response. The *Contact* header is also specified in the 180 RINGING response. However, the 200 OK response is used to determine the callee location. Therefore we check the validity of the caller's current address in the INVITE message and the callee's current address in the 200 OK response. The necessary SIP extensions for our scheme will thus be a part of these two messages. Our scheme requires transfer of public key information of both parties and a digital signature on the Contact address for verification purpose.

To address this, both the INVITE message and the 200 OK response are extended to have the contact address signature and the user public key as part of the SIP message body. The contact address signature will be a part of all INVITE messages and 200 OK responses. On the other hand, since the public key must be transmitted securely only the first time or when it changes, we propose a header *PK-Exchange* which indicates that the INVITE request contains the public key of the caller and that the intermediate proxies must set up a secure channel between them to pass this key to the callee. In response message, the 200 OK from the callee will now carry its public key and will also be passed through this secure channel to the caller. The public key and the contact address are passed in the body because they can be specified in special formats and are too large for the header. The appropriate *Content-type* should also be specified in the header.

When *PK-Exchange* header is not present, the intermediate proxies will not set up SIPS secure channel to protect SIP messages. Instead, *Contact* message header will be protected solely by user signature. This ensures light work load on proxy servers. In case validation fails, the callee needs to reply with an appropriate 4xx response. In this case the caller must fallback and try to reestablish the session using a multi-hop SIPS channel, once again passing its key. This is because the failed validation may be due to an old key cached at the callee.

End points that have these SIP extensions must realize that if the 200 OK response does not have a contact signature then they are talking to an older client and the address binding validation cannot be guaranteed. In such a case the end point can decide to drop the call or continue it with the knowledge that the integrity of the contact address has not been established. The older clients typically will discard this newer information. If this is the interaction, then our scheme will, to the best of our knowledge, not break any existing deployments.

Figure 2 and 3 illustrate the message exchange with these extensions. Some sample SIP messages are shown in the appendix.

## 4 Fault/Intrusion Tolerance of SIP Servers

We assumed that SIP servers are non-malicious for protocols presented in the previous section. In reality, they can fail and thus stop functioning. SIP servers can even be victims of network intrusion. Controlled by intelligent adversary, they can exhibit arbitrary behavior, suffering from Byzantine faults [6]. In this section, we introduce protocols to deal these issues.

Replication is a basic technique for fault tolerance. Redundant SIP proxy service has been recommended in [7] for better availability, which is straightforward since only crash

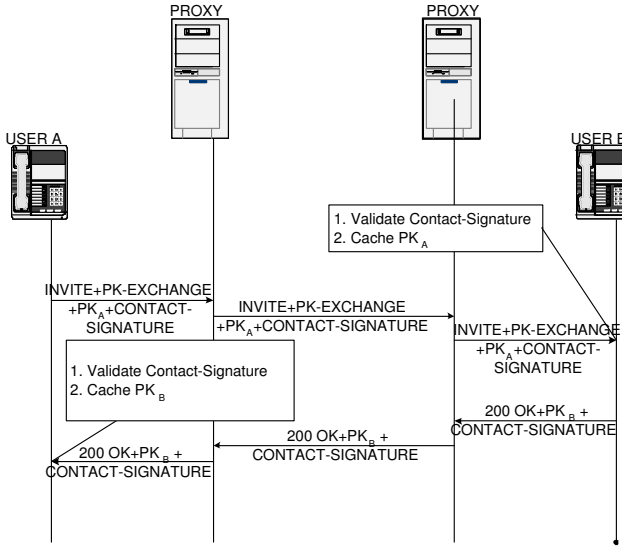


Figure 2. Initial Session Setup

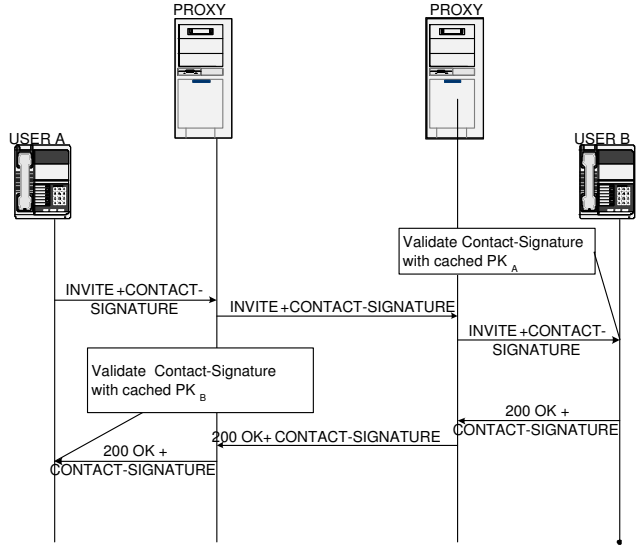


Figure 3. Session Setup After User Public Key Acquired

faults are considered. As assumed in the previous section, SIP proxy service, its registrar and location service are co-located on a single host, for simplicity of discussion without losing generality.

#### 4.1 Byzantine Quorum Systems

The data needed to be replicated in our scheme include user public key, SIP address binding/contact address. Generally speaking, these data do not require atomic consistency [5]. Read operations that are not concurrent with any write operation need to return the current version of data. If there are concurrent write operations, read operations can return either the latest version before write operations happen or what is being written. In SIP context, there is no need to order multiple read operations that all overlap with a single write operation.

Byzantine quorum systems can achieve the consistency level required by SIP data. Quorum based protocols are generally more efficient and scale better than protocols that achieve atomic consistency, such as state machine approaches [9]. In quorum systems, data read/write operations are done with quorums, subsets of all servers. This means not all servers need to be available for every data operation, and thus allows quorum systems to provide higher availability than other replication schemes. This also means a quorum-based protocol imposes lighter average workload since servers do not need to be involved in all data operations. Another merit of quorum-based protocols is its better scalability due to low message complexity, compared with other fault tolerant mechanisms like state machine approach. For these reasons, we choose Byzantine

quorum protocols to access replicated data. In our proposed scheme in section 3, there are two different kind of data: user public key and address binding. Client public key is generic data, which can be read/written by masking Byzantine quorums[6]. Address binding is self-verifiable data (signed by clients), which can be accessed through dissemination Byzantine quorums [6].

##### 4.1.1 User Public Key Protocol

Suppose we have  $N$  replicas of SIP services for a given domain, and we want to tolerate up to  $B$  Byzantine faults. Clients execute algorithm 1 to register its public key to multiple registrar services. Algorithm 2 is used to retrieve a user's public key.

<p><b>Input:</b> <math>K</math>: user public key</p> <ol style="list-style-type: none"> <li>1 send <math>K</math> and current time to all <math>N</math> servers ;</li> <li>2 <b>while</b> have not got <math>(N - B)</math> server confirmation <b>do</b></li> <li>3   wait for more server responses ;</li> <li>4 <b>end</b></li> </ol>
---

Algorithm 1: Client Public Key Registration

Here the write quorum of user public key is  $N - B$ , and the read quorum is  $3B + 1$ , so the intersection size is at least  $(N - B) + (3B + 1) = 2B + 1$ . This guarantees that at least  $B + 1$  non-compromised servers are in the intersection, and they will return the latest user public key. Figure 4 provides an example of quorum operation on user

**Input:**  $K$ : user public key id

```

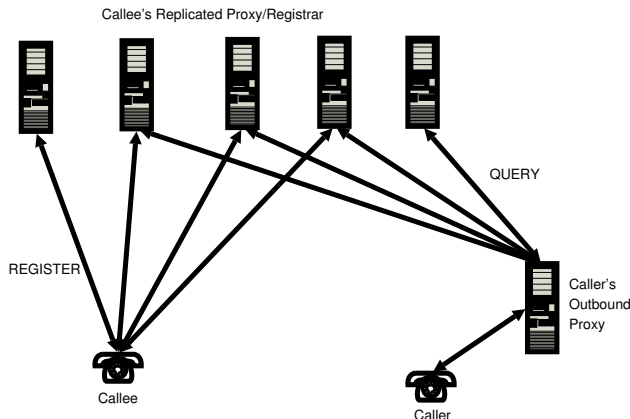
1 repeat
2   read  $K$  from  $3B + 1$  servers ;
3    $result :=$  newest  $K$  returned by at least  $B + 1$ 
   servers ;
4 until no more than  $B$  responses are newer than  $result$ ;
5 return  $result$  ;

```

**Algorithm 2:** Query User Public Key  $K$

public keys. In a system with 5 proxy/registrar replica, fault threshold  $B$  is set to be 1, both read quorum size and write quorum size equals 4. So, client registers with 4 registrars and 4 proxy replica are queried during a call setup. The intersection between a read quorum and any write quorum contains at least 3 servers, this guarantees that at least 2 queried proxy servers participated in the latest public key registration and they are not faulty or compromised. These 2 servers will return to the querier the latest public key.

Because a user is the only writer of his public key, client side physical clock time can be used as the timestamp of user public key, instead of a logical timestamp usually used in Byzantine quorum systems. In the query protocol, we check if there are more than  $B$  responses newer than the chosen  $result$ , and restart the query operation if so. This *countermanding* technique [1] is used to detect ongoing update operation on user public key and thus make sure old values are not returned. We believe that query operations are more frequent than updates, so we use the largest possible write quorum  $N - B$  for a given system size  $N$ , which leads to the minimum read quorum size of  $3B + 1$ . When using the optimistic quorum collection protocol we developed in our previous work [4], only  $2B + 1$  servers need to be accessed for query in the best case scenario.



**Figure 4. Quorum Read/Write on User Public Key, with  $N=5$ ,  $B=1$**

#### 4.1.2 Contact Address Protocol

Algorithm 3 and algorithm 4 are quorum protocols used to access user contact address. Again, the largest write quorum size and the smallest read quorum size are used for good read performance. The minimum quorum intersection size is  $(N - B) + (2B + 1) = B + 1$ , which guarantees at least one non-compromised server is in the intersection, which will return the latest contact address. Again, the number of servers accessed for query can be reduced to  $B + 1$  servers using our optimistic quorum collection.

**Input:**  $A$ : user signed contact address

```

1 send  $A$  and current time to all  $N$  servers ;
2 while have not got  $(N - B)$  server confirmation do
3   wait for more server responses ;
4 end

```

**Algorithm 3:** Client Address Registration

**Input:**  $I$ : user id

```

1 read  $I$ 's contact address from  $2B + 1$  proxy servers ;
2 return the newest one with proper user signature ;

```

**Algorithm 4:** Query Client Public Key  $K$

The query operations of both user public key and contact address can be done in parallel with other steps of call setup process. After getting the first response of user public key/contact address, the call setup process can proceed without waiting for more responses. While the call is being set up, the client can process more query responses and decide if the one received previously is the correct response. If not, the call setup will be restarted with the correct value. In normal cases, the first response is most often the correct value, so the call setup latency experienced by users will be the same as in the non-replicated system.

## 5 Evaluation

We have experimentally evaluated our non-replicated user signature scheme. A prototype is implemented by extending an open source SIP server SIP Express Router (SER) [12]. We compared SIP, SIPS and our user signature scheme on proxy server throughput in call setup process.

Experiments are done at Emulab [11]. Client/server machines used in the experiments are all PCs with Intel Xeon 3.0 GHz CPU, 1 GB RAM, running Redhat Linux 9, and they are connected by a 100 Mbps switched Ethernet. Server throughput is measured by SIP invitation request served per minute. In our experiments, we used around 100 client machines to saturate a single server machine.

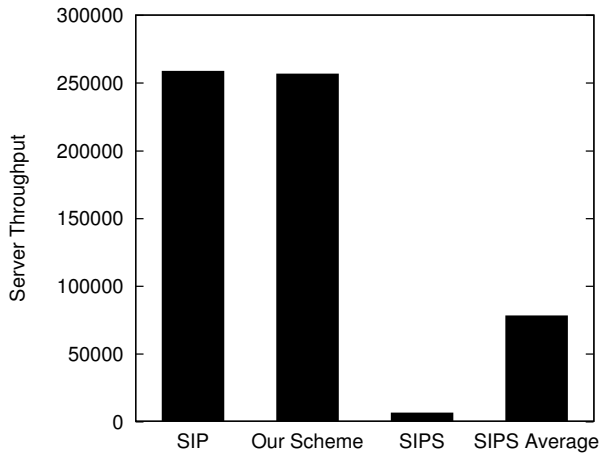


Figure 5. Server Throughput

From left to right, figure 5 shows proxy server throughput of SIP, our user signature scheme, SIPS and SIPS with persistent session or session resumption. As the data shows, our scheme achieves similar proxy server throughput as standard SIP, while providing strong protection on contact address and other fields in SIP messages. Though SIPS also provides end-to-end strong protection, the throughput is only 2% of what our user signature scheme can achieve, when no persistent SSL session or SSL session resumption is involved. When assuming that 50% of SIPS requests can be handled by existing or resumable SSL session (50% of SIPS traffic is generated by frequent callers), the fourth bar shows the averaged SIPS throughput. Even compared with this averaged SIPS case, our user signature scheme stills shows significantly higher throughput. These results are not that surprising, since our user signature scheme eliminates most of scenarios when proxy servers need to be involved in expensive public key operations.

## 6 Summary

This paper reports a lightweight scheme that protects SIP contact addresses through the usage of user signature, which avoids the reliance on a public key infrastructure by chaining trust among SIP entities across domains. Fault/intrusion tolerance is achieved by service replication. In addition, user public key distributed in our scheme can also be used for user authentication and media session protection. We have collected some preliminary experimental results and plan to conduct further investigation on scalability, performance and reliability.

## References

- [1] L. Alvisi, D. Malkhi, E. Pierce, M. Reiter, and R. N. Wright. Dynamic byzantine quorum systems. In *International Conference on Dependable Systems and Networks*, volume 1, 2000.
- [2] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose. DNS security introduction and requirements, RFC 4033, Mar. 2005.
- [3] C. Cachin and A. Samar. Secure distributed DNS. In *Proceedings of Dependable Systems and Networks*, pages 423–432, Jun 2004.
- [4] L. Kong, D. Manohar, A. Subbiah, M. Sun, M. Ahamad, and D. Blough. Agile store: Experience with quorum-based data replication techniques for adaptive byzantine fault tolerance. In *Proceedings of the Int'l Symposium on Reliable Distributed Systems*, 2005.
- [5] L. Lamport. On interprocess communication, part 1: Basic formalism. *Distributed Computing*, 1:77–85, 1986.
- [6] D. Malkhi and M. Reiter. Byzantine quorum systems. *Distributed Computing*, 11(4), 1998.
- [7] J. Rosenberg and H. Schulzrinne. Session initiation protocol (SIP): Locating SIP servers, RFC 3263, June 2002.
- [8] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. SIP: Session initiation protocol, RFC 3261, June 2002.
- [9] F. B. Schneider. Implementing fault-tolerant services using the state machine approach: A tutorial. *ACM Computing Surveys*, 22(4), Dec 1990.
- [10] X. Wang, Y. Huang, Y. Desmedt, and D. Rine. Enabling secure on-line dns dynamic update. In *Proceedings of 16th Annual Computer Security Applications Conference*, 2000.
- [11] B. White, J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. Hibler, C. Barb, and A. Joglekar. An integrated experimental environment for distributed systems and networks. In *Proc. of the Fifth Symposium on Operating Systems Design and Implementation*, pages 255–270, Boston, MA, Dec. 2002. USENIX Association.
- [12] SER - SIP Express Router, <http://developer.berlios.de/projects/ser/>.
- [13] J. Peterson, C. Jennings. Enhancements for Authenticated Identity Management in the Session Initiation Protocol (SIP). Internet draft draft-ietf-sip-identity-06.
- [14] J. Evers., VOIP security prototype gets an airing, CNET News.com, July 28, 2005.

## APPENDIX

Initial INVITE message with both user public key and contact signature and corresponding response:

```

INVITE sip:bob@biloxi.com SIP/2.0
Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bKnashds8
Max-Forwards: 70
To: Bob <sip:bob@biloxi.com>
From: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 314159 INVITE
PK-Exchange: True
Contact: <sip:alice@pc33.atlanta.com>
Content-Type: multipart/related; boundary=boundary42
Content-Length: 568

--boundary42
Content-type: application/octet-stream
Content-transfer-encoding: base64
iD8DBQFD7w09IblKXNifrB8RAtciaJ9ZBbzHavblwyXkELNg1b8DoAcBkQCg7Eq0
CbNjDdvZmJEVum906GcPA8o=
=jbIu

--boundary42
Content-type: application/octet-stream
Content-transfer-encoding: base64
mQGIBEPu/6QRBACIlm+WXu3XCgTk/CNUPLK1fVshcUvCozBY5ThX0rCx2qksAxgo
2yKjvWBLZsgZjUvWJFkpnAvBPf17ROKXJt1JgQup8ORuWD7uqvjFLWp3dkxyXSds
...
CgkQIblKXNifrB9slACgx8m+oVMg0ayixB9N9Vfp7Jf5kUMAoLJsKWRyffdBqbWR
iPtPIunvEdK3
=MCq6

--boundary42
Content-Type: application/sdp
Content-Length: 147

(sdp not shown)
--boundary42-

200 OK Response
SIP/2.0 200 OK
Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bKnashds8
;received=192.0.2.1
To: Bob <sip:bob@biloxi.com>;tag=a6c85cf
From: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 314159 INVITE
Contact: <sip:bob@192.0.2.4>
Content-Type: multipart/related; boundary=boundary42
Content-Length: 568

--boundary42
Content-type: application/octet-stream
Content-transfer-encoding: base64
iD8DBQFD7xcgQNxuTKUnYWIRAuDFAJ93Sy7lsCZK8m0yspeeL0fWfYauNACeN92s
faxYw9gFDCmyQagJJQB0JKc=
=invO

--boundary42
Content-type: application/octet-stream
Content-transfer-encoding: base64
mQGIBEPvE84RBACnH4A3uryK9lmMPfHjeoakYfkDOXhulcU8v7Ait3R22v++RZN
jdhhkrP/Rv0JZ8tvpKRYAn4y5BZNLv8xogNIEo/aLRw9WX+uYTegHKr+BicT8aLV
...
w+kAnlQ61+EwVwI8aaOevMs6DlpmMIYdAJwPGNo0EtmAWNmtOqiRmkGxU79qZQ==

```

=aSo5

--boundary42  
 Content-Type: application/sdp  
 Content-Length: 147

(sdp not shown)  
 --boundary42

### Subsequent INVITE message after user public key exchange and response:

INVITE sip:bob@biloxi.com SIP/2.0  
 Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bKnashds8  
 Max-Forwards: 70  
 To: Bob <sip:bob@biloxi.com>  
 From: Alice <sip:alice@atlanta.com>;tag=1928301774  
 Call-ID: a84b4c76e66710  
 CSeq: 314159 INVITE  
 Contact: <sip:alice@pc33.atlanta.com>  
 Content-Type: multipart/related; boundary=boundary42  
 Content-Length: 354

--boundary42  
 Content-type: application/octet-stream  
 Content-transfer-encoding: base64  
 iD8DBQFD7w09IblKXNifrB8RAtciAJ9ZBbzHavblwyXkELNglb8DoAcBkQCg7Eq0  
 CbNjDdvZmJEVum906GcPA8o=  
 =jbIu

--boundary42  
 Content-Type: application/sdp  
 Content-Length: 147

(sdp not shown)  
 --boundary42

200 OK Response  
 SIP/2.0 200 OK  
 Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bKnashds8  
 ;received=192.0.2.1  
 To: Bob <sip:bob@biloxi.com>;tag=a6c85cf  
 From: Alice <sip:alice@atlanta.com>;tag=1928301774  
 Call-ID: a84b4c76e66710  
 CSeq: 314159 INVITE  
 Contact: <sip:bob@192.0.2.4>  
 Content-Type: multipart/related; boundary=boundary42  
 Content-Length: 346  
 --boundary42  
 Content-type: application/octet-stream  
 Content-transfer-encoding: base64  
 iD8DBQFD7xcgQNXuTKUnYWIRAUdFAJ93Sy7lsCZK8m0yspeeL0fWfYauNACeN92s  
 faxYw9gFDCmyQagJJQB0JKc=  
 =inv0

--boundary42  
 Content-Type: application/sdp  
 Content-Length: 147

(sdp not shown)  
 --boundary42