Private Anonymous Messaging With Friends

Ruchith Fernando , Cristina Nita-Rotaru
Department of Computer Science, Purdue University
West Lafayette, IN
{rfernand, crisn}@cs.purdue.edu

***Abstract*—The abstract goes here.**

I. Introduction

We identify a set of requirements in broadcasting messages among trusted peers using a peer to peer setup. Then we propose a scheme for contacts of a peer to obtain broadcast updates of the peer using a pull mechanism where the contacts' identities are maintained anonymous. We propose a modification to the hierarchical identity based encryption scheme proposed by Boneh et. al [1] where a peer can request messages of a common peer from another peer while remaining anonymous.

Following section introduces the setup of the network of peers and how they are connected and the requirements that we try to satisfy with our scheme. This is followed by the preliminary notions and then we describe our solution that meets the stated requirements. Implementation of the proposed cryptographic primitives is presented in section V followed by future directions of this work and conclusion.

II. Problem

A. Background

A peer in this system is a user who has a set of other peers registered with it as contacts. This peer registration is bi-directional. In other words when peer A becomes a contact of peer B, peer B becomes a contact of peer A.

A peer intends to send messages to all its contacts. All of these messages are to be delivered to the peer's contacts at that point of time. This is similar to the notion of microblogging. (Example: Twitter[2]). Such a message is identified as an $update$.

We demote the a peer generating an $update$ as P and its contacts as the set $C = \{C_{P_i}\}$ where $i \in \{1, ..., n\}$ where n is the number of contacts of P.

B. Requirements

We identify the following requirements in distributing messages in the above system.

- A peer P should be able to simply send its $update$ M_P only to those contacts who are available online at the point of time it sends the update using direct connections to those peers. We denote the set of online contacts as $C^+ \subseteq C$ where $|C^+| \geq 1$.
- Those other contacts of P who were offline at when P sent M_P should be able to obtain M_P when they are available online. We denote these contacts as $C^- \subset C$.
- Any $C_{P_i} \in C^-$ will be able to publish a query requesting an update of P. This is called an update request and is denoted by Q_P.
- Any $C_{P_i} \in C^+$ will be able to publish a response to a Q_P. This response is denoted by S_P and an eavesdropper with polynomially bounded resources should not be able to compute the original M_P using S_P.
- The contact who provides S_P should not be able to learn who generated Q_P.
- The contact who generates Q_P and receives the corresponding S_P should not be able to learn who generated S_P.
- When the composition of C changes to new set of peers C', P should be able to update private configuration of the members of C' with the issue of a public message.
- After such an update those peers in the set $C - C'$ should not be able to obtain an $update$ of P.

The scheme we propose in section IV addresses all these requirements.

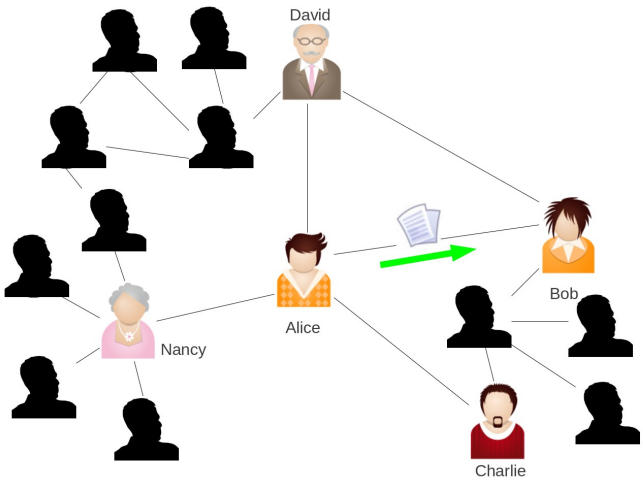

Fig. 1. A user (Alice) and her contacts (Bob, Charlie, David and Nancy)

For example consider figure 1 above. In this situation Alice is P. Alice has four contacts: Bob, Charlie, David and Nancy, out of which only Bob gets the $update$ M_P. Therefore: $C = \{Bob, Charlie, David, Nancy\}$, $C^+ = \{Bob\}$ and $C^- = \{Charlie, David, Nancy\}$.

Note that a peer only trusts and has knowledge of its immediate contacts and is not aware of connections between those

peers and their contacts. There are practical implementations of the notion of friend-only networks such as Freenet/Darknet [3] and ...

III. Priliminary Notions

In this section we introduce the necessary background information that our work is based on.

A. Hierarchical Identity Based Encryption (HIBE)

Identity based encryption first proposed by Shamir[4] is a public key encryption scheme where the identity of an entity can be used as the public key. The first complete solution for this was presented by Boneh and Franklin [5]. Any party who intends to send a message to another will simply use a set of public parameters of a trusted authority along with the identity of the recipient will encrypt using this scheme. The recipient of the cipher text will be able to obtain the corresponding private key from the third party (who executes private key generation algorithm for the given identity after authenticating the requester) and decrypt the cipher text to obtain the plain text.

This idea of identity based encryption was extended to a hierarchy of identities [6], [1], where at each level the private key is used as the input to the key generation algorithm along with the global parameters defined by the root. The HIBE system is defined in [1] as follows (which we modify in deriving out scheme):

Let $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_1$ be a bilinear map where $\mathbb{G}$ is a group of prime order p. An identity is defined as $ID = (I_1, ..., I_k) \in (\mathbb{Z}^*_p)^k$ where k is the depth of the hierarchy that the ID belongs to.

There are four algorithms: $Setup$, $KeyGen$, $Encrypt$ and $Decrypt$. l is the maxium depth of the hierarchy allowed.

- $Setup(l)$, generates the public parameters and the master key as follows:
  - Select a generator $g \in \mathbb{G}$ and a random $\alpha \in \mathbb{Z}_p$
  - Set $g_1 = g^\alpha$
  - Pick random $g_2, g_3, h_1, ..., h_l \in \mathbb{G}$
  - $params = (g, g_1, g_2, g_3, h_1, ..., h_l)$
  - $master - key = {g_2}^\alpha$

- $KeyGen(d_{ID_{k-1}}, ID)$, generates the private key of the given k^{th} level ID using a $k-1$ level private key ($k \leq l$). First suppose the $k-1$ level private key was generated using the master key :
  - Select a random $r \in \mathbb{Z}_p$
  - Output $d_{ID_{k-1}}$:
    $({g_2}^\alpha \cdot (h_1{}^{I_1} \cdots {h_{k-1}}^{I_{k-1}} \cdot g_3)^r, g^r, {h_k}^r, ..., {h_l}^r) = (a_0, a_1, b_k, ..., b_l)$

  Now the k^{th} level private key:
  - Select a random $t \in \mathbb{Z}_p$
  - Output d_{ID_k} :
    $(a_0 \cdot {b_k}^{I_k} \cdot (h_1{}^{I_1} \cdots {h_k}^{I_k} \cdot g_3)^t, a_1 \cdot g^t, {h_{k+1}}^t, ..., {h_l}^t)$

- $Encrypt(params, ID, M)$, encrypts a message $M \in \mathbb{G}$ using the public key $ID = (I_1, ..., I_k)$:
  - Select a random $s \in \mathbb{Z}_p$
  - Output CT :
    $(e(g_1, g_2)^s \cdot M, g^s, (h_1{}^{I_1} \cdots {h_k}^{I_k} \cdot g_3)^s)$
    $= (A, B, C)$

- $Decrypt(d_{ID}, CT)$, dercypts a given cipher text of the above form (A, B, C) using the given private key of the form $(a_0, a_1, b_k, ..., b_l)$.
  $(A \cdot e(a_1, C))/(e(B, a_0)) = M$

Next section describes how this scheme is used to meet the requirements identified in section II.

IV. Proposed Solution

Here we present the scheme that addresses the requirements identified in section II.

A. Peer setup

A peer P will have a two level HIBE system parameters. This is by calling $setup(2)$. This will generate generates the public parameters and the master key of the peer as follows:

- Select a generator $g \in \mathbb{G}$ and a random $\alpha \in \mathbb{Z}_p$
- Set $g_1 = g^\alpha$
- Pick random $g_2, g_3, h_1, h_2 \in \mathbb{G}$
- $params = (g, g_1, g_2, g_3, h_1, h_2)$
- $master - key = {g_2}^\alpha$

B. Registering a contact

The main idea is to setup a two level ($l = 2$) HIBE system at each peer. When a peer P registers a C_{P_i} it will create a new random first level identifier $I_{r_i} \in \mathbb{Z}_p$ and corresponding private key ($d_{I_{r_i}}$). The private key and the identifier will be communicated to C_{P_i} using a private channel. $d_{I_{r_i}}$ is of the form $({g_2}^\alpha \cdot (h_1{}^{I_{r_i}} \cdot g_3)^r, g^r, {h_2}^r, {h_3}^r)$, where $r \in \mathbb{G}$ is random.

- C_{P_i} keeps both I_{r_1} and $d_{I_{r_1}}$ private along with the public parameters of P
- P stores the tuple $< I_{r_i}, r >$, [1]

C. A contact requesting an update

When P sends an $update$ message it may send the update directly to available contacts by encrypting the message using their corresponding identifiers. The interesting case is when a contact $C_{P_{req}}$ needs to obtain the latest $update$ of P and P is no longer available online. In such a situation, as highlighted by in the requirements, $C_{P_{req}}$ will be able to generate a request for P's update. This is generated as follows:

Suppose the identifier assigned to $C_{P_{req}}$ by P is I_{r_1}

- Select a random $I_{r_2} \in \mathbb{Z}_p$
- Set $ID_{req} = h_1{}^{I_{r_1}} \cdot h_2{}^{I_{r_2}}$
- Update Request to be published = $< P, ID_{req} >$, here P is an identifier string of P known to all P's contacts.

[1] This is used to update contact parameters in the case of a re-key.

$C_{P_{req}}$ publishes $< P, ID_{req} >$ and any of P's other contacts will be able to respond to this request. This request information can simply be made publicly available using a common medium. The steps in creating the response is described next.

D. Encryption and update response

When a contact of P observes the tuple $< P, ID_{req} >$ and decides to serve this request it will first encrypt the latest $update$ message M_P from P using the following modified encryption function ($Encrypt'$) and P's public parameters $params_P$.

$Encrypt'(params_P, ID_{req}, M_P)$:

- Select a random $s \in \mathbb{Z}_p$
- $CT_{resp} = (e(g_1, g_2)^s \cdot M, g^s, (ID_{req} \cdot g_3)^s) = (A, B, C)$

The contact can now publish the tuple
$< P, ID_{req}, CT_{resp} >$.

E. Re-key

The set of contacts at a peer C can change in two ways:

- When a new contact joins
- when an existing contact is removed

When a new contact ($C_{P'}$) joins the peer P simply can carryout new contact registration without and this doesn't require any changes to the parameters. The new contact will be able to request updates of the peer from its other contacts in the set ($C - C_{P'}$).

However when P needs to remove a contact $C_{P'}$ from the list of contacts, it has to update its parametes. We present an approach where we generate public information that the set $C - C_{P'}$ will be able to use to configure themselves.

In peer setup, the generated HIBE configuration if of the form $params = (g, g_1, g_2, g_3, h_1, h_2)$ and $master - key = {g_2}^{\alpha}$ where $g_1 = g^{\alpha}$ and $\alpha \in \mathbb{Z}_p$ is random. In the case of re-key a peer :

- Geneates a new random $\alpha' \in \mathbb{Z}_p$
- Sets $master - key = {g_2}^{\alpha'}$
- Set $g_1 = g^{\alpha'}$

With this change P will have to update the private keys of the contacts. Note that in contact registration process P stored the tuple $< I_{r_i}, r >$ for each contact C_{P_i}.

To update contacts:

First generate a random $u \in \mathbb{Z}_p$

Initialize a list $< id'_i, A_i >$ and for each contact $C_{P_i} \in C$:

- generate the first component of the private keys of the contacts as ${g_2}^{\alpha'} \cdot ({h_1}^{I_{r_i}} \cdot g_3)^{r_i} = A$. This r value is from $< I_{r_i}, r >$.
- Add $< {I_{r_i}}^u, A >$ to the $< id'_i, A_i >$ list.

Finally the complete re-key information to be published is : $< g_1, u, < id'_i, A_i >>$. Note that id'_i is the identifier of C_{P_i} blinded using u.

V. Evaluation

Anonymous ID
Cipher text
Random re-key
Note that in the hierarchical identity based encyrption scheme [1] the plain identity values of an entity are used in generating the cipher text. During the encryption the following value is derived using the identity values of the target.

VI. Implementation

The proposed scheme was implemented in Java as a library using Java Pairing Based Cryptography [7] library. The demo application developed uses this library in to demonstrate the features of this library. This work available under LGPL at anon-encrypt project hosted in google code [8].

A. Library

Mapping plain text to elements : Encoder and Decoder

Block cipher implementation

Talk about test cases also
Talk about output formats

B. Demo application

Command line application
Mention Zenity[9]

VII. Future Work

TODO Add comparison with broadcast encryption somewhere

A. Size of re-key information

Currently we generate a minimum amount of information that is required to re-key a peer and its contacts. But in this scheme the re-key informaiton is of order n where n is the number of contacts of the peer. It would be interesting to evaluate the possibility of reducing the size of this public information while maintaining the same properties.

B. Incentives to forward messages

This sheme relies on the fact that the contacts belonging to the set C^+ (those who holds the latest M_P) will respond to an $update$ request Q_P with correct a response S_P. It will be useful to evaluate the possibility of coming up with an incentive scheme for contacts in C^+ to respond to Q_Ps. [TODO: describe properties of such a scheme]

C. Security roof

We plan to prove that an adversary with polynomially bounded resources will not be able to ...??

D. Implementation of message routing

The current implementation only covers the cryptographic primitives. It will be interesting to use these with a peer to peer network where the peers are connected only to their private contacts and evaluate the performance. It might be possible to be implemented as a plugin to Freenet [10].

VIII. Conclusion

The conclusion goes here.

References

[1] Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. In Ronald Cramer, editor, *Proceedings of Eurocrypt 2005*, LNCS. Springer, 2005.

[2] Twitter. http://twitter.com.

[3] Ian Clarke, Oskar Sandberg, Matthew Toseland, and Vilhelm Verendel. Private communication through a network of trusted connections: The dark freenet.

[4] Adi Shamir. Identity-based cryptosystems and signature schemes. In *Proceedings of CRYPTO 84 on Advances in cryptology*, pages 47–53, New York, NY, USA, 1985. Springer-Verlag New York, Inc.

[5] Dan Boneh and Matthew Franklin. Identity-based encryption from the weil pairing. *SIAM J. Comput.*, 32:586–615, March 2003.

[6] Jeremy Horwitz and Ben Lynn. Toward hierarchical identity-based encryption, 2002.

[7] The Java Pairing Based Cryptography Library (jPBC). http://gas.dia.unisa.it/projects/jpbc/.

[8] Project anon-encrypt at Google code. http://code.google.com/p/anon-encrypt/.

[9] Zenity. http://freshmeat.net/projects/zenity.

[10] Freenet. http://freenetproject.org/.