2PPS – Anonymous Pub/Sub based on PIR

S. A. Gaballah, C. Coijanovic, T. Grube, T. Strufe, M. Mühlhäuser

Introduction

Multicast communication in open groups:
- Applications: Microblogging, newsfeeds, open chat rooms, and forums.
- Properties:
  - Users communicate based on mutual interests (e.g., topics or hashtags).
  - Groups can be joined by anyone at any time.
  - Usually, there is no personal connection or trust between the users.
- Communication Model: publish/subscribe messaging

- Privacy issue: The broker has full knowledge of users and their interests.

2PPS aims to solve this issue by using Private Information Retrieval (PIR) for both publishing and receiving messages.

Threat Model and Security Goals

Threat Model:
- We assume a Dolev-Yao adversary.
- The adversary can corrupt all but one server, and an arbitrary number of clients.
- We require that no malicious client can successfully launch DoS attacks.
- Also, we demand that the malicious servers will not deny service availability since PIR requires participation from all servers.

Security Goals:
- Sender Unobservability: The adversary cannot identify the actual sending activity of an honest client.
- Receiver Unobservability: The adversary cannot identify the actual receiving activity of an honest client.

Related Work

Existing anonymous group communication systems do not fully meet our requirements:
- [1]–[3] require trusted group members.
- [2] and [3] don’t provide both strong sender and receiver anonymity.
- [4] is vulnerable to traffic analysis attacks.

References


2PPS Approach

2PPS consists of a small set of servers and a large number of clients. The main phases of the system are as follows:

- **Phase 1: Anonymous Publishing** (using DPF [6])
  - The client creates a private write request by generating a set of DPF-shares: \( (d_0, d_1, \ldots, d_m) \leftarrow \text{Gen}(l/m, i) \), where \( l \) is the random index that the message \( m \) should be written into and \( t_m \) is the topic of the message \( m \).
  - One DPF-share is sent to each of the servers.
  - Each server expands the received DPF-share to a vector with the same size as \( db_r \), which is added to its \( db_m \) state.

- **Phase 2: Managing Published Messages**
  - Servers combine their \( db_m \) states to reveal the published messages.
  - Each message in \( db_m \) is moved to its corresponding topic-block in \( db_r \).

- **Phase 3: Anonymous Subscribing** (using IT-PIR [5])
  1. **Private Subscription Registration**
     - The client creates a subscription request \( \text{req} = \text{Enc}_{\text{pki}}(s_i, q_i) \) for each server \( i \), where \( s_i \) is a shared secret and \( q_i \) is a PIR query.
     - The client chooses one of the servers to be its primary server \( P \), and sends the subscription requests to it.
     - \( P \) forwards these requests to the corresponding servers where they are stored.
  2. **Private Messages Retrieval**
     - Each server computes a response for each stored subscription request.
     - The server obfuscates its response by XORing it with \( s_i \) and forwards the result to \( P \).
     - Responses are XORed by \( P \), and the result is sent to the client.
     - The client XORs the received reply with all shared secrets to reveal the published messages.

* The shared secret \( s_i \) is locally updated each round at both client and server using a cryptographic hash function.

Future Work

We plan to extend our work in the following directions:

**Topic Management:**
- There is a need for an efficient way to add new topics and remove inactive ones.

**Topic Blocks:**
- PIR requires a fixed block size, but the number of messages per topic can vary.
- One possible solution is having multiple smaller blocks for the popular topics.

**Formal Security Analysis:**
- To prove that 2PPS actually reaches the stated privacy goals, we intend to use game based proofs [7].

**Performance Evaluation:**
- We are implementing a prototype of 2PPS to measure the computational and communication costs of our system on both client and server sides, and results will be compared to the state of the art (e.g., [2], [5], and [6]).