OblivP2P: An Oblivious Peer-to-Peer Content Sharing System

Yaoqi Jia, Tarik Moataz, Shruti Tople and Prateek Saxena

National University of Singapore
Traffic Analysis in P2P Systems

• P2P content sharing systems
  – 150 million users/month
  – 3.35% of all world bandwidth

• Long term global traffic analysis
  – E.g., ISP’s, Global BitTorrent Monitor, Bitstalker
What can an Adversary do?

- Leakage Channels
  - Plaintext data
    - Secure channel
  - Assume existing defenses
  - Time
    - Fixed Interval

- Access Patterns

Linkability
Problem

Current Solutions

– Anonymous Systems e.g., Mix Networks, Tor

Hide Online Identity

Unlinkability

✓ Long term
✓ Global

Is anonymizing enough?
Contributions

OblivP2P Protocol
- Guarantee unlinkability
- Obliviousness in P2P systems

Implementation
- Link: https://github.com/jiayaogijia/OblivP2P-Code

Evaluation
- No Centralized Bottleneck
- Linear Scalability with peers
Problem
Insufficiency of Existing Solutions

Intersection, Hitting Set [AK’03] or Statistical Disclosure Attacks [KP’04]
Main Insight: Oblivious Access Pattern

• Oblivious RAM
  – Hide access patterns between CPU and memory
  – Data is shuffled in the memory periodically

• Applied to:
  – Cloud Storage [SS’13A], [SS’13B],[LO’13]
  – Filesystem [WST’12]

• Can we directly apply ORAM to P2P systems?
Problem Definition

Trusted Tracker
ORAM Background

- Tree-Based ORAM (Path ORAM)
  - *Read*
    - Fetches a path from the tree containing the block
    - Stores the path in the local storage (stash)
  - *Write*
    - Selects a random path in the tree
    - Shuffles the blocks in the stash and the path
Mapping ORAM to P2P

Trusted Client
- Position Map, Stash
  - Fetch path
  - Decrypt

Trusted Tracker
- Position Map, Stash
  - Fetch a path
  - Decrypt

Untrusted Server
- Peer-to-Peer Network

OblivP2P-0 Protocol
OblivP2P-0: Tracker as bottleneck

- Tracker fetches $O(\log N)$ blocks per access

Need a Distributed Oblivious P2P Protocol
OblivP2P-1 Protocol
Naïve approach: Removing Bottleneck

Performance

Security

Trusted Tracker
Position Map, Stash

Send
<path, position, key>

Request

Initiator

Decrypt

Fetch a path

Peer-to-Peer Network
Challenges

• ORAM writes
  – Recently accessed block at the root
  – Less frequently accessed block at the leaves

• “Block History”
  – Shared resources

• Security flaw in P2P systems
  – Multiple users access the same resource
New Primitive: Oblivious Selection

Selects a block *without*: 

- Block Position
- Cryptographic Key
- No Centralized Bottleneck
• Step 1: PIR over ORAM
  – Obliviously select a block from a path

Diagram:
- Trusted Tracker
- Initiator
- Request
- Compute an Encrypted Share using PIR
- Block Position
- Cryptographic Key
- No Centralized Bottleneck
• Step 2: Seed-Homomorphic PRG
  – Decrypt shares without giving away the key

Trusted Tracker

Send key share

Compute a Decrypted Share using SH-PRG

Initiator

Block Position

Cryptographic Key

No Centralized Bottleneck
Security
OblivP2P is an Oblivious P2P Protocol

Any two equal length access sequences by two peers are indistinguishable for any p.p.t. “honest-but-curious” adversary

1. **Number of dishonest peers is in** $O(N^\varepsilon)$, where $\varepsilon < 1$

2. **Theorem:** If $\forall N > 1$, and $\forall \varepsilon < 1$, $\exists m > 1$ such that $2^{\log N \cdot m \cdot (1-\varepsilon)} \in \text{negl}(\lambda)$ then OBLIVP2P-1 is an oblivious P2P protocol
Evaluation
Experimental Setup

• 15 DeterLab servers – \((2^{14})16000\) peers

• Each server shares a bandwidth of 128 MBps

• Block size of 512 KB similar to BitTorrent
No Centralized Bottleneck

~ 128 MB /req

~ 1 MB /req
Linear Scalability with Peers

• Larger networks can scale up performance
  – 3.59 MB/s is due to our limited test infrastructure
• Bottleneck remaining is purely computational

3.59MB/sec
• Propose hiding data access patterns in P2P systems

• OblivP2P - First work to repurpose ORAM in Peer-to-Peer systems

• OblivP2P is linearly scalable and highly parallelizable with the peers in the network
Thanks!

Email: shruti90@comp.nus.edu.sg

Link: https://github.com/jiayaqijja/OblivP2P-Code