Protecting Anonymity in Dynamic Peer-to-Peer Networks

Krishna Puttaswamy, Alessandra Sala, Christo Wilson, Ben Y. Zhao
Computer Science Department
University of California, Santa Barbara

Introduction

Variety of internet applications
- Browsing, VoIP, Streaming, etc.

Current anonymity systems
- Lagging behind these applications
- TOR, widely popular
  - But geared towards low-overhead apps.
  - Little support for contemporary apps.

Natural alternative
- Peer-to-Peer model for anonymity

P2P Anonymous Networks

- Many proposals: Tarzan, Salsa, etc.
- Primary Challenge: Churn
  - Rebuild connection for every relay churn
- Disadvantages
  - Performance: application specific
  - Anonymity: degrades anonymity quickly
- This talk: how to tolerate churn, and achieve high anonymity?
  - Our contribution: system called Bluemoon

1. Background
Current Systems

Effect of Churn

Predecessor Attack

- Predecessor and successor visible
- Multiple malicious nodes
  - Timing attack to detect if on same path
  - Suspect with higher probability
- Malicious nodes share information

Visualizing Predecessor Attack
Impact of Predecessor Attack

- Can hurt all relay-based systems
- Inevitable in path-based systems
  - Eventually the end-points will be seen by attackers
- Apps. continue to have longer sessions
- Very simple to perform
- Can be performed with low resources
  - Practically demonstrated recently
- No practical solution so far

Our Goal: High Anonymity

- Intuition: build persistent paths
  - Reduces the impact of churn
  - Avoids path rebuilds
  - Improves both anonymity and performance
- Path persistence via Hooks
  - Hook: A group of nodes that share a key
  - Any member can decrypt and route
- Attacker Model
  - A fraction of nodes are malicious and collude
  - Perform various passive attacks, timing attacks

2. Communication via Hooks

Overview

- Hook: a group of fixed number of nodes
- Any member can do the routing
  - Introduced in Cashmere [NSDI 05]
  - Constructs groups probabilistically
- Bluemoon: a group has g nodes always
Hook Implementation

- Need fixed number of nodes in a group
- Use DHTs to implement Hooks
  - Store Hooks reliably in the network
  - Configure replication factor to g
- A Hook stored on a random DHT Id
  - Symmetric key: decryption
  - Next-hop Id: forwarding

Anonymous Paths with Hooks

- Happens in two phases
- Phase 1: Path setup
  - Create temp. symmetric keys and install hooks
  - Random walk based hook setup for anonymity
  - Receiver in the receiver Hook
    - Embed receiver in any of the Hooks

Onion Construction at Source

- Bluemoon uses symmetric keys

Data Transfer

- Forward the Onion to the first Hook
- Session key via public key encryption
- Random bits after the receiver Hook
3. Analytical Results

Key Analytical Results

- Source Anonymity
  - TOR/Crowds: same even for longer paths
  - Bluemoon: increases with longer paths
- Receiver Anonymity
  - Increases with Hook size
- Number of rounds to break unlinkability
  - Increases with both Hook size and path length
  - With very conservative parameters
  - At least two orders of magnitude better than TOR
- Detailed analytical results in the paper

4. Experimental Results

Setup

- Simulation
  - Measure the degradation of anonymity
  - Compare against other systems (TOR)
  - Used churn model from Gnutella study
- Implementation of Bluemoon in C++
  - Ran our prototype on PlanetLab
    - Anonymous browsing application on Bluemoon
    - Throughput measurements
Number of Path Rebuilds in TOR

Resilience to Predecessor Attack

Throughput on PlanetLab

Conclusions

- P2P anonymous communication system
  - Provides high anonymity
  - Hooks withstand churn, build persistent paths
- Analysis Results
  - Predecessor attack significantly slowed down
  - Detailed analysis in the paper
- Experiments
  - Running prototype on PL
  - 2Mbps throughput

No path rebuilds in Bluemoon for all path lengths!

Probability of Success

Number of Rounds (20% Malice, L = 16)

UCSB 24

UCSB 24
Thank you!