Anonymous Communications and Reputation

The Free Haven Project

Roger Dingledine, Reputation Technologies

Talk overview

- Introduction to MIXes/remailers
- The Problem
- Approach one: witnesses and receipts
- Approach two: cascades

MIX network features

- Use multiple nodes to increase trust
- Messages change appearance after decryption
- Each MIX batches and reorders received messages
- Store and forward (slow) to maintain anonymity

The Problem

- The current remailer infrastructure is
  - unreliable (dropped/repeated messages)
  - inefficient (high/unpredictable latency)

- Unreliability decreases anonymity

Ways of Improving Reliability

- Build protocols with provable reliability guarantees

- Add reputation to “improve” reliability

- Provide economic incentives for reliability

- Distinction between reliability and robustness

Related Work

- MIXes (Chaum)
- Robust MIX-nets (Flash Mix, Universally Verifiable MIX)
- Deployed Remailer Systems (cypherpunks, Mixmaster)
- Remailer statistics (Levien's statistics, Jack B Nymble 2)
- Network routing protocols (OSPF, etc)

First approach

- "A Reputation System to Increase MIX-net Reliability"

- Presented at Info Hiding Workshop, Pittsburgh (April 2001)

- Joint work with Michael J Freedman (MIT), David Hopwood (Independent), David Molnar (Harvard)

Need to Verify Failures

- Verifying successes is not useful: spoofing is easy
- Failures represent MIX unreliability
- Forcing failures *is* unreliability

Ways To Verify Failures

- Publish all intermediate messages (public ledger)

- Web MIXes

- Witnesses and Receipts

Witnesses and Receipts to Verify Failures

- N_{i+1} gives N_i a receipt for each accepted message
- Each message has a deadline after which it has "failed"
- If N_i fails to deliver, he asks witnesses to try
- Witness returns receipt if success, else a failure statement
- Thus senders can check receipts and claim failures

Good MIXes demonstrate honesty

- Honest N_i delivers to N_{i+1} or to witnesses
- ... and receives either a receipt or a set of witness statements
- If sender challenges, he can provide receipt or statements (Majority of statements is credible)

Bad MIXes Are Caught

- Attacks: don't accept, or silently drop
- Witnesses will catch MIXes that don't accept
- MIXes that silently drop can't show receipts/statements
- Sender claims failure to witnesses, who check

Reputation System Requirements

- Automated: sender software automatically uses reputations
- Verifiable: reputations can be verified; claims can be checked
- Dynamic: e.g. reflect recent trends in behavior
- Must maintain anonymity provided by MIX-net

Our Reputation System

- Witnesses tally failure claims from MIXes and senders

- Witnesses send test messages to get verified successes

- Witnesses publish scores; sender software automatically chooses paths

- Senders throw out MIXes w/out some threshold of successes, then weight remaining MIXes by number of failures

Traffic Analysis

- Messages to witnesses unencrypted for public verification
  (A small adversary can watch more traffic)

- Higher reputation $\Rightarrow$ more traffic to analyze

- Adversary can sabotage other nodes to get more traffic

- But greater reliability $\Rightarrow$ more users $\Rightarrow$ stronger anonymity

Second approach

- "Reliable MIX Cascade Networks through Reputation"
- Submitted to Financial Cryptography (March 2002)
- Joint work with Paul Syverson (Naval Research Lab)

Still Unsolved from First Approach

- Global witnesses are trust and communication *bottlenecks*

- Owning high reputation nodes means you own more paths?

Adversary Goals

- Break Anonymity:
  - Discover linkability between sender and receiver
  - Identify the sender or receiver of a given message
  - Trace a sender forward (or a receiver backward) to any messages.
- Break Reliability: deny service to users

Threat Model — Adversary can:

- Passively read *all traffic*
- Insert, modify, or delay some messages
- Compromise some fraction of the MIXes

What's a MIX cascade?

- Fixed path through the MIX network
- Longer cascades $\Rightarrow$ lower chance all bad nodes $\Rightarrow$ more anonymity
- Longer cascades $\Rightarrow$ lower chance all good nodes $\Rightarrow$ less reliability
- Cascades provide more defense against *intersection attack*.

Design Overview

- Cascades rearrange periodically
- Nodes send test messages to monitor their cascades
- Senders can demonstrate decryptions to show failure
- A node fails its own cascade if it detects misbehavior
- $+1$ reputation if you succeed, -1 if you fail.

Communal Randomness

- All nodes commit, then all reveal
- But nodes can influence communal value by not revealing?
- “Temporarily secret commitment”
  can break with computation
- Xor might not work (maybe malleable). So concatenate and hash.

Heuristics for picking cascades

- Increase cost of breaking anonymity (pick randomly)
- Increase cost of breaking reliability (pick from high-reputation nodes)

Creeping Death

- Fail cascade if more damage to good nodes than bad
- Adversary can get to any point in reputation spectrum!
- Works even with few nodes (10-20%)

Need web of trust — Advogato

- No central or monitoring authority
- Number of bad nodes certified is based on number of confused nodes (good nodes that might certify bad nodes)
- Alice certifies by trustworthiness, not expected performance

So how do we choose cascades?

- Pick a reasonable safety factor S (eg 1 in 10^6 paths bad)
- Choose first cascade randomly from large enough pool of high-reputation nodes
- Replace chosen nodes to maintain pool size
- When pool contains all remaining nodes, just order remaining cascades randomly

How do we decide this pool size?

- Assume a bound on P, the total fraction of adversary-owned nodes

- L is length of cascade, C is number of cascades in chain, LC is path length

- So we can calculate range R from this. Adjust parameters to taste.

Detecting Misbehavior

- Entry point: Incoming messages rejected?
- Inside cascade: Messages replaced with dummy messages?
- Exit point: Messages not delivered?

Detecting Misbehavior at Entry Point

- Alice can send into any node. They all deliver to the head.
- Thus nodes can insert indistinguishable test messages
- Alice gets a receipt (if not, she tries elsewhere)
- Head publishes batch snapshot (hashes of messages)
- If message not in snapshot, receipt proves misbehavior

Detecting Misbehavior Inside Cascade

- The head can publish message in batch snapshot, but silently drop it!

- Alice might become suspicious and send a test message

- If test message is dropped, Alice can demonstrate message decryption to any node

- An honest node will check the batch snapshot and agree

Detecting Misbehavior at Exit Point

- Tail bounces traffic to all nodes. All nodes deliver.
- If test message doesn't arrive, somebody failed
- Optimize: if tail collects a delivery receipt, no broadcast

Test messages

- Nodes reuse sender addresses in test messages
- Similarly, they address test messages to previous recipients
- Reusing addresses helps protect against time-based intersection attack
- Plaintext is distinguishable —— sender must encrypt for reliability

Quality of Service, Resource Management

- Nodes send hourly heartbeat or failure message to Reputation Servers

- Users compare QoS and reputation from each cascade $\Rightarrow$ load balancing

- Reputation Servers are redundant and static — easier to manage

- Users must download *all* heartbeats, or via MIX-net or PIR for anonymity

Future Directions

- Proof of bandwidth (like proof of work) to reduce reliance on web of trust

- Convincing origins and destinations for dummy traffic; better bandwidth use

- More research on creeping death?

- Reliability metric and model (same with Efficiency)