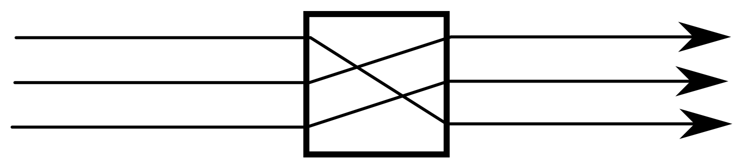
# Anonymity

Christian Grothoff
christian@grothoff.org
http://grothoff.org/christian/

"It's a series of tubes." –Ted Stevens

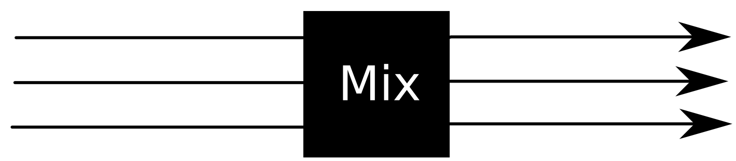
# Review: Mixing

David Chaum's mix (1981) and cascades of mixes are the traditional basis for destroying linkability:



# Review: Mixing

David Chaum's mix (1981) and cascades of mixes are the traditional basis for destroying linkability:



# Agenda

- Definitions and Metrics
- Techniques, Research Proposals and Systems
  - Dining Cryptographers, Mixes
  - **Mixminion**
  - PipeNet, Busses
  - Mute, Ants, StealthNet

# Mixminion

G. Danezis, R. Dingledine, D. Hopwood and N. Mathewson describe *Mixminion: Design of a Type III Anonymous Remailer*:

- based on mixmailers (only application is E-mail)
- possibility to reply
- directory servers to evaluate participating remailers (reputation system)
- exit policies

# Mixminion: key idea

The key idea behind the replies is splitting the path into two legs:

- the first half is chosen by the responder to hide the responder identity
- the second half was communicated by the receiver to hide the receiver identity
- a crossover-node in the middle is used to switch the headers specifying the path

# Mixminion: replay?

Replay attack were already an issue in previous mixnet implementations.

- Mixes are vulnerable to replay attacks
- Mixminion: servers keep hash of previously processed messages until the server key is rotated

$\Rightarrow$ Bounded amount of state in the server, no possibility for replay attack due to key rotation

# Mixminion: Directory Servers

- Inform users about servers
- Probe servers for reliability
- Allow a partitioning attack unless the user always queries all directory servers for everything

# Mixminion: Nymservers

- Nymservers keep list of use-once reply blocks for a user
- Vulnerable to DoS attacks (deplete reply blocks)
- Nymservers could also store mail (use one reply block for many messages).

# Mixminion: obvious problems

- no benefits for running a mixmailer for the operator
- quite a bit of public key cryptography
- trustworthiness of directory servers questionable
- servers must keep significant (but bounded) amount of state
- limited to E-mail

# Mixminion: more problems

- exit nodes are fair game for legal actions
- no accounting to defend against abuse
- statistical correlation of entities communicating over time possible (observe participation)
- vulnerable to DoS attacks

$\Rightarrow$ bridging between an anonymous network and a traditional protocol is difficult

# Reputation

R. Dingledine and P. Syverson wrote about *Reliable MIX Cascade Networks through Reputation*:

- traditional approach uses external trusted witnesses that probe the mix
- this design allows a mix-cascade to monitor itself

# Key idea

- nodes send test-messages to monitor their own cascade
- nodes announce the failure of their own cascade, damaging the reputation of all nodes in the cascade

# Reputation: problems

- Reputation of the reporters
- does not detect failure instantly (loss)
- adversary could create fresh identities

# Zero Knowledge Proofs

W. Ogata, K. Kurosawa, K. Sako and K. Takatani introduced the concept of a *Fault Tolerant Anonymous Channel*:

- nodes can prove that they function correctly without exposing secret information
- concrete protocol is applicable to MIX networks

# ZKP: Sender

Each sender $P_i$ computes $B(m_i, R_i)$ where $m_i$ is his message, $R_i$ is a random polynominal $R(x)$ of degree $k - 1 := \lfloor \frac{n-1}{2} \rfloor$ such that $R(0) = m$ and

$$B(m, R) := [E_1(R(1), x_1), \ldots, E_n(R(n), x_n)] \qquad (1)$$

where $x_i$ are random numbers and $E_i$ is a homomorphic cipher using the public key of mix $i$.

A ZKIP is used to show correctness of the sender's calculations.

# ZKP: Center

Each mix chooses a random permuation $\pi$ and publicizes a reencryption of each $B(m_i, R_i)$:

$$[B(m_{\pi(1)}, R_{\pi(1)} + U_{\pi 1}), \ldots, B(m_{\pi(l)}, R_{\pi(l)} + U_{\pi l})] \quad (2)$$

where $U$ is a random polynominal of degree $k-1$ such that $U(0) = 0$.

A ZKIP is used to show correctness of the calculation.

# ZKP: Decryption

The last mix publicizes:

$$[B(m_{\phi(1)}, \overline{R}_{\phi(1)}), \ldots, B(m_{\phi(l)}, \overline{R}_{\phi(l)})] =: [c_{i,1}, \ldots c_{i,n}] \quad (3)$$

for some permutation $\phi$. Then each mix $j$ decrypts $c_{i,j}$ and publishes $v_{i,j}$ for $i = 1, \ldots, l$. Then everybody can recover $m_{\phi(i)}$ from $k$ or more $v_{i,j}$.

Each mix uses ZKIP to show correctness of the calculation.

# Zero Knowledge: problems

- Many public key operations per transaction
- Why should node operators want to run this protocol?

# PipeNet

Wei Dei suggested *PipeNet*:

- initiator knows receiver identity, but not vice-versa
- layered encryption, forwarding without delay
- constant traffic on each link to avoid observability

Is this useful?

# Buses

A. Beimel and S. Dolev introduce *Buses for Anonymous Message Delivery*:

- Anonymity like in the public transportation system.
- A bus is a group of messages traveling on the network.
- Buses travel **fixed scheduled** routes.

# Buses: claim to fame

- sender and receiver anonymity (no search)
- not based on statistical properties
- communication causes no visible change on the network

# Buses: Communicaton Optimal Protocol

- One Bus
- with $n^2$ seats
- travels on a ring of $n$ nodes.

A message $M$ from $p_i$ to $p_j$ travels as $E_K(M)$ on seat $s_{i,j}$ where $K$ is either a symmetric key known to $p_i$ and $p_j$ or the public key of $p_j$.

# Buses: choices

Any implementation of this basic idea must define three essential properties of the system that are also critical for performance:

- size of the bus(es)
- latency (average number of stations until a passenger reaches his destination)
- number, frequency and routes of the buses

# Buses: Reducing the number of seats

The following idea can reduce the number of seats:

- In order to send a message, a node picks a random seat and puts the message there.
- In order to hide that a message was sent, all other seats must be changed.
- Decrypt all seats with the private key of the local host, encrypt seat with message onion-style.

How many seats do we expect to need for $m$ messages?

# Buses: Problems with seat reduction

- Each node must perform lots of public key operations, even on empty seats.
- Easy to attack (overwrite all seats with garbage).
- Accidential overwriting makes communication unreliable and introduces the need to send acknowledgments (increasing traffic and latency)

# Buses: Reducing latency

Use shortest-path routing:

- assume some graph over the nodes, with a bus traveling on each link in both directions in every time-slot.
- route seats through this graph on the shortest path to the receiver

# Buses: Problems with latency reduction

- routing information must be propagated
- seats must have some form of routing header
- large amount of traffic and often empty seats

# Buses: question

The bus schedule is known (or predictable).

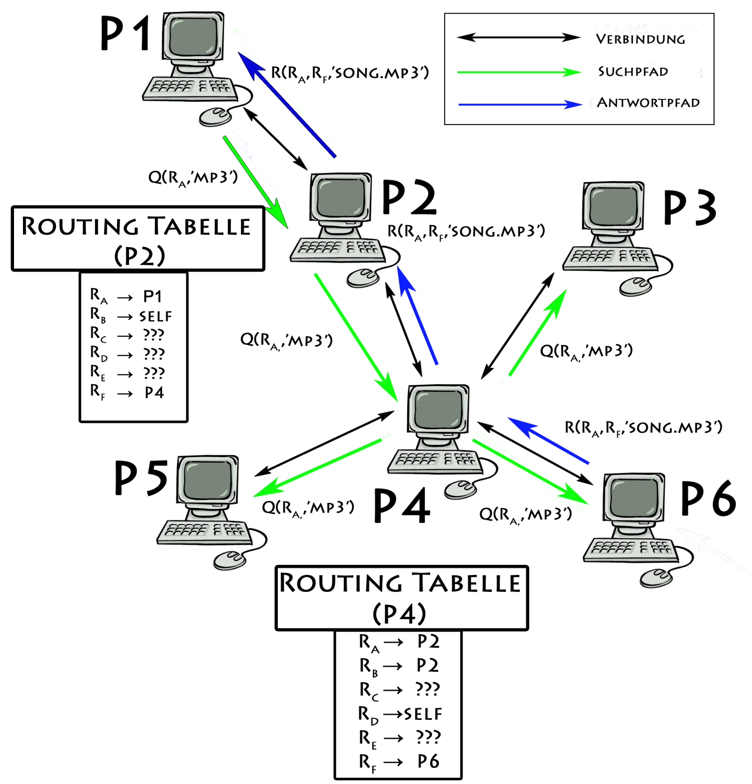
Supposed the adversary is also the recipient of a message.

**What can an active adversary do?**

# Buses: other problems

- scalability questionable ($O(n)$ and worse)
- potentially lots of noise (empty seats)
- many variations with individual benefits and drawbacks

# RShare/StealthNet



# Mute/Ants

Properties that a search-limiting mechanism should have:[1]

1. Single Integer Representation
2. Distributed Enforcement
3. Total Limit
4. Deterministic
5. Account for Branching
6. Account for Results

[1]according to Mute author Jason Rohrer

# Utility Counters

UC starts at zero. Without hop counter:

$$UC_{new} = UC_{old} + \alpha * |localResults| + \beta * |forwardSet| + \gamma$$

Improved formula with hop counter:

$$UC_{new} = UC_{old} + \alpha * |localResults| * HC + \beta * |forwardSet|^{1+\frac{1}{HC}} + \gamma$$

What is the impact of using UCs on anonymity?

# Mute Sender Anonymity

Use a hybrid approach for flodding:

- Initiator picks random 20-byte SHA1 hash value
- Each hop re-hashes the current value
- If last bytes is $\leq 51$, switch to utility counters

Does this solve the problem?

# Mute Responder Anonymity

Use a third approach for the end:

- Forward with branching until UC hits the limit
- Then switch to chain mode
- Each node on startup once determines an operational mode $n$ with probability $p(n)$, and in chain mode forwards to the same $n$ neighbours, where:

$$p(n) = \begin{cases} \frac{3}{4} & n = 0 \\ 2^{-n+2} & n > 0 \end{cases} \tag{4}$$

Does this solve the problem?

# Copyright