DC & Mix Networks

Bachelor Thesis

V. Manea

Faculty of Computer Science

June 28, 2011

Introduction

Anonymity

"not being identified in a set of subjects" [7].

Unobservability

"undetectability of the item against all subjects involved or not in it" [7].

These properties are necessary for systems that provide:

- electronic voting, shopping, browsing
- medical records (k-anonymity),
- governments and agencies, oppressive regimes,
- email transmission.

The most widespread mechanisms for providing these two properties are

1. DC Networks and
2. Mix Networks.

Contributions

We analyzed the state of the art in DC [4] and especially Mix [5] systems. The thesis surveys DC net properties, Mix net general properties, mix flushing algorithms, mix proof techniques and some deployed systems. We also propose an:

- **algorithm that solves the problem of messages delayed forever** in the mix, which is the case for many anonymity providing techniques. Our algorithm has great minimum anonymity and good maximum anonymity;
- **application of the Parallel Mix Network model** [1] by using the Hadoop open source distributed computing framework [2] based on the Map Reduce programming model [3]. We integrated a version of the algorithm in our application.

DC Networks

Problem

Send the message in a network, such that nobody can prove you sent it.

Solution

The solution is a DC Network. It consists of the following procedures:

1. Participant i exchanges symmetric keys with other participants:
$$k_{i,j} = -k_{j,i}, \text{ for any } i \neq j \in \{1, 2, ..., n-1, n\}.$$
2. At most one participant sends $m_i \neq 0$ in his public sum
$$s_i = k_{i,1} + ... + k_{i,n} + m_i.$$
3. The sums are all added up, yielding the result t:
$$t = s_1 + s_2 + ... + s_{n-1} + s_n = \Sigma(k_{i,j}) + \Sigma(m_k) =$$
$$= m_1 + m_2 + ... + m_{n-1} + m_n = m_i.$$

By seeing only the values s_j, an observer from outside is unable to trace the exact value m_i without knowing all $m_k \neq m_i$.

Mix Server

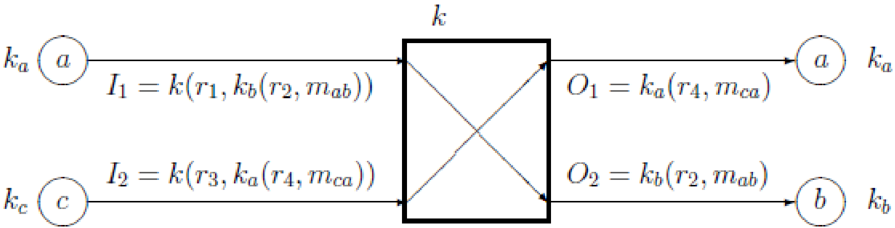
Problem

Send messages anonymously through the network.

Solution

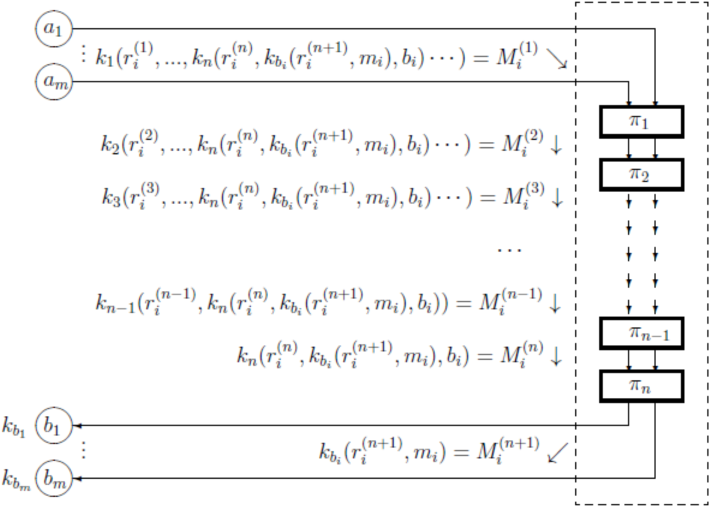
Receive n messages, secretly reorder them, and then send them.

The mix hides the correspondences between I_i and O_i. For n randomly valued inputs, the probability of having $I_i \to O_i$ for some i is n^{-1}.



Mix Networks

What if some servers are corrupt and share their permutation?

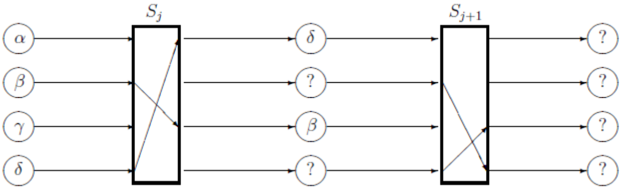


Proof techniques (selective)

Problem

Mixes must prove they performed correctly without revealing π [6].

- **Randomized partial checking**: based on making public parts of the path a message travels through the mix.



- **Optimistic mixing**: use message reencryption, but with
$$< E_y(m_i), E_y(h(m_i)) >=< (g^r, m_i y^r), (g^{r'}, h(m_i) y^{r'}) >.$$
- **Proof of subproduct**: prove that
$$\Pi(m_i) = \Pi(m_i') \wedge |M| = |M'|$$
for two sets $M = \{m_i\}$ and $M' = \{m_i'\}$ chosen at random.

Flush algorithms (selective)

Problem

How to resist blending attacks [6]?

- Threshold flush mix: wait for n messages to gather.
- Timed flush mix: wait for a fixed time of t units of time before flush.
- **Threshold pool mix**: When $n + N$ messages have entered the mix, chose random n to flush.
- Timed pool mix: Every t seconds, fire at random and still keep N messages inside.
- Timed dynamic pool mix: select $f(n)$ messages, where f is a function of the number of messages inside.
- **Stop and go mixes**: prepare message with time frame for each hop. If it comes either sooner of later, the message is discarded.
- **Red green black mixes**: when red messages do not come in, green messages are generated to hide black messages.

Round Robin Queue Flush algorithm

Problem

Some messages may remain in the mix forever.

Solution

Queue messages and visit queues circularly.

clear queues and list
add N dummy messages
start the rounds of the protocol
while *true* **do**
    clear input and output arrays
    read M input messages from $[input]$ to I
    spread the M input messages from I to queues Q_*
    select M messages from queues Q_* to O
    randomize array O before output
    write M messages from O to $[output]$

Round Robin Queue Flush algorithm

Data Structures

- Q queues and
- 1 circular double linked list.

Time Complexity

- The algorithm advances to the next queue in each round. This solves the undeterminate amount of time a message remains in the mix.
- The preprocessing yields to an $\mathcal{O}(N + Q)$ time complexity, where N is the pool size and Q is the number of queues.
- The round is in $\mathcal{O}(M)$ time, where M is the threshold size M.

Anonymity

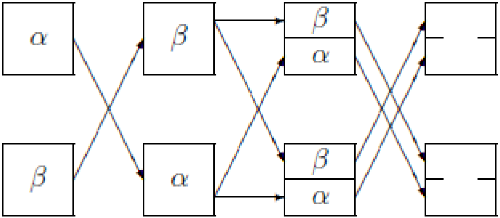
- The anonymity has the lower bound of $M + N$. In this case, the time spent by the message in the mix server is $\mathcal{O}(1)$ rounds.
- The maximum anonymity is upper bounded by $\mathcal{O}(MQ + N)$. The delay is $\mathcal{O}(MQ)$ rounds.

Parallel Mix Networks

Problem

It seems Mix Networks are sequential. Can they perform in parallel?[1].

- The first round $\rightarrow$ equiround. Each server mixes the assigned bucket.
- The rounds $1, 2, ..., n' - 1, n'$ $\rightarrow$ rotation:
$$S_{k+1}(j) = S_k((n + j - 2) \bmod n + 1).$$
- The round $n' + 2$ is a distribution of round $n' + 1$.
- The rounds $n' + 2, n' + 3, ..., 2n', 2n' + 1$ are also rotations.



MapReduce

Problem

Can we use Map Reduce [3] for Parallel Mixes?

- the map function processes a key-value pair to generate a set of intermediate key-value pairs,
- The reduce function merges all intermediate values associated with the same intermediate key.

An example of word counter in a large file using map reduce:

Input: document name as key
Input: document contents as value
Output: word and 1 as intermediate pair

for $word \in value$ **do**
  $[output] \leftarrow (word, 1)$

Input: word as key
Input: iterator of numbers as values
Output: word and 1 as intermediate pair

$result \leftarrow 0$
for $count \in values$ **do**
  $result \leftarrow result + count$
$[output] \leftarrow (key, result)$

Application Map Reduce Flow

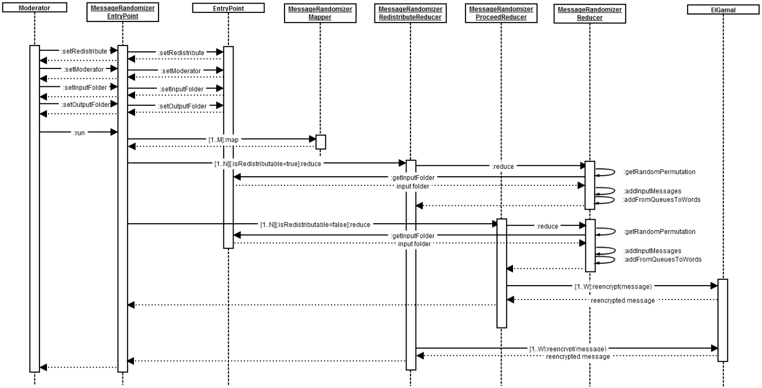
Parallel Mix Network Map Reduce particular steps are:

1. $input \rightarrow Probable\ Mix\ Assigner \rightarrow assigned$.
2. $assigned \rightarrow Assigned\ Mix\ Counter \rightarrow counted$.
3. $counted \rightarrow General\ Minimum\ Computer \rightarrow minimized$.
4. $assigned \rightarrow Message\ Selector \rightarrow selected$.
5. $selected \rightarrow Message\ Filter\ [criteria = \{t, f\}] \rightarrow \{round_0, next\}$.
6. $round_k \rightarrow Message\ Randomizer\ [redistribute = f] \rightarrow round_{k+1}$.
7. $round_{n'-1} \rightarrow Message\ Randomizer\ [redistribute = t] \rightarrow round_{n'}$.
8. $round_k \rightarrow Message\ Randomizer\ [redistribute = f] \rightarrow round_{k+1}$.

Application Sequence Diagram

The sequence diagram for the two Map Reduce steps:

- $round_k \rightarrow Message\ Randomizer\ [redistribute = false] \rightarrow round_{k+1}$.
- $round_{n'-1} \rightarrow Message\ Randomizer\ [redistribute = true] \rightarrow round_{n'}$.



Conclusions

1. **There is always a trade-off** between anonymity, secrecy, correctness and speed, as described in the techniques we surveyed.
2. **We proposed an algorithm for the starving message problem**, which is with the current techniques. The algorithm is efficient and its minimum and maximum anonymity are comparable to some algorithms in the literature.
3. **We implemented an application of the Parallel Mix Network** model by using the Hadoop open source distributed computing framework based on the Map Reduce programming model.

Thank you

Selected bibliography

P. Golle, A. Juels
Parallel Mixing
ACM Conference on Computer and Communications Security, 2004

Apache Software Foundation
Apache Hadoop
http://hadoop.apache.org; http://hadoop.apache.org/common/docs/current/

J. Dean, S. Ghemawat
MapReduce: Simplified Data Processing on Large Clusters
Symposium on Operating System Design and Implementation, 2004

Chaum, David
The Dining Cryptographers Problem: Unconditional Sender and Recipient Untraceability
Journal of Cryptology, volume 1, issue 1, 1988

Chaum, David
Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms
Communications of the ACM, volume 24, issue 2, 1981

Edman, Matthew; Yener, Bülent
On Anonymity in an Electronic Society: A Survey of Anonymous Communication Systems
ACM Computing Surveys, 2009

Pfitzmann, Andreas; Hansen, Marit
Anonymity, Unlinkability, Undetectability, Unobservability, Pseudonymity, and Identity Management A Consolidated Proposal for Terminology
http://dud.inf.tu-dresden.de/Anon_Terminology.shtml