k-Anonymous Message Transmission
Luis von Ahn
Andrew Bortz
Nick Hopper
The Aladdin Center
Carnegie Mellon University

Sender Anonymous Protocol
Adversary cannot identify the sender
of a particular message

Receiver Anonymous Protocol
Adversary cannot identify the receiver
of a particular message

Some Applications
Secret Love Letters
Anonymous Crime Tips
Distribution of Music
Sender and receiver anonymity can be achieved with a trusted third party

In This Talk
We will present a scheme for anonymous communication that is efficient and requires no trusted third parties

The Model
Reliable Communication
The adversary can see all communications in network
The adversary can own some of the participants
A participant owned by the adversary may act arbitrarily
The Rest of the Talk

DC Nets
Why DC Nets have never been implemented
k-Anonymity
An efficient scheme

DC Nets: Key Idea

Divide time into small steps
At step $t$, party $i$ wants to send message $M_i \in \mathbb{Z}_m$
If party $j$ doesn’t want to send a message at step $t$, they must send $M_j=0$

DC Nets: Key Idea

Each party distributes their $n$ shares

\[ M_i = s_{i,1} + s_{i,2} + \ldots + s_{i,n-1} + (M_i - (s_{i,1} + \ldots + s_{i,n-1})) \]

\[ s_{i,n} \]
DC Nets: Key Idea

All parties add up every share that they have received and broadcast the result (Let $B_i$ denote Party $i$'s broadcast)

$$B_i = s_{1,i} + s_{2,i} + \ldots + s_{n,i}$$

DC Nets: Key Idea

If only one of the $M_i$ is nonzero, then:

$$B_1 + B_2 + \ldots + B_n = M_i$$

DC Nets: Key Idea

All parties add up every share that they have received and broadcast the result (Let $B_i$ denote Party $i$’s broadcast)

$$M_i = s_{i,1} + s_{i,2} + \ldots + s_{i,n-1} + s_{i,n}$$

$$B_i = s_{1,i} + s_{2,i} + \ldots + s_{n,i}$$

$$B_1 + B_2 + \ldots + B_n = M_1 + M_2 + \ldots + M_n$$

DC Nets: Problems

It is very easy for the adversary to jam the channel!

Communication complexity is $O(n^2)$
Full Anonymity Versus k-Anonymity

We will relax the requirement that the adversary learns nothing about the origin of a given message.

We will accept k-anonymity, in which the adversary can only narrow down his search to k participants.

The Rest of the Talk

DC Nets
Why DC Nets have never been implemented
k-Anonymity
An efficient scheme

k-anonymous message transmission (k-AMT)

Idea: Divide N parties into “small” DC-Nets of size O(k). Encode M_t as (group, msg) pair.

\[ s_{1,1} + s_{1,2} + s_{1,3} + s_{1,4} = (G_t, M_t) \]

How to compromise k-anonymity

- If everyone follows the protocol, it’s impossible to compromise the anonymity guarantee.
- So instead, don’t follow the protocol: if Alice can never send anonymously, she will have to communicate using onymous means.
How to break k-AMT (I)

- Don’t follow the protocol: after receiving shares \( s_{1,r}, \ldots, s_{k,r} \) instead of broadcasting \( s_r \), generate a random value \( r \) and broadcast that instead.
- This will randomize the result of the DC-Net protocol, preventing Alice from transmitting.

Stopping the “randomizing” attack

- Solution: Use Verifiable Secret Sharing. Every player in the group announces (by broadcast) a commitment to all of the shares of her input.
- These commitments allow verification of her subsequent actions.

k-anonymous message transmission (k-AMT) with VSS

Before starting, each player commits to \( s_{i,1} \ldots s_{i,k} \) via Pedersen commitment \( C(s,r) = g^r h^s \)

\[
s_{i,1} + s_{i,2} + s_{i,3} + s_{i,4} = x_i = (G,M)
\]

<table>
<thead>
<tr>
<th>C</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( g^{r_1} h^{s_{1,1}} )</td>
<td>( g^{r_1} h^{s_{1,1}} )</td>
</tr>
<tr>
<td>2</td>
<td>( g^{r_2} h^{s_{1,2}} )</td>
<td>( g^{r_2} h^{s_{1,2}} )</td>
</tr>
<tr>
<td>3</td>
<td>( g^{r_3} h^{s_{1,3}} )</td>
<td>( g^{r_3} h^{s_{1,3}} )</td>
</tr>
<tr>
<td>4</td>
<td>( g^{r_4} h^{s_{1,4}} )</td>
<td>( g^{r_4} h^{s_{1,4}} )</td>
</tr>
</tbody>
</table>

k-anonymous message transmission (k-AMT) with VSS

Before starting, each player commits to \( s_{i,1} \ldots s_{i,k} \) via Pedersen commitment \( C(s,r) = g^r h^s \)

\[
s_{i,1} + s_{i,2} + s_{i,3} + s_{i,4} = x_i = (G,M)
\]

<table>
<thead>
<tr>
<th>C</th>
<th>1</th>
<th>k</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( g^{r_1} h^{s_{1,1}} )</td>
<td>( g^{r_1} h^{s_{1,1}} )</td>
</tr>
<tr>
<td>2</td>
<td>( g^{r_2} h^{s_{1,2}} )</td>
<td>( g^{r_2} h^{s_{1,2}} )</td>
</tr>
<tr>
<td>3</td>
<td>( g^{r_3} h^{s_{1,3}} )</td>
<td>( g^{r_3} h^{s_{1,3}} )</td>
</tr>
<tr>
<td>4</td>
<td>( g^{r_4} h^{s_{1,4}} )</td>
<td>( g^{r_4} h^{s_{1,4}} )</td>
</tr>
</tbody>
</table>

\[
g^{r_k} h^{s_{i,k}}
\]

\[
g^{r_k} h^{s_{i,k}}
\]
How to break k-AMT (II)

- The multiparty sum protocol gives k participants a single shared channel: at most one person can successfully transmit each turn.
- So: Transmit every turn! VSS still perfectly hides the value of each input; no one will know who is hogging the line.

Accommodating more than one sender per turn

- Idea: we can run several turns in parallel. Instead of sending commitments to shares of a single value, generate shares of 2k values.
- If Alice picks a random “turn” to transmit in, she should have probability at least 1/2 of successfully transmitting.

Accommodating more than one sender per turn

Before starting, each player picks slot s, sets $x_{i,s} = (G_t M_t)$, $x_{i,1} = ... = x_{i,2k} = 0$, and chooses $s_{i,j,m}$ so that $\sum_m s_{i,j,m} = x_{ij}$
Catching a cheater

- Idea: each party can use her committed values to prove (in zero knowledge) that she transmitted in at most one slot, without revealing that slot.
- If someone did cheat, she will have a very low probability of convincing the group she did not.

Zero-Knowledge proof of protocol conformance

- $P_i \rightarrow (\text{All})$:
  - Pick permutation $\rho$ on $\{1...2k\}$
  - Send $C(x') = C(x_{\rho(0)}, r'_0),..., C(x_{\rho(2k)}, r'_{2k})$
- $(\text{All}) \rightarrow P_i$: $b \in \{0,1\}$
- $P_i \rightarrow (\text{All})$:
  - if $b = 0$: open $2k-1$ 0 values;
  - else reveal $\rho$, prove (in ZK) $x' = \rho(x)$

Efficiency

- $O(k^2)$ protocol messages to transmit $O(k)$ anonymous messages: $O(k)$ message overhead
- Cheaters are caught with high probability
- Zero Knowledge proofs are Honest Verifier and can be done non-interactively in the Random Oracle Model, or interactively via an extra round (commit to verifier coins)