Dissent: Making Strong Anonymity Scale

Aleksandra Zakrzewska, Katherine Tan, Ethan Weinberger, Justin Shi, Leo Shimonaka, Alex Tong
Motivations

- Network communication is great, but we do not want it to be public
- Anonymity preserves democratic culture, freedom of speech

I am scared to share my message.
Motivations

- Network communication is great, but we do not want it to be public
- Anonymity preserves democratic culture, freedom of speech
- We want to preserve message anonymity in a large set
Motivations

- Network communication is great, but we do not want it to be public
- Anonymity preserves democratic culture, freedom of speech
- We want to preserve message anonymity in a large set
- There’s usually a tradeoff between security and convenience
  - However, more convenient systems typically attract more users
- A larger anonymity set gives increased anonymity, which means that some “weaker” systems may end up having a security advantage in practice
  - Tor vs DC-nets
- What about a system that is secure and convenient/scalable?
  - Dissent

I want my message to be secure, but also get there quickly
Previous State of the Art is not Good

- Onion Routing (Tor)
- DC-nets

Choice between large but weak anonymity set or strong but small anonymity set
Onion Routing (Tor)

Overview:

- Sender encrypts message repeatedly into an Onion
- Each server decrypts one layer and sends it to the next destination
- No one knows both the source and destination of the message
Onion Routing (Tor)
Onion Routing (Tor)

User encrypts chooses an N-hop path through the servers and encrypts their message N times to create an onion. Each layer specifies the next hop in the path.
Onion Routing (Tor)

Each server peels (decrypts) one layer and passes the onion on to the next specified server.
Onion Routing (Tor)

Each server peels (decrypts) one layer and passes the onion on to the next specified server.
Onion Routing (Tor)

Each server peels (decrypts) one layer and passes the onion on to the next specified server.
Onion Routing (Tor)

Each server peels (decorts) one layer and passes the onion on to the next specified server.
Onion Routing (Tor)

Each server peels (decrypts) one layer and passes the onion on to the next specified server.
Onion Routing (Tor)

The end user receives the message.
Onion Routing (Tor)

Is this secure?
Onion Routing (Tor)

Advantages:

- Ease of use and scalability attracts many users
- Anonymity set is large
Onion Routing (Tor)

Vulnerabilities:

- Traffic analysis
- Active disruption attacks
- Isolation of anonymous and non-anonymous communication
Traffic Analysis

If an ISP (or the NSA) can see traffic in and out of the system, they can still figure out who is sending messages by correlating the message flows!
Onion Routing (Tor)

Vulnerabilities:

- Traffic analysis
  - Today, many national ISPs can monitor and fingerprint traffic patterns and de-anonymize Tor flows starting and ending in their territory
- Active disruption attacks
- Isolation of anonymous and non-anonymous communication
Onion Routing (Tor)

Vulnerabilities:

- **Traffic analysis**
  - Today, many national ISPs can monitor and fingerprint traffic patterns and de-anonymize Tor flows starting and ending in their territory

- **Active disruption attacks**
  - DoS attacks on servers that cannot be controlled by an adversary can route traffic to malicious servers
  - Servers can collude to perform better traffic analysis

- **Isolation of anonymous and non-anonymous communication**
Onion Routing (Tor)

Vulnerabilities:

- **Traffic analysis**
  - Today, many national ISPs can monitor and fingerprint traffic patterns and de-anonymize Tor flows starting and ending in their territory

- **Active disruption attacks**
  - DoS attacks on servers that cannot be controlled by an adversary can route traffic to malicious servers
  - Servers can collude to perform better traffic analysis

- **Isolation of anonymous and non-anonymous communication**
  - Tools often fail to isolate anonymous and non-anonymous communication
  - This allows application-layer identity leaks
Onion Routing (Tor)

- Result: Tor offers a large, but weak anonymity set
DC-nets Overview

- **Small**, but **strong** anonymity set
- To exchange a 1-bit message:
  - Every member shares a secret random coin with each of the $N - 1$ group members
  - Each member individually XORs together the values of all the coins he/she shares to create a ciphertext
    - The anonymous sender additionally XORs their 1-bit message
  - All members broadcast their ciphertexts to everyone else
  - Each member XORs all the ciphertexts to reveal the anonymous message without revealing the sender
DC-nets

To exchange a 1-bit message, each member shares a secret random coin with each of the other $N - 1$ members.
DC-nets

Cleartext message

To exchange a 1-bit message, each member shares a secret random coin with each of the other $N - 1$ members.
To exchange a 1-bit message, each member shares a secret random coin with each of the other $N - 1$ members.

Each member individually XORs the values of the coins he/she shares.

DC-nets

Cleartext message

1

1

1

0

1

0

1

Each member individually XORs the values of the coins he/she shares.
To exchange a 1-bit message, each member shares a secret random coin with each of the other $N - 1$ members.

Each member individually XORs the values of the coins he/she shares.

Cleartext message

DC-nets
DC-nets

To exchange a 1-bit message, each member shares a secret random coin with each of the other $N - 1$ members.

Each member individually XORs the values of the coins he/she shares.

To exchange a 1-bit message, each member shares a secret random coin with each of the other $N - 1$ members.
To exchange a 1-bit message, each member shares a secret random coin with each of the other $N-1$ members.

Each member individually XORs the values of the coins he/she shares.

Finally, broadcast the ciphertexts and XOR the results to get the message.
To exchange a 1-bit message, each member shares a secret random coin with each of the other $N - 1$ members.

Finally, broadcast the ciphertexts and XOR the results to get the message.
DC-nets

To exchange a 1-bit message, each member shares a secret random coin with each of the other \( N - 1 \) members.

Each member individually XORs the values of the coins he/she shares.

Finally, broadcast the ciphertexts and XOR the results to get the message.
Since every coin is shared between exactly 2 members, the XORs cause all bits but the anonymous message to cancel!
DC-nets: Why They Work

Since every coin is shared between exactly 2 members, the XORs cause all bits but the anonymous message to cancel!
Since every coin is shared between exactly 2 members, the XORs cause all bits but the anonymous message to cancel!
DC-nets: Why They Work

Since every coin is shared between exactly 2 members, the XORs cause all bits but the anonymous message to cancel!
DC-nets: Why They Work

Since every coin is shared between exactly 2 members, the XORs cause all bits but the anonymous message to cancel!
For an n-bit message, imagine the same protocol but with n coins instead of 1 coin per pair.
Advantages of DC-nets

- Unlike Tor, DC-nets are resistant to traffic analysis
  - All members send ciphertexts for each message round
  - All members transmit equal-length messages
Limitations of DC-Nets

- Need for a scheduling mechanism
  - Only one anonymous message at a time to avoid collisions
- Ease of disruption
  - Imagine one malicious user sends random bits all the time
- Not scalable:
  1) Each node must compute and combine $O(N)$ ciphertexts
     a) Total time: $O(N^2)$
  2) Network churn
     a) Every round needs to wait for ALL members to submit their ciphertexts
     b) One slow member delays the whole system
     c) Member exiting system forces system to restart
We want a **practical** and **scalable** anonymous group communication system resistant to traffic analysis.
Proposed Solution: Dissent

Reemphasizing motivations, we want:

- Improved scalability (computation/communication efficiency)
- Robustness (resistance from slowdown caused by slow node or network churn)
- The ability to detect malicious disrupting nodes
- All of this without sacrificing strong anonymity
Design & Deployment Assumptions

- Assume a cloud-like multi-provider deployment model
- Anytrust assumption
  - (at least one uncompromised provider)
  - Do NOT need to know which server(s) are trustworthy
- Do NOT have assumptions on trustworthiness of users
Protocol Outline

- Servers periodically run scheduling process given clients' private keys
- Clients know what transmission slot is theirs
- Clients do not know which slots are for which other clients
- Servers also do not know which clients hold which slots, thanks to the Secret Shuffle™

[Diagram of Secret Shuffle™ process]
Per exchange round:
Per exchange round:
Per exchange round:

\[ S_{c,x} \oplus S_{c,y} \oplus N_c \]
\[ S_{a,x} \oplus S_{a,y} \oplus M_a \]

\[ S_{b,x} \oplus S_{b,y} \]
\[ S_{c,x} \oplus S_{c,y} \oplus N_c \]
Per exchange round:
\[ S_{c,x} \oplus S_{c,y} \oplus N_c \]
\[ S_{a,x} \oplus S_{a,y} \oplus M_A \]
\[ S_{b,x} \oplus S_{b,y} \]

\[ S_{c,x} \oplus S_{c,y} \oplus N_c \]
\[ S_{a,x} \oplus S_{a,y} \oplus M_A \]
\[ S_{b,x} \oplus S_{b,y} \]

\[ S_{c,x} \oplus S_{c,y} \oplus N_c \]
\[ S_{a,x} \oplus S_{a,y} \oplus M_A \]
\[ S_{b,x} \oplus S_{b,y} \]

\[ S_{c,x} \oplus S_{c,y} \oplus N_c \]
\[ S_{a,x} \oplus S_{a,y} \oplus M_A \]
\[ S_{b,x} \oplus S_{b,y} \]

\[ S_{c,x} \oplus S_{c,y} \oplus N_c \]
\[ S_{a,x} \oplus S_{a,y} \oplus M_A \]
\[ S_{b,x} \oplus S_{b,y} \]

\[ S_{c,x} \oplus S_{c,y} \oplus N_c \]
\[ S_{a,x} \oplus S_{a,y} \oplus M_A \]
\[ S_{b,x} \oplus S_{b,y} \]

\[ S_{c,x} \oplus S_{c,y} \oplus N_c \]
\[ S_{a,x} \oplus S_{a,y} \oplus M_A \]
\[ S_{b,x} \oplus S_{b,y} \]

\[ S_{c,x} \oplus S_{c,y} \oplus N_c \]
\[ S_{a,x} \oplus S_{a,y} \oplus M_A \]
\[ S_{b,x} \oplus S_{b,y} \]

\[ S_{c,x} \oplus S_{c,y} \oplus N_c \]
\[ S_{a,x} \oplus S_{a,y} \oplus M_A \]
\[ S_{b,x} \oplus S_{b,y} \]

\[ S_{c,x} \oplus S_{c,y} \oplus N_c \]
\[ S_{a,x} \oplus S_{a,y} \oplus M_A \]
\[ S_{b,x} \oplus S_{b,y} \]

\[ S_{c,x} \oplus S_{c,y} \oplus N_c \]
\[ S_{a,x} \oplus S_{a,y} \oplus M_A \]
\[ S_{b,x} \oplus S_{b,y} \]

\[ S_{c,x} \oplus S_{c,y} \oplus N_c \]
\[ S_{a,x} \oplus S_{a,y} \oplus M_A \]
\[ S_{b,x} \oplus S_{b,y} \]

\[ S_{c,x} \oplus S_{c,y} \oplus N_c \]
\[ S_{a,x} \oplus S_{a,y} \oplus M_A \]
\[ S_{b,x} \oplus S_{b,y} \]

\[ S_{c,x} \oplus S_{c,y} \oplus N_c \]
\[ S_{a,x} \oplus S_{a,y} \oplus M_A \]
\[ S_{b,x} \oplus S_{b,y} \]

\[ S_{c,x} \oplus S_{c,y} \oplus N_c \]
\[ S_{a,x} \oplus S_{a,y} \oplus M_A \]
\[ S_{b,x} \oplus S_{b,y} \]
\[
\begin{align*}
S_{c,x} \oplus S_{c,y} \oplus M_c \\
S_{a,x} \oplus S_{a,y} \oplus M_a \\
S_{b,x} \oplus S_{b,y}
\end{align*}
\]
$M_A \oplus M_C$

I do not know who sent this msg

$M_A \oplus M_C$

I do not know who sent this msg

$M_A \oplus M_C$

I do not know who sent this msg
Did we achieve our goals?
Scalability

- First let’s review DC nets again
DC-nets

To exchange a 1-bit message, each member shares a secret random coin with each of the other $N - 1$ members.
To exchange a 1-bit message, each member shares a secret random coin with each of the other $N - 1$ members.
To exchange a 1-bit message, each member shares a secret random coin with each of the other $N - 1$ members.
Why is dissent scalable?

- Only need to share secrets between client/server pairs
- Less network traffic
- Less computational load on clients
Tolerating Network Churn

Recall from DC-nets

Network churn:

1) Every round needs to wait for ALL members to submit their ciphertexts
2) One slow member delays the whole system
3) Member exiting system forces system to restart
Tolerating Network Churn

- Clients only share secrets with servers, and NOT other clients
  - A client’s ciphertext is independent of the status of other clients
- As long as servers agree on the set of client ciphertexts after some deadline, we don’t care about clients leaving or taking excessive amounts of time
Strength in Numbers

- Outside actors may have some knowledge on the Dissent group
- Users may wish to only send messages if the Dissent group is large
- Servers keep participation count → clients decide to publish based on count
Strength in Numbers
Strength in Numbers
Strength in Numbers
Identifying Disruptors

How to Identify a Dishonest Server?
Identifying Disruptors

- Server A thinks something went wrong. Creates an accusation.
- How to identify a witness bit? “This 1 at index i should be 0”
- How to send?
- How to trace disruptor?
Identify a witness

- Problem: Predictable cleartext
- Solution: Randomize with cryptographic padding

Example: HTTP Header Starts with GET
01000111 01000101 01010100

Disruptor flips second bit, corrupting the request without leaving a witness bit
Sending Accusation

- Send over DC Net?
- Problem: Repeated disruption
- Solution: Verifiable Shuffle
Tracing Disruptor

- Gather everything but the client plaintext
- Recompute to deduce where the error occurred

Cases:

- Server i does not complete accusation phase
- Server i switched bit before commitment
- Servers’ version of ciphertext does not match client j’s version \( \text{XOR}_{i} s_{ij} \neq c_{j} \)
  - Honest client can then compute \( s_{ij} \) to find malicious server i
Participation and Anonymity Metrics

- Having a measure of how anonymous you are may be desired.
- “Strength in numbers” - Clients may only want to transmit messages in a reasonably sized anonymity set.
- Dissent uses Participation Counts
Participation Counts

- Servers know currently online clients and clients participating in each round.
  - **Participation Count**: Number of participating clients
- Process:
  - Servers can publish participation count for a previous round
  - Clients can choose to only send nonempty messages for large-enough participation counts.
Handling Fluctuation in Participation Counts

- Dissent assumes clients come and go as they please
- **Attack Vector:**
  - Published participation count and following round's participation count can be very different
- **Solution:**
  - Define constant $\alpha$ in range $[0, 1]$ at each server.
  - Given previous participation count $P$, postpone round completion until $\alpha * P$ ciphertexts are received.
  - Add hard timeout to discard all messages in rounds with less than $\alpha * P$ participants.
Eliminating Empty Slot Overhead

- Fixed-width slots is wasteful. E.g. Chat Clients.
- How do we allocate slots for clients on a need basis?
- Dissent’s scheduling scheme uses two slots per client: 1-bit Request Slot and Variable length Message Slot
Requesting Slots

- Initially, variable message slot is closed.
- **Opening Message Slot:**
  - Client sets request slot to 1 in round $r$ to open its message slot
  - Message slot is open in subsequent round, $r+1$
- **Growing / Closing Message Slot:**
  - Message slot includes length field to increase and decrease slot size in subsequent rounds.
Handling DDoS Attacks abusing Request Slots

- **Request Slots** introduce attack vector:
  - If attacker can guess when victim will transmit, attacker can send 1 in victim’s request slot to cancel victim’s open request.

- **Solution:**
  - Client initially sets request bit unconditionally.
  - If opening a message slot ever fails, randomize request bit in subsequent rounds until message slot opens.
Limitations

- Cannot handle Internet scale networks
- Vulnerable to Intersection Attacks
- Downtime for any server failure
- Statically managed Groups and Servers
## Scalability Comparison

<table>
<thead>
<tr>
<th></th>
<th>DC-Net</th>
<th>Dissent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Messages</td>
<td>$O(N^2)$</td>
<td>$O(N)$</td>
</tr>
<tr>
<td>Secrets</td>
<td>$O(N^2)$</td>
<td>$O(N*M)$</td>
</tr>
<tr>
<td>Anonymity Size</td>
<td>$O(H)$</td>
<td>$O(H)$</td>
</tr>
</tbody>
</table>

$N = \# \text{Clients}$  
$M = \# \text{Servers}$  
$H = \# \text{Honest}$