DISSENT: Accountable, Anonymous Communication

Joan Feigenbaum
http://www.cs.yale.edu/homes/jf/

Joint work with Bryan Ford (PI), Henry Corrigan-Gibbs, Ramakrishna Gummadi, Aaron Johnson (NRL), Vitaly Shmatikov (UT Austin), Ewa Syta, and David Wolinsky

Supported by DARPA
Problem Statement

• A group of $N \geq 2$ parties wish to communicate anonymously, either with each other or with someone outside of the group.

• They have persistent, “real-world” identities and are known, by themselves and the recipients of their communications, to be a group.

• They want a protocol with four properties:
  ✓ Integrity
  ✓ Anonymity
  ❖ Accountability
    o Efficiency
Accountability

• Group member $i$ exposes group member $j$ if $i$ obtains proof, verifiable by a third party (not necessarily in the group), that $j$ disrupted a protocol run.

• The protocol maintains accountability if no honest member is ever exposed, and, after every run, either:
  - every honest member successfully receives every honest member’s message, or
  - every honest member exposes at least one disruptive member.
Need for Anonymity (1)

- Communication in hostile environments

From the BAA: “The goal of the program is to develop technology that will enable safe, resilient communications over the Internet, particularly in situations in which a third party is attempting to discover the identity or location of the end users or block the communication.”
Need for Anonymity (2)

- Cash transactions
- Twelve-step programs (pseudonymy)
- Law-enforcement “tip” hotlines
- Websites about sensitive topics, e.g., sexuality, politics, religion, or disease
- Voting
- ...
Need for Accountability

• Authoritative, credentialed group, e.g.:
  o Board of Directors of an organization
  o Federation of journalists (… think Wikileaks)
  o Registered voters

• Internal disagreement is inevitable.
• Infiltration by the enemy may be feasible.
  ➢ Disruption is expected and must be combated.

? It’s not clear that “accountability” is the right word to use here (… and that’s part of a longer story).
Outline

• Prior work on anonymous communication

• Basic DISSENT protocol (ACM CCS 2010)

• Results to date
Outline

• Prior work on anonymous communication

• Basic DISSENT protocol (ACM CCS 2010)

• Results to date
Major Themes in Prior Work

• General-purpose anonymous-communication mechanisms
  o MIX networks and Onion Routing (OR)
  o Dining-Cryptographers networks (DC-nets)

• Special-purpose mechanisms, e.g.:
  o Anonymous voting
  o Anonymous authentication, e.g., group or ring signatures
  o E-cash
Basic Operation of Onion Routing

- Client picks a few (e.g., three) anonymizing relays from a cloud of available relays.
- He then builds and uses an onion of cryptographic tunnels through the relays to his communication partner.
Basic Operation of Onion Routing

• Client picks a few (e.g., three) anonymizing relays from a cloud of available relays.
• He then builds and uses an onion of cryptographic tunnels through the relays to his communication partner.
Basic Operation of Onion Routing

• Client picks a few (e.g., three) **anonymizing relays** from a cloud of available relays.
• He then builds and uses an **onion** of cryptographic tunnels through the relays to his communication partner.
Basic Operation of Onion Routing

• Client picks a few (e.g., three) **anonymizing relays** from a cloud of available relays.
• He then builds and uses an **onion** of cryptographic tunnels through the relays to his communication partner.
Basic Operation of Onion Routing

• Client picks a few (e.g., three) **anonymizing relays** from a cloud of available relays.
• He then builds and uses an **onion** of cryptographic tunnels through the relays to his communication partner.
Properties of Onion Routing

• Key advantages:
  o Scalable to large groups of clients and relays
  o Can be made interactive (e.g., Tor)
  o Widely deployed (e.g., Tor)

• Key disadvantages:
  o Many vulnerabilities to traffic analysis
  o No accountability: Anonymous disruptors can
    – Spam or DoS-attack relays or innocent nodes
    – Compromise other users’ anonymity
  [Borisov et al. ’07]
Dining Cryptographers (DC-nets)

- Information-theoretic group anonymity
- Ex. 1: “Alice+Bob” sends a 1-bit secret to Charlie.
Dining Cryptographers (DC-nets)

- Information-theoretic group anonymity
- Ex. 1: “Alice+Bob” sends a 1-bit secret to Charlie.
Dining Cryptographers (DC-nets)

- Information-theoretic group anonymity
- Ex. 1: “Alice+Bob” sends a 1-bit secret to Charlie.

Alice’s Secret

Alice

Alice+Bob’s Shared Random Bit

Bob

1

Charlie

0
Dining Cryptographers (DC-nets)

- Information-theoretic group anonymity
- Ex. 1: “Alice+Bob” sends a 1-bit secret to Charlie.
Dining Cryptographers (DC-nets)

• Information-theoretic group anonymity
• Ex. 1: “Alice+Bob” sends a 1-bit secret to Charlie.
Dining Cryptographers (DC-nets)

- Information-theoretic group anonymity
- Ex. 2: Homogeneous 3-member anonymity group
Dining Cryptographers (DC-nets)

- Information-theoretic group anonymity
- Ex. 2: Homogeneous 3-member anonymity group

![Diagram of Dining Cryptographers (DC-nets)]
Dining Cryptographers (DC-nets)

- Information-theoretic group anonymity
- Ex. 2: Homogeneous 3-member anonymity group
Dining Cryptographers (DC-nets)

- Information-theoretic group anonymity
- Ex. 2: Homogeneous 3-member anonymity group
Dining Cryptographers (DC-nets)

- Information-theoretic group anonymity
- Ex. 2: Homogeneous 3-member anonymity group
Dining Cryptographers (DC-nets)

- Information-theoretic group anonymity
- Ex. 2: Homogeneous 3-member anonymity group
Properties of DC-nets Schemes

• Key advantages:
  o Provable, information-theoretic anonymity
  o Resistance to traffic analysis and collusion

• Key disadvantages:
  o Not easy to scale up or implement efficiently
  o Not widely deployed
  o No accountability: Anonymous disruptors can
    – Spam or DoS-attack the group without discovery
    – Force group reformation without being eliminated
Outline

- Prior work on anonymous communication

- *Basic DISSENT protocol (ACM CCS 2010)*

- Results to date
Starting Point: Verifiable, Anonymous Shuffling
[Brickell and Shmatikov ’06]

- N parties with equal-length messages $m_1, ..., m_N$ send $m_{\pi(1)}, ..., m_{\pi(N)}$ to a data collector.
- The protocol provably provides
  - Integrity: $\{m_1, ..., m_N\} = \{m_{\pi(1)}, ..., m_{\pi(N)}\}$
  - Anonymity: $\pi$ is random and not known by anyone.
  - Resistance to traffic analysis and collusion
- DISSENT adds accountability and the ability to handle variable-length messages efficiently.
Basic DISSENT Protocol: Overview

• Assumptions:
  – Equal-length messages
  – Each group member has a signature key pair; all messages are signed.

• Phase 1: Setup
  – Each member chooses two encryption key pairs for this run.

• Phase 2: Onion encryption
  – Each member encrypts his message with everyone’s encryption keys.

• Phase 3: Anonymization
  – Each member applies a random permutation to the set of messages.

• Phase 4: Validation
  – Each member $i$ checks that (uncorrupted) $m_i$ is in the permuted set.

• Phase 5: Decryption or Blame
  – If all phase-4 checks succeed, decrypt all of the messages.
  – Else, honest members run a protocol that allows each of them to expose at least one disruptive member.
Phase 1: Setup

• Recall that
  o Members know each others' public verification keys.
  o Members sign (and verify signatures on) all messages.

• Each group member $i$ chooses:
  o Secret message $m_i$ (and pads it if necessary)
  o Outer encryption key pair $(O_i, O'_i)$
  o Inner encryption key pair $(I_i, I'_i)$

• Each group member $i$ broadcasts public encryption keys $O_i, I_i$
Phase 2: Onion Encryption

Each group member $i$:

- Encrypts $m_i$ with inner keys $I_N,\ldots,I_1$ to create $m'_i$
- Encrypts $m'_i$ with outer keys $O_N,\ldots,O_1$ to create $m''_i$
Phase 2: Onion Encryption

Each group member \( i \):

- Encrypts \( m_i \) with inner keys \( l_N, \ldots, l_1 \) to create \( m'_i \)
- Encrypts \( m'_i \) with outer keys \( O_N, \ldots, O_1 \) to create \( m''_i \)

Example with \( N = 3 \):
Phase 2: Onion Encryption

Each group member $i$:

- Encrypts $m_i$ with inner keys $I_N,\ldots,I_1$ to create $m'_i$
- Encrypts $m'_i$ with outer keys $O_N,\ldots,O_1$ to create $m''_i$

Example with $N = 3$:

\[ m_1, m_2, m_3 \]
Phase 2: Onion Encryption

Each group member $i$:

- Encrypts $m_i$ with inner keys $l_N,\ldots,l_1$ to create $m'_i$
- Encrypts $m'_i$ with outer keys $O_N,\ldots,O_1$ to create $m''_i$

Example with $N = 3$:

\[
\begin{align*}
    m'_1 &= \{ \{ \{ m_1 \} \} l_3 \} l_2 \} l_1 \\
    m'_2 &= \{ \{ \{ m_2 \} \} l_3 \} l_2 \} l_1 \\
    m'_3 &= \{ \{ \{ m_3 \} \} l_3 \} l_2 \} l_1
\end{align*}
\]
Phase 2: Onion Encryption

Each group member $i$:

- Encrypts $m_i$ with inner keys $I_N, ..., I_1$ to create $m'_i$
- Encrypts $m'_i$ with outer keys $O_N, ..., O_1$ to create $m''_i$

Example with $N = 3$:

- $m''_1 = \{ \{ \} \} \{ m_1 \} \{ I_3 \} \{ I_2 \} \{ I_1 \} \{ O_1 \} \{ O_2 \} \{ O_3 \}$
- $m''_2 = \{ \{ \} \} \{ m_2 \} \{ I_3 \} \{ I_2 \} \{ I_1 \} \{ O_1 \} \{ O_2 \} \{ O_3 \}$
- $m''_3 = \{ \{ \} \} \{ m_3 \} \{ I_3 \} \{ I_2 \} \{ I_1 \} \{ O_1 \} \{ O_2 \} \{ O_3 \}$
Phase 3: Anonymization (1)

- Member 1 collects \((m''_1, \ldots, m''_N)\).
- For \(i \leftarrow 1\) to \(N\), member \(i\)
  - Decrypts the \(i^{th}\) layer of outer-key encryption
  - Randomly permutes the resulting list (of partially decrypted messages) and (temporarily) saves the random permutation
  - Forwards the permuted list to member \(i+1\) (if \(i < N\))
- Member \(N\) broadcasts the permuted \(m'_i\) list.
Phase 3: Anonymization (2)

\[ m''_i = \{ \{ \} \} \quad m'_i = \{ \{ \{ m_i \} \} l_3 \} l_2 \} l_1 \{ o_3 \} o_2 \} o_1 \]
Phase 3: Anonymization (2)

Input to member 1:
encrypted messages $m''_i$
Phase 3: Anonymization (2)

$m''_i = \{ \{ \{ \}$ $m'_i = \{ \{ \{ m_i \}$ $\}l_3 \}l_2 \}l_1$ $\}O_3 \}O_2 \}O_1$

Input to member 1:
encrypted messages $m''_i$

Node 1:
Decrypt, Permute
Phase 3: Anonymization (2)

\[ m''_i = \{ \{ \{ m_1 \} \} \} \]
\[ m'_i = \{ \{ \{ m_i \} \} \{ I_3 \} \{ I_2 \} \{ I_1 \} \{ O_3 \} \{ O_2 \} \{ O_1 \} \]  

Input to member 1: encrypted messages \( m''_i \)

Node 1: Decrypt, Permute

\[ \{\{\{ m_3 \}\}\}\] \]
\[ \{\{\{ m_1 \}\}\}\] \]
\[ \{\{\{ m_2 \}\}\}\] \]
Phase 3: Anonymization (2)

\[ m''_i = \{ \{ \{ \quad m'_i = \{ \{ \quad m_i \quad }_{l_3} \quad }_{l_2} \quad }_{l_1} \quad }_{O_3} \quad }_{O_2} \quad }_{O_1} \]

**Input to member 1:**
encrypted messages \( m''_i \)

- Node 1: Decrypt, Permute
- Node 2: Decrypt, Permute
Phase 3: Anonymization (2)

\[ m''_i = \{ \{ \{ \} \} \} \]
\[ m'_i = \{ \{ \} \} \]
\[ \{ I_3 \} \{ I_2 \} \{ I_1 \} \]
\[ \{ O_3 \} \{ O_2 \} \{ O_1 \} \]

Input to member 1:
encrypted messages \( m''_i \)

Node 1: Decrypt, Permute

Node 2: Decrypt, Permute

Input to member 1:
encrypted messages \( m''_i \)
Phase 3: Anonymization (2)

\[ m''_i = \{ \{ \{ m_1 \} \} \} \quad m'_i = \{ \{ m_i \} \} \quad I_3 \quad I_2 \quad I_1 \quad O_3 \quad O_2 \quad O_1 \]

**Input to member 1:**
encrypted messages \( m''_i \)

**Node 1:** Decrypt, Permute

**Node 2:** Decrypt, Permute

**Node 3:** Decrypt, Permute
Phase 3: Anonymization (2)

Input to member 1: encrypted messages $m''_i$

Node 1: Decrypt, Permute

Node 2: Decrypt, Permute

Node 3: Decrypt, Permute
Phase 3: Anonymization (2)

Output from member n:
partly decrypted messages $m'_i$
in random, secret order

Node 1: Decrypt, Permute

Node 2: Decrypt, Permute

Node 3: Decrypt, Permute
Phase 4: Validation

After the anonymization phase, no member knows the final permutation, but every member \( i \) should see his own \( m'_i \) in the list!

Each member \( i \) looks for \( m'_i \) in the permuted list.

- **Present** \( \rightarrow \) member \( i \) broadcasts “GO”.
- **Absent** \( \rightarrow \) member \( i \) broadcasts “NO-GO” and destroys his inner decryption key \( l'_i \).
Phase 5: Decryption or Blame

• Each member $i$ collects all GO/NO-GO messages.

• **GO messages from all nodes (including self):**
  - Each member $i$ broadcasts his own inner decryption key $I'_i$.
  - All members use keys $I'_1, \ldots, I'_N$ to decrypt all the $m'_j$, revealing all the cleartext messages $m_j$.

• **NO-GO message from any node:**
  - Each member $i$ broadcasts the proof that he decrypted and permuted properly in Phase 3.
  - All members use these proofs to expose disruptor(s).
How DISSENT Provides Accountability

• Any NO-GO message obliges *all* members to “prove their innocence,” i.e., that they:
  o correctly encrypted messages in Phase 2
  o correctly decrypted/permutated in Phase 3
  o correctly validated the final list in Phase 4

• This process reveals the “secret” permutation but leaves the permuted cleartexts $m_j$ undecipherable: They are protected by all honest nodes' inner decryption keys, which have not been revealed.
Handling Variable-Length Messages

• Anonymous-shuffle protocols pad all messages to a common length in order to resist traffic analysis.

• What if the message load is unbalanced, e.g.:
  - Member $i$ wants to send an $L=646$MB video.
  - Members $j \neq i$ have nothing to send in this run of the protocol.

• The group must shuffle the video and $N-1$ 646MB padded cleartexts, resulting in $O(NL)$ bits per node and $O(N^2L)$ bits total.
Basic “Bulk Send” variant

- Use the (slow) accountable-shuffle protocol to exchange randomly permuted metadata.
- Interpret the random permutation as a “schedule” for exchange of data, which is done using DC-nets.
- Accountability of the DISSENT shuffle allows each group member to verify that all members transmitted the correct data in the proper DC-nets “timeslot.”
- Cost of the case in which just one member wants to send $L=646\text{MB}$ drops to $O(L)$ bits per node and $O(NL)$ bits total.
Basic Bulk Send (1)

Shuffle **metadata** describing the messages that the nodes want to send.

<table>
<thead>
<tr>
<th>Message Length</th>
<th>L₁</th>
<th>L₂</th>
<th>L₃</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRNG Seeds</td>
<td>{ − }₁</td>
<td>{S₂→₁}₁</td>
<td>{S₃→₁}₁</td>
</tr>
<tr>
<td></td>
<td>{S₁→₂}₂</td>
<td>{ − }₂</td>
<td>{S₃→₂}₂</td>
</tr>
<tr>
<td></td>
<td>{S₁→₃}₃</td>
<td>{S₂→₃}₃</td>
<td>{ − }₃</td>
</tr>
</tbody>
</table>
Basic Bulk Send (1)

**Shuffle metadata** describing the messages that the nodes want to send.

<table>
<thead>
<tr>
<th>Message Length</th>
<th>PRNG Seeds</th>
<th>Permutated Message Descriptors</th>
</tr>
</thead>
<tbody>
<tr>
<td>( L_1 )</td>
<td>( { S_{1\rightarrow2} }_2 )</td>
<td>( { S_{3\rightarrow1} }_1 )</td>
</tr>
<tr>
<td>( { - }_1 )</td>
<td>( { S_{1\rightarrow3} }_3 )</td>
<td>( { S_{3\rightarrow2} }_2 )</td>
</tr>
<tr>
<td>( L_2 )</td>
<td>( { S_{2\rightarrow1} }_1 )</td>
<td>( { S_{2\rightarrow3} }_3 )</td>
</tr>
<tr>
<td>( { - }_2 )</td>
<td>( { S_{2\rightarrow3} }_3 )</td>
<td>( { - }_3 )</td>
</tr>
<tr>
<td>( L_3 )</td>
<td>( { S_{3\rightarrow1} }_1 )</td>
<td>( { S_{2\rightarrow1} }_1 )</td>
</tr>
<tr>
<td>( { S_{3\rightarrow2} }_2 )</td>
<td>( { S_{1\rightarrow2} }_2 )</td>
<td>( { S_{3\rightarrow2} }_2 )</td>
</tr>
<tr>
<td>( { - }_3 )</td>
<td>( { S_{1\rightarrow3} }_3 )</td>
<td>( { S_{2\rightarrow3} }_3 )</td>
</tr>
</tbody>
</table>
Basic Bulk Send (2)

The shuffled message descriptors form a schedule for a DC-nets transmission.

Permuted Message Descriptors

<table>
<thead>
<tr>
<th>L₃</th>
<th>L₁</th>
<th>L₂</th>
</tr>
</thead>
<tbody>
<tr>
<td>{S₃→1}₁</td>
<td>{ – }₁</td>
<td>{S₂→1}₁</td>
</tr>
<tr>
<td>{S₃→2}₂</td>
<td>{S₁→2}₂</td>
<td>{ – }₂</td>
</tr>
<tr>
<td>{ – }₃</td>
<td>{S₁→₃}₃</td>
<td>{S₂→₃}₃</td>
</tr>
</tbody>
</table>
Basic Bulk Send (2)

The shuffled message descriptors form a **schedule** for a DC-nets transmission.

<table>
<thead>
<tr>
<th>Node 1 →</th>
<th>R($S_{3→1}$)</th>
<th>$M_1 \oplus ...$</th>
<th>R($S_{2→1}$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Node 2 →</td>
<td>R($S_{3→2}$)</td>
<td>R($S_{1→2}$)</td>
<td>$M_2 \oplus R($S_{2→1}$) \oplus ...$</td>
</tr>
<tr>
<td>Node 3 →</td>
<td>$M_3 \oplus R($S_{3→1}$) \oplus ...$</td>
<td>R($S_{1→3}$)</td>
<td>R($S_{2→3}$)</td>
</tr>
</tbody>
</table>

Permuted Message Descriptors

- **L3**
  - \{$S_{3→1}$\}_1
  - \{$S_{3→2}$\}_2
  - \{−\}_3

- **L1**
  - \{−\}_1
  - \{S_{1→2}\}_2
  - \{S_{1→3}\}_3

- **L2**
  - \{S_{2→1}\}_1
  - \{−\}_2
  - \{S_{2→3}\}_3
Outline

• Prior work on anonymous communication

• Basic DISSENT protocol (ACM CCS 2010)

• Results to date
Results to Date (1)

• Reduced latency
  o Multiple bulk sends per shuffle
• Increased scalability (OSDI 2012)
  o Groups with 5000+ members
  o $N$ clients, $M$ servers
  o Secure against both active disruption by up to $N-2$ clients and liveness attacks by a (tunable) constant fraction of clients. This enables ``churn tolerance.''
  o Secure against active disruption by up to $M-1$ servers (but not against liveness attacks by servers).
Results to Date (2)

• Applications
  o “Anonymity scavenging” for wide-area microblogging
  o WiNon: DISSENT-based Web Browsing
    ✓ “Strong, small” anonymity sets instead of the “large, weak” sets offered by Tor-based browsing tools
  o WiNon + Tor
    ✓ Diverse, wide-area anonymity against weak attacker
    ✓ Local-area anon./deniability if attacker can defeat Tor

• Formal proofs that basic DISSENT satisfies
  o Integrity
  o Anonymity
  o Accountability
Ongoing and Future Work

• Protection against ``intersection attacks’’
• Protection against liveness attacks on servers
• Formal security proofs for enhanced DISSENT protocols
• Integration with other anonymity protocols