Dissent:
Group Anonymity

Presented at ACM CCS 2010
By: Ford et. Al.

By: Kritarth Anand
Arunim Samat
“Wikileaks” Problem

Alice

Bob

Eve

Chris

Image courtesy NASA Johnson Space Center
Alice

Bob

Eve

Chris

Has a classified military video from Iraq
Is this video authentic?
Wants to:
1. Anonymously publish video to the group
2. Solicit anonymous comments
Wants to:
1. Break anonymity
2. Stop initial video publication
Why do we want anonymity online?

Many motivations:
• Discuss sensitive/controversial topics safely; protect freedom of speech
• Citizens of authoritarian states evading repression
• Voting in elections or deliberative organizations
• Collaborative content creation/editing, e.g., Wikipedia
• Protect secrecy of bids in commercial auctions
• Law-enforcement “tip” or whistleblowing hotlines
• Peer review processes for research, journalism

*Dedis Group, Yale university*
Proposal 1: Onion Routing

MIX networks, onion routing systems: e.g., Tor
- Tunnel through a series of anonymizing relays
- Protects even if any one is malicious or hacked

*Dedis Group, Yale university*
Limitations of Onion Routing

Vulnerable to traffic analysis, correlation attacks

- compromised, colluding first & last hop relays

- compromised entry relay and server

- first & last links cross same monitored network path

*Dedis Group, Yale university*
Proposal 2: Dining Cryptographers (DC-nets)

• “Alice+Bob” sends a 1-bit secret to Charlie.
Dining Cryptographers (DC-nets)

Another fundamental Chaum invention from the 80s...

• Ex. 2: Homogeneous 3-member group anonymity

* Dedis Group, Yale university
Dining Cryptographers (DC-nets)

Tantalizingly strong anonymity guarantees:
• Unconditional information-theoretic anonymity
• Optimal security against traffic analysis & collusion

Never successfully used in practical systems:
• No provision for accountability or proportionality
  – Malicious member can jam by sending random bits
• Not readily scalable to large groups
  – Especially with node failure, network churn

*Dedis Group, Yale university*
Dissent Model
Dissent Network Model

Quasi-Client/Server Model:

- **Client** nodes represent group members (users) wishing to post messages anonymously
- **Server** nodes are intermediaries that facilitate anonymous group communication

*Dedis Group, Yale university*
Dissent Network Model

Dissent “servers” could actually be:
• Dedicated or volunteer servers, like Tor relays
• Super-peers chosen from clients, P2P-style
• Cloud-based virtual services run professionally

*Dedis Group, Yale university*
Dissent Algorithm-Client

1. Client $i$ calculates his pseudonym key, $pk_{\pi_i}$ and determines his slot location, $\pi_i$, within the DC-net exchange.

2. Client $i$ computes $M$ pseudo-random strings using the shared secrets, $K_{i,j}$, that client $i$ shares with each server $j$. Client $i$ XORs these strings together to produce $c_i$. The client then XORs his cleartext message, $m_i$, into his slot $\pi_i$ within the ciphertext $c_i$, to produce his ciphertext, $c'_i = c_i \oplus (0 \ldots m_i \ldots 0)$. Client $i$ then transmits a signed copy of $c'_i$ to one or more servers.

3. Clients wait to receive a message from their server that contains the exchange’s cleartext signed by all servers: $\tilde{m}, \text{sig}_1, \ldots, \text{sig}_M$. Upon receiving the message, the client repeats from step 2.

*Dedis Group, Yale university*
1. Servers collect clients’ ciphertexts $c'$ until a deadline has passed, at which point the server will no longer accept new ciphertexts $c'$ for this exchange.

2. Each server $j$ shares with every other server a list of clients, $\overrightarrow{l}_j$, who submitted ciphertexts in this exchange.

3. Upon receiving all $\overrightarrow{l}$, each server $j$ computes pseudo-random strings, $s_{i,j}$, using their shared secret with each online client $i$. Each server $j$ XORs theses strings together along with the ciphertexts $c'$ they received, to produce a server ciphertext: $s_j = s_{0,j} \oplus \cdots \oplus s_{N,j}$. The server then calculates a commit to this ciphertext, $commit_j = HASH(s_j)$, and shares the commit with the other servers.

4. Upon receiving all other servers’ $commit$, server $j$ will share $s_j$ with the other servers.

5. Each server verifies that $commit_j = HASH(s_j)$, computes the cleartext message, $\overrightarrow{m} = s_1 \oplus \cdots \oplus s_M$ and signs $\overrightarrow{m}$ transmitting the signature, $sig_j$, to other servers.

6. Servers accumulate all $M$ $sig$ and distribute $\overrightarrow{m}$, $sig_1$, $\ldots$, $sig_M$ to the online clients. The protocol repeats.
Ex: $N=3$ clients, $M=2$ servers

*Dedis Group, Yale university*
Our Project

Provide Anonymous Communication on Facebook, Integrating dissent on Facebook.

*Dedis Group, Yale university*
Facebook: Dissent Network Model

*Dedis Group, Yale university*
Thank You