CPSC 467b: Cryptography and Computer Security

Instructor: Michael Fischer
Lecture by Ewa Syta

Lecture 25
April 18, 2012

Anonymous Communication

Anonymity[1]

Anonymity is the state of being not identifiable within a set of subjects, the anonymity set.

Anonymity set is the set of all possible participants in a system that could have been the sender or recipient of a particular message.

Sender anonymity is the state of being not identifiable within the sender anonymity set as the sender of a particular message.

Receiver anonymity is the state of being not identifiable within the receiver anonymity set as the receiver of a particular message.

[1]A. Pfitzmann and M. Hansen, *Anonymity, unobservability, and pseudonymity: A consolidated proposal for terminology*, 2008

Goal of anonymity systems[1]

In general, anonymity systems seek to provide *unlinkability* between sent messages and their true recipients (receiver anonymity), and between received messages and their true senders (sender anonymity).

Benefits of anonymous communication[2]

- Investigative journalism
- Whistleblowing
- Law enforcement
- Self-help
- Personal privacy protection
- Avoiding persecution

[2] R. Kling, Y-C. Lee, and A. Teich, *Assessing Anonymous Communication on the Internet: Policy Deliberations*, Journal of Information Society, 1999

Harms of anonymous communication[2]

- Spamming
- Deception
- Hate mail
- Impersonation and misrepresentation
- Online financial fraud
- Other illegal activities

Approaches to anonymous communication

There are two main approach to anonymous communication:

- DC-nets
- Mix networks

Dining cryptographers problem[3]

Three cryptographers are sitting down to dinner at their favorite three-star restaurant. The waiter informs them that the meal has been paid anonymously by one of the cryptographers or the National Security Agency. The cryptographers respect each other's right to make an anonymous payment, but they wonder if the NSA paid.

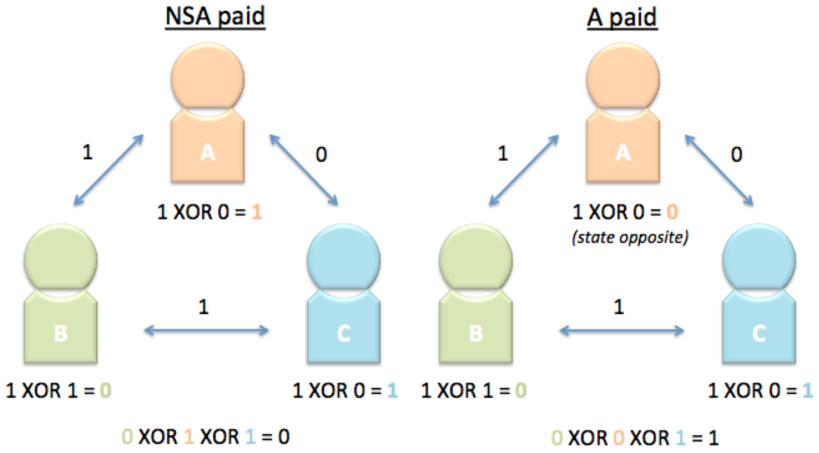
They find out by executing a two-stage protocol.

[3] D. Chaum, *Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms*, Communications of the ACM, 1981

DC-nets

1. Establish a secret.
   - Each cryptographer secretly flips an unbiased coin with the cryptographer on his right.
2. Reveal the secret.
   - Each cryptographer says whether the two coins he sees fell on the same side or on different sides.
   - If one of the cryptographers is the payer, he states the opposite of what he sees.

If the result is 0, then NSA paid. If it is 1, then one of the cryptographer paid. However, cryptographers who did not pay do not know which one did.


NSA paid
A
B
C
1
0
1
1 XOR 0 = 1
1 XOR 1 = 0
1 XOR 0 = 1
0 XOR 1 XOR 1 = 0
A paid
A
B
C
1
0
1
1 XOR 0 = 0
(state opposite)
1 XOR 1 = 0
1 XOR 0 = 1
0 XOR 0 XOR 1 = 1

DC-nets

The protocol is simple, elegant and unconditionally secure (assuming a secure channel) if carried out faithfully.

However, there are two major issues:

- Collisions – if two cryptographers paid for the dinner, their messages will cancel each other out. It means that only one participant can transmit at the same time.
- Disruptions - the last cryptographer can change the final result.

Herbivore[4]

Herbivore is a distributed anonymous communication system, providing private file sharing and messaging over the Internet.

It lets people anonymously publish and retrieve documents, and guarantees that even the most resourceful adversary cannot compromise this anonymity.

Built to be self-organizing, Herbivore relies on neither central servers nor trusted parties, and ultimately provides anonymity by drawing on its community of users.

[4] http://www.cs.cornell.edu/people/egs/herbivore/

Mix-networks

A *mix* is a process that accepts encrypted messages as input, groups several messages together into a batch, and then decrypts and forwards some or all of the messages in the batch.

A mix network consists of mix routers. Mix routers forward the message along the specified path. Some mix routers may intentionally delay sending messages, send them in batches of specified size, etc.

The goal is to obscure the associations between incoming and outgoing messages.

Onion routing[5]

Onion routing is based on the idea of mix networks.

A message is iteratively wrapped with layers of encryption to from "an onion" that specifies properties of the connection along the route.

Each onion router along the route uses its public key to decrypt the entire onion. It exposes the identity of the next onion router, cryptographic information used, and the embedded onion. The router pads the onions and forwards to the next hop.

The last node in the chain removes the remaining layer of encryption and forwards the message to its destination.

[5] http://www.onion-router.net/

Tor[6]

Tor is a low latency anonymity network based on onion routing. It consists of a number of relays constantly encrypting and then randomly bouncing messages.

The random path packets take is extended one hop at a time, and each relay along the way knows only the previous and the next relay. A separate set of encryption keys is negotiated for each hop along the circuit.

Once a circuit has been established, it is used to exchanged different kinds of data. For efficiency, Tor uses the same circuit for connections that happen within the same ten minutes or so.

[6] The Tor Project `https://www.torproject.org/`

Tor

The goal is to protect users' from network surveillance.

No individual relay ever knows the complete path.

Neither an eavesdropper nor a compromised relay can link the connection's source and destination.

However, Tor cannot (and does not attempt to) protect the traffic entering and exiting the network.

Limitations of existing schemes

Method	Weakness
Mix Nets, Tor	Traffic analysis attacks
Group and Ring Signatures	Traffic analysis attacks
Voting Protocols	Short, fixed-length messages
DC Nets	Anonymous DoS attacks

DISSENT- Accountable Anonymous Group Communication

Dissent[8]

DISSENT stands for Dining cryptographers Shuffled Send Network. It provides accountability by adding a *blame* phase.

DISSENT is suitable for latency-tolerant applications because its scalability is limited.[7]

DISSENT consists of two sub-protocols: a *shuffle* protocol and a *bulk* protocol.

[7] The version presented here. There are other scalable versions of DISSENT.

[8] H. Corrigan-Gibbs and B. Ford, Dissent: Accountable Group Anonymity, 17th ACM Conference on Computer and Communications Security (CCS 2010)

Security Goals

Anonymity - a group of $k \leq N - 2$ colluding members cannot learn the associations between honest members and their secret messages with a probability significantly better than random guessing.

Integrity - upon completion of the protocol all members know that either the protocol successfully completed or the protocol failed.

Accountability - no honest member is ever exposed and upon protocol failure at least one dishonest member is exposed.

The Shuffle protocol

The shuffle protocol builds on a data mining protocol by Brickell and Shmatikov.[9]

DISSENT adds protection against DoS attacks by malicious group members by adding *go/no-go* and *blame* phases.

[9] J. Brickell and V. Shmatikov, *Efficient Anonymity-Preserving Data Collection*, KDD 2006

Possible outcomes

Each protocol run will either succeed or fail.

1. Success.
   - All members faithfully follow the protocol.
   - Secret messages are delivered.
   - Secret permutation are unrecoverable.
2. Failure
   - Some member(s) deviates from the protocol.
   - Messages are unrecoverable.
   - At least one attacker exposed.

The Shuffle protocol

The shuffle protocol operates in phases. Each member sends at most one unique message per round.

Phase 1. Inner and Outer Key Pairs Generation

Phase 2. Data Submission

Phase 3. Anonymization

Phase 4. Verification

Phase 5a. Decryption

Phase 5b. Blame.

Initial setup

Each member:

- has a signing key pair (u_i, v_i),
- a secret message m_i of fixed length L to send anonymously,
- maintains a *tamper-evident log*[10] of all messages it sends and receives.

All members:

- agree on a session nonce n_R uniquely a protocol run,
- exchange their signing keys,
- agree upon a common ordering of members.

[10] A. Haeberlen, P. Kouznetsov, and P. Druschel, *PeerReview: Practical Accountability for Distributed Systems*, SOSP 2007

Phase 1. Key generation

Each member i chooses two ephemeral encryption key pairs called inner (I_i^{sec}, I_i^{pub}) and outer (O_i^{sec}, O_i^{pub}) and shares with other group members.

Each message contains a hash of the current log and is signed with the signing key u.

Phase 2. Data Submission

Each member i iteratively encrypts its message m_i using all inner public keys to create the inner ciphertext C'_i.

Then, C'_i is further encrypted with all outer keys creating the outer ciphertext C_i.

If encryption fails at any point, member i sets an internal flag GO_i to false for accountability purposes.

Phase 3. Anonymization

Member 1, the group leader:

- collects all ciphertexts,
- randomly permutes its elements,
- strips one layer of encryption from each ciphertext using her private key O_1^s,
- forwards to the next member who repeats the process.

If any member detects a duplicate or invalid ciphertext, they set $\mathrm{GO}_i =$ false.

Removing all layers of outer encryption yields a set of inner ciphertext messages, $\vec{C}'_{1,\ldots,N}$, which is broadcasted.

Phase 4. Verification

At this point, all members hold $\vec{C'}_{1,\ldots,N}$, which is a randomized set of their inner ciphertexts.

Each member inspects $\vec{C'}_{1,\ldots,N}$ to verify that their inner ciphertext is present. If not, set $\textsc{go} =$ false.

Now, each member i creates a “go/no-go” message which consists of a hash of all messages it sent or received in prior phases $\textsc{hash}\{\vec{B}\}$ and $\textsc{go}_i$.

The group proceeds to phase 5a only if every member reported $\textsc{go} =$ true for the expected $\textsc{hash}\{\vec{B}\}$. Otherwise the group enters phase 5b.

Phase 5a. Decryption

Each member broadcasts her inner private key I_i^{sec} to all members.

Member i verifies that each I_j^{sec} matches I_j^{pub}. If not, i exposes j using the signed message from phase 1.

Otherwise, i removes the remaining N levels of encryption from $\vec{C}_N$, resulting in a permutation of the submitted data $m_1, \ldots, m_N$, and the protocol completes successfully.

Phase 5b. Blame

Each member destroys her inner private key I_i^{sec}, and then reveals her outer private key O_i^{sec}, inner ciphertext C_i' and all signed messages she received and sent in phases 1–4.

Each member i uses this information to check the behavior of each member j in phases 1–4.

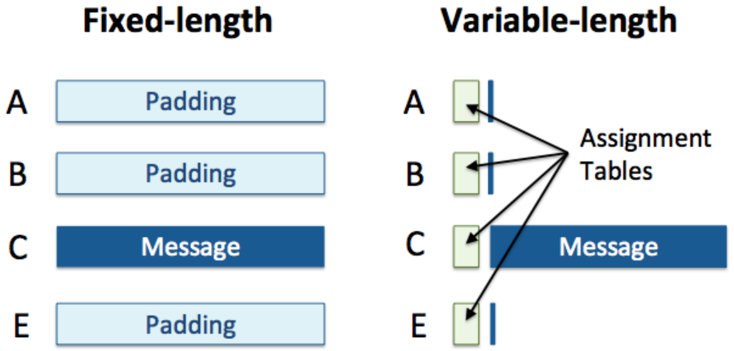
Shuffle performance

The shuffle protocol provides the desired security properties, but is inefficient.

Members are required to send messages of equal length (Why?) and always send a message even if they have nothing to say. (Why?)

The Bulk protocol allows to send variable length messages in a more efficient manner.

Variable-length is better



Shuffle: NL_{max} bits versus Bulk: $L_{total} + kN$ bits

The Bulk Protocol

The bulk protocol builds on DC-nets and uses the shuffle in place of the DoS-prone slot reservation systems to to prearrange the DC-nets transmissions.

A set of a set of message descriptors submitted by each member is anonymously shuffled. The shuffled order of the message descriptors indicates the order in which the anonymous senders should transmit their secret messages.

Such approach guarantees each member exactly one message slot per round.

The Bulk Protocol

Phase 0. Session Key Pair Generation.

Phase 1. Message Descriptor Generation

Phase 2. Message Descriptor Shuffle

Phase 3. Data Transmission

Phase 4. Message Recovery

Phase 5. Blame

Phase 0. Key Generation

Each member i chooses an ephemeral encryption key pair (x_i, y_i) and broadcasts to other members.

Phase 1. Message Descriptor Generation

Each member i chooses a random seed s_{ij} for each member j, then for each $j \neq i$, $C_{ij} = \text{PRNG}\{L_i, s_{ij}\}$.

Member i now XORs her message m_i with each C_{ij} for $j \neq i$:

$$C_{ii} = C_{i1} \oplus \ldots \oplus C_{i(i-1)} \oplus m_i \oplus C_{i(i+1)} \oplus \ldots \oplus C_{iN}$$

Member i forms a *message descriptor* $d_i = \{L_i, \text{HASH}\{m_i\}, \vec{H}_i, \vec{S}_i\}$, where:

- L_i – the length of m_i,
- $\text{HASH}\{m_i\}$ – hash of m_i,
- $\vec{H}_i$ – a vector of hashes of C_{ij},
- $\vec{S}_i$ – a vector of encrypted seeds.

Phase 2. Message Descriptor Shuffle

The group runs the shuffle protocol.

Each member i submits its fixed-length descriptor d_i as the secret message.

The shuffle protocol broadcasts all descriptors in some random permutation π to all members, so d_i appears at position $\pi(i)$ in the shuffle output.

Phase 3. Data Transmission

Each member j now recognizes its own descriptor d_j in the shuffle, and sets $C'_{jj} = C_{jj}$.

From other descriptors j:

- decrypts S_{ij} with private key x_j to reveal seed s_{ij},
- computes ciphertext $C_{ij} = \text{PRNG}\{L_i, s_{ij}\}$,
- checks $\text{HASH}\{C_{ij}\}$ against H_{ij}.
    - if ok, j sets $C'_{ij} = C_{ij}$,
    - if not, j sets C'_{ij} to an empty ciphertext $C_{ij} = \{\}$.

Member j sends each C'_{ij} to the target in π-shuffled order.

Phase 4. Message Recovery

The designated target checks each C'_{ij} it receives from member j against H_{ij} from descriptor d_i.

If C'_{ij} is empty or $\textsc{hash}\{C'_{ij}\} \neq H_{ij}$, then message slot $\pi(i)$ was corrupted and the target ignores it.

For each uncorrupted slot $\pi(i)$, the target recovers i's message by computing:

$$m_i = C'_{i1} \oplus \ldots \oplus C'_{iN}$$

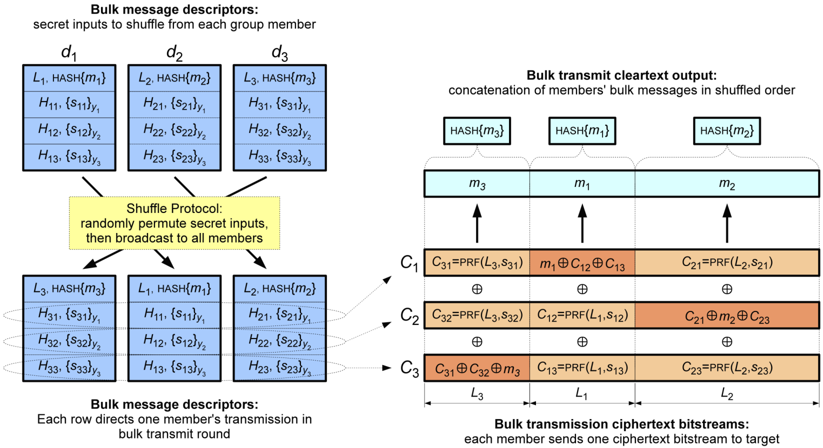
Phase 5. Blame

All members run the shuffle protocol again and member i whose message was corrupted sends an *accusation* $A_i = \{j, S_{ij}, s_{ij}, R_{ij}\}$, where

- j – the faulty member,
- S_{ij} – the encrypted seed,
- s_{ij} – the seed i assigned to j,
- R_{ij} – random bits used for encryption.

Other members verify that the accusation is valid by replying encryption of the seed, generating the pseudo-random sequence and verifying its hash.

It the accusation is valid, then the group exposes j as faulty.


Bulk message descriptors:
secret inputs to shuffle from each group member
d_1
d_2
d_3
L_1, HASH$\{m_1\}$
$H_{11}, \{s_{11}\}_{y_1}$
$H_{12}, \{s_{12}\}_{y_2}$
$H_{13}, \{s_{13}\}_{y_3}$
L_2, HASH$\{m_2\}$
$H_{21}, \{s_{21}\}_{y_1}$
$H_{22}, \{s_{22}\}_{y_2}$
$H_{23}, \{s_{23}\}_{y_3}$
L_3, HASH$\{m_3\}$
$H_{31}, \{s_{31}\}_{y_1}$
$H_{32}, \{s_{32}\}_{y_2}$
$H_{33}, \{s_{33}\}_{y_3}$
Shuffle Protocol:
randomly permute secret inputs,
then broadcast to all members
Bulk message descriptors:
Each row directs one member's transmission in bulk transmit round
Bulk transmit cleartext output:
concatenation of members' bulk messages in shuffled order
HASH$\{m_3\}$
HASH$\{m_1\}$
HASH$\{m_2\}$
m_3
m_1
m_2
C_1
C_{31}=PRF(L_3,s_{31})
$m_1 \oplus C_{12} \oplus C_{13}$
C_{21}=PRF(L_2,s_{21})
C_2
C_{32}=PRF(L_3,s_{32})
C_{12}=PRF(L_1,s_{12})
$C_{21} \oplus m_2 \oplus C_{23}$
C_3
$C_{31} \oplus C_{32} \oplus m_3$
C_{13}=PRF(L_1,s_{13})
C_{23}=PRF(L_2,s_{23})
L_3
L_1
L_2
Bulk transmission ciphertext bitstreams:
each member sends one ciphertext bitstream to target

Performance

A DISSENT prototype was tested under Emulab on groups of 44 nodes connected via simulated wide-area links.

- It incurs 1.4-minute latency when distributing messages up to 16MB among 16 nodes with 100mn inter-node delays.
- It handles large message loads, both balanced and unbalanced, in about $3.5\times$ the time required for non-anonymized communication.

A 1MB message can be sent anonymously in:

- less than 1 minute in a 4-node group
- 4 minutes in a 20-node group
- 14 minutes in a 40-node group

Additional Resources

- The Tor Project, `https://www.torproject.org/`