BTI 4202: Anonymity

Christian Grothoff

Berner Fachhochschule

26.5.2023
Learning Objectives

Introduction to Anonymity
Basic Designs for Anonymizing Systems
Tor
Distributed Systems Theory
Fallacies of distributed computing
Boyd’s theorem
CAP Theorem
Zooko’s Triangle
Self stabilization
Distributed Systems & Security
Secure Multiparty Computation
Part I: Introduction to Anonymity
Suppose Alice and Bob communicate using encryption.

What can Eve still learn here?
Motivation

Suppose Alice and Bob communicate using encryption. What can Eve still learn here?

Eve cannot read the data Alice and Bob are sending, but:

- Eve knows that Alice and Bob are communicating.
- Eve knows the amount of data they are sending and can observe patterns.

⇒ Patterns may even allow Eve to figure out the data
“We present a traffic analysis attack against over 6000 webpages spanning the HTTPS deployments of 10 widely used, industry-leading websites in areas such as healthcare, finance, legal services and streaming video. Our attack identifies individual pages in the same website with 89% accuracy, exposing personal details including medical conditions, financial and legal affairs and sexual orientation. We examine evaluation methodology and reveal accuracy variations as large as 18% caused by assumptions affecting caching and cookies.” [8]
Anonymity Definitions

Merriam-Webster:

1. not named or identified: “an anonymous author”, “they wish to remain anonymous”
2. of unknown authorship or origin: “an anonymous tip”
3. lacking individuality, distinction, or recognizability: “the anonymous faces in the crowd”, “the gray anonymous streets”
   – William Styron
Anonymity Definitions

Andreas Pfitzmann et. al.:

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

Andreas Pfitzmann et. al.:

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

EFF:

“Instead of using their true names to communicate, (...) people choose to speak using pseudonyms (assumed names) or anonymously (no name at all).”
Anonymity Definitions

Andreas Pfitzmann et. al.:

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

EFF:

“Instead of using their true names to communicate, (...) people choose to speak using pseudonyms (assumed names) or anonymously (no name at all).”

Mine:

A user’s action is anonymous if the adversary cannot link the action to the user’s identity
The user’s identity

includes personally identifiable information, such as:

- real name
- fingerprint
- passport number
- IP address
- MAC address
- login name
- ...

...
Actions

include:

- Internet access
- speech
- participation in demonstration
- purchase in a store
- walking across the street
- ...

...
Anonymity: Terminology

- **Sender Anonymity:** The initiator of a message is anonymous. However, there may be a path back to the initiator.

- **Receiver Anonymity:** The receiver of a message is anonymous.
Pseudonymity

"Nurse"  Bla, bla, bla, bla, bla, ...

"Viking"  Bla, bla, bla, bla, bla, ...

Eve  "OK, "Nurse" and "Viking" seem to communicate. But who are they? Could be any of them.

Fred
Bob
Alan
Gary
Bob
Alice
Pseudonymity

- A pseudonym is an identity for an entity in the system. It is a “false identity” and not the true identity of the holder of the pseudonym.
- Nobody, but (maybe) a trusted party may be able to link a pseudonym to the true identity of the holder of the pseudonym.
- A pseudonym can be tracked. We can observe its behaviour, but we do not learn who it is.
Evaluating Anonymity

How much anonymity does a given system provide?

- Number of known attacks?
- Lowest complexity of successful attacks?
- Information leaked through messages and maintenance procedures?
- Number of users?
Anonymity: Basics

- **Anonymity Set** is the set of suspects
- Attacker computes a **probability distribution** describing the likelihood of each participant to be the responsible party.
- Anonymity is the stronger, the larger the anonymity set and the more evenly distributed the subjects within that set are.
Anonymity Metric: Anonymity Set Size

Let $\mathcal{U}$ be the attacker’s probability distribution and $p_u = \mathcal{U}(u)$ describing the probability that user $u \in \Psi$ is responsible.

\[
\text{ASS} := \sum_{u \in \Psi} 1
\quad (1)
\]
Large Anonymity Sets

Examples of large anonymity sets:

- Any human
- Any human with Internet access
- Any human speaking German
- Any human speaking German with Internet access awake at 3am CEST
Examples of large anonymity sets:
- Any human
- Any human with Internet access
Large Anonymity Sets

Examples of large anonymity sets:

- Any human
- Any human with Internet access
- Any human speaking German
- Any human speaking German at 3am CEST
Large Anonymity Sets

Examples of large anonymity sets:

- Any human
- Any human with Internet access
- Any human speaking German
- Any human speaking German with Internet access awake at 3am CEST
Anonymity Metric: Maximum Likelihood

Let $\mathcal{U}$ be the attacker’s probability distribution describing the probability that user $u \in \Psi$ is responsible.

$$ML := \max_{u \in \Psi} p_u$$ (2)
Anonymity Metric: Maximum Likelihood

- For successful criminal prosecution in the US, the law requires $ML$ close to 1 ("beyond reasonable doubt")
- For successful civil prosecution in the US, the law requires $ML > \frac{1}{2}$ ("more likely than not")
- For a given anonymity set, the best anonymity is achieved if

$$ML = \frac{1}{ASS}$$  \hspace{1cm} (3)
Anonymity Metric: Entropy

Let $\mathcal{U}$ be the attacker’s probability distribution describing the probability that user $u \in \Psi$ is responsible. Define the effective size $S$ of the anonymity distribution $\mathcal{U}$ to be:

$$S := - \sum_{u \in \Psi} p_u \log_2 p_u$$

where $p_u = \mathcal{U}(u)$. 

\[
\begin{align*}
\text{-log2}(1/n) & \quad 0 & \quad 2 & \quad 4 & \quad 6 & \quad 8 & \quad 10 \\
0 & \quad 200 & \quad 400 & \quad 600 & \quad 800 & \quad 1000
\end{align*}
\]
Interpretation of Entropy

\[ S = - \sum_{u \in \Psi} p_u \log_2 p_u \tag{5} \]

This is the expected number of bits of additional information that the attacker needs to definitely identify the user (with absolute certainty).
Suppose we have 101 suspects including Bob. Furthermore, suppose for Bob the attacker has a probability of 0.9 and for all the 100 other suspects the probability is 0.001.

What is $S$?
Entropy Calculation Example

Suppose we have 101 suspects including Bob. Furthermore, suppose for Bob the attacker has a probability of 0.9 and for all the 100 other suspects the probability is 0.001.

What is $S$?

- For 101 nodes $H_{max} = 6.7$

$$S = -rac{100 \cdot \log_2 0.001}{1000} - \frac{9 \cdot \log_2 0.9}{10}$$

$$\approx 0.9965 + 0.1368$$

$$= 1.133...$$
Attacks to avoid

Hopeless situations include:

▶ All nodes collaborate against the victim
▶ All directly adjacent nodes collaborate
▶ All non-collaborating adjacent nodes are made unreachable from the victim
▶ The victim is required to prove his innocence
R. Dingledine and P. Syverson wrote about *Open Issues in the Economics of Anonymity*:  

- Providing anonymity services has economic disincentives (DoS, legal liability)  
- Anonymity requires introducing inefficiencies  

⇒ Who pays for that?
Economics & Anonymity

R. Dingledine and P. Syverson wrote about *Open Issues in the Economics of Anonymity*:

- Providing anonymity services has economic disincentives (DoS, legal liability)
- Anonymity requires introducing inefficiencies
  ⇒ Who pays for that?

The anonymizing server that has the best reputation (performance, most traffic) is presumably compromised.
Part II: Anonymizing Systems
Anonymity: Dining Cryptographers

“Three cryptographers are sitting down to dinner. The waiter informs them that the bill will be paid anonymously. One of the cryptographers maybe paying for dinner, or it might be the NSA. The three cryptographers respect each other’s right to make an anonymous payment, but they wonder if the NSA is paying.” – David Chaum
Mixing

David Chaum’s mix (1981) and cascades of mixes are the traditional basis for destroying linkability:
Mixing

David Chaum’s mix (1981) and cascades of mixes are the traditional basis for destroying linkability:
Timed Mix

Buffer $n$

$T_T$ fires

$t$
Pool mix
G. Danezis, R. Dingledine, D. Hopwood and N. Mathewson describe Mixminion [2]:

- based on mixmailers (only application is E-mail)
- possibility to reply
- directory servers to evaluate participating remailers (reputation system)
- exit policies
Mixminion: key ideas

When a message traverses mixminion, each node must decrypt the message using its (ephemeral) private key.

The key idea behind the replies is splitting the path into two legs:
- the first half is chosen by the responder to hide the responder identity
- the second half was communicated by the receiver to hide the receiver identity
- a crossover-node in the middle is used to switch the headers specifying the path
Replay attacks were an issue in previous mixnet implementations.

- Mixes are vulnerable to replay attacks
- Mixminion: servers keep hash of previously processed messages until the server key is rotated

⇒ Bounded amount of state in the server, no possibility for replay attack due to key rotation
Mixminion: Directory Servers

- Inform users about servers
- Probe servers for reliability
- Allow a partitioning attack unless the user always queries all directory servers for everything
Mixminion: Nymservers

- Nymservers keep list of use-once reply blocks for a user
- Vulnerable to DoS attacks (deplete reply blocks)
- Nymservers could also store mail (use one reply block for many messages).
Mixminion: obvious problems

- no benefits for running a mixmailer for the operator
- quite a bit of public key cryptography
- trustworthiness of directory servers questionable
- servers must keep significant (but bounded) amount of state
- limited to E-mail (high latency)
Mixminion: open problems

- exit nodes are fair game for legal actions
- no accounting to defend against abuse / DoS attacks
- statistical correlation of entities communicating over time possible (observe participation)

⇒ bridging between an anonymous network and a traditional protocol is difficult
Break
Part III: Tor
Tor

- Tor is a P2P network of **low-latency** mixes which are used to provide anonymous communication between parties on the Internet.
- Tor works for any TCP-based protocol
- TCP traffic enters the Tor network via a SOCKS proxy
- **Common usage:** client anonymity for web browsing
Onion Routing

- Multiple mix servers
- Path of mix servers chosen by initiator
- Chosen mix servers create “circuit”
  - Initiator contacts first server $S_1$, sets up symmetric key $K_{S_1}$
  - Then asks first server to connect to second server $S_2$; through this connection sets up symmetric key with second server $K_{S_2}$
  - ... 
  - Repeat with server $S_i$ until circuit of desired length $n$ constructed
Onion Routing Example

- Client sets up symmetric key $K_{S_1}$ with server $S_1$
Onion Routing Example

Via $S_1$ Client sets up symmetric key $K_{S_2}$ with server $S_2$
Onion Routing Example

- Client encrypts $m$ as $K_{S_1}(K_{S_2}(m))$ and sends to $S_1$
Onion Routing Example

- $S_1$ decrypts, sends on to $S_2$, $S_2$ decrypts, revealing $m$
Tor - How it Works

- Low latency P2P Network of mix servers
- Designed for interactive traffic (https, ssh, etc.)
- “Directory Servers“ store list of participating servers
  - Contact information, public keys, statistics
  - Directory servers are replicated for security
- Clients choose servers randomly with bias towards high BW/uptime
- Clients build long lived Onion routes ”circuits“ using these servers
- Circuits are bi-directional
- Circuits are of length three
Example of Tor client circuit
Servers are classified into three categories for usability, security and operator preference.

- **Entry nodes (aka guards)** - chosen for first hop in circuit
  - Generally long lived "good" nodes
  - Small set chosen by client which are used for client lifetime (security)

- **Middle nodes** - chosen for second hop in circuit, least restricted set

- **Exit nodes** - last hop in circuit
  - Visible to outside destination
  - Support filtering of outgoing traffic
  - Most vulnerable position of nodes
Hidden Services in Tor

- Hidden services allow Tor servers to receive incoming connections anonymously.
- Can provide access to services available *only* via Tor:
  - Web, IRC, etc.
  - For example, host a website without your ISP knowing.
Hidden Services Example 1

Step 1: Bob picks some introduction points and builds circuits to them.
**Hidden Services Example 2**

Step 2: Bob advertises his hidden service -- XYZ.onion -- at the database.
Hidden Services Example 3

Step 3: Alice hears that XYZ.onion exists, and she requests more info from the database. She also sets up a rendezvous point, though she could have done this before.
Hidden Services Example 4

Step 4: Alice writes a message to Bob (encrypted to PK) listing the rendezvous point and a one-time secret, and asks an introduction point to deliver it to Bob.
Hidden Services Example 5

Step 5: Bob connects to the Alice’s rendezvous point and provides her one-time secret.
Step 6: Bob and Alice proceed to use their Tor circuits like normal.
Types of Attacks on Tor

- Exit Relay Snooping
- Website fingerprinting
- Traffic Analysis
- Intersection Attack
- DoS
Homework

- Install Tor
- Configure Tor relay
- Setup hidden service
- Perform risk analysis for deanonymization
Part IV: Distributed Systems Theory
The 8 Fallacies of Distributed Computing

1. The network is reliable
2. Latency is zero
3. Bandwidth is infinite
4. The network is secure
5. Topology does not change
6. There is one administrator
7. Transport cost is zero
8. The network is homogeneous
Limits on authentication

Theorem (Boyd’s Theorem I)

“Suppose that a user has either a confidentiality channel to her, or an authentication channel from her, at some state of the system. Then in the previous state of the system such a channel must also exist. By an inductive argument, such a channel exists at all previous states.”

Theorem (Boyd’s Theorem II)

“Secure communication between any two users may be established by a sequence of secure key transfers if there is a trusted chain from each one to the other.”
Solution space: Zfone Authentication (ZRTP) [7]

Idea: combine human interaction proof and baby duck approach:
- $A$ and $B$ perform Diffie-Hellman exchange
- Keying material from previous sessions is used (duckling)
- Short Authentication String (SAS) is generated (hash of DH numbers)
- Both users read the SAS to each other, recognize voice

\[ \Rightarrow \] ZRTP foils standard man-in-the-middle attack.
CAP Theorem [4]

No distributed system can be consistent, available and partition tolerant at the same time.

- Consistency: A read sees the changes made by all previous writes
- Availability: Reads and writes always succeed
- Partition tolerance: The system operates even when network connectivity between components is broken
Blockchain Trilemma

Blockchains claim to achieve three properties:

- **Decentralization**: there are many participants, and each participant only needs to have a small amount of resources, say $O(c)$
- **Scalability**: the system scales to $O(n) > O(c)$ transactions
- **Security**: the system is secure against attackers with $O(n)$ resources

The Blockchain trilemma is that one can only have two of the three.
Ryge’s Triangle postulates three key management goals for a system associating cryptographic keys with addresses or names:

- Non-interactive: the system should require no user interface
- Flexible: addresses/names can be re-used by other participants
- Secure: the system is secure against active attackers

Ryge’s triangle says that one can only have two of the three.
A name system can only fulfill two!
Zooko’s Triangle

Secure
Global Memorable Hierarchical Registration
Cryptographic Identifiers
Petname Systems

DNS, “.onion” IDs and /etc/hosts/ are representative designs.
Zooko’s Triangle

Secure

Global
Hierarchical Registration
Memorable

Cryptographic Identifiers
mnemonic URLs
SDSI

Petname Systems

DNSSEC security is limited (adversary model!)
A system is self-stabilizing, if starting from any state, it is guaranteed that the system will eventually reach a correct state (convergence).

Given that the system is in a correct state, it is guaranteed to stay in a correct state, provided that no fault happens (closure).

Self-stabilization enables a distributed algorithm to recover from a transient fault regardless of its nature.

Example: Spanning-tree Protocol from Networking!
Part V: Distributed Systems & Security
Sybil Attack

Background:

- Ancient Greece: Sybils were prophetesses that prophesized under the divine influence of a deity. Note: At the time of prophecy not the person but a god was speaking through the lips of the sybil.

Sybil Attack

Background:

▶ Ancient Greece: Sybils were prophetesses that prophesized under the divine influence of a deity. Note: At the time of prophecy not the person but a god was speaking through the lips of the sybil.


The Sybil Attack [3]:

▶ Insert a node multiple times into a network, each time with a different identity

▶ Position a node for next step on attack:
  ▶ Attack connectivity of the network
  ▶ Attack replica set
  ▶ In case of majority votes, be the majority.
Defenses against Sybil Attacks

- Use authentication with trusted party that limits identity creation
- Use “external” identities (IP address, MAC, e-mail)
- Use “expensive” identities (solve computational puzzles, require payment)

Douceur: Without trusted authority to certify identities, no realistic approach exists to completely stop the Sybil attack.
Eclipse Attack: Goal

- Separate a node or group of nodes from the rest of the network
- Isolate peers (DoS, surveillance) or isolate data (censorship)
Eclipse Attack: Techniques

- Use Sybil attack to increase number of malicious nodes
- Take over routing tables, peer discovery
  ⇒ Details depend on overlay structure
Eclipse Attack: Defenses

- Large number of connections
- Replication
- Diverse neighbour selection (different IP subnets, geographic locations)
- Aggressive discovery ("continuous" bootstrap)
- Audit neighbour behaviour (if possible)
- Prefer long-lived connections / old peers
Poisoning Attacks

Nodes provide false information:

- wrong routing tables
- wrong meta data
- wrong performance measurements
Nodes can:

- measure latency to determine origin of data
- delay messages
- send messages using particular timing patterns to aid correlation
- include wrong timestamps (or just have the wrong time set...)
Part VI: Secure Multiparty Computation
Secure Multiparty Computation (SMC)

- Alice und Bob haben private Daten $a_i$ and $b_i$.
- Alice und Bob führen ein Protokoll aus und berechnen gemeinsam $f(a_i, b_i)$.
- Nur einer von beiden lernt das Ergebnis (i.d.R.)
Adversary models

Honest but curious

Dishonest and curious
Secure Multiparty Computation: Scalar Product

We want to calculate

$$\sum_i a_i b_i$$

(9)

- Original idea by Ioannidis et al. in 2002 [5] (use:
  $$(a - b)^2 = a^2 - 2ab + b^2$$)

- Refined by Amirbekyan et al. in 2007 (corrected math) [1]
Let Alice’s secret value be \( a \in \mathbb{Z} \). Alice sends to Bob \((g_i, h_i) = (g^{r_i}, g^{r_ia+a_i})\) with random values \( r_i \) for \( i \in M \).

Bob answers with:

\[
\left( \prod_{i \in M} g_i^{b_i}, \prod_{i \in M} h_i^{b_i} \right) = \left( \prod_{i \in M} g_i^{b_i}, \left( \prod_{i \in M} g_i^{b_i} \right)^a g^{\sum_{i \in M} a_i b_i} \right)
\]

Alice can then calculate:

\[
\left( \prod_{i \in M} g_i^{b_i} \right)^{-a} \cdot \left( \prod_{i \in M} g_i^{b_i} \right)^a \cdot g^{\sum_{i \in M} a_i b_i} = g^{\sum_{i \in M} a_i b_i}.
\]

Assuming \( \sum_{i \in M} a_i b_i \) is sufficiently small, then Alice can compute the scalar product by solving the DLP.


Brad Miller, Ling Huang, A.D. Joseph, and J.D. Tygar. 
I know why you went to the clinic: Risks and realization of https traffic analysis. 