Anonymous Web Browsing

Software Security

Steffen Helke

Chair of Software Engineering

7th November 2018

Brandenburgische
Technische Universität
Cottbus - Senftenberg

Objectives of today's lecture

- ➜ Repetition – definitions of *anonymity* and classification of *remailers*
- ➜ Understanding the principles of *anonymisation services*
- ➜ Reflecting the *differences between TOR and JAP*
- ➜ Being able to reproduce two *protocols for the most important use cases* of TOR

What exactly do we mean by anonymity?

Definition (given by Pfitzmann)

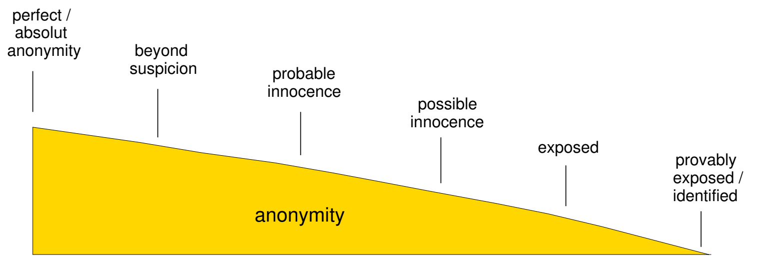
A person in a role R is anonymous relative to an event E and an attacker A, if for every person not cooperating with A, the anonymous person has the role R in E with a probability truly greater than 0 and truly smaller than 1 after every observation from A.

What exactly do we mean by perfect anonymity?

Definition (given by Pfitzmann)

A person in a role R relative to an event E and an attacker A is perfectly anonymous, if for every person not cooperating with A the anonymous person has the role R in E with the same probability before and after an observation from A.

Other Definitions (Degrees) of Anonymity



Source: M. Reiter, A. Rubin: *Crowds: Anonymity for Web Transactions*, 1999.

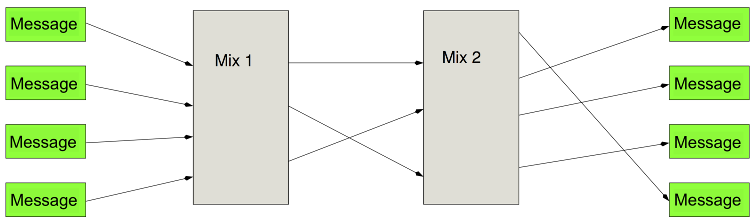
- **Beyond suspicion**
  → no more likely than any other potential sender
- **Probable innocence**
  → no more likely to be the sender than not to be the sender
- **Possible innocence**
  → there is a nontrivial probability that the real sender is someone else

Strategies for Anonymization

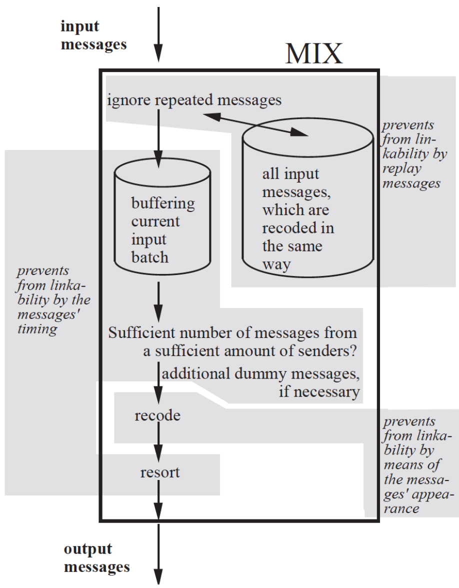
What is a Mix server?

Basic idea of mixing according to [Chaum, 1981]

- Provides unlikability between incoming and outcoming messages
- Mixes collects messages, changes their coding and forward them in different order



Basis Functions of a Mix server?



Source: A. Pfitzmann: Script – *Security in IT-Networks*, 2012

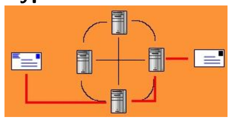
What **types of remailers** do you know?

Classification

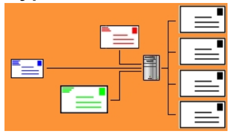
- Pseudonymous remailers (**Type 0**)
- Cypherpunk remailers[1] (**Type I**)
- Mixmaster remailers (**Type II**)
- Mixminion remailers (**Type III**)

Type 0

Type I



Type II



➜ And what else can the **Type III** mailers do?

[1] *Cypherpunk* is an artificial word derived from cipher, cyber and punk

What other anonymization services do exist?

Problem

- Remailers have *too long response* times
- Applications such as web browsing require low latency
- Approach of remailers (use of MIXes as brokers between users and service providers) have to be transferred to other protocols

Software (Selection)

- Anonymization Proxy
- Jondos (formerly JAP, Java Anon Proxy)
- Tor (The Onion Router)

Simple Solution with a Proxy

Idea

- Put a proxy server in between, via which all users must access the Web services
- Servers that offer services only see the IP address of this proxy server

Problem

- Data needed for de-anonymization is located on this proxy server
- Users must blindly trust the proxy server

Better Solution with Jondos/JAP

Idea

- Routing messages over more than one Mix server, communication will be encrypted

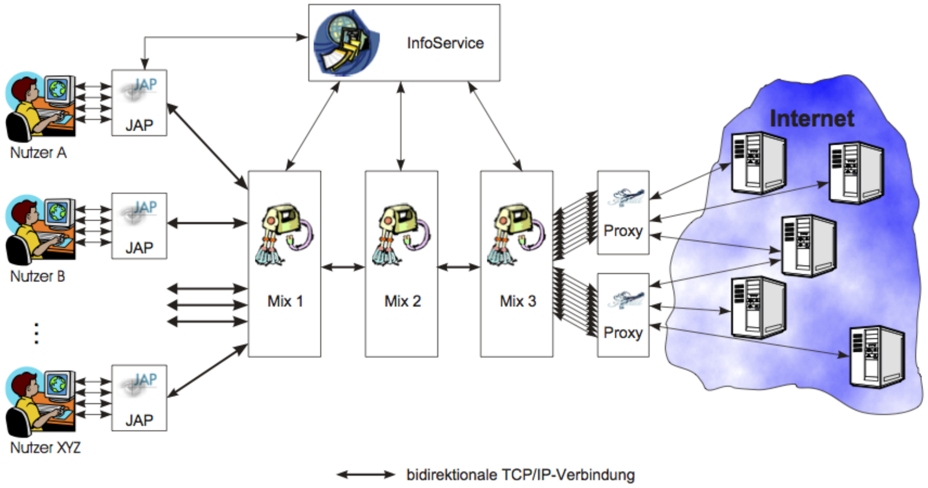
Example with three MIX-es

1. Initiator sends a request to *Mix1* server, *Mix1* can see that the data came from the Initiator
2. Forwarding the data to *Mix2*, *Mix2* can only see that the data came from *Mix1*
3. Forwarding the data to the *Mix3* server, which can only see that the data came from *Mix2*
   ➜ finally *Mix3* sends a request to the web server

Assumption

- Providers of the three MIXes do not work together, e.g. by a self-commitment of the providers

Architecture of Jondos/JAP



Source: S. Köpsell: AnonDienst - Design und Implementierung, 2004, http://anon.inf.tu-dresden.de

Solution with Tor

Idea

- Distributed anonymous network

Properties of Tor

- Mix servers are not only provided by official providers
- *Each person* is authorized to contribute their own node for the Tor network
- System automatically searches for available mix servers

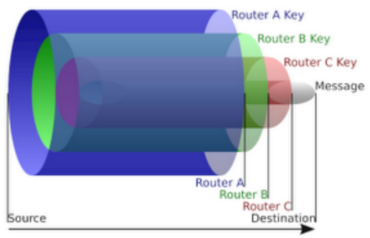
Problems

- Organization could offer a large number of nodes and thus the ability to control the entire network
- Browsing speed decreases significant when using Tor
- Last step *Mix N* → Web Server is unencrypted by default

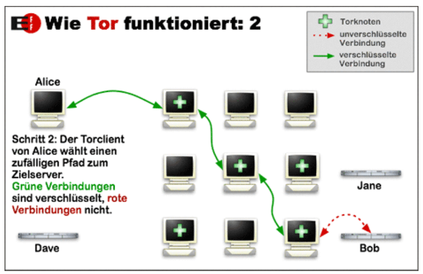
Tor: The Onion Router

Idea

- Use of a multi-layer encryption scheme
- Number of nodes to be used can be set individually by the user



Source: *http://en.wikipedia.org/wiki*



Source: *http://www.torproject.org*

Differences between Jondos/JAP & Tor

Jondos/JAP

- Cascades: fixed chain of Mixes
- Only one Mix cascade can be selected as user
- Generation of artificial messages
- Fixed number of servers (approx. 16)
- Supports HTTP/HTTPS/FTP
- Good performance with commercial version

Tor

- Dynamically variable routes of Mixes: random selection
- User has no control
- No artificial message generation
- Open network, many servers (2014, approx. 6000)
- Software can only be used as SOCKS proxy
- Performance varies depending on the selected paths

Types of Tor Nodes

1. **Onion Proxy:** User client program to connect to the network
2. **Onion Router:** Server for forwarding anonymous connections (middle server and exit server)
3. **Entry Guard:** Onion router, which acts as an entry point for the Tor network
4. **Directory Service:** Provides essential information about other servers on the network
5. **Introduction Point:** Server is required for hidden services in order to receive a message from a service user as a service provider
6. **Rendezvous Point:** Server is also used for hidden services as an anonymous communication point between service provider and user
7. **Bridge Relays:** These servers are highly protected and are intended for use by people from censored Internet networks

Features of TOR

➜ Classification according to degree of anonymity for participants

1. Use of public services, whereby *only* the user should be anonymous
2. Offer and use hidden services, whereby *both* user and provider should be anonymous

Features of TOR

➜ Classification according to degree of anonymity for participants

1. Use of public services, whereby *only* the user should be anonymous
2. Offer and use hidden services, whereby *both* user and provider should be anonymous

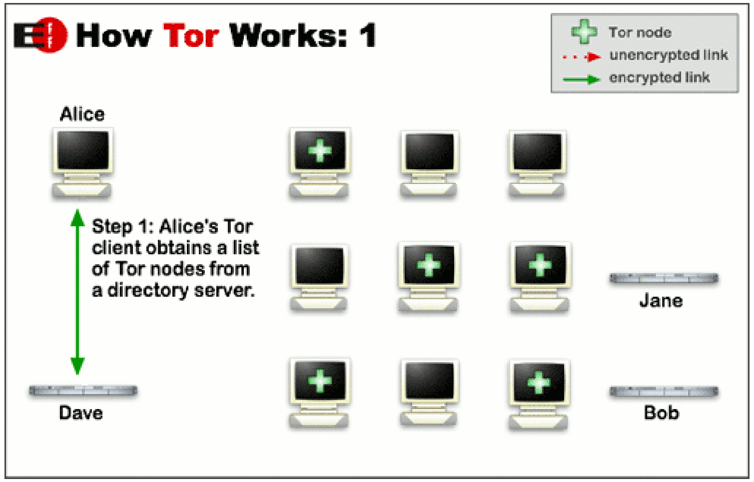
How to use public services anonymously?

Procedure

1. Alice defines the length of the routing (number of nodes) to the service and her client requests a list of Tor nodes from the directory server
2. Alice's client selects a random path to the service, taking into account the previously defined path length
3. The path is changed periodically for further requests to the service

Protocol Step 1

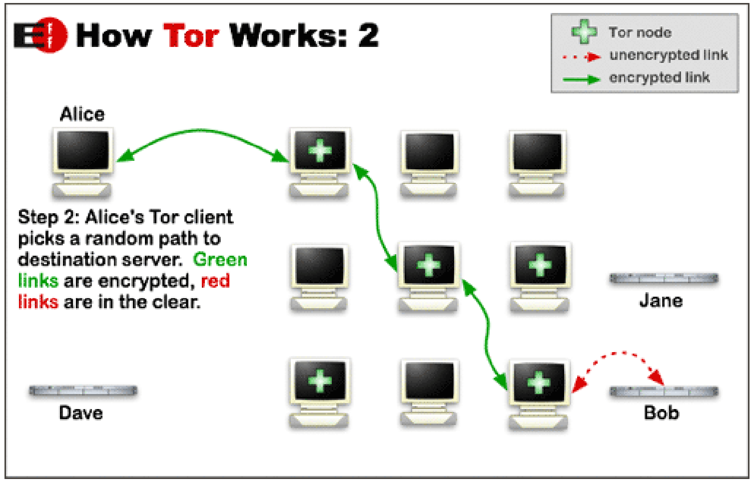
- ➜ Alice's client requests a list of Tor nodes from the directory server



Quelle: http://www.torproject.org

Protocol Step 2

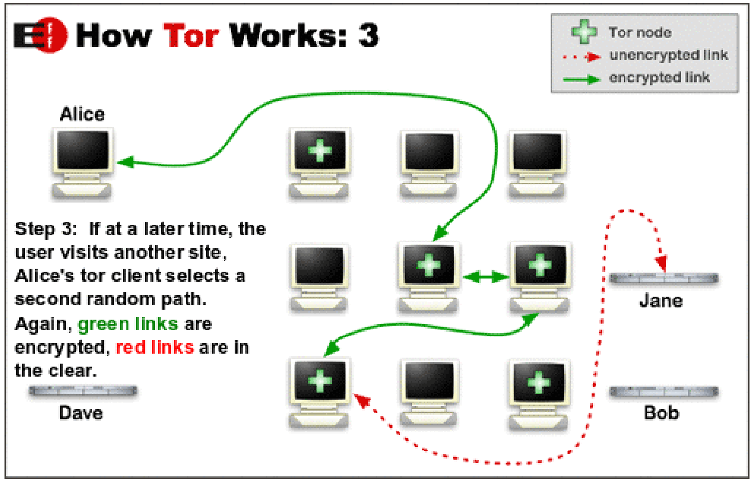
- ➜ Alice's client selects a random path to the service, taking into account the previously defined path length



Quelle: http://www.torproject.org

Protocol Step 3

- ➜ The path is changed periodically for further requests to the same or to other services

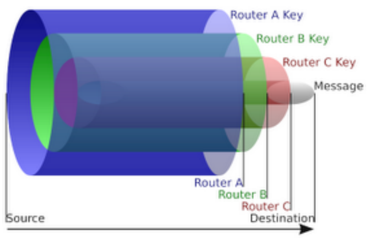


Quelle: http://www.torproject.org

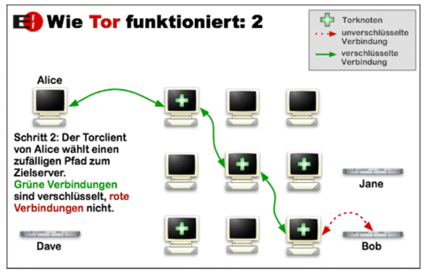
Tor: The Onion Router

How it really works?

- Implementation of onion-like encryption by *symmetric encrypted channels* ➜ called: circuits
- Asymmetric cryptography is used for key exchange ➜ Diffie-Hellman Protocol



Quelle: *http://en.wikipedia.org/wiki*



Quelle: *http://www.torproject.org*

Diffie-Hellman Key Exchange

1. Choose p and g randomly, where p is a *prime number* and g is a *primitive root of unity* for $\mathbb{Z}_p^*$

   mit $\mathbb{Z}_p^* = \{a : \mathbb{Z}_p \mid gcd(a, p) = 1\}$ und $\mathbb{Z}_p = \{0, \ldots, p-1\}$

   ➜ p and g are public

2. Alice and Bob have to choose randomly a and b of $\mathbb{Z}_p$

   ➜ a and b are secret

Alice: p, g, a — Bob: p, g, b

3. $A := g^a \bmod p$ (Alice → Bob)

4. $B := g^b \bmod p$ (Bob → Alice)

5. Alice calculates $K := B^a \bmod p$ and Bob $K := A^b \bmod p$

   ➜ K is the key for the symmetric encryption

Repetition Number Theory

Properties of the Primitive Root of Unity

- Let p be a prime number with $\mathbb{Z}_p = \{0, \ldots, p-1\}$

  Then is g *primitive root of unity*, if $g \in \mathbb{Z}_p^*$ and $\{1, \ldots, p-1\} = \{g^1, \ldots, g^{p-1}\}$

  ➜ The primitive root of unity g is also called *Generator* of $\mathbb{Z}_p \setminus \{0\}$

Correctness of the Generator

- Correctness of g can be proven efficiently if $p-1$ is factorizable

  ➜ if e.g. $p - 1 = 2 \cdot r$ and r is a prime number, so we have to prove that the following conditions are *not* satisfied

  $g^2 \equiv 1 \bmod p$ and $g^r \equiv 1 \bmod p$

Attack to Diffie-Hellman Key Exchange

➜ Traditional *Man-in-the-Middle attack*:
Eve pretends to Alice as Bob and to Bob as Alice

Alice: p, g, a — Eve: p, g, z — Bob: p, g, b

1. $A_a := g^a \bmod p$ (Alice → Eve)

2. $A_e := g^z \bmod p$ (Eve → Bob)

3. $B_b := g^b \bmod p$ (Bob → Eve)

4. $B_e := g^z \bmod p$ (Eve → Alice)

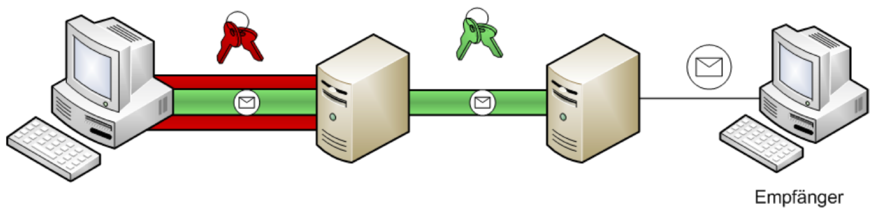
5. Alice calculates $K_a := (B_e)^a \bmod p$ und Bob $K_b := (A_e)^b \bmod p$

6. Eve calculates $K_a := (A_a)^z \bmod p$ und $K_b := (B_b)^z \bmod p$

   ➜ Using K_a und K_b Eve is able to listen in on the entire communication and even make changes

<u>Countermeasure:</u> Signing and encrypting using asymmetric algorithms!

How to establish a TOR connection?

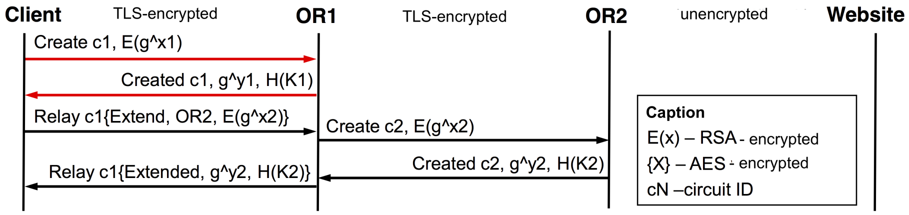


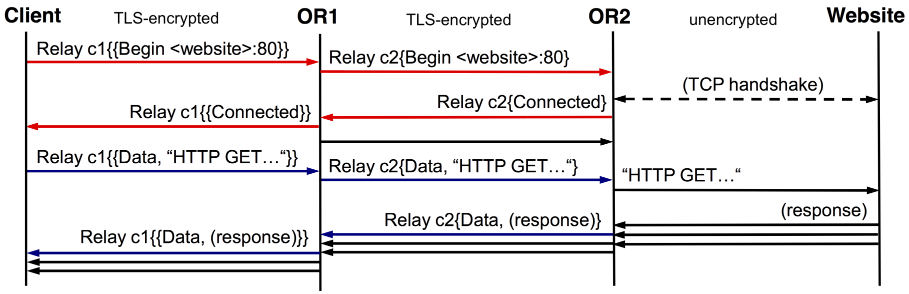
Procedure

1. Constructing Circuits (negotiating a symmetric key with each Onion Router on the circuit) based on asymmetric keys and Diffie-Hellman protocol
2. Data exchange via telescope-like channels based on TCP

Source: Figure of M. Ströbel, *Tor und Angriffe gegen TOR*, Seminar Paper, TUM, SS 2009

TOR Key Exchance & Communication





Source: S. Hasenauer, C. Kauba, S. Mayer: *Tor - The Second Generation Onion Router*, Seminar Slides, Uni Salzburg. http://www.cosy.sbg.ac.at/ held/teaching/wiss_arbeiten/slides_10-11/TOR.pdf, last access: 4.11.2014