Anonymous Web Browsing & DC-Nets

# Software Security

**Steffen Helke**
Chair of Software Engineering

12th November 2018

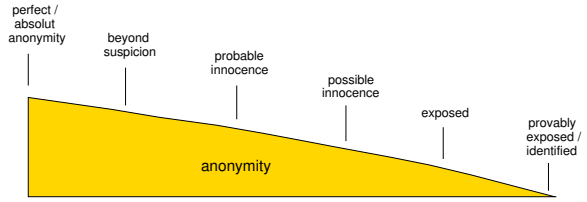
Brandenburgische
Technische Universität
Cottbus - Senftenberg

## Objectives of today's lecture

- ➜ Repetition, e.g. Differences between TOR and JAP
- ➜ Being able to reproduce two *protocols for the most important use cases* of TOR
- ➜ Understanding the principles of a DC-protocol – *Dining Cryptographers*



## Example to illustrate Degrees of Anonymity




Assumption: In our world we have only two potential sender, $A$ and $B$.

- $Q_A$ = '$A$ is identical to the observed sender $S$'
- $Q_B$ = '$B$ is identical to the observed sender $S$'

- **Beyond suspicion**
  → no more likely than any other potential sender,
  e.g. $P(Q_A) = 30\%$ and $P(Q_B) = 30\%$
- **Probable innocence**
  → no more likely to be the sender than not to be the sender
  e.g. $P(Q_A) = 50\%$ and $P(\neg\, Q_A) = 50\%$
- **Possible innocence**
  → there is a nontrivial probability that the real sender is someone else
  e.g. $P(Q_A) = 30\%$ and $P(Q_B) = 15\%$



## What are the differences between Jondos/JAP & Tor?

**Jondos/JAP**

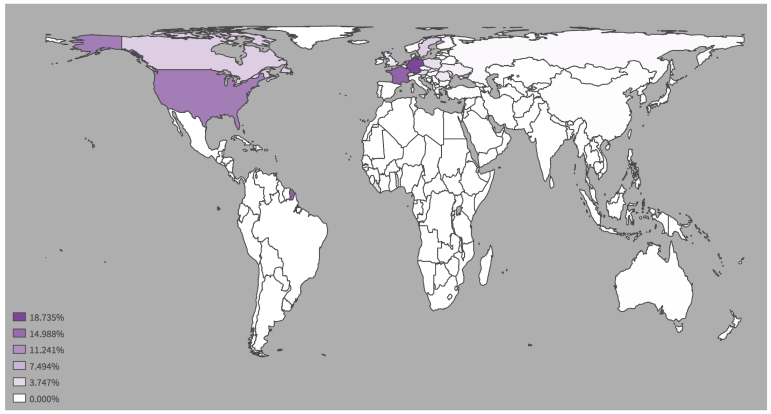
- Cascades: fixed chain of Mixes
- Only one Mix cascade can be selected as user
- Generation of artificial messages
- Fixed number of servers (approx. 16)
- Supports HTTP/HTTPS/FTP
- Good performance with commercial version

**Tor**

- Dynamically variable routes of Mixes: random selection
- User has no control
- No artificial message generation
- Open network, many servers (2018, approx. 6500)
- Software can only be used as SOCKS proxy
- Performance varies depending on the selected paths

# Why is it hard to provide your own exit node?

➜ Running your own exit node is not easy and requires a lot of experience. You may need to justify the fact that some illegal content is transported over your node. Depending on your Internet Service Provider (ISP), this can result in blocking your server or confiscating your server by the police.




This map shows the total exit probability of each country's relays as a percentage of the exit probabilities of all relays in the network. This probability is calculated based on consensus weights, relay flags, and bandwidth weights in the consensus. Path selection depends on more factors, so that this probability can only be an approximation.

*Source:* `https://metrics.torproject.org/`



# 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



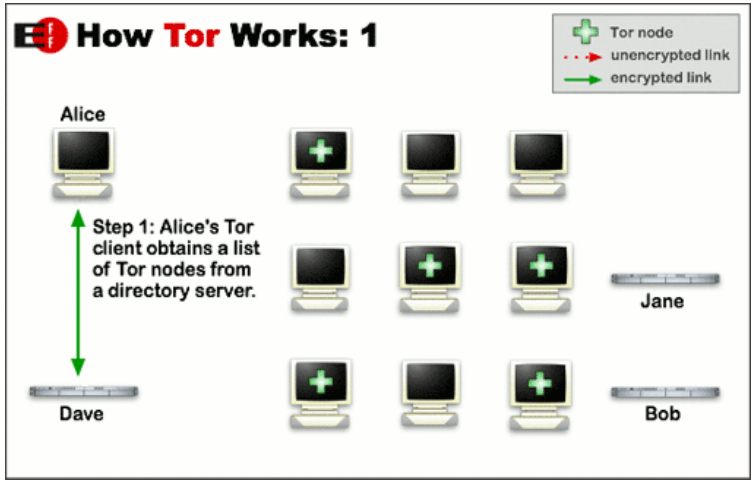
# How to use public services anonymously?

**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

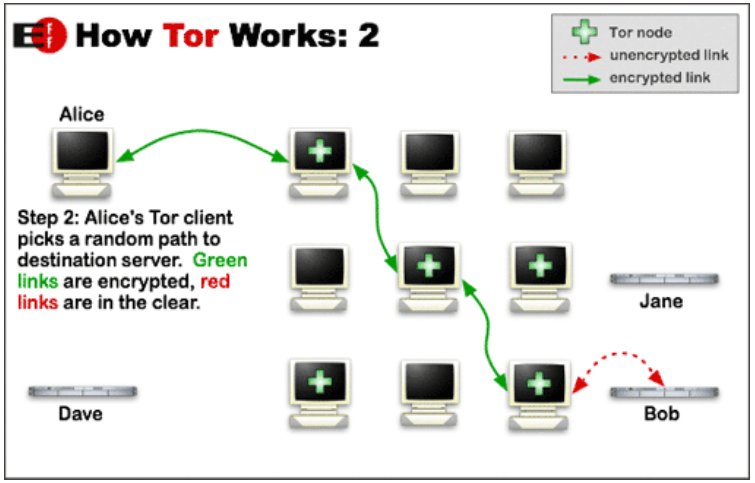



Quelle: http://www.torproject.org



# Protocol Step 2

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

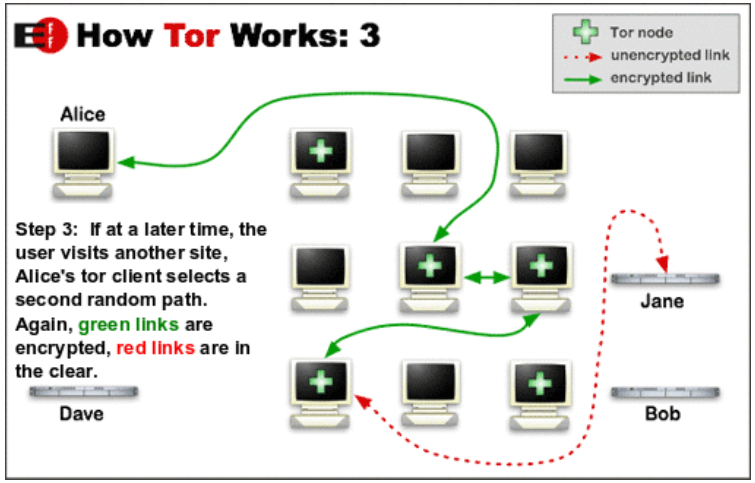



Quelle: http://www.torproject.org



# Protocol Step 3

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



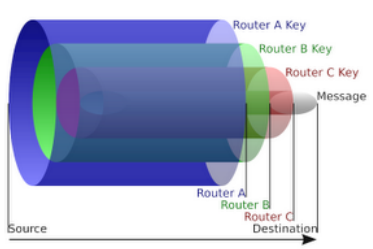

Quelle: http://www.torproject.org



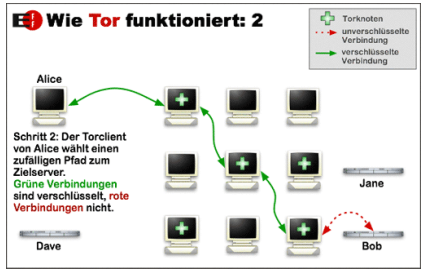
# Tor: The Onion Router

**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

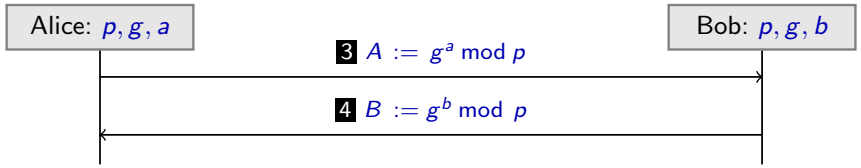
1. Choose $p$ and $g$ randomly, where $p$ is a *prime number* and $g$ is a *primitive root of unity* for $\mathbb{Z}_p^*$

   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




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

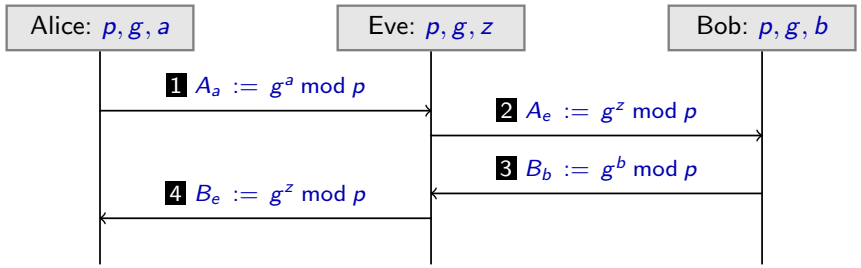
  ➜ if e.g. $p - 1 = 2 \cdot r$ and $r$ is a prime number, so we have to prove that the following conditions are *not* satisfied

  $g^2 \equiv 1 \bmod p$ 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



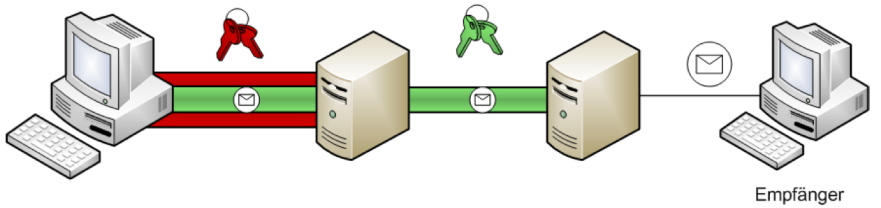

5. Alice calculates $K_a := (B_e)^a \bmod p$ und Bob $K_b := (A_e)^b \bmod p$
6. Eve calculates $K_a := (A_a)^z \bmod p$ und $K_b := (B_b)^z \bmod p$

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

Countermeasure: Signing and encrypting using asymmetric algorithms!



# How to establish a TOR connection?

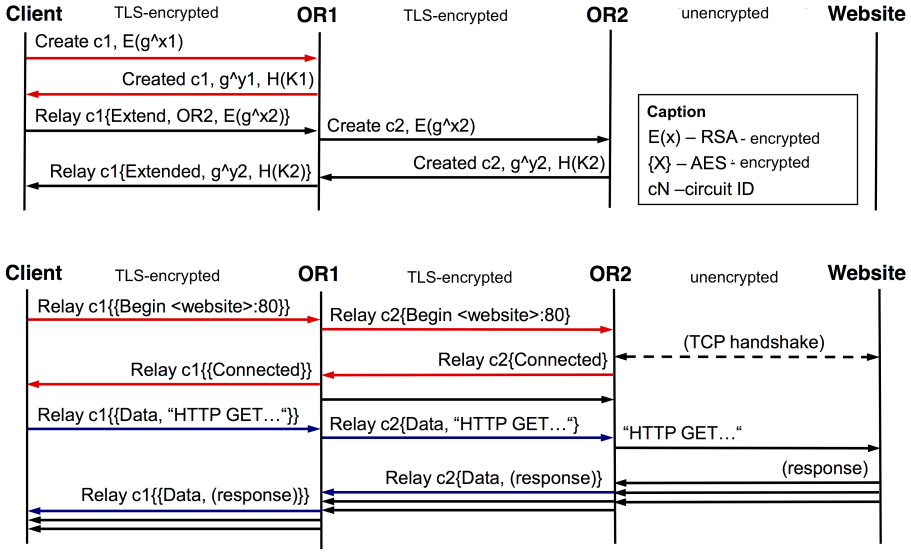



## Procedure

1. Constructing Circuits (negotiating a symmetric key with each Onion Router on the circuit) based on asymmetric keys and Diffie-Hellman protcol
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



# 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



# Hidden Services

**Procedure**

1. Provider (Bob) generates a long-term *key pair* to identify his service
2. Bob selects some introduction points randomly, and send his *public key* and the *List of introduction points* to the directory server, further he builds a circuit to each of his introduction points
3. User (Alice) learns about Bob's service from someone (just the hash of Bob's public key), e.g. Bob told her this information, or she found it on a website like this http://x.y.z.onion
4. Alice collects more *details of Bob's service* from the directory server by showing the learned hash value and retrieving the list of introduction points and Bob's public key



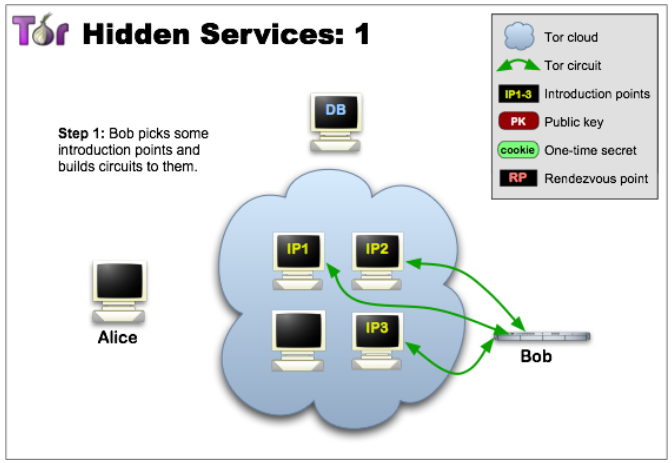
# Hidden Services (2)

5. Alice randomly selects any Tor server as a rendezvous point and builds a circuit to this server
6. Alice sends an encrypted message to Bob via one of Bob's contact servers, the message contains the selected *rendezvous point*, a *secret* (cookie) and the start of a DH handshake
7. Bob receives the message and establishes an anonymous *connection to the rendezvous point*
8. Bob places following data as a contact for Alice at the rendezvous point: *cookie*, the second half of the DH handshake, and a hash of the session key

➜ Finally Alice is able to identify Bob's service and both remain anonymous

# Protocol Step 1 (Hidden Services)

➜ Provider (Bob) generates a long-term *key pair* to identify his service, selects some introduction points randomly and builds a circuit to each of them

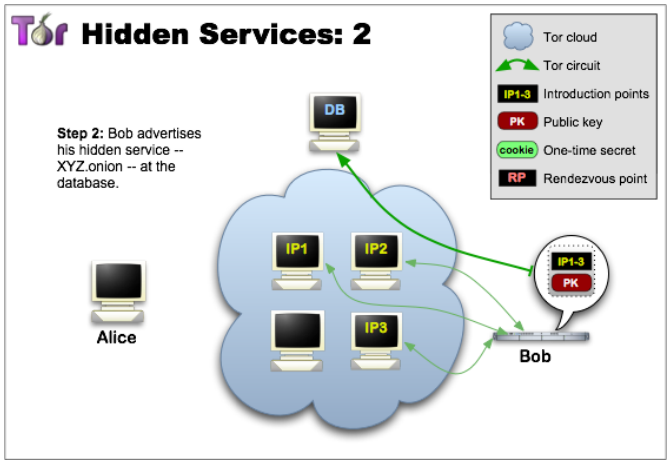



Source: http://www.torproject.org



# Protocol Step 2 (Hidden Services)

➜ Bob send his *public key* and the *List of introduction points* to the directory server

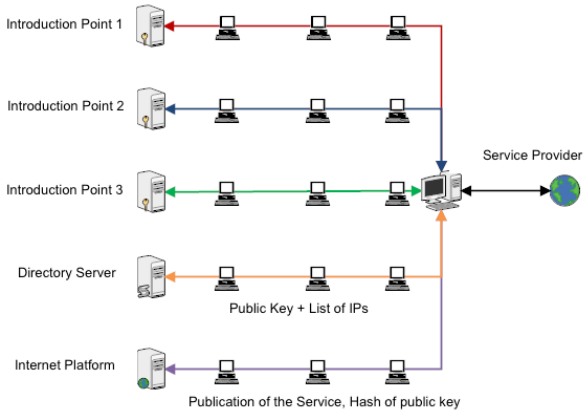



Source: http://www.torproject.org



# Protocol Step 1 & 2 (Hidden Service)

➜ All protocol steps are implemented via anonymous circuits, i.e. no direct access from Bob to contact or directory servers

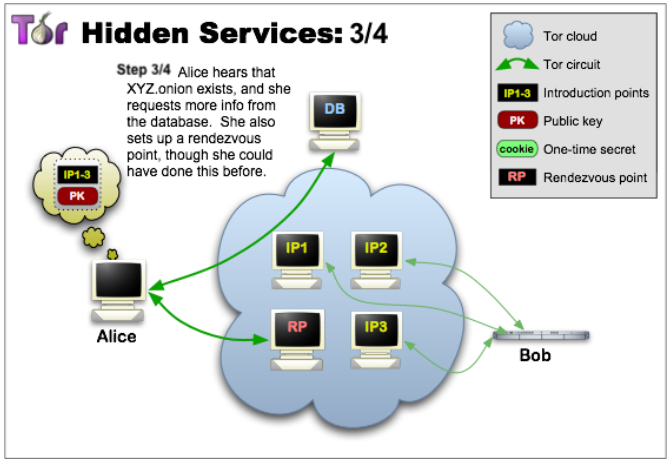



Source: Adapted from http://hp.kairaven.de/



# Protocol Steps 3 & 4 (Hidden Services)

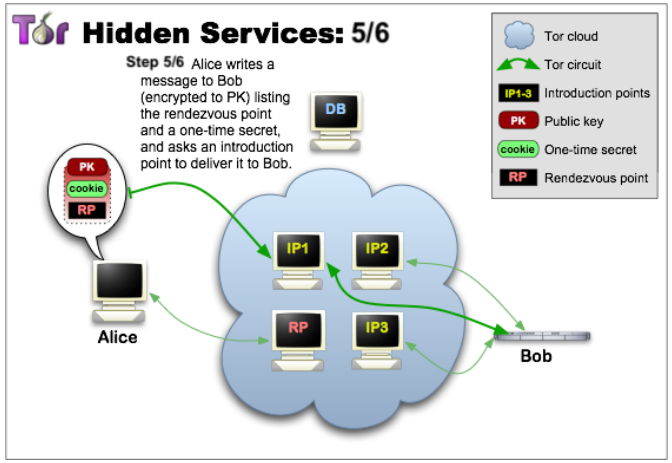
➜ Alice collects *details of Bob's service* from the directory server by showing a learned hash value of the service and retrieving the list of introduction points and Bob's public key




Source: http://www.torproject.org

# Protocol Steps 5 & 6 (Hidden Services)

➜ Alice sends an encrypted message to Bob via one of Bob's contact servers, the message contains the selected *rendezvous point*, a *secret* (cookie) and the start of a DH handshake

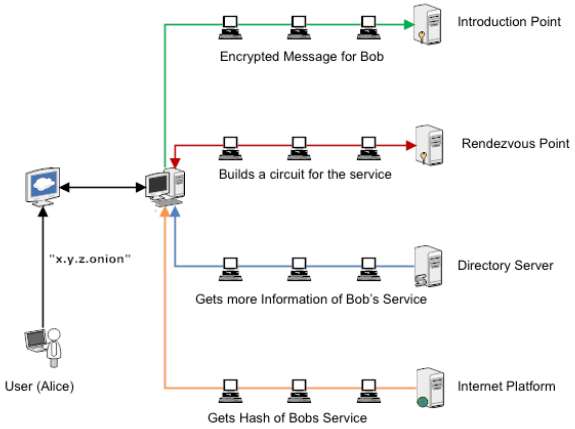



Source: http://www.torproject.org



# Protocol Steps 3, 4, 5 & 6 (Hidden Services)

➜ All protocol steps are implemented via anonymous circuits, i.e. no direct access from Alice to contact or directory servers

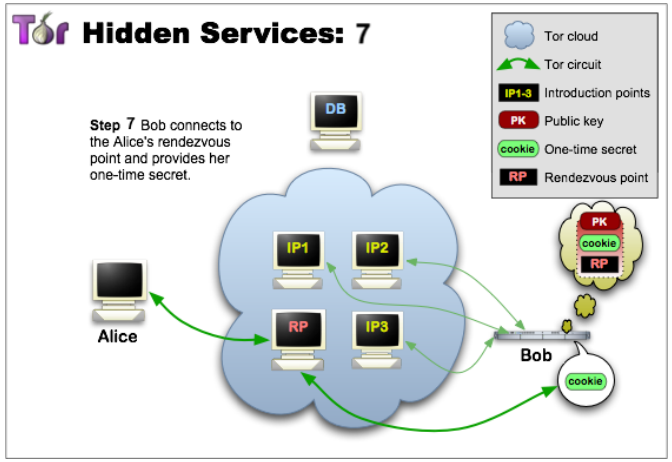



Source: Adapted from http://hp.kairaven.de/



# Protocol Step 7 (Hidden Services)

➜ Bob receives the message, establishes an anonymous *circuit to the rendezvous point* and places some data for Alice

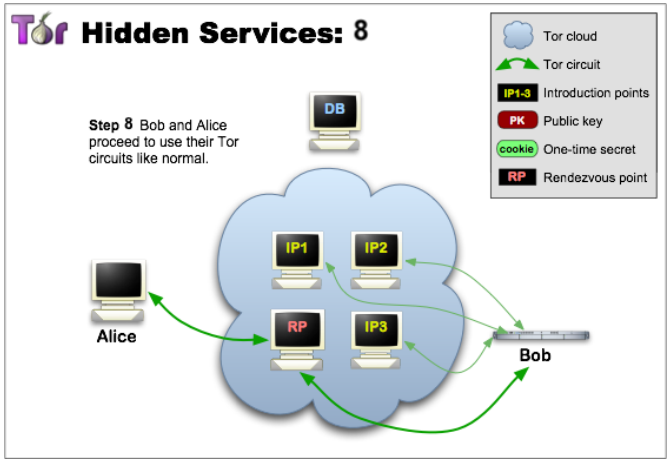



Source: http://www.torproject.org



# Protocol Step 8 (Hidden Services)

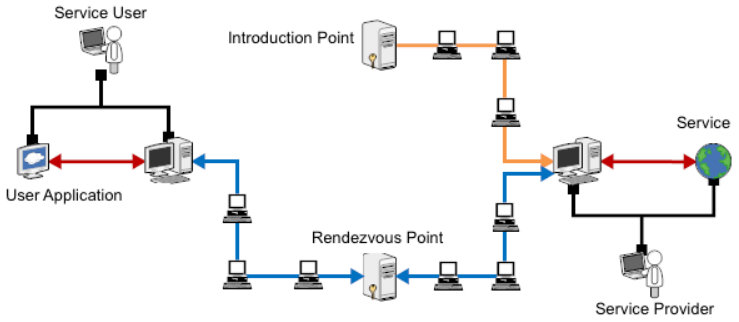
➜ Finally Alice is able to identify Bob's service at the rendezvous point and both remain anonymous




Source: http://www.torproject.org

# Protocol Steps 7 & 8 (Hidden Services)

➜ All protocol steps are implemented via anonymous circuits, i.e. Alice and Bob have no direct access to the rendezvous point




Source: Adapted from http://hp.kairaven.de/



# Some Attacks to the TOR Protocol

**Traffic Analysis**

➜ Observation of the first and last node in a connection and statistical evaluation to determine the source of the connection

➜ Observation of latencies for individual packets or the bandwidth of data streams

**Application Protocols**

➜ The attacker provides his own web server to spy on TOR users by injecting code into the client system

**Other Attacks**

➜ If the integrity tests are poorly implemented, packages can be tagged and the messages tracked

➜ DoS attacks to either limit the use of TOR or redirect traffic to compromised servers