I2P – Anonymous low latency networking

Anonymität ist nicht binär.

Oder: der Versuch, ein schwer angreifbares Netzwerk zu erstellen.

www.i2p2.de

Übersicht

I2P ist ein „low latency“ Mix Netzwerk:

- Geringe Latenz im sec. Bereich, die gering genug für IRC ist; im Gegensatz zu freenetproject.org mit hoher Latenz
- Mix Netzwerk – Nachrichten basierter Traffic wird über eine einstellbare Anzahl von anderen I2P Router geleitet bevor es am Ziel ankommt

Grundlagen

I2P versucht ein Netz aufzubauen, welches eine ausgewogene Gewichtung zwischen Anonymität, Zuverlässlichkeit, Bandbreite und Latenz bietet.

Es gibt keinen zentralen Server der angegriffen werden kann und damit das Netz zusammenbrechen lässt, alles nötige ist dezentralisiert.

Das normale Internet mit seinen Applikationen ist nicht auf Anonymität ausgelegt, deswegen müssen die Applikationen angepasst/ersetzt werden oder ggf. die Daten vor dem Übertragen gefiltert werden.

I2P versteckt nicht die Tatsache, das I2P auf einem PC rennt, es verschlüsselt die Daten und versteckt den Weg der Daten durchs Netz.

I2P ist auf ein internes Netz designed, es gibt keine offiziellen Outproxies ins normale Netz.

Zahlen

- Größe: ca. 550-700 Router zur Zeit
- Ca. 400 Server (leasesets)
- Router aus ca. 40 versch. Staaten
- Latenz ca. 1-4 sec im IRC
- Transferraten von ca. 10-20 kb/sec im Schnitt
- Ca. 100-150 aktive Connections (UDP und TCP)
- IRC, Gnutella Client, torrent Clients, edonkey Client in extra angepassten Versionen
- Z.Z. 2 aktive Hauptentwickler

Startup

- Java Applikation mit encryption core in ASM
- Client für Linux/Mac/Windows auf www.i2p2.de
- Beim Startup holt der client seednodes (Informationen über andere I2P Router) und die FloodFill Router
- Danach check auf Erreichbarkeit der Ports (8887)
- Startup des Routers und Startup eines Browsers mit lokaler Router Konsole
- Startup der lokalen Serverapplikationen
- Versuch die Tunnel über andere Router aufzubauen
- Bekanntgabe der lokalen leasesets an die FloodFill

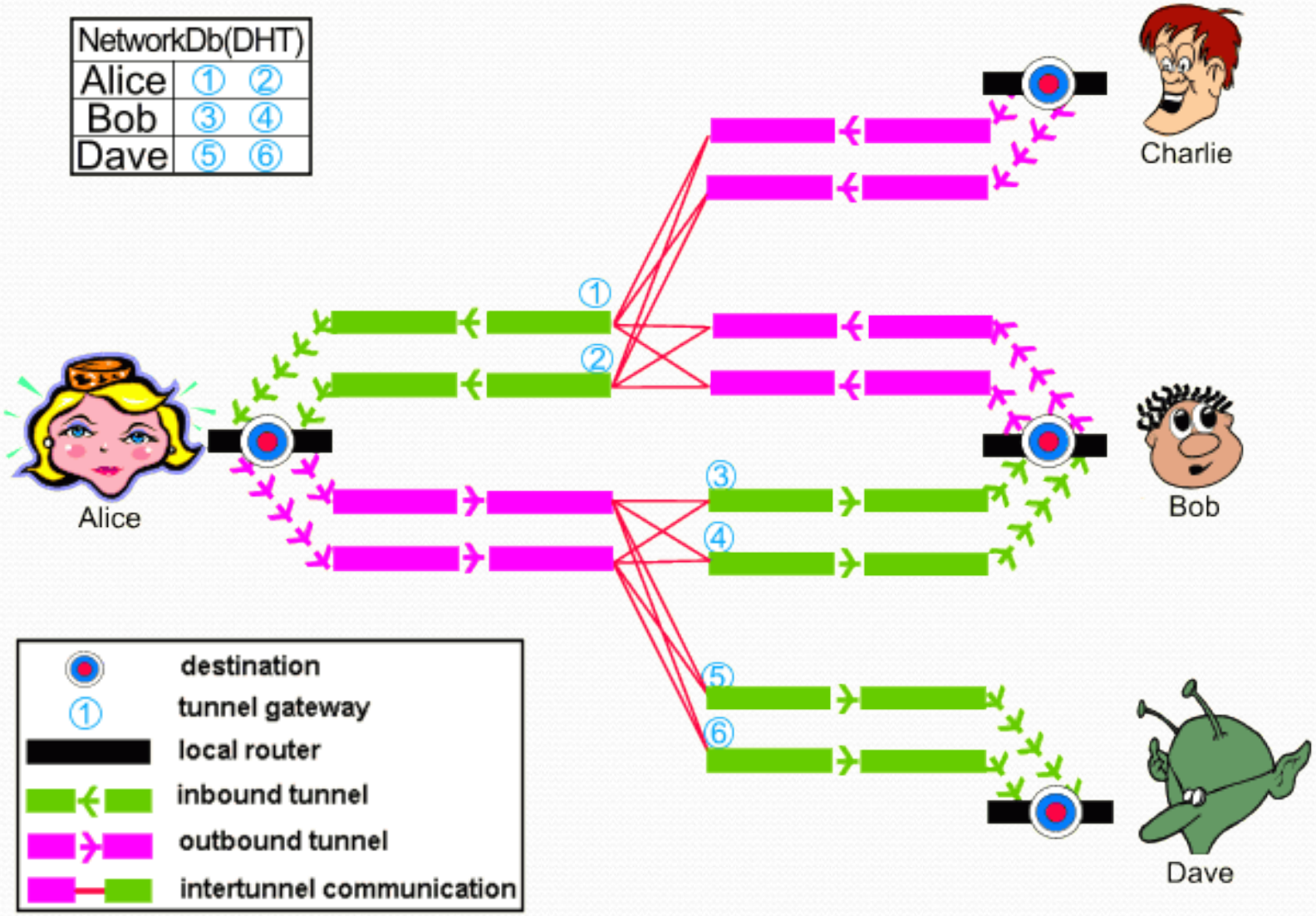
FloodFill Router

- Zuerst: kadmelia Datenbank für leasesets, war zu fehleranfällig
- Jetzt: Datenbank auf 1-n FloodFill Router
- 1 Leaseset ist die Information, über welche In-Tunnel eine Destination im I2P Netz erreicht werden kann
- Destination: 516bit base64 encoded String als ID
- Max. Lebensdauer eines Leasesets: 10 Minuten, danach werden neue Tunnel aufgebaut und in einem neuem Leaseset publiziert
- I2P Router fragen die FloodFill Router um für eine Destination ein Leaseset zu bekommen
- FloodFill Router tauschen leaseset Infos untereinander aus
- Z.Z. 5 FloodFill, sollte bis ca. 100.000 Router ausreichen
- FloodFill ist nur temporär bis eine bessere Lösung kommt

Routing

- Generelle Unterteilung in send/receive Tunnels, die auf verschiedenen Wegen im Netz aufgebaut werden (im Gegensatz zu tor, bei dem send/receive den selben Weg nehmen)
- Teilung in In/Out Tunnels:
  - Ein Weg von der Quelle zum Ziel wird durch den In + Out Tunnel aufgebaut
  - Für den send Tunnel baut der Client den Out-Tunnel auf, der Server den In-Tunnel
  - Für den receive Tunnel baut der Client den In-Tunnel auf, der Server den Out-Tunnel
  - Die Anzahl der Hops per In/Out Tunnel sind beim Client und Server konfigurierbar (ebenfalls die Anzahl der Tunnel je In/Out Weg)
  - Ein Client sieht also nicht, wieviele Hops der gesamte Tunnel bis zum Server hat! (nur seinen in/out Teil der Tunnel)

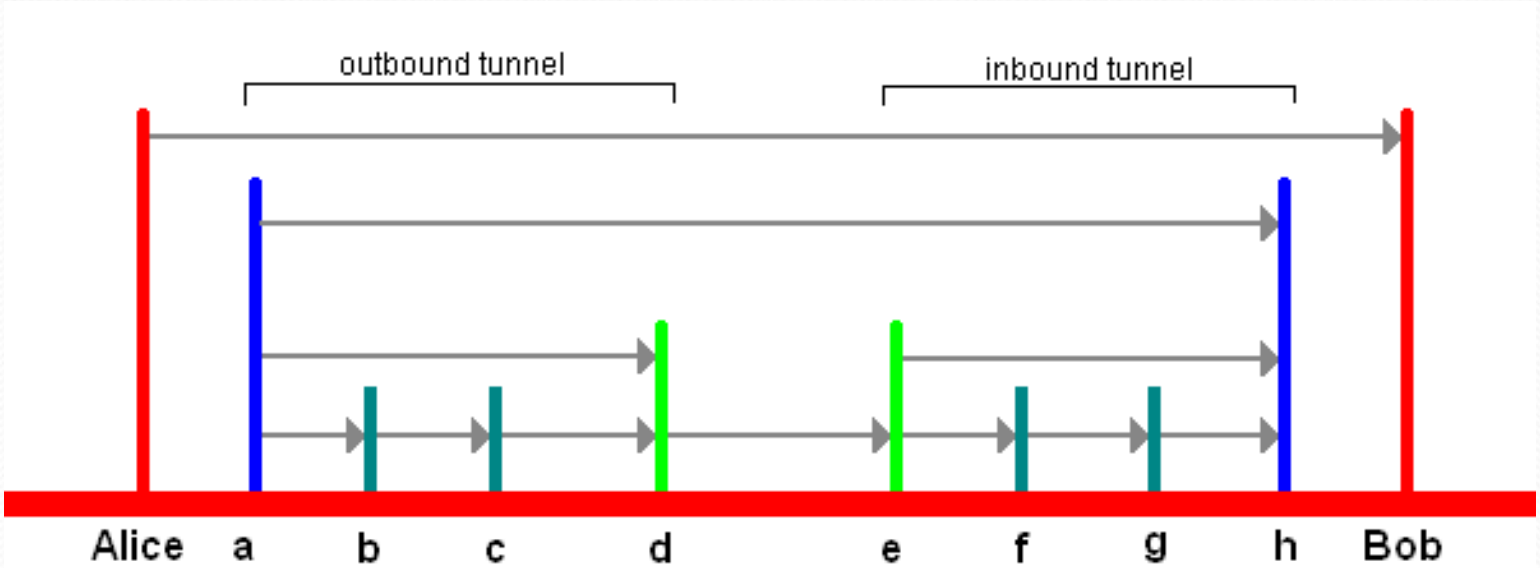
Routing


NetworkDb(DHT)
Alice ① ②
Bob ③ ④
Dave ⑤ ⑥
Charlie
Alice
Bob
Dave
destination
tunnel gateway
local router
inbound tunnel
outbound tunnel
intertunnel communication

Verschlüsselung

- Verschlüsselung der Daten wird auf 3 oder 4 Ebenen ausgeführt
  - End-to-end Encryption
  - „Garlic“ Encryption
  - Tunnel Encryption
  - Inter-router Transport Encryption
- ElGamal/AES+SessionTag Encryption meistens, 2048 ElGamal und 256bit AES in CBC mode mit PKCS#5 padding für 16 byte blocks
- Mehr infos unter: http://www.i2p2.de/how_cryptography

Verschlüsselung



end to end encryption (ElGamal/SessionTag+AES)	Alice to Bob	tunnel encryption (private key AES)	a to d, e to h
garlic encryption (ElGamal/SessionTag+AES)	a to h	transport encryption (DH/STS+AES, etc)	a to b, b to c, c to d, d to e, e to f, f to g, g to h

Tunnel

- Eines der zentralen Elemente von I2P
- 2 verschiedene Tunnel-Klassen:
  - Exploratory Tunnels – Werden wie normale Tunnel aufgebaut, Datenerhebung für Erreichbarkeit und Geschwindigkeit der anderen Router werden hierrüber erhoben
  - Client Tunnels – die eigentlichen „Daten“ Tunnels, Applikationen senden ihre Daten durch diese Tunnels
- Participating Tunnels – andere Tunnel die durch den Router geroutet werden (nur Anzahl limitierbar)

Client Tunnels

- „frei“ konfigurierbar in Anzahl (1-4) und Hops (0-4)
- zufallsgesteuerter Variabilität (+/- 0-3 Hops) für die Anzahl der Hops
- Dazu 0-3 Backup Tunnel einstellbar
- Jede Applikation kann eigene Client Tunnels erstellen/nutzen
- Shared Clients – Tunnel wird gemeinsam von div. Applikationen genutzt

Besondere Tunnelarten

- Es existieren 2 besondere Arten von I2P Tunnel:
  - IRC (Client)Tunnel – es werden diverse CTCP Nachrichten ausgefiltert aus dem Stream des IRC Clienten (u.a. DCC oder IP Adressen, die die Clienten senden)
  - http (server) Tunnel – hier werden ebenfalls diverse Nachrichten aus dem Stream ausgefiltert (u.a. die Browserkennung ausgetaucht)

I2P DNS

- Server/Clients sind über den destination Key definiert
- Dieser wird beim Startup des Servers generiert und in einer Datei gespeichert
- Da dieser zu lang zum merken ist, wurde ein einfaches DNS System aufgesetzt
- Jeder Router hat eine hosts.txt Datei, die als lookup-Tabelle benutzt wird
- Per default wird mit susiDNS diese hosts.txt gegen 1 Server im I2P Netz abgeglichen (push UND poll)
- Man kann gegen weitere DNS Server (die syncen sich gegenseitig) die Datei abgleichen oder dieses abstellen
- Dazu existiert die lokale Datei userhosts.txt, die nicht streng lokal ist
- Host-Kollisionen SIND möglich (und auch erwünscht)!

Applikationen

- Eingebaute Applikation als java-Applet im Browser auf der Statuskonsole aufrufbar:
  - Susimail – einfacher Zugriff auf anon Email Dienst postman.i2p
  - SusiDNS – einfache HostDB (hosts.txt) Verwaltung
  - I2PSnark – Snark torrent Client im Browser für I2P
- Dazu diverse andere Clients:
  - Imule – emule client für I2P (ggf. MIT integriertem I2P router => get it, start it, use it)
  - I2Phex – Phex Gnutella Client für I2P angepasst
  - Azureus Plugin – Bittorrent für I2P mittels Azureus (dangerous)
  - I2p-bt /i2p.bytemonsoon tracker – bittorrent client/tracker

Fazit

- I2P hat vieles eingebaut um „sicher“ zu sein
- Einfacher Betrieb möglich
- Latenzzeiten sind gering genug
- Kein TOR Ersatz – nicht als anonymen Zugriff aufs normale Netz gedacht
- Hohe CPU Last durch viel En/Decryption und hohe Anzahl von Tunnel
- „hohe“ Bandbreite (>30 kb/sec) nötig um eigenen Traffic in fremden Traffic zu „verstecken“

Live-Show

- Windows mit SSH Tunnel auf root server mittels Putty
- Fragen?