I2P, The Invisible Internet Projekt

jem

September 20, 2016 at Chaostreff Bern

Content

Introduction

- About Me
- About I2P

Technical Overview

- I2P Terminology
- Tunnels
- NetDB
- Addressbook
- Encryption
- Garlic Routing
- Network Stack

Using I2P

- Services
- Using I2P with any Application
- Tips and Tricks (and Links)

Conclusion

Introduction

About Me

- Just finished BSc Informatik at BFH
- Bachelor Thesis: "Analysis of the I2P Network"
- Focused on information gathering inside and evaluation of possible attacks against I2P
- Presumes basic knowledge about I2P
- Contact: jens@jabber.chaostreffbern.ch (XMPP) or jens@jenix.net (GPG 0x28562678)

Introduction

About I2P: I2P = TOR?

Similar to TOR...

- Goal: provide anonymous communication over the Internet
- Traffic routed across multiple peers
- Layered Encryption
- Provides Proxies and APIs

...but also different

- Designed as overlay network (strictly separated network on top of the Internet)
- No central authority
- Every peer participates in routing traffic
- Provides integrated services: Webserver, E-Mail, IRC, BitTorrent
- Much smaller and less researched

Introduction

About I2P: Basic Facts

- I2P build in Java (C++ implementation I2Pd available)
- Available for all major OS (Linux, Windows, MacOS, Android)
- Small project –> slow progress, chaotic documentation, ...
- Core team consists of few people "spread around several continents", many vacant positions
- Volatile services: many dead links, announced services / websites / project with unknown status (probably dead)
- I2P is becoming more popular, especially with growing concerns about TOR's security (First Darknet Shops migrating to I2P)
- I2P seems to be quite popular in the Russian-speaking part of the world (many websites in Cyrillic)

Introduction

About I2P: Timeline

- Project started in 2003
- Major throw back in November 2007, when lead developer (jrandom) left the project, took important infrastructure with him (including official website i2p.net) and since disappeared
- Lots of reorganizing needed afterwards (new website, new release keys, etc.), slowed the development of I2P down
- Today: v0.9.26 (2016-06-07), deemed stable and secure by devs, though no complete code review done (yet)
- Target: New version every 6 - 8 weeks (currently behind schedule)

Introduction

About I2P: Network Size

- Hard to guess amount of users or services
- Some numbers:
  - Amount of simultaneous routers observed during Thesis: about 6000 - 7000 (stable), but accuracy of this number unknown
  - Number dropped to 1000 - 1500 currently, but may be even more inaccurate due to changes in the network
  - Current entries in official Addressbook: 368, but many sites unreachable (may be temporary or permanent)

Technical Overview

I2P Terminology

- Participating peers called Routers
- Eepsite: Service accessible via I2P
- Identity: SHA256-Hash of encryption keys
- 2 types of identifier inside the network:
  - routerInfo: Identity, IP-Address and Port of router (used to contact a Router)
  - leaseSet: Identity, Tunnel-Gateway and Tunnel-ID of service (used to contact a Service)

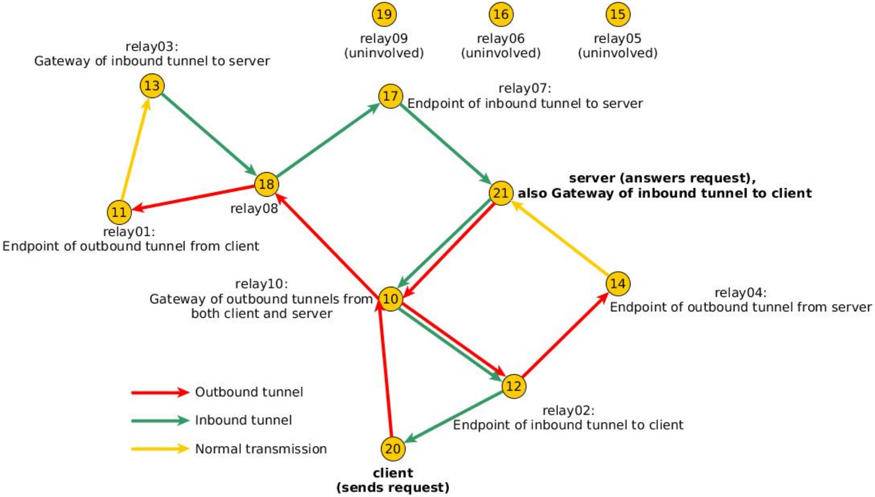
Technical Overview

Tunnels

- Like circuits in TOR
- Fix set of routers used to forward traffic (default: 3 per tunnel)
- 2 Types: Inbound for receiving, Outbound for sending packets
- Created by every router
- Valid for 10 Minutes (then new ones are created)
- Multiple tunnels per service
- Gateway: First router of a tunnel
- Endpoint: Last router of a tunnel
- Unique Tunnel-ID
- Gateway and Tunnel-ID part of leaseSet

Technical Overview

Tunnels



19 relay09 (uninvolved)

16 relay06 (uninvolved)

15 relay05 (uninvolved)

13 relay03: Gateway of inbound tunnel to server

17 relay07: Endpoint of inbound tunnel to server

18 relay08

21 **server (answers request), also Gateway of inbound tunnel to client**

11 relay01: Endpoint of outbound tunnel from client

10 relay10: Gateway of outbound tunnels from both client and server

14 relay04: Endpoint of outbound tunnel from server

12 relay02: Endpoint of inbound tunnel to client

20 **client (sends request)**

🟥 Outbound tunnel

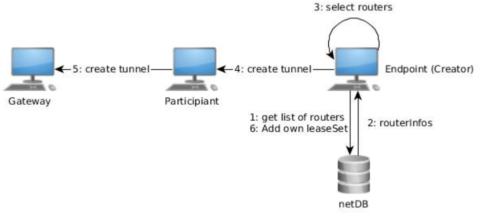
🟩 Inbound tunnel

🟨 Normal transmission

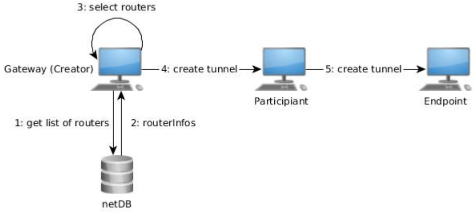
Technical Overview

Tunnel Building

Inbound:



Outbound:



- Router collects routerInfos from netDB and selects participants of tunnel
- Computes the creation message for each participant and sends it to first router
- This one learns that a new tunnel is being created and forwards the message
- Every participant only learns, that he should forward packets from the previous router to the next one (identified by Tunnel-ID)
- Only our Router knows that he is the endpoint respectively the gateway

Technical Overview
NetDB

- Contains contact information for all routers and services (routerInfos and leaseSets)
- Distributed database: spread across participating routers (called floodfill routers)
- floodfill routers automatically selected based on capabilities or if they volunteer
- Kademlia DHT approach: Identifier mapped to an address space, the 7 "closest" floodfill routers are used to store entries
- If floodfill router does not have the requested entry (so it is not one of the 7 closest), it knows floodfill routers that are closer and redirects to them
- Kademlia DHT to be replaced by other mechanism in the future due to possible attacks against it (control the 7 closest floodfill routers)
- Demo: netDB entries in I2P Router Console

Technical Overview

Addressbook

- Destination: leaseSet “name”, encoded in base32 / base64
- Example: uwyqjovhwu2vsam7ijqxzzuwvweu3rza5b7hphmgjunbflgldvua.b32.i2p
- Destination hard to remember
- Use resolver hostname –> destination (like DNS)
- Done by Addressbook
- Public part (synchronized with published entries)
- Private part (higher priority during lookup)
- All entries modifiable, possibility to publish entries
- Jump Services provide resolving (like public DNS servers)
- Demo: Addressbook entries in I2P Router Console / Jump Services

Technical Overview

Address Lookup

- Real address of service: destination (leaseSet)
- Opt: Resolve hostname –> destination via Addressbook
- leaseSet queried from NetDB
- Tunnel information (Gateway and Tunnel-ID) extracted from leaseSet
- NetDB lookup for routerInfo of Gateway
- Send packets to Gateway using Outbound Tunnel

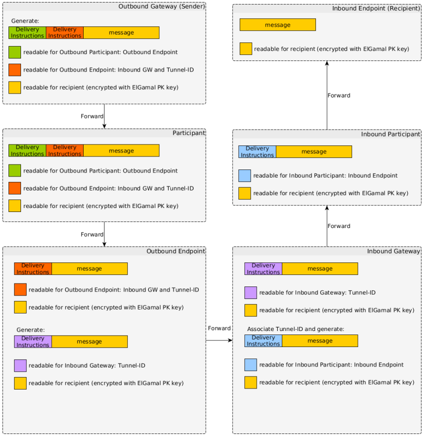
Technical Overview

Encryption

- 2 different encryption schemes
- AES256-CBC with session key inside tunnel
- Layered Encryption: apply multiple layers of encryption for each hop in the tunnel
- ElGamal outside tunnels (NetDB lookups, transport between tunnels)
- Public Key in Identity
- Message Authentication with EdDSA25519 signatures
- Signing Key in Identity
- Daily key-rollover

Technical Overview

Layered Encryption



Outbound Gateway (Sender)

Generate:

Delivery Instructions | Delivery Instructions | message

🟩 readable for Outbound Participant: Outbound Endpoint

🟧 readable for Outbound Endpoint: Inbound GW and Tunnel-ID

🟨 readable for recipient (encrypted with ElGamal PK key)

Forward

Participant

Delivery Instructions | Delivery Instructions | message

🟩 readable for Outbound Participant: Outbound Endpoint

🟧 readable for Outbound Endpoint: Inbound GW and Tunnel-ID

🟨 readable for recipient (encrypted with ElGamal PK key)

Forward

Outbound Endpoint

Delivery Instructions | message

🟧 readable for Outbound Endpoint: Inbound GW and Tunnel-ID

🟨 readable for recipient (encrypted with ElGamal PK key)

Generate:

Delivery Instructions | message

🟪 readable for Inbound Gateway: Tunnel-ID

🟨 readable for recipient (encrypted with ElGamal PK key)

Forward

Inbound Gateway

Delivery Instructions | message

🟪 readable for Inbound Gateway: Tunnel-ID

🟨 readable for recipient (encrypted with ElGamal PK key)

Associate Tunnel-ID and generate:

Delivery Instructions | message

🟦 readable for Inbound Participant: Inbound Endpoint

🟨 readable for recipient (encrypted with ElGamal PK key)

Forward

Inbound Participant

Delivery Instructions | message

🟦 readable for Inbound Participant: Inbound Endpoint

🟨 readable for recipient (encrypted with ElGamal PK key)

Forward

Inbound Endpoint (Recipient)

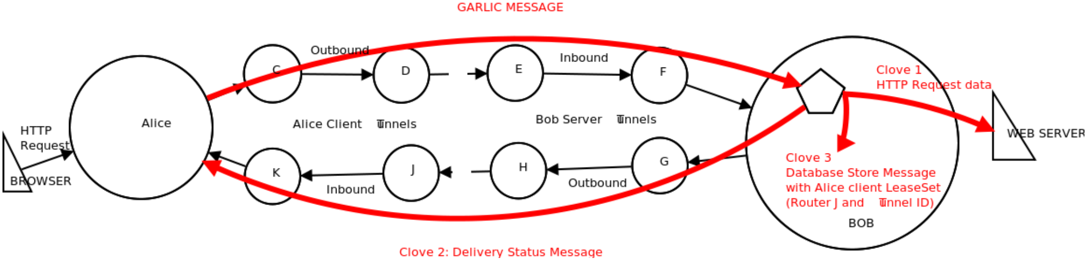
message

🟨 readable for recipient (encrypted with ElGamal PK key)

Technical Overview

Garlic Routing

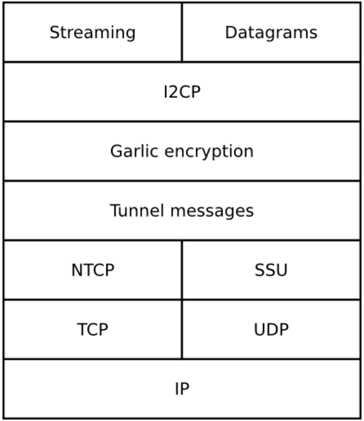
- Term based on „Onion Routing“ (TOR)
- Multiple meanings:
  - Combine multiple messages for the same recipient
  - Protect message with multi-layered encryption
  - Use both AES and ElGamal encryption
- Goal: prevent Timing Attacks (finding paths by correlating incoming and outgoing packets)



Technical Overview

Network Stack

- I2P uses self-developed protocols
- "Management" protocols:
  - I2CP (I2P Client Protocol, let application communicate via I2P)
  - I2NP (I2P Network Protocol, used to manage I2P network itself)
- Transport protocols:
  - NTCP (Java-NIO based TCP)
  - SSU (Secure Semi-reliable UDP)
- Full documentation on I2P Website



Streaming | Datagrams

I2CP

Garlic encryption

Tunnel messages

NTCP | SSU

TCP | UDP

IP

Using I2P
Services

I2P offers many different Services

- Hosting / Browsing
- eMail: susimail (postman over I2P) & I2P-Bote (Kademlia DHT-based mail system)
- Chat: IRC (with Relay-Bots between I2P and the Internet), Jabber & I2P-Messenger (serverless, based on destination keys)
- Filesharing: integrated BitTorrent-client "I2PSnark", additional programs (e.g. iMule) –> provides "base" traffic
- Blogs / Forums: Syndie
- "Cloud": Tahoe-LAFS cloud over I2P
- Unofficial gateways to www and TOR
- Adapt own application to use I2P

Using I2P with any Application

I2P provides different APIs to use it with any application

- I2PTunnel translates ip:port into I2P destination
- SOCKS Proxy
- SAM v3: Libraries for C, C++, Go and Haskell
- BOB: Library for Go, Python, Twisted
- I2PControl: JSON-RPC2 interface to control I2P router from within an application
- Full documentation for every API on I2P website

Using I2P

Tips & Tricks (and Links)

- TOR-Browser can be configured to use both TOR and I2P by using the extension FoxyProxy (Caution: Third-Party extension). Tutorial: `http://thetinhat.i2p/tutorials/darknets/i2p-browser-setup-guide.html`
- eepstatus (List of available I2P sites): `http://identiguy.i2p`
- Access Eepsites from the Internet by adding .xyz: e.g. site.i2p –> site.i2p.xyz (not working for every site)
- I2P Observer (Result of my Thesis to gather information about I2P): `https://jenix.net/i2p-observer`

Conclusion

Should I use I2P?

Pro:

- Small project
- Not (yet) in focus of surveillance (?)
- Many great ideas to strengthen security (Personal impression after Thesis)
- Developers are aware of possible problems:
  - early switches to strong cryptography
  - fast responses to possible issues (e.g. disabling potential insecure ciphers)
- Diversity always desirable
- Many build-in services and easy adaptation for any application

Conclusion

Should I use I2P?

Con:

- No proof of security yet (Missing code audit)
- Much smaller network (therefor easier to monitor)
- Major changes needed to address published attack possibilities (focused mainly on netDB)
- Visible I2P network often feels deserted

Conclusion

Should I use I2P?

So, should I use I2P?

- Depends on your personal stance towards Pros and Cons.
- If you want to: `https://www.geti2p.net`