Technical Documentation

I2P Technical Overview

Intro and Threat Analysis
Overview of the network and its capabilities

Specifications and Protocol Stack
Network architecture

Roadmap and Proposals
Development roadmaps

I2P Application Layer

App Development Overview and Guide
Tagline here

App Layer API and Protocols
High-level, easy to use APIs for applications written in any language to send and receive data

End-to-End Transport API and Protocols
The end-to-end protocols used by clients for reliable and unreliable communication.

Client-to-Router Interface API and Protocol
The end-to-end protocols used by clients for reliable and unreliable communication

I2P Network Protocol Documentation

End-to-End Encryption
How client messages are end-to-end encrypted by the router

Network Database
Distributed storage and retrieval of information about routers and clients

Router Message Protocol
I2P is a message-oriented router. The messages sent between routers are defined by the I2NP protocol.

Transport Layer
The protocols for direct (point-to-point) router to router communication

Tunnels
Selecting peers, requesting tunnels through peers, encrypting and routing messages through these tunnels.

Other Router Topics
Tagline here

I2P Research Documentation

Academic Research
Tagline here

Academic Papers and Peer Review
Tagline here

Open Research Topics
Tagline here

I2P Metrics
Tagline here

Vulnerability Response Process
Tagline here

Technical Introduction

https://geti2p.net/en/docs/how/tech-intro

I2P is a scalable, self organizing, resilient packet switched anonymous network layer, upon which any number of different anonymity or security conscious applications can operate. Each of these applications may make their own anonymity, latency, and throughput tradeoffs without worrying about the proper implementation of a free route mixnet, allowing them to blend their activity with the larger anonymity set of users already running on top of I2P.

Applications available already provide the full range of typical Internet activities - **anonymous** web browsing, web hosting, chat, file sharing, e-mail, blogging and content syndication, newsgroups, as well as several other applications under development.

- Web browsing: using any existing browser that supports using a proxy.
- Chat: IRC, Jabber, I2P-Messenger.
- File sharing: I2PSnark, Robert, iMule, I2Phex, PyBit, I2P-bt and others.
- E-mail: susimail and I2P-Bote.
- Blog: using e.g. the pebble plugin or the distributed blogging software Syndie.
- Distributed Data Store: Save your data redundantly in the Tahoe-LAFS cloud over I2P.
- Newsgroups: using any newsgroup reader that supports using a proxy.

Unlike web sites hosted within content distribution networks like Freenet or GNUnet, the services hosted on I2P are fully interactive - there are traditional web-style search engines, bulletin boards, blogs you can comment on, database driven sites, and bridges to query static systems like Freenet without needing to install it locally.

With all of these anonymity enabled applications, I2P takes on the role of the message oriented middleware - applications say that they want to send some data to a cryptographic identifier (a "destination") and I2P takes care of making sure it gets there securely and anonymously. I2P also bundles a simple streaming library to allow I2P's anonymous best-effort messages to transfer as reliable, in-order streams, transparently offering a TCP based congestion control algorithm tuned for the high bandwidth delay product of the network. While there have been several simple SOCKS proxies available to tie existing applications into the network, their value has been limited as nearly every application routinely exposes what, in an anonymous context, is sensitive information. The only safe way to go is to fully audit an application to ensure proper operation and to assist in that we provide a series of APIs in various languages which can be used to make the most out of the network.

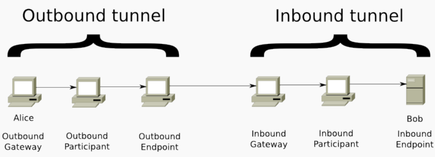
I2P is not a research project - academic, commercial, or governmental, but is instead an engineering effort aimed at doing whatever is necessary to provide a sufficient level of anonymity to those who need it. It has been in active development since early 2003 with one full time developer and a dedicated group of part time contributors from all over the world. All of the work done on I2P is open source and freely available on the website, with the majority of the code released outright into the public domain, though making use of a few cryptographic routines under BSD-style licenses. The people working on I2P do not control what people release client applications under, and there are several GPL'ed applications available (I2PTunnel, susimail, I2PSnark, I2P-Bote, I2Phex and others.). Funding for I2P comes entirely from donations, and does not receive any tax breaks in any jurisdiction at this time, as many of the developers are themselves anonymous.

Operation

Overview

To understand I2P's operation, it is essential to understand a few key concepts. First, I2P makes a strict separation between the software participating in the network (a "router") and the anonymous endpoints ("destinations") associated with individual applications. The fact that someone is running I2P is not usually a secret. What is hidden is information on what the user is doing, if anything at all, as well as what router a particular destination is connected to. End users will typically have several local destinations on their router - for instance, one proxying in to IRC servers, another supporting the user's anonymous webserver ("I2P Site"), another for an I2Phex instance, another for torrents, etc.

Another critical concept to understand is the "tunnel". A tunnel is a directed path through an explicitly selected list of routers. Layered encryption is used, so each of the routers can only decrypt a single layer. The decrypted information contains the IP of the next router, along with the encrypted information to be forwarded. Each tunnel has a starting point (the first router, also known as "gateway") and an end point. Messages can be sent only in one way. To send messages back, another tunnel is required.



Two types of tunnels exist: **"outbound" tunnels** send messages away from the tunnel creator, while **"inbound" tunnels** bring messages to the tunnel creator. Combining these two tunnels allows users to send messages to each other. The sender ("Alice" in the above image) sets up an outbound tunnel, while the receiver ("Bob" in the above image) creates an inbound tunnel. The gateway of an inbound tunnel can receive messages from any other user and will send them on until the endpoint ("Bob"). The endpoint of the outbound tunnel will need to send the message on to the gateway of the inbound tunnel. To do this, the sender ("Alice") adds instructions to her encrypted message. Once the endpoint of the outbound tunnel decrypts the message, it will have instructions to forward the message to the correct inbound gateway (the gateway to "Bob").

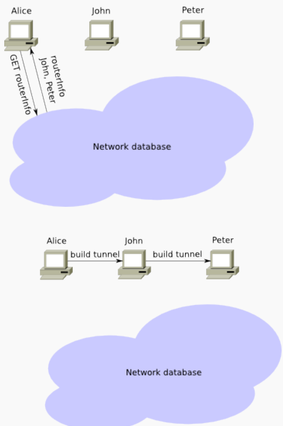
A third critical concept to understand is I2P's **"network database"** (or "netDb") - a pair of algorithms used to share network metadata. The two types of metadata carried are **"routerInfo"** and **"leaseSets"** - the routerInfo gives routers the data necessary for contacting a particular router (their public keys, transport addresses, etc), while the leaseSet gives routers the information necessary for contacting a particular destination. A leaseSet contains a number of "leases". Each of this leases specifies a tunnel gateway, which allows reaching a specific destination. The full information contained in a lease:Inbound gateway for a tunnel that allows reaching a specific destination

- Time when a tunnel expires
- Pair of public keys to be able to encrypt messages (to send through the tunnel and reach the destination).

Routers themselves send their routerInfo to the netDb directly, while leaseSets are sent through outbound tunnels (leaseSets need to be sent anonymously, to avoid correlating a router with his leaseSets).

We can combine the above concepts to build successful connections in the network.

To build up her own inbound and outbound tunnels, Alice does a lookup in the netDb to collect routerInfo. This way, she gathers lists of peers she can use as hops in her tunnels. She can then send a build message to the first hop, requesting the construction of a tunnel and asking that router to send the construction message onward, until the tunnel has been constructed.



When Alice wants to send a message to Bob, she first does a lookup in the netDb to find Bob's leaseSet, giving her his current inbound tunnel gateways. She then picks one of her outbound tunnels and sends the message down it with instructions for the outbound tunnel's endpoint to forward the message on to one of Bob's inbound tunnel gateways. When the outbound tunnel endpoint receives those instructions, it forwards the message as requested, and when Bob's inbound tunnel gateway receives it, it is forwarded down the tunnel to Bob's router. If Alice wants Bob to be able to reply to the message, she needs to transmit her own destination explicitly as part of the message itself. This can be done by introducing a higher-level layer, which is done in the streaming library. Alice may also cut down on the response time by bundling her most recent LeaseSet with the message so that Bob doesn't need to do a netDb lookup for it when he wants to reply, but this is optional.

And More on……https://geti2p.net/en/docs/how/tech-intro

Threat Model and Analysis

https://geti2p.net/en/docs/how/threat-model

What do we mean by “anonymous”?

Your level of anonymity can be described as "how hard it is for someone to find out information you don't want them to know?" - who you are, where you are located, who you communicate with, or even when you communicate. "Perfect" anonymity is not a useful concept here - software will not make you indistinguishable from people that don't use computers or who are not on the Internet. Instead, we are working to provide sufficient anonymity to meet the real needs of whomever we can - from those simply browsing websites, to those exchanging data, to those fearful of discovery by powerful organizations or states.

The question of whether I2P provides sufficient anonymity for your particular needs is a hard one, but this page will hopefully assist in answering that question by exploring how I2P operates under various attacks so that you may decide whether it meets your needs.

We welcome further research and analysis on I2P's resistance to the threats described below. More review of existing literature (much of it focused on Tor) and original work focused on I2P is needed.

Network Topology Summary

I2P builds off the ideas of many other systems, but a few key points should be kept in mind when reviewing related literature:
I2P is a free route mixnet - the message creator explicitly defines the path that messages will be sent out (the outbound tunnel), and the message recipient explicitly defines the path that messages will be received on (the inbound tunnel).
I2P has no official entry and exit points - all peers fully participate in the mix, and there are no network layer in- or out-proxies (however, at the application layer, a few proxies do exist)
I2P is fully distributed - there are no central controls or authorities. One could modify some routers to operate mix cascades (building tunnels and giving out the keys necessary to control the forwarding at the tunnel endpoint) or directory based profiling and selection, all without breaking compatibility with the rest of the network, but doing so is of course not necessary (and may even harm one's anonymity).

We have documented plans to implement nontrivial delays and batching strategies whose existence is only known to the particular hop or tunnel gateway that receives the message, allowing a mostly low latency mixnet to provide cover traffic for higher latency communication (e.g. email). However we are aware that significant delays are required to provide meaningful protection, and that implementation of such delays will be a significant challenge. It is not clear at this time whether we will actually implement these delay features.

In theory, routers along the message path may inject an arbitrary number of hops before forwarding the message to the next peer, though the current implementation does not.

The Threat Model

Second page title? “The Threat Model” Remove?

I2P design started in 2003, not long after the advent of [Onion Routing], [Freenet], and [Tor]. Our design benefits substantially from the research published around that time. I2P uses several onion routing techniques, so we continue to benefit from the significant academic interest in Tor.

Taking from the attacks and analysis put forth in the anonymity literature (largely Traffic Analysis: Protocols, Attacks, Design Issues and Open Problems), the following briefly describes a wide variety of attacks as well as many of I2Ps defenses. We update this list to include new attacks as they are identified.

Included are some attacks that may be unique to I2P. We do not have good answers for all these attacks, however we continue to do research and improve our defenses.

In addition, many of these attacks are significantly easier than they should be, due to the modest size of the current network. While we are aware of some limitations that need to be addressed, I2P is designed to support hundreds of thousands, or millions, of participants. As we continue to spread the word and grow the network, these attacks will become much harder.

The network comparisons and "garlic" terminology pages may also be helpful to review.

Brute Force Attacks

A brute force attack can be mounted by a global passive or active adversary, watching all the messages pass between all of the nodes and attempting to correlate which message follows which path. Mounting this attack against I2P should be nontrivial, as all peers in the network are frequently sending messages (both end to end and network maintenance messages), plus an end to end message changes size and data along its path. In addition, the external adversary does not have access to the messages either, as inter-router communication is both encrypted and streamed (making two 1024 byte messages indistinguishable from one 2048 byte message).

However, a powerful attacker can use brute force to detect trends - if they can send 5GB to an I2P destination and monitor everyone's network connection, they can eliminate all peers who did not receive 5GB of data. Techniques to defeat this attack exist, but may be prohibitively expensive (see: Tarzan's mimics or constant rate traffic). Most users are not concerned with this attack, as the cost of mounting it are extreme (and often require illegal activity). However, the attack is still possible, for example by an observer at a large ISP or an Internet exchange point. Those who want to defend against it would want to take appropriate countermeasures, such as setting low bandwidth limits, and using unpublished or encrypted leasesets for I2P Sites. Other countermeasures, such as nontrivial delays and restricted routes, are not currently implemented.

As a partial defense against a single router or group of routers trying to route all the network's traffic, routers contain limits as to how many tunnels can be routed through a single peer. As the network grows, these limits are subject to further adjustment. Other mechanisms for peer rating, selection and avoidance are discussed on the peer selection page.

Timing Attacks

A I2P's messages are unidirectional and do not necessarily imply that a reply will be sent. However, applications on top of I2P will most likely have recognizable patterns within the frequency of their messages - for instance, an HTTP request will be a small message with a large sequence of reply messages containing the HTTP response. Using this data as well as a broad view of the network topology, an attacker may be able to disqualify some links as being too slow to have passed the message along.

This sort of attack is powerful, but its applicability to I2P is non obvious, as the variation on message delays due to queuing, message processing, and throttling will often meet or exceed the time of passing a message along a single link - even when the attacker knows that a reply will be sent as soon as the message is received. There are some scenarios which will expose fairly automatic replies though - the streaming library does (with the SYN+ACK) as does the message mode of guaranteed delivery (with the DataMessage+DeliveryStatusMessage).

Without protocol scrubbing or higher latency, global active adversaries can gain substantial information. As such, people concerned with these attacks could increase the latency (using nontrivial delays or batching strategies), include protocol scrubbing, or other advanced tunnel routing techniques, but these are unimplemented in I2P.

References: Low-Resource Routing Attacks Against Anonymous Systems

Intersection Attacks

Intersection attacks against low latency systems are extremely powerful - periodically make contact with the target and keep track of what peers are on the network. Over time, as node churn occurs the attacker will gain significant information about the target by simply intersecting the sets of peers that are online when a message successfully goes through. The cost of this attack is significant as the network grows, but may be feasible in some scenarios.

In summary, if an attacker is at both ends of your tunnel at the same time, he may be successful. I2P does not have a full defense to this for low latency communication. This is an inherent weakness of low-latency onion routing. Tor provides a similar disclaimer.

- Partial defenses implemented in I2P
- strict ordering of peers
- peer profiling and selection from a small group that changes slowly
- Limits on the number of tunnels routed through a single peer
- Prevention of peers from the same /16 IP range from being members of a single tunnel
- For I2P Sites or other hosted services, we support simultaneous hosting on multiple routers, or multihoming

Even in total, these defenses are not a complete solution. Also, we have made some design choices that may significantly increase our vulnerability:
- We do not use low-bandwidth "guard nodes”
- We use tunnel pools comprised of several tunnels, and traffic can shift from tunnel to tunnel.

Tunnels are not long-lived; new tunnels are built every 10 minutes.
Tunnel lengths are configurable. While 3-hop tunnels are recommended for full protection, several applications and services use 2-hop tunnels by default.In the future, it could for peers who can afford significant delays (per nontrivial delays and batching strategies). In addition, this is only relevant for destinations that other people know about - a private group whose destination is only known to trusted peers does not have to worry, as an adversary can't "ping" them to mount the attack.

Reference: One Cell Enough

And More on……https://geti2p.net/en/docs/how/threat-model

Denial of service Attacks **Tagging Attacks** **Partitioning Attacks** **Predecessor Attacks**

Harvesting Attacks **Identification Through Traffic Analysis** **Sybil Attacks** **Buddy Exhaustion Attacks**

Cryptographic Attacks **Floodfill Anonymity Attacks** **Other Network Database Attacks**

Central Resource Attacks **Development Attacks** **Implementation Attacks (bugs)**

Other Defenses > Blocklists

Specifications

This page provides the specifications for various components of the I2P network and router software. These are living documents, and the specifications are updated as modifications are made to the network and software. The proposal documents that track changes to these specifications can be viewed here.

•"Last updated" is the last date when the specification given within a document was altered in any way, except for changes to the "accurate for" information.
•The "accurate for" column gives the version of the I2P network and reference Java implementation that the document is verified to be valid for. Because the documents are usually only updated when changes are made, the listed versions can sometimes be several releases behind. This does not mean that documents with old listed versions are necessarily inaccurate, but small differences may creep in during the course of development. Periodic reviews are conducted to update the "accurate for" information.

The I2P Project is committed to maintaining accurate, current documentation. If you find any inaccuracies in the documents linked below, please enter a ticket identifying the problem.

Title	Category	Last updated	Accurate for	Link
Common structures Specification	Design	2021-04	0.9.49	HTML I TXT
Low-level Cryptography Specification	Design	2020-09	0.9.47	HTML I TXT
Tunnel Creation Specification	Design	July 2019	0.9.41	HTML I TXT
Tunnel Message Specification	Design	2021-01	0.9.49	HTML I TXT
NTCP 2	Transports	2021-03	0.9.50	HTML I TXT
SSU Protocol Specification	Transports	2021-06	0.9.50	HTML I TXT
Datagram Specification	Protocols	February 2019	0.9.39	HTML I TXT
ECIES-X25519-AEAD-Ratchet	Protocols	2020-11	0.9.47	HTML I TXT
Encrypted LeaseSet Specification	Protocols	June 2019	0.9.41	HTML I TXT
I2CP Specification	Protocols	2020-11	0.9.48	HTML I TXT
I2NP Specification	Protocols	2021-07	0.9.51	HTML I TXT
Streaming Library Specification	Protocols	May 2020	0.9.46	HTML I TXT
Access Filter Format Specification		April 2019	0.9.40	HTML I TXT
Addressbook Subscription Feed Commands		2021-01	0.9.49	HTML I TXT
B32 for Encrypted Leasesets		2020-08	0.9.47	HTML I TXT
Blockfile and Hosts Database Specification		2020-09	0.9.47	HTML I TXT
Configuration File Specification		March 2020	0.9.45	HTML I TXT
ECIES-X25519 Router Messages		None	None	HTML I TXT
ECIES-X25519 Tunnel Creation		None	None	HTML I TXT
GeoIP File Specification		December 2013	0.9.9	HTML I TXT
Plugin Specification		November 2019	0.9.43	HTML I TXT
Red25519 Signature Scheme		2020-08	0.9.47	HTML I TXT
Software Update Specification		June 2021	0.9.51	HTML I TXT

Other Specification Documents

These will eventually be migrated to the new specifications system.

- ElGamal/AES+SessionTags
- NTCP
- I2PControl
- SAM v3
- BOB
- Bittorrent
- Naming and Address Book

Protocol Stack Chart

Here is the protocol stack for I2P. See also the Index to Technical Documentation.

Each of the layers in the stack provides extra capabilities. The capabilities are listed below, starting at the bottom of the protocol stack.

- **Internet Layer:**

IP: Internet Protocol, allow addressing hosts on the regular internet and routing packets across the internet using best-effort delivery.

- **Transport Layer:**
  TCP: Transmission Control Protocol, allow reliable, in-order delivery of packets across the internet.
  UDP: User Datagram Protocol, allow unreliable, out-of-order delivery of packets across the internet.
- **I2P Transport Layer:** provide encrypted connections between 2 I2P routers. These are not anonymous yet, this is strictly a hop-to-hop connection. Two protocols are implemented to provide these capabilities. NTCP builds on top of TCP, while SSU uses UDP.
  NTCP: NIO-based TCP
  SSU: Secure Semi-reliable UDP
- **I2P Tunnel Layer:** provide full encrypted tunnel connections.
  Tunnel messages: tunnel messages are large messages containing encrypted I2NP (see below) messages and encrypted instructions for their delivery. The encryption is layered. The first hop will decrypt the tunnel message and read a part. Another part can still be encrypted (with another key), so it will be forwarded.
  I2NP messages: I2P Network Protocol messages are used to pass messages through multiple routers. These I2NP messages are combined in tunnel messages.
- **I2P Garlic Layer:** provide encrypted and anonymous end-to-end I2P message delivery.
  I2NP messages: I2P Network Protocol messages are wrapped in each other and used to ensure encryption between two tunnels and are passed along from source to destination, keeping both anonymous.

The following layers are strictly speaking no longer part of the I2P Protocol stack, they are not part of the core 'I2P router' functionality. However, each of these layers adds additional functionality, to allow applications simple and convenient I2P usage.

- **I2P Client Layer:** allow any client to use I2P functionality, without requiring the direct use of the router API.
  I2CP: I2P Client Protocol, allows secure and asynchronous messaging over I2P by communicating messages over the I2CP TCP socket.
- **I2P End-to-end Transport Layer:** allow TCP- or UDP-like functionality on top of I2P.
  Streaming Library: an implementation of TCP-like streams over I2P. This allows easier porting of existing applications to I2P.
  Datagram Library: an implementation of UDP-like messages over I2P. This allows easier porting of existing applications to I2P.
- **I2P Application Interface Layer:** additional (optional) libraries allowing easier implementations on top of I2P.
  I2PTunnel
  SAM/SAMv2/SAMv3(*), BOB
  I2P Application Proxy Layer: proxy systems.
  HTTP Client/Server, IRC Client, SOCKS, Streamr

Finally, what could be considered the **'I2P application layer'**, is a large number of applications on top of I2P. We can order this based on the I2P stack layer they use.

- **Streaming/datagram applications**: i2psnark, Syndie, i2phex…
- **SAM/BOB applications**: IMule, i2p-bt, i2prufus, Robert…
- **Other I2P applications**: Syndie, EepGet, plugins...
- **Regular applications**: Jetty, Apache, Monotone, CVS, browsers, e-mail...

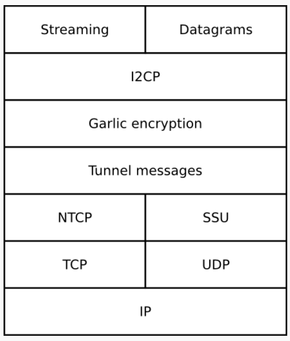


Figure 1: The layers in the I2P Network stack.

* Note: SAM/SAMv2 can use both the streaming lib and datagrams.

Development Roadmaps

Development Roadmaps

This page was last updated in 2021-09.

This is the official project roadmap for the desktop and Android Java I2P releases only. Some related tasks for resources such as the website and plugins may be included.

For details and discussion on specific items, search on gitlab or zzz.i2p. For contents of past releases, see the release notes. For other project goals, see the meeting notes.

We do not maintain separate unstable and stable branches or releases. We have a single, stable release path. Our normal release cycle is 13 weeks, with releases in February, May, August, and November.

Older releases are at the bottom of the page.

0.9.48

Released: December 1, 2020

- ECIES router tunnel build record
- Avoid old DSA-SHA1 routers
- Block same-country connections when in hidden mode
- Deprecate BOB
- Drop support for Xenial
- Ratchet efficiency improvements and memory reduction
- Randomize SSU intro key
- Enable system tray for Linux KDE and LXDE
- More SSU performance improvements
- Continue transition to Git
- Operators guides for reseed services
- Windows Installer "Install as Windows Service" bugfixes and improvements.
- Implement controlled vocabuary as part of Information Architecture improvements
- Alternate destination header/meta tag for web sites offering I2P mirrors
- Snark in the Browser: Use torrents as alternates sources for resources embedded in an I2P Site
- Snark in the Browser: Demo a torrent-backed web page
- Finish ji2p-cluster which adds the k8s part of the code
- Publish reasonable contact information for infrastructure admins

0.9.49

Released: February 17, 2021

- SSU send individual fragments
- SSU Westwood+
- SSU fast retransmit
- SSU fix partial acks
- ECIES router encrypted messages
- Start rekeying routers to ECIES
- Start work on new tunnel build message (proposal 157)
- More SSU performance improvements
- i2psnark webseed support
- Start work on i2psnark hybrid v2 support
- Move web resources to wars
- Move resources to jars
- Fix Gradle build
- Hidden mode fixes
- Change DoH to RFC 8484
- Fix "Start on Boot" support on Android
- Add support for copying b32 addresses from the tunnels panel on I2P for Android client
- Add SAMv3 Support to I2P for Android
- Revise CSS on the default I2P Site to resemble console Light theme
- Document setup/configuration of default I2P site on the project site
- Add icons and symbols used in I2P router console Light theme to router console Dark theme
- Complete transition to Git
- Donation page redesign and backend (deployment)
- Review and update information about VCS, Code Repositories, and Mirrors across the entire website.

And more on.....https://geti2p.net/en/get-involved/roadmap