Gosling: Build Anonymous, Secure, and Metadata-Resistant Peer-to-Peer Applications using Tor Onion Services

morgan (she/they)
morgan@blueprintforfreespeech.net

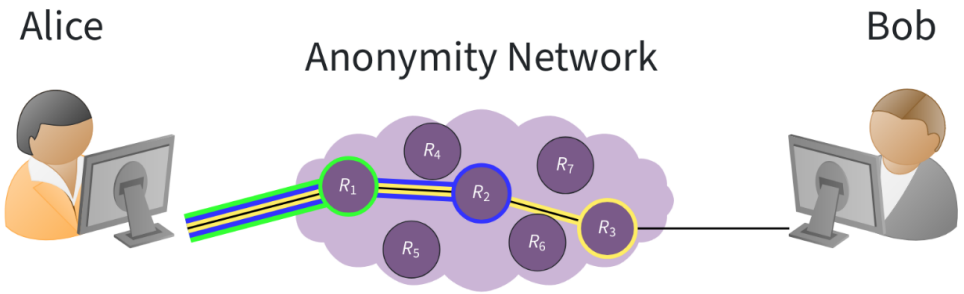
Introductions

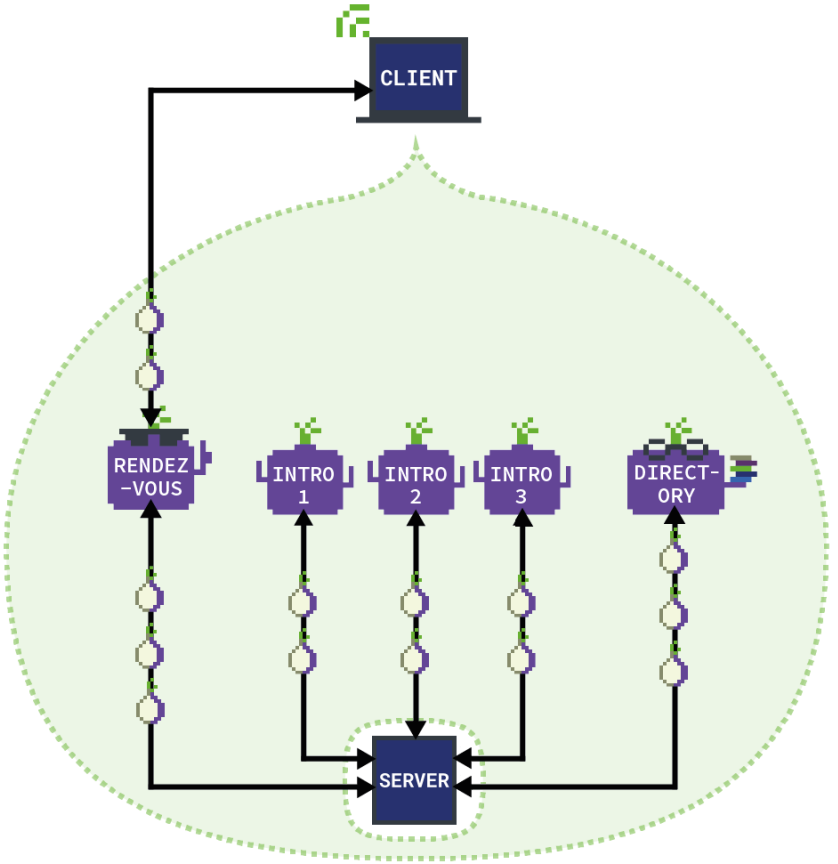
- Software Developer working for Blueprint for Free Speech
  - Develop and maintain:
    - Gosling Rust crate
    - Ricochet-Refresh instant messenger
- Browser Team Lead at the Tor Project
  - Lead a team of software engineers developing Tor Browser, Tor Browser for Android, and Mullvad Browser

What is Gosling and What Does it Do?

- Rust library for building realtime peer-to-peer applications with the following properties:
  - End-to-End Encryption with Forward Secrecy
  - Anonymity
  - Hole Punching
  - Censorship Circumvention
  - Client Authentication
  - Metadata Resistance
  - Optional Application-Specific Authorisation Extensions

Built on Tor and Onion-Services





How Does It Work?

- Every user has an id, an onion service id:
  - 6l62fw7tqctlu5fesdqukvpoxezkaxbzllrafa2ve6ewuhzphxczsjyd.onion
- This id serves dual purpose:
  - a destination (an onion service) for connecting peers
  - an identifier used for authenticating clients when connecting to other peers (onion services)
- Each peer hosts an onion service, which other peers may connect to

End-to-End Encryption

- All communications between peers are end-to-end encrypted with forward secrecy

Anonymous

- Peers do not need to know each other's real IP address to communicate
  - The *required* usage of Tor makes it impossible to accidentally leak the user's IP address through usage of Gosling
- No central registrar, no sign-ups, no need to associate a peer with a phone number or other personally identifying information

Hole Punching

- Peers do not need to have publicly accessible open ports for other peers to connect to them
- Peers only need to make outgoing connections to the Tor network

Censorship-Circumvention

- There is no authority or gate-keeping on who can or cannot run an onion service which could block a peer from receiving connections
- All peer-to-peer traffic stays within the Tor network
- If you can connect to the Tor network, then you have full access to other peers
    - If not, then you can use pluggable-transports and bridges to connect

Wait A Second...

Client Authentication

- In the Tor network, onion services are self-authenticating, but the clients connecting to them are not!
- Normally, clients do not have onion service ids
- **Problem**: This is supposed to be an authenticated peer-to-peer system! How does an onion service verify connecting clients are who they say they are?

Client Authentication (cont)

- Clients using Gosling make a claim that they are the owner of some onion service id
- An onion service id is actually *public* key
- If a user connects to your service, and claim they are the owner of onion service id abc...234.onion, what they are *really* claiming is they control the *private* key which maps to this *public* key
- To verify a Gosling client is telling the truth, we ask them to sign a carefully crafted message with their *private* key, and the onion service verifies the signature using the client's provided *public* key

Optional Application-Specific Extensions

- Protocol has some flexibility to allow for some additional application-specific authentication barriers or requirements such as:
  - Peer block/allow lists
  - Shared secrets/invite codes
  - Proof-of-Work/Stake schemes

Metadata-Resistance

- Communication contents are fully end-to-end encrypted, and stay entirely within the Tor Network
- Clients' real identities are unknown to each other
- No way to determine who peers have connected to; no way to generate a ‘social graph’ of peers
- Sounds great, so what's the problem?

An Interesting Property of Onion Services

- *Anyone* can attempt to connect to your onion service and determine if it is currently online
    - A profile of an onion service's online/offline status can be built by repeatedly doing this
- *Probably* not a problem if your onion service is a website or some other service that is meant to be always online
- *Maybe* a problem when the onion service is running in a personal computing environment because a device's online/offline status maps pretty closely to human user using/not using their device
- Tor's onion service client authorisation feature does not fix this problem, only decreases the time resolution

Cyber-Stalking

- Malicious 3rd parties can 'cyber-stalk' users by simply trying to connect to them to identify their online/offline status
- Quite malicious 3rd parties could *potentially* discover your guard node (e.g. by simultaneously knocking guard nodes offline and cyber-stalking users)
- Quite malicious+capable 3rd parties can therefore de-anonymise users if they can also see who a guard node is connected to with various methods (e.g. legal wiretaps, running malicious guard nodes, etc)

So keep it secret...

- If you only share your onion service id privately with your trusted contacts, then you do not need to worry about this metadata leak
    - version 3 onion services are not enumerable or discoverable by 3rd parties

What We Would Like...

- Authenticated peers should be able to connect to and communicate with each other
- Unauthenticated peers should not be able to determine each others online/offline status
- Unauthenticated peers should be able to become authenticated

- **Problem 1:** You can't do all three at once...
- **Problem 2**: Core application functionality potentially *requires* sharing your onion service id with untrustworthy users

Gosling's Solution:

- Spread our onion service's responsibility across more onion services:
  - 1 'identity' service
  - N 'endpoint' services (one for each authenticated peer)
- Identity service acts as the gatekeeper for authenticating and accepting or rejecting new peers and distributing endpoint service credentials
- Endpoint services are where the actual peer-to-peer application-specific communications happens

1. Gosling Specification: https://gosling.technology/gosling-spec.xhtml

Gosling's Solution (cont):

- This division of labor across multiple onion services means the identity service is not *required* for core functionality, meaning it can be disabled if there is no need to acquire more peers
- Disabling the identity service prevents unauthenticated users from cyber-stalking you, even if you have shared your onion service Id publicly
- Ultimately about giving users control over their visibility

Documentation:

Design Document:
- **https://gosling.technology/design-doc.xhtml**

Protocol Specification:
- **https://gosling.technology/gosling-spec.xhtml**

Usage Guide:
- **https://gosling.technology/usage-guide.xhtml**

Example Apps:
- **https://github.com/blueprint-freespeech/gosling/tree/main/source/examples**

Crate docs
- **https://docs.rs/gosling/latest/gosling/**

Example Chat Applications (Rust and C++)

```
Welcome to example_chat_rs!
Type help for a list of commands
> help
Available commands:
  help COMMAND              Print help for COMMAND
  init-context              Initialise the gosling context
  start-identity            Start the identity onion-service
  stop-identity             Stop the identity onion-service
  request-endpoint          Connect to identity onion-service and request an endpoint
  start-endpoint            Start an endpoint onion-service
  stop-endpoint             Stop an endpoint onion-service
  connect-endpoint          Connect to a peer's endpoint onion-service
  drop-peer                 Drop a connection to a peer
  list-peers                List all of the currently connected peers
  chat                      Send a message to a connected peer
  exit                      Quits the program
================================================================================
> init-context
```

Defending Free Speech – Blueprint Projects

Open-source tools and support for those who speak up

No SLAPP Anlaufstelle

Germany's first support center against legal intimidation

SLAPP = Strategic Lawsuits Against Public Participation

40+ cases · 18 legal experts · since May 2024

- **Case Triage**
- **Lawyer Connections**
- **Pattern Analysis & Documentation**
- **Training & Prevention**

Partners: FragDenStaat, RSF Germany, DJV, dju in ver.di + others

Advocating EU Anti-SLAPP Directive noslapp.de

STAIRS

AI-powered whistleblower support – open source

"First-responder chatbot for people facing retaliation"

5-Country EU Consortium · Project running 2025-2027

- **Multilingual legal chatbot (EU Directive 2019/1937)**
- **End-to-end encryption & anonymization**
- **Mental health resources & referrals**
- **Case management for CSOs**

Stack: Laravel · Vue · Python · Docker

Funded by EU CERV Programme stairsproject.eu

Defending Free Speech – Blueprint Projects

Open-source tools and support for those who speak up

Ricochet-Refresh

Anonymous and private instant messaging

- **Under Active Development:**
  - **Integrating Gosling**
  - **Migrating to wxWidgets**
  - **Improving usability**
- **Platforms: Windows, macOS, Linux**

Stack: Rust · C++

Funded by NLnet

ricochetrefresh.net

tor-interface crate

Easy Tor integration in Rust using c-tor or Arti

- **Simple API supporting:**
  - **Bootstrapping**
  - **Pluggable-Transports & Bridges**
  - **Connecting to endpoints**
  - **Stream isolation**
  - **Hosting onion services**

Funded by NLnet+RIPE

crates.io/crates/tor-interface