Welcome to Gosling!

Blueprint for Free Speech
media@blueprintforfreespeech.net

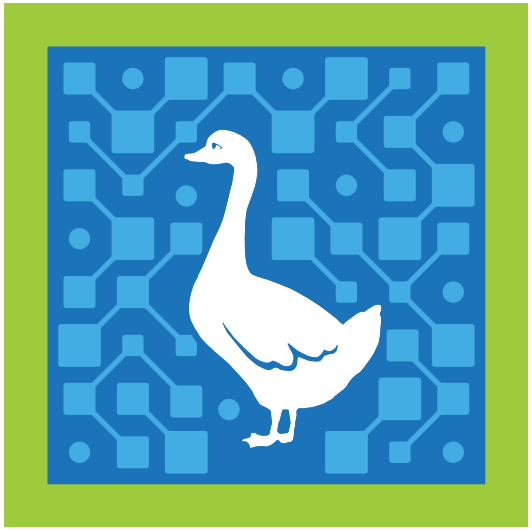
- Acknowledgment
- Introducing Blueprint for Free Speech people on this project – and why are we building Gosling?
- All about Gosling

blueprint for FREE SPEECH

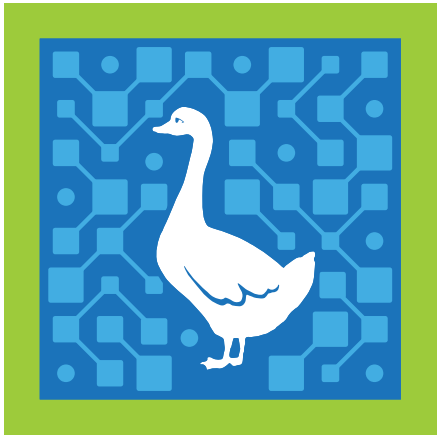
We acknowledge with gratitude the support of

RIPE NCC

In further development of …



blueprint for FREE SPEECH



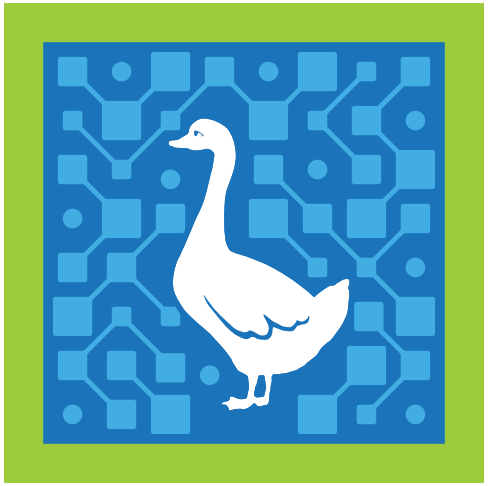
Why is Blueprint developing Gosling?

To improve the underlying infrastructure that provides anonymity/privacy for everyone but especially those who need and want it more...

Journalists and whistleblowers, human rights workers and sources, academic researchers, domestic violence victims etc

blueprint for FREE SPEECH

RICOCHET REFRESH



What does Blueprint for Free Speech do?

Supports the right to freedom of expression globally

Through…

- Building tools that support this, particularly the relationship between journalists and whistleblowers
- And policy advocacy
- And law reform

PATFox

Pioneering anti-SLAPP Training for Freedom of Expression

Expanding Anonymous Tipping

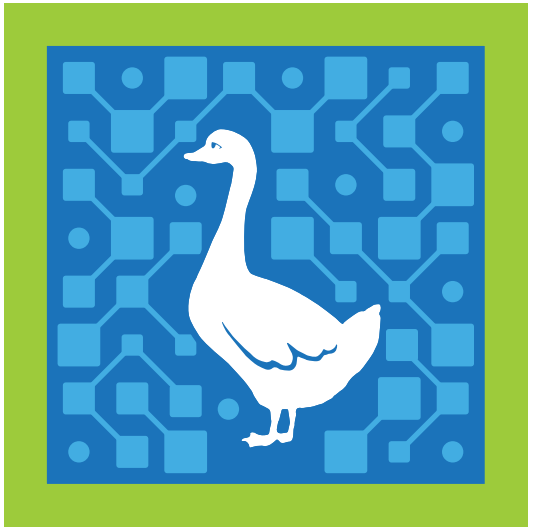
WIDELY EXPANDING ANONYMOUS TIPPING TECHNOLOGY DEPLOYMENT, OPERATION, AND TRUSTWORTHINESS TO COMBAT CORRUPTION IN EASTERN AND SOUTHERN EUROPE

Thank you

For further information, please see our website
http://blueprintforfreespeech.net

Adding Arti Backend(s) Support to the Gosling Library

richard (they/them)
richard@blueprintforfreespeech.net



What is Gosling and What Does it Do?

- Rust library which provides peer-to-peer connectivity with the following features built-in:
  - End-to-End Encrypted
  - Anonymous
  - Hole Punching
  - Censorship Circumvention
  - Client Authentication
  - Optional Application-Specific Extensions
  - Metadata Resistance

How Does It Work?

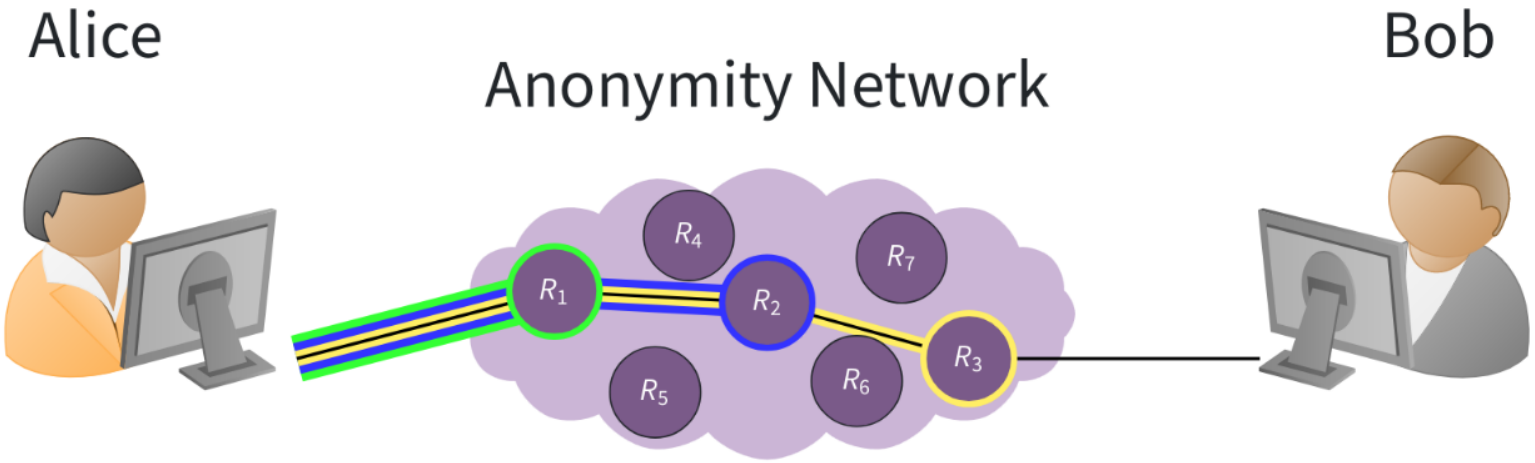
- Each user has a unique id like:
    - 6l62fw7tqctlu5fesdqukvpoxezkaxbzllrafa2ve6ewuhzphxczsjyd
- Users have only to share their id with other users, successfully complete a handshake, and they can connect to and send traffic to each other with all the afore-mentioned properties!

Right.. But How Does It Work?

- Built on Tor and Tor Onion Services

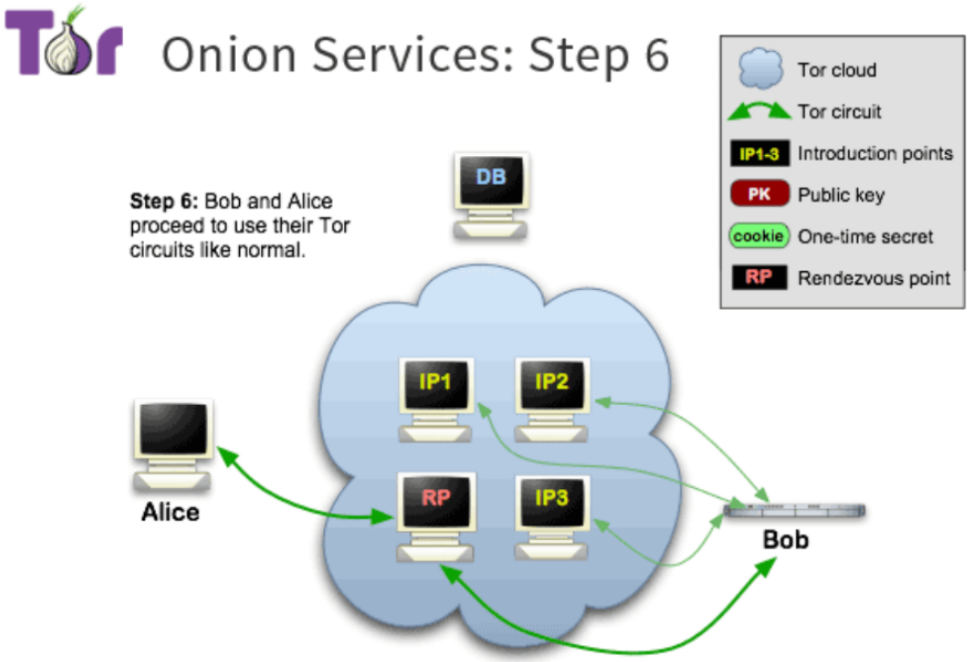
Tor

- Tor Network is a community of relay operators, each running tor aka little-t tor, c-tor, or the legacy tor daemon
- Users create circuits to their destination within the Tor Network:
    - 1st Hop - Guard Relay: knows IP address of user and guard relay
    - 2nd Hop - Middle Relay: knows the guard relay and the exit relay
    - 3rd Hop - Exit Relay: knows middle relay, final destination and contents of traffic



Onion Services

- Onion Service traffic never leaves the Tor Network
  - Onion Service defines a set of introduction points within the Tor Network
  - Onion Services registers these introduction points in a distributed database in the Tor Network
  - Client connects to one of these introduction points, and negotiates a rendezvous point on another relay
  - Client + Onion Service each create circuits to the rendezvous point and begin talking



How Does it Work (cont)

- 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 Encrypted

- All communications between peers are end-to-end encrypted

Anonymous

- Peers do not need to know each other's 'real' IP address to communicate

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

Censorship-Circumvention

- There is no centralised ‘registrar’ of Onion Services which can 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
- (Maybe a big ‘if’)

Censorship-Circumvention (cont)

- Suppose you are in a place which blocks Tor such as:
    - China, Iran, Russia
    - Schools, Universities, Libraries
    - Offices, Government Buildings
- We can use pluggable transports to circumvent the block!
- Pluggable transports disguise your traffic as something else
- For example:
    - Snowflake[1] disguises your traffic as WebRTC

1. Snowflake: https://gitlab.torproject.org/tpo/anti-censorship/pluggable-transports/snowflake

Wait A Second...

- So you may be thinking something like: "Ok, so you have a library which routes your traffic through the Tor Network and inherits all its features. Good job, so what?"
- Bear with me

Authenticated

- Thanks to clever cryptography (*hand waving*), Onion Services are self-authenticating
- But clients are not, you do not need any authentication to connect to an Onion Service
- Clients do not have Onion Service Ids
- **Problem**: This is supposed to be a peer-to-peer system! How does an Onion Service verify connecting clients are who they say they are?

Authenticated (cont)

- If a user connects to your service, and claim they are the owner of onion service id abcd...234.onion, what they are *really* claiming is they control the *private* key which maps to the *public* key which is encoded in their onion service id.
- To verify the client is telling the truth, we ask them to sign a (carefully crafted) message[1] with their *private* key, and the onion service verifies the signature using the client's provided *public* key (derived from their claimed onion service id)

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

Optional Application-Specific Extensions

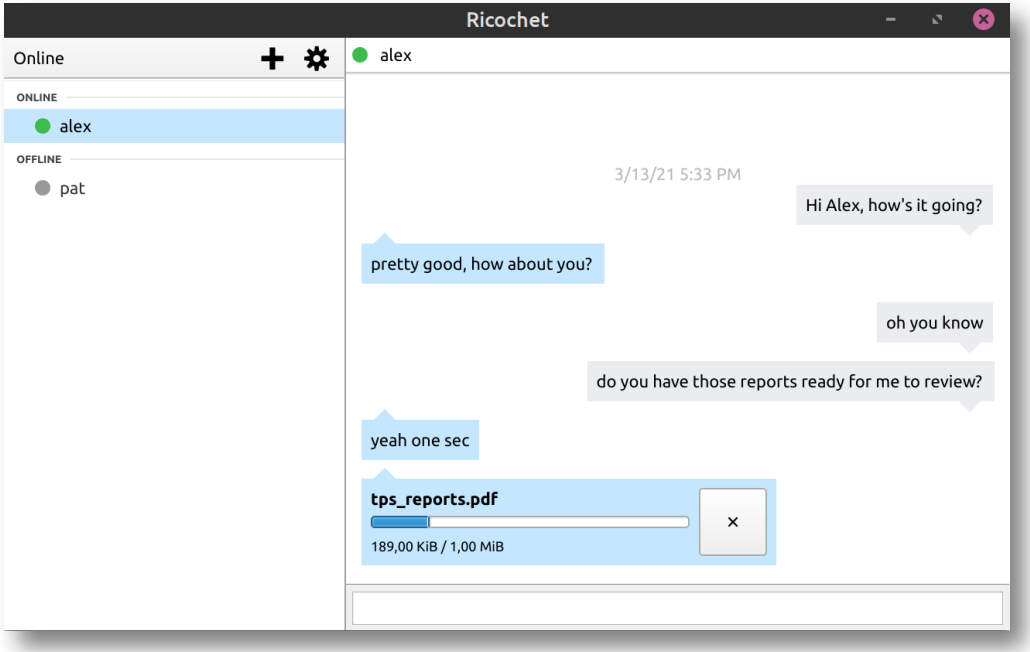
- Protocol has some flexiblity to allow for some additional application-specific authentication barriers or requirements such as:
    - Peer block/allow lists
    - Shared secrets/invite codes
    - Proof-of-Work/Stake schemes

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?

Some History: Ricochet-Refresh

- Peer-to-peer instant messenger via tor onion services
- Anonymous chat + file transfer
- Similar peer (contact/friend) authentication mechanism as described previously
- At least one of the peers must be running an Onion Service for the other peer to connect to in-order to chat



An Interesting Property of Onion Services

- Anyone (authenticated peer or not) can attempt to connect to your Onion Service and determine if it is currently online
- Therefore, a profile of the Onion Service's online/offline status can be built by repeatedly doing this
- Not really a big deal if your Onion Service is for a website or some other service that is meant to be always online
- *Kind* of a big deal when that Onion Service is running in a personal computing environment because PC online/offline status maps pretty closely to human user using/not using their computer

Whoops, Metadata Leak!

- Malicious 3rd parties can easily 'cyber-stalk' users by simply trying to connect to them
- Quite malicious 3rd parties could *also* discover your guard node by simultaneously knocking guard nodes offline and cyber-stalking users
- Quite malicious+capable 3rd parties could de-anonymise users if they can see who a guard node is connected to (using wiretaps for example, or running a malicious guard node and getting lucky)

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

- You can’t do all three at once

Gosling's Solution

- Spread a peer's Onion Service's responsibility across more Onion Services:
    - One 'identity' service
    - N 'endpoint' services (one for each authenticated peer)
- Identity service acts as the gatekeeper for accepting new peers and distributing endpoint service credentials
- Endpoint services are where actual peer-to-peer communications happen

Implications and Trade-Offs

- The public identity service is not required for application functionality if you have collected enough peers
    - Identity services may be optionally disabled (depending on the application)
- Access by an authenticated peer may be revoked by simply no longer running their associated endpoint service
    - Endpoint services may *also* be optionally disabled if you want to appear offline even to your peers

Tor Integration in Gosling

- The **gosling** crate gets its Tor functionality from the **tor-interface** crate (which we also maintain)
- **tor-interface** defines a **TorProvider** trait which requires conforming implementations to implement a certain set of functions related to connecting to the Tor Network, creating and connecting to Onion Services, etc.
- Currently we have 2 complete **TorProvider** implementations:
    - **mock_tor_client**
    - **legacy_tor_client**

Tor Integration: mock_tor_client

- Minimal local and in-process **TorProvider** for testing
- Never reaches the Tor Network
- Internet access not required
- Invaluable for unit and fuzz testing the **gosling** protocol crate and any protocol which may use **gosling** at its foundation

Tor Integration: legacy_tor_client

- Launches and owns a local c-tor process
- Managed via the control port protocol
- Just a very standard Tor controller implementation which many other tor-using applications have had to implement for themselves

Arti

- Arti (A Rust Tor implementation) is an in-progress re-implementation of c-tor in Rust being developed by the Tor Project's Network Team.
- c-tor is currently in maintenance-mode, where possible no new functionality is being added
- Long-term goal to completely replace c-tor both in client software (such as Tor Browser, Onion Share, Ricochet-Refresh, cwtch, etc) and as network relays with Arti
- There are currently three ways to use Arti from client software:
    - The arti-client Rust crate (library)[1]
    - The Arti binary (c-tor's eventual replacement)[2]
    - TorVPN (Android app/service)[3]

1. arti-client: https://crates.io/crates/arti-client
2. arti: https://crates.io/crates/arti
3. TorVPN: https://gitlab.torproject.org/tpo/applications/vpn

Tor Integration: arti_client_tor_client

- Integrates the **arti-client** crate directly, in-process
- Not yet feature complete, currently missing:
    - Onion Service Client Authorisation
- **arti-client** *should* have all our required features in the 0.19.0 release next month
- We developed a few minor feature and bug-fix patches in the just released version 0.18.0
- We expect this portion to be initially complete by July, though there will likely be a long tail of bug-fixes as **arti-client** is likely to periodically break compatibility before 1.0

Tor Integration: arti_daemon_tor_client

- Will be similar to our current **legacy_tor_client TorProvider:**
  - Out-of-process Arti
  - Outgoing connections via local SOCKS5 proxy
  - Communications via new JSON-RPC[1] based RPC protocol[2]
- RPC system and first APIs are being developed now in **arti**, so we expect our implementation work to begin in June.

1. json-rpc: https://www.jsonrpc.org/specification
2. rpc: https://gitlab.torproject.org/tpo/core/arti/-/blob/main/doc/dev/notes/rpc-meta-draft.md

Tor Integration: tor_vpn_tor_client

- Arti-based VPN client for Android
- Still in early stages of development
- First public alpha is scheduled for Q4 of 2024
- Plan to begin working on this **TorProvider** backend Summer of 2024
- Expecting a lot of interesting Android-specific challenges

Why Are We Doing This Work?

- Improved privacy guarantees for Ricochet-Refresh
    - Ricochet-Refresh v4.0 will use Gosling
- Future-proofing Gosling
    - c-tor is going away in a few years
- Bring Ricochet-Refresh to mobile
    - TorVPN makes this a realistic possibility
- Make developing privacy-preserving peer-to-peer applications as easy as possible

Links

- Blueprint For Free Speech: **https://blueprintforfreespeech.net**
- Ricochet-Refresh
  - website: **https://ricochetrefresh.net**
  - github: **https://github.com/blueprint-freespeech/ricochet-refresh**
- Gosling
  - website: **https://gosling.technology/**
  - github: **https://github.com/blueprint-freespeech/gosling**