README Best Practices for tor-go

Contents

Research date: 2026-02-17

Executive Summary

Analysis of 15+ successful Go library READMEs (wireguard-go, age, mkcert, quic-go, kcp-go, bine, caddy, syncthing, and others) reveals that the best security-sensitive Go library READMEs are **short, factual, and confident**. They say less, not more. They state what the project does, show it working, and link to godoc for everything else.

The optimal README for tor-go should be **under 100 lines**, follow the "library landing page" archetype (not a documentation page), and avoid every common security-project anti-pattern: no disclaimers, no apologies, no comparison tables, no roadmaps, no Tor explainers.

Recommended README Blueprint

Section Order (with rationale)

```
1. Title + one-line description          ← universal; always first
2. Badges (3 max)                        ← immediate trust signals
3. Feature bullet list (5-8 items)       ← what it does, specifically
4. Quick Start: CLI (2-3 shell lines)    ← show it working immediately
5. Quick Start: Library (link or brief)  ← for embedding use case
6. Installation (go get)                 ← one line
7. Documentation (pkg.go.dev link)       ← defer API docs to godoc
8. Security (reporting channel)          ← builds trust for security projects
9. License (one line)                    ← brief, at the bottom
```

This order follows the pattern from quic-go, age, and wireguard-go – the three closest analogs to tor-go.

Title

Follow the quic-go pattern: put "pure Go" in the H1.

> `# tor-go`
>
> *A Tor client implementation in pure Go.*

Or as a single line:

> `# tor-go: A Tor client implementation in pure Go`

Badges (exactly 3)

```
[![Go Reference](https://pkg.go.dev/badge/github.com/cvsouth/tor-go.svg)](https://pkg.go.dev/gi
[![Build Status](https://github.com/cvsouth/tor-go/actions/workflows/test.yml/badge.svg)](https
[![Go Report Card](https://goreportcard.com/badge/github.com/cvsouth/tor-go)](https://goreportc
```

Why these three: - Go Reference appears in 11/13 top Go projects (near-mandatory) - CI Status appears in 9/13 (signals automated testing) - Go Report Card appears in 5/13 (Go-specific quality signal)

Do NOT include: license badge (0/13 projects use one), download count, social badges, coverage (liability if low).

Features (bullet list, 5-8 items)

State what is implemented, specifically. This is where "zero dependencies" goes (as one bullet among many, not a headline). Follow the quic-go/caddy pattern of listing protocol compliance.

Example:

```
- Tor v3 client protocol (link handshake, circuit building, stream multiplexing)
- 3-hop onion-routed circuits with ntor key exchange
- v3 onion service client (.onion address resolution and connection)
- SOCKS5 proxy server for transparent traffic routing
- Directory authority consensus fetching with cryptographic validation
- Bandwidth-weighted path selection (guard/middle/exit)
- Pure Go with minimal dependencies (only `golang.org/x/crypto` and `filippo.io/edwards25519`)
```

Quick Start: CLI

Lead with the CLI tool. Follow the mkcert/age "show it working" pattern:

```
go install github.com/cvsouth/tor-go/cmd/tor-client@latest
tor-client
```

Then show verification:

```
curl --socks5-hostname 127.0.0.1:9050 https://check.torproject.org/api/ip
# {"IsTor":true,"IP":"..."}
```

This answers "what does this do?" in 3 lines before any explanation.

Quick Start: Library

The library API is too complex for a README snippet (50-70 lines minimum for a working example). Two options:

Option A (recommended for now): Link to the example binary.

> *See `cmd/tor-client` for a complete working example, or browse the API documentation.*

Option B (if a convenience API is added later): Show a minimal Go snippet:

```
conn, err := torgo.Dial("tcp", "example.com:80")
```

Option B requires implementing a high-level convenience layer (like bine's `tor.Start()`). This is the single highest-impact improvement for both the README and developer adoption.

Security Section

Every credible security project has a security reporting channel. Keep it brief:

```
## Security

To report a security vulnerability, please email [security contact] or open a
[GitHub security advisory](link).
```

Logo (optional)

10/13 top Go projects have a logo. For a library (not application), a small logo (180-300px) is appropriate. Light/dark mode support via `<picture>` is a polished touch used by age and Bubble Tea.

What the README Should NOT Contain

Based on evidence from Q1 (0/9 projects) and Q4 (anti-patterns):

Omit	Why
Disclaimers / "use at your own risk"	Erodes trust; bine, age, obfs4 use zero disclaimers
Roadmap / TODO list	Communicates incompleteness; 0/9 top projects include one
Tor protocol explanation	Anyone searching for a Go Tor library already knows what Tor is
Comparison tables	Risky for security projects; 0/9 top projects include one
Go API code examples	Universal pattern: all 9 libraries defer to godoc
Changelog	Use GitHub Releases instead
Affiliation disclaimer	bine and obfs4 don't disclaim Tor affiliation
"Security Considerations" essay	Don't re-explain Tor's threat model; link to Tor Project docs
Self-deprecating language	Destroys confidence; 0/15+ examined projects do this
More than 3 badges	Sweet spot is 3; more looks cluttered

Do / Don't Checklist

Do

- Open with what it IS in one sentence
- Put "pure Go" in the title (quic-go pattern)
- Use exactly 3 badges (Go Reference, CI, Go Report Card)
- List features as specific protocol capabilities, not adjectives
- Lead with a CLI quick start (2-3 shell lines)
- Show verification output (`{"IsTor":true}`)
- Mention "minimal dependencies" as one bullet in features
- Include a security reporting channel
- Link to pkg.go.dev for API documentation
- State limitations as scope boundaries ("Client only. Does not host onion services.")
- Keep it under 100 lines

Don't

- Add disclaimers, warnings, or "experimental" labels
- Apologize for current limitations
- Include Go code examples (defer to godoc)
- Explain what Tor is or how onion routing works
- Claim the implementation is "secure" or "anonymous"
- Add a roadmap or TODO list
- Compare with bine, Arti, or C Tor
- Use a license badge
- Include more than 3 badges
- Add an affiliation disclaimer at the top

Tone Guidance

The research distills into one formula:

```
Trust = Specificity + Brevity + Factual Tone - (Disclaimers + Vagueness + Self-Deprecation)
```

Write like wireguard-go and quic-go: state facts, link to authoritative sources, let the code speak. The README's job is to get out of the way.

Word Count Target

- **Total README:** 60-100 lines of markdown (~300-500 words of prose)
- **Description:** 1-2 sentences
- **Features:** 5-8 bullet points
- **Quick Start:** 5-8 lines (including code blocks)
- **Everything else:** links, not prose

Suggested Convenience API (Strategic Recommendation)

Q3 research revealed that tor-go's library API requires 50-70 lines for a minimal working example. Every comparable library that succeeds in README presentation (bine, age, mkcert) either has a CLI tool or a high-level convenience API.

Consider adding a thin convenience layer:

```
// Package torgo provides a high-level Tor client.
func Dial(network, address string) (net.Conn, error)
func ListenSOCKS(addr string) (net.Listener, error)
```

This would: 1. Make the README code example possible in 3-5 lines 2. Lower the adoption barrier for developers who don't need circuit-level control 3. Follow the pattern of every successful Go networking library (net.Dial, tls.Dial, etc.)

This is a library design recommendation, not a README recommendation – but the two are tightly coupled. The README exposed the usability gap.

Sources

All findings derived from primary evidence (actual README files fetched from GitHub):

- wireguard-go, age, mkcert, gost, go-ethereum, syncthing, caddy, quic-go, kcp-go (Q1)
- age, mkcert, caddy, hugo, cobra, gin, chi, bubbletea, gorilla/mux, testify, zap, bine, quic-go, go-plugin (Q2)
- bine, age, mkcert, kcp-go, wireguard-go, tor-go cmd/tor-client (Q3)
- bine, age, CIRCL, memguard, obfs4, wireguard-go, Vault, go-libp2p, Pond, yggdrasil-go, CertMagic, mihomo (Q4)