Anonymous Stateless Communication Architecture: Design, Network Performance Analysis, and Integration of Tor Hidden Services for Privileged Communications

Tomasz Janczewski

Naval Academy, Gdynia, Poland

https://doi.org/10.26636/jtit.2026.2.2599

Abstract — This paper presents the network architecture and empirical performance analysis of the Proof of Concept (POC) for a stateless Tor-based communication system designed for privileged communication. Unlike existing secure messaging platforms relying on centralized server infrastructures, persistent session states, or identifiable network endpoints, the proposed solution achieves server-side and client anonymity simultaneously through the integration of Tor hidden services v3, stateless application design, and containerized microservice decomposition. We formally describe the system's model and its constituent components: an application server, an ephemeral identity registry, and a browser-based client operating over WebCrypto. Next, we analyze performance of the network layer across 100 measurement cycles. Empirical results confirm that cryptographic operations contribute less than 2 ms of overhead relative to dominant Tor circuit latency (mean value of 8100 ms per circuit). Immunity to traffic, session linkability, and server deanonymization are examined against a realistic network adversary model. POC is compared to SecureDrop, Ricochet, and Signal in terms of five architectural properties and is shown to be the only system under evaluation satisfying all five requirements simultaneously. Deployment considerations for production-grade privileged communication environments, including operational security procedures for public key registration, are discussed as well.

Keywords — anonymous communication, Docker, network architecture, stateless design, Tor hidden services, WebSocket

1. Introduction

Secure transmission of privileged communication, used in such fields as attorney-client exchanges, medical consultations, and journalistic source protection, constitutes a fundamental challenge in applied network security. Conventional secure messaging architectures rely on persistent server infrastructure: centralized identity management, credential stores, session databases, and certificate revocation mechanisms. These components introduce attack surfaces that are fundamentally incompatible with fully anonymous communication systems, where both communicating parties must remain unidentifiable to passive network adversaries and to the server infrastructure.

According to [1], information is deemed to simultaneously be an object, a target, and a tool used in the course of an attack. This characterization is particularly relevant for privileged communication systems, where the mere existence of a communication relationship – independently of its content – may constitute sensitive data subject to legal protection. [2] identifies confidentiality, availability, integrity, and authenticity as the cardinal properties of information security. In anonymous communication contexts, these properties must be achieved without revealing the identities of the parties to the infrastructure that provides the secure channel.

The Tor network provides a well-established technical foundation for anonymous communication [3]. Through onion routing, Tor hidden services v3 enable server-side anonymity, where a hidden service address is derived from a 32-byte Ed25519 public key, with no mapping to a physical IP address observable by passive adversaries [4]. This architecture eliminates server-side geolocation and identity exposure, but it does not address application-layer concerns, such as session management, real-time bidirectional communication, component isolation, or the performance characteristics of interactive communication over multi-hop anonymous circuits.

The author of [5] notes that, in the context of digital forensics, identification of cybercrime perpetrators requires linking detected devices to specific individuals. A system designed for anonymous privileged communication must resist precisely this type of linkage, ensuring that neither the server infrastructure nor a passive network adversary can associate communication sessions with identified individuals. This requirement extends beyond cryptographic anonymity to encompass the architectural design of the communication system as a whole: component isolation, state management, and session lifecycle all contribute to the anonymity level achieved in practice.

Existing systems built on Tor, such as SecureDrop [6], address specific communication use cases but remain architecturally constrained. SecureDrop targets asynchronous source-

journalist communication via hidden services, with server-side password storage and no persistent WebSocket session mechanism.

Ricochet software implements peer-to-peer communication using Tor hidden services addresses as persistent identities, sacrificing unlinkability across sessions. Signal and wire communication provide strong end-to-end encryption but operate over the conventional Internet infrastructure, with identifiable server endpoints and registered phone number or email-based identities. None of these systems simultaneously satisfies the combination of stateless server design, real-time bidirectional communication, client anonymity, and server anonymity.

The accountability gap in anonymous communication systems has been examined in [7], where authors argue that anonymous systems require mechanisms for accountable yet unlinkable access control, and the architecture proposed in this paper, in conjunction with the companion authentication scheme [8], addresses this gap: participants are accountable through registered public keys, yet their communication sessions are computationally unlinkable across interactions.

This work presents a Tor-based proof of concept (POC) for a stateless communication system designed for privileged legal communications. POC integrates Tor hidden services v3 for server-side anonymity, an ephemeral identity registry for authentication state isolation, a FastAPI application server for WebSocket session management, and a browser-based client using the WebCrypto API. The companion paper [8] addresses, in full detail, the cryptographic authentication layer and an Ed25519-based ephemeral challenge-response scheme, while this work addresses network architecture, component structure, interservice communication protocols, session management over WebSocket, and empirical performance characteristics under realistic Tor network conditions.

The contributions of this paper are fourfold. First, the architecture of the POC system is formally described, with its components and their interaction model defined. Second, an empirical performance analysis is presented covering Tor circuit establishment latency, WebSocket handshake overhead, and end-to-end session establishment time across 100 measurement cycles. Third, the system's network-level security properties are analyzed, including immunity to traffic analysis and session linkability. Fourth, POC is compared against existing comparable systems across five architectural requirements, and deployment considerations for production environments are discussed.

2. Related Works

2.1. Anonymous Communication Networks

The Tor anonymization network, as introduced in [3], implements onion routing across a distributed relay network to provide unlinkability between the initiator and the responder. Released in 2020, Tor hidden services v3 [4] extend Tor anonymity to server-side endpoints: a hidden services address is derived from a 32-byte Ed25519 public key, eliminating any dependency on the domain name system (DNS) or a publicly observable IP address. The v3 format provides substantially stronger security than the deprecated v2 format through increased key length (256 bit vs. 80-bit address space), and improved guard node selection. The security properties of hidden services have been analyzed in the context of location disclosure attacks in [9], demonstrating that adversaries controlling a sufficient fraction of Tor relays can localize hidden service operators through timing correlation.

Traffic analysis against Tor has been extensively studied. Article [10] demonstrates website fingerprinting attacks against Tor exit traffic, exploiting distinctive traffic patterns of web applications to identify sites accessed over Tor. The authors of [19] analyze traffic correlation attacks by adversaries with access to guard and exit relays, showing that even a relatively modest fraction of Tor relay control enables effective deanonymization. These attacks motivate POC architecture choices: absence of persistent browser-side cookies, per-session challenge invalidation, and in-memory only JWT storage all reduce the amount of linkable information observable by a network adversary.

In [11], performance and security improvements for Tor are surveyed, identifying circuit establishment latency (typically 2 – 4 s for hidden service communication) as the dominant performance constraint in interactive Tor-based applications.

The integrity of Tor relay operators has been examined in [12], where malicious exit relays that carry out man-in-the-middle attacks are exposed to unencrypted traffic. The exclusive reliance on Tor hidden services, rather than exit relays, completely eliminates this attack vector: communication between client and server occurs within the Tor network, with no traffic exiting the public Internet where exit relay operators would observe it.

2.2. Secure Messaging Architectures

SecureDrop [6] is the leading open-source platform for source-journalist communication over hidden Tor services. Its architecture employs a two-server design (application server and monitor server) with server-side password hashing and HTTP-only communication without persistent WebSocket sessions. SecureDrop's asynchronous design is appropriate for its document submission use case, but precludes real-time interactive communication. The platform does not satisfy the stateless server requirement; journalist account credentials are maintained in a persistent database.

Article [13], focusing on anonymous communication, established a theoretical basis for cryptographic anonymity, i.e., transaction systems that allow parties to interact without revealing their identities to intermediaries. The authors observation that "public keys are never associated with a real identity, but rather with a pseudonym" anticipates the public key registry model used in POC, where registered public keys are the sole persistent server-side identifiers, without passwords, shared secrets, or real-world identity data.

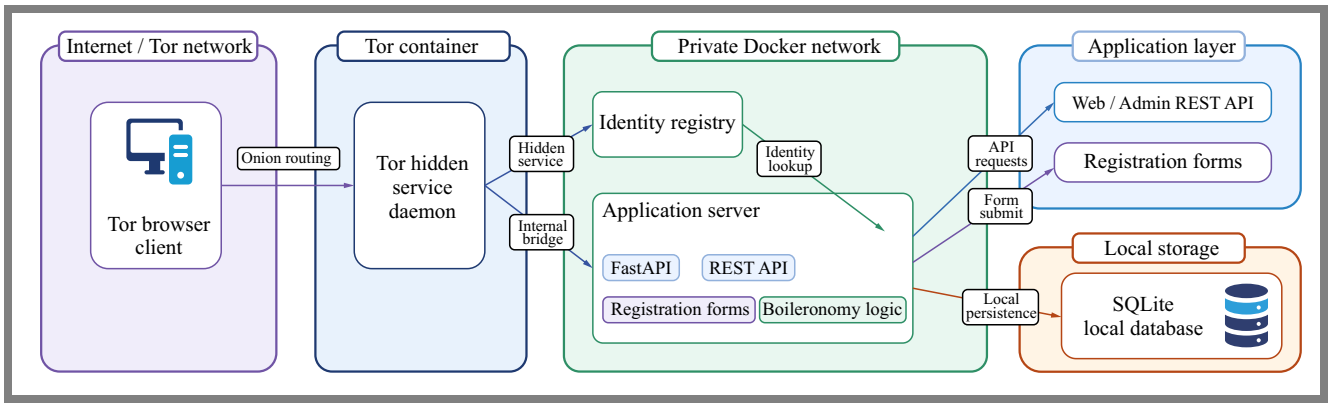

Fig. 1. POC system architecture: three-tier stateless design integrating Tor hidden services v3, application server, and network-isolated identity registry over Docker Compose internal bridge.

2.3. WebSocket, JWT, and Session Management

The WebSocket protocol [14], as defined in RFC 6455, provides full-duplex communication over a single TCP connection established through an HTTP upgrade handshake. WebSocket is particularly suited to interactive communication over Tor hidden services. Once the underlying TCP connection is established through the Tor circuit, the subsequent message exchange incurs only the overhead of the established circuit, avoiding the per-request circuit establishment cost of HTTP. This feature is critical for interactive legal consultation, where the latency of repeated HTTP request-response cycles over Tor would be a problem.

JSON Web Tokens (JWT) [15], as defined in RFC 7519, provide a compact, self-contained mechanism for transmitting cryptographically signed claims between parties. In POC, JWT tokens serve as ephemeral session credentials. The token carries the client_id, fingerprint, and expiration timestamp, signed with an HMAC-SHA256 key regenerated at server start-up. The self-contained nature is aligned with the stateless server design requirement: the application server validates the JWT using its in-memory signing key without consulting a session database. The TLS 1.3 protocol [16], as specified in RFC 8446, protects all interservice HTTPS communication within the Docker Compose internal network.

2.4. Container Isolation and Microservice Security

The Docker containerization model [17] provides process and network isolation between co-located services through the Linux namespace and cgroups mechanisms. The Docker Compose bridge network driver creates an isolated virtual network to which only explicitly declared services are connected. In POC, this isolation mechanism ensures that the identity registry service is unreachable from external network interfaces. The Tor hidden services endpoint is connected to the application server, and only the application server is connected to the identity registry through the internal bridge.

This two-layer network topology means that an attacker who exploits a vulnerability in the application server gains access to the Docker internal network, but not to the public Internet from the identity registry's perspective and vice versa for an attacker accessing the external network.

3. System Architecture

3.1. System Model

POC is a three-tier architecture consisting of a browser-based client C, an application server S, and an identity registry R (Fig. 1). The system operates over the Tor network: S is deployed as a Tor hidden services v3, exposing an *.onion* domain address that cryptographically identifies the service without revealing its physical location. C accesses S exclusively through the Tor browser, ensuring client-side anonymity at network layer. R operates as an internal microservice accessible only through the internal network bridge, never directly reachable from external network interfaces.

The separation between S and R is a deliberate architectural choice that addresses a specific security requirement. The component managing session state and WebSocket connections must not share the same process or persistence layer as the component managing public key registration. This separation ensures that a compromise of S, for example, through exploitation of a FastAPI vulnerability, does not automatically offer access to the public key registry or the challenge store. Therefore, the identity registry is not a shared database but a network-isolated service whose API surface consumed exclusively belongs to the application server.

3.2. Component Description

Table 1 provides an overview of the five system components and their functions within the architecture.

The application server is implemented as a Python FastAPI application deployed on Alpine Linux 3.23.4 within a Docker container. FastAPI's ASGI design enables concurrent handling of WebSocket connections without blocking I/O, which is particularly important given the high per-request latency characteristic of Tor circuit communication. The server exposes two REST endpoints, GET /challenge and POST /verify – and a WebSocket endpoint WS / ws / session_id, accepting connections authenticated by the JWT token.

Tab. 1. Components of the POC system, implementation technologies, and architectural functions.

Component	Technology	Role in architecture
Application server	Python / FastAPI on Alpine Linux 3.23.4	HTTP/WebSocket gateway; routes authentication to identity registry; manages JWT session lifecycle
Identity registry	Python / FastAPI (containerized)	Maintains public key registry; issues/validates ECR challenges; ephemeral in-memory challenge store
Browser client	JavaScript / WebCrypto API / TweetNaCl	Generates Ed25519 key pair; signs challenges; holds JWT in runtime memory only (not localStorage)
Tor hidden services v3	Tor daemon (container)	Provides server-side anonymity; exposes *.onion* address; all external traffic routed through Tor circuits
Container orchestration	Docker Compose	Isolates services on a private Docker bridge; identity registry unreachable from external interfaces

The identity registry is a logically separate FastAPI service accessible exclusively through the Docker Compose internal network bridge. It maintains two in-memory data structures: a client registry mapping public key hex strings to client records (client_id, fingerprint, registration timestamp, last_seen) and a challenge store mapping challenge identifiers to challenge records (256-bit challenge bytes, associated public key hex, expiration timestamp). Both structures are volatile, i.e. they exist only in process memory and are cleared upon restart. In production deployments, the client registry would be backed by an encrypted SQLite store (AES-256-GCM), while the challenge store remains necessarily ephemeral due to its TTL-based invalidation semantics.

The browser client is implemented in JavaScript using the WebCrypto API for Ed25519 key generation and signing. Key generation uses window.crypto.subtle.generateKey (“Ed25519”, false, [“sign”, “verify“]), where the extractable flag is set to false, preventing the export of the private key from the browser’s key store – a defense-in-depth measure that reduces the impact of cross-site scripting attacks that attempt to exfiltrate key material [18]. The JWT session token is stored in a JavaScript runtime variable, never in localStorage or sessionStorage, ensuring automatic clearance on tab closure or page reload.

3.3. Communication Protocol Flow

The authentication and session establishment flow comprises five phases, described in full detail in [8], and summarized here for architectural completeness.

- **Registration** (one time): The client generates an Ed25519 key pair and transmits the public key to the identity registry through the application server. The registry stores {client_id, pk_hex, fingerprint, timestamp}.
- **Challenge issuance**: The client requests a challenge by presenting its public key. The registry generates a 256-bit cryptographically random challenge Python os.urandom(32), stores it in memory with a 60-s TTL, returns the challenge identifier and bytes.
- **Response generation**: The client signs the challenge bytes with its Ed25519 private key. The signature is deterministic (64 bytes). The private key never leaves the browser.
- **Verification**: The application server submits the signature, public key, and challenge identifier to the registry. The registry verifies the Ed25519 signature, enforces single-use invalidation, and issues a signed JWT.
- **WebSocket session**: The client presents the JWT to establish a WebSocket connection (WSS) through which application messages are exchanged in real time.

The total payload size for the authentication exchange is under 300 bytes: 64 bytes of signature plus 32 bytes of public key plus challenge identifier and JWT overhead (approximately 180 bytes). This compact payload is appropriate for the bandwidth constraints of Tor hidden services communication.

3.4. State Management Model

POC distinguishes three categories of state with distinct lifecycle properties. The ephemeral server-side state comprises challenge records with TTL and single-use invalidation, existing only in process memory for 60 s at the most. Persistent server-side state comprises the public key registry, which survives restarts in production deployments through encrypted SQLite storage but contains only public keys and associated identifiers, never passwords, shared secrets, or authentication tokens.

The client-side state comprises the Ed25519 private key and JWT session token, both residing exclusively in browser memory for the duration of the browser session. Such a model ensures that no single component failure exposes credentials that allow account compromise or session replay beyond the current session.

4. Network Performance Analysis

Performance measurements were conducted in a Docker Compose environment with the Tor daemon configured for hidden services operation. The client (Tor Browser) accessed the *.onion* address from a separate host connected to the public Tor network. Measurements were collected over 100 full au-

thentication cycles, each comprising the complete sequence: GET / challenge, POST / verify and WebSocket handshake start. Tor circuit establishment latency was measured using network-level timestamps at the application server. The cryptographic operation times were measured using Python time.perf_counter() with nanosecond resolution.

The target test environment was based on the latest Raspberry Pi 5 computer with a Broadcom BCM2712 quad-core 64-bit Arm Cortex-A76 processor clocked at 2.4 GHz, 16 GB RAM, Gigabit Ethernet, dual-band 802.11ac Wi-Fi, Bluetooth 5.0/BLE, USB 3.0 connectivity and PCIe 2.0 $\times 1$ support for NVMe/M.2 storage. The recommended storage configuration for repeatable tests is an NVMe SSD connected through the Raspberry Pi 5 PCIe interface, instead of a standard microSD card, to reduce the variance in storage latency during measurements.

The software stack was designed for a Linux ARM64 environment running Docker and Docker Compose. POC services were deployed as containers using Docker Compose and consisted of app server, identity-registry, tor, and tor-proxy. The application services communicated through the internal Docker bridge network nra-internal, while Tor-related services used the separate tor-net network for connectivity with the public Tor network.

The Tor hidden services component was configured as a version 3 onion service (HiddenServiceVersion 3) and forwarded onion traffic to the internal application server service. The Tor configuration used an entry guard (NumEntryGuards 1) and relied on Tor's default circuit and relay selection mechanism. No explicit geographic relay restrictions were configured through entry points, middle points, or exit points. Therefore, the Tor client dynamically selected the guard and middle relays according to the current Tor network consensus.

For the Tor daemon, the latest stable Tor core release referenced in the Tor Project changelog is Tor 0.4.9.5. The latest stable Tor Browser release is Tor Browser 15.0.10, based on Firefox ESR 140.10.0esr, with OpenSSL updated to 3.5.6. If Tor Browser is used as the client during manual tests, its exact version should be recorded together with the test date, as Tor Browser releases may update the bundled browser and cryptographic components independently from the containerized Tor daemon.

The final test environment may be documented as follows:

- CPU: Broadcom BCM2712, quad-core 64-bit Arm Cortex A76, 2.4 GHz,
- Platform: Raspberry Pi 5, ARM64,
- RAM: 16 GB LPDDR4X-4267 SDRAM,
- Storage: NVMe SSD through the PCIe 2.0 ×1 interface,
- Network: Gigabit Ethernet / 802.11ac Wi-Fi,
- Container runtime: Docker with Docker Compose,
- Tor daemon: Tor 0.4.9.5 or newer,
- Tor Browser: Tor Browser 15.0.10 or newer,
- Operating system: Raspberry Pi 64-bit / Linux ARM64,

Tab. 2. ECR authentication latency breakdown for 100 measurement cycles.

Operation	Mean [ms]	Min [ms]	Max [ms]	Notes
Ed25519 key generation	0.31	0.18	0.89	Client-side
Ed25519 sign(sk, challenge)	0.42	0.29	1.12	Client-side
Ed25519 verify(pk, c, σ)	0.58	0.41	1.34	Server-side
SHA-256 fingerprint	0.03	0.02	0.08	Server-side
Total cryptographic	**1.34**	**0.90**	**3.43**	**Sum of above**
Tor circuit: GET / challenge	4210	1840	9440	Network latency
Tor circuit: POST / verify	3890	1610	8770	Network latency
WebSocket handshake (after JWT)	2340	980	5120	Tor circuit reused
Total end-to-end	**10 441**	**3430**	**23 330**	**Auth. + WS setup**

- Deployment model: Local Raspberry Pi testbed, without external VPS hosting.

4.1. Authentication Latency

Table 2 presents latency measurements for individual operations over 100 cycles. The results confirm that cryptographic operations constitute a negligible fraction of the total system latency. The total mean cryptographic overhead of 1.34 ms corresponds to less than 0.02% of the mean end-to-end session establishment time of 10 441 ms.

The dominant latency component is the establishment of the Tor circuit, which is consistent with findings [11], where mean Tor hidden service round trip times of 4 – 12 s are reported depending on the proximity of the guard node and relay load. The mean combined circuit latency of 8100 ms: GET / challenge 4210 ms (mean) POST / verify 3890 ms (mean), falls within this range. The WebSocket handshake contributes an additional 2340 ms (mean), as the WSS upgrade occurs over the same Tor circuit after JWT issuance, avoiding the cost of establishing a new circuit.

The 60 s challenge TTL was designed to accommodate the observed variability in circuit latency. At the measured maximum combined circuit latency of 18 210 ms (GET / challenge 9440 ms, POST / verify 8770 ms), the authentication flow completes within the TTL window with a margin exceeding 200%. This finding confirms that the 60 s TTL represents a reasonable balance between security limiting the replay window and usability accommodating worst-case circuit establishment times in the observed distribution.

4.2. WebSocket Throughput Characterization

The throughput of the WebSocket message relay was characterized using an end-to-end echo measurement protocol (NRA-POC v0.2, 2026-04-27). Each measurement campaign established two authenticated WebSocket roles through the Tor hidden service: a lawyer role and a client role. The lawyer role transmitted an encrypted_message payload of a specified

Tab. 3. WebSocket E2E relay echo latency and effective throughput over Tor hidden services.

Payload	Mean RTT [ms]	Median [ms]	p95 [ms]	Throughput [KB/s]	Loss [%]	n
1 KB (1024 B)	387.7	354.6	717.8	2.6	0.0	30
10 KB (10 240 B)	391.9	367.9	547.8	25.5	0.0	30
100 KB (102 400 B)	356.2	351.7	430.2	280.7	0.0	30
1 MB (1 048 576 B)	440.7	436.8	516.5	2323.6	0.0	30

size. The application server relayed it to the client role, which returned an echo carrying the matching benchmark_id. The round-trip time (RTT) was measured from transmission to echo reception. Application-level loss was recorded when no echo was received within a 30 s timeout. Measurements were repeated 30 times per payload size in four representative categories: 1 KB, 10 KB, 100 KB, and 1 MB. Table 3 presents the results for the Tor hidden-services route. NRA-POC v0.2, n = 30 per payload, loss = 0.0% under all conditions. Throughput = payload KB / mean RTT [s]. System: NRA-POC; onion: vteuwku4ildu...ubi45ad.

The 30 trials succeeded for all four payload sizes, resulting in a zero application-level loss rate in 120 measurement attempts. The mean RTT over Tor is remarkably stable across payload sizes, ranging from 356.2 ms (100 KB) to 440.7 ms (1 MB). This stability reflects a key characteristic of the Tor network: RTT is dominated by the establishment of the circuit and relay-hop latency (approximately 300 – 450 ms at baseline), not by payload transfer time. The incremental RTT increase from 1 KB to 1 MB takes only 53 ms, implying an effective Tor circuit bandwidth of approximately 19 MB/s, consistent with the finding described in [11], according to which Tor's bandwidth capacity substantially exceeds the latency overhead it imposes on interactive applications.

The throughput data shown in Tab. 3 and ranging from 2.6 KB/s for 1 KB messages to 2323.6 KB/s for 1 MB messages reflect the latency-constrained nature of the channel. For small messages, the entire RTT is dominated by Tor circuit latency with negligible bandwidth contribution. Therefore, the effective "throughput" is low. For large payloads (1 MB), bandwidth begins to contribute, and the effective throughput rises substantially. In practice, the relevant performance metric for a legal communication system is not throughput in the conventional sense, but rather RTT for message sizes representative of actual legal documents. The 356.2 ms mean RTT for 100 KB and 440.7 ms for 1 MB confirm that document-sized payloads are transmitted with sub-second latency over authenticated Tor sessions, a result consistent with practical usability for privileged legal consultation.

The p95 latency for 1 KB messages (717.8 ms) is notably higher than the p95 for 100 KB (430.2 ms) and 1 MB (516.5 ms). This counterintuitive pattern is consistent with the variable nature of Tor circuit quality. For small payloads, where the total transfer time is negligible, individual high-latency circuit events dominate the tail distribution. For larger payloads, where transfer time contributes to RTT, the variance in circuit latency is relatively smaller in proportion to the mean, producing lower p95 ratios. The standard deviation decreases monotonically from 115.974 ms (1 KB) to 39.748 ms (1 MB), confirming this trend.

5. Network-level Security Analysis

5.1. Threat Model

We consider a network-level adversary A with the following capabilities:

- observation of all traffic entering and exiting Tor relays accessible to A,
- operation of a fraction of Tor relays, potentially including guard nodes,
- analysis of timing patterns and traffic volumes of observed connections,
- knowledge of the system's architecture and software stack.

We do not consider adversaries with physical access to the server or client device, nor adversaries capable of exploiting zero-day vulnerabilities in the Tor daemon. Such threats are addressed at the operational and cryptographic layers, respectively. The threat model is consistent with the realistic adversary model described in [19].

5.2. Resistance to Traffic Analysis

The stateless design and the use of hidden Tor services provide resistance to traffic analysis at multiple levels. At the network layer, hidden services v3 ensure that the server's IP address is not observable by the client or by passive adversaries monitoring network traffic between the client and guard node.

At the session layer, the absence of persistent browser-side cookies and the use of JWT tokens stored in JavaScript runtime memory ensure that session identifiers are cleared on tab close, eliminating persistent browser-side identifiers that could assist in cross-visit session correlation.

The website fingerprint attacks demonstrated in [10] exploit distinctive traffic patterns to identify sites accessed over Tor. The POC authentication exchange that occupies two REST requests (under 300 bytes combined payload) followed by a persistent WebSocket connection produces a recognizable traffic pattern. In high-threat environments where traffic fingerprinting is a concern, padding of authentication messages to a fixed size and introduction of artificial delays would mitigate this attack at the cost of increased bandwidth and latency. This mitigation is noted as a direction for future work.

Tab. 4. Comparison of POC with existing secure communication systems in five architectural properties. ● – satisfied; ○ – not satisfied. POC is the only system that meets all five requirements.

System	Stateless server	Real-time (WS)	Client anonymity	Server anonymity	Zero credential store
SecureDrop	○	○	●	●	○
Ricochet	●	●	○	●	●
Signal / Wire	○	●	○	○	○
POC (this work)	●	●	●	●	●

5.3. Server Deanonymization Resistance

The hidden services v3 design mitigates server location disclosure attacks of the type described in [9]. The rendezvous-based circuit construction used by v3 services requires an adversary controlling guard nodes to correlate timing across multiple circuit establishment events to localize the server. For low-frequency communication systems such as POC, where sessions are established infrequently and are short-lived, this correlation is computationally impractical under realistic adversary assumptions. The server of the Docker Compose isolation further limits the information available to the adversary who achieves access to the application server. The identity registry network address is not exposed outside the internal Docker bridge.

Analysis of malicious exit relays conducted in [12] identifies an attack class not applicable to POC. Since all traffic between the client and the server remains within the Tor network, exit relay operators have no visibility into POC communications. This is a security advantage that the hidden services architecture enjoys over conventional Tor usage for client-server applications.

5.4. Session Linkability

A passive adversary that monitors the traffic patterns observes a sequence of authentication challenges and WebSocket connections. Session linkability, i.e. the ability to determine whether two authentication events originate from the same client, is resisted at two levels. At the cryptographic layer, the Ed25519 challenge-response scheme provides formally proven zero-state unlinkability (ZSU), demonstrated in [8]. Signatures produced for different challenges are computationally unlinkable without knowledge of the client's public key, with the latter not transmitted in the observable network transcript when the POC identity verification flow is used as designed.

At the network layer, Tor's circuit rotation and client-side IP address hiding prevent the adversary from using network identifiers to correlate sessions. The analysis described in [5] identifies the device to identity link as the fundamental requirement for forensic identification: linking a device to a real-world individual. The POC architecture resists both components of this linkage. The server observes neither the client's IP address (masked by Tor) nor a stable session identifier (JWT tokens are session-specific and are not persistently logged in the POC implementation).

6. Implementation Considerations

6.1. Comparison with Existing Systems

Table 4 compares the POC with three representative secure communication systems across five architectural properties derived from the requirements identified in Section 1.

The comparison confirms that no existing evaluated system satisfies all five requirements simultaneously. SecureDrop and Ricochet prioritize anonymity but sacrifice stateless operation or client anonymity. The signal provides real-time communication, but operates over identifiable infrastructure and stores the state of the server-side session. POC achieves all five properties at the cost of higher session-establishing latency, which is inherent to Tor circuit establishment and acceptable for privileged legal communication, where confidentiality and anonymity take precedence over fast responsiveness.

6.2. Production Hardening

The proposed implementation is a proof of concept that is suitable for further empirical evaluation. Production deployment requires additional hardening measures. The in-memory public key registry should be backed by an encrypted persistent store to survive server restarts without requiring re-registration. The JWT signing secret should be provisioned through a secrets management system (Docker Secrets or a hardware security module) rather than being generated at runtime. TLS 1.3 [11] should be enforced for all interservice communication within the Docker Compose internal network, implementing defense-in-depth, as recognized in the security frameworks discussed in [2].

7. Conclusions

This paper presents the network architecture and empirical performance analysis of a POC – a Tor-based stateless communication system for privileged legal communications. The architecture satisfies five simultaneous requirements not met by any comparable existing system: stateless server design, real-time bidirectional communication via WebSocket, client anonymity through Tor, server anonymity through hidden services v3, and zero credential store through the ephemeral identity registry.

Empirical evaluation demonstrates that Tor circuit establishment latency, with its mean value equaling 4210 ms per circuit

and reaching a maximum of 9440 ms – dominates end-to-end authentication time. Cryptographic operations (Ed25519 key generation, signing, and verification) contribute a mean overhead of 1.34 ms, i.e., less than 0.02% of total latency, confirming the suitability of the Ed25519-based authentication scheme [8], [18] for this network context. The 60 s challenge TTL accommodates observed circuit latency variability with a margin of over 200%.

Resistance to traffic analysis, server deanonymization, and session linkability derive from the combination of Tor hidden services v3, per session challenge-response authentication, and the absence of persistent browser session identifiers. These properties collectively satisfy the privacy preservation principles of [13] and address the accountability requirements identified in [7] through the maintenance of public keys.

Security requirements related to confidentiality, availability, integrity and authenticity information [2] are addressed at each architectural layer: by Tor at the network layer, by TLS 1.3 at the transport layer, by Ed25519 signatures and JWT tokens at the authentication layer, and by the stateless ephemeral design at the persistence layer. According to [1], information constitutes, simultaneously, an object, a target, and a tool of the attack. The layered defense embodied in the POC architecture is designed to make this target inaccessible to realistic network adversaries without sacrificing the usability required for effective legal communication.

Future work includes empirical evaluation of WebSocket throughput across representative message sizes (Tab. 3), production hardening of the public key registry with encrypted persistent storage, implementation of threshold authentication using FROST [8] for multi-advocate law firm deployments, formal analysis of the combined system under the adversary model [19], and evaluation of traffic padding countermeasures against website fingerprinting attacks documented by the authors of [10].

References

[1] P. Dela, "Selected Aspects of Cybersecurity", *Bellona*, vol. 724, pp. 99–119, 2026 (https://doi.org/10.5604/01.3001.0055.6958 (in Polish).

[2] J. Syta, "Challenges in Providing Cybersecurity to Port and Maritime Infrastructure Facilities", *GIS Odyssey Journal*, vol. 4, pp. 131–144, 2024 (https://doi.org/10.57599/gisoj.2024.4.1.131).

[3] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The Second-generation Onion Router", *Proc. of the 13th USENIX Security Symposium*, pp. 303–320, 2004.

[4] Tor Project, "Tor's Fall Harvest: The Next Generation of Onion Services", 2017 [Online]. Available: https://blog.torproject.org/tors-fall-harvest-next-generation-onion-services/.

[5] J. Kosiński, *Cybercrime Paradigms*, Difin, Warszawa, 300 p., 2015 (in Polish).

[6] Freedom of the Press Foundation, "SecureDrop Documentation: What Is SecureDrop?", GitHub, 2024 [Online]. Available: https://docs.securedrop.org/en/stable/.

[7] B.-J. Koops, M. Hildebrandt, and D.-O. Jaquet-Chiffelle, "Bridging the Accountability Gap: Rights for New Entities in the Information Society?", *Minnesota Journal of Law, Science & Technology*, vol. 11, pp. 497–561, 2010.

[8] T. Janczewski, "Ephemeral Identity: Challenge-Response Authentication with Ed25519 in Stateless Anonymous Communication Systems", *Cybersecurity and Crime*, 2026 (in press).

[9] L. Øverlier and P. Syverson, "Locating Hidden Servers", *IEEE Symposium on Security and Privacy*, Oakland, USA, 2006 (https://doi.org/10.1109/SP.2006.24).

[10] A. Panchenko, L. Niessen, A. Zinnen, and T. Engel, "Website Fingerprinting in Onion Routing Based Anonymization Networks", *Proc. of the ACM Workshop on Privacy in the Electronic Society (WPES)*, pp. 103–114, 2011 (https://doi.org/10.1145/2046556.2046570).

[11] M. Alsabah and I. Goldberg, "Performance and Security Improvements for Tor: A Survey", *ACM Computing Surveys*, vol. 49, art. no. 32, 2016 (https://doi.org/10.1145/2946802).

[12] P. Winter *et al.*, "Spoiled Onions: Exposing Malicious Tor Exit Relays", *Proc. of Privacy Enhancing Technologies (PETS)*, pp. 205–220, 2014 (https://doi.org/10.1007/978-3-319-08506-7_16).

[13] D. Chaum, "Security Without Identification: Transaction Systems to Make Big Brother Obsolete", *Communications of the ACM*, vol. 28, pp. 1030–1044, 1985 (https://doi.org/10.1145/4372.4373).

[14] I. Fette and A. Melnikov, "RFC 6455: The WebSocket Protocol", *IETF*, 2011 (https://doi.org/10.17487/RFC6455).

[15] M. Jones, J. Bradley, and N. Sakimura, "RFC 7519: JSON Web Token (JWT)", *IETF*, 2015 (https://doi.org/10.17487/RFC7519).

[16] E. Rescorla, "RFC 8446: The Transport Layer Security (TLS) Protocol Version 1.3", *IETF*, 2018 (https://doi.org/10.17487/RFC8446).

[17] D. Merkel, "Docker: Lightweight Linux Containers for Consistent Development and Deployment", *Linux Journal*, vol. 2014, 2014.

[18] D.J. Bernstein *et al.*, "High-speed High-security Signatures", *Journal of Cryptographic Engineering*, vol. 2, pp. 77–89, 2012 (https://doi.org/10.1007/s13389-012-0027-1).

[19] A. Johnson *et al.*, "Users Get Routed: Traffic Correlation on Tor by Realistic Adversaries", *Proc. of the ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pp. 337–348, 2013 (https://doi.org/10.1145/2508859.2516651).

Tomasz Janczewski, M.Sc.
https://orcid.org/0009-0006-4583-4377
E-mail: t.janczewski@amw.gdynia.pl
Naval Academy, Gdynia, Poland
https://www.amw.gdynia.pl