Classification of Tor Metadata Combinations for Investigation*

Yiseul Choi, Junga Kim, and Seongmin Kim†

Sungshin Women's University, Seoul, Korea
{220256035, 220256036, smkim}@sungshin.ac.kr

Abstract

Tor is a widely deployed anonymity network that relies on layered encryption and multi-hop circuits to conceal user identity and communication patterns. While extensive research has focused on traffic analysis and deanonymization, little work has been devoted to a systematic examination of the metadata inherently generated during Tor's internal operations. This study conducts a source code, driven analysis of Tor's circuit creation, stream multiplexing, and cell transmission processes, identifying critical metadata such as circuit purpose, stream identifiers, and cell exchange patterns. Based on this analysis, we evaluated the potential for utilizing metadata from an investigative perspective. Unlike conventional deanonymization techniques, we demonstrate how metadata can assist investigations by identifying the scope of information that can be discerned from it based on its visibility. Conclusively, we examine changes in metadata based on visibility scope and discuss its potential for use as evidence. This demonstrates the feasibility of metadata-based classification in real-world investigative contexts, laying the foundation for future integration of this methodology into monitoring and forensic systems.

Keywords— Tor, Anonymity, Hidden service, Metadata, Circuit

1 Introduction

Tor(The Onion Router) is a representative overlay network designed to ensure the anonymity of internet users, performing multi-layer encryption via thousands of voluntary relays worldwide[18, 21]. Unlike typical proxy-based networks, it constructs multi-hop circuits through client-selected Guard, Middle, and Exit relays[5]. During this process, while the client knows the overall path, individual relays only know information about themselves and their neighboring nodes. This allows the user's actual origin and destination to be concealed during transit[23]. This structure helps hide user traffic and evade surveillance.

One of Tor's key features, hidden service (HS), provides an mutual anonymity protecting both sender and receiver[19]. While this allows users' freedom of expression, such as political speech and avoiding censorship[2, 17], it is also widely used for dark web based crimes[1, 3]. Criminal methods are becoming increasingly sophisticated, including the distribution of child sexual abuse material, illegal markets, drug trafficking, and ransomware planning and deployment[16, 6]. Tor is often central to cybercrime, raising investigative demand for tracking and evidential techniques.

Ensuring anonymity is fundamental to Tor's network design and the community's ethical goals. Consequently, methods that expose or actively utilize internal metadata could weaken user privacy[15]. Therefore, an inevitable trade-off exists between protecting anonymity and securing criminal evidence[14]. However, prior research has demonstrated that Tor is vulnerable to certain deanonymization techniques, such as traffic correlation attacks and AS-level adversary analysis [1, 8, 7]. These studies highlight that deanonymization, while feasible under specific conditions, often comes with significant ethical concerns

*Proceedings of the 9th International Conference on Mobile Internet Security (MobiSec'25), Article No. 85, December 16-18, 2025, Sapporo , Japan.

†Corresponding author

and potential collateral privacy risks.[8], while German authorities have been reported to use timing correlation to uncover darknet operators, sparking debate about collateral privacy risks.

To mitigate such threats, Tor employs mechanisms such as bridge nodes, which resist censorship, and circuit multiplexing, which reduces fingerprintability of individual streams[21]. These design choices reflect Tor's design principle: standard components neither log nor expose user-identifiable metadata, leaving law-enforcement access both technically constrained and conceptually incompatible.

Against this backdrop, we progressively examine the trade-offs regarding the extent of digital evidence investigators can obtain depending on visibility across Tor components. At higher visibility levels, investigators may access richer metadata but at the cost of a wider privacy impact; at lower levels, anonymity is preserved but evidential value diminishes.

This study, premised on this dilemma, aims to explore the practical feasibility of utilizing metadata observable within realistic investigative scopes. Specifically, we (1) organize key metadata generated during circuit construction and communication by tracing Tor source code (circuitbuild.c, relay.c, command.c, etc.), and (2) map the extracted metadata to investigative scenarios such as tracking hidden services or tracing malicious exit traffic[9]. In this way, the study seeks to enhance evidential potential for investigations without fundamentally undermining Tor's design philosophy.

2 Background and Related Work

Tor is a representative overlay network designed to provide anonymity and privacy for Internet users. To achieve this, Tor routes user traffic through multiple relay nodes, where each node can only decrypt the encryption layer specifically assigned to it. This layered encryption mechanism is commonly compared to peeling an onion and is thus referred to as onion routing [4]. The Tor network consists of several key components: the Onion Proxy (OP), Tor relays, Directory Authorities (DAs), and hidden service–related nodes (e.g., introduction point and rendezvous point). A client who operates OP is responsible for constructing circuits and initiating individual TCP streams over them. To this end, the client periodically downloads the consensus document issued by the Directory Authorities, which contains the list of available relays and network state. Based on this information, the client establishes a circuit composed of guard, middle, and exit nodes.

2.1 Circuit Architecture

Communications on Tor are carried out through circuits. A circuit is constructed by sequentially connecting a series of relay nodes selected by the client. The fundamental architecture adopts a minimum three-hop structure, consisting of guard, middle, and exit relays. Each circuit is identified within a channel by its circuit ID, and multiple streams can be multiplexed over a single circuit.

Classification of Circuit Types. Tor circuits can be categorized into *General circuits* and *Hidden Service circuits* based on their intended purpose. It is also important to note that "Directory-related circuits" do not constitute an official Tor circuit type. The process of directory fetching (for consensus and descriptors) is ordinarily executed over short-lived one-hop circuits by means of the `RELAY_DIR` command within general-purpose circuits.

A General circuit is the standard form, in which the client establishes a route through guard–middle –exit nodes to connect to an external server. This enables the transmission of ordinary traffic, such as web browsing or TCP sessions. A Hidden Service circuit is a specialized type that supports anonymous communication between an Onion Service and its clients. An Onion Service designates specific relays as Introduction Points and registers this information within the Tor network's hidden service directories, which clients access using the service's .onion address(.onion). Clients then select a rendezvous point (RP) and convey this choice to the onion service via the introduction point (IP), using a non-identifiable communication method. As a result, the control flow of Hidden Service circuits is more complex than that of standard circuits.

A directory circuit differs in both purpose and configuration. Unlike data circuits, it consists of a single hop. Clients periodically connect to a Directory Authority or a Directory Cache Relay to

download the most recent network state (consensus document). This short-term one-hop circuit is used by the Onion Proxy (OP) when fetching consensus data. Table 1 summarizes the properties of these circuit types.

Circuit Types	Primary Purpose	The cell being used	Related fields /structures
General Circuit	Client ↔ External Internet (Web browsing, TCP sessions)	`CREATE/CREATED`, `EXTEND/EXTENDED`, `RELAY_BEGIN/`, `RELAY_CONNECTED`	`chosen_exit`, `purpose =GENERAL`
Hidden Service Circuit	Onion Service ↔ Client connection	`ESTABLISH_INTRO`, `INTRO_ESTABLISHED`, `INTRODUCE1/2`, `RENDEZVOUS1/2`	`origin_circuit_t`, `crypt_path_t`, `purpose=HS_CLI ENT/SERVICE`
Directory-related Circuit	Network Consensus /Descriptor Exchange	`CREATE/CREATED`, `RELAY_DIR`	`has_opened`, `purpose =GENERAL` (short-lived, one-hop)

Table 1: Tor Circuit Types and Features

2.2 Cell Overview

All communications in Tor are transmitted in fixed-size units called cells. Circuit expansion, data stream generation, and relay-to-relay transmission are all performed on the basis of this cell structure. Each cell is 512 bytes in size and consists of a header and a payload, with each relay interpreting only the portion relevant to it. This design conceals traffic patterns by ensuring that every cell maintains a uniform size, preventing from inferring how much meaningful data each cell actually carries.

A key feature of the cell structure is its integration with layered encryption. During circuit construction, a client (OP) establishes a session key with each relay included in the circuit. The client then encrypts the payload in multiple layers, one for each relay. As the cell traverses the circuit, each relay removes only its corresponding encryption layer. Intermediate relays can access only the portion of data for which they hold a key, while the actual TCP payload is revealed solely at the final exit relay. The combination of fixed cell size and onion encryption ensures strong anonymity guarantees.

Beyond their role as traffic delivery units, cells constitute the fundamental protocol elements underpinning Tor's operations. They are used for circuit creation (`CREATE`, `EXTEND`), stream establishment (`BEGIN`, `CONNECTED`), data exchange (`DATA`), and service connections (`ESTABLISH_INTRO`, `RENDEZVOUS1/2`). Furthermore, embedded identifiers (e.g., *stream_id*, *circuit_id*) and digests provide critical metadata for managing circuits and streams. In this sense, cells are the building blocks enabling Tor's functionality. Beyond their role in traffic delivery, the structural properties and embedded metadata of cells also provide a potential vantage point for understanding Tor's internal operations and, more importantly, for assessing the scope of digital evidence available in investigative contexts.

2.3 Literature review

Prior studies have extensively investigated both the vulnerabilities and improvements of Tor. Murdoch and Danezis [15] demonstrated that traffic analysis based on timing patterns could compromise anonymity. Overlier and Syverson [24] revealed methods for deanonymizing hidden services by targeting introduction points. Biryukov et al. [1] first demonstrated the feasibility of large-scale deanonymization of hidden services, and subsequent works have expanded this line of research. For example, Matic et al. [14] examined how location leaks can compromise hidden services, while more recent measurement studies revealed that AS-level adversaries remain a realistic threat under certain network conditions [7].

Other investigations highlighted practical risks, such as misconfigured onion services in real-world deployments [22] and denial-of-service(DoS) attacks targeting the Tor directory protocol [13]. These studies indicate that despite ongoing improvements, Tor continues to face both technical and operational vulnerabilities.

On the defensive side, Kim et al. [10] proposed leveraging Trusted Execution Environments to reinforce the integrity of Tor's ecosystem. Additionally, several works have explored circuit padding and congestion-control mechanisms to mitigate traffic correlation and enhance performance [23, 14]. These prior efforts provide essential context for analyzing Tor's architecture and security properties in our study. However, no prior work has specifically focused on enabling auditability of Tor for investigative purposes. In particular, achieving this goal without contradicting Tor's fundamental design philosophy poses a significant challenge.

3 Tor Communication Hierarchy

To lay the foundation for investigation on illegal activities in Tor, we first examined the latest Tor implementation to identify protocol-level metadata that may serve as potential evidence. Our review focuses on two complementary aspects: the hierarchical communication structure and the circuit establishment process.

3.1 Communication Layers: Channel–Circuit–Stream

Tor network communication is structured across three hierarchical layers: *channel, circuit, and stream*. The lowest layer, the *channel*, is a TLS connection established between two nodes. Channels are long-lived, reusable, and capable of carrying multiple circuits simultaneously. Such multiplexing reduces the overhead of establishing a new TLS connection for each circuit and conserves network resources.

A *circuit* is an independent logical path constructed over a channel by sequentially connecting relays selected by the client. It constitutes the actual transmission path for Tor traffic and is distinguished by a unique circuit ID within a specific channel (identified by its own channel ID). Onion encryption layers are sequentially applied along the circuit, which follows the 3-hop structure. Once a TLS connection is established between the client and its guard relay, multiple circuits can be multiplexed over this secure channel, each operating independently within the Tor protocol. This design not only improves efficiency in resource utilization but also strengthens anonymity by making it difficult for external observers to associate traffic flows with specific circuits.

At the application layer, multiple *streams* can be multiplexed over a single circuit. A stream represents a TCP session, identified by a stream ID, and allows independent connections to different destinations while sharing the same circuit. For example, opening multiple browser tabs within a single Tor session results in the creation of multiple streams within a single circuit.

In summary, the above hierarchy can be expressed as a 1:N:M relationship: multiple circuits can exist over a single channel, and multiple streams can operate over each circuit. This layered design enables Tor to efficiently manage limited connection resources while ensuring anonymity and performance. Note that a channel and its circuit IDs, along with stream IDs play a crucial role in managing this hierarchy, forming the foundation for routing and session management in Tor.

3.2 Circuit establishment

In Tor, a circuit is established by sequentially connecting relays selected by the client. This process involves the exchange of control cells and Diffie–Hellman-based key negotiations, with independent session keys derived at each hop. The circuit is incrementally expanded until the 3-hop structure is complete, after which TCP sessions with external servers can be initiated. In addition, circuits are managed independently within a single channel and multiplexed by the module `circuitmux_t`. This allows Tor to operate dozens of circuits simultaneously over a limited number of TLS connections[20].

The circuit establishment procedure begins with an OP sending a `CREATE` cell to an initial guard node, initiating a Diffie–Hellman key exchange. Once the session key is established, the guard replies with a `CREATED` cell, thereby completing the first hop. The client then extends the circuit by forwarding an `EXTEND` cell (via the guard) to a middle node. Upon successful negotiation, the middle node returns an `EXTENDED` cell. Following the same process, an exit node is appended to the path through the middle relay. Once the exit node responds positively, the client can transmit a `BEGIN` cell to request a TCP session with an external destination. If the connection is established, the exit replies with a `CONNECTED` cell, after which application data is exchanged through `RELAY_DATA` cells.

Modern versions of Tor employ `CREATE2/CREATED2` and `EXTEND2/EXTENDED2` cells to enhance security. These cells adopt the Ntor handshake, a Diffie–Hellman variant designed for stronger forward secrecy and improved resistance to replay attacks. In this process, extension requests are forwarded to the intended next node rather than being processed locally by the relay forwarding them. Upon successful key negotiation, the next node responds with an `EXTENDED2` cell, completing the secure extension of the circuit.

Hidden Service Circuits. For Hidden Service circuits, the key distinction from general circuits lies in the exchange of specialized control cells, such as the onion service descriptor, `ESTABLISH_INTRO`, `INTRODUCE1/2`, and `RENDEZVOUS1/2`, during the circuit establishment process. An Onion Service begins by sending an `ESTABLISH_INTRO` cell to designate a relay as an Introduction Point, to which the relay responds with an `INTRO_ESTABLISHED` cell. The service then publishes its onion service descriptor, containing information about its IPs. A client retrieves this descriptor to identify an appropriate IP and, through it, delivers the information of a chosen RP to the onion service.

In summary, circuit establishment follows these key control cell exchanges:

1. `CREATE/CREATED`: First-hop handshake and session key setup with the Guard relay.
2. `EXTEND/EXTENDED`: Add subsequent relays to the circuit.
3. `BEGIN/CONNECTED`: Establish a TCP session with the destination via the Exit node.
4. `CREATE2/EXTEND2`: Ntor-based extension providing hop-by-hop key exchange with stronger security.
5. `ESTABLISH_INTRO`, `INTRODUCE`, `RENDEZVOUS` (HS only): Specialized cells for client–Onion Service rendezvous.

4 Circuit Lifecycle Analysis

Understanding Tor's circuit lifecycle is essential not only for protocol analysis but also for identifying where and how circuit- and stream-level metadata may serve as potential evidence in investigative contexts. To this end, we systematically examined the Tor codebase to gain a comprehensive understanding of how circuits are created, managed, and terminated.

Basically, Tor maintains a pool of pre-built circuits for different purposes (e.g., general use or hidden services) [12]. When a user request arrives, Tor attempts to reuse an existing circuit; if none is suitable, a new one is created. Circuit construction begins with the selection of relays from the consensus-approved network view, followed by incremental extension through the guard–middle–exit path. During this process, control cells (e.g., `CREATE/EXTEND`, `BEGIN/CONNECTED`) and cryptographic handshakes establish hop-by-hop session keys. Once a circuit is fully formed, it is registered and made available to carry application traffic.

User data is transmitted as streams multiplexed over established circuits. This process involves three main stages: 1) creating and attaching a stream to a circuit, 2) establishing the external TCP connection through the exit node, and 3) transmitting application payloads as encrypted cells. This lifecycle ensures that Tor can efficiently manage resources while supporting multiple independent communication flows, and also highlights the key points where protocol metadata, including circuit identifiers, stream identifiers, and control cell types, emerges and persists.

Figure 1 illustrates how streams are created, attached, and terminated in Tor. The `connection_new()` function creates a stream object `entry_connection_t` and invokes a function (`circuit_get_best()`)

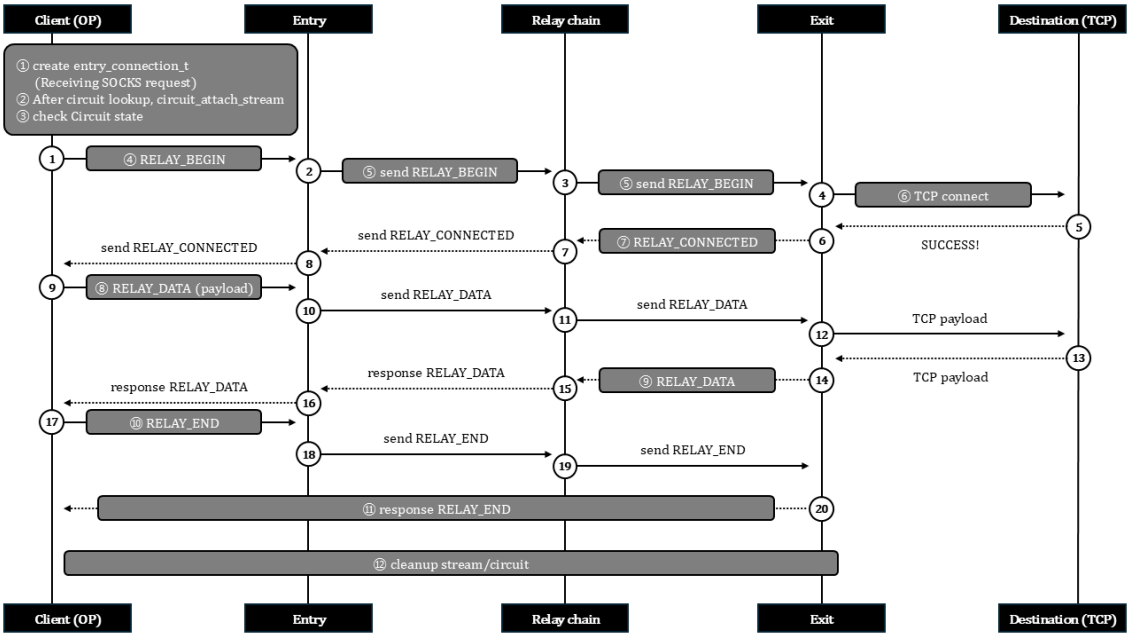

Figure 1: Stream Connection Lifecycle in Tor

to search the `global_circuitlist` for a suitable circuit that satisfies the request conditions (e.g., port, destination, exit policy). If a matching circuit is found, stream attachment proceeds without creating a new circuit, and a unique *stream_id* is assigned. Data transmission then occurs through `RELAY` cells, classified as upstream (client-to-server) or downstream (server-to-client). Each relay inspects the `recognized` field of the received cell: intermediate relays decrypt a single encryption layer and forward the payload, while the final destination relay (Exit or OP) extracts and processes the `RELAY_DATA` payload.

Stream termination may be initiated either by the OP or by the exit (e.g., upon receiving a FIN from the remote server or an I/O error). In both cases, a `RELAY_END` cell is generated, embedding the termination cause. When the `RELAY_COMMAND_END` command is identified, `connection_edge_end_close()` finalizes the teardown by closing the stream state and releasing associated buffers and sockets. A circuit terminates when stream transmission is completed, the idle timer expires, or when errors, such as cell decryption failures occur. Termination reasons are recorded in the circuit's `marked_for_close_reason` field. Once a termination event is triggered, the circuit is placed in the `circuits_pending_close` queue for cleanup. This queue is periodically traversed, during which the circuit is removed from both `global_origin_circuitlist` and `global_circuitlist`, its connections are closed via a `DESTROY` cell, and memory resources are released.

In summary, circuits in Tor are represented by three related structures: `origin_circuit_t` (for circuits created by the client), `or_circuit_t` (for those maintained by relays), and their common parent `circuit_t`. These structures store essential metadata, such as purpose, state, hop list, and termination reason. Streams are represented by `connection_t`, which maintains information on the attached circuit, flow control windows, stream identifier, and end reason. At the lowest level, all communication is encapsulated in `cell_t`, which contains the circuit identifier, command type, and payload, with specialized sub-structures parsed according to the command. These representations highlight how Tor maintains circuits, streams, and cells in a consistent manner, and how key metadata emerges as part of this lifecycle. Such metadata provides a basis for exploring what can be observed or leveraged in investigative contexts, raising key questions about the balance between observability and Tor's fundamental design principles.

5 Classification of Investigative Metadata

Previous studies on Tor have largely concentrated on deanonymization research, focusing on traffic-analysis techniques such as website fingerprinting and related methods. While valuable, these efforts overlooked the systematic analysis of metadata generated during Tor's operation. As a result, there are currently no viable approaches to distinguish user behavior on the client side, identify potential risks within the network, or leverage such information for investigative purposes.

To address this gap, this study proposes a structured classification of investigative metadata derived from the architecture and operational flow of Tor's circuits, streams, and cells. By examining how identifiers, control signals, and protocol artifacts emerge and persist across these layers, we aim to highlight metadata elements that can support investigative visibility and explore how such visibility can be reconciled with Tor's core design principles.

5.1 Investigation Scenario

We introduce investigation scenarios that evaluate how metadata visibility can be leveraged to identify HS operators and clients. By examining both perspectives, we aim to verify the investigative usefulness of Tor metadata and highlight how different vantage points can contribute to constraining operators or reducing client-driven demand.

Tor hidden services enable website operators to provide content without revealing their IP addresses, while also preserving client anonymity. These features have made hidden services the backbone of most crimes committed on the Dark Web. In this context, operators function not merely as users but as infrastructure providers. Even if client access is reduced, services remain active unless the operators themselves are identified, making it difficult to eliminate the root source of criminal activity.

At the same time, clients play a critical role as demand drivers. By repeatedly accessing illegal services (e.g., marketplaces for drugs or ransomware) they sustain the ecosystem and enable its continuous expansion. Moreover, clients may engage in resale or secondary misuse, amplifying the potential for further crimes. Tracking client activity is therefore also crucial to preventing secondary harm. Figure 2 demonstrates the Tor components involved in these scenarios, illustrating the relationship between clients, operators, and the relevant metadata sources.

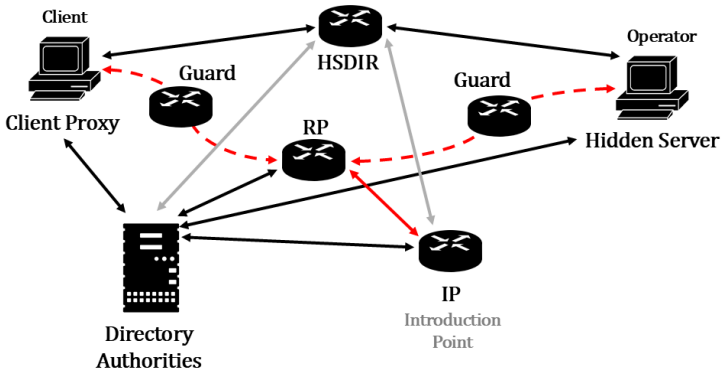

Figure 2: Proposed Scenario of client and operator

5.2 Observation Scope Analysis

Tor is designed to strictly separate the roles and information accessible to clients and relays in order to preserve anonymity. While this separation strengthens security, it simultaneously restricts visibility

from an investigative or auditing perspective. In particular, many metadata fields are exposed only to the OP or only to onion routers, which means that the type and scope of inferences vary depending on the vantage point. Therefore, an analysis of the metadata observable at the OP and OR is required to delineate the scope of information available for investigative use in the above scenario.

As the entity handling the user's direct request, the OP has access to user-centric metadata, such as the target address, request time, and the circuit's intended purpose. These values are primarily maintained in structures like `origin_circuit_t` and `socks_request_t`, which capture the overall path and service requested by the client. In contrast, relays do not observe the full circuit context. Instead, they manage transmission-oriented information through structures, such as `or_circuit_t` and `channel_t`, which include connection identifiers, the number of cells transmitted, and circuit IDs linked to preceding nodes. Although OP-side and OR-side data structures share the common parent `circuit_t`, identical fields can serve different roles depending on context. For example, the `purpose` field in `origin_circuit_t` encodes explicit circuit intent (e.g., `C_GENERAL` or `C_HSDIR_GET`), whereas in `or_circuit_t` it functions merely as a classification flag. Table 2 summarizes these differences by contrasting metadata observable at the OP, OR, and common levels.

Type	Information	Structure & Field	Investigative Relevance

OP side (client)	Target Address
	Request Time
	Circuit Purpose
	Exit Selection Result
		`socks_request_t.address`
		`origin_circuit_t.purpose`
		`cpath_build_state_t.chosen_exit`
			Can reveal what service was requested, when, and through which chosen exit path; useful for linking client behavior with specific hidden services.

OR side (relay)	Number of Cells Transmitted
	Circuit ID of Previous Connected Node
	Connection Frequency
		`or_circuit_t.circ_id`
		`channel_t`
		`n_received_cellsetc`
			Provides traffic volume and relay-level activity; can help identify suspicious patterns such as bot activity or anomalous relay use.

Common	Number of Streams
	Circuit Length
	Connection Status
		`circuit_t.n_streams`
		`isolation_flags`
		`state`
			Visible at multiple vantage points; can indicate overall circuit utilization, but with limited forensic value when viewed in isolation.

Table 2: OP-Side vs OR-Side Metadata Visibility and their Investigative Relevance

6 Metadata Identifiability via Visibility Scopes

In this section, we discuss the types of metadata and their potential utility as evidence based on the scope of visibility. As described in section 5, the internal metadata accessible at OP and OR nodes differs, which necessitates evaluating the evidential value of metadata for each scenario according to its vantage point. The metadata under consideration refers to data internally maintained by Tor during operation and not ordinarily recorded in external logs. For the purpose of analysis, we assume that

such metadata is logged and stored locally, and that it becomes accessible when the corresponding node is seized. Figure 3 incrementally demonstrates how inferences change based on the observation of visibility scope.

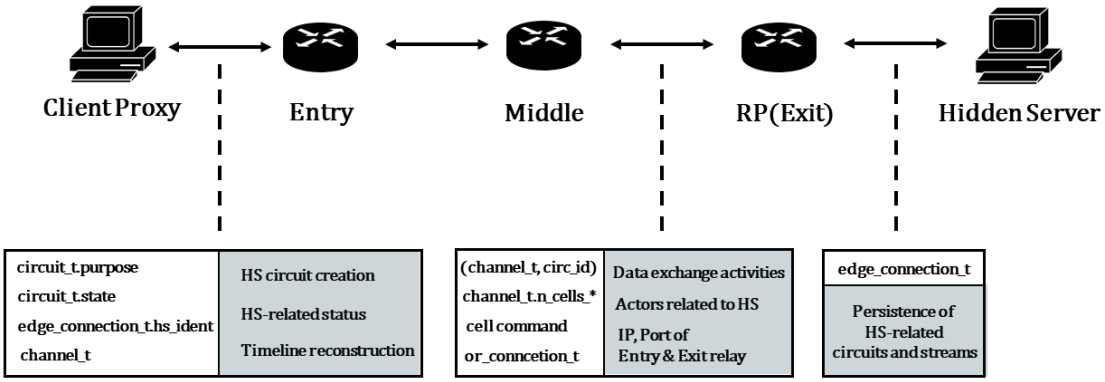

Figure 3: Identifiable Information by visibility scope

Host PC First, because a host PC connected to the Tor network operates on the OP side, it can expose the richest set of metadata. Representative examples include the internal states maintained in circuit and stream structures, circuit identifiers and state values, and data related to Tor cells. Such information directly demonstrates the PC owner's connection to Tor. In particular, the circuit purpose field is crucial, as it indicates the intended use of a circuit and allows inferences about which services the user accessed.

For hidden service operators, the process involves selecting a relay to act as an IP, establishing a circuit with that relay, and registering the service information in the HSDir. Clients connecting to a hidden service randomly select a RP and transmit this information to the operator through the IP. Consequently, when a host PC is seized, if the value of `circuit_t.purpose` corresponds to any of {`CIRCUIT_PURPOSE_S_ESTABLISH_INTRO`, `CIRCUIT_PURPOSE_S_INTRO`, `CIRCUIT_PURPOSE_S_CONNECT_REND`, `CIRCUIT_PURPOSE_S_REND_JOINED`}, the node can be reasonably suspected of functioning as a hidden service operator.

In addition, if the `circuit_t.purpose` value is set to {`CIRCUIT_PURPOSE_C_INTRODUCING`, `CIRCUIT_PURPOSE_C_INTRODUCE_ACK_WAIT`, `CIRCUIT_PURPOSE_C_INTRODUCE_ACKED`, `CIRCUIT_PURPOSE_C_ESTABLISH_REND`, `CIRCUIT_PURPOSE_C_REND_READY`, `CIRCUIT_PURPOSE_C_REND_READY_INTRO_ACKED`, `CIRCUIT_PURPOSE_C_REND_JOINED`}, the node can be reasonably suspected of functioning as a hidden service client. Both operators and clients establish circuits to a RP, but operators additionally maintain circuits to Introduction Points, whereas clients do not. To further demonstrate intent, the `circuit_t.state` field can also be leveraged. If a circuit with a hidden-service-related purpose has its state set to `OPEN`, it signifies that the circuit has been fully established, thereby confirming the intention to operate or access a hidden service. This combination of purpose and state values provides strong evidence for distinguishing general users from those involved in hidden service operation or access.

The `edge_connection_t` structure, which manages connections between external endpoints and the Tor network, contains the `hs_ident` field. For a hidden service operator, this field links a stream to its corresponding HS circuit, enabling the operator to determine through which HS circuit a request arrived. For a client, it links the SOCKS connection to the targeted hidden service, thereby indicating which .onion address the request was directed to. Thus, the presence of `hs_ident` on a seized host PC suggests involvement in hidden service activity.

Within the `channel_t` structure, which represents the TLS connection between the OP and each relay node, the IP address and port of the connected entry relay can be identified. This information

makes it possible to verify when and with which relay a direct connection was established. In addition, timestamp fields (e.g., `timestamp_created`, `timestamp_active`, and `timestamp_last_had_circuits`) can be used for timeline reconstruction. Together, these values serve as concrete metadata for correlating user activity with specific relays over time, strengthening their evidential significance.

Host PC with Entry, Middle Relay At this stage, if visibility extends to the entry relay of the circuit identified on the host PC, it becomes possible to verify whether the circuit was actually in use. Specifically, if the pair `channel_t` and `circ_id` previously flagged as belonging to a hidden service circuit is also observed at the entry relay, and the corresponding `channel_t.n_cells_*` counters increase, this indicates that data exchange occurred at the network layer through the established circuit. Moreover, when `channel_t.n_cells_recved/xmitted` values significantly increase on the same channel, and the associated cell command is `RELAY`, this provides strong evidence that the hidden service circuit was actively transmitting traffic. Such findings strongly suggest that the actor is genuinely associated with the hidden service. Note that it must be verifiable that a specific `circID/streamID` was active at a given time. Therefore, the analysis must also confirm that the corresponding stream was transmitted from the exit relay to the actual destination server, ensuring temporal and contextual consistency.

The middle relay is an intermediate node in the Tor network that forwards traffic between the entry relay and the exit relay, maintaining separate `or_connection_t` instances with each. The `canonical_orport` field of `or_connection_t` reveals the IP address and port of the connected relays, information that could later assist investigative agencies in identifying and securing the corresponding entry and exit nodes.

Host PC with Entry, Middle, Exit Relay Assuming the visibility scope extends to the host PC, the entry relay, and the middle relay, and all previously described information has been identified, the exit Relay can further enhance the reliability of the acquired evidence. In the case of exit relay, it can determine the final destination of a connection using `edge_connection_t.address` and `edge_connection_t.port`. By cross-verifying the host PC's transmission and reception timestamps with the exit Relay's traffic records (e.g., transmitted bytes, packet counts), it becomes possible to confirm that the hidden service–related circuit and stream persisted all the way through to the exit relay. However, hidden service traffic differs from ordinary traffic: an HS client communicates directly with the HS server through a rendezvous circuit, and the operator terminates traffic at the RP. As a result, the destination information available at the exit Relay does not reveal the identity of either the hidden service operator or the client.

For vanilla Tor, metadata is maintained only in runtime memory and is not persistently logged. The data used to preserve anonymity exists as in-memory objects rather than being stored on disk. As a result, even when nodes are seized, investigators can typically verify only limited information available in existing logs, such as service start and stop times or process status. However, this approach is incomplete, since users can delete or restrict logs through the log-level settings in the `torrc` file.

As we demonstrated in this section, Tor metadata demonstrates significant potential as an evidentiary source if it can be logged and retained, which confirms the necessity of a more systematic storage mechanism. More importantly, future work must carefully balance investigative usefulness with the preservation of user anonymity, ensuring that any proposed solution can support legitimate investigations without undermining Tor's fundamental design principles.

7 Conclusion

This paper analyzed the circuit and stream structures within the Tor network and demonstrated the potential for utilizing the metadata employed in these processes. While prior research has primarily focused on traffic analysis and deanonymization techniques aimed at undermining Tor's anonymity, we examined the investigative utility of evidence collection based on visibility scope by structuring Tor's operational processes in terms of the metadata they generate. This demonstrates how the scope of identifiable information varies depending on how metadata is combined, contributing to understanding the balance between investigation and anonymity. Without complex traffic decryption or protocol modifications, it evaluates the minimal metadata that Tor inherently generates. We determined the

significance of metadata such as circuit purpose, circuit ID, stream ID, and cell exchange patterns based on whether they are observable at the host PC, entry node, middle node, or exit node level. Depending on the level at which identifiable information is revealed, it could potentially compromise the anonymity of other users, raising issues that may conflict with the direction pursued by the Tor community. To prevent such problems, it is necessary to clearly define data that reveals anonymity only when combined with client-side information or to design new data items, if required. This approach would preserve other users' anonymity while ensuring auditability. Future work will address running the Tor community's trusted nodes—the directory server and directory authority—within a trusted environment, such as Intel SGX, while simultaneously logging and preserving metadata [10, 11]. The goal is to design a system that can determine the minimum amount of metadata necessary for investigations without compromising other users' anonymity and that can securely store and manage it.

Acknowledgments

This work is partly supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. RS-2024-00351898 and No. RS-2025-02263915), the MOTIE under Training Industrial Security Specialist for High-Tech Industry (RS-2024-00415520) supervised by the Korea Institute for Advancement of Technology (KIAT), and the MSIT under the ICAN (ICT Challenge and Advanced Network of HRD) program (No. IITP-2022-RS-2022-00156310) supervised by the Institute of Information & Communication Technology Planning & Evaluation (IITP).

References

[1] Alex Biryukov, Ivan Pustogarov, and Ralf-Philipp Weinmann. Trawling for Tor hidden services: Detection, measurement, deanonymization. In *2013 IEEE Symposium on Security and Privacy (S&P)*, pages 80–94. IEEE, 2013. Available at https://ieeexplore.ieee.org/document/6547103.

[2] John Borland. For tor, publicity a mixed blessing. [online], December 2013. Available at https://www.wired.com/2013/12/tor-publicity-mixed-blessing.

[3] Nicolas Christin. Traveling the silk road: A measurement analysis of a large anonymous online marketplace. In *Proceedings of the 22nd International Conference on World Wide Web (WWW)*, pages 213–224. ACM, 2013. Available at https://dl.acm.org/doi/10.1145/2488388.2488408.

[4] Roger Dingledine, Nick Mathewson, and Paul Syverson. The onion router (tor) research home page. [online], 2000–2025. Available at https://www.onion-router.net/, last accessed Sep. 2025.

[5] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The Second-Generation Onion Router. In *13th USENIX Security Symposium (USENIX Security 04)*, San Diego, CA, August 2004. USENIX Association.

[6] Europol. Internet organised crime threat assessment (iocta) 2023. [online], 2023. Available at https://www.europol.europa.eu/cms/sites/default/files/documents/IOCTA%202023%20-%20EN.pdf.

[7] Gabriel Karl Gegenhuber, Markus Maier, Florian Holzbauer, Wilfried Mayer, Georg Merzdovnik, Edgar Weippl, and Johanna Ullrich. An extended view on measuring Tor as-level adversaries, 2023. Available at https://doi.org/10.1016/j.cose.2023.103302.

[8] Andy Greenberg. Tor says fbi paid carnegie mellon $1m to help unmask users. [online], November 2015. Available at https://www.wired.com/2015/11/tor-says-feds-paid-carnegie-mellon-1m-to-help-unmask-users/.

[9] Rob Jansen, Paul Syverson, Nicholas Hopper, and Roger Dingledine. Shadow: Running Tor in a box for accurate Tor experimentation. In *19th Annual ISOC Network and Distributed System Security Symposium (NDSS 2012)*, 2012.

Available at https://www.ndss-symposium.org/ndss2012/ndss-2012-programme/shadow-running-tor-box-accurate-and-efficient-experimentation/.

[10] Seongmin Kim, Juhyeng Han, Jaehyeong Ha, Taesoo Kim, and Dongsu Han. Enhancing security and privacy of Tor's ecosystem by using trusted execution environments. In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI '17)*, pages 145–161, Boston, MA, March 2017. USENIX Association.

[11] Seongmin Kim, Juhyeng Han, Jaehyeong Ha, Taesoo Kim, and Dongsu Han. Sgx-tor: A secure and practical tor anonymity network with sgx enclaves, October 2018. Available at https://doi.org/10.1109/TNET.2018.2868054.

[12] Zachary Liebl. zach-liebl-tor: Tor project experimental code repository. https://gitlab.torproject.org/ZachLiebl/zach-liebl-tor. GitLab repository, last accessed Sep. 2025.

[13] Zhongtang Luo, Jianting Zhang, Akshat Neerati, and Aniket Kate. Five minutes of ddos brings down tor: Attacks on the tor directory protocol and mitigations. [online], September 2025. Available at https://arxiv.org/abs/2509.10755.

[14] Srdjan Matic, Platon Kotzias, and Juan Caballero. CARONTE: Detecting location leaks for deanonymizing Tor hidden services. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security (CCS '15)*, pages 1455–1466. ACM, 2015. Available at https://dl.acm.org/doi/10.1145/2810103.2813667.

[15] Steven J. Murdoch and George Danezis. Low-cost traffic analysis of Tor. In *2005 IEEE Symposium on Security and Privacy (S&P '05)*, pages 183–195. IEEE, 2005. Available at https://ieeexplore.ieee.org/document/1431230.

[16] Yumingzhi Pan, Zhen Ling, Yue Zhang, Hongze Wang, Guangchi Liu, Junzhou Luo, and Xinwen Fu. TORCHLIGHT: Shedding LIGHT on real-world attacks on cloudless IoT devices concealed within the Tor network. [online], January 2025. Available at https://arxiv.org/abs/2501.16784.

[17] The Freedom of the Press Foundation. Securedrop. https://securedrop.org/, last viewed Jul. 09, 2025, 2013–2025.

[18] The Tor Project. Tor Project Official Website. https://www.torproject.org/, last viewed Sep. 2025, 2002–2025.

[19] The Tor Project. Tor proposal 224: Next-generation hidden services. [online], 2013. Available at https://spec.torproject.org/proposals/224-rend-spec-ng.html.

[20] The Tor Project. Tor design: Circuit and stream multiplexing. [online], 2025. Available at https://spec.torproject.org/glossary.html?highlight=multiplex%20stream%20circuit#general-network-definitions.

[21] The Tor Project. Tor protocol specification. [online], 2025. Available at https://spec.torproject.org/.

[22] Pascal Tippe and Adrian Tippe. Onion services in the wild: A study of deanonymization attacks. In *Proceedings on Privacy Enhancing Technologies (PoPETs 2024)*, volume 2024, pages 291–310, October 2024.

[23] Philipp Winter and Stefan Lindskog. How the Great Firewall of China is blocking Tor. In *2nd USENIX Workshop on Free and Open Communications on the Internet (FOCI 12)*, Bellevue, WA, 2012. USENIX Association.

[24] L. Øverlier and Paul F. Syverson. Locating hidden servers. In *2006 IEEE Symposium on Security and Privacy (S&P '06)*, pages 100–114, Berkeley / Oakland, CA, USA, 2006. IEEE.